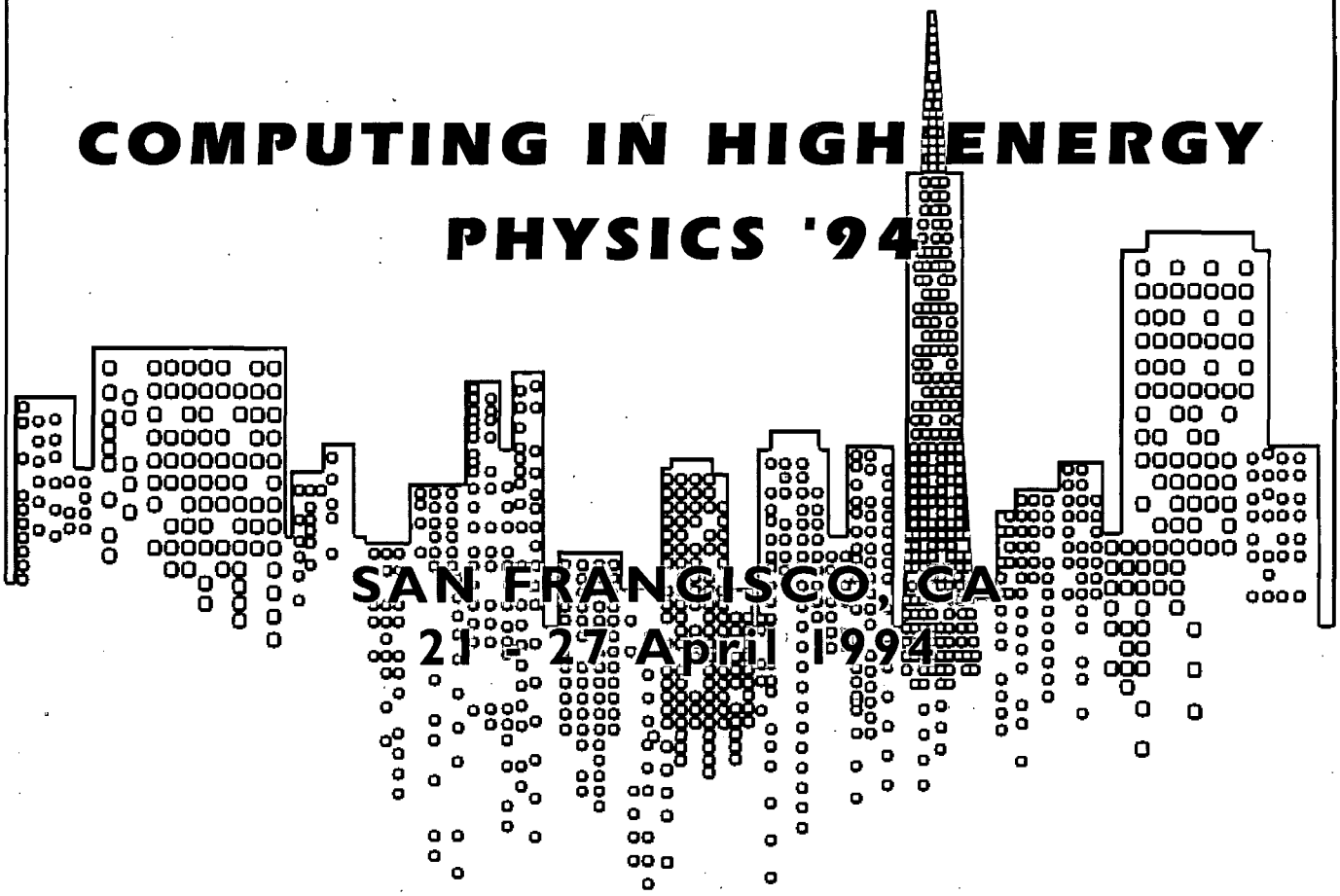
LBL-35822
CONF-940492
UC-405

COMPUTING IN HIGH ENERGY PHYSICS '94



SAN FRANCISCO, CA
21 - 27 April 1994

CHEP 94

Organized and Hosted by Lawrence Berkeley Laboratory
University of California • Berkeley, California 94720

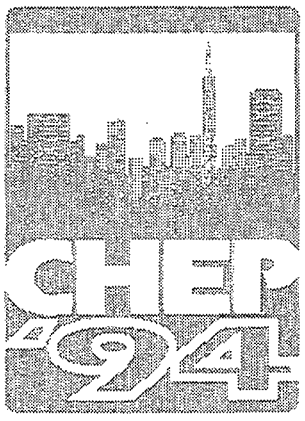
Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098

DISCLAIMER

PROCEEDINGS OF THE CONFERENCE ON COMPUTING IN HIGH ENERGY PHYSICS '94

SAN FRANCISCO, CA
21 - 27 April 1994

Proceedings Editor: S.C. Loken



Organized and Hosted by
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

This work was supported by the Director, Office of Energy Research, Office of Scientific Computing, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098

Forward

CHEP '94 continues the tradition of conferences that bring together physicists and computer scientists to discuss all aspects of computing for High Energy Physics. The 1994 meeting was organized by the Lawrence Berkeley Laboratory and was held in the Hyatt Regency Hotel in San Francisco, California. Funding was provided by the United States Department of Energy, Office of Scientific Computing.

In planning CHEP '94, the organizers followed a new format: three days of parallel sessions, a one-day break, then three days of plenary sessions. The eight parallel sessions were divided into four tracks, covering a broad range of subjects. Following the one-day break, there were three days of plenary sessions, which included invited lectures as well as talks providing summaries of each of the parallel session areas. A poster session also gave an opportunity for informal discussion.

This was the first conference in the CHEP series where packet video conferencing was used to transmit all plenary talks. The sessions were carried live on the international Multicast Backbone, MBone. During the meeting, approximately 400 people used the MBone broadcast to hear and see the talks at CHEP. In addition, participants could view abstracts from the conference on the World Wide Web via MOSAIC and access conference information via electronic mail and MOSAIC.

These proceedings contain all of the papers that were submitted to the conference. As is frequently the case, not all presenters completed a written version of their talk. To ensure early publication, however, we have collected all available papers into this volume. The plenary papers are followed by the parallel session papers and the poster papers.

The organizers would like to express their appreciation to the many people who contributed to the success of the meeting. Mollie Field, Del Thomas, and Dot Akins managed all of the conference arrangements. Cynthia Coolahan pulled together the plans and schedule for the parallel sessions. Julie McCullough did the technical editing of these proceedings. In addition, the organizers would like to thank the speakers and the session convenors for their help in putting together a productive meeting.

Stewart C. Loken
Chairman

Scientific Advisory Committee

Drew Baden—Maryland, USA
Rudy Bock—CERN
Joel Butler—Fermilab, USA
John Cavallini—DOE, Washington, USA
He-Shen Chen—IHEP, Beijing, China
Larry Cornell—SSCLab, USA
Trevor Daniels—RAL, UK
Manuel Delfino—Barcelona, Spain
Atul Gurtu—Tata Institute, India
Philippe Heusse—LAL, Orsay, France
Paul Kunz—SLAC, USA
Juergen May—DESY, Germany
Richard Mount—Caltech, USA
Tom Nash—Fermilab, USA
Michael Ogg—Carleton, Ottawa, Canada
Risto Orava—Helsinki, Finland
Vadim Petukhov—IHEP, Serpukhov, Russia
Alberto Santoro—CBPF, Rio de Janeiro, Brazil
Terry Schalk—UC Santa Cruz, USA
Rick Stevens—ANL, USA
Leonid Tkatchev—JINR, Dubna, Russia
Enzo Valente—INFN, Rome, Italy
Jean-Pierre Vialle—LAPP, Annecy, France
Yoshiyuki Watase—KEK, Tsukuba, Japan
Chip Watson—CEBAF, USA
David Williams—CERN

Local Organizers, Lawrence Berkeley Laboratory

Stewart Loken, Chairman
Gerald Abrams
Matthew Bloomer
Al Clark
Cynthia Coolahan
Chris Day
Craig Eades
Harvard Holmes
Charles McParland
Doug Olsen
David Quarrie
Tom Trippe

Parallel Session Convenors

Chip Watson, CEBAF	Session 1: Triggering, Data Acquisition, Online and Control Systems
Les Robertson, CERN	Session 2: Hardware Architectures
Les Cottrell, SLAC	Session 3: Networks and Interconnections
Yohei Morita, KEK:	Session 4: Data Handling and Storage Systems
Vicky White, Fermilab	Session 5: Software Methodologies, Languages and Tools
Larry Cornell, SSCLab	Session 6: User Interfaces and Visualization
Simone Giani, CERN	Session 7: Computation
Bill Lidinski, Fermilab	Session 8: Information Systems and Multimedia

CONTENTS

Session 8: Information Systems and Multimedia

Posters

Plenary Talks

FROM SUBSISTANCE FARMING TO AGRO-BUSINESS *

Manuel Delfino
Supercomputer Computations Research Institute
Florida State University
Tallahassee, FL 32306-4052 USA
(on leave from Universitat Autonoma de Barcelona, SPAIN)

Abstract

A review of the parallel session on Hardware Architectures is presented. Legacy systems from the "subsistance farming" era, where small groups from experiments built or assembled their own loosely-coupled parallel systems, must still be maintained, but the major trend is towards a re-birth of the Computer Center, using "agro-business" technology, the large scale use of loosely-coupled parallel systems using a high performance network to communicate with storage subsystems.

INTRODUCTION

A broad range of systems were presented, ranging from an upgrade of the reconstruction farm at the OPAL experiment at LEP to an essentially new computer center assembled at DESY for the HERA experiments. The dominant theme at the previous CHEP conference was workstation farms appended to mainframes, such as CSF at CERN and BASTA at IN2P3. Users logged into the mainframe would submit jobs to these appended farms, which would provide the CPU to operate on data imported from disks or tapes belonging to the mainframe. At that time, Fermilab already had independent farms, eliminating the need for mainframes by using replicated storage based on Exabyte tapes. As we will see, at this conference the workstation farms are no longer appendages to mainframes, the mainframes in fact have essentially dissappeared and storage is viewed as attached to the network, equally accessible to all workstations.

LEGACY SYSTEMS

The term legacy system is used in industry to describe equipment which is obsolete or no longer fits into accepted hardware or software architecture models, but must nevertheless be kept running. Most of the LEP reconstruction farms can be considered legacy systems. Increasing maintenance costs dictate that at least parts of these survivors from the "subsistance farming" era must be upgraded periodically. The OPAL paper describes nicely how this group decided to make

*This work was partly supported by the US Department of Energy under contracts DE-FG05-92ER40742 and DE-FG05-85ER250000 and by CICYT, Spain

a rather large investment of human resources to rewrite their control software using standards such as POSIX, TCP/IP and X-windows. This should result in hardware independent software, saving much work in the many future years that this subsystem will be operational. A component that unfortunately does not exist yet, but which clearly would have been very useful to them, is a standardized distributed queueing system. Other remarkable aspects of this work are the emphasis on documentation and the use of experienced programmers, both rare in HEP.

THE SHRINKING MAINFRAME

Although an appropriate mainframe would most likely still be the best solution to computing hardware for HEP, existing mainframes are being shutdown or replaced by smaller versions (down-sizing). The main reason is that the *existing* mainframes are not *appropriate*, having too little CPU and large costs. Their dissapearance has not been more rapid due to two very disparate facts. First, they usually host expensive tape devices, such as robots with proprietary interfaces, which in turn hold all the physics data. Second, although many people now use personal workstations, there are still many who have character terminals and depend on mainframes for utilities such as mail, news, text editing and program development. This leads to a "co-habitation" between systems, nicely explained in the paper from IN2P3. They also emphasize standards such as POSIX and suffer the lack of a standardized queueing system. Their reflections, and those in papers of other parallel sessions, on the investments needed in providing workgroup servers and intelligent desktop devices to all users, as well as the need for user education in these new systems, should be carefully considered.

UNIX SYSTEMS

Building on the foundation of work done on the CERN Cray and workstation clusters at various sites, UNIX systems have now become prevalent in HEP offline batch processing. Their use for interactive analysis is also growing, at a slower pace mostly limited by the financial constraints of replacing existing workstations and terminals on people's desks. The use of traditional UNIX in other applications, such as data acquisition or software design and development, is still open to question.

Papers from several groups at DESY and Fermilab describe what I have humorously termed "agro-business", or more appropriately Centrally Operated Distributed Systems. The amount of CPU power that has been amassed in these very plain, rather inexpensive, rack-mounted systems is very impressive, and applications with large CPU to I/O ratios, such as simulations, are very well served. The fact that some of these systems, particularly the SGI Challenge, are capable of Symmetric Multi-Processing has not been thoroughly exploited so far, except to ease the balancing of load in the absence of a sophisticated queueing system. Perhaps this will change in the future as various SMP systems with larger number of processors become available. What is important is that these machines are very powerful and affordable.

The bottleneck mentioned everywhere is in I/O to these CPUs. Disk subsystems with sizes in the hundreds of GB provide

a cache for the dozens of TB of data held on tape. These tapes are 8 mm Exabytes at Fermilab and proprietary Ampex high capacity cartridges at DESY. Sustained integrated data transfer rates approaching 100 MB/s are measured, indicating these distributed systems can in fact exceed the I/O capabilities of traditional mainframes. The weak point is that, although everyone would like to view the mass storage as "network attached", the components available today do not implement this model in a mature and robust way. Network attached storage is simulated by attaching tape and disk devices to workstations designated as I/O servers, which in turn provide access to other CPUs via high-speed networks such as switched FDDI, HIPPI and FibreChannel. Whether the future will be a maturing of this scheme, or the exploitation of SMP systems with hundreds of CPUs on a bus that would replace the high-speed network, is still open to question. The latter option would provide an interesting outcome, where the replacement of mainframes by workstation farms ends in the collapse of the workstation farm into a single system, a sort of new super-mainframe, with the consequence of a rebirth of the traditional computer center.

Another subject that is being explored is that of parallel I/O, as illustrated in the paper from CERN on a distributed database application, running on a Meiko CS2 with 32 processors, each accessing a separate disk containing a portion of the database.

SOME OPEN QUESTIONS

These presentations bring out a series of issues that will have to be dealt with soon. First and foremost is the need to train and retrain the users (i.e. physicists) on the effective use of these new architectures and operating systems. The human interfaces are not particularly easy to use, and many are finding out that building a robust, friendly and intuitive interface is very difficult, and perhaps impossible in the world of amateur physicist programmers. Many user-friendly components are still missing, such as location independent data set addressing by users, based on a proper Hierarchichal Mass Storage system. Perhaps the answer is that desktop systems will evolve in the direction of using Personal Computers with off-the-shelf easy to use software packages, with only a few experts running large productions on the farms or pseudo-mainframes.

The enormous data rates, both present and future, bring up the question of whether HEP computing will have to become more centralized, with even more massive computing installations at the accelerator laboratories and an end to the export of data to home institutions. If this is the case, then the emphasis should shift from having an appropriate local computing facility to having a powerful wide area network, supporting remote X-window sessions and client-server processing between desktop devices at home and computer centers at the labs.

Once again, various types of tape media are being used at various installations. Even if export of data from the accelerator labs is eliminated, the reverse flow of simulated data produced at home institutions will necesitate exchangeable tape media, so decisions by any given institution must be taken carefully.

The paper presented by Ogg and collaborators explores an interesting option

for the future, which hinges on three points, two technological ones, one of which is discussed in the paper, and a sociological one which is only implied. The general idea is to build a geographically dispersed meta-computer, scheduling jobs to the most appropriate facility rather than moving data around. This would eliminate the reverse flow mentioned in the previous paragraph. Instead, jobs from any user that need that particular simulation dataset would be routed to run at the production facility. The first ingredient in this model is a substantial increase in wide-area network capacity to support the flow of control structures, intermediate results and outputs between the components of this meta-computer. The second ingredient would be software technology, along the lines of POSIX but much more sophisticated, which would truly implement hardware independent programs, including access to data, graphics output, etc. while maintaining a secure environment. Lastly, and perhaps most difficult of all, is the opening of computing facilities in various countries to all users of an international collaboration, for example. Although there are many difficulties in this scheme, some solutions can be seen around the corner and it is very worthwhile to explore it.

CLOSING REMARKS

Even though complex forms of collaboration, such as a worldwide HEP meta-computer, are difficult and probably far in the future, we can expect simpler collaborations amongst the various laboratories and universities in HEP to yield mature support for these architectures over the next few years. The most important problem to be solved is to isolate software seen by users, both system and application, from the hardware it is running on, while maintaining high efficiency.

The rationalization of the various flavors of Distributed Queueing Systems, replacing independent development paths by solid support of a mature version would be very nice indeed. The development of the so-called Common UNIX Environment, which would provide users with a hardware independent system view of UNIX workstations, would allow users to concentrate on their physics work rather than on understanding the miriad small differences between UNIX implementations. Training of users in the use of standards such as POSIX, TCP/IP and modern programming techniques, are long term investments which will eventually yield applications which are more reliable and easier to maintain.

ANALYZING TERABYTES OF DATA AT FERMILAB *

Stephen Wolbers
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia IL 60510 USA

Abstract

Computing demands of High Energy Physics are increasing steadily due to the demands of larger datasets and increasingly sophisticated detector systems and analysis techniques. Fermilab has been meeting these demands by the use of many different computing techniques. Most of these techniques attempt to utilize the most cost-effective computing resources while providing effective solutions to the problems that are created by multi-Terabyte data samples and large collaborations. New strategies are being developed to allow improved access to the data.

INTRODUCTION

During the past 5-10 years at Fermilab the typical experiment has written increasing amounts of raw data to tape in each data run. In addition, the events have become more complicated due to increased energy and intensity of the incoming beams and the improvements that have been made to the detector systems. The increasing availability of computing power has also allowed experiments to become more sophisticated in their analysis programs. The final data sets used for physics analyses have also increased dramatically due to the larger data samples and due to improved triggers and reconstruction algorithms that allow larger and better final event samples to be kept for analysis.

*This work is supported by the U.S. Department of Energy under Contract No. DE-AC02-76CH03000.

All of these trends have forced Fermilab to focus on providing improved and cost-effective computing to handle the massive amounts of data that are being generated. Different computing solutions have been used for event reconstruction, splitting and filtering, and physics analysis. Though some notable successes have been achieved there are still improvements to be made to continue to keep up with the demands of the experiments.

INCREASING COMPUTING NEEDS

Experiments at Fermilab have been writing and analyzing large and increasing amounts of data. This is certainly not unique to Fermilab and reflects many trends in scientific computing. During the 1990-91 accelerator run the "typical"

experiment wrote approximately 2 TB of data to tape, with one experiment writing over 40 TB. During the 1992-93 run CDF wrote 2 TB of data and D0 wrote 8 TB of data. It is expected that CDF will write 6-8 TB and D0 18-24 TB of data during the 1994 run. Larger datasets are anticipated from future data runs. There is no reason to expect the trend of writing increasing amounts of data to tape will not continue.

CPU Needs

The CPU required to handle the event reconstruction (and other processing and analysis steps) are constantly increasing. The largest single requirement is event reconstruction. There are many reasons for the increase including; the increasing number of events, the increased complexity of each event, and the increased sophistication of the reconstruction algorithms. More effort is being used to maximize the physics potential of the data sets and less to saving CPU time due to the increasing availability of more CPU power. Capabilities exist to reconstruct events more than once in many cases. The availability of more CPU power has certainly had an influence on the amount of CPU time used for event reconstruction. The trend toward increased CPU usage for all parts of offline processing will continue.

Final Dataset Sizes

Each experiment has to reconstruct their dataset and split it into samples which can be quickly analyzed to produce physics results. The final samples are growing extremely rapidly. Experimental collaborations are becoming larger, making access to the ever-growing datasets a more and more crucial problem for fast and effective data analysis. The bottleneck of slow serial access to large datasets has the effect of limiting physics analysis, especially as the datasets continue to grow.

UNIX FARMS

The UNIX Farms at Fermilab are used for the reconstruction of raw data [1]. The CPU-to-I/O ratio is sufficiently high that loosely-coupled computing solves this CPU-intensive task. The farms can be characterized by describing the hardware, software, and the running experience of the last few years. More details can be found in another contribution to this conference. [2]

Hardware

The UNIX Farms consist of extremely cost-effective UNIX workstations connected via ethernet and divided into worker and I/O nodes. The majority of nodes are worker nodes and consist of rack-mounted UNIX workstations (minus the keyboards and screens) containing the minimum memory needed to avoid swapping, and a small amount of local disk for the operating system. The I/O nodes are UNIX servers or workstations and are connected to local SCSI disk and 8mm tapedrives. The UNIX farms currently consist of 10,000 MIPS of computing with over 300 workstations and about 100 tapedrives.

Software

The parallel processing code cps (cooperative processes software) allows the events to be sent to processes running on worker nodes and is an effective way to

provide parallel computing. One of the most important tools used on the farms is the tape-mounting software ocs (operator communications software) which is used to handle the large tape-mounting activity on the farms (and elsewhere at Fermilab). In addition, a batch system that allows queuing of jobs has been developed and is in use. Finally, many utilities are in place to allow debugging, optimization, and viewing of jobs on the farms.

Experiences

The UNIX farms are extremely successful in providing large amounts of cost-effective computing to the experimental users. Both CDF and D0 are able to reconstruct data as quickly as they collect it. The farms have sufficient processing power for present needs and no upgrades are necessary in the near future.

SPLITTING/FILTERING

After reconstruction the data is divided into many physics subsets. The subsets each consist of a sample of events relevant for a set of physics analysis topics and/or is a sample of events useful for background measurements and studies. Though not required, it is oftentimes the case that each event is compressed into a much smaller format containing only quantities essential for physics analysis.

Techniques

Each experiment chooses to handle this step of analysis in the way that matches best the physics of the experiment and to match the computing systems that are available for the process. Ideally the task would be specified independently of the hardware available but physical limitations (tape, disk, CPU) all dictate that many different systems are in use.

Examples – CDF and D0

CDF and D0 use somewhat different systems for performing the splitting and compressing of physics datasets after reconstruction. CDF splits datasets on the I/O portion of their UNIX farms. The SGI system consists of a Challenge XL with 4 processors, 80 GBytes of disk and 12 8mm tapedrives. The IBM system consists of an RS6000/590 and RS6000/580 with 140 GBytes of disk and 14 tapedrives. The events as they are reconstructed on the worker nodes of the farms are split into 25 physics streams (with some events being written to more than one stream). The output events can be stored in one of two formats – DST (full information) or PAD (physics analysis dataset). It is expected that the DST dataset will occupy approximately 6 TBytes and the PAD datasets about 875 GBytes from the 1994-95 run.

D0 reconstructs the data on their farms and produces two sets of output events. The first is the full-size (STA) dataset and the other is a compressed format dataset (DST). These datasets are further split on other computing systems. The STA sets are split into physics streams which fill approximately 17 TBytes. The DST sets are split into many more physics streams that sum to about 4 TB. Due to this large size a new reduced format (MDS) was invented to reduce further the size of data so that the whole sample could be compressed to about 250 GBytes. The STA sets are split on two dedicated SGI Crimsons fit-

ted with disk and tapedrives. The remaining splitting and filtering is done on D0FS, a large VMS cluster that is also used to serve data for analysis.

ANALYSIS

The physics analysis of the final datasets is accomplished on a wide variety of computing systems and in a number of different ways. Better access to datasets can and will improve the physics analysis of the data. There are many ways to characterize the many styles of physics analysis that are used. One way is to examine the three main sets of experiments (fixed-target, CDF and D0) to see what is being done.

Fixed-Target

The fixed-target experiments at Fermilab have access to a wide variety of computing systems both at Fermilab and at their collaborating institutions. In general it is possible to handle small datasets and final analysis steps (PAW) on local workstation clusters, most of which tend to be UNIX-based. The point at which the data becomes too large to handle locally varies but normally a sample which is smaller than 10 8mm tapes (or about 20 GB for single-density tapes) can be handled on a local system.

Fermilab has established two central UNIX systems which are meant to allow the fixed-target user community to access larger datasets for physics analysis. The two systems are CLUBS and FNALU. CLUBS consists of a set of SGI and IBM workstations connected with Ultranet for fast data access, the Load Leveler batch scheduling system, UNITREE hierarchical storage management, and access to external 8mm, 3480, and 9-track tapedrives and an STK silo. This system, with a CPU capacity of over 500 MIPS, allows users to analyze data that is staged from tape to disk via a staging system. Data is stored in the STK silo for rapid, reliable and multiple access. A dataset, once read in to the STK silo from 8mm tape, can be repeatedly read much more rapidly and reliably than from 8mm itself. A hierarchical system has proven effective in the past on the Amdahl mainframe in reducing manual tapemounts and providing reliable access for many physicists to a common dataset.

FNALU is a central UNIX system consisting of IBM and SGI computers divided into interactive and batch components. The AFS file system is being used on this system to provide home directory, product and some data access. The interactive systems are also used as front-ends to the CLUBS system. Batch jobs are prepared and submitted from FNALU to CLUBS. Local batch capacity on FNALU is available for users and applications that are not well-matched to the CLUBS system.

CDF

CDF uses two large central facilities for access to their datasets. The first is a large VMS Cluster (FNALD) consisting of about 500 MIPS of processing and 400 GB of disk space and a connection to an STK silo. In addition there are a large number of 8mm tapedrives directly connected to the Cluster to allow access to datasets. The STK silo contains CDF PAD datasets (and a small amount of DST datasets) which can be staged to disk and analyzed on the VMS cluster. The second system used is a UNIX sys-

tem consisting of an SGI 4D/480 (cdfsga), about 100 GB of staging disk, 16 8mm tape drives, and a connection to the same STK silo. The connection to the silo allows CDF UNIX analysis access to the datasets via a staging mechanism.

In addition to the central systems CDF has large local VMS and UNIX clusters which are heavily used for data analysis. The analyzer and the analysis project determine where each analysis will actually be done. Access to the data on the local machines is available from local disk or tape or from data copied over the network.

D0

D0 has established a large file server (D0FS) to provide access to large datasets. D0FS is a VMS Cluster consisting of about 500 GB of disk and 30 tapedrives along with 2 exabyte tape robots. The disk is currently spread across 34 workstations. The cluster is connected via FDDI to the D0 analysis clusters consisting of VMS (predominantly) and UNIX workstations. Users have the ability to access data using D0FS to serve data over the network or by reading local copies of the data.

Analysis strategy

The techniques that have been developed for analysis are not sufficient to meet the current and growing needs. The limitations of the current systems tend to limit or make difficult many analysis projects. The larger data samples that exist and that are going to be created in the future make it more important that strategies for data and computing access be investigated. The use of robots and hierarchical storage is essential in order that the huge manual tape-mounting load be reduced. One of the most difficult problems is the lack of scalability of most solutions now in use. Each increase in system size tends to create bottlenecks which are difficult if not impossible to overcome.

FUTURE DIRECTIONS

Analysis needs and computing in general benefit from having data as close as possible to computing power. Experience has shown that coupling computing and data as tightly as possible leads to large improvements in the ability to handle the data effectively. An attempt to implement such a strategy in order to analyze ever-growing amounts of physics data is the CAP (Computing for Analysis Project) system at Fermilab. In addition to putting data close to computing the project is scalable, leading to the ability to naturally handle increased data needs.

CAP

CAP is a project with a goal of providing HEP experiments with quick and reliable access to large amounts of data. The design goals are to store up to 100 TB of data in a tape robot (or robots), manage the files via a storage management scheme, read the data quickly onto a large (> 300 GB) disk pool and provide sufficient parallel I/O and computing to read through large datasets quickly. Various schemes of data management and storage are being prototyped in order to understand which techniques will be most effective in handling the needs of experiments in processing large amounts of data quickly.

The CAP hardware currently consists

of an IBM SP/1 multiprocessor system as the parallel compute and I/O system, the VESTA parallel file system from IBM Yorktown, UNITREE to manage the data, and a cartridge tape robot.

One of the major challenges of the system is to provide data organization and access that provides the speed and functionality necessary both for system performance and for analysis needs. Possible solutions include object-oriented data structures and other techniques for storing pieces of events to allow more efficient and logical access.

A successful implementation of this strategy promises a much improved system for handling the ever-increasing size of datasets. The CAP project should allow datasets of order 100's of GB to be handled with short turnaround times. This will create opportunities for data analysis which either do not exist today or are rendered rather difficult due to the long time and difficult data handling problems involved. The ability to handle the large datasets quickly will lead to better physics.

CONCLUSIONS

High-Energy Physics data sizes are growing ever-larger and this size increase is focussing effort on data-handling and data-processing of these growing sets. Fermilab has been blessed (or burdened) with a very large data-handling problem. A mix of ad-hoc solutions to the various aspects of the problem has been invented and is reasonably effective given current demands. This mix includes UNIX Farms for event reconstruction, a combination of UNIX and VMS Clusters for event stripping and filtering and a different and more diverse combination of UNIX and VMS Clusters, robotics, staging software and applications for physics analysis.

One of the ideas for improvement is to integrate fast data access with fast CPU resources. The CAP project at Fermilab is a system that can provide increased capabilities in an integrated system for handling the increasing demands of experiments. This approach can help us in making data analysis simpler and more effective for the ever-growing data samples of HEP.

ACKNOWLEDGEMENTS

Many groups of the Computing Division at Fermilab have contributed to this work, including the Farms, Unix Systems Support and VMS System Support Groups, the CLUBS group, the CDF and D0 groups and many others who contribute to the successful integration and maintenance of all the many computing systems. Data Center Services has done a magnificent job handling the huge amount of data that comes into the Feynman Center.

REFERENCES

1. F. Rinaldo and S. Wolbers, " Loosely Coupled Parallel Processing at Fermilab, " Computers in Physics, 7, 184, Mar/Apr, 1993.
2. M. Fischler, F. Rinaldo, S. Wolbers, "Production Farms at Fermilab", Contribution to CHEP94.

DATA MINING WITH PIAF

R.Brun, O.Couet, A.Nathaniel, F.Rademakers
CERN
Geneva, Switzerland

Abstract

Using the Parallel Interactive Analysis Facility, PIAF, users can query in real-time multi-gigabyte databases. Interactive response times are obtained by executing queries in parallel on a cluster of high speed workstations. For compute intensive queries an almost linear speedup is achieved using five workstations in parallel. In this paper we describe our Ntuple database model, the PAW system used to query, analyze and visualize the data as well as the hardware and software aspects of the PIAF system.

Introduction

As the amount of data from High Energy Physics detectors continues to increase, so does the demand for high performance tools for analyzing these data. Histograming packages like HBOOK [1] have been for many years standard tools for the statistical data analysis of physics parameters extracted from raw data, data summary tapes (DST), and finally mini and micro DSTs (see figure 1). In the 1980's the concepts of the Ntuple and PAW [2], the Physics Analysis Workstation, were introduced to the High Energy Physics community. An Ntuple is a convenient way of storing variables extracted from the detector data on an event-by-event basis. PAW is a powerful tool for the analysis of physics data stored in an Ntuple.

Initially, PAW provided only a very simple Ntuple structure, the Row-Wise-Ntuple (RWN), intended as a replacement for micro DSTs. The main disadvantage of RWNs is that the complete file must be read even if only a subset of the variables is referenced in a query. PAW version 2 introduced the concept of Column-Wise-Ntuples (CWN), where the amount of disk I/O is proportional to the number of variables used in the query. Since an Ntuple usually contains many more variables than are actually used CWNs allow interactive analysis of much larger data samples and can eventually replace mini DSTs (figure 2).

Nevertheless, the data samples of High Energy Physics experiments increase at a much higher rate than processor speeds do. PIAF is an extension to PAW which can keep up with the ever increasing amount of data by employing parallel processing. Although PIAF was designed and developed for the High En-

ergy Physics community it can equally well be applied in other fields where large databases must be analyzed in a limited amount of time. In general, PIAF is well suited for decision support or statistical database queries (data mining).

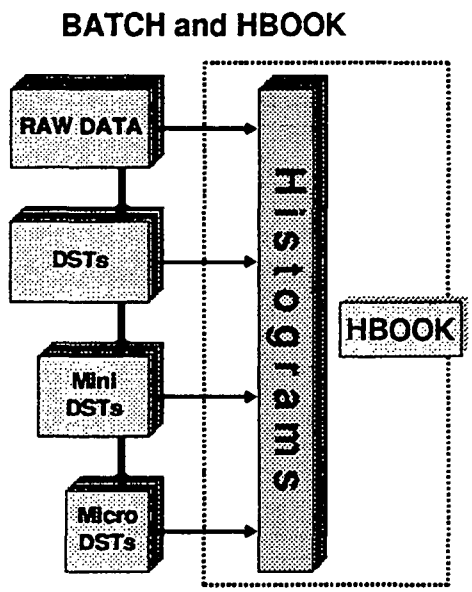


Figure 1. The Batch and HBOOK era

Short History

The PAW project was launched at CERN in 1986. The first public release of the system was made at the beginning of 1988. Many extension and improvements in the area of graphics and data visualization were added in the following years. Column-Wise-Ntuples were designed and implemented during 1991 and 1992.

The PIAF project was launched at the end of 1992 and the client-server communication between PAW and PIAF was incorporated into the July 1993 version. Initially, queries were executed by a single PIAF server per client. (Already then the PIAF system proved to be very popular amongst the first experimental users because it provided far more CPU power and disk space than the average personal workstation.) The parallel version which splits the workload between the machines in the PIAF cluster arrived in September 1993. This transition was transparent for the PIAF users as the necessary changes affected the server side only. In November 1993 PIAF was opened as a public service of the CERN computer center.

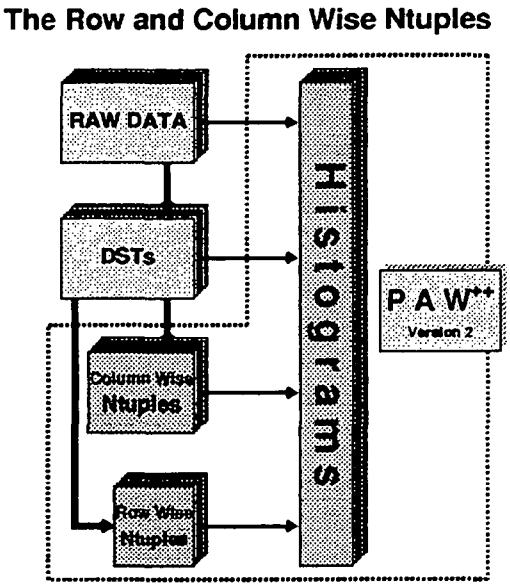


Figure 2. The Ntuples and PAW

The Column-Wise-Ntuple Data Model

In the Ntuple model the data is perceived as a table with a fixed number of columns (attributes) and a variable number of rows (events). One of the main tasks of PAW is to project any of the attributes into histograms according to various selection criteria.

Storage Scheme

Statistical data analysis requires direct access to complete columns, and the revised Ntuple storage mechanism has been optimized for this. The elements of each column are stored sequentially

into individual memory buffers (full vertical fragmentation). During Ntuple filling full memory buffers are flushed to a key-indexed direct-access file and reused for the accumulation of new data (see figure 3). Each key is unique and can be calculated from the row and column numbers. Hence CWNs also provide direct access to each row.

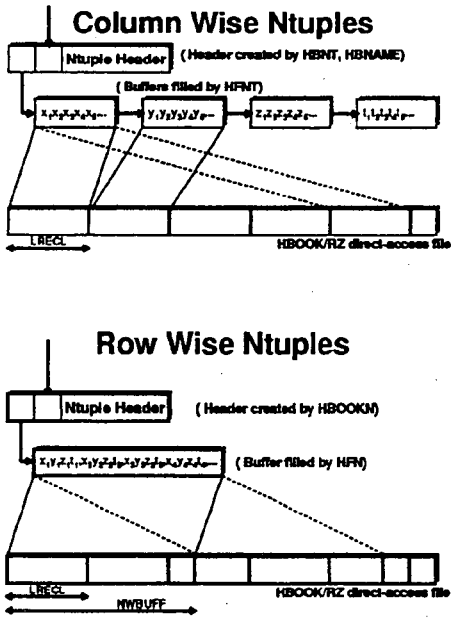


Figure 3. Column Wise and Row Wise Ntuples

Ntuple files are written in a machine-independent format which allows transfers by binary `ftp` or sharing via NFS between machines of different architectures. The column access time is independent of the total number of columns and depends only on the number of events rather than the file size. Only an absolute minimum amount of data needs to be read from file during queries. For example, accessing one column of a 100 columns Ntuple will require to read only 1% of the complete database. This storage scheme also makes it possible to add new columns a-posteriori as well as new rows without having to rewrite the complete database.

Logically columns are grouped into blocks. When defining an Ntuple blocks of columns can be added incrementally. This feature allows different program modules to define their own blocks whereas otherwise an Ntuple would have to be defined in one central place. Physically, however, all columns are stored independently in their own buffers.

Data Types

The Ntuple database supports the following basic data types: 4 and 8 byte floating point numbers, 4 byte integers, 4 byte bit masks, 4 byte Booleans, and character strings of up to 32 characters. Arrays of each of these data types are also supported.

Data Compression

For floating point and integer data types a value range can be specified which allows the system to pack a data item into the minimum number of bits. For example, if an integer column can only have values between 0 and 7, 3 bits are enough to store it. Booleans are always packed into 1 bit.

Arrays can be of variable length which depends on another column, the so-called index variable. For example, if an array column A(m) depends on the index variable N only the first N array elements are stored. Thanks to a unique indexing mechanism direct access to any row remains possible without storing additional information.

PAW - The Physics Analysis Workstation

PAW [3] is an interactive data analysis and presentation tool designed with the processing and graphics capabilities of modern workstations and high-end PCs in mind. It performs best on medium and high-end workstations with a high resolution graphics display. However, substantial effort has been made to ensure that PAW can also be used from graphics terminals and in non-interactive batch mode on mainframes or compute servers.

PAW comes in two brands. The basic version with a command line interface works on any platform. PAW++ provides a point-and-click interface for all workstations supporting the OSF/Motif windowing system. Both user interface chores are handled by KUIP (Kit for a User Interface Package) [4]. KUIP parses the commands, verifies the arguments and calls the relevant PAW action routines. KUIP also provides command aliases, a macro programming language and on-line help information. In Motif mode KUIP manages an object browser and a terminal emulator.

PAW can be used on many different graphics output devices due to the device-independent HIGZ [5] graphics package. HIGZ has drivers for most major graphics systems such as X11, GKS, PHIGS, GL, Post-Script, MacIntosh, PC, etc.

The Ntuple Query Processor

To facilitate the analysis and understanding of the data stored in Ntuples PAW has a wide range of commands that allow the user to query, plot and print variables (i.e. columns) or functions of variables. The four main commands are: PLOT, PROJECT, SCAN, and LOOP.

The PLOT command fills a histogram showing the statistical distributions of the variables and displays it. The projections can have up to 4 dimensions using perspective viewing and colors. As the other three commands do as well, the PLOT command allows for a selection criterion. The selection can contain comparisons between arithmetic expressions of variables combined with the logical operators AND, OR, and NOT. Complete flexibility is achieved by allowing user written Fortran or C functions in the selection expression.

The PROJECT command fills a predefined one- or two-dimensional histogram. It can be repeated several times to accumulate statistics from different Ntuples in the same histogram. In order to display the histogram a separate command has to be issued.

The SCAN command prints for every selected row a given subset of variables.

The LOOP command invokes for each row a user function which can be written either in Fortran or C. The function has access to all Ntuple variables of the current row for filling them into histograms etc. PAW incorporates the Fortran interpreter COMIS [6] which is available for all platforms but which allows to call only those external routines which have been linked into the PAW executable. Where supported by the operating system, dynamic linking of user routines compiled by the native Fortran or C compiler has been added recently to the PAW features.

Only the Ntuple columns used in the selection expression are read from disk and stored in a dedicated cache for reuse. In the case of user functions this is non-trivial since it requires to analyze

them for all variables actually used in the query.

Ntuple Chaining

Single Ntuple files can have a size of up to a few hundred Megabyte. The CHAIN command allows to define a set of files containing the same Ntuple structure which is then treated as a single logical Ntuple. Thus the maximum database size is limited only by the amount of disk space available.

The PIAF Facility

The Parallel Interactive Analysis Facility's main design goal was to create a system for analyzing multi-gigabyte databases in real time, that could be scaled with increasing amounts of data and that would have an affordable price tag. In order to reach this ambitious goal we run the Ntuple database engine and PAW's query processor in parallel on a dedicated cluster of high-end workstations interconnected by a high-bandwidth low-latency network. Another design goal was to make the system as transparent as possible for the PAW users. Also the system should not be limited to the workstation brand used in the PIAF cluster at CERN.

In the next sections we describe the hardware configuration currently in service at CERN and the organization of the PAW-PIAF client-server software. The paper concludes with the discussion of usage and performance figures.

The PIAF Hardware

The PIAF cluster currently installed at CERN (figure 4) consists of five HP 755 workstations each with 128 Mbyte RAM, 2.6 Gbyte internal disk and two RAID disks (one 6.5 Gbyte and one 10 Gbyte unit). The machines are interconnected by an FDDI network and a conventional Ethernet. In the near future (Q2-94) the cluster will be expanded by adding three machines and upgrading each node to 256 Mbyte RAM. In Q3-94 the CPU's will be replaced by 50% faster models.

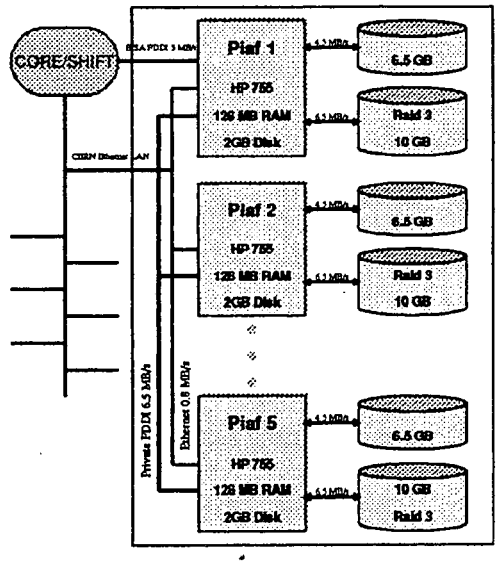


Figure 4. The PIAF hardware (April 1994)

1.6 Gbyte of the internal disk is configured as swap space, while the remaining 1 Gbyte is used for the operating system and the users' working directories. Ntuple files are stored on the RAID disks. Since the Ntuples are staged from tape or transferred from file servers onto PIAF, they are considered as "volatile" data and therefore not backed-up. Some form of security is provided by the RAID systems (HP-C2425 and HP-C2430) running in RAID mode 3, which combines high throughput (up to 6.5 Mbyte/s) with data protection. By cross-mounting the RAID systems, using NFS over FDDI, each machine in the cluster has access to a disk pool of 82 Gbytes.

Hardware Robustness

The stability of the cluster has been of no concern up to now. Between November 1993 and April 1994 there was no unscheduled down-time. One disk of a RAID unit failed and was replaced without loosing any data.

The PIAF Software

The core of the PIAF server software is the PAW query processor and the Ntuple database engine. These core components are packaged together with interprocess communication routines in the `piafserv` program. Depending on its start-up arguments `piafserv` acts either as master server or as slave server. Each user connecting to PIAF will get one master server and N slave servers, where N is the number of available nodes (see figure 5).

The client is the user's local PAW session which communicates with the PIAF master server through a TCP socket. The messages exchanged are usually small enough to allow the use of PIAF also through a WAN or SLIP connection. The master server coordinates the database queries: it forwards the query commands to the slave servers, receives and merges the results, in the form of histograms or lists of row numbers, and feeds the final results back to the client.

Each slave knows by its *group-view* which part of the total database it has to process. The group-view consists of two numbers: the slave server's unique ID (from 1 to N) and N (the number of active slave servers). If a slave does not respond within a certain time limit it is considered dead and the master server sends out a new group-view to the remaining slaves.

Since all slave servers traverse an equal share of the total database the system is automatically load-balanced. In order to maintain this balance there are no interactive sessions allowed on the CERN PIAF cluster.

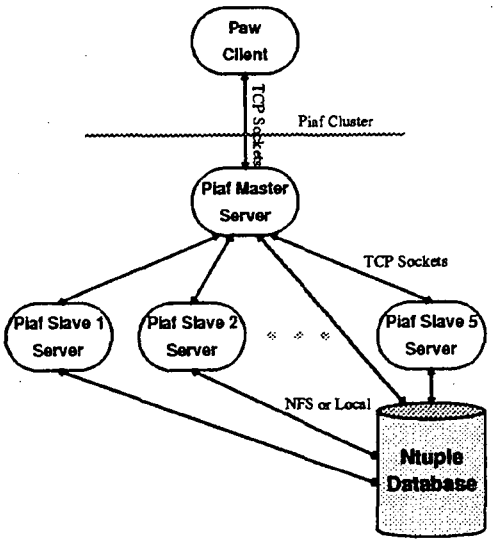


Figure 5. The PIAF processes

Client-Server Communication

The PIAF master server is started by the `inetd` super-daemon when a client connects to a specific port on one of the nodes in the PIAF cluster. The `inetd` daemon starts the small front-end server `piafront`, which handles the user authentication. To distribute the master servers evenly over all nodes `piafront` also checks on which node in the cluster the master server should be run. If the authentication was incorrect or the node is not the right one `piafront` sends back an authentication failure or reroute message. Otherwise it overlays itself with the master server. The master server started depends on a version tag sent by the client. This allows for incompatible

changes in the server software without affecting users still running old client software.

Once the master server is launched successfully it, in turn, starts the slave servers in a similar way. Then the client sends some initial state information to the master server which forwards it, including an initial group-view, to the slaves. During a session the user can interactively change the number of slaves participating in a query.

To meet stringent performance, functionality and portability requirements the client-server communication is done via TCP sockets. Message passing libraries, such as PVM, were considered but rejected due to the extra overhead and configuration chores they introduce. Also, using sockets directly enabled finer-grain control over communication behavior (e.g. signal driven asynchronous I/O and buffering).

Transparent Usage

One design goal was to keep the system as transparent as possible for the PAW users. The set of new commands to access PIAF is limited to a bare minimum. The opening and closing of a PIAF connection is done using the commands CONNECT and DISCONNECT. When connecting to PIAF the user is asked for his PIAF login and password. Once connected to PIAF the user can stage Ntuple database files using the PUT and STAGE commands.

To access data files on PIAF the user simply precedes the file name with the keyword '//piaf'. For example:

```
PAW > HISTO/FILE 1 //piaf/staff.hbook
```

tells PAW to connect the file 'staff.hbook' on PIAF to the logical unit 1. Compare this with the command to connect the local file 'staff.hbook' to logical unit 2:

```
PAW > HISTO/FILE 2 staff.hbook
```

From this moment on Ntuples in both files can be queried in the same way. The only difference is that when an Ntuple from logical unit 1 is queried the request is transferred to the master server and via the master server to the slave servers. For example the command:

```
PAW > NT/PLOT //LUN1/10.age cost>10000
```

will be executed in parallel on PIAF, whereas the command:

```
PAW > NT/PLOT //LUN2/10.age cost>4000
```

will be executed by the local PAW session.

Since PIAF is a general public service a lot of effort was spent on robustness. Especially, PIAF is able to gracefully handle keyboard interrupts by users who want to abort an on-going query.

Usage and Performance Monitoring

In order to be able to track the usage and performance of the system an extensive monitoring scheme was implemented using the BSD syslog facility. The information thus gathered tells us, for example, how many users used the system, how much data was processed, what kind of queries were executed, the real and CPU times used per query, the Ntuple cache hit rate, etc. After storing this data into Ntuples, PIAF is used for the analysis.

Figure 6 shows PAW and PIAF usage on the IBM mainframe and Unix workstations at CERN since January 1993. PAW itself is routinely used by more than 1000 users per week. The top part of the

graph shows the amount of data analyzed per week. The 50 users currently daily on PIAF process on average 500 Gigabytes per week, to be compared with only 150 Gigabytes by the 1000 PAW users analyzing local data sets. The number of users and the amount of data processed per week increases very rapidly. We expect PIAF to become very soon the main data cruncher on the CERN site. The bottom part of the graph shows a similar trend for the CPU usage: 4000 CERN Units hours per week on PIAF.

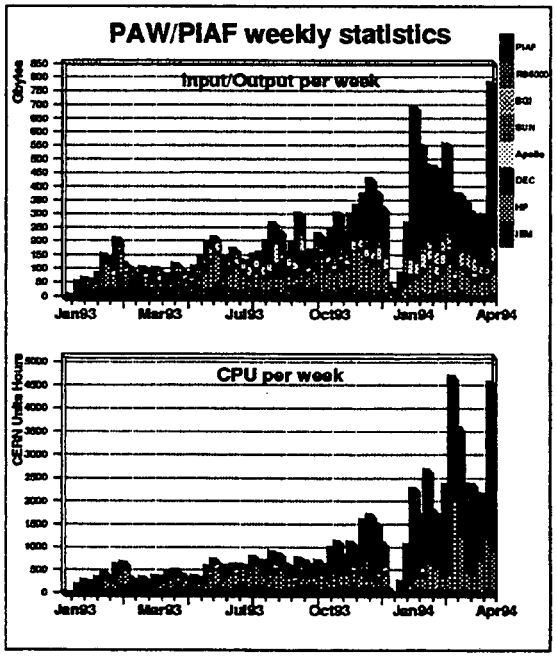


Figure 6. PAW and PIAF usage at CERN. (One CERN Unit is approximately 5 MIPS.)

Statistics with the Data Model

We are currently collecting statistics which will allow us to understand better the type of queries as well as the percentage of the database really used per query. Figure 7 shows the result of a few days' snapshot involving 137 000 queries. The top part of the figure shows the number of columns in the user Ntuple databases (average 57 columns). The bottom graph shows the percentage of the number of columns really involved in one query (average 20%).

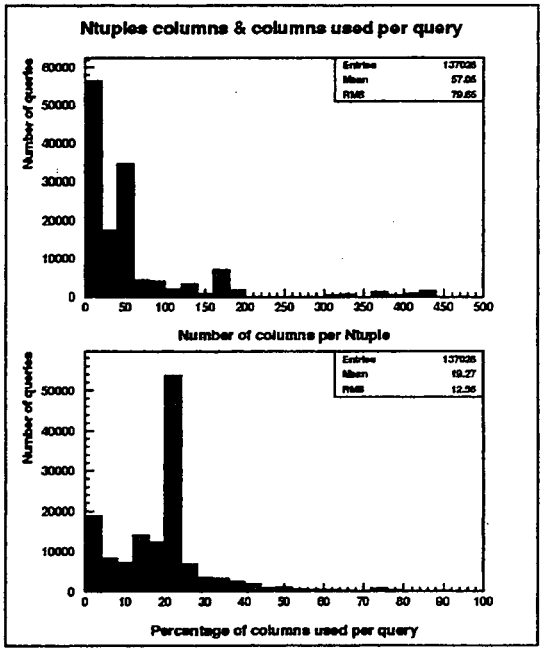


Figure 7. Data Base profile and Columns use

Caching

In a normal PAW/PIAF session, the user issues often queries involving the same columns or combinations of columns again and again. The database engine in common to PAW and PIAF contains a caching mechanism which keeps the most frequently used columns in memory. The size of the cache is a function of the available amount of memory and of the local swapping space. We are continuously trying to improve this crucial part of the system.

Using 5 slaves in parallel, the system can read and process a column of 40 million entries in about 120 seconds. This time is independent of the total number of columns in the database). Processing a

column already in the cache takes about 50 seconds.

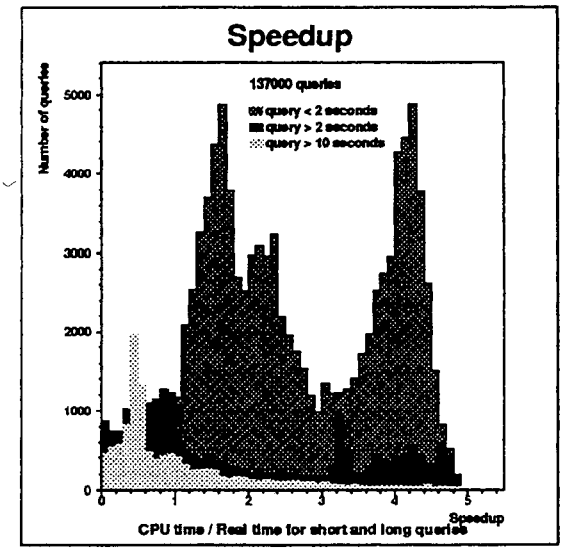


Figure 8. Speedup obtained with 5 processors

Performance Figures

For one single user, the PIAF performance scales nearly linearly with the number of processors participating in the query. This is due to the small amount of time spent in the master-slave communication compared to the time it takes to traverse the database. The larger the Ntuple the better the speedup since the amount of communication is independent of the number of records processed. In some cases even a more-than-linear speedup can be observed. This happens when the database is too large to fit into the cache of a single machine. Running in parallel on N nodes, however, the speedup is more than a factor N if each node is able to cache its part of the database.

In real life, obviously, several queries from different users may compete for resources. Figure 8 shows the speedup (CPU time on all nodes divided by real time) for different types of queries. In the case of short queries, the probability to have two competing queries is small and the speedup is typically 4 (using 5 processors). When the time to process one query increases, the probability to have several competing queries also increases. Despite the small size of the current system, the PIAF users enjoy a response time far better than they could expect on their own workstation. In addition, the PIAF system provides a data storage capacity unthinkable for single-user workstations.

Problem Areas

First time access to a file does not result in much speedup because the slaves have to read from a single disk device where the database is stored. In order to avoid this bottleneck we are investigating the design of a parallel file system. Files will be striped, during staging, in N equal parts each stored on a separate device. Ideally, the splitting should be done such that each slave will find its share of the file on a local disk. Otherwise the next bottleneck will become the network throughput. To alleviate the network bottleneck we plan to interconnect the cluster via a HiPPI network which should yield a throughput of about 25 Mbyte/s.

Conclusions

In this paper we presented the optimized Ntuple database with column-wise access to statistical data. Further, we described the interactive analysis and presentation tool PAW and its Ntuple query language. Finally, the hardware and software aspects of the Parallel Interactive Analysis Facility, PIAF, were explained.

The first phase of the PIAF project was quite successful. Since November

1993 the system is open to the public and services many users with a remarkable robustness and stability. We are planning to extend considerably the power of the system in the coming months.

Nevertheless, many areas still need to be researched and clarified, for example, a parallel file system, a scheme for automatic data migration, and the scalability with respect to an increasing number of users and nodes. Having real users on the PIAF system has given us new insights in the queries profile.

REFERENCES

1. HBOOK Users Guide (version 4.21), CERN Program Library Y250 (1994).
2. R. Brun et al., PAW: A General Purpose Portable Software Tool for Data Analysis and Presentation, Comp. Phys. Comm. 57, 432-437 (1989).
3. PAW Users Guide, CERN Program Library Q121 (1993).
4. KUIP Users Guide, CERN Program Library I202 (1994).
5. HIGZ Users Guide, CERN Program Library Q120 (1993).
6. COMIS - Compilation and Interpretation System, CERN Program Library L210 (1994).

HARDWARE AND SOFTWARE ISSUES FOR FUTURE EXPERIMENTS

Ken Peach,
Department of Physics and Astronomy,
University of Edinburgh,
Edinburgh EH9 3JZ, UK

Abstract

Issues influencing the design and implementation of the computing environment are discussed in the context experiments now being designed for the fixed target programmes at FNAL and CERN.

INTRODUCTION

There has traditionally been a sharp division between *offline* and *online* computing. Rapid changes in computing technology – leading to *downsizing* of the mainframe and to buffering of data which diminishes *real-time* aspects of control room computing – are blurring the distinction. Higher event rates and larger data volumes also involve a radical departure from bus-based data acquisition systems developed in the 1970's and 1980's, to a new architecture using independent, parallel links; this needs new methods to assemble the information related to a single event. Developments in storage technology hardware, software and networking, influence the development of integrated experimental computing systems. Finally, the route to very large CPU and memory may involve MPP.

This paper is organized into five main sections. Firstly, there is a brief review *Yesterday, Today and Tomorrow* of the development of experiment computing from data acquisition to data analysis. Then, there is a discussion *Before and After the Event* of the implications of technology changes at the interface between the real-time and standard computing environments. In *Control and Data* the design principles of modern counting-room computing systems are briefly discussed, followed by a more detailed discussion *Logging, Farming and Transportation* of some key issues. Finally, a few comments on longer term developments are mentioned in *MIPS and Memory.*

YESTERDAY, TODAY & TOMORROW

There are seven distinct phases through which event data pass, from the detector to the final publication. These are:–

1. Event Assembly
2. Software Filtering
3. Online Manipulation & Monitoring
4. Archive Storage (*Tape*)
5. Offline Reconstruction and Processing
6. Data Summary *Tapes*
7. Analysis

Montecarlo *data* is usually injected to the chain at the 4th step, and thereafter can

Table 1. Characteristics of the data flow for the three ϵ'/ϵ experiments at CERN and FNAL

	NA31	NA48	KTeV
Data taking	1984-9	1995-	1995-
Spill length (seconds)	2.7	2.7	20
Duty Cycle (%)	20	20	35
Design Goals			
Events rate (kHz)	0.75	10	15
Data Volume (Mbytes/sec)	1.5	100	150
Time average (Mbytes/sec)	0.3	20	50

dominate the computing requirements.

In the 1970's (*yesterday*), there were just three separate elements in this scheme. Event assembly was performed by the online *mini* (PDP11 or NORD, for example) accessing the detector directly through CAMAC or less usually another bus, there was no software filter because there was insufficient computing power, the online computer being fully occupied monitoring the data and writing to tape (1600bpi!). These tapes were then transported to the mainframe (usually IBM), which did the rest of the computing – reconstruction, data reduction and analysis. The height of sophistication in graphical devices was the Tektronix 4010.

In the 1980's (effectively *today*), the event assembly had moved out into the bus (FASTBUS or VME), and bus-based processor farms built round single board computers implemented the software filter. The online system was a much larger, general purpose mid-range computer (VAX, NORD500 etc), which controlled the experiment, obtained the assembled data from buffer memories embedded in the bus, monitored the data and wrote them to tape. The tapes (6250bpi and latterly 3480) were then transported to the mainframes (even bigger IBM's) for reconstruction. Much analysis was still performed on these central systems, although it gradually migrated to group or departmental/institute machines (VAX'es, HP...). Sometime during the decade, colour terminals were discovered, which released the repressed artistic talents of physicists for display, rivalled only by the more flamboyant birds of paradise. The end of the decade saw the introduction of workstation clusters for many of these tasks.

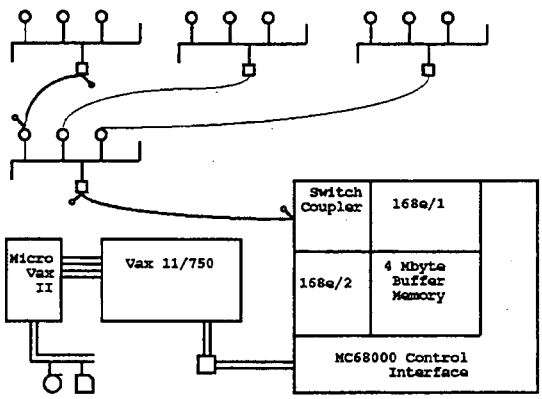
The 1990's (*tomorrow*) takes the trend further. Workstation clusters now appear everywhere – for online work, for reconstruction and for analysis. Buses such as CAMAC, FASTBUS and VME are being replaced by point-to-point links (HIPPI, FCS, SCI...). Large disk farms are replacing *tapes* for the storage of DST's, and the archive media are becoming more exotic – Exabytes, DD-n, DTP... Finally, the assembly of the event is moving from software to hardware.

It is interesting to compare how the computing system for experiments has evolved by considering some specific examples. Since I know the issues involved well, I shall use the NA31 [1] and NA48 [2] experiments at CERN, and KTeV [3] at FNAL; what these experiments have in common is that they are studying the same physics (ϵ'/ϵ). These studies thus illustrate how the *requirements* have developed over the years and, most interestingly from my point of view, how the similar *solutions* are. Table 1 shows the principal characteristics of the data flow for each of these experiments.

The NA31 data acquisition system is

typical of data acquisition systems designed in the early 1980's – see Fig. 1. FASTBUS was used as the main data

Figure 1. The NA31 data acquisition system



path, with the events being *built* in the bus using a Kinetics Systems FASTBUS Block Mover [4]. The assembled data were transmitted to one of two IBM 168e emulators [5], where a fast filter programme reduced the data by a factor of nearly two. Accepted events were written to a large (4MByte!) buffer memory, and read between SPS spills into a VAX 11/750. Towards the end of the data taking, the VAX was supplemented by a microVAX-II. Data were written originally to 1600 bpi tapes, then to 6250 bpi tapes and finally to IBM 3480 cartridges on an Aspen system. (Now, all of the data have been copied to Exabyte.) However, as we shall see, this architecture does not scale well to higher event rates and data volumes.

BEFORE AND AFTER THE EVENT

The most dramatic architectural change during the development of data acquisition systems has been in the assembly of the data from a particular event in the detector into a single data structure, suitable for archive recording and offline analysis. Initially, such event assembly was under the direct control of the data acquisition computer, reading CAMAC after an interrupt signalling the occurrence of an event. Much intellectual effort was devoted to making this interrupt processing efficient – either by the computer manufacturer in the design of the context switching mechanism, or by the online programmer in trying to short-circuit the safeguards and bottlenecks introduced by the hardware or software. More recent designs have used *Event Builders* in the data acquisition bus itself to perform this task – running a minimal operating system allows a rapid response to interrupts, and relieves the data acquisition machine of this chore. However, there are limitations to this approach. For example, under programme control it usually takes time – of order μseconds – respond to an interrupt. Gaining bus mastership also takes a similar time, as does the scanning and addressing of each source of data. If we allow 5μseconds for each of these the *zero-data* overhead for 10 independent data sources is 60μseconds, leading to 60% occupancy of the bus at $10kHz$ *before* any real data have been transferred [1]. It is interesting to note that the excellent general data acquisition system developed at CEBAF [6] has achieved $30kHz$ rates for small event sizes from a small number of sources, in agreement with the above rough estimates.

The problem with bus-based, data-pull architectures is that they are not *scalable* – the bus (and its protocol) are

[1] Strategies can be adopted to reduce these overheads, but in general only at the cost of increased complexity.

the limiting factor, and as the event rate goes beyond some limit, the throughput *must* go down. For these reasons, some designers have abandoned the bus-based architecture for a much simpler transport system – multiple, independent, point-to-point links, along which the data *delivers itself* to a large buffer memory. The advantages of this scheme are several.

1. More capacity can be provided by increasing the number of links.
2. Point-to-point links can have very low protocol overheads.
3. Point-to-point links are in general lower cost than bus-based systems.

In addition, the link can be made unidirectional, since the bulk of the data flows in only one direction. A bus-based architecture allows data flow in both directions equally, although this functionality is rarely used in practice. It is better therefore to separate the main data path from the control path (see below). Of course, there are new problems – for example, controlling the data rate. However, this turns out to have a surprisingly simple solution – use a single return line (WAIT or XOFF) to control the flow. Since the general structure of such systems is that data is transferred from a buffer into a FIFO, it is possible, knowing the capacity of the link, to predict well in advance when the system will flood, and hence send the flow control signal sufficiently early to avoid difficulty. The signal is asynchronous, and can be made completely immune to noise etc. An attractive feature is that temporary congestion on one link has no effect on the other links, and the system approaches saturation gracefully. Both the NA48 and KTeV (DART) [7] data acquisition systems use very similar structures – see Table 2.

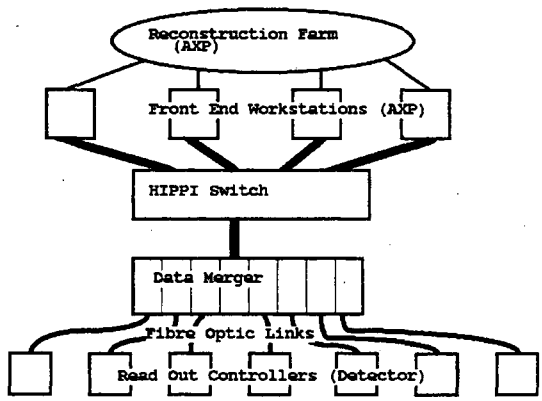
Table 2. Comparison of the NA48 and KTeV event assembly

	NA48	KTeV
Total Data Rate (Mbytes/sec)	100	150
No. of Links	>10	>5
Speed/link (Mbytes/sec)	13	40
Protocol	custom	RS486
Medium	fibre	copper

Differences emerge *after* the event, in part because of the very different duty cycles and spill structures at the SPS and TEVATRON. In the NA48 system (see Fig. 2), the links terminate in the *DataMerger*, which has a 2Mbyte buffer for each link. The data are stored event by event in each link, and concatenated by a fast token-controlled interface to HIPPI and transferred directly into a workstation – currently a *Digital* AXP 3000 series TurboChannel machine. A HIPPI-to-Turbochannel interface has been designed and commercialized [8], and delivers at least 83Mbytes/sec sustained transfers. This architecture is possible because the maximum size of the data for a whole burst is 256Mbytes, well matched to reasonably cost-effective workstations. For KTeV (see [7]), the total data volume for a single spill could be 3Gbytes. The solution here is not to assemble the event at this point, but to reduce the data by accessing the data directly while it is stored in VME memories.

Both experiments have designed *scalability* into the system. For NA48 this is achieved by using a HIPPI switch to select a workstation for a complete spill of data, and providing sufficient destinations to be able to process the data before the pool of workstations is exhausted. KTeV

Figure 2. Sketch of the NA48 data acquisition system



choose to scale their data by depositing the events into different VME memory modules (selected by the event number in the event packet), and by using SGI Challenge multi-processor machines to achieve the required CPU power.

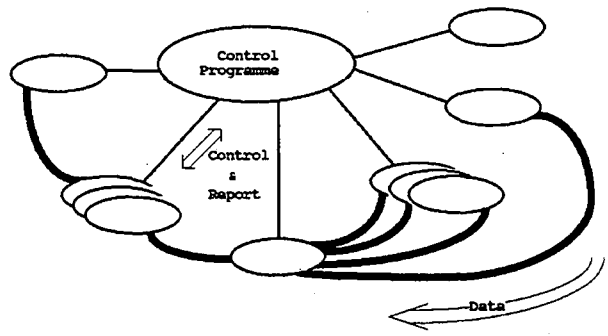
CONTROL & DATA

A very important consideration is the medium for controlling the experiment. In traditional, bus-based data acquisition systems, equipment setup and diagnostics, trigger definition etc were configured through the *same* bus system used for the transport of the bulk data. While this has a certain appeal – economic reuse of the bus – it is nevertheless inelegant. The volumes of data flowing in the two directions are very different, and of a very different character. The control functions are in general *not* interleaved with the data, and the extra complexity of bi-directional, complex protocol buses (FASTBUS springs to mind) mean in the end extra cost.

The more modern approach is then to separate *completely* data and control. The data transport has been discussed above; since each detector subsystem will in practice come with its own computer for the engineers to setup and test, it makes sense to use this route also for control, using whatever local area network is available. Most control functions (for example, setting up or starting a run) are not time critical at the milli-second level, and hence the performance overhead of a properly designed and managed local area (experiment) network is light.

The resulting architecture (see Fig. 3) for data and control is thus like the road layout of any major city (*Paris* being perhaps the most elegant example). In the centre is the control console (*L'Arc de Triomphe*), and radiating from it are the arterial roads which carry the command and response signals to each of the subsystems (the *Portes*). The data is transported in a multi-lane highway (the *Boulevard Périphérique*) around the edge.

Figure 3. Control and Data Architecture



In future, as both trigger rates and data volumes increase, even more radical departures from the traditional bus-based, data-pull architecture will have to be considered. However, the trend will probably continue to be to point-to-

point links transporting the data asynchronously to event assembly *in hardware*, albeit relatively simple hardware. The secret is to design the data flow scheme so that the event assembly is simple – perhaps just by memory address, for example.

Once the event is assembled, it must be processed, monitored and archived, which leads to a discussion of ...

FARMING, LOGGING & TRANSPORTATION

... which, as Delfino has pointed out [9] has evolved from subsistence farming to Agribusiness. There are three related topics – the CPU required to satisfy the requirements of the separate stages of monitoring, reconstruction and analysis, the storage of the data and the network fabric which connects these elements together. Now, it cannot be stressed too often *these three subjects* – CPU, storage and networking – *should be considered as a single system* and should be incorporated into a global data model.

Here, the benchmark is, in some respects, provided by the DESY system [10]. At DESY data recording has traditionally be centrally organized. However, the central *organizer* has been equally traditionally been a large IBM mainframe. More recently, as Ernst has discussed at this conference [10], the advantages of this approach have been retained, but the new structure is built not around a central mainframe, but around a high-performance *network switch.* Complementing this network is a site-wide data structure, which is an implementation of the IEEE Hierarchical Storage Model.

The claim was made that this is a *cost effective* solution to data storage issues; this is undoubtedly true. However, the initial cost is high, and it is therefore most appropriate for large data centres. Here, the particular features of DESY are important, supporting as it does only a small number of large experiments. The initial capital investment is rather a small part of the total cost. What is less clear is whether this can be also implemented for sites – such as CERN and FNAL – where a large number of small or medium sized experiments periodically take data.

Many of the features of a modern computing centre are also to be found in the computing systems located in control rooms and institutes – clusters of workstations, PC's, X-terminals, storage devices etc connected by a central backbone network or ring. The difference is only in the relative scale of computing systems, and the even greater difference in the scale of the respective budgets.

The key question relates to the archive medium. As Mount remarked [11], the days of a standard, interchangeable medium (such as the good old 1600bpi round tape) are probably gone for ever. Given that even small experiments take data over many years, the consequences of a wrong decision taken at the beginning can be serious. (Few people have the faith in Exabytes displayed by FNAL.)

One possible solution is to abandon writing data directly from the experimental computers and instead treat the problem as one of data migration – in effect the DESY model. This has a number of advantages. Firstly, it removes completely the traditional problems of online systems related to the handling of media – wrong tape mounted, right tape but wrongly labelled, unloaded a tape before it has completed writing etc – all too fa-

miliar to online programmers [2].

In the process, something rather magical happens – the traditional distinction between *online* and *offline* more or less disappears. Control and data have been separated, and data recording abolished, replaced by the HSM. All programmes access the data in the same way, and there is rather little code which is specifically *online* or exclusively *offline*.

A further advantage of this approach to data logging is that it allows in principle efficient use of computing resources. As noted above, workstation farms now feature in many places – in the control room, in the data centre and in the group & institutes. Once the data is accessible transparently, the machines in these various clusters can be allocated dynamically to tasks as demand dictates.

Critical to the above structure is the disk farm. One of the weaknesses of the processor farm is currently the access to large disk storage. Attaching to a single machine gives a single point of failure, and multiple cross-mounting of disks can be inefficient. In this context, then, it is interesting to look at some ideas being developed by IBM for a low cost, medium performance alternative to SCSI, called SSA (Serial Storage Architecture) [12]. This is being proposed as an ANSI standard. Basically, SSA is a bi-directional serial link with generous addressing capabilities using a light protocol. It maps on to SCSI-II and provides currently 20Mbytes/sec in each direction. The protocol has automatic retry at the hardware level, and has a single-chip implementation. IBM's aim is $15 per port. SSA can be configured as a linear chain, ring, crossbar etc for a mix of CPU's and storage devices, and utilises through-routing (similar to SCI). The protocol is worth serious study by farm managers, and also perhaps more generally; IBM are to be encouraged.

This underlines the importance of the *data network* – something made explicit in the DESY model by the central position accorded to the *GigaRouter*. Of course, networking may not solve *all* problems of getting the data to the CPU – as Tannenbaum remarked [13] "*never underestimate the bandwidth of a pickup truck*". Wolbers [14] showed us a Terabyte in a cabinet drawer the size of a suitcase; transported by air anywhere in the world (about 20h) gives a mean bandwidth of about 13.5Mbytes/sec – or 155Mbit ATM at full speed!

Why might the data need to be brought to remote CPU's?

MIPS & MEMORY

There is a healthy debate about whether there is a need for Massively Parallel Processing (MPP) in *experimental* particle physics – QCD have already demonstrated the need in *theoretical* particle physics. The argument *against* is essentially that the raw CPU power will always be more cost effectively provided by a network of workstations than in an integrated box, since the box manufacturer always wants to give *added value* to the processing power through hardware and software features which may either not be needed or difficult to exploit. The rather small impact of vector processors on experimental particle physics code is often quoted as an example. On the other hand, the management of thou-

[2] Some physicists seem particularly adept at creating tape unit conditions unimagined and unimaginable to the conscientious programmer.

sands of machines loosely coupled by a general network is complex, and there may be natural scaling limits to such systems. These can of course be extended by using clusters of multi-processor machines, which generates what might be called *moderately* parallel processing.

There is, however, a class of problems for which the workstation farm approach may not be cost effective, and that is where large CPU needs are associated with large *memory* requirements. The reason is that memory costs become dominant beyond some limit – today when the memory required is greater than about 128Mbytes. Distributing the memory around a large number of workstations transfers the problem to the network, which may not be able to support the large bandwidth required. Here, MPP *may* be cost effective. For a discussion of these matters, see the parallel session talks by Bertolotto [15] and Panzer-Steindel [16].

SUMMARY

This brief review has examined some issues to be addressed in the design of the computing environment from the control room to the conference report. Developments in computing technology allow radical new approaches, integrating all aspects of experimental computing. Advances in networking allow new solutions to standard problems of experimental data acquisition and control, which are at the same time more powerful and less complicated. While the scale of the problems is not quite the same as for the LHC experiments, the challenge is likely to provide sufficient intellectual stimulus for many over the next few years.

REFERENCES

1. GD Barr *et al*: Phys. Lett. B317 (1993) 233-242; H Burkhardt *et al*: Nucl. Inst. & Meth. A268 (1988) 116
2. GD Barr *et al*: "Proposal for a precision measurement of ϵ'/ϵ in CP-violating $K^0 \rightarrow 2\pi$ decays", CERN/SPSC/90-22, July 1990
3. K Arisaka *et al*: Experiment E732, FNAL
4. J Pregernig: IEEE Trans. Nucl. Sci. NS-33 (1) (1986) 797
5. PF Kunz *et al*: Nucl. Inst. & Meth. 9 (1976) 435; M Rost *et al*: Nucl. Inst. & Meth. 242 (1985) 153
6. G Heyes *et al*: these proceedings
7. R Pordes *et al*: these proceedings
8. W Bozzoli *et al*: "Conference Record of the Eighth Conference on Real-time Computer Applications in Nuclear, Particle and Plasma Physics", Vancouver, June (1993) 105-107. The HIPPI to TurboChannel Interface HTIP-800D is manufactured by Hytec Electronics, 64, Amy St, Leicester LE 3 2FB, UK
9. M Delfino: these proceedings
10. M Ernst: these proceedings
11. R Mount: these proceedings
12. Copies of the SSA specification documents can be obtained from Global Engineering Documents, 15 Inverness Way East, Englewood, CO80112-5704, or from IBM-UK at PO Box 6, Havant, Hampshire PO9 1SA, UK.
13. A S Tannenbaum: *Computer Networks*, Prentice-Hall International, ISBN 0-13-166836-6 (1988)
14. S Wolbers: these proceedings
15. L Bertolotto *et al*: these proceedings
16. B Panzer-Steindel *et al*: these proceedings

METHODOLOGIES, LANGUAGES AND TOOLS
Session Summary

Katsuya Amako
Physics Department
KEK, National Laboratory for High Energy Physics
1-1 Oho, Tsukuba, Ibaraki 305, Japan

Abstract

This is a summary of the "Methodologies, Languages and Tools" session in the CHEP'94 conference. All the contributions to methodologies and languages are relevant to the object-oriented approach. Other topics presented are related to various software tools in the down-sized computing environment.

INTRODUCTION

The basic approach of the present summary is to digest the talks given in the session one by one. It serves as an introduction to each contribution. A reader who wants to know the details can read the corresponding paper.

There are 21 contributions in the sessions. In this summary, the author categorized these as follows:

- Methodologies and their tools;
  - Methodologies in on-line software designs;
  - Methodologies in off-line software designs;
  - Tools for methodologies;
- Languages;
  - Languages in on-line software implementations;
  - Languages in off-line software implementations;
- Generic tools.

The first category includes contributions relevant to usage of software engineering methodologies for designs of on-line and off-line software systems. It also contains software tools to support methodologies. The second category has contributions related to program languages for implementations of on-line and off-line software systems. The third category contains topics of generic tools in the down-sized computing environment.

METHODOLOGIES AND THEIR TOOLS

Whether the introduction of a software engineering methodology bring improvement of the productivity and the quality of HEP software systems has been debated repeatedly in a series of these conferences [1, 2, 3]. Although there have been no convincing proof of their definite success so far, demands to such a methodology is high, especially in large scale experiments in the future.

In the session, there were two contributions [4, 5] on applications of object-oriented (OO) methodologies to on-line

systems of on-going experiments. These showed that OO methodologies are already not just for a trial but a practical and indispensable tool for some of on-line software designers. Also, there was one contribution for formal methodology [6]. For off-line systems, there were three contributions [7, 8, 9] of serious studies for feasibility of OO methodologies. Because off-line applications are more close to most physicists than on-line ones, these presentations brought about hot discussions. Some of these are: 1) do all physicists need to understand OO methodologies? 2) an ideal software system should allow most physicists to integrate their own codes to it without the knowledge of OO methodology nor OO language; 3) most physicists should not touch to the code, instead, a Macintosh-like user environment should be provided for their physics analyses, etc. There were no clear conclusions from these discussions.

Structured Analysis / Structured Design (SA/SD) once drew the attention of HEP experiment groups. Several groups used it for designs of their software systems, but its usage declined gradually. It is pointed out that one reason for this decline is that there were few computer aided software engineering (CASE) tools to support the methodology [3]. This problem seems to be not exist in OO methodologies, because many CASE tools are available in the present market. In the session, there were three reports [10, 11, 12] on the projects to develop tools which support OO methodologies. Because of defects seen in CASE tools in the present market, these projects try to construct a free tool for the HEP community.

Whether OO methodologies and their CASE tools bring a new approach of software development in HEP is not clear yet. We need to keep our eyes on how these evolve in future.

The following are digests of all contributions belong to methodologies and their tools.

Methodologies in On-line Software

One of the most extensive application of a methodology to a working system is given by **M. Flasinski** [4] of DESY. ZEX is a distributed real-time diagnostic/control system for the ZEUS experiment, which was implemented to help on-line operators to keep the experiment in optimum running condition. The Booch object modeling technique was employed to design the system.

The decomposition of the ZEX system by the Booch method yielded a multidimensional frame structure which allows to treat each detector component according to various aspect of its functionality. He concluded that the OOA/OOD approach is very useful for designing and implementing expert systems, because their experience showed that the methodology leads to a well-partitioned maintainable system.

Another example of the application of a methodology to a working system is presented by **S. Dû** [5] of Orsay. Requirements to the DELPHI on-line event display system, such as its interactive nature and necessity of a regular evolution depending on the physicists' wishes, naturally lead them to employ an OOA/OOD methodology to the design and implementation. The system was designed by the Objecteering tool, which is based on "class-relation method". The tool pro-

vides an integrated development environment which includes graphical model editor, C++ code generator, model compiler and documentation generator.

Benefits of the methodology found are: 1) modular development of subdetectors packages; 2) allowance of a flexible development strategy, and; 3) easy to manage application evolution. He also concluded that the CASE tool is essential for the project.

W. Lourens [6] of Universiteit Utrecht presented the application of a formal development methodology to online trigger systems for LHC experiments. This is the only contribution in the conference on formal methodologies. Formal methodologies are software engineering techniques that apply mathematical theories to construct a model of a software system. Because methodologies require mathematical skills, there is a great barrier for many software developers to comprehend and to apply them to a practical software system.

The goal of the project presented in the talk is to demonstrate the usefulness of the formal method (VDM++) by applying it to LHC second level trigger systems, where reliability is critical issue. In the presentation, it was demonstrated how an abstract specification can be refined to a more concrete one by using the example of the application to data storage part of the level 2 trigger system.

Methodologies in Off-line Software

A. Schaffer [7] of Orsay presented a progress report and the future plan of a project on the study of OO methodology to the software of the ATLAS experiment.

The project tries to build a systematic and structured approach to construct a software system for LHC experiments. The goals of the project are: 1) to establish a well structured software development environment for LHC; 2) to adapt the use of an OO methodology to HEP environment, and; 3) to be able to recommend, or not, the OO approach to their colleagues. The items studied so far are: 1) training program to learn OO programming (C++ and Eiffel) and methodology; 2) prototypes for testing tools and languages, and for reproductions of a part of the ATLAS reconstruction software, and; 3) evaluation of different methodologies and CASE tools, etc.

M. Maquina [8] of CERN reported a use of the OO methodology in the design and implementation of an interface program which enables to exchange detector models between GEANT and a CAD system. To transfer a detector geometry, the ISO standard called STEP (STandard for Exchange of Product model data) was used. STEP offers a system-independent description of product/commodity information for presenting and exchanging in computer aided systems.

The basic idea to exchange detector models is to express the internal geometrical representation of GEANT in STEP. The STEP representation of GEANT was then converted to a C++ class library. This GEANT class library serves as a base for the interface program to exchange detector models. They used the Object Modeling Technique (OMT) to design this program. The program is tested with GEANT 3.16 and it is also compatible with the version 3.21.

Another contribution which uses OMT was presented by **K. Amako** [9]

of KEK. OMT was applied to reengineer the GEANT program from the object-oriented view. The goal of the project is to study the feasibility of OOA/OOD for the development of a large scale software for future HEP experiments. The project was originally started as a research and development of the SSC software system. The reengineering allows to reuse software assets (both ideas and algorithms) the HEP community accumulated so far.

For the reengineering, the analysis and design steps advocated in OMT are followed rather consistently. The project started with the object analysis. The problem statement was generated from the GEANT manual. The problem statement contains requirements that are general to a detector simulator. However, unique techniques in GEANT like multiple positioning of a volume, the volume division, etc. are included in it. Base on this problem statement, the object model and the dynamic model were constructed. It was emphasized that data structures play an important role in the OO approach.

Tools for Methodologies

S. Fisher [10] of RAL presented a project to construct an OO CASE tool called omo. The motivations of developing this CASE tool are: 1) existing tools are oriented in the concepts they support towards C++; 2) it is advantageous to have control of an OO CASE tools so to be able to try new methodological ideas within minimum delay, and; 3) to make a tool available to the HEP community with free. Main features of omo are; 1) it is at the same time a design tool and a reverse engineering tool; 2) it allows a user to work at different domains like at the code level, at the diagram level, etc., and; 3) it has simple user interface, context sensitive help and safe. Free software components are used for its construction. The extensive usage of the free software reduced drastically the actual codings. He concluded that it is not difficult to build ones own CASE tools if the right components are used.

Another project to construct a CASE tool is given by **B. Rousseau** [11] of CERN. He pointed out the following problems in the currently available CASE tools: 1) the mapping between methodology and language is far to simple; 2) there is no run time support for the management of the association, and; 3) they are not configurable. To overcome these problem the project called OPLA (Object Programming LAboratory) was started. The feature of the project are: 1) toward a seamless OO development environment that let a programmer focus on abstractions; 2) well designed mapping between various OO methodologies and languages; 3) open architecture, and; 4) automatic code generation. Once these are realized, the consistency within the design-to-coding loop is guaranteed, and programmers can focus on abstractions rather than on implementation details.

A prototype of the configurable code generation system is available. It maps the OMT to C++ language.

C. Maidantchik [12] of CERN reported a hypertext tool called HyperDev. This tool integrates various information from the different phases of a software development into a hypertext. For example, in a software development under OMT, a HyperDev user writes problem statements by filling templates the tool

provides. Then the user generates object models, dynamical models and functional models by hand or by using OMTool. Then the user integrates the results into HyperDev. In the implementation phase, the user creates codes and these are again integrated into HyperDev. During the maintenance phase, these documents are updated. Thus the tool can be used to manage the software through its life cycle.

HyperDev is constructed using the NCSA Mosaic tool. This tool is very interesting and useful for those who are developing a software system with a software methodology.

LANGUAGES

Like methodologies, topics of programming languages usually trigger serious debates in our community. This is symbolically shown in the title of another session: "Language Wars - What Will It Be?". In the present session, however, there was no such wars nor debates. The major reason is that all speakers only talked about C++. However, no debate does not mean our community has found a definite answer to the choice of program language. We all well know that FORTRAN, whatever it evolves in future, is the mother tongue for most physicists. We foresee serious disputes especially in future HEP experiment groups. Here one thing we have to remember is that, although a methodology does not necessary specify a particular language, an OO language is a natural choice once the OO approach is introduced into a software design.

There were three talks [13, 14, 15] on the usage of C++ in on-line software systems. All speakers emphasized the merit of C++. For off-line application, there were again three talks [16, 17, 18] on C++. They told that there are numerous merits to use this language in the data analysis stage of HEP experiments. The participants of the present session witnessed the appearance of a young generation, to whom the FORTRAN language is a disgusting one [17].

The following are digests of all contributions belong to languages.

Languages in On-line Software

D. R. Myers [13] of CERN described the advantage and problems encountered in using OOP in the control and data-acquisition system for the upgraded CERN neutrino beam-line. The system consists of a HP workstation running the slow-control package (FactoryLink) and two front-end PCs running LynxOS for beam control and flux monitor.

The code in the PCs has been written in an OO fashion using GNU C++. The system has classes which can be categorized into the following three groups: 1) Specialized classes which correspond to actual beam-line components; 2) Classes which model CAMAC branches, creates, etc., and; 3) Classes which also deal with the computer interface but which model the functionality at an abstract level.

The advantage of using the C++ in the system is that it allows to map physical components in the system (magnets, CAMAC modules, etc.) to objects in the software.

F. Riccardi [14] of CERN reported a project to construct an object-oriented environment aimed to build distributed real time DAQ systems. This environ-

ment is called REMOS. In modern control and DAQ systems, various inter-process communications of variety of bandwidths are included. Because of this, a wise bus bandwidth administration is needed to achieve high performance.

The project introduced a distributed operating system paradigm, enforced in the system by object-oriented approach, encapsulating low level details of inter-process communication. The heart of the DAQ system is the REMOS server which provides a semaphore/queue like synchronization facility and allocation of distributed memory resources, user process communication by RPC, etc., completely unaware of its actual location.

To design/implement this system, the OO approach with C++ was employed. Observed advantages of the OO approach are: 1) easy to build a hierarchy of communication classes; 2) adding more control to them, and; 3) higher reliability of the system. They also measured the system performance.

G. Oleynik [15] of Fermilab presented on dbs, which stands for DART Bootstrapping Services. DART is the data acquisition system for experiments at Fermilab in future. dbs is the first component of run-control for this system. It is a rlogin session multiplexer, which allows a user, running a single program, to start up any number of remote login sessions, feed shell commands to them, and collect the output from the sessions into a single (or multiple) logfiles.

The DART project uses C++. Significant productivity gains were attained by using a low cost commercial class library, tools.h++ from Rogue Wave, using their hash dictionary class to map session name.

Languages in Off-line Software

N. Katayama [16] of Cornell University gave a talk on the track reconstruction program in C++ for the CLEOII vertex detector. This is the first practical application of OOP to the event reconstruction code. In the talk, he explained in detail the classes he and his collaborator designed in the program. He also remarked the spiral approach he employed for the design of the program.

Various features of C++, such as classes and derived classes, constructor and destructor, operator overloading, etc. produce clean and compact codes. He concludes: 1) C++ can be used in a rapid prototyping; 2) Coding is simpler and one faces a fewer bugs; 3) C++ is probably usable for solving real problems in HEP; 4) Designing, implementing and testing classes is an iterative process, and: 5) Once a prototyping is done, one should walk through the current implementation thoroughly. Unlike FORTRAN programs, one can hope to do this and will end up with the code that can be maintained by other people.

S. Misawa [17] of University of California reported the design and implementation of the object-oriented software system for the fixed target experiment at Fermilab E771. Goals of the project are: 1) to utilize OO to create an easy to use system, and; 2) to use OO to simplify the task of developing an entire analysis system to the point of allowing a single person to implement and understand the system.

The system consists of two major subsystems: "event delivery" and "event analysis". He designed various classes as components for these two systems.

The features of these system components are: 1) "LEGO" or building block like functionality; 2) simple object behavior; 3) no common blocks or global variables, and; 4) no initialization routines. Unfortunately, the system architecture was restricted by requirement to be compatible with FORTRAN system. He considers that a significant code simplification/reduction is probably achieved if the FORTRAN system compatibility is sacrificed.

B. Nolty [18] of Caltech reported the project of how to make C++ attractive to FORTRAN users. FARFALLA is a C++ data-management/input- output package he and his collaborator developed. They designed FARFALLA so that FORTRAN people need not be bothered by the language barrier.

Steps they took for the design are; 1) retaining a procedural programming style; 2) protecting users from C++ technicalities (cast, new); 3) spending more time on documentation than coding, and; 4) assuming the reader has no knowledge of C or C++.

The results obtained by this project are: 1) some people ignored it; 2) some dabbled and lost interest, and; 3) a handful of people adopted it and are using it. People who belong to the third category are those who have only written FORTRAN before. They were quite impressed by the speed and ease with which FORTRAN programmers begun to do analysis in C++ with their package.

The reporter emphasized that, to make FORTRAN people attractive to C++, it is important not to make arguments about efficiency, nor to try to influence policy, but to make tools in C++ easy to understand.

GENRIC TOOLS

The shift of the HEP computing environment from the main frame to workstations demands physicists to use new tools for the daily activities of their researches. The tools reported in the sessions are for management of programs, codes and data [20, 21]. Tools for code optimization are also presented [19]. The most of the tools discussed are from the UNIX world. These are commonly used in the UNIX community, though they are relatively new to HEP. There is a barrier for many physicists to switch to these new tools, because old tools are still enough to work with. Also the environment of mixed flavors of workstations leads to a hard life for those who want to transport programs and data among various machines [22]. Other topics discussed are tools in the distributed computing environment [23, 24].

The following are digests of all contributions belong to generic tools.

Code Optimization Tools

S. Yarp [19] of CERN gave a status report on the project to study code optimization of various HEP off-line software programs (simulation, reconstruction, analysis, etc.) which run in CERN's CORE environment. Major reasons he is interested in code optimization are: 1) faster processor will come, but many be they are not quickly enough; 2) such processors may be more sensitive to program structure, and; 3) even multi-processor systems and MPPs may be caught by Amdhal's law, etc.

The way he studies code optimization is to inspect codes and monitor code executions by using "prof", "pixie" on vari-

ous machines. Typical behaviors he observed in HEP programs like GEANT, Aleph reconstruction program are; 1) CPU-time distribution: very flat - no hot spot; 2) ZEBRA forces frequent calculation of indexes, and; 3) Inlining is no universal panacea.

Program/Data Management Tools

E. Berman [20] of Fermilab reported about the distributed development environment for SDSS software. This is the only contribution from non-HEP fields. The SDSS is an international collaboration to survey the sky and make a 3-dimensional map of the Universe. The project runs from 1995 through 2000. They need to have a common development environment, because the collaboration is wide spread geographically and many people at different institutions write the SDSS data processing and analysis codes. To establish this environment, they use tools which are common in the Unix world: 1) RCVS for source code management; 2) UPS for configuration management; 3) UPR for distribution management, 4) GCC for C- compiler; 5) WWW for document management; 6) Tcl for data processing software, and; 7) Tk for GUI.

They have had positive experience with the all tools and each is in daily used by collaboration. The reporter remarked that the reason the experiment could easily adopted new tools and methodologies is that they don't need to worry about a lot of history and existing infrastructure as the HEP community has.

In his second contribution to the session, **N. Katayama** [21] of Cornell University reported his experience of moving from CMZ to CVS for maintaining CLEO software libraries. This shift is motivated by the fact that CMZ doesn't interface well with tools in the UNIX world (make, emacs, etc.).

He concluded that: 1) CVS, gmake, cpp and autoconf together make a nice code development and/or library maintenance environment for a large collaboration; 2) the new system is simpler and more reliable; 3) it is in transit now and to expect to complete moving by the fall; 4) car2cvs provides a means to convert from the old to the new system; 5) for those who like to continue using CMZ, car2cvs will also allow them to submit library changes in CAR format files, and; 6) on a few (non-UNIX) platforms where CVS, gmake or autoconf are not supported, cross compilation can be used.

M. Marquina [22] of CERN, in his second talk, explained the complex world of migrating codes and data, which is consist of: 1) mixed environment of 32-bit and 64-bit machines; 2) big-endian and little -endian, and: 3) various floating point formats (IBM, VAX, IEEE).

He concluded that: 1) FORTRAN source may be made 64-bit compatible on machines providing 64-bit true emulation (CONVEX, SUN), and on machines providing "correct" floating point 64-bit emulation (IBM now, DEC+HP coming); 2) recommended to move all binary data support to 32-bit IEEE (big-endian + little-endian), and; 3) request compiler implementors to provide adequate data transmission in interlanguage calls.

Tools for Distributed Environments

P. Avery [23] of University of Florida presented a project called UFMulti. The

goal of the project is to speed up a single HEP application by distributing it across many CPUs with as little user code as possible. Features of the project are: 1) CPU and I/O are parallelized; 2) similar to Fermilab CPS, but different data passing/communications paradigms; 3) explicit data transfer done with direct socket calls; 4) other interprocess communications done with RCP, and; 5) assume standard UNIX workstations and network protocols.

UFMulti provides a high level toolkit to realize the above features. The most important structural element in UFMulit is NetQueue which provides a buffered data path between reading and writing tasks. He showed the results of tests of NetQueue performance. He also showed the results of tests using real analysis code from the CLEO experiment. The result showed that it can get large speedup factors, depending on amount of CPU vs. I/O per event.

T. Finnern [24] of DESY told about his experience of installing huge number of X-terminals in DESY: now the number is about 500 and it will be soon 600. His talk included administrative issues as well as the user's perspective. He also focused his talk on security aspects, load balancing, software management and conductivity between different systems.

CONCLUSION

We learned from the session that object-oriented methodologies are infiltrating into people who design on-line and off-line software. Those who are using these methodologies are: 1) ZEUS expert system designers; 2) DELPHI on-line event display designers; 3) LHC on-line level 2 trigger designers; 4) GEANT-CAD interface designers, and; 5) LHC and ex-SSC off-line software designers. The methodologies we heard through the session are OMT, Fusion, Objeteering, VDM++ and Booch.

Especially, for the designers of the HERA one-line software systems, these methodologies are practical and indispensable ones. Comparing to on-line, usage of OO methodologies in off-line applications is still in its early stage. However, serious studies are ongoing.

There have been a lot of debates since the SA/SD era whether software methodologies really bring a systematic approach for the development of HEP software systems. What is different from the SA/SD era and now is that a lot of CASE tools which have direct couplings to methodologies are appearing. These are HEP originated tools and also ones from vendors. These may potentially bring a completely different way of programming to our community. However, what we have to remember here is that just using a tool does not give a quality software [3].

Concerning new program languages, on-line and off-line people both emphasized advantages of using C++ in their software developments. There was no argument to against to them. The participants heard OO jargon everywhere in the session. However, what we have to remember is that most physicists are using FORTRAN in their daily physics activities and it is their native language.

REFERENCES

1. R. K. Bock, "A Summary of the Oxford Conference on Computing in High Energy Physics", CHEP89, Oxford, Conference Proceedings, Comp.

Phys. Comm. 57 (1989) 1.

2. J. Knobloch, "Reality of Software Engineering in High Energy Physics", CHEP91, Tsukuba, Conference Proceedings., Universal Academy Press, Tokyo.
3. S. Loken, "Software Engineering: What do experiments need?", CHEP92, Anecy, Conference Proceedings, CERN 92-07.
4. M. Flasinski, et al, "Use of Object-Oriented Software Methodology to Design of Expert/Control Systems for HEP: the ZEUS Expert System Experience", these proceedings.
5. S. Dû and J. Laugier, "Object Structure Modelling in the DELPHI Online Event Display", these proceedings.
6. W. Lourens, et al., "The formal development method VDM++ and its application to second level trigger systems for LHC experiments", these proceedings.
7. A. Schaffer, et al., "Object Oriented Approach for Software Development for LHC experiments", these proceedings.
8. M. Maquina, et al., "An Object-oriented Technique for Detector Models Used by a Simulation Program", these proceedings.
9. K. Amako, et al., "Object Oriented Analysis and Design of GEANT Based Detector Simulator", these proceedings.
10. S. Fisher and D.J. Candlin, "Omo - A Tk/tcl based object modelling tool to support Eiffel", these proceedings.
11. B. Rousseau et al., "A Configurable Code Generator for OO Methodologies", these proceedings.
12. C. Maidantchik and M Isaac, "HyperDev: Hypertext Tool to Support O-O Software Development", these proceedings.
13. D.R. Myers, et al., "Use of Object-Oriented Techniques in Beam-Line Control System", these proceedings.
14. F. Riccardi, et al., "REMOS: A Portable Object Oriented Environment for Multiprocessor Real Time Applications", these proceedings.
15. G. Oleynik, et al., "dbs - DART Bootstrapping Services", these proceedings.
16. N. Katayama and M. Smyth, "A Tracker in C++", these proceedings.
17. S. Misawa, "The E771 Object Oriented Software System", these proceedings.
18. B. Nolty and C. Walter, "FARFALLA: C++ Data Management That Event FORTRAN People Can Love", these proceedings.
19. S. Jarp, "Quo Vadis Code Optimisation in HEP??", these proceedings.
20. E. Berman,et al., "The Distributed Development Environment for SDSS Software", these proceedings.
21. N. Katayama, "Maintaining Software Libraries at CLEO - Moving from CMZ to CVS", these proceedings.
22. M. Marquina, "Towards a 64-bit Version of Scientific Libraries", these proceedings.
23. P. Avery, et al., "The UFMulti Project", these proceedings.
24. T. Finnern, "Experience at DESY - R2", these proceedings.

Experience building object-oriented systems*

Alan Breakstone
University of Hawaii, Honolulu, HI 96822 USA

ABSTRACT

I describe the experience gained by the group of developers of the Gismo project, a relatively large software package to do detector simulation and event reconstruction using object-oriented techniques.

1. INTRODUCTION

Most software for high energy physics is written in FORTRAN, a procedural computer language. Over the past several years computer scientists have been developing languages which use a different way of addressing problems in computation using object-oriented techniques. A group of physicists and programmers began a prototype project at CERN during the summer of 1990 to study if detector simulation and particle transport could benefit from object-oriented design [1]. This project was christened Gismo, which stands for Graphical Interface for Simulation and Monte Carlo with Objects.

The success of this prototype project, written using the Objective-C language for NeXT workstations, culminated in a larger group of physicists and programmers becoming interested in expanding the project to be a more general and useful tool, not only for high energy physics, but also for other fields of physics and for medical applications where radiation transport is simulated, such as radiation dose therapy and positron emission tomography. Two workshops have been held, at SLAC in July, 1991, and at the University of Florida in January, 1992 to evaluate the prototype and redesign Gismo. A list of members of the Gismo collaboration is given in Table 1.

*Work supported by Department of Energy grant number DE-FG 03-94ER40833.

Table 1: Members of the Gismo Collaboration

Authors of Code and/or Code System:	
Bill Atwood	SLAC
Alan Breakstone	University of Hawaii
David Britton	McGill University
Toby Burnett	University of Washington
Graham Cross	McGill University
David Myers	CERN
Paul Rensing	SLAC/CERN
Gary Word	Rutgers/SSC Lab/Citicorp
Consultants, Workshop Participants, and Users:	
Ejaz Ahmad	University of Florida
Dave Aston	SLAC
Christian Arnault	Orsay
Paul Avery	University of Florida
Alex Bielajew	Inst. of Radiation Standards
Chandra Chegirddy	University of Florida
Ruediger Gross-Hardt	University of Bonn
Walter Innes	SLAC
Paul Kunz	SLAC
Gary McGrath	Univ. of California, Irvine
Paolo Palazzi	CERN
Tom Pavel	SLAC
Ernest Prabhakar	Caltech
Cam Sanders	University of Wisconsin
Art Snyder	SLAC
Mick Storr	CERN
Dennis Weygand	Brookhaven National Lab
Saul Youssef	Florida State University

The goals of the project are to allow detector simulation and event reconstruction in one program, reusing many of the same objects, to have a platform-independent kernel written in C++, to run the kernel on Unix platforms, to have an application with a Graphical User

Interface (GUI) for rapid prototyping of simple detector elements, producing one-event displays, and visual methods for debugging the kernel, and to have user "hooks" for customization, mainly using the inheritance feature of object-oriented languages.

By early 1993 the kernel and graphical user interface (for NeXT workstations) were well enough developed for serious simulations, mostly by Gismo developers. Gismo was first released for general use in August, 1993. The most recent version as of this writing is the 0.4.1 version, installed on March 22, 1994 in the anonymous ftp area of SLAC.

2. EXAMPLES FROM GISMO

To better illustrate the Gismo project, I have selected a few examples of simulations done for various projects. Gismo allows one to define complex detectors, such as a prototype of the SLAC B-factory detector shown in Figure 1. One can create particles and propagate them through the detector. Particles can interact in a continuous way with ionization energy loss and multiple Coulomb scattering or in a discrete way with electromagnetic interactions using EGS4 [2], illustrated in Figure 2, and hadronic interactions using Gheisha [3], illustrated in Figure 3. Gismo also can han-

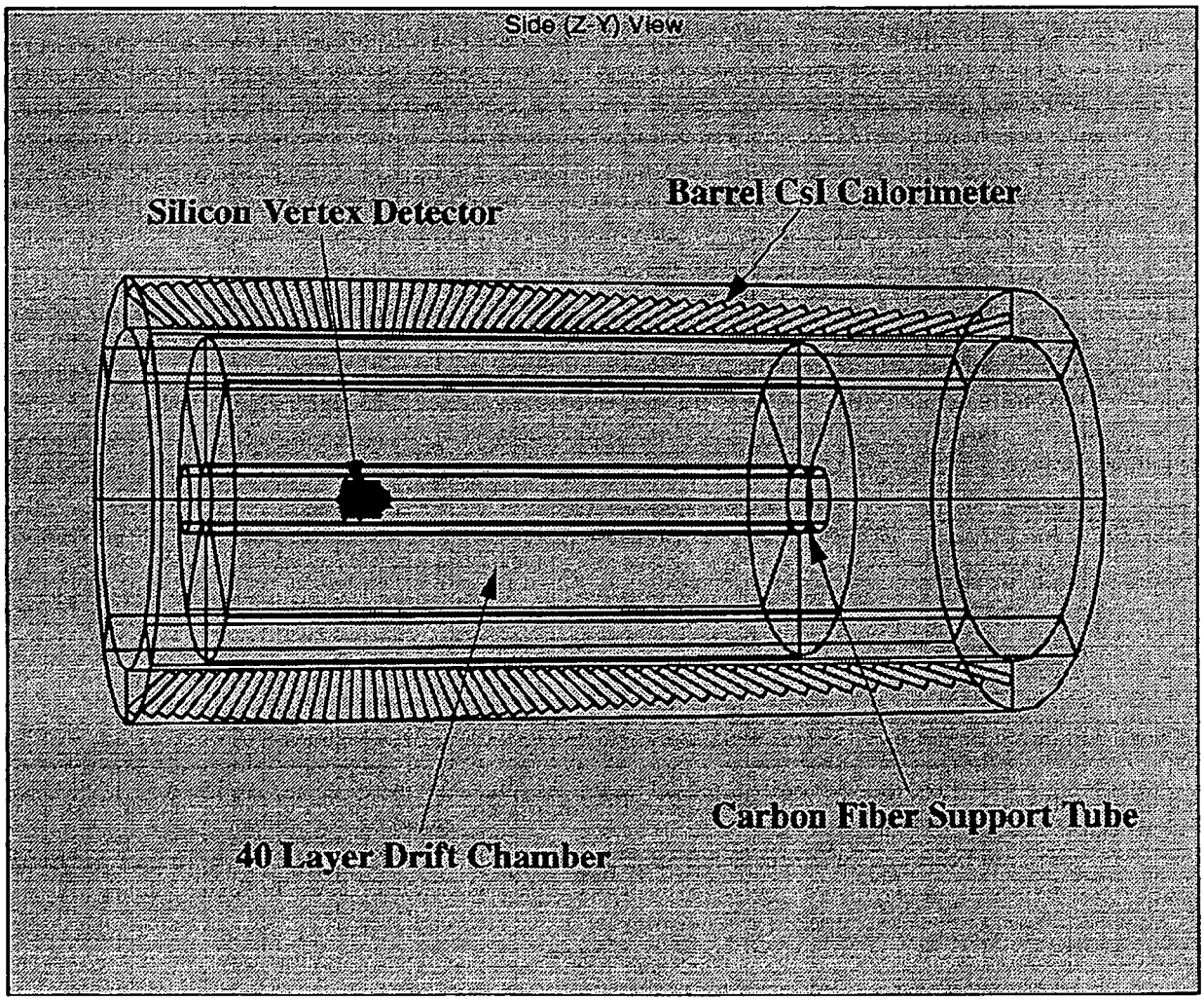


Figure 1. A SLAC B-factory "long barrel" detector. The four major components are labeled. Not shown but in the simulation is the beryllium beam pipe. The detector is immersed in a 1 T magnetic field along the beam axis.

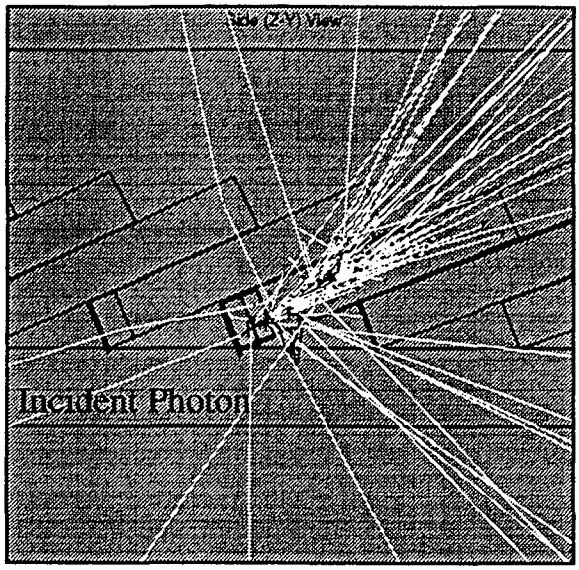


Figure 2. The electromagnetic shower produced by a 1 GeV photon incident on a CsI block in the "long barrel" B-factory detector. The shower cutoff energy is 1 MeV for all particles. Photons are shown in white, electrons and positrons in black. The calorimeter response is also shown.

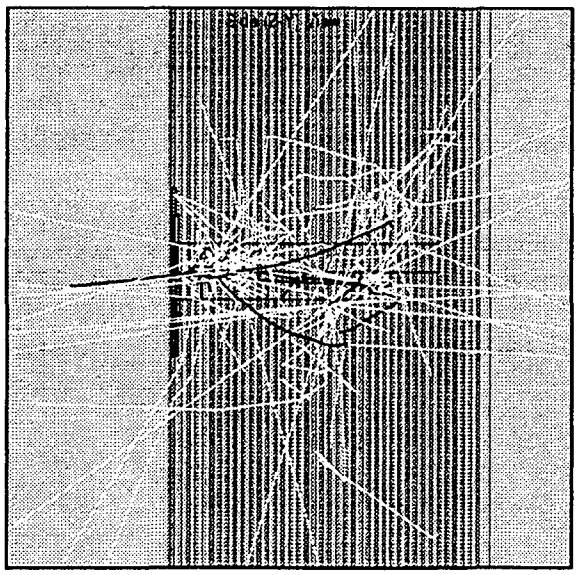


Figure 3. The hadronic shower produced by a 10 GeV π^+ incident on a uranium-liquid argon sampling calorimeter in a 1 T magnetic field. Neutrals (mostly neutrons) are shown in white, charged particles in black. The calorimeter response is indicated by the black towers.

dle particle decays, such as the $\Upsilon(4s)$ decay shown in Figure 4.

Gismo can also create and propagate optical photons. It can simulate detector responses, as is shown in the figures. It also can output Hippo [4] ntuples for histogramming purposes.

Currently the Gismo kernel consists of nearly 50,000 lines of code. It is being used for sev-

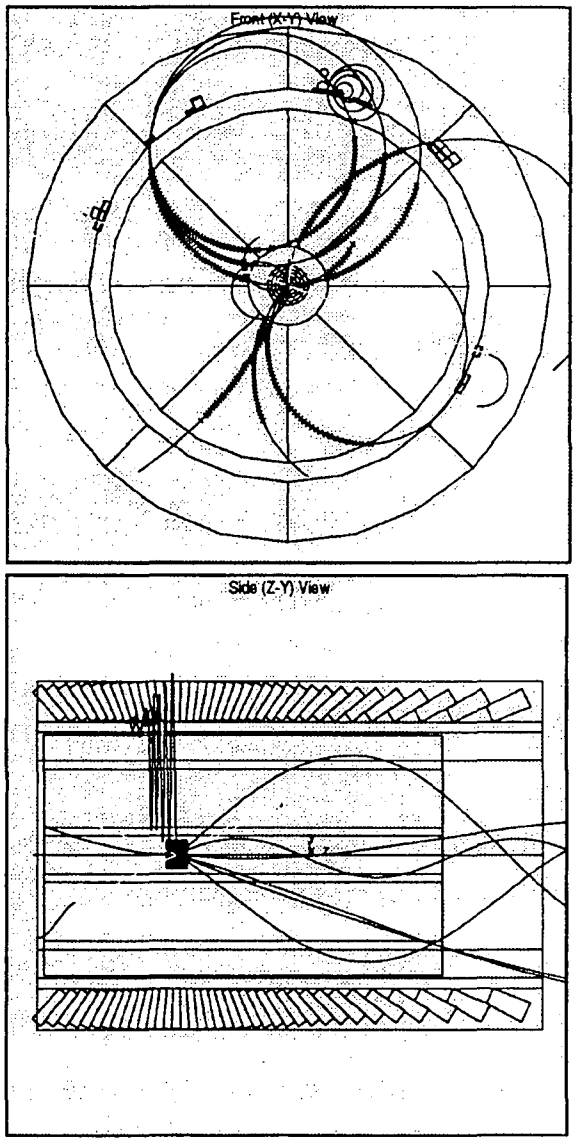


Figure 4. End and side views of an $\Upsilon(4s)$ decay in a variant of the "long barrel" B-factory detector. The response of the various detector elements is shown in the end view only.

eral projects, including the Gamma-ray Large Area Silicon Telescope (GLAST) project by Bill Atwood, the GEM calorimeter by Toby Burnett, the BaBar B-Factory detector by David Aston and Gary McGrath, a scintillating fiber detector by Ruediger Gross-Hardt, beam-line elements by Nick Walker, and Beijing Spectrometer (BES) time-of-flight counters (using optical photons) and the BELLE B-factory detector by this author.

3. USE OF OOP IN GISMO

An example of the way in which Gismo uses object-oriented programming techniques is illustrated by the distanceToLeave function in the Volume class, shown below. This code is used to calculate the intersections of trajectories with three-dimensional geometrical objects. Each Volume "has a" list of Surfaces which define it. The Surface class inherits from a list link class so its subclasses can be put in lists. Note that both Volume and Surface are virtual classes since the user only instantiates their subclasses.

The code loops through each Surface in the list to find the closest one. Polymorphism is used in several places. The appropriate distanceToLeaveSurface function in the trajectory classes (Ray and Helix) are called depending on whether the reference to the object r is a Ray or Helix. In these trajectory classes, the distanceToLeaveSurface function calls the appropriate distanceAlongRay or distanceAlongHelix functions in the Surface subclasses, depending on whether the Surface is a Plane, Cylinder, etc. They in turn use function overloading to find out if a potential intersection point is within the boundary of finite-sized figures (such as Rectangles) which bound a Volume.

This use of function overloading and the virtual function table of C++ to implement polymorphism is used extensively in Gismo. Note that one can add new types of trajectories or

```
double Volume::distanceToLeave( const
        Ray& r, ThreeVec& p, const Surface
        *&sf ) const
{
// Distance along the Ray r or any of its
// derived classes to the nearest Surface to
// leave the Volume.
// Gives the point p of intercept and a pointer
// to the Surface which is hit.
// If there is no intersection, the Surface
// pointer is set to 0.
// This point is not guaranteed to be inside
// (or outside) the Volume.
        double d = 0.0, t = FLT_MAX;
        ThreeVec temp ( t, t, t );
        p = temp;
        sf = 0;
        Surface *s =
            (Surface *)surface_list->first();
// Loop over each Surface in Volume's list,
// find the distance to that Surface, and pick
// the smallest positive (or zero) distance.
// The distanceToLeaveSurface member
// function of Ray will make sure that the Ray
// or its derived class is leaving the Surface
// (defined by the direction of the normal to
// the Surface) at the point of intersection.
        while ( s ) {
            d = r.distanceToLeaveSurface( s,
                temp );
            if ( ( t > d ) && ( d >= 0.0 ) ) {
                t = d;
                p = temp;
                sf = s;
            }
            s = (Surface *)s->next();
        }
        return t;
}
```

new types of surfaces without modifying this code, illustrating the reusability of object-oriented code.

4. LESSONS LEARNED

In designing and implementing the Gismo project, the developers learned a number of

lessons concerning object-oriented design and implementation using the C++ language. We found that the spiral design cycle is the appropriate way to deal with a project of this size and complexity. The main difficulty is to get the correct abstraction. Often this is not realized at the design stage and problems only show up during implementation and testing of the code. Thus it made sense to start writing and testing prototype code early in the design cycle. We often rewrote vast sections of the code, but found that this was relatively easy to do due to the good encapsulation provided by object-oriented design.

Another lesson we learned was to divide the project into manageable pieces with each piece as independent of the others as possible. This was particularly important for this project since the developers are geographically widely scattered. Since we are a small group, communication was rather easy. We avoided the division between code designers and code implementors which is often done in industry.

We found that C++ is a good language to use for an object-oriented program. The compilers catch many potential bugs at compile time. It was easy to implement object-oriented features such as inheritance and polymorphism. One problem for porting the code is the lack of an ANSI standard for C++. This hopefully will be resolved in the not too distant future.

Early in the design phase we studied various base class libraries for suitability for Gismo. One serious constraint, both for base class libraries and for design tools, was cost. In particular we did not want to base Gismo on any class library which would require Gismo users to buy some software license from a commercial vendor, since this would severely limit the number of physicists interested in using Gismo. The developers also were very restricted in the amount of money we could spend on design tools, so we did not base our designs on the use of any commercial product.

One point we found crucial was that one needs a makefile system as well as the source code. A couple of the Gismo developers spent most of their efforts on this. We eventually settled on a makefile scheme based on imake from the X11 consortium.

From the very beginning we have stressed the toolkit aspect of Gismo. Our users (mostly particle physicists) are very sophisticated. They will want to customize the simulations to their detectors and add functionality for things which the developers could not anticipate. In addition users may want only parts of Gismo, so should be able to easily choose what they want.

We learned that the key to object-oriented design is a good interface, both for users and, within the project, for developers. The implementation can easily change without affecting users. One must spend a lot of time to develop an interface which will not change. This is particularly true for constructors. For the Gismo project it was perhaps fortunate that we did not develop a large user base this early in its development, since a large user base severely constrains the developers by forcing them to freeze the interface for the users, perhaps before they've found the correct abstraction for the problem at hand.

5. CURRENT STATUS AND FUTURE

The Gismo kernel is available via anonymous ftp from SLAC (ftp ftp.slac.stanford.edu, look in pub/sources). The current production version is 0.4.1. One can compile it using GNU gcc version 2.3.1 or later. One needs X11R4 or later for the imake makefile system. Along with the kernel is a sample test job called GismoTest which uses Hippo. Libraries based on this version have been installed at SLAC for three platforms: IBM RS/6000 AIX, NeXT, and Sun4 workstations.

This author has also ported Gismo to a DEC-Station with Ultrix at the University of Hawaii. The Gismo project is now beginning to attract real users, not just the developers.

In the future Gismo plans to use the freely available CLHEP base class library for lists, vectors, etc. We will continue our work on event and track reconstruction. There are a number of planned improvements to physics processes, such as a better treatment of multiple Coulomb scattering and adding non-constant magnetic fields. We want to provide an interface to event generators, such as MC++ [5]. There is much work to be done on detector and event I/O. The first step in this direction has been to provide capability to read generated events in STDHEP format. Currently there is only a NeXT GUI. We would like to port this to X-windows platforms. In the more distant future we would like to improve the Gismo graphics to include 3-dimensional rendering. Perhaps someday Gismo will have some sort of CAD interface. Also the documentation is woefully inadequate. Hopefully users can help with many of these projects, since the developers' time is very limited.

6. CONCLUSIONS

Gismo is a large and successful software project based on object-oriented programming techniques. It is in a continual state of redesign and refinement. The Gismo developers have found that the spiral design model is the correct one for this task. Based on our experience, we discourage a top-down approach to program design for projects of this magnitude. We are very much encouraged by increased interest in object-oriented software by the high energy physics community.

REFERENCES:

[1] W. B. Atwood, *et al.*, International Journal of Modern Physics C **3**, 459 (1992).

[2] W. R. Nelson, *et al.*, SLAC-0265 (1985).

[3] R. Brun, *et al.*, CERN-DD/EE/84-1 (1987). More recently, see GEANT User's Guide, GEANT 3.15, Revised July 13, 1992.

[4] Michael F. Gravina, *et al.*, SLAC-PUB-5921 (1992).

[5] L. Lonnblad and A. Nilsson, Computer Physics Communications **71**, 1 (1992).

Session 1

Triggering, Data Acquisition, Online, and Control Systems

A BIT-SERIAL FIRST-LEVEL CALORIMETER TRIGGER FOR LHC DETECTORS

C. Bohm, X. Zhao, G. Appelquist, M. Engström
S. Hellman, S-O. Holmgren, E. Johansson, N. Yamdagni
Department of Physics, University of Stockholm, S-113 85 Stockholm, Sweden

ABSTRACT

A first-level calorimeter trigger design, implemented as a farm of local bit-serial systolic arrays, is presented. The massive bit-serial operation can achieve higher processing throughput and more compact designs than conventional bit-parallel data representation. The construction is based on high speed optical fiber data transmissions, Application Specific Integrated Circuits (ASICs) and multi-chip modules (MCMs) packaging technologies.

INTRODUCTION

The primary task for first level triggers in high energy physics detectors is to reduce the large flow of primary data by eliminating clearly uninteresting events. This is especially important for detectors at the Large Hadron Collider (LHC) where the size of the data rate is without precedence. In this case the reduction will be achieved by continuously monitoring global but low resolution information from the calorimeter and muon subdetectors in search for signals that are characteristic for desired events. Reduction rates between 10^3 and 10^4 are anticipated in the first level.

If high granularity calorimeter data are merged into larger 0.1x0.1 trigger cells measured in the η-φ space, i.e. in pseudo rapidity (roughly axial) and azimuthal angle, 60x60 calorimeter trigger cells will cover the calorimeter surface within ± 3 units of η. If the proton bunch collision frequency is 40 MHz, data from 7200 calorimeter trigger cells will arrive every 25 ns. The additional factor 2 is picked up because two calorimeters are involved, a hadron and an electro-magnetic calorimeter, one outside the other. The large number of high data rate inputs suggests using optical fibers rather than coaxial cables for transferring trigger data from the detector to the trigger processor.

The trigger processor is designed to identify electrons by looking for isolated energy depositions (clusters) in the electromagnetic calorimeter which have a small energy leakage into the surrounding hadron calorimeter. It will use cluster finding algorithms [1] to identify jets and compute missing transverse energy to deduce the presence of neutrinos.

These operations are best performed by pipelined processing of all trigger cells in parallel. The electron identification is achieved by first forming trigger sums with neighbours to the right and neighbours below. The maximum of these two sums is then used to estimate the energy of an assumed particle that has deposited most of its momentum in the given cell. The isolation is established by making certain that the particle energy is much larger than the total energy in an environment not including the primary deposition. The leakage is treated similarly, by requiring that the energy deposition into the hadron calorimeter should be sufficiently small. The chosen algorithm demands access to a 4x4 trigger cell environment from both calorimeters for each trigger cell evaluation.

Possible jet-algorithms can be based on the merging of blocks of say 3x3 trigger cells into jet super cells, counting the number of clusters above a certain threshold. Other algorithms may be based on the addition of all trigger cells within 4x4 blocks to obtain a sliding window which will move over the entire surface, counting the number of local maxima.

An additional purpose for the first level trigger is to provide the second level trigger with coordinates identifying areas of special interest within data from selected events. This is to allow the second level trigger to concentrate on limited regions of the data which may then be examined in full detail. This process will lead to a further elimination of about 99% of the data remaining from level one.

A BIT-SERIAL IMPLEMENTATION OF A FIRST-LEVEL TRIGGER PROCESSOR

The structure of the trigger algorithms have strong implications on the choice of processing architecture. The fact that the calculation performed for each trigger cell requires data from an environment, implies that if all trigger cells are shared among a limited number of computational units, these units will have to exchange boundary information to have access to sufficient data to treat trigger cells along their borders. The fraction of information that will have to be exchanged decreases with increased size of the processing units. Thus the larger the computational unit, the less interconnections between the units. Since the natural choice is to use ASICs (Application Specific Integrated Circuits) to implement calculation units, the number of interconnections will also be bounded from above by the maximum number of inputs/output connections allowed in a package using today's technology.

The information exchange between the computational units can also be regarded as a splitting (fan-out) of the data paths from border trigger cells so that they can reach several units. The advantage of large processing units is then demonstrated by the fact that the average fan-out required will decrease with increased unit size

Comp. unit size	Input data size	I/O-efficiency (output/input)	Av. fan-out
1X1	4x4	6%	16
4X4	7x7	33%	3.1
6X6	9x9	44%	2.3

Table showing the advantage of large computational units

Bit-serial digital signal processing techniques are characterized by simplicity, compactness, efficiency and short latencies, especially when performing operations like addition. The simplicity allows higher clock rates than is possible in parallel designs (no problems with carry propagation). The fact that the trigger algorithms mostly consist of simple additions followed by comparisons, together with the throughput and latency requirements, imply that choosing bit-serial architectures wherever possible is advantageous.

With algorithms based on 4x4 environments a single bit-serial trigger processor for 6x6 trigger cells will require 162 inputs (9x9x2). This number is large but acceptable. 162x8, however, as is required by a more strait forward bit-parallel processor, is clearly not realistic with present technology. Time multiplexing will help in this case, though. Large computational unit are also preferable when evaluating and merging partial results from each point operation. A larger unit implies that less merging needs to be performed outside the unit.

A tentative design was made on basis of above considerations but with the earlier assumed bunch crossing frequency 67 MHz [3]. A clock rate of 270 MHz was then considered to bit-serially process input data with 8 bits precision. The feasibility of reaching and exceeding such performances with CMOS circuits using available technology has been demonstrated in many applications [4]. However, the 4-fold increased clock rate is not sufficient to reach the required processing rate in one bit-serial processor alone. The residual increase is therefore achieved by relying on the combined capacity of a farm of 3 bit-serial processors. Even with this 3-fold increase in total chip area the net surface gain using bit-serial operations is considerable.

In a later modified design, adapted to the presently stated bunch crossing frequency value of 40 MHz [5], the possibility of entering the trigger cell data via optical fibers directly into a Multi Chip Module (MCM) was considered. In this design the farm concept was implemented on the MCM level with a small ASIC farm (with still fewer members e.g. 2) to implement processing units in MCMs. Each operating with clock rate of 320 MHz. However, further increasing the clock rate to 400 MHz removes the need for a farm entirely.

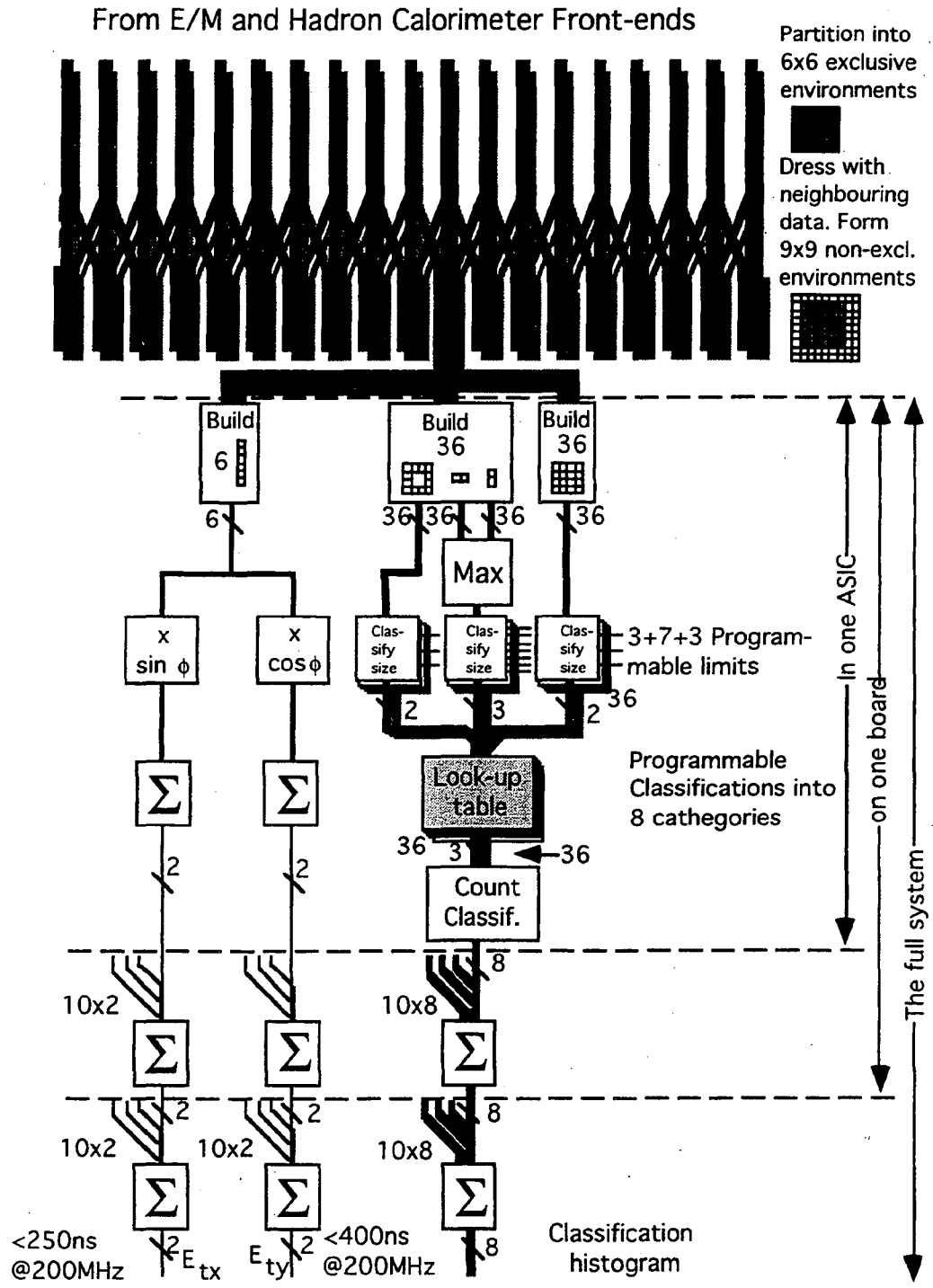


Figure 1. The electron detection and E_T evaluation computation flow. All blocks except look-up table are based on bit-serial implementation.

The designs have been described and simulated down to a fairly detailed Register-Transfer level using VHDL. The computation flow of the electron identification and transverse energy evaluation is shown in Fig 1. Here mutually exclusive 6x6 environments are expanded by sharing information from neighbours (into 9x9), different sums are then formed for each trigger cell and classified according to value. The total classification of all energies are expressed in a 7-bit code, one for each trigger cell.

This code is then translated via a look-up table into one of 8 categories (corresponding to 8 different physics features). The number of occurrences of each feature are counted for each 6x6 environment (i.e. each ASIC), expressed in a bit-serial form then merged via bit-serial addition on the board level (about 10 ASICs) and finally on the full system level. The result is reported to the central first level trigger processor.

The x and y components of the transverse energy are also evaluated for all trigger cells belonging to the ASIC and summed. This involves a bit-serial multiplication with sine ϕ and cosine ϕ. The two results are then sent out in bit-serial form as well, to allow easy merger with data from other units. The final result is an estimator of the missing transverse energy.

The possibility to implement the suggested design depends on the feasibility of critical implementation details. An important task is therefore to resolve these uncertainties using references to literature or data from experiments. Thus will, for example, the feasibility of the required large number of fiber inputs directly into a MCM (162 fibers at .4 Mb/s) be studied in a demonstrator. Preliminary results are encouraging. The performance of high speed bit-serial arithmetic operations and interference problems caused by the massive high-speed processing on the chip will also be studied experimentally.

CONCLUSIONS

The suggested bit-serial first-level calorimeter trigger will achieve high throughput, compactness and short latencies. The major part of the computations will be performed in one ASIC minimizing the need for connectors, thus allowing high reliability. Data from different ASICs and different modules are merged via a simple bit-serial adder tree ASIC. The design also allows expansion in different directions to improve the trigger efficiency.

ACKNOWLEDGMENTS

The authors would like to express their gratitude for valuable contributions concerning high speed electronic circuits from C. Svensson, H. Johansson and J. Yuan at Linkoping University. P. Bodö, H. Hentzell, M. Tuber have provided expertize on MCMs. U. Westergren, M. Mohktari from Industrial Microelectronics Center in Stockholm contributed with expertize on high speed electronics and optocommunication.

The authors would also like to acknowledge many useful suggestions from the members of the RD-27 collaboration, within which the LHC first-level trigger research and development project has been carried out.

REFERENCES

1. N.Ellis, J.Garvey, *CERN 90-10*, vol3, p. 80 (1990).
2. RD-27 Status Report, *CERN/DRDC 93-32*, (1993).
3. C. Bohm et. al. *CERN RD-27 note 6*, (1993).
4. J. Yuan and C. Svensson, *IEEE Spectrum*, p. 52 (Feb. 1991).
5. C. Bohm et. al. A bit-serial first-level calorimeter trigger for an LHC detector, accepted for publication in IEEE Trans. Nucl. Sci.

Results of On-Line Tests of the ENABLE Prototype, a 2nd Level Trigger Processor for the TRT of ATLAS/LHC¶

K.-H. Noffz[1)], R. Zoz[1,2)], A. Kugel[1)], F. Klefenz[1)], R. Männer[1,2)]
[1)] Lehrstuhl für Informatik V, Universität Mannheim, Mannheim, Germany
[2)] Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Heidelberg

Abstract

The Enable Machine is a systolic 2nd level trigger processor for the transition radiation tracker (TRT) of ATLAS/LHC. The task of the processor is to find the best candidate for a lepton track in a high background of pions according to the EAST benchmark algorithm [2] in less than 10 μs. As described earlier [1, 2] this is done in three steps. First all interesting tracks are histogrammed by accumulating for each track the coincidences between the track mask and the region-of-interest (RoI). Next the best defined track is identified. Eventually this track is classified as e or π.

A prototype has been developed and tested within the EAST/RD-11 collaboration at CERN. It operates at 50 MHz and finds up to 400 tracks in less than 10 μs. It is assembled of an interface board and one or more histogrammer boards. The modular design makes the Enable Machine easily scalable. The histogrammer units are systolic arrays consisting of a matrix of 36 field programmable gate arrays. Through this it is possible to optimize the trigger algorithm, to adapt it to a changed detector setup, and even to implement completely new algorithms.

For the beam tests in autumn 1993 at CERN the overall functionality within the detector environment could be shown. We were able to link successfully the Enable prototype to the detector raw data stream as well as to the data acquisition system.

1. Introduction

Second level trigger algorithms often are quite similar. Normally they require moderately complex pattern recognition combined with a very high data rate while they show a high degree of parallelism.

Systolic arrays consisting of a matrix of field programmable gate arrays (FPGAs) are well suited to such problems. Each chip consists internally of up to 1000 processing elements with programmable interconnects, and has a flip-flop toggle rate of 100 MHz. Thus the use of reprogrammable logic provides great flexibility and high speed.

This paper presents a high speed pattern recognition processor for 2nd level triggering, the Enable Machine, which exploits these advantages of FPGAs. We describe the architecture and the implementation of a prototype and present first results from data taken during the beam period in autumn 1993.

2. Algorithm

As described elsewhere [3] the 2nd level trigger task at the TRT of ATLAS/LHC is the following. Every 10 μs the 1st level trigger, based on calorimeters, selects subsets of the whole detector image, RoIs. They are analyzed by the 2nd level trigger. In the RoI particle tracks appear as straight lines of different slope according to their transversal momentum p_t. From these tracks the best candidate for a lepton track has to be found and its transversal momentum has to be evaluated.

The Enable Machine finds tracks by histogramming the coincidence of RoI image data and predefined search roads. The maximal slope of search roads defines the p_t cut. Two histograms are accumulated for the same RoI using two different (low and high) recording thresholds. The best lepton track candidate is searched in the "low" RoI image, and together with the corresponding track data in the "high" RoI image lepton tracks can be identified via an appropriate weighting function.

3. Implementation

The main processing unit of the Enable machine consists of a matrix of 36 FPGAs of type Xilinx XC3190. The use of FPGAs has two important advantages: First, it provides high speed pro-

¶ This work has been supported by the German Minister for Research and Technology under grant 056MA52P, by the Gesellschaft für Schwerionenforschung Darmstadt (GSI) under grant HD Spe C, and by the RD-11- and RD-6 collaborations at CERN

cessing since the algorithm is executed directly in hardware. Second, it provides high flexibility since the FPGAs can be reconfigured, i.e., different algorithms can simply be downloaded. For additional flexibility the search roads are defined by the contents of a lookup table (called activation RAM). The weighting function is also implemented in another lookup table.

The inherent parallelism of the TRT algorithm leads directly to the massively parallel and pipelined architecture of the Enable machine. RoI image columns are handled in parallel, different slopes in a pipeline (Fig. 1).

The Enable machine runs with a clock frequency of 50 MHz that allows a maximal data throughput of 400 MByte/s. Depending on the RoI image dimensions the processing time is between 1µs and 6µs (z=1...256). One FPGA matrix allows the parallel search for 400 patterns.

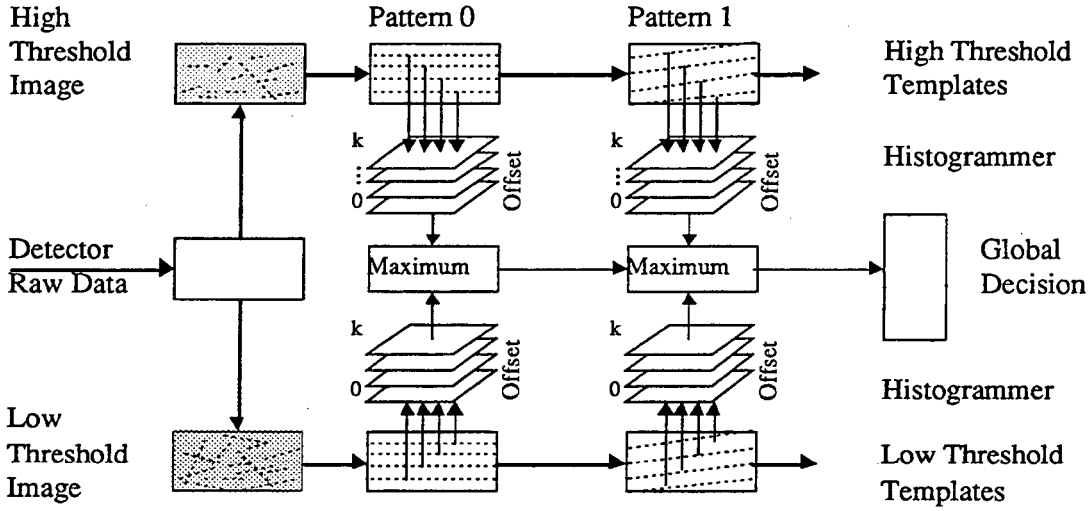


Figure 1

4. Hardware Setup

The prototype Enable Machine is assembled of an interface board and one or more histogrammer boards (one for the test setup).

Event data are received by two HIPPI channels on the interface board and are broadcasted to the histogrammer units. In the histogrammer units the event is processed and transmitted back to the interface board. Here the results from each histogrammer board are combined to the global trigger decision of the Enable Machine. This modular concept makes the Enable Machine easily scalable from 400 to 8000 patterns where each pattern corresponds to one possible lepton track.

Event data, i.e., RoI images of size 256×32×2 bit, are received by the Enable Machine's interface board via two HIPPI lines that provide a maximal input data rate of 200 MB/s. The HIPPI destination interface assembled on the interface board has been developed with Dubna/Russia in a joint project.

A histogrammer unit of the Enable Machine mainly consists of a matrix of 36 gate array devices of type Xilinx XC3190 and over 4 MBytes of synchronous SRAM for pattern definition and lookup tables. For debugging and monitoring purposes the full histogram content can be read out from the histogrammer board via VME. Like for the interface board it is also possible to download events from the VME host and to process them for testing purposes.

The whole system houses in a VME crate. The board size is 40×38 cm^2 (6 height units). The Enable Machine is controlled and programmed by a commercial VME module containing a µSPARC processor. It is used as the bus master whereas the Enable boards are VME slaves only (Fig. 2). This is sufficient for trigger purposes.

5 System Embedding

For the beam tests in autumn 1993 at CERN the Enable Machine was embedded in a special test environment of the TRT. The detector data were input via the TRT router, a special system that copies RoI windows out of the detector data stream and send them to the Enable machine via 2 HIPPI links.

The Enable VME crate was linked to the data acquisition crate via a VIC bus module containing 4 MBytes of memory. During bursts the memory was filled by Enable. In the time gap between bursts the data acquisition system read this buffer.

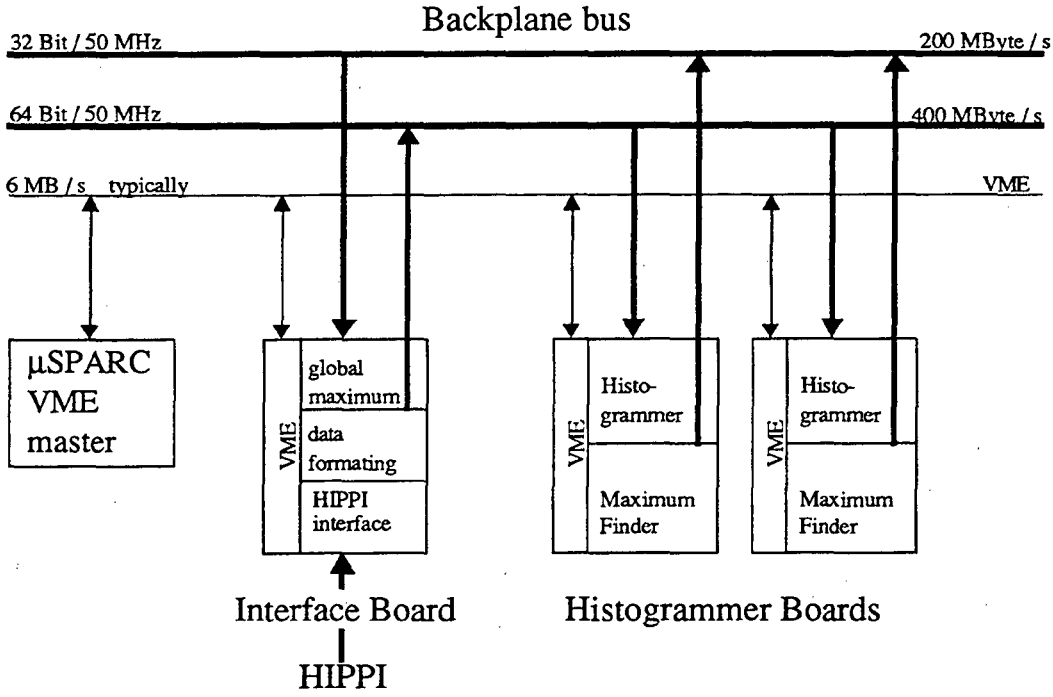


Figure 2

6 Results from the beam period 1993

The Enable Machine is the first 2nd level trigger processor developed for ATLAS which was tested under beam conditions. For the beam tests in autumn 1993 at CERN it was intended to show the overall functionality within the detector environment. This comprised data input handling, correct histogramming of events, and connection to the data acquisition system.

During the run the Enable Machine could be successfully integrated in the test detector data acquisition system and 10^5 events could be recorded on tape. It was possible to forward a continuos event data stream from the detector through the router to the Enable Machine, although several problems were encountered. The main problems arose because error handling between the router and Enable was not thoroughly defined, and due to synchronization faults between the two HIPPI lines. Offline analysis verified that apart from some minor bugs both the interface board and the histogrammer board produced correct results.

It was possible to identify events with the corresponding raw event data which proved that the transfer from the detector through the router to the Enable Machine worked correctly. So the principle functionality of the Enable Machine could be shown.

7 Conclusions and Outlook

The pattern recognition task for most 2nd level trigger problems are of moderate complexity, offer a high degree of parallelism, and have to deal with very high data rates. For these kinds of requirements a FPGA based processor like the Enable machine with its massively parallel architecture is very well adapted. The execution in hardware provides additional speed.

But apart from traditional special-purpose processors designed for one certain problem, reconfigurable FPGAs offer a new quality. Their massive use in form of a FPGA matrix allows the reconfiguration of the whole processor. Reconfiguration can be used to implement different trigger algorithms on a single machine. We are investigating the implementation of several LHC 2nd level trigger algorithms on Enable in order to extract an optimal architecture for a general purpose FPGA matrix.

It is intended to build such a processor with a more generalized architecture. That general purpose machine should be able to solve many LHC 2nd trigger problems. A prototype should be available end of 1995.

[1] Klefenz F., Zoz R., Noffz K.-H., Männer R.: The ENABLE Machine - A Systolic Second Level Trigger Processor for Track Finding; Proc. Comp. in High Energy Physics, Annecy, France; CERN Rep. 92-07 (1992) 799-802

[2] Badier J., Bock R.K., Busson Ph., Centro S., Charlot C., Davis E.W., Denes E., Gheorghe A., Klefenz F., Krischer W., Legrand I., Lourens W., Malecki P., Männer R., Natkaniec Z., Ni P., Noffz K.-H., Odor G., Pascoli D., Zoz R., Sobala A., Taal A., Tchamov N., Thielmann A., Vermeulen J., Vesztergombi G.: Evaluating Parallel Architectures for two Real-Time Applications with 100KHz Repetition Rate; IEEE Tr. Nucl. Sci., Vol. 40, No. 1 (1993) 45-55

[3] Klefenz F., Männer R., Noffz K.-H., Zoz R.: EAST note 92-01

Programmable Level-1 Trigger with 3D-Flow Processor Array

Dario Crosetto - SSCL*

ABSTRACT

The 3D-Flow parallel processing system is a new concept in processor architecture, system architecture, and assembly architecture. Compared to the electronics used in present systems, this approach reduces the cost and complexity of the hardware and allows easy assembly, disassembly, incremental upgrading, and maintenance of different interconnection topologies. The 3D-Flow parallel-processing system benefits high energy physics (HEP) by allowing: (1) common, less costly hardware to be used in different experiments, (2) new uses of existing installations, (3) tuning of trigger based on the first analyzed data, and (4) selection of desired events directly from raw data. The goal of this parallel-processing architecture is to acquire multiple data in parallel (up to 100 million frames per second) and to process them at high speed, accomplishing digital filtering on the input data, pattern recognition (particle identification), data moving, and data formatting. The main features of the system are its programmability, scalability, high-speed communication, and low cost. The compactness of the 3D-Flow parallel-processing system in concert with the processor architecture allows processor interconnections to be mapped into the geometry of sensors (detectors in HEP) without large interconnection signal delay, enabling real-time pattern recognition. The overall 3D-Flow project has passed a major design review at Fermilab. (Reviewers included experts in computers, triggering, system assembly, and electronics.)

1. INTRODUCTION

This work originated by understanding the requirements of Level-1 triggers [1] for various experiments, past and present, as well as one designed for the future. Each experiment was studied in some detail, with visits to the site when possible, with the aim of defining a system architecture, processor architecture, and assembly architecture that had a commonality of features to implement all of the experiments.

The goal is to implement a new, programmable Level-1 trigger by using a "3D-Flow" processor system. This will simplify the hardware and reduce the cost of Level-1 trigger systems. It can be used in current experiments and is intended to open doors to new ways of doing triggering in experimental high energy physics. This new, more powerful tool will allow implementation of different first-level trigger algorithms, enabling researchers to find interesting events with much greater flexibility than existing approaches offer.

The concept is rather simple. The user translates any digital filter, and/or pattern recognition, and/or data-moving algorithm (from Monte Carlo simulation) into a real-time program of the type described in Table 4 of Report SSCL-PP-445 (August 1993). The user's effort is minimal, typically requiring writing of only one and one-half pages of code.

Currently, different experiments use different electronics hardware that is not applicable to other experiments. The 3D-Flow architecture is very flexible, using only one small electronic board (12 cm × 12 cm) that includes 4 × 3D-Flow processor chips.

*Operated by the University Research Association, Inc. for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

Each 3D-Flow chip can accommodate four 3D-Flow processors.

The way in which the 3D-Flow parallel-processing system maps the processing elements to the detector elements guarantees fast timing. Along with an important parameter in the performance of a Level-1 trigger system is fast data communication between elements. The 3D-Flow system allows permits arrangment of the processing elements in the same relative positions as the detector elements. allowing implementation of different topologies. In a parallel-processing system. where results of a calculation of pattern recognition may be dependent on the data coming from the neighboring elements. the overall communication speed will obviously be determined by the longest cable. Thus it is important to keep cables short and approximately the same length. (The FIFOs compensate for the small differences in cable length.) The 3D configuration permits this.

The current status of the system design was achieved during the past three years at the SSCL. and the chip design has been completed. All cabinets (10) for a system of 1280 calorimeter trigger tower signals have been built. as have all 80 cooling racks and 60 Mini-Racks of the 3D-Flow system. Prototypes of daughterboards. motherboards. receiverboard. and controllerboard have been built. and tests on the transmission of the signals have been performed with BiCMOS drivers up to 140 MHz.

The engine that would be available in the present system (10 cabinets with 80 Mini-Racks) has a capability of 2 (8) Tera-operations-per-second at peak performance and will require only 240 (960) 3D-Flow daughterboards (12 cm × 12 cm). and 960 (3840) 3D-Flow processor chips. The size and performance of this engine could easily be upgraded in the future. The user can use this engine with an identical. almost infinite-length algorithm program executed on each processor (Single Instruction Multiple Data mode: SIMD) or with different programs on different processors for a finite-length program (Multiple Instruction Multiple Data mode: MIMD. limited to the size of the program memory for each 3D-Flow processor).

2. 3D-FLOW PROCESSOR ARCHITECTURE

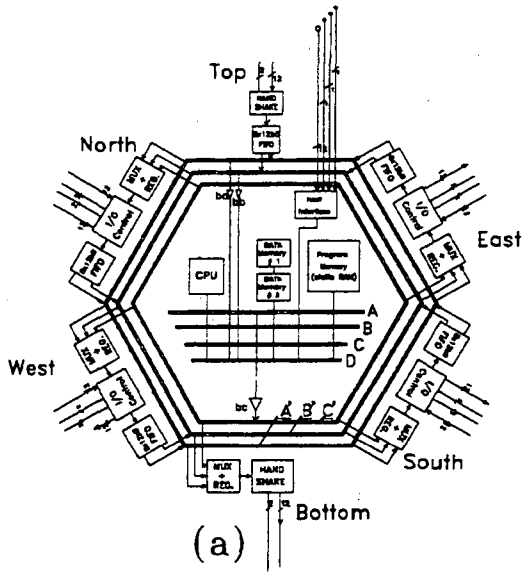
The 3D-Flow processor is a programmable. data stream pipelined device that allows fast data movements in six directions with digital signal-processing capability. Its cell architecture is shown in Figure 1(a). the input/output in Figure 1(b).

The main features are: MIMD/SIMD mode of operation: fast simple integer processor with several units operating in parallel: simultaneous processing and data moving operations in six directions with maximum capability of 13 operations each cycle: 16-bit integer arithmetic. 32-bit precision accumulators: five 8-bit parallel input ports. five 8-bit parallel output ports.

Each 3D-Flow processor in SIMD operation mode receives the same instruction through 48 input lines and operates synchronously with the other processors. There is no dependency on data present at the input or output: thus. the cable length between 3D-Flow processors must be appropriate to allow communication of data to arrive in time with respect to the instruction execution cycle.

The MIMD operation mode allows instead the loading of different programs in each 3D-Flow processor prior to execution. In addition the data dependency at the input and output ports is enabled to allow the system to operate on a data-driven principle. In this case. program execution is controlled by the presence of the data at five ports (North. East. West. South. and Top) according to the instructions being executed. A clock synchronizes the operation of the

cells. At each input port of the 3D-Flow processor there is a FIFO that derandomizes the data from the calorimeter to the processor array. North. East. West. and South ports are 8-bit parallel bi-directional on separate lines for input and output. while the Top port is 8-bit parallel input only. and the Bottom port is 8-bit parallel output only. North. East. West. and South ports are used to exchange data between adjacent processors belonging to the same 3D-Flow array (stage).



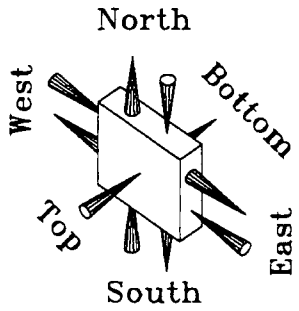


Fig. 1. (a) 3D-Flow processor cell
(b) 3D-Flow input/output.

Top and Bottom ports are used to route input data and output results between stages under program control. Each 3D-Flow cell consists of a Multiply Accumulate and Divide unit (MAC/DIV); arithmetic logic units (ALUs); comparator units: encoder units: a register file: a program memory. an interface to the serial RS232 (used to preload programs. data. and thresholds. and to debug and monitor during their execution); and two data memories (used also as a look-up table to linearize the compressed signal. to remove pedestals. and to apply calibration constants). Four core buses allow communication among these units. while three-ring buses also allow simultaneous communication among I/O.

For an instruction limited to six different operands. in a single cycle the 3D-Flow can perform the following tasks: move the six 16-bit values from six different sources to one (or up to ten) different destination units: and calculate three different results from a multiply/accumulate (or divide) and from two ALUs on the same operands in different units: perform compare operations among the same operands. or between four operands and 32 preloaded threshold values: and initiate an encoding operation. Several 3D-Flow processing elements. shown in Figure 1. can be assembled to build a parallel-processing system. as shown in Figure 2.

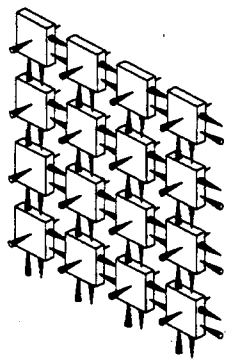


Fig. 2. One stage (or layer) of 3D-Flow parallel processing system.

3. INTRODUCING THE THIRD DIMENSION IN THE 3D-FLOW SYSTEM

In applications where the processor algorithm execution time is greater than the time interval between two consecutive data inputs. one stage (or layer) of 3D-Flow

processor is not sufficient. The problem can be solved by introducing the third dimension in the 3D-Flow parallel-processing system, as shown in Figure 3.

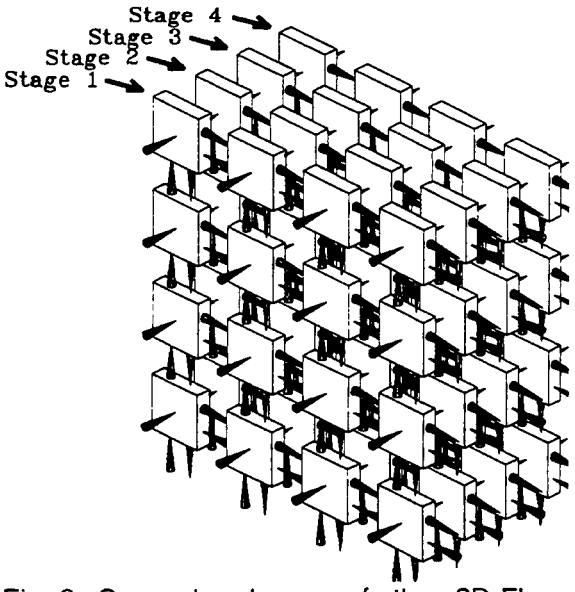


Fig. 3. General scheme of the 3D-Flow pipeline parallel-processing architecture.

A schematic view of the system is presented in Figure 3, where the input data from the external sensoring device are connected to the first stage of the 3D-Flow processor array. The program execution at stage 1 must not only route the new incoming data from the sensor to the next stage in the pipeline (stage 2), but must also execute its own algorithm. Thus, in the pipelined 3D-Flow parallel-processing architecture, each processor executes an algorithm on a set of data from beginning to end (e.g., the event in HEP experiments, or the picture in graphic applications). Input data flows from the "Top Layer" to the appropriate "Layer" where it is processed. Results from this processing flow to the "Bottom Layer" of the 3D-Flow system. Four counters at each processor arbitrate the position of the bypass/in-out switches position in order to achieve the proper routing of data.

Figure 4 shows the timing (at the bunch crossing rate) of the input data to each stage (or layer) and the algorithm execution time (latency) in the 3D-Flow pipelined architecture. Figure 5 shows the arbitration by the 3D-Flow internal counters of the position of the bypass/in-out switches for a 3D-Flow system made of three layers and with the following configuration: maximum input data rate is 1/8 of the 3D-Flow processor clock frequency; algorithm length is 24 steps, and two input and two output values for each event.

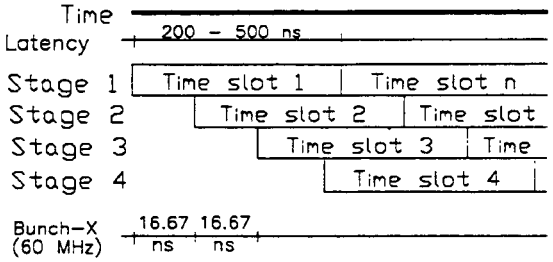


Fig. 4. Timing diagram of four 3D-Flow pipelined stages.

Figures 6 and 7 show assembly details of the 3D-Flow system and Mini-Racks.

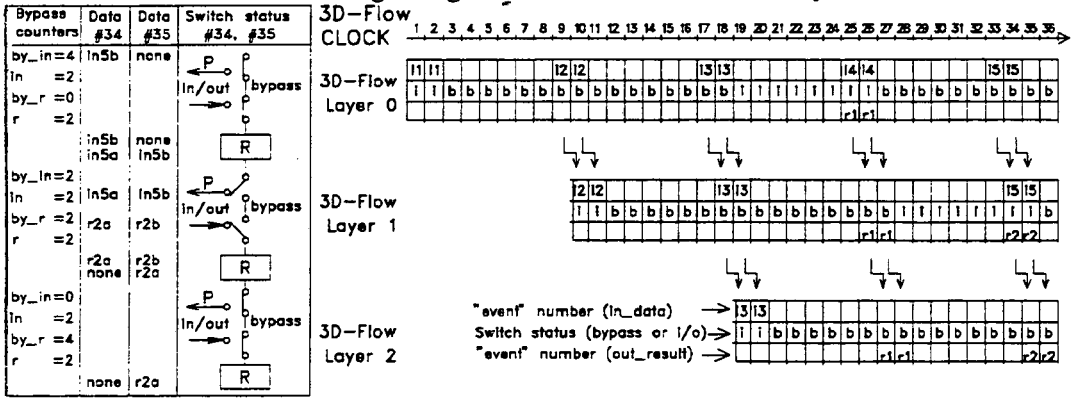


Figure 5. Position of the switches for the data flow (Input/Output) from "Top Layer" to "Bottom Layer."

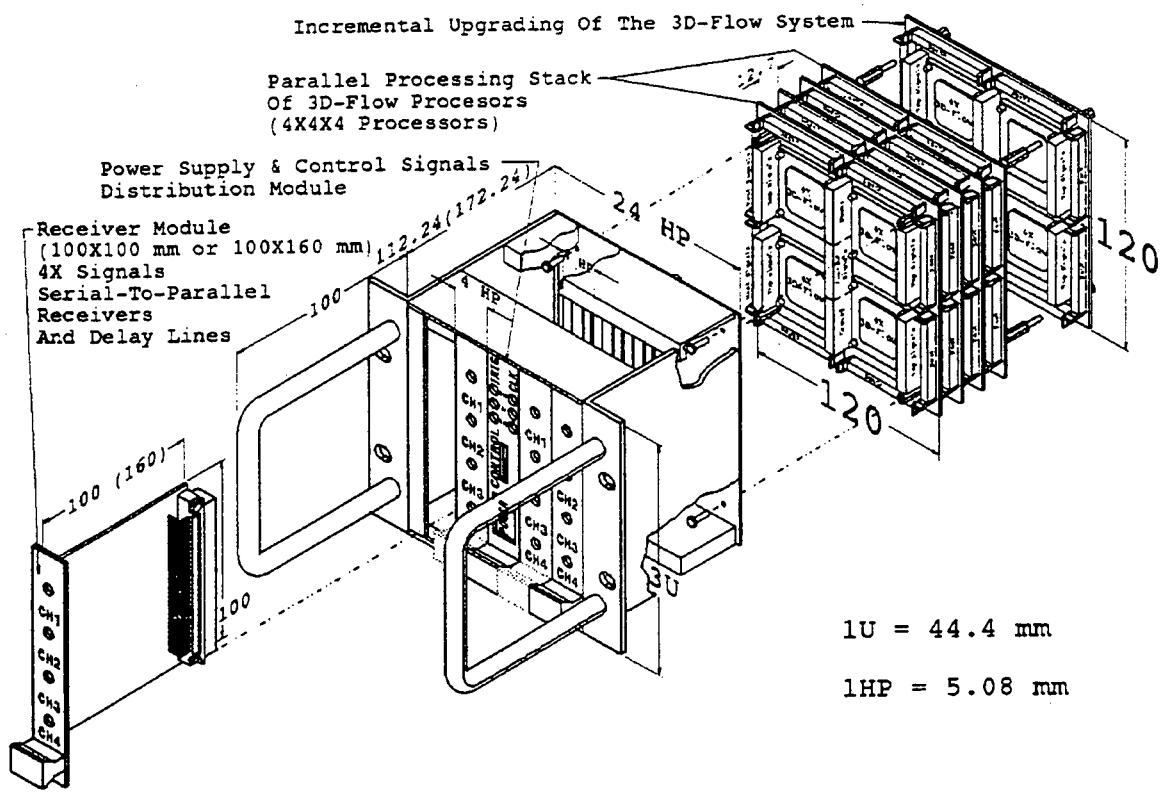


Figure 6. How to assemble, disassemble, and add 3D-Flow processor boards to a Mini-Rack.

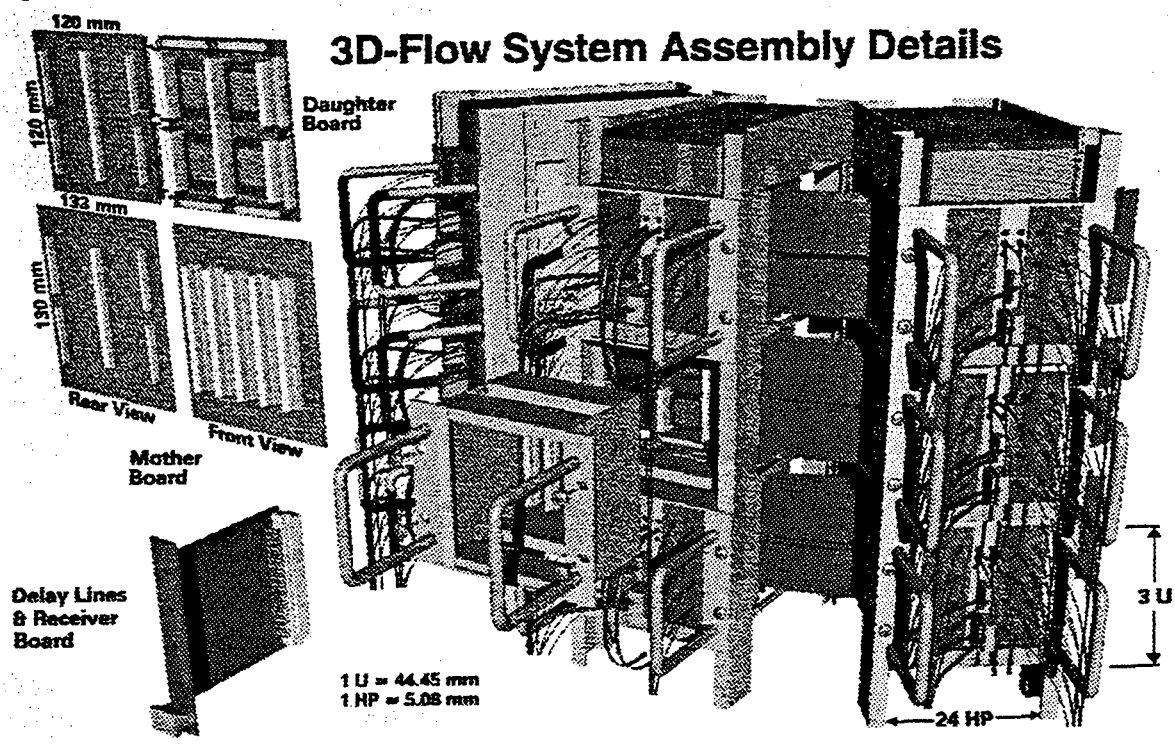


Figure 7. 3D-Flow System Assembly Details.

4. 3D-FLOW PROTOTYPES

Figures 8 and 9 show the prototype receiverboards, motherboards, daughterboard and the flexible printed circuit on which transmission tests have been performed up to 140 MHz. The 3D-Flow processor is designed at the Gate level and is awaiting only production. Meanwhile, 10 cabinets accommodating 80 Mini-Racks for a system with 1280 calorimeter trigger channels have been built.

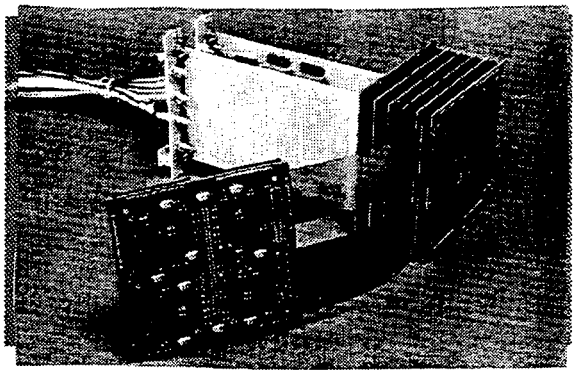


Fig. 8 3D-Flow Mini-Rack prototype assem-bly for the test of signal Transmission between adjacent daughterboards.

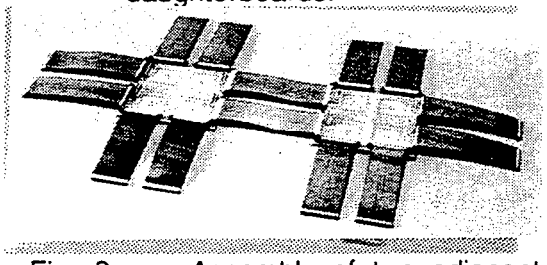


Fig. 9 Assembly of two adjacent prototype daughterboards.

5. CONCLUSIONS

A new approach to a programmable Level-1 trigger has been described based on a study of the requirements of Level-1 triggers of various experiments past, present, and future. The result is a new approach that, when compared to current and past designs, has a number of important advantages:

1. It is more FLEXIBLE, because of its programmability;
2. It has LOWER COST, in both the development and production phases;
3. It has APPLICATIONS to other problems in addition to the calorimeter trigger problem (e.g., real-time trigger tracking);
4. It is EASILY UPGRADED, incrementally;
5. It is EASILY SCALED, both in size (for different detectors) and in speed (for different experiments running on different colliders at different bunch crossing rates); and
6. It offers a POTENTIAL COMMON SOLUTION to the requirements for several current and future experiments.

The core of the 3D-Flow system is the 3D-Flow chip and the 3D-Flow daughterboard (12 cm × 12 cm). The daughterboard (as well as its motherboard) has already been developed. The 3D-Flow chip has been designed and is awaiting fabrication. The overall 3D-Flow parallel-processing system passed a major technical review at Fermilab on 14 December, 1993.

The programmability feature affords physicists the opportunity to include among the first-level trigger some algorithms otherwise left for the higher-level trigger: the result might be better rejection of background events. Today, as for the past three years, physicists continue to perform Monte Carlo simulations, and they are continually revising the trigger requirements to improve background rejection from the trigger algorithm.

REFERENCES

1. L. Mapelli, "The Challange of Triggering and Data Acquisition at Supercollider Experiments," Nuclear Instruments and Methods in Physics Research, A315 (1992) 460-471.
2. D. Crosetto, "3D-Flow Processor for a Programmable Level-1 Trigger," Computing in High Energy Physics, CHEP92, Annecy, France (1992), 803-806.

THE ZEUS CALORIMETER FIRST LEVEL TRIGGER

C. Foudas, I. Ali, B. Behrens, C. Fordham, A. Goussiou, M. Jaworski,
J. Lackey, P. Robl, S. Silverstein, W. H. Smith *U. Wisconsin*,
J. Dawson, D. Krakauer, R. Talaga *Argonne National Laboratory*

Abstract

We present results on the efficiency and performance of the ZEUS detector Calorimeter First Level Trigger (CFLT) using data taken during the 1993 HERA physics run. The CFLT is designed to process events in a digital pipeline applying pattern recognition algorithms and fast digital summation techniques in order to collect interesting physics events and reduce background from beam gas interactions.The total FLT efficiency was 98% for neutral current events and 85% for charged current events above Q^2 of 10 GeV^2. The introduction of the isolated electron trigger increases the CFLT beam gas rejection by a factor of two.

INTRODUCTION

The Zeus detector operates at the HERA storage ring which provides collisions between 820 GeV protons and 26.7 GeV electrons with a rate of 10 MHz (every 96 nsec). Interesting physics events occur at a rate of a few Hz while background from proton beam gas interactions has a much higher rate, on the order of 100 KHz. The calorimeter first level trigger is designed to tag charged and neutral current, photoproduction, and exotic physics events with high efficiency while reducing beam gas background below 1 KHz. This is accomplished by looking for isolated muons and electrons, calculating total (E_{tot}), transverse (E_t) and missing transverse energy (E_{mis}), and finding patterns of energy deposition. These tasks are performed in a pipeline clocked every 96 nsec. The results from the CFLT are transmitted to the ZEUS Global First Level Trigger (GFLT)[1] where the FLT decision is made. The selection at first level trigger is made after 4.4 μsec corresponding to a latency of 46 beam crossings. The dead time is typically less than 0.5%.

CFLT DATA FLOW

CFLT Front End

The ZEUS calorimeter is made of depleted uranium plates and plastic scintillator read out by wavelength shifter bars into 12,864 photomultiplier tubes (PMTs). The PMT signals are received by the Front End Cards (FECs)[2] and combined by the Trigger Sum Cards (TSCs) on the calorimeter to form 896 calorimeter trigger towers (20×20 cm) with separate projective electromagnetic (emc) and hadronic (hac) sections. The CFLT begins with the TSCs, which receive 5% of the signal from the FECs. On the TSCs these signals are integrated, shaped and summed over the left and right PMTs that make up a trigger tower. The TSCs use fast analog circuitry to restore the signals back to baseline in time to receive data from the next crossing. Each TSC transmits the signals from two trigger towers via a 60 m shielded twisted pair cable to the elec-

tronics house, where it enters the Trigger Encoder Cards (TECs) for digitization.

CFLT digital pipeline

The TSC signals are received by 224 TECs installed in 16 9U VME crates at the Zeus electronics house. There the trigger towers are arranged in 16 7×8 grids over the calorimeter, each served by a CFLT crate. Every 96 nsec the TECs digitize the analog signals from each tower using two 8-bit flash ADCs for high and low gain. These 8-bit quantities and a high/low range bit are then converted to energy by applying calibration corrections using a set of lookup tables preloaded with the CFLT calibration constants. The resolution is 48.8 MeV per LSB and the dynamic range from 0 to 400 GeV. The electronic noise level per CFLT tower was measured to be below 200 MeV, enabling the use of trigger thresholds as low as 464 MeV. A second level of lookup tables preloaded with the Zeus detector geometry constants converts these data to E_{tot}, E_t, E_x, and E_y, which are fed into separate adder trees for hadronic and electromagnetic sums. These adder trees calculate the sums of the four towers served by each TEC. An additional set of lookup tables receiving 6-bit compressed scale energy information from both the hadronic and electromagnetic section of each tower compares the energy deposition against 6 programmable thresholds and calculates the energy of the tower on a 3 bit scale (threshold bits). The energy deposition of each tower is also classified as Quiet(Q) or Minimum Ionizing(M) or Electromagnetic(E). This information is sent on an 83 MHz backplane from the 14 TECs contained in one CFLT crate to its two Trigger Adder Cards (TAC)[3].

The TACs process the data into quantities used by the Calorimeter Trigger First Level Processor (CFLTP)[4]. The TACs use the threshold bit information to calculate energies in 8 programmable subregions of the 56 towers serviced by one crate. These sums are used by the CFLTP for calculating energy deposited in specific regions, such as towers adjacent to the beampipe. The threshold bits are also used to provide histograms of tower energy, calculating the number of towers passing each threshold. The CFLTP will use these histograms for a future jet trigger which will search for energy clusters. The TACs also use the full resolution 8-bit quantities (E_{tot}, E_t, E_x, E_y) to continue the sums to the 56 tower level. Lastly, the TACs use the Q,E,M bits to search with memory lookup tables for patterns of electron or muon energy deposited in 1-4 towers, surrounded by quiet towers. If the electron or muon is found isolated on an edge of the 56 tower region, this information is passed separately to the CFLTP, which compares the edge with the adjoining edge in the adjacent region. For this reason the TACs also flag edges as *Quiet* and pass this information to the CFLTP. The CFLTP then sums the totals found over the whole calorimeter and transmits them to the GFLT which uses them to derive the FLT decision.

CFLT ALGORITHMS

The FLT decision during the 1993 run was based on the OR of the following conditions: $E_{tot} \geq 15.0$ GeV, $E_t \geq 11.6$ GeV, $E_{mis} \geq 12.00$ GeV, total *emc* energy ≥ 10 GeV, barrel calorimeter *emc* energy ≥ 3.4 GeV, rear calorimeter *emc*

energy away from the beam pipe $\geq$ 2.0 GeV, and rear calorimeter *emc* $\geq$ 3.75 *GeV*. A number of other subtriggers were also used in coincidence with the luminosity detector trigger in order to collect photoproduction events. The total FLT rate was 75 *Hz* for a HERA luminosity of 0.7×10^{30}, which corresponds approximately to 1/20 of the design luminosity. The efficiency of each subtrigger relative to the rest was studied using CFLT data. The overall efficiency of this algorithm for neutral current events was 98% and for charged current events 85%. The rear calorimeter *emc* (*REMCth*) subtrigger was the most efficient trigger for collecting neutral current events. More than 92% of the ZEUS neutral current sample were tagged by this subtrigger and about 49% of it was collected exclusively by *REMCth*. The purity of *REMCth* was low and about 95% of the events from this trigger were rejected by the second level trigger[1] as originating from beam gas interactions. In 1994 the HERA luminosity is expected to increase by a factor of 5 and the *REMCth* rate has been estimated to be prohibitively high at the FLT level. Therefore this trigger will be replaced by isolated electron trigger which has the same efficiency and a factor of 2 higher purity.

ISOLATED ELECTRON ALGORITHM

The CFLT executes the isolated electron algorithm (ISOE) by searching in the calorimeter for patterns of 1-4 towers with the E bits set surrounded by towers with the Q bits set. When such a pattern is found it results to a positive CFLT decision. A tower is defined as quiet if the *emc* and the *hac* energies deposited in that tower are smaller than the two programmable thresholds Q_{emc} and Q_{hac} respectively.

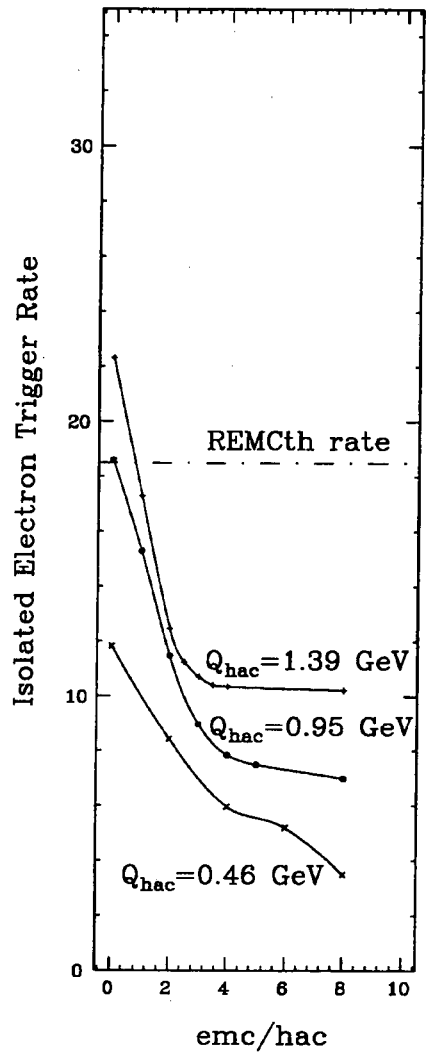


Fig. 1: ISOE beam gas background rate in arbitrary units as a function of Q_{hac} and *emc*/*hac* parameters. The dot-dashed line indicates the rate of the *REMCth* at 3.75 *GeV*.

Electron bits (E) are set based on a user programmable function $E = F(emc, hac)$. In order to study and optimize this algorithm we collected data from several *ep* data runs with the ISOE

trigger active. The optimum values for Q_{emc}, Q_{hac}, and $F(emc, hac)$ were selected in order to obtain maximum beam gas rejection and high physics acceptance. We have chosen $Q_{emc} = 2.52\ GeV$ so that the trigger has 100% efficiency (including energy sharing between towers) at electron energy of 5 GeV, and $Q_{hac} = 0.95$ GeV. The electron bits were set if any of the following conditions were satisfied: $E = \{(emc \geq Q_{emc}).AND.(hac \leq Q_{hac})\}$ OR $\left\{(emc \geq Q_{emc}).AND.(\frac{emc}{hac} \geq 3)\right\}$.

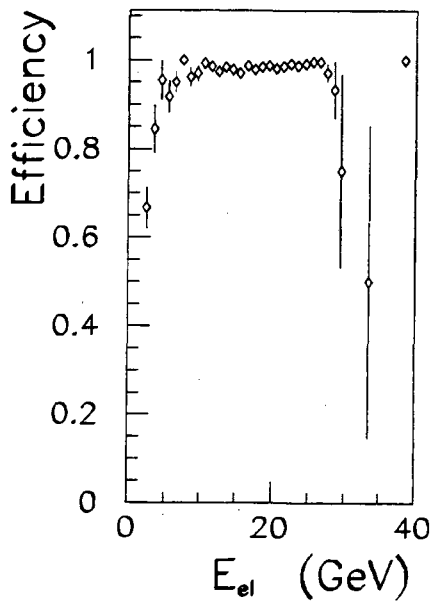


Fig. 2: ISOE efficiency versus electron energy. The efficiency rises between E_{el} 2.5 and 5 GeV due to events where the electron energy is shared between trigger towers (Q_{emc} = 2.52 GeV). Events with $E_{el} \geq 30$ GeV are not electrons since they are outside the kinematic limit.

This ISOE configuration results to a factor of 2 reduction in Beam Gas background relative to the *REMCth* and has the same efficiency as the *REMCth* for neutral current events. In Figure 1 we present the beam gas reduction achieved using the ISOE relative to the simple *REMCth* trigger at 3.75 GeV as a function of the Q_{hac}, emc/hac cuts. The overall efficiency of the ISOE relative to the standard ZEUS electron finder was 98%. In Figure 2 we present this efficiency as a function of the electron energy E_{el}.

CONCLUSIONS

The ZEUS CFLT has performed reliably and according to design over the past 3 HERA physics runs, enabling us to collect over 600 nb^{-1} of physics data. The digital pipelined structure of the CFLT enabled us to collect data with essentially zero dead-time. The programmability of the CFLT via memory lookup tables has proven very useful and its configuration was changed often to accomodate the various physics requirements. In addition, the ISOE pattern recognition trigger was proven a powerful tool for Beam Gas background rejection. This work is supported by the DoE, under Contract No. DE-AE02-76ER0881.

REFERENCES

1. W.H. Smith, K. Tokushuku, L. Wiggers, in *Proceedings of the International Conference on Computing in High Energy Physics*, Annecy, France, September, 1992
2. A. Caldwell *et al.*, Nucl. Inst. Meth. **A321** 356 (1992).
3. C. Fordham *et al.* , in *Proceedings of the 1993 IEEE Nuclear Science Symposium*, San Fransisco, CA, November, 1993.
4. J.W. Dawson *et al.*, IEEE Transactions on Nuclear Science, **37**, 2198 (1990).

Neural Electron/Pion Discriminator With a Projective Fiber Calorimeter

J.M. Seixas , L.P. Caloba and R. Rajagopal
COPPE/EE/UFRJ, C.P. 68504, Rio de Janeiro 21945, Brazil

Abstract

An electron/pion discrimination system is developed using artificial neural networks. Longitudinal and radial information collected by a projective fiber calorimeter are analyzed. Results show that the system is able to identify up to 99.8% of electrons with 0.1% of pions misclassified as electrons.

Introduction

Scintillating optical fibers have been used in calorimeters as the active material in a variety of recent experiments. They deliver fast signals and allow for a sufficiently precise and compact calorimeter design. For the following analysis, a 19 modules projective prototype [1] was used. Modules have the shape of one quadrant of a truncated pyramid with a depth of 2 meters. Each module contains 2346 fibers with a diameter of 1 mm embedded in a lead matrix. All fibers run parallel to one another in the direction of the incoming particles. By building such a detector, it was possible to achieve an effective segmentation into an electromagnetic (e.m.) and an hadronic section so that each section was independently readout. This segmentation is useful for electron/pion discrimination, as electrons, unlike pions, deposit almost all their energy in the e.m. section.

Calorimeter signals were acquired in such a way that concurrent energy and time signal acquisitions were possible. The energy of each section was measured by using a charge ADC [1] and time signals were sampled by a fast Digital Storage Oscilloscope (DSO) [2]. The sampled signal was obtained by combining the seven innermost modules with respect to the interaction point of the incoming particle beam. Figure 1 shows typical pulses.

This paper describes the application of neural networks for electron/pion discrimination. One can explore the ability of these networks in identifying multidimensional correlations on pattern features in order to realize a high-performance discriminator, which can contribute to build efficient trigger systems. In fact, recent works [2, 3] have reported results on neural triggering systems for LHC. Network simulations were performed by using JETNET 2.0 package [4] and backpropagation was the learning procedure adopted.

[1] LeCroy 2280, 2282E modules.

[2] Tektronix DSA 602A, 11A32 modules.

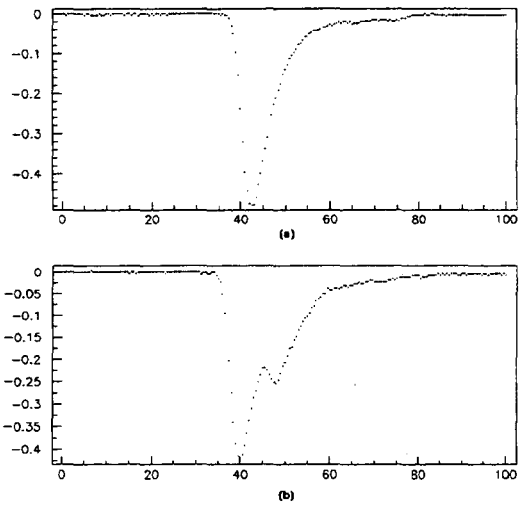


Figure 1. Signals for 40 GeV electron (a) and pion (b) beams. The sampling rate is 2 GSa/s.

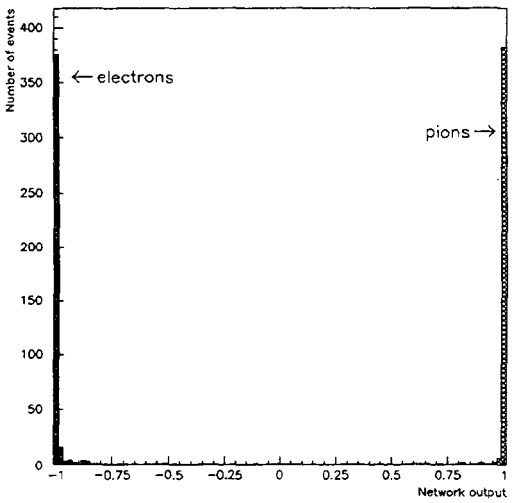


Figure 2. Neural Network output for 20 GeV particles.

Longitudinal Information

The signals sampled by the DSO show the depositing process in depth. The prototype under consideration has fiber ends which are made reflective by aluminium sputtering. Therefore, as pions normally start showering deeper in the calorimeter, their signals generally exhibit a double peak structure (Figure 1 (b)). Full signals were recorded, but only samples lying inside a 100 ns fixed window were fed into the neural network input nodes, the selection being performed by cutting the base line points that carry noise information. A 100-50-1 neural network was used.

The obtained result shows that less than 0.1% (6%) of pions are misclassified as electrons when a 99.8% electron efficiency is kept for 40 (20) GeV particles. At 95% electron efficiency, the error for pions was 0% (4%). This is to be compared with nonneural methods. As an example, computing the $E_{had}/E_{e.m.}$ ratio [1], for 95% (92%) electron efficiency, 0.3% (4%) of the pions are misidentified, for 40 (20) GeV data. Therefore, the proposed neural discriminator exhibits a better performance when compared with classical methods. Figure 2 shows how the network classifies events.

It is known [5] that the discrimination based on a constant fraction measurement is improved by filtering the tail introduced on calorimeter signals by the signal cable used to transmit them. However, there was no improvement in the performance of the neural discriminator when filtered pulses were fed into the network. Therefore, the neural network performs some filtering action in its hidden layer, so that results with and without cable filtering are similar.

Radial Information

The way particles deposit their energy inside the detector can be seen in Figure 3. For radial information analysis, the network was fed with the measured energy from all 19 e.m. and 19 hadronic cells. The energy values were normalized by total energy and the same set of events used for the longitudinal analysis was considered. A 38-38-1 network was

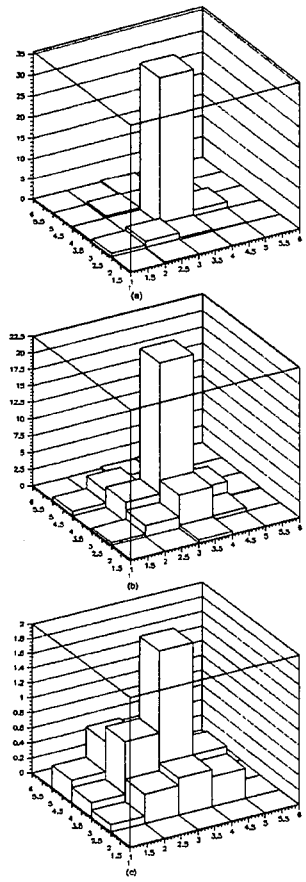


Figure 3. Typical radial profiles for electrons (a), e.m. section, and pions, e.m. (b) and hadronic (c) sections.

used. For 99.8% electron efficiency, 0.1% (1.4%) of pions were misclassified as electrons for 40 (20) GeV data.

Conclusions

A neural network approach to an electron/pion discriminator system using projective fiber calorimeter information was developed. At 40 GeV, it was shown that using longitudinal or radial information, discrimination errors of less than 0.1% were found for pions when an electron efficiency up to 99.8% is kept. This performance is better than reported results using classical methods.

Acknowledgements

We are thankful for the support that has been provided to this work by CNPq (Brazil) and CERN. We would like to thank our colleagues of the RD1 Collaboration for their contribution during data acquisition and C.C. Jannnuzzi for helping in part of the analysis.

REFERENCES

1. J. Badier et al. Nucl. Instr. and Meth. **A 337** (1994) 326.
2. J.M. Seixas et al. A Second-Level Trigger System Based on Calorimeters and Using Neural Networks for Feature Extraction and Electron/Jets Discrimination. IV Int. Conf. on Calorimeters for High En. Phys. Elba, Italy, 1993.
3. J.M. Seixas et al. Neural Networks Applied to a Second-Level Trigger System. III Int. Workshop in Softw. Eng., Artificial Intell. and Expert Sys. for High-En. and Nucl. Phys. Oberammergau, Germany, 1993.
4. L. Lönnblad et al. Comput. Phys. Commun. **70** (1992) 167.
5. J.M. Seixas. Fast Front-End Electronics for Electron/Pion Separation in a Lead/Scintillating Fiber Calorimeter. II Int. Conf. on Calorimetry for High-En. Phys. World Scientific, pp 445, 1992.

A First-Level Calorimeter Trigger for LHC Experiments - Design Studies, and Beam Tests of a First Prototype Trigger System

A.T. Watson
I.P. Brawn, R.E. Carney, Y. Ermolin, J. Garvey, R.J. Staley
School of Physics and Space Research, The University of Birmingham, Birmingham B15 2TT, UK.

E. Eisenhandler, M. Landon
Department of Physics, Queen Mary and Westfield College, London E1 4NS, UK.

C.N.P. Gee, A.R. Gillman, R. Hatley, V. Perera
Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, OX11 0QX, UK.

N.N. Ellis
CERN, 1211 Geneva 23, Switzerland

Introduction:

First-level triggering for LHC proton-proton collider experiments poses significant experimental challenges. At a luminosity of 10^{34} cm^{-2} s^{-1} the inelastic proton-proton collision rate will be approximately 1 GHz. With a bunch-crossing interval of 25 ns this corresponds to approximately 25 collisions per bunch-crossing. The level-1 accept rate will be limited by the readout time of the front-end electronics to less than 100 kHz, requiring a rejection factor of 10^4-10^5. Physics signatures will be based upon combinations of high-p_T leptons, photons, partons and weakly-interacting particles, and therefore the trigger must be sensitive to these particles. This paper describes tests of a first prototype calorimeter trigger processor and developments in the design of a complete first-level calorimeter trigger system.

Trigger Algorithms:

The choice of algorithms is based upon extensive and ongoing physics simulation studies. These have been described elsewhere [1-3]. The inputs to the algorithms are transverse-energy (E_T) sums from "supercells" in the electromagnetic and hadronic calorimeters, of transverse granularity 0.1×0.1 in pseudorapidity-azimuth and summed in depth. The most complex algorithms are required for the electromagnetic cluster trigger (Figure 1). The trigger requires an electromagnetic cluster E_T greater than a cluster threshold as well as electromagnetic and hadronic isolation sums less than isolation thresholds. The trigger windows slide and overlap so as to cover the calorimeters fully. Studies indicate that such an algorithm gives acceptably sharp trigger thresholds and rates of a few $\times 10^4$ Hz for a range of trigger conditions satisfying the requirements of LHC pp experiments [1-5].

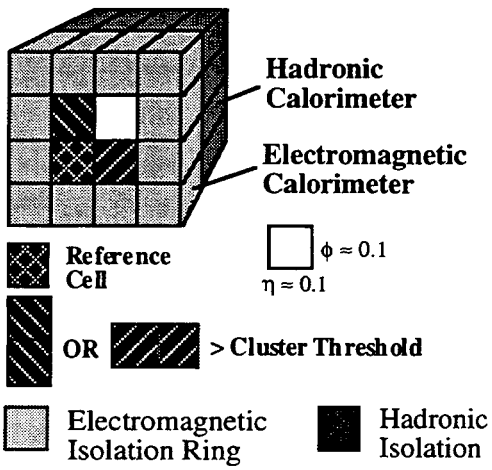


Figure 1: e/γ trigger algorithm

Trigger Prototype:

In order to investigate and demonstrate the implementation of such trigger algorithms, an ASIC has been constructed to implement the electromagnetic part of the algorithm shown in Figure 1 (hadronic processing was excluded to reduce cost). The ASIC is a 0.8μ CMOS gate array operating in pipeline mode. It takes as

input 16 8-bit linear ADC values and calculates cluster and isolation sums, which are compared with two sets of thresholds. In addition the sum of the 16 input values is computed, an operation required as input to the jet trigger. The ASIC delivers the 12-bit energy sum and the two trigger decision bits with latencies of six and seven clock cycles respectively. It has been described in more detail elsewhere [3].

The ASIC was tested in a cluster processor demonstrator system using a prototype LHC calorimeter. The system contains nine ASICs and fully processes 3×3 overlapping windows, requiring inputs from 6×6 trigger cells. Two backplanes were used, one with high-density Teradyne connectors (320 signal pins plus 80 grounds) for the data and clock connections and the other using a simple control/addressing protocol to interface to VMEbus. The module contains 256-deep circular buffers to capture time-slices of the input ADC data and the ASIC processing results. Comparison of the results with the inputs enables the performance of the trigger hardware to be studied in detail.

Beam Tests:

Data were recorded in CERN test beams with both the RD-3 barrel ("Accordion") and endcap ("Spanish Fan") liquid-argon calorimeter prototypes. Analysis of the "Spanish Fan" data is still at an early stage, so the following results come from the barrel calorimeter tests. In addition other tests have been performed with the RD-33 ("TGT") liquid argon calorimeter prototype.

Trigger cells of granularity approximately 0.1×0.1 were formed by analogue summation of signals from the calorimeter cells. The signals were digitised using a linear flash-ADC system sampling at the LHC bunch-crossing frequency of 40 MHz (some data were recorded at the original LHC frequency of 67 MHz) and passed to the cluster processor operating at the same speed. Data were taken with beam energies between 10 GeV and 300 GeV and a mixture of electron and pion beams, over a wide range of beam positions. Data were recorded both stand-alone, and also through the RD-3 data-acquisition system, to enable comparison with the full calorimeter data.

Figure 2 shows the trigger threshold curve obtained with 50 GeV electrons compared with the expectations from an ATLAS Monte Carlo. Since the beam spot-size was small compared with a trigger cell (about the size of a single calorimeter cell, whereas the trigger cell was 4×4 calorimeter cells) data from a number of runs with beams in different positions within the trigger cell were combined to produce this threshold curve. A good agreement was found between the observed and expected performance of the cluster algorithm.

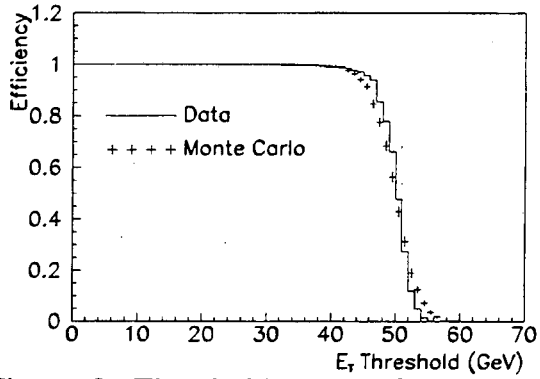


Figure 2: Threshold curves from test data compared with Monte Carlo

The reliability with which the trigger processor executed the algorithms was studied by comparing the ASIC results with the expectation based upon the input ADC data and the trigger and isolation thresholds.

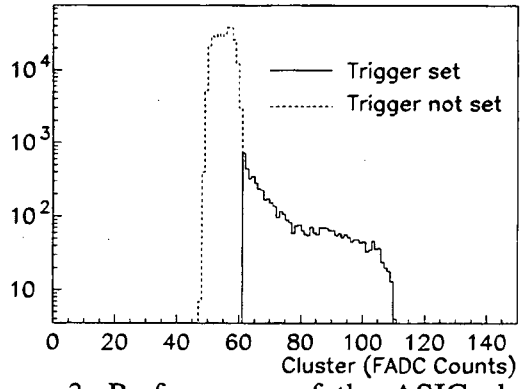


Figure 3: Performance of the ASIC cluster threshold

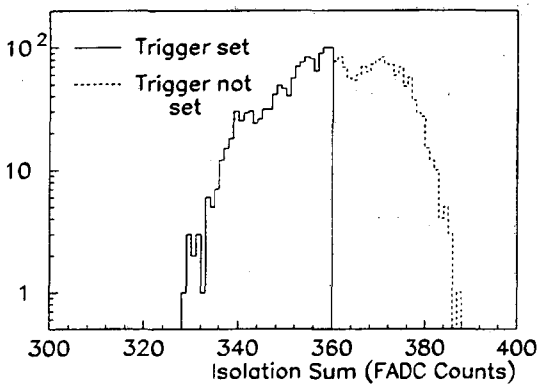


Figure 4: Performance of the ASIC isolation threshold

The performance of the ASICs in forming cluster and isolation sums and comparing with thresholds is illustrated in Figures 3 and 4. In Figure 3 the cluster sum, reconstructed from the input data, is plotted for events with and without the corresponding trigger bit set. Here the isolation threshold was set high to permit a clean test of the clustering. In Figure 4 the isolation sums are similarly plotted for events with clusters above the cluster threshold. In both plots it can be seen that the execution of the trigger algorithm at the full LHC clock speed was faultless.

Bunch-Crossing Identification Studies:

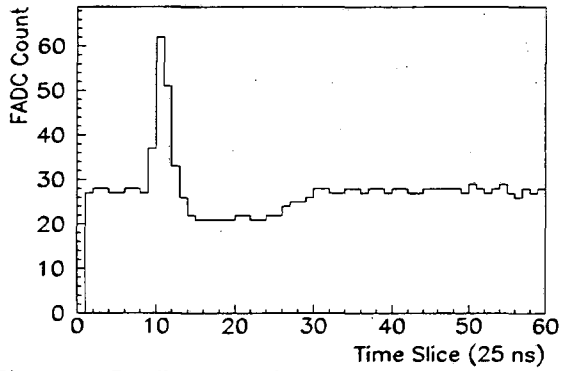


Figure 5: Pulse shape from Accordion calorimeter

Figure 5 shows a pulse from the Accordion calorimeter, after digitisation by the trigger FADC. A key feature is that the pulse occupies about four LHC beam-crossings in time. For the final trigger, filtering is therefore needed to identify the peak pulse-height and timing, and to suppress off-peak data which would otherwise contribute to (or veto) triggers in other bunch-crossings.

We have been studying the use of digital filtering for this purpose. Using software simulation, a range of algorithms has been studied, based upon convolution of the data from several samplings followed by peak-finding applied to the results of the convolution. Using data recorded with the Accordion calorimeter prototype as input, it was found that simple peak-finding always gave correct identification of pulses down to five counts (approximately 6 GeV), with little improvement being found from using more complicated filtering. Small pulses are the most difficult to filter, and studies in this area are continuing. Lower electronic system noise and higher-resolution ADCs would permit the performance of all algorithms to be improved.

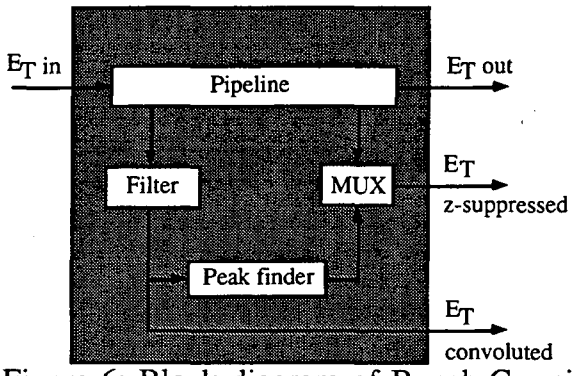


Figure 6: Block diagram of Bunch-Crossing Identification module

A first-prototype bunch-crossing identification module has been constructed and tested with the "Spanish Fan" endcap calorimeter prototype. The module, which filters data from a single channel, consists of a five-element deep pipelined Finite Impulse Response (FIR) filter followed by a five-element programmable comparator (peak-finder). It is illustrated in Figure 6. By setting the parameters of the filter and peak-finder appropriately, a range of algorithms could be tested in real-time at the full LHC bunch-crossing rate. The next trigger prototype will include similar bunch-crossing identification capabilities for all input channels, using Xilinx FPGAs for the digital signal processing.

Towards a Full LHC Calorimeter Trigger:

The current prototype has demonstrated that algorithms with the performance needed for LHC first level triggering can be implemented using currently-available ASIC technology and operated at the necessary speed . In order to construct a full system, however, a greater degree of integration is needed [6]; the current module processes only 9 channels, so using this modularity a full-scale 4000-channel trigger would require approximately 450 such modules. It is straightforward to process a larger area of the calorimeter in a single ASIC, but the pin counts required for parallel data input quickly become prohibitive. For example, an ASIC fully processing 4×4 trigger cells would require ~800 I/O pins alone, and a module with four such ASICs would require ~2000 backplane connections for inputs.

One solution to this problem would be to use serial input of data to the ASICs. This would provide an eightfold reduction in the number of I/O connections, but would require a corresponding increase in the bandwidth per connection. Current CMOS ASIC technology could not accept data at these rates. However, after pedestal subtraction, bunch-crossing identification and applying a noise threshold to the trigger cell E_T sums (typically 1 GeV) the occupancy of the calorimeters is expected to be only a few percent. Transmission of only zero-suppressed data would therefore reduce the bandwidth, but the resulting asynchronous data would then require bunch-crossing tagging and matching. Although having a greater latency, one advantage of this technique is that it would be inherently self-synchronising.

In order to demonstrate the viability of such a solution, as second trigger prototype is being constructed. In this system the digitised calorimeter pulses are pedestal-subtracted and applied to a bunch-crossing identification filter. The non-zero data are then tagged, serialised and transmitted optically at a rate of 160 MBd.

The cluster-finding ASIC converts the incoming serial bitstreams to parallel data, matches the tags and them passes the re-synchronised data to the trigger algorithm block. Note that while data transmission is asynchronous, the trigger decision itself is fully synchronous.

Design studies for this system are well advanced. Detailed system simulations indicate that the asynchronous transmission introduces negligible data loss and no data corruption. To reduce the non-recurrent engineering costs a dual-mode ASIC is being designed, which will include all necessary system functionality. Beam tests with prototype LHC calorimeters will be performed in mid-1995.

Acknowledgements:

We gratefully acknowledge the considerable help we received from RD3, both in allowing us to use signals from their calorimeter and in performing the beam tests.

This work is supported by the UK PPARC.

References:

[1] N. Ellis et. al., *Level-1 Trigger System*, ATLAS note DAQ-NO-005

[2] A. Watson, *Physics Simulation Studies of the First-Level electron/photon Trigger*, RD27 note 13

[3] I.P. Brawn et. al., *Beam Tests of a Prototype Level-1 Calorimeter Trigger for LHC Experiments*, RAL-94-020, (Submitted to Nuclear Instruments and Methods).

[4] ATLAS Letter of Intent, CERN/LHCC/92-4

[5] CMS Letter of Intent, CERN/LHCC/92-3

[6] V. Perera, *A First-Level Calorimeter Trigger Processor for the LHC*, RD27 note 8

THE LEVEL-1 CALORIMETER TRIGGER FOR THE CMS DETECTOR *

S. Dasu, T. Gorski, J. Lackey, D. Panescu, W. H. Smith, and W. Temple
Department of Physics
University of Wisconsin
Madison, WI 53706 USA

Abstract

We present results from simulation studies and implementation ideas for the level-1 calorimeter trigger for the CMS detector at the LHC collider. QCD background trigger rates and signal efficiencies for electron and photon triggers using several simple level-1 algorithms are discussed. Jet trigger rates and efficiencies are studied for various calorimeter tower sums. Some of the technical challenges in implementing these simple trigger algorithms in a high rate environment are also explored. Design of an high speed adder ASIC is briefly discussed.

INTRODUCTION

The CMS detector[1] is designed to study high p_T physics in 14-15 TeV center of mass proton-proton collisions at the LHC. In order to extend the mass range for Higgs searches and study the TeV scale the LHC is designed to operate at the luminosity of 10^{34} cm^{-2} s^{-1}. The physics program includes the study of electroweak symmetry breaking, studies of the top quark, searches for new heavy gauge bosons, probing quark and lepton substructure, looking for supersymmetry and exploring for other new phenomena. Most of these physics topics are best studied in the events with high p_T leptons or photons because the profusely produced QCD events obscure hadronic channels.

*This work is supported by the Department of Energy Contract No. DE-AC02-76ER00881 and Texas National Laboratory Research Commission Grant No. RGFY93-205.

Triggering is one of the extraordinary challenges facing detector designers at the high luminosity LHC collider. For the nominal LHC design luminosity of 10^{34} cm^{-2} s^{-1}, an average of 25 events occur at the beam crossing frequency of 25 nsec. This input rate of 10^9 interactions every second must be reduced by a factor of at least 10^4 to 100 KHz, the maximum rate that can be analyzed by subsequent processing in an on-line computer farm. This must be done for all channels without dead time. CMS has chosen to reduce this rate in two steps. The first level stores all data for 3 μs, after which no more than a 100 KHz rate of the stored events is accessible by the on-line processor farm. The farm processors first analyze a subset of the event data using fast algorithms operating as a "virtual level-2", to further filter events. A hierarchy of such filters screen the events before saving

data to disk at 100 Hz.

We have explored the trigger issues extensively for the SDC detector which was to have been built at the SSC [2, 3]. The physics interest and the background challange is similar for the CMS experiment at the LHC. We therefore build upon the trigger work on the SDC experiment that has direct application in CMS. However, the choice of the trigger algorithms, and, therefore, the trigger hardware realization, depend on the detector geometry and performance. Additionally, the greater pileup expected at the LHC, and, the lack of a physical level-2 trigger, are likely to place more stringent requirements on the trigger.

TRIGGER SIMULATION

As a first step in this effort we have reworked our SDC fast simulation program to describe the CMS detector geometry and materials. The response of the calorimeter is calculated in this program using standard shower parameterizations available in literature. Before trigger simulation the detector response to hard scattering events and a suitable number of Poisson distributed minimum bias events at the desired luminosity were added. For the purpose of EM trigger simulation, calorimeter transverse energy deposits, in each $0.1\eta \times 0.1\phi$ trigger tower region, were summed and converted to an 8-bit linear scale, with a resolution of 0.5 GeV, to simulate the dynamic range limitation.

The trigger response was then simulated for each event using various level-1 algorithms, including several we have devised and algorithms from other sources such as RD-27[4]. Most algorithms considered all possible two EM trigger tower pairs to cut on the summed EM transverse energy. Our SDC-style EM trigger algorithm involves two separate cuts on the longitudinal and transverse isolation of the EM energy deposit. The first cut requires a hit tower HAD to EM energy ratio, $H/E < 0.05$. The second cut requires transverse isolation, i.e. a cut on sum of HAD transverse energies in the nearest eight towers surrounding the hit tower, $H_1 < 1.5$ GeV. In order to reduce the number of bits of information exchanged between electronics cards we limit the dynamic range of neighboring tower HAD information to 3 bits. Overflows of both the 8 bit scale for EM and central HAD towers, and 3 bit scale for neighboring HAD towers are treated as maxima. The RD-27 algorithm [4] involves only one cut on the sum energy of the 12 border EM towers and 16 HAD towers in the $0.4\eta \times 0.4\phi$ region, i.e, $\Sigma E_2 + \Sigma H_2 < 5$ GeV.

We have studied the expected trigger rates for various algorithms from the QCD background, and the efficiencies for several high p_t physics processes. We have found that the electromagnetic trigger algorithm proposed for the SDC detector works satisfactorily for the CMS trigger. We find that the transverse isolation criterion which provided marginal improvement for the SDC is essesential to trigger on interesting physics at the LHC. We have also compared our SDC style algorithm to the one proposed by CERN RD-27 group [4] for Atlas trigger. The rates from QCD two jet production background are shown for various algorithms in Figure 1. The efficiencies of the electron trigger for these algorithms obtained using W decay events including the minimum bias background is shown in Figure

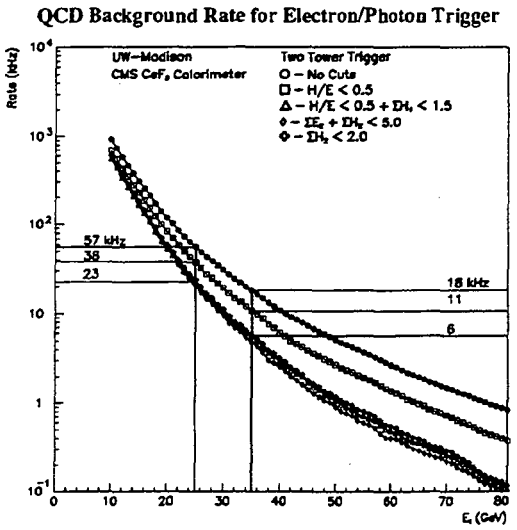


Figure 1. Single electromagnetic cluster trigger background rate for CMS is plotted versus the two tower sum transverse energy deposited with various cuts. The circular points are without any cuts, the square points and the triangular points are for our algorithm, the diamond and plus shaped points are for the RD-27 algorithm.

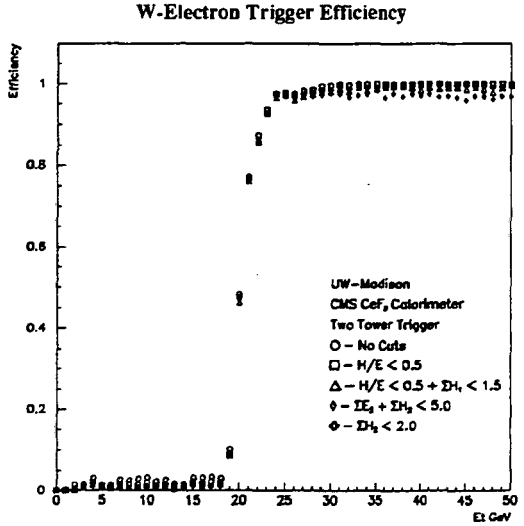


Figure 2. Efficiency of triggering on electrons from W decay events for CMS is plotted versus p_T of the electron for various algorithms with labeling as defined in the Figure 1.

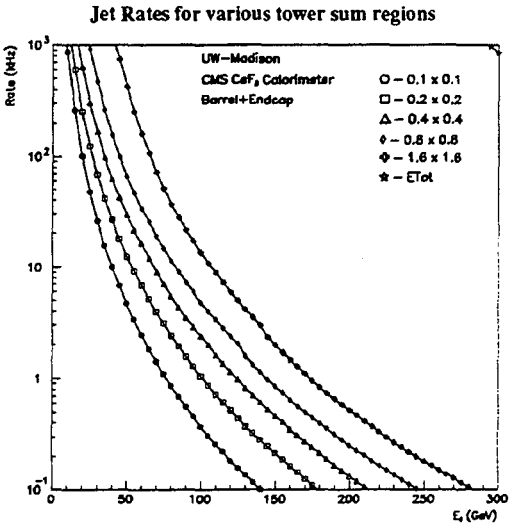


Figure 3. Jet trigger rates using various tower region sums are plotted versus the E_t deposited in the calorimeter region. The rate includes only barrel and endcap calorimeters, i.e. $|\eta| < 2.6$.

2. We find that our SDC algorithm performs similarly to the RD-27 algorithm. There are two interesting features in our SDC algorithm. Firstly, our proposed algorithm requires smaller amount of data than the RD-27 algorithm.. Secondly, our algorithm allows for a non-isolated electron trigger for the initial low luminosity running at the LHC when one might be interested in studying B physics which requires a non-isolated electron trigger.

We have studied jet trigger rates (see Figure 3) and efficiencies (see Figure 4) and found that transverse energy sums in $0.8\eta \times 0.8\phi$ regions provide sufficient performance. We have also studied efficiencies for various high E_t physics signals as a function of chosen thresholds for single and double EM triggers. Although the efficiency for very high E_t physics processes resulting in two or more electrons/photons is very good, efficiency for top and W triggers is somewhat low for

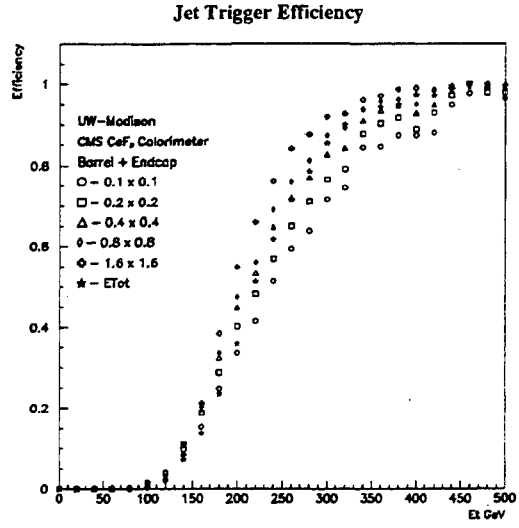


Figure 4. Efficiency of jet trigger for various tower region sums is plotted versus the jet p_T.

the nominal rates/thresholds proposed for CMS (see Table 1).

CMS TRIGGER DESIGN

The CMS Level 1 Calorimeter trigger data is received from the front end electronics on optical fibers in digital form, either on an 8-bit nonlinear scale that is decoded with memory lookups or on a 10-bit linear scale. The fiber receiver converts the serial optical data to parallel data on copper. It synchronizes the incoming data to the trigger system 25 nsec clock and aligns it to the correct crossing number. The fiber receiver does error detection, using error codes transmitted with the data, and logs the errors for subsequent readout.

After reception and synchronization, the calorimeter trigger data is fed to two circuits. The first uses memory lookup tables to convert the energies into E_t, E_x, and E_y and then injects these quantities into a summation network, based on a high speed GaAs Adder ASIC [5], and computes sums of energies on groups of towers for jet triggers and over the entire calorimeter for the global E_t and missing P_t triggers. The second circuit performs our isolation algorithm, described earlier, to identify electrons and photons. The results of both circuits are compiled by subsequent logic into lists of jets and electrons passing various thresholds and other conditions, as well as the global energy sums. These are forwarded to the global level 1 trigger.

We are presently in the process of completing the adder ASIC design (see Figure 5) and will produce a test GaAs integrated circuit. We will build a VME board to test this ASIC at speeds up to 160 MHz.

We plan to cycle the CMS calorimeter trigger system at four times the LHC speed, i.e. 160 MHz, in order to attain a compact size at an affordable cost. System engineering issues to realize such a design are being studied.

REFERENCES

1. CMS, The Compact Muon Solenoid, Letter of Intent, CERN/LHCC 92-3, 1992; CERN/LHCC 93-48, 1993.
2. W.H. Smith, in *Proc. of the Int. Conf. on Comp. in HEP*, Annecy, France, 1992.
3. S. Dasu, *et al.*, *Proc. of the Int. Conf. on Cal. in HEP*, Isola d'Elba, Italy, 1993.
4. N. Ellis *et al.*, CERN/DRDC 92-17, 1992.
5. D. Panescu *et al.*, *Proc. of the 1992 IEEE Nucl. Sci. Symp.*, Orlando, FL.

Singles Threshold GeV	Doubles Threshold GeV	Background Rate kHz	Higgs trigger Efficiency %	Top trigger Efficiency %	W trigger Efficiency %
20	10	54.5	99	85	79
25	12.5	23.0	97	79	67
30	15	10.8	95	71	52
35	17.5	5.6	92	63	33
40	20	2.9	87	57	16
45	22.5	1.7	82	49	7
50	25	1.1	74	43	4

Table 1. CMS calorimeter trigger efficiency for Higgs decays to two photons, top decays to electron, and W decays to electron processes are shown for our algorithm as a function of trigger threshold with corresponding rates from QCD background.

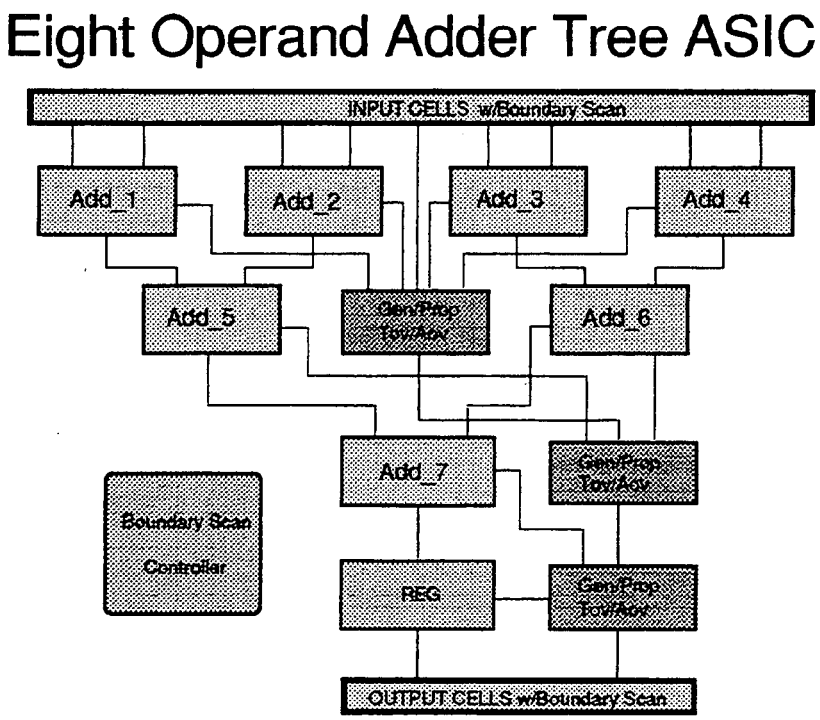


Figure 5. Adder ASIC design block diagram.

Evaluation of a parallel processing system and parallel algorithm for triggering in new generation colliders

E. Davis†, G. Busetto‡, C. Carpanese‡, S. Centro‡,
R. Heaton§, P. Ni†, D. Pascoli‡, E. Siliotto‡, and G. Urso‡

‡ Dipartimento di Fisica and INFN Sezione di Padova
Via F. Marzolo 8, I35131 Padova, Italy

† Dept. of Computer Science, North Carolina State University
Raleigh, North Carolina 27695-8206

§ MCNC, PO Box 12889
Research Triangle Park, NC 27709-2889

Abstract

A new system based on the Blitzen parallel processing architecture, for application to second level triggering in high energy colliders, is described. The simulated performance of this system in real-time cluster finding is presented. The efficiency has been evaluated on a sample of CDF calorimeter data by comparison with off-line analysis. The system has also been tested in track finding with the TRD algorithm from the CERN/EAST collaboration.

Introduction

Parallel processing has been investigated as a means of handling the high rates and complex, multiple interactions foreseen during the future collider experiments. The challenge is to find interesting events from the vast quantity of data produced by detectors. Selectivity is expected to be implemented in stages with each stage, or "trigger level", refining selection by increasing research complexity. The first level of such a hierarchical computing structure is expected to have an output event rate on the order of 100KHz [2].

For the second level trigger many architectures have been proposed [8]. A very promising one is based on the use of embedded processors acting on the data of "Regions of Interest" (RoI) tagged by the first level trigger. A parallel processing array using the Blitzen device [3] seems to be a very good candidate, since detectors can be described as large arrays of cells which provide information that can be likened to images. These images may easily be mapped on an array of elementary processing elements.

The performance of a Blitzen array of 16×16 processing elements has been evaluated running two benchmark algorithms defined by the EAST collaboration at CERN ([4], [5], and [6]). These benchmarks were used to compare the results of various pipeline and parallel machines [2]. Simulated data sets provided by EAST were Monte-Carlo generated. After describing the Blitzen architecture, this paper gives the performance of the Blitzen machine for cluster energy finding using a

sample of CDF calorimeter experimental data.

Blitzen Architecture

The Blitzen device, implemented as a custom VLSI chip, is a building block for massively parallel computing systems [3]. The referenced paper has a complete description of the device so we present only the major features that will aid in understanding this paper. Each chip has an 8×16 array of bit-serial processing elements (PEs) that operate in SIMD mode, i.e., all of the PEs simultaneously execute the same instruction on different data sets.

Each PE has an arithmetic unit, a few registers, and 1 K bits of random access memory. PEs are interconnected with an X-shaped grid that provides communication between each PE and its eight nearest neighbors. I/O data transfers make use of 4-bit buses with 16 PEs per bus. This interconnection and I/O architecture makes it possible to build arrays of different shapes and dimensions from the chips.

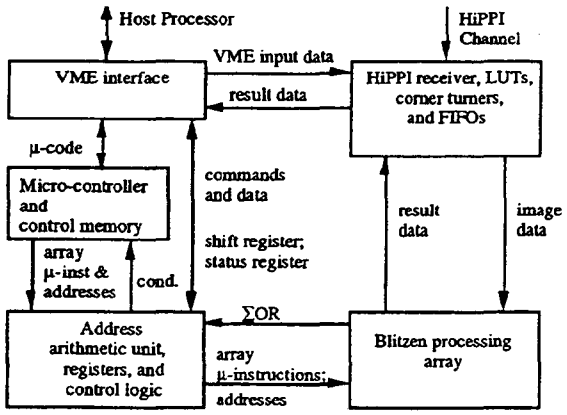


Figure 1. Prototype System Structure.

A prototype system has recently been developed [7]. Features of the system, shown in Fig. 1, include a high bandwidth HiPPI data interface, a VME interface for control functions and access to a host system, control logic, and a 16×16 array of processing elements using the Blitzen device. The input data path contributes to the algorithmic performance in that it contains, for optional use, both a lookup table and a data reformatting feature. The lookup table can convert incoming data in certain useful ways. For example, strings of equally weighted bits can be converted into coded values as part of the input flow. Data reformatting, or corner-turning, can be used to align data as it is delivered from the readout systems with the I/O structure of the processing array.

This system is implemented as a single board with a VME form factor. It is intended to be used in conjunction with a host machine. Programs can be transferred from the host to the control memory and execution initiated by the host. The parallel system can proceed independently of the host, yet it can communicate both data and control information asynchronously.

Current status is that the system is operational and has been tested with TRD track-finding, one of the two benchmarks mentioned earlier. Results are identical to those produced by simulation and reported in [5]. The rest of this paper reports performance and efficiency results of a parallel algorithm for cluster finding with CDF calorimeter data that were determined from a simulation of the system before the hardware was available.

The CDF Calorimeter

The calorimeter of CDF [1] is based on projective towers pointing to the nominal interaction region. It is divided into

three sections covering different regions in the polar angle θ with respect to the beam axis. The segmentation of the calorimeter that the second level trigger can access has equal intervals of rapidity $\eta = -\ln\tan(\frac{\theta}{2})$ and azimuthal angle ϕ with respect to the beam line. The intervals are $\Delta\eta = 0.1$ and $\Delta\phi = 15^o$. These towers are treated as distinct elements in the segmentation that results in a 44×24 matrix mapping of the full calorimeter.

The Off-line Cluster Finder Algorithm

The off-line cluster reconstruction algorithm first locates all towers with a transverse energy E_t above a threshold of $1GeV$. Every additional tower with E_t above $100MeV$ and located at $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2} < 0.7$ with respect to the centroid of the cluster is then associated. The centroid of this cluster is calculated and all the towers within ΔR are grouped together in a new cluster. This procedure is repeated until the cluster is unchanged.

The On-line Cluster Finder Algorithm

It is assumed that Regions-of-Interest tagged by the first level trigger are windows of 16×16 towers. Each tower provides values of both hadronic and electromagnetic energy deposition. We assign the energy values of one tower to one PE. This mapping allows the analysis of one event by processing tower data in parallel.

According to the topology of the CDF calorimeter, the energy of a cluster is assumed to be inside a sub-window of 5×5 towers. There are 12×12 positions of a 5×5 sub-window within a 16×16 window. Each of these positions represents a potential cluster. The cluster searching criteria is based on the value of energy deposition in the sub-window and in its central tower. The 16×16 PE array executes the following computational steps:

1. Sum the electromagnetic and hadronic energy of each tower.
2. Compute the total energy values of all 12×12 potential clusters. At the end of this step each PE contains the energy value of the sub-window of which it is the center, as well as the energy of its associated tower.
3. Look for the sub-windows and the related central towers with energy values above given thresholds $Th2$ and $Th1$ respectively.

Comparison of Results

The performance of the parallel on-line algorithm has been compared to the off-line analysis results, both applied to a sample of CDF jet data. Events from $p\bar{p}$ collisions at $\sqrt{s} = 1.8TeV$ were recorded with a minimum trigger requirement of a transverse energy E_t of $20GeV$ collected in a cluster. Such a trigger was mainly designed for tuning the calorimeter response to jets of low energy.

The parallel algorithm has been applied with three threshold values on the central tower energy and no condition on the cluster total energy value. The effect of the threshold is shown in the cluster multiplicity distributions reported in Fig.2. Part(a) shows the off-line results. Parts (b), (c), and (d) show the parallel algorithm results for $Th1 = 0.64$, 1.28, and $2.56GeV$ respectively. A qualitative agreement is reached with the intermediate threshold value, while a clear overestimate of the number of clusters is obtained with the lowest threshold. This effect is due to low energy clusters, as is

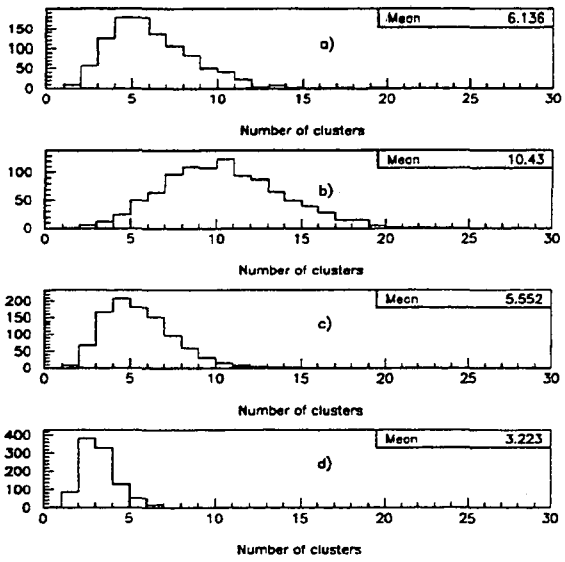


Figure 2. Cluster multiplicity distributions.

shown in Fig.3 where the energy distributions are presented for the off-line (solid) algorithm and for the three thresholds.

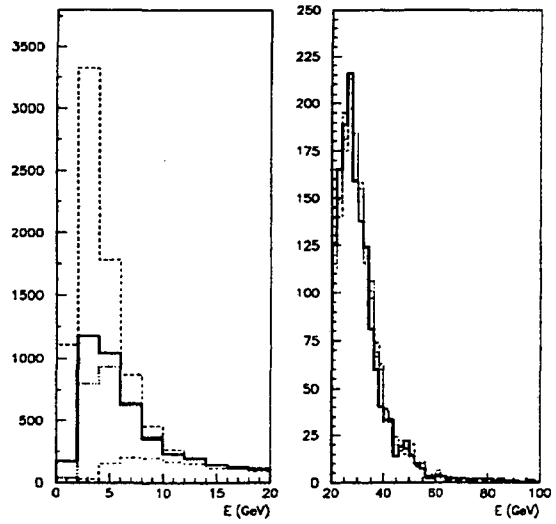


Figure 3. Cluster energy distributions.

A loss of efficiency is expected when the threshold is increased, however the multiplicity per event and the occupancy can produce critical situations in a real experiment for lower threshold values. We studied in detail the efficiency, matching both energy ($\Delta E_t < 3\sigma_{E_t}$) and position ($\Delta R < 0.7$) of clusters found by the parallel algorithm with respect to the off-line. The result is reported in Fig.4a where the efficiency is plotted versus the energy of the cluster for the three thresholds. As expected the lowest threshold of $0.64GeV$ produces the highest efficiency. We then try to manage the high multiplicity effect by increasing the threshold value $Th2$ of the total cluster energy. Fig.4b shows the efficiency for $Th1 = 0.64GeV$ with $Th2$ values of 0.64, 1.28, 2.56 and $5.28GeV$. The effect of imposing both thresholds is a sharper and higher efficiency curve and a possible control of multiplicity by optimizing the value of $Th2$.

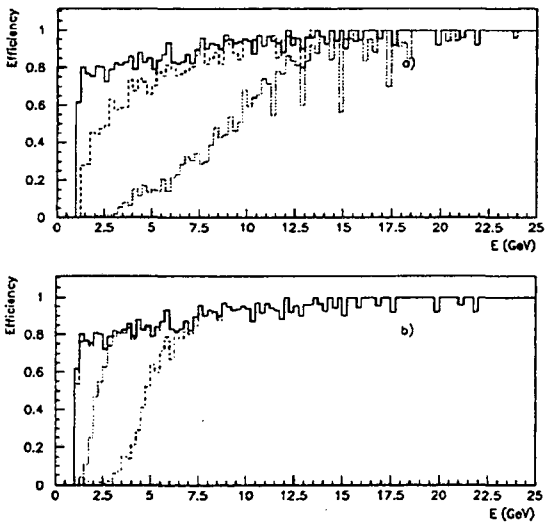


Figure 4. Clustering efficiency.

Execution Times

We give execution times for the algorithm in terms of instruction cycle counts. Since a Blitzen instruction is executed in one clock cycle, we can provide a time conversion according to the operating frequency of the Blitzen chips. The computational steps were outlined previously.

Activity	Cycles	time (μs) @20MHz	time (μs) @50MHz
input data	96	4.8	1.9
compute algorithm	458	22.9	9.2
output results	60	3.0	1.2
Total algorithm	614	30.7	12.3

We ran the same algorithm, written in C and optimized for sequential execution, on a few commercial machines. Execution times, reported in the table below, are for the computational part of the algorithm with no I/O. Blitzen times for the same computational work are also listed for comparison.

System	CPU	clock MHz	time μs
SUN IPX	SPARC	40	216
SUN SS10/30	SuperSPARC	36	105
DECsystem 5500	R3000	30	200
DEC 3000-500	DECchip21064	150	76
HP 9000/730	RISC 7100	66	85
HP 9000/735	RISC 7100	99	57
BLITZEN		20	23
		50	9

Conclusion

The reported results show that a signal processor based on a SIMD engine, designed originally for pattern recognition and image analysis, can reach the required performance in real-time, second level trigger applications. A key point is the good system balance between the array size and the required I/O bandwidth, that can be achieved with application specific hardware designed around Blitzen devices. The inclusion of lookup tables and corner-turning logic AS part of the single board system reduces the need for special routing logic in the readout architecture of the experiment. Other tests [2] performed on commercial processors gave less satisfactory results mainly because of the I/O limitations.

REFERENCES

1. F. Abe, et al., "The CDF Detector: an Overview", Nuclear Instrumentations and Methods, Vol. A271, 1988, pp.387.
2. J. Badier et al, "Evaluating Parallel Architectures for two Real-Time Applications with 100KHz Repetition Rate", IEEE Trans. on Nuclear Science, Vol. 40, No. 1, February 1993, pp. 45-55.
3. D. W. Blevins, E. W. Davis, R. A. Heaton, and J. H. Reif, "BLITZEN: A Highly Integrated Massively Parallel Machine", Journal of Parallel and Distributed Computing, Vol. 8, No. 2, February 1990, pp. 150-160.
4. S. Centro et al, "A Parallel Algorithm for Feature Extraction from Transition Radiation Detector Data: Benchmark Results Using the Blitzen Parallel Processor", CERN/EAST Note 92-11, CERN, Geneva, Switzerland, July 1992. Also TR DFPD 92/EI/28, Dept. of Physics, Univ. of Padova, Italy, July 1992.
5. S. Centro, E. Davis, P. Ni, and D. Pascoli, "Results of Second Level Trigger Algorithms Using the Blitzen Parallel Machine", CHEP '92, Annecy, France, September 1992.
6. S. Centro et al, "Results of Second Level Trigger Algorithm on Spacal Calorimeter Data Using the Blitzen Parallel Machine", CERN/EAST Note 93-06, CERN, Geneva, Switzerland, March 1993. Also TR DFPD 93/EI/14, Dept. of Physics, Univ. of Padova, Italy, March 1993.
7. S. Centro, et al, "Real-time Parallel Processing in High Energy Physics: Architecture of the Blitzen Data Acquisition System", Microprocessing and Microprogramming 40 (1994) 167-178.
8. Proc. of the Conf. on Computing in High Energy Physics, CHEP '92, Annecy, France, September 1992.

A SECOND-LEVEL TRIGGER CONCEPT BASED ON COMMUNICATING DIGITAL SIGNAL PROCESSORS

P. Wegner, U. Gensch, H. Leich
DESY Deutsches Elektronen-Synchrotron
Institut für Hochenergiephysik, Zeuthen, Germany

Abstract

A design for a second level-trigger system based on communicating signal processors is discussed. For the proposed HERA-B experiment at DESY-Hamburg different approaches for parallel Kalman filter track finding algorithms have been tested using an array of TMS320C40 processors. Possible hardware implementations of second level trigger architectures are presented including a special high bandwidth communication channel.

INTRODUCTION

In a multi-stage-trigger environment for future High Energy Physics experiments the second-level trigger architecture requires both high computing power and high bandwidth connections. New communicating signal processors like the Texas Instruments TMS320C40 (C40) [1] can fulfill this task. In the context of Research and Development activities for the proposed HERA-B experiment some parts of a possible second-level trigger have been tested.

HERA-B

The aim of this experiment [2] at the HERA proton ring will be to study CP violation in $B \rightarrow J/\Psi K_S$ decays. The B-mesons will be produced in interactions of high energetic protons with nuclei of internal targets.

To reach the required statistical significance the experiment has to handle several pA events per bunch, the bunch distance will be 96 ns and therefore the total rate will be about 30-50 MHz. To achieve large rejection factors and to retain large efficiencies for $B \rightarrow J/\Psi K_S$ a refined trigger for J/Ψ will have to be applied. Tasks of the trigger will be lepton identification, determination of momenta and an estimation of the dilepton invariant mass.

TRIGGERING

The first-level trigger is based on special hardware, and uses only restricted information. It will provide a rejection factor of about 10^3.
The second-level trigger will use data from tracking layers even inside the magnet volume. Using lepton tags from the first level and a vertex tag from the Si-

vertexdetector, and including the drift times an additional rejection factor of 10 can be obtained. A further reduction can be obtained if also data (electron probability) from the Transition Radiation Tracker (TRT) is used.

ARCHITECTURES

Track finding in the second-level trigger follows the recursive Kalman filter method[3]. The required decision time of a few hundred microseconds for a lepton pair requires the use of parallel architectures for computing the Kalman filter. For that purpose different concepts have been tested:

- DSP farming
- Pipelining
- Algorithmic decomposition.

Using a processor farm (similar to the third-level trigger approach known from various experiments) one track is processed by one DSP. This DSP therefore must have access to the data of all detectors involved in the second-level trigger schema, therefore the startup latency depends only on the communication time. A DSP farm provides an algorithm independent architecture which can be used if the processing time on one DSP is sufficient for the second-level trigger task. Then the number of processors needed for farms is much lower compared to pipelining.

Tracking with a DSP pipeline means that each processor is responsible for one detector part (eg. one tracking chamber). Every DSP has access to the data of only this detector part. A dedicated pipeline has to be used enabling the application of the recursive Kalman filter method, and is therefore very algorithm dependent. Each Kalman filter step corresponds to one pipeline step and therefore needs a transfer of the previous filtered state vector and the corresponding covariance matrix. The decompositon into different processing steps together with the communication time for every step leads to a large startup latency time before first results are available. From this follows that a continuous data stream is necessary to fill the pipeline properly.

Algorithmic decomposition can be a decomposition of one single step (eg. the separation of prediction and filtering in the case of the Kalman filter) inside a pipeline to achieve a higher granularity. Compared to the previously described pipeline architecture the required number processors increases as well as the latency time, but dependent on the time of the largest fine grained processor step an increase of the overall processing speed can be achieved.

Another kind of algorithmic decomposition is realized by connecting coprocessors to the DSPs are used. "Coprocessor" stands for additional DSPs with a link or shared memory connection or Field Programmable Gate Arrays (FPGAs) with a memory mapping interface to the DSP [6]. The latter could be useful if special bit operations on combined data words are needed which will take a long time on DSPs.

The output of the above communicating DSP systems (the track parameters) are combined with features from the other detectors build up a global decision. As proposed in [4][5] a special arrangement of C40 processors can be used for the calculation of the invariant masses of the lepton pairs, and additional global informa-

tion useful for second-level trigger decisions.
All described architectures have been testet in a restricted form using a VME-slave board equipped with four C40-TIMs which is connected to a SUN workstation via a SBus-VME adapter. The TIMs on the board are internally connected by C40-links with an unbuffered transfer rate of 20 Mbytes/sec. The clock rate of the on-TIM C40 is 40 MHz. Time measurements are done by using one of the C40 timers with a resolution of 100ns.

RESULTS

For the tests a Kalman filter algorithm derived from that used in the off-line pattern recognition has been applied. Five track parameters have to be determined to realize a three-dimensional track following. Therefore one has to handle a 5 element state vector and a 5x5 covariance matrix. The distributed programs have been written in mixture of C and C40-Assembler.
The input data have been taken from the Monte Carlo simulation of a foreseen HERA-B detector setup.
In the case of a processor farm consisting of 4 C40s the processing time for one Kalman filter step was about $65\mu s$ which gives a total processing time for 10 planes involved in the second level trigger of $650\mu s$. Naturally, during this time 4 tracks can be processed in parallel.
In a 4-stage pipeline of C40s connected via C40 links every Kalman filter step takes about $60\mu s$. The communication time for transfering the setup vector and the covariance matrix is about $6\mu s$. This gives a startup latency of about $264\mu s$ ($4 * 60\mu s + 4 * 6\mu s$, including the time for output communication), and a total processing time for a properly filled pipeline of about $66\mu s$. If again 10 planes are assumed to be necessary for the second level trigger, 10 processors are needed for one track. The latency runs up to $660\mu s$.
If the Kalman filter step is divided into the prediction and the filter itself, the prediction takes about $40\mu s$, and the filter about $12\mu s$. The communication time for state vector, covariance matrix, and some error values is about $7\mu s$. Based on 10 planes the reduction of the total processing time to $47\mu s$ leads to a latency of $940\mu s$ at a doubled number of processors needed and a more sophisticated algorithmic structure.

COMMUNICATIONS

Two architectures are considered for the hardware implementation of the second-level trigger :

- push architecture
- pop architecture

Push architecture means in this context that for each level-1 trigger decision the corresponding data of all electronic channels needed for the level-2 trigger decision are read out and transferred via fast communication channels into the global level-2 data buffer. Data necessary for trigger calculations (so called 'regions of interest' - ROI) is then selected and distributed e.g. via a router.
In the case of a pop architecture the level-2 data buffer is distributed within the read out crates. Only ROI data are selected and transferred on request from the processors of the farm, thus communication channels with lower bandwidth

are required. It will be sufficient to use standard C40 channels for the HERA-B experiment. To perform the acquisition of ROI data from the buffers within one crate a read-out controller with C40-link interfaces and VME master capability has to be used. A specific C40 link implementation with optical data transmission and pipelining prevents bandwidth degradation over long distances and provides a high level of EMI protection within a 'noisy' detector environment.

The pop architecture seems to be better adapted to the problem of Kalman filter implementation. Each Kalman filter step , i.e. tracking through a superlayer, is calculated by one processor. The various track candidates ($\sim$ 10) are treated in parallel on different processors.
ROI information is popped on request from the level-2 event buffers individually for each track.

CONCLUDING REMARKS

A system consisting of four communicating C40-DSPs has been used to test a possible second-level trigger scheme for HERA-B based on a tracking algorithm. The timing results obtained indicate that this kind of architecture could play an important role in triggering at future High Energy Physics experiments as HERA-B. Especially new developments as the Analog Devices ADSP-21060 SHARC, the Texas Instruments five processor video DSP TMS320C80 (MVP), or a much higher clock rate TMS320C40 successor show great promise of a doubling of processor speed and bandwidth during the next two years.

ACKNOWLEDGEMENTS

We are grateful to Siegmund Nowak and Rainer Mankel for providing the Monte Carlo data, and for discussions on the tracking algorithms.
We have also benefitted greatly from the work within the CERN/EAST(RD11) collaboration, and we would like to express our gratitude to Rudy Bock and Iosif Legrand for the helpful discussions on second-level trigger architectures.

REFERENCES

1. TMS320C40 User's Guide, Texas Instruments, 1992.
2. Albrecht et al., "An Experiment to Study CP Violation in the B System Using an Internal Target at the HERA Proton Ring",Letter of Intent, DESY-PRC 92/94, October 1992.
3. R.Frühwirth,Nucl. Instr. and Meth., A262(1987)444.
4. R.Bock,J.Carter,I.C.Legrand, "Real time, data driven architectures simulated in concurrent C++ for the LHC second-level trigger," CERN/EAST Note 93-15.
5. R.Bock,J.Carter,I.C.Legrand, "What can artificial neural networks do for the global second level trigger", CERN/EAST Note 94-08.
6. A.Borgers,F.Goldbach,D.Ockeloen, "Realtime, Frond-end Parallel Signal Processing using Texas Instruments' TMS320C40 Digital Signal Processors", talk given on the C40 Workshop at Utrecht University, December 1993.

The DØ Level 1.5 Calorimeter Trigger

James T. Linnemann
for the DØ Calorimeter Level 1.5 Project *
Michigan State University
Department of Physics and Astronomy
East Lansing, MI 48824 USA

Abstract

We present a trigger based on commercial DSP's currently being installed in the DØ experiment.

The DØ Trigger

The DØ trigger consists of a deadtimeless hardware phase (Level 1, [1]) which selects events for digitization, and a software phase (Level 2 [2]) using a processor farm. One of the scarcest resources in DØ triggering is the bandwidth into Level 2. Level 1.5 is an intermediate hardware trigger level designed to improve the quality of events chosen for digitization. Level 1.5 makes decisions in 10-200 μsec, but incurs deadtime while waiting for the decision.

The Level 1.5 Calorimeter Trigger

The physics goal for the Level 1.5 Calorimeter Trigger was to achieve a reduction of a factor of 5 in electromagnetic triggers with efficiency > 90%.

The technical goals included holding deadtime below 10 % (which, assuming a maximum 1KHz input rate, implied a decision time of 100 μsec or less), short construction time (which encouraged use of commercial hardware) and flexibility for future development (which encouraged a programmable processor).

The Level 1 decision is based on Et measurements for 1280 towers of $\Delta\eta$ x $\Delta\phi$ = .2 x .2 . Separate sums are available for the electromagnetic (EM) and hadronic (HAD) portion of the towers. Time constraints limit the Level 1 decision to consideration of single towers or of sums over large fixed-size blocks of towers. Both electrons and jets may split their energy across two or more towers. To maintain good efficiency, Level 1 thresholds for electrons and jets must be set much lower than the Et of the actual objects of interest. The Level 1.5 trigger can consider neighbors of towers above a threshold. The first algorithm implemented is for electrons. It applies a threshold to the 2x1 EM sum of a tower above the Level 1 threshold added to its highest Et neighbor. In addition it applies an isolation

*B. Gibbard (BNL), U. Heintz (Columbia), M. Narain (FNAL), J. Drinkard, A. Lankford, S. Pier (UC Irvine), A. Baden, S. Eno (Maryland), M. Abolins, D. Edmunds, S. Fahey, R. Genik, S. Gross, P. Laurens, J. Linnemann, D. Owen (MSU), G. Griffin, G. Snow (Nebraska), Z. Zhang (Stony Brook), D. Wirjawan (Texas A+M)

cut. It calculates a 3x3 (EM + HAD) sum around the Level 1 candidate tower and makes a cut on [2x1] / [3x3] > min. In typical operating conditions, this algorithm is expected to give a rejection of 3-10 for efficiency > 90%.

The Level 1.5 Calorimeter Trigger Hardware

Preliminary design studies made it clear that for ANY design, the key problem was access to the input data. The 2560 8-bit signals are spread across 10 Level 1 trigger racks. Our other design goals led us to consider a programmable processor. The algorithms under consideration are simple enough that many microprocessors have the necessary speed, but digital signal processors are attractive because they are also optimized for fast access to data. It was clear that the data must be available to the processors directly, as VME bus cycles are far too slow to meet the time requirement of 100 μsec. for the Level 1.5 decision. We chose a commercial VME board, the Hydra II by Ariel. This board carries 4 TMS320C40 DSP's. It was vital to our selection that 4 of the 6 I/O ports of each DSP were available for data transfer from outside the board. The remaining ports are used to ring-connect the DSPs on the board. There is adequate on-board local and shared fast memory. The processor runs at 50 ns per instruction and the communication ports transfer 1 byte per 50 ns. We also purchased most of the other Level 1.5 hardware needed to write Level 1.5 results into the data stream.

Transport of Data from Level 1 to Level 1.5

We chose to assign 11 DSP's to local processing of candidates (essentially one per Level 1 rack). This gives up to a 10-fold speedup due to parallelism. The local DSP's then report results to a global DSP. Each DSP handles Level 1 candidates from all 32 ϕ towers in a patch 4 η towers wide. To execute our algorithm, the DSP must also have information about neighbors of the candidates. This access is simplest if the DSP also has all needed neighbors locally. So we chose to send each DSP data for two neighboring racks and to assign the patches (each 1 rack wide) to the boundary between 2 racks. Special cases at the ends of the calorimeter were eliminated by providing fake Et=0 data for the nonexistent neighbors.

To transport the Level 1 data to the Level 1.5 trigger crate, driver cards were added to the existing hooks on the Level 1 cards. There is enough time available that each of the 32 Level 1 cards in a rack need not have its own cables to send data to Level 1.5. One sequencer card per rack controlled which driver owned the rack's cable, and a total of 10 cables carried data from the Level 1 racks to the Level 1.5 crate. The cables were 10m long and 2 bytes wide. The signals were sent as differential ECL, with 100 ns between data transfers.

A receiver in the Level 1.5 crate translated the signals to the CMOS required by the DSP communication ports and split them into two single byte signals (EM and EM+Hadronic) for each of two communication ports. These signals for each rack were sent to the two neighboring DSP's which required the rack's signals. Thus the full transfer takes place in roughly 13 μsec, or a rate of about 400 MB/s into the Level 1.5 trigger.

We decided to rebuild the Level 1 can-

didate list in Level 1.5, as access to the original Level 1 list was much more complex than access to the Level 1 input data. This implied that the Level 1 threshold settings had to be supplied to Level 1.5 as well.

The Level 1.5 Calorimeter Trigger Software

The Level 1.5 Calorimeter Trigger processing steps are: 1) collect all Trigger Tower energies from the Level 1 Calorimeter Trigger, 2) sequentially scan 128 Trigger Towers/DSP looking for candidates, 3) use an algorithm to confirm some of the selected candidates, 4) combine the results and form a decision based on a global analysis.

We refer to step 3 as the "tool" code, while the rest is independent of algorithm and we refer to it as frame code. The tool code comprises only 10% of the total source code. The frame code was written to support multiple algorithms in a single crate. All code was developed in DSP assembler, assisted by the commercial parallel debugger. The simulator (written in FORTRAN) is integrated with the Level 1 and Level 2 simulators. The simulator is used for verification of the DSP code (on samples of passed and failed events) and for estimation of efficiency and rejection. The trigger simulator is driven by the same ASCII file that controls the actual triggering online. Facilities exist for monitoring rejection rates online, and for histogramming the location of objects found in Level 1.5.

Status and Performance

Readout of 6 of the 10 racks is installed. Two racks have been fully debugged on test data into the complete Level 1.5 trigger crate. We anxiously await our first exposure to real events and the measurement of the actual mean Level 1.5 decision time, but bench tests to date are consistent with 85-100 μsec.

Conclusions

More than most HEP triggers, this project depended on purchased components. This allows a straightforward upgrade path as new DSPs of the family become available. Choice of DSPs as engines allows straightforward addition of a jet algorithm. The addition of a second crate is also straightforward, as the readout of the Level 1 data has sufficient fanout for many crates.

REFERENCES

1. M. Abolins, D. Edmunds, P. Laurens, J. Linnemann, B. Pi, "The Fast Trigger For The D0 Experiment" Nuclear Instruments and Methods in Physics Research, A289, 1990, pp.543-560. and

   M. Abolins, D. Edmunds, P. Laurens, J. Linnemann, B. Pi, "A High Luminosity Trigger Design for the Tevatron Collider Experiment in D0" IEEE Transactions on Nuclear Science, Vol.37, No.1, 1989, pp.384-389.

   S. Abachi et al., "The D0 Detector", Nucl. Instr. and Meth. A 338 (2-3) (1994) pp. 185 - 253.

2. J. Linnemann, "The D0 Software Trigger" Proceedings of Computing in High Energy Physics, 1992, Annecy France, p 199-201 (CERN, 1992).

IMPROVEMENTS IN THE CPLEAR TRIGGER AND DATA ACQUISITION SYSTEM

The CPLEAR Collaboration :
R. Adler[2], T. Alhalel[2], A. Angelopoulos[1], A. Apostolakis[1], E. Aslanides[11], G. Backenstoss[2], C.P. Bee[11], O. Behnke[17], J. Bennet[9], V. Bertin[11], F. Blanc[7,13], P. Bloch[4], Ch. Bula[13], P. Carlson[15], M. Carroll[9], J. Carvalho[5], E. Cawley[9], S. Charalambous[16], M. Chardalas[16], G. Chardin[14], M.B. Chertok[3], M. Danielsson[15], A. Cody[9], S. Dedoussis[16], M. Dejardin[4], J. Derre[14], M. Dodgson[9], J. Duclos[14], A. Ealet[11], B. Eckart[2], C. Eleftheriadis[16], I. Evangelou[8], L. Faravel[7,11], P. Fassnacht[11], J.L. Faure[14], C. Felder[2], R. Ferreira-Marques[5], W. Fetscher[17], M. Fidecaro[4], A. Filipčič[10], D. Francis[3], J. Fry[9], E. Gabathuler[9], R. Gamet[9], D. Garreta[14], T. Geralis[13], H.-J. Gerber[17], A. Go[3], P. Gumplinger[17], C. Guyot[14], A. Haselden[9], P.J. Hayman[9], F. Henry-Couannier[11], R.W. Hollander[6],E. Hubert[11], K. Jansson[15], H.U. Johner[7], K. Jon-And[15], P.R. Kettle[13], C. Kochowski[14], P. Kokkas[8], R. Kreuger[6], T. Lawry[3], R. Le Gac[11], F. Leimgruber[2], A. Liolios[16], E. Machado[5], P. Maley[9], I. Mandić[10], N. Manthos[8], G. Marel[14], M. Mikuž[10], J. Miller[3], F. Montanet[11], T. Nakada[13], A. Onofre[5], B. Pagels[17], P. Pavlopoulos[2], F. Pelucchi[11], J. Pinto da Cunha[5], A. Policarpo[5], G. Polivka[2], H. Postma[6], R. Rickenbach[2], B.L. Roberts[5], E. Rozaki[1], T. Ruf[4] L. Sacks[9], L. Sakeliou[1], P. Sanders[9], C. Santoni[2], K. Sarigiannis[1], M. Schäfer[17], L.A. Schaller[7], A. Schopper[4], P. Schune[14], A. Soares[14], L. Tauscher[2], C. Thibault[12] F. Touchard[4], C. Touramanis[9], F. Triantis[8], D.A. Tröster[2], E. Van Beveren[5], C.W.E. Van Eijk[6], S. Vlachos[2], P. Weber[17], O. Wigger[13], C. Witzig[17], M. Wolter[17], C. Yeche[14], D. Zavrtanik[10] and D. Zimmerman[3].

[1]University of Athens, [2]University of Basle, [3]Boston University, [4]CERN, [5]LIP and University of Coimbra, [6]Delft University of Technology, [7]University of Fribourg, [8]University of Ioannina, [9]University of Liverpool, [10]J. Stefan Inst. and Phys. Dep., University of Ljubljana, [11]CPPM, IN2P3-CNRS et Université d'Aix-Marseille II, [12]CSNSM, IN2P3-CNRS, [13]Paul-Scherrer-Institut(PSI), [14]DAPNIA/SPP, CE Saclay, [15]Royal Institute of Technology, Stockholm, [16]University of Thessaloniki, [17]ETH-ITP Zürich,

presented by François TOUCHARD

Abstract

The CPLEAR experiment studies CP violation in the (K^0, $\overline{K^0}$) system. We present the trigger architecture and the data acquisition system, as well as forthcoming upgrades

INTRODUCTION

The CPLEAR experiment aims to measure the parameters of CP violation in the (K^0, $\overline{K^0}$) system. Its originality – when compared to other experiments using high energy beams of neutral kaons – resides in the use of strangeness tagged particle beams. By selecting the annihilation channels $p\bar{p}\rightarrow K^0K^-\pi^+$and $p\bar{p}\rightarrow\overline{K^0}K^+\pi^-$, the strangeness of the neutral kaon is given by the charge of the associated charged kaon. The decay of the neutral kaon is studied in the 2 π ($\pi^+\pi^-$and $\pi^0\pi^0$), 3 π ($\pi^+\pi^-\pi^0$) and semileptonic ($\pi\ell\nu$) channels . Unfortunately, the branching ratios of the interesting production channels are very low (2 10^{-3}) and the required statistics of neutral kaon events is high in order to have a sufficient accuracy on the CP violation parameters. The experiment must therefore be able to stand a high $\bar{p}$ rate (1 MHz), the trigger must have a high rejection ($\approx$ 1000) and the data acquisition system must be able to record up to 1000 events per second.

The detector has a conventional cylindrical topology [1]. The 200 MeV/c $\bar{p}$ beam from the CERN LEAR storage ring annihilates at rest in the gaseous hydrogen target. Moving radially outwards, one first finds the tracking devices : 2 proportional chambers (PC's), 6 drift chambers (DC's) and two layers of streamer tubes (ST's) for a fast online z information. Exterior to the ST's are the Particle Identification Detectors (PID's) : 32 modules, each consisting of a threshhold Čerenkov counter sandwiched between two scintillators. The Čerenkov liquid has been chosen so that one can discriminate fast particles (pions) from slow ones (kaons) in the phase space of the experiment. Finally, one finds an electromagnetic calorimeter made of lead and streamer tubes. The entire detector is mounted inside a dipole magnet. The total number of channels is approximatively 70'000 and the event size lies between 2 and 2.4 kbytes, depending on the event topology.

TRIGGER

The trigger is multilevel. It uses algorithms the sophistication of which increases as the selection goes on.

Decision logics

Table 1 summarizes the successive criteria, the execution times and the rejection rates. In the first step (*Early Decision Level*), charged kaons are identified by a $S\bar{C}S$ pattern in the PIDs (i.e. no light in the Čerenkov while the two scintillators in the same sector have been hit). One also demands at least one more charged particle (hit in the inner scintillator). When these patterns have been found, a strobe is sent to all front ends to start digitization. Slow pions faking kaons are rejected by a fast cut on the transverse momentum. The p_T cut logic uses coincidences between adjacent wires in DC1 and DC6.

The pattern recognition is performed in two steps. First, the *Intermediate Decision Level* prepares the fine grain tracking. It receives data from the previous trigger stage and hit patterns from sectors of chambers. These sectors are groups of wires, 64 for every chamber and situated in front of the 32 PID modules. With this data, one can discriminate between primary and secondary tracks. Primary tracks have hits in the proportional

chambers and are candidates for particles produced at the annihilation vertex. Secondary tracks are candidates for late neutral kaon decay. A cut is performed on the number of tracks.

The fine tracking is performed in a dedicated hard-wired processor : *HWP1*. The first stage of HWP1 performs the actual tracking by scanning tracks from the scintillators towards the target and using the wire information of the hit sectors. A cut is performed on the number of tracks. Then, tracks are ordered in registers according to their identity (kaon candidate, primary, secondary pions). A new cut is performed on the number of tracks. The next stage of the processor computes the track parameters : curvature ρ, initial angle ϕ, distance of minimal approach ϵ, z coordinate at the ST's radius. It also derives the longitudinal and transverse components of the particle momenta. The last stage of the processor computes the sum $p_K + p_\pi$ and applies the cut $p_K + p_\pi >$ 750 MeV/c.

In parallel with the kinematical cut, the processor *HWP2* refines the particle identification. It gets the now digitized data from the PID front ends and compares the energy deposited in the inner scintillator (dE/dx), the difference in the time of flight of the primary particles (ToF) and the number of photoelectrons created by the kaon candidates in the Čerenkov counters to the expected values calculated by using the tracks parameters.

The last processor in the trigger decision, *HWP2.5*, works on events having only two charged tracks, i.e. candidates for a neutral kaon decay into two π^0. No correlation is made at this point between showers and charged tracks so that it is not possible to discriminate neutral showers from charged ones. A minimum number N_0 of showers is demanded. The decision cuts the K_L^0 events which have not decayed inside the volume of the detector (98% of them). It also improves the event reconstruction quality by imposing a minimum number of neutral showers.

Processor	Act on	Decision	Get info from	Execution time	Rate reduction
EDL	K candidates	N(K)>0 N(tracks)>1	PID	60 ns	4.1
	p_T	$p_T > 280 MeV/c$	DC1,DC6	400 ns	3.1
IDL	primary tracks secondary tracks	N(tracks)	PC1,PC2, DC1,DC2,DC5, DC6 sectors	80 ns	1.7
HWP1	fine tracking	N(tracks)	PC's, DC's, ST's	1.9 μs	1.4
	tracks parametrization	N(tracks)	detailed bus info from previous stage	500 ns	1.3
	kinematics	$p_K + p_\pi$	previous stage	500 ns	1.5
HWP2	particle identification	dE/dx ToF N(phe)	PID's HWP1	1.9 μs	6.4
HWP2.5	calorimeter showers	$N_{showers} > N_0$	calorimeter front ends	17 μs	2.9 (N_0 = 6)

Table 1. Trigger decisions

It should be noted that we have adopted a sectorized dialogue between the trigger processors and the detector front ends. Moreover, there is no handshake procedure in the dialogue : the processor puts an address on the bus and receives after a predefined number of clocks the address (as a cross-check) and the required information. This procedure gives the best compromise between speed and reliability.

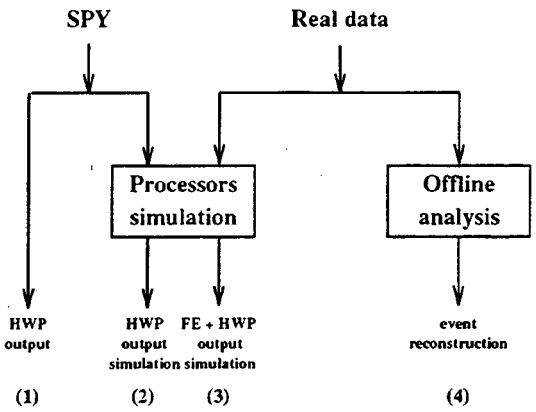


Fig. 1 Trigger monitoring

Monitoring

The monitoring of the trigger is performed via the normal readout channel. Data from the processors is sent to the readout system through a custom-built interface named SPY. Data consists of the hit maps, the front end data received by the processors as well as the intermediate and final results of the processing.

Local monitoring is performed in the local readout processor in the same way as for other detectors. The most powerful tool for the trigger monitoring is the use of a sophisticated simulation of the processors. The detailed behavior of the processors is reproduced in a programme which accepts both the SPY data and the full event as input. As will be seen later, full events (including SPY data) are continuously sent to the Online Vax cluster where they are treated by dedicated programmmes (Fig. 1) which calculate the functionality (1 *vs.* 2), the efficiency (1 *vs.* 3) and the resolutions(1 *vs.* 4) of the trigger processors. Those quantities are monitored and are available on line.

Status and future

All the described processors are now implemented and have satisfactorily worked since the middle of 1992. Two new decisions are in preparation and will be tested and eventually included for production in the early days of the data taking period beginning in May 1994.

• Microchamber

A very large fraction of the background comes from direct annihilation of $\bar{p}$ into four charged pions : $p\bar{p}\rightarrow\pi^+\pi^-\pi^+\pi^-X^0$. This background can be reduced by cutting on the event multiplicity as close as possible to the primary vertex. To do this, we are going to modify the target area by inserting a micro proportional chamber. This chamber is 120 mm long and 30 mm diameter. It has 96 wires and strips at 90° and 42°. The maximum drift time is 30 ns, allowing a fast decision. We expect to improve the total rejection of the trigger by a factor 3 to 4.

• Kinematic event filtering

A part of the background consists of events with some neutral pions : $p\bar{p}\rightarrow\pi^0K^+\pi^-\overline{K^0}$, $p\bar{p}\rightarrow K^+K^-\pi^+\pi^-n\pi^0$, $p\bar{p}\rightarrow\pi^+\pi^-\pi^+\pi^-n\pi^0$. We intend to reduce it by a kinematic event filtering using a neural network algorithm acting on all

four tracks events. The inputs of the network are the two components of the momentum, p_T and p_z, and the initial angle of the tracks ϕ. The network has been trained using Monte Carlo data. Its efficiency has been evaluated with both Monte Carlo and filtered real data. We have found an acceptance of good events of 82% while the total rejection on 4 track events was improved by a factor 2. The network has been implemented on a DSP processor, the 96002 from Motorola in a VME environment. The execution time is 38 μs per event. The number of good events per second does not increase, but the number of written tapes is reduced by a factor 2. One will therefore gain on the offline processing or have the possibility to relax some of the trigger constraints.

DATA ACQUISITION SYSTEM

The Data Acquisition System of the experiment has already been described in CHEP'92 [2]. We shall therefore only recall its main features before presenting the forthcoming upgrade.

The general layout of the system is presented in Fig. 2. The hardware is based on the *Valet+* system designed by CERN and built around the FIC8230 (68020 CPU) card from CES. Data is read out from front ends through various busses (VME, CAMAC, FASTBUS). It is then DMA'd from the readout processors to dual port memories in the crate of the Event Builder via point to point VSB connections. Note that the Fastbus front end of the Calorimeter is directly read by the Event Builder via a Fastbus to VSB interface (STR 330 module from Struck). The event is assembled in a Zebra compatible structure by the Event Builder which dispatches blocks of events to downstream clients processors via a VIC bus : TAPE which serves a IBM 3480 compatible cassette drive, MSP which sends events to the Online Vax Cluster for monitoring and FARM which feeds a transputer farm.

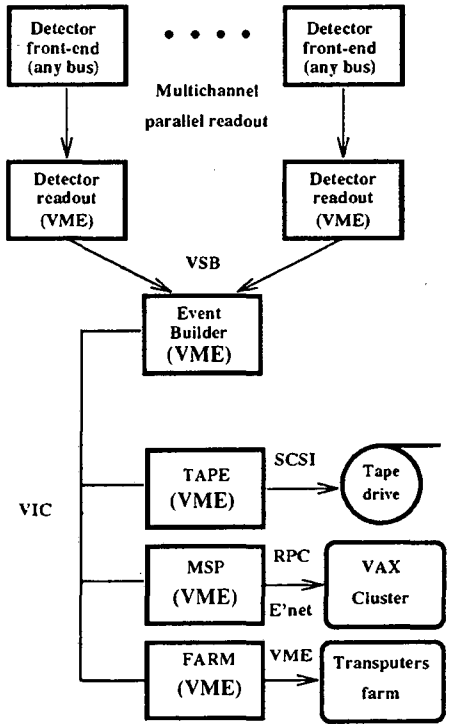


Fig. 2 DAQ layout

On the Vax side, we use the *MODEL* package developed at CERN for the LEP era. The operating system is VMS 5.5. The cluster is composed of a microvax 3800 and 8 Vaxstations 3100 or 4000-90. Run and trigger configuration, alarm message handling, run control and slow control are all handled from the cluster.

Monitoring

Monitoring is performed at various levels :

- in individual readout processors, which can analyse data from individual detectors
- in the MSP Valet+ which can perform consistency checks of the event structure

• on the Vax cluster : events received from MSP can be transfered to the Cluster via Ethernet and injected in a data stream built on the MODEL *MBM* package. They are extracted and analyzed by clients such as the online event display, a trigger monitoring programme running the trigger simulation and a data quality analysis programme. All accesses to the results of the analysis, consisting of histograms and N-tuples stored on the Valet+ system or the Vax cluster, are centralized in a customized version of PAW named CPPAW.

Upgrade

The readout busy time is the main limitation of the DAQ system : the currently used Struck 302 I/O interface between the front ends and the readout processors has no FIFO, no local memory and allows a maximum transfer speed of 0.7 Mbytes/s. A simple – from the front ends point of view – improvement has been found by using the CES RIO module based on a MIPS R3051 processor [3]. This module has a local memory of 4 Mbytes. It has been customized to have two 1024 word deep FIFO registers and to mimic the behavior of the Struck 302 so that no modification of the front ends is necessary. The readout time will be reduced from 3 μs/word to 1.2 μs, therefore allowing a maximum transfer speed of 2 Mbytes/s. This new configuration will be tested during the first 1994 data taking period.

Transputer farm

The transputer farm installed downstream of the Event Builder is a collaboration between CPLEAR and the European Union project GP-MIMD which aims to develop a Scalable Data Acquisition System using Transputers [4].

A configuration with 32 T800 transputers has been tested in run for feasibility. Of these transputers, 10 were running online the full CPLEAR offline production for 2 weeks in 1993. The other transputers worked as data and recording device drivers. Filtered exabyte cassettes have been written directly at a reduced sampling rate of 10 Hz. In the future, 64 T9000 transputers will be added to the configuration which should be able to cope with the full rate of data taking.

Acknowledgements

I would like to thank all the engineers who have been involved in this work : D. Sacker from Basle, C. Engster, W.G. Heyes, C. Jacobs, L. Van Königsveld, E.J. Watson and H. Wendler from CERN, A. Calzas, B. Dinkespiller, Y. Gally from Marseille, N. Karkhour, D. Linget from Orsay, J.P. Bard, J.C. Michau from Saclay.

REFERENCES

1. L. Adiels *et al.*, Test of CP violation with K^0 and $\overline{K^0}$ at LEAR, Proposal CERN-PSCC/85-6/P82
2. C.P. Bee *et al.*, Proceedings of the CHEP'92 conference, CERN 92-07
3. B. Eckart *et al.*, 8th Conference on real time computer applications, Vancouver, June 1993, report TRIUMPH 93-1
4. M. Ward *et al.*, Interim report for the scalable data acquisition system, ESPRIT project P5404 GP-MIMD deliverable

THE CDF ULTRANET DATA ACQUISITION SYSTEM

J. Patrick, H. Areti, G. W. Foster, G. Goeransson, U. Joshi, R. Harris, E. Kennedy,
K. Maeshima, J. Pangburn, M. Schmitz, K. Treptow, S. Zimmerman, T. Zmuda
Fermi National Accelerator Laboratory, Box 500, Batavia, IL 60510

T. Daniels, P. Maksimovic, R. Mattingly, S. Pavlon, P. Sphicas, S. Tether, D. Vucinic
MIT, 77 Massachusetts Avenue, Cambridge, MA 02139

F. Abe, E. Hayashi, Y. Morita, M. Nomachi
KEK and University of Tsukuba, Tsukuba-shi, Ibaraki-ken 305, Japan

K. Biery, P. Musgrave, K. Ragan, K. Strahl
McGill University, 3600 University Street, Montreal, Quebec, Canada H3A 2TB

J. Conway, J. Doroshenko, T. Devlin, L. Groer, R. Kennedy, C. Loomis, J. Mueller, T. Watts
Rutgers University, Frelinghuysen Road, Piscataway, NJ 08854

H. Kim, A. Holscher, P. Sinervo
University of Toronto, 60 St. George Street, Toronto, Ontario, Canada M5S 1A7

K. Sliwa, M. Timko
Tufts University, 4 Colby Street, Medford, MA 02155

Abstract

The high level portion of the data acquisition system of the CDF experiment operating at the Fermilab Tevatron proton-antiproton collider has been upgraded for the current data taking run. This new system is based on the commercial Ultranet network. VME based CPUs transmit front end data through an Ultranet hub to a series of Silicon Graphics server systems where high level algorithms make the final trigger decision. The system architecture provides for simultaneous data transfers between multiple front end and multiple SGI systems. Also, extensive use is made of off the shelf commercial components.

INTRODUCTION

CDF is one of two detectors at the Fermilab proton-antiproton collider. First operated in 1985, the accelerator luminosity has reached 10 times the original design, and a series of improvements is expected to bring another factor of 10 increase over the next 6-8 years. In order to cope with the higher resulting event rate, the original data acquisition system must be upgraded. This upgrade is further motivated by reliability and maintenance problems posed by the numerous

custom modules in the system designed over 10 years ago, especially considering the remaining life of the experiment.

Original CDF Data Acquisition System

The original CDF data acquisition system is described in references [1] and [2]. Very briefly, front end electronics is contained in either FASTBUS crates (about 60) or custom RABBIT crates. Custom readout controllers (SSPs for FASTBUS, MXs for RABBIT) collect data from the local crates. In response to a hardware trigger, an Event Builder collects data from the readout controllers over FASTBUS Cable Segments and delivers it via a FASTBUS – FNAL Branch Bus – VME connection to a "farm" of Silicon Graphics server systems. Here FORTRAN algorithms make the final trigger decision. Accepted events are read by Branch Bus – FASTBUS links by one or more VAX systems and written to tape. Maximum throughput is about 25 Hz (200 KB/event) into Level 3, limited by large tranfer setup overheads in the Event Builder and message overheads in the method of data flow control.

Requirements for Upgrade system

To deal with expected improvements in the accelerator luminosity, the new system is required to support an event rate into Level 3 of at least 100 Hz and be expandable to higher rates in the future. In addition, to minimize in-house maintenance it should make maximum use of commercial components where possible. Furthermore the system must interface to the current FASTBUS and RABBIT front end systems, as well as adapt to future upgrades in the front end electronics. And whereas any interruption in operation was not considered acceptable, the new system must completely coexist with the old system, with either being selectable by software on a run by run basis.

Overview of Upgrade System

A diagram of the new architecture is show in Fig. 1. Local FASTBUS front end data are collected by new FASTBUS Readout Controllers (FRCs) replacing the SSPs. A series of six VME based CPUs collects data from the FRCs over a new custom "Scanner Bus". These CPUs then write the data through an Ultranet Hub to an upgraded farm of Level 3 processors. Accepted events are sent via Ultranet to a dedicated Consumer Server system where they are logged, and a sample distributed to workstation based monitor processes via FDDI. Data flow control is provided by a dedicated Scanner Manager communicating with the rest of the system via a reflective memory network. The Ultranet hub is logically a switch, so that that data are transfered between multiple Scanner CPUs and Level 3 systems simultaneously. Complete events are not assembled until they reach Level 3.

FASTBUS Readout Controllers

The FRCs are new custom devices that collect data from local FASTBUS front end devices. They are based on the MIPS R3000 processor, with FASTBUS master and slave interfaces based on Xylinx programmable gate arrays. VxWorks, from Wind River Systems serves as real time operating system is used. An auxilliary card contains an interface to the Scanner Bus. Triple ported video ram

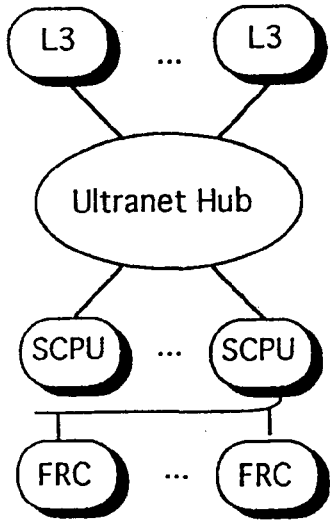


Figure 1. Architecture of the new system.

is used to provide high speed simultaneous memory access from the CPU, and the FASTBUS and Scanner Bus ports.

Scanner CPU

Data are read from a subset of FRCs by a set of six VME based Synergy SB/063 68030 processors running the VxWorks operating system. Transfer is over a custom developed 16 bit parallel, multidrop bus based on RS-485 technology known as the "Scanner Bus". A VSB/VME dual access memory receives data from the Scanner Bus via a VSB interface. From this memory the data are then transfered to Level 3 over Ultranet by a VME based interface.

Ultranet

Ultranet is a high performance, propietary, commercial network from CNT/Ultranet of Minneapolis, MN. It is a "star" architecture, with point to point links to a central hub. Serial links transmit data at a rate of 256MBits/sec. Transfer rates between VME based interfaces are limited to 13 MBytes/sec, excluding overheads. TCP/IP protocol with an enlarged packet size and BSD socket API is used. Part of the protocol processing is done in the interface reducing the load on the host CPU. While overheads are significant, they can be substantially hidden by having multiple transfers in progress simultaneously. Maximum aggregate bandwidth has been measured to be about 50 MBytes/sec for a single backplane hub. However the hub can be configured with multiple sub-backplanes increasing the potential throughput provided all transfers do not go through the central backplane.

Scanner Manager

Control of the data flow is done by tasks running on a dedicated VME based CPU called the Scanner Manager. This process ensures for example that all fragments of a particular event are delivered to the same Level 3 processor. Communication with the Scanner CPUs and Level 3 systems is via a dedicated reflective memory network (Scramnet, from Systran Corp. of Dayton, OH). Memories are located in each Scanner CPU and Level 3 VME crate and connected by fiber optic cable. A write operation to a word in one module is automatically propagated to all modules in the network using a propietary protocol. This provides a very fast, low overhead means of communication, as well as being completely independent of the data flow medium.

Level 3 Trigger

The Level 3 processor farm runs a similar algorithm as the previous system to decide which events should be logged. Events are classified into several broad

categories which determine their offline production priority, including express line and other data streams. In addition, Level 3 must now receive the 6 separate Scanner CPU fragments and build the complete event as previously done by the Event Builder. The internal control system has been substantially modified to interface to the rest of the upgrade system. Also, to provide sufficient processing capacity to deal with the larger input event rate, four new Silicon Graphics 8 processor Challenge systems have been added bringing the total compute power to over 3200 VUPs.

Consumer Server

Accepted events are sent via Ultranet to a dedicated SGI system. There, the Consumer Server process distributes them to local data logging processes as well as remote monitoring processes via TCP/IP and FDDI. Consumer processes may request events based on their classification by the Level 3 algorithm. Events are logged to local SCSI disks and spooled to 8mm tape asynchronously. A total logging rate of approximately 2 MB/sec, corresponding to approximately 20 Hz, can be supported. Typical rates for normal data taking are 6-8 Hz.

Performance and Status

Tests of the system as currently configured with approximately the expected event size give a maximum rate into Level 3 of approximately 110 Hz with a null Level 3 control system, and somewhat less with the full Level 3 system. However to keep the dead time due to front end digitization to a reasonable level, the system will be limited to around 50 Hz for the current run. Thus little effort has so far been expended in maximizing this rate. The new system has been use for part of the time and is expected to become the default for data taking very soon.

Summary

The CDF data acquisition system has been upgraded with a new system based on commercial VME based CPUs and Ultranet network. The system has a substantially higher throughput than the previous FASTBUS based system. The current data taking run is expected to end in mid-1995. At that time, all CDF front end electronics will be replaced, and the system upgraded further to deal with that and also improve the rate capability by another factor of 2.

ACKNOWLEDGEMENTS

This work was supported by the US Department of Energy and National Science Foundation, the Canadian Natural Sciences and Engineering Council, and the Japanese Ministry of Science, Education and Culture. We gratefully acknowledge the contributions of the the Fermilab Computing Division to many aspects of this upgrade. We also thank Atlantic Aerospace Electronics Corporation and DESY for providing Ultranet software for the VxWorks and SGI Challenge systems respectively.

REFERENCES

1. E. Barsotti, *et. al.*, " FASTBUS Data Acquisition for CDF " Nucl. Inst. Meth. A269, 82 (1988).
2. D. Quarrie, *et. al.*, "The CDF Online System" Proc. Oxford CHEP, Comp. Phys. Comm. 57, 325 (1989).

The CHORUS Data Acquisition System

Giuseppe Carnevale
Università degli Studi di Napoli "Federico II" - CERN
Beverley Friend - CERN
Jaap Panman - CERN
Fabio Riccardi
Università degli Studi di Napoli "Federico II" - INFN Napoli - CERN

Abstract

In this paper we present the data acquisition (DAQ) system of the CHORUS experiment at CERN. We concentrate our attention on the hardware and software architecture of this system, especially on the inter process communication system and in the way it was implemented, that constitutes one of the most innovative aspects of this DAQ design.

Introduction

In this paper we describe the data acquisition (DAQ) system of the CHORUS experiment [1], which will take data from April 1994 to the end 1995 at the SPS Wide Band Neutrino Beam facility of CERN (European Organization for Nuclear Research, Geneva, Switzerland).The aim of the experiment is to search for neutrino flavor mixing.

The search for ν_μ-ν_τ oscillations will be performed by detecting the appearance of a tau neutrino in a pure muon neutrino beam. As a direct observation is impossible, it is necessary to detect the decay of particles that are produced in the interaction of ν_τ in the detector target.

The Detector

The detector target consists of a stack of 800 kg of nuclear emulsions, followed by a fiber tracking system for the precise (1 mm^3) determination of the vertex of the neutrino interaction from the particles that come out of the emulsion stack. A high resolution calorimeter and a muon spectometer will allow performing kinematical cuts on the above mentioned reaction outcomes.

The emulsions will be exposed to the neutrino beam for about a year, and then removed and scanned with an automatic microscope scanning device, that integrates the information coming from the tracks on the emulsions with those recorded on-line by the DAQ system, allowing reconstruct the event geometry with a spatial resolution of 1 μm, that is required to spot the tau decay signature. The emulsion scanning time is estimated to be about a year.

DAQ Requirements

The bulk of the DAQ data will come from the CCD based readout system of the fiber tracker. It consists of 58 CCD cameras, for a total of about 9 million pixels. We expect to take at most two neutrino events per SPS burst, that is about 36

MBytes of CCD data in 6 ms. Two subsequent neutrino bursts are interleaved by 2.5 sec in the SPS cycle, that has a period of about 14 sec.

The calorimeter and the spectrometer will produce about 15 KByte of data per neutrino burst.

The DAQ Hardware Architecture

The front-end of the CHORUS DAQ system is made of 38 different CPUs, running the OS-9 real time kernel plus the REMOS object oriented multiprocessor real time environment, collaborating to the task of reading the digitized signals coming from the detector, collecting, validating, compressing, and eventually streaming them onto a storage device.

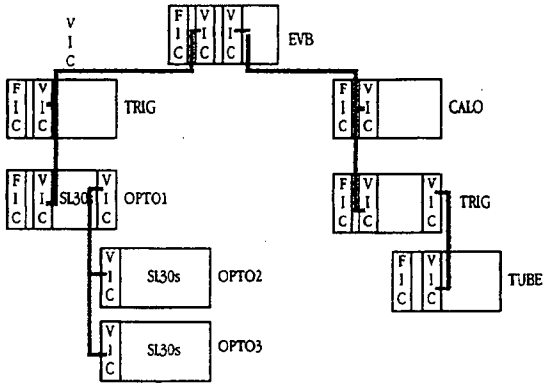


Fig. 1: The CHORUS DAQ Hardware Architecture.

Each main section of the detector system, tracker (OPTO), trigger (TRIG), calorimeter (CALO) and spectrometer (SPEC), is equipped with a local DAQ system. All these sub-systems are VME, VSB and CAMAC based, interconnected via a standard VME Inter Crate (VIC) bus (CES VIC 8251) in a tree like structure (see fig. 1). All local systems are equipped with a CES FIC8234 VME controller, connected to a main Event Builder (EVB) system, that coordinates all their activities. The EVB and the OPTO FICs are equipped with a double CPU for improved performance.

The OPTO subsystem is equipped with an array of 29 Eltec SL30 VME image processors, that read the image data from the 58 cameras' frame buffers, compress (10:1), format and ship them to the local master processor (OPTO).

The EVB is then linked to a cluster of six IBM RS 6000 workstations running the UNIX operating system, that are used for the Run Control user interaction and the on-line event display monitoring.

The DAQ Software Architecture

All the DAQ code has been written using the C++ programming language. With the object-oriented facilities of this language, we built a library of objects that integrates the operating system calls dealing with shared memory segments and semaphores, with those offered by our multiprocessor run-time support.

REMOS is made of a collection of C++ classes interfacing user application to a server task that dispatches messages across the VIC bus network (see fig. 2).

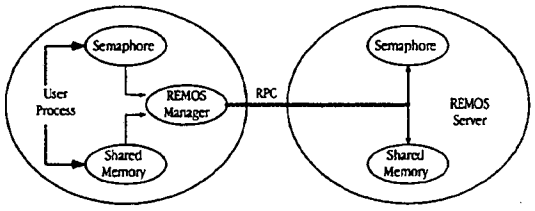


Fig. 2: The REMOS object oriented IPC.

As a basis of all our system we designed a special OS9 driver that allows user processes to install interrupt service routines (ISR). They can synchronize with interrupt requests and the data that the ISR collects at interrupt time is automatically buffered and delivered to the user process at synchronization time.

In this way the REMOS server and the communication system have been designed using the so called *micro kernel* approach, that avoids to use kernel based drivers, in favor of user tasks, allowing to achieve true real time performance. The fact that all our *drivers* and *servers* are user level processes makes them preemptible in any moment, preventing them from blocking the system during I/O.

A companion paper [2] presented at the same conference covers in detail the features of this environment.

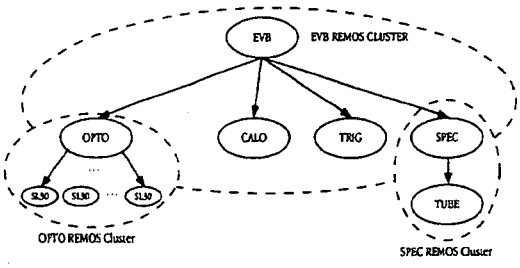


Fig. 3: The software architecture of the CHORUS DAQ system.

To optimize the performance of the VIC bus, we organized the whole DAQ hyerarchically in a master/slave fashion (see fig. 3), so that the bus is owned by only one system (the master), eliminating any time consuming bus contention.

To make an efficient and maintainable DAQ system we defined a strong computation paradigm: a data flow architecture that subdivides the system in a set of simple tasks, exchanging data among them and synchronizing through data availability. We devised a set of buffered ports allowing to construct a clean mechanism for inter-process communication, automatically matching the different process speeds. The presence of data in the input buffer and the availability of free space in the output buffer are used to schedule the activity of these processes, maximizing system's performance.

A message exchange protocol, based on this communication system, is used by the EVB to coordinate the activities of the subsystems, and to collect the relevant information on the status of the DAQ.

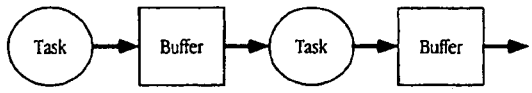


Fig. 4: Outline of the CHORUS DAQ buffer management scheme.

The run control program, on the Event Builder CPU, communicates with a graphical interface program on the UNIX cluster, that handles the interaction between the user and the DAQ system. A sample of the raw on-line data is sent to the UNIX cluster for real time detector monitoring, and to histogramming tasks on the local systems.

Conclusions

CHORUS is a middle sized physics experiment, but it features many of the DAQ requirements that characterize some of the future large collider experiments. Therefore it constitutes an ideal test bench for experimenting and validating a set of engineering techniques that can be fruitfully used in many other similar contexts.

References

1 N. Armenise et al., CHORUS Collaboration, CERN - SPSC/90-42; M. de Jong et al., CHORUS Collaboration, CERN - PPE/93-134

2 G. Carnevale et al, "REMOS: A Portable Object Oriented Environment for Multiprocessor Real Time Applications", to be published in the proceedings of the CHEP'94 conference, San Francisco, USA, 1994.

The RD13 Data Acquisition system

Aguer M[4], Ambrosini G[1], Bee C.P.[2], Buono S[2], DuvaL P.Y. [3], Etienne F[3], Ferrari R[2],
Ferrato D[3], Fumagalli[1], Huet M[4], Jones R[2], Mapelli L[2],
Le Van Suu A[3], Mornacchi G[2], Polesello G[1], Prigent D[2], Qian Z[3], Rondot C[3], Sanchez-Corral E[2],
Skiadelli M[2], Spiwoks R[2], Tisserant S[3], Touchard F[3]

1: Dipartimento di Fisica dell'Universita'e INFN Sezione di Pavia, Italy

2: CERN, Geneva, Switzerland

3: Centre de Physique des Particules de Marseille, IN2P3, France

4: Departement de Physique Nucleaire - STEN, C.E. Saclay, France

The RD13 project was originally approved in April 1991 for the study of an LHC oriented data taking system and is now nearing the completion of its third phase of activities. The guiding objectives of the project include the construction of a scalable DAQ framework capable of evolving with detector requirements, the use of a Real-Time UNIX operating system with multiprocessor support, the evaluation and use of commercial products for all aspects of software production and a software engineering approach to software design and implementation. Special attention is also given to the modelling and simulation of full architecture solutions and their demonstration via small scale setups based on different technologies. To assess the viability of RD13 a fully operational data acquisition system fulfilling the initial requirements and based on the selected components has been successfully used as a data taking system of LHC detector prototypes that present interesting and challenging readout problems.

1. Introduction

Approved in April 1991 for the study of an LHC oriented data taking system, the RD13 project is now nearing completion of the third year of its development according to the plan proposed and approved in June 1993 [1]. Two versions of the RD13 data acquisition framework have been successfully completed for use in controlled experimental environments. The main objective of the project is the study of solutions to the data taking challenges of LHC-like experimentation: the construction of a data acquisition framework capable of evolving with detector requirements where modern hardware and software components can be exploited and innovative technological solutions to the problem of software development can be applied.

The time scale for LHC experimentation and the inadequacy of the existing readout and High Energy Physics methods for online software developments make a 'top down' design premature. A more appropriate preparation for LHC is then to spend some time and resources in investigating system components and system integration aspects.

Emphasis is given to data acquisition architecture studies by means of both system modelling and small scale prototypes based on different technologies.

Improvements to the system are under way and plans for its use in more complex readout situations involving multiple subdetectors have been defined.

2. Activities and Results

2.1. The RD13 Data Acquisition System

Before describing the RD13 activities and outcomes we recall the definition of one of the structuring concepts of the DAQ design: the DAQ-Unit is a set of interconnected DAQ-Modules, using shared memory to move event data between them. Two main functional components have been defined in the DAQ: the back-end and the frontend. The former fulfills the role of the development environment, control management, high-level monitoring and user interface. The latter constitute the DAQ environment proper, whose main functions

are data readout and distribution, low-level monitoring and data recording.
The system is currently based on VME-bus, using the VICbus (Vertical Inter-Crate) to link VME crates and to integrate back-end workstations with the frontend via the SVIC (Sbus to VIC interface) and VIC interfaces. The processing elements are the MIPS 3000 RAID 8235 which responds to our intention to evaluate a RISC architecture in the frontend and the Motorola 68040 VME card CES FIC8234 running a recently ported version of LynxOS operating system. The hardware layout reflects one of the aspects of software development to which RD13 has given great importance: the platform independence of the DAQ software, i.e. the possibility to port the RD13 DAQ system to different environments.

A first version of the RD13 has been tailored to the requirements of the readout setup of three 64-channels silicon detectors (RD2-SiT) [2] that included a 67MHz analog pipeline and has been successfully run as the data taking system during November 1992 test-beam period. VME based and RISC technology (RAID8235) was used as the main acquisition processor running a real-time UNIX operating system (EP/LX).
A total of more than 5 million RD2 events were written to tape directly from the frontend processor, while monitoring was performed on a back-end SUN workstation connected to the VME system via the VIC interconnect. The acquisition rate was dominated by the data transfer over a long VSBbus, nevertheless we could measure DAQ rates of more than 1800 Hz for 750-byte events.
An upgraded version of the RD13 DAQ with enhancements to support multiprocessor environment (i.e. the redistribution of a DAQ-Unit from one to more than one processor) and event building technology has been successfully used for data taking of the RD6 detector prototype (technical run of November 1993): an LHC-oriented Transition Radiation Detector prototype [3]. RD6 needs a DAQ system adequate for the demanding readout conditions of their prototype that includes a 3000 straw detector equipped with 6 high bandwidth readout channels and a router based level-2 like trigger system (RD11). The DAQ prototype implementation has been based on the current multiprocessor features of EP/LX [4] and on a HiPPI Protocol based readout DAQ-Module. The multiprocessor features of EP/LX have proven to be globally reliable and robust.

A series of test-beam runs will take place in 1994 with detector readout setups including more than one detector prototype.

2.2. Modelling: a step towards LHC-like architectures

The hierarchical structure, high data rates, data collection parallelism and stringent requirements on the technologies impose a detailed study and evaluation of architectures and components integration.
The RD13 project has developed a modelling framework [5] for this purpose based on MODSIM II, an object-oriented discrete-event simulation language.
A library of DAQ components allow to describe a variety of DAQ architectures and different algorithms and hardware options in a modular and scalable way and used to start functional modelling of a LHC-like trigger.
A graphical user interface (GUI) is used for configuration, initialization and online monitoring of the simulation program. The modeling framework has been successfully used for studies of the event building, the event distribution and the interfaces to the upstream and downstream parts of the DAQ system.
We are planning to make further use in simulations of functional models of LHC experiments including readout, triggering

and event building.

2.3. *Event Building studies*

It is widely acknowledge that alternative solutions to today's bus-based data mergers have to be envisaged, providing a high level of parallelism. We have chosen to investigate the feasibility of parallel switches based on two different industrial standards: the High Performance Parallel Interface (HiPPI) and the Asynchronous Transfer Mode (ATM). The availability of commercial HiPPI switches and the recent development of a HiPPI/VME interface, compatible with the RD13 DAQ environment, made a HiPPI based setup a natural candidate for realising a first event builder prototype. Based on a hardware setup installed in the RD13 laboratory, a readout software event builder has been implemented using a previous version of the HiPPI Destination Firmware [6] and an RD13 library for the structuring of event data [7]. Suitable algorithms are being studied using simulations based on models of LHC detector data distributions.

2.4. *Software Engineering*

The evaluation of software engineering techniques has been a focus of activity throughout the three years period of the project. Various main lines have been tackled.

2.4.1. *Artifex methodology and CASE tool*

The application initially selected for a pilot activity with software engineering technology has been the Data Flow Protocol: the most demanding real-time component of the RD13 DAQ. A market survey and some initial experience with CASE tools lead the collaboration to select the Artifex [8] CASE tool and its associated methodology, which consists of a number of integrated CASE tools supporting an operational-like software development lifecycle. The protocol was successfully developed and a complete Artifex based data acquisition system has been running during a test-beam period jointly done by RD13 and the RD2 LHC detector prototype [2]. The final product ran very reliably although, as was expected, performance-wise it did not compare favorably with the hand-coded RD13 data acquisition system.
The second application was the design, simulation and implementation of a tracking algorithm, for the ATLAS SIT detector [8], targeted to run in the level-2 trigger embedded processor. The final product has been run on Monte Carlo data and compared to the original Fortran program: the resulting algorithm result while matching the functionality of its Fortran counterpart, performance-wise, it was some 6 times slower.
The last exercise was the re-engineering of the run control system, originally design in terms of finite state machines and developed with other commercial tools [10]. Petri- nets and the distributed facilities of Artifex coupled with the tolerant performance requirements appeared well suited to the task. The original state machines have been translated into an Artifex model, simulated and then ported to the target RD13 environment consisting of back-end and frontend processors. Although Artifex has demonstrated to cover the full software lifecycle in a seamless continuous way and the code generated to be reliable, the drawbacks lie mainly in the 'static' nature of the modelling where instead a certain degree of dynamism is necessary in the final application.
Details of this work are available through the World Wide Web at "http://rd13doc.-cern.ch/welcome.html" [11].

2.4.2. *Software metrics*

The measurement and interpretation of suitably defined software metrics are key issues towards better quality (reliability, maintainability, etc.) software. Activities in this area have therefore been undertaken with the

aim of assessing the validity of these concepts, the capabilities of available tools and monitoring the quality of the RD13 software. The logiscope tool (sold by Verilog, France) provides facilities for examining the structure of source code, testing it and performing quality assessment using the most widely accepted software metrics.

2.4.3. OODBMS

The data acquisition of a HEP experiment needs a large number of parameters to describe its hardware and software components. These parameters have to be accessed by different parts of the DAQ system. Object-Oriented DataBase Management Systems (OODBMS) seem to fit the requirements of such applications quite successfully. We have made an investigation [12] of two commercial OODBMS, Gemstone [13] and ITASCA [14] with the aim of comparing the two systems and study the applicability of the Object-Oriented approach with respect to the relational system already in use. The results of this work are reported elsewhere in these proceedings [15].

2.4.4. Code Management

The amount of programming effort, number of people involved and code produced in RD13 is such that a generalized scheme for source code management has become necessary. We have adopted the Concurrent Version System, CVS [16], and defined a scheme for the organization of the source code and the maintenance of the releases of the RD13 data acquisition system [17].

We are considering the evaluation of an object oriented methodology and CASE tool such as the Object Management Workbench (OMW) system [18]. OMW provides a set of tools to integrate the analysis, design, implementation and execution stages of the software development lifecycle. It is an X Windows-based application which runs in a UNIX environment and interfaces to an application development environment called the Kappa Development system which allows you to build object oriented applications interactively [19].

3. Conclusions and Future Plans

Encouraged by the positive and promising results obtained so far, the RD13 Collaboration has proposed to continue the project, with activities in the three main areas already defined in the proposal of June 1993:

1) DAQ system upgrade to a multi-DAQ version, essential when data distribution (event building, data switching) stages are present in the system to be used in forthcoming test-beam sessions.

2) Laboratory developments, including

a) Modelling of full parallel LHC-like readout layouts, as a further step towards LHC-oriented architectures and further testing of the present RD13 modelling framework;

b) Event Building studies with architectures based on HiPPI and ATM technologies.

3) Continue emphasis on *software engineering* by extending the use of CASE tools, addressing software quality issues and use of the ITASCA OODBMS.

4. References

[1] L.Mapelli et al., A scalable Data Taking System at a Test Beam for LHC, CERN/DRDC 90-64, CERN/DRDC 91-23, CERN/DRDC 92-1, CERN/DRDC 93-25.

[2] S.Buono et al., A hardware setup for the SiTP test readout, RD13 Technical Note 41.

[3] B.Dopgoshein et al., RD6 Status Report, CERN/DRDC/91-47, October 1991.

[4] G.Ambrosini et al., Real-Time UNIX in HEP Data Acquisition, Proceedings of ICALEPCS'93, October 1993.

[5] G. Ambrosini et al., Modelling of Data Acquisition Systems. These Proceedings.
[6] S.Buono et al., User's libraries for the RIO/HiPPI 8262/D Module, RD13 Technical note 80.
[7] G. Ambrosini et al., Event Format in the Read-Out Module, RD13 Technical Note 109.
Available through WWW "http://rd13doc.cern.ch/welcome.html".
[8] ARTIFEX, ARTIS1991, Artifex Environment User Guide.
[9] ATLAS Coll., Letter of Intent, CERN/LHCC/92-4, October 1992.
[10] R. Jones et al., Building Distributed Run-control in UNIX, Proceedings of CHEP'92,1992, CERN 92-07.
[11] G.Ambrosini et al., Software Engineering Techniques and CASE tools in RD13, Proceedings of ICALEPCS'93, October 1993.
[12] M. Skiadelli, Object Oriented database system evaluation for the DAQ system. RD13 Technical Note 108. Available through WWW at "http://rd13doc.cern.ch/welcome.html".
[13] An Introduction to GemStone V3.0. Servio Corporation 1992.
[14] ITASCA Distributed Object Database Management System. Technical Summary for Release 2.1. Itasca Systems, Inc. 1992.
[15] G. Ambrosini et al., OODBMS for a DAQ system. These Proceedings.
[16] B. Belliner, CVS II: II:parallelizing software development, Proceedings of the Winter 1990 USENIX Conference, 1990.
[17] G. Fumagalli et al., RD13 Source Code Management, RD13 Technical Note 58.
[18] OMW User's Guide, Version 1.0, IntelliCorp, Inc., OMW1.0-UG-2, 1994.
[19] Kappa User's Guide, Version 3.0, IntelliCorp, Inc., K3.0-UG-2, 1993.

Modelling of Data Acquisition Systems

S. Buono, I. Gaponenko[1], R. Jones, L. Mapelli, G. Mornacchi, D. Prigent,
E. Sanchez-Corral, R. Spiwoks[2], M. Skiadelli
CERN, Geneva, Switzerland

G. Ambrosini, G. Fumagalli, G. Polesello
Dipartimento di Fisica dell'Universita e Sezione INFN di Pavia, Italy

P.Y. Duval, A. Le Van Suu
Centre de Physique des Particules de Marseille, IN2P3, France

K. Djidi, M. Huet
Departement de Physique Nucleaire - STEN, C.E. Saclay, France

The RD13 project was approved in April 1991 for the development of a scalable data taking system suitable to host various LHC studies [1]. One of its goals is to use simulations as a tool for understanding, evaluating, and constructing different configurations of such data acquisition (DAQ) systems. The RD13 project has developed a modelling framework for this purpose. It is based on MODSIM II [2], an object-oriented discrete-event simulation language. A library of DAQ components allows to describe a variety of DAQ architectures and different hardware options in a modular and scalable way. A graphical user interface (GUI) is used to do easy configuration, initialization and on-line monitoring of the simulation program. A tracing facility is used to do flexible off-line analysis of a trace file written at run-time.

1. Introduction

The DAQ systems for a detector at a future collider like LHC will have to cope with unprecedented high data rates (~10 GByte/s), parallelism (100 to 1000 processors) and new technologies (e.g. SCI, ATM) [3]. Simulation of different architectures, algorithms and hardware components can be used to predict data throughput, the memory space and cpu power required and to find bottlenecks before such a system will be actually constructed. Therefore one needs a modelling framework with a high level of description and a clear mapping between the system to be built and the system to be modelled. The framework has to be modular and scalable to allow simulations of the different configurations from simple systems up to full DAQ systems for big detectors.

2. Implementation

The modelling framework presented in this paper is written in MODSIM II [2] which is an object-oriented language for discrete event simulation and has its own graphics library.

The modelling framework itself consists of a library of generic objects for the DAQ simulation (DSL, DAQ Simulation Library), a graphical user interface (GUI)

1. On leave from the Budker Institue of Nuclear Physics, Novosibirsk, Russia.

2. Also at the University of Dortmund, Germany.

and a tracing facility for off-line analysis of the simulation results.

The package has been developed in the RD13 project and is still evolving while a working version is available [4]. It has been used for small applications and is used for event building studies and is being considered for DAQ simulations by the ATLAS collaboration [5].

3. The DAQ Simulation Library

The DAQ Simulation Library consists of generic objects to describe any kind of DAQ system. The basic elements are:

- **Items** are information carrying data accumulations that are passed in a DAQ system, e.g. event data, trigger signals.
- **Processes** are the active objects in a DAQ system passing items and acting on them, e.g. read-out or recording process.
- **Resources** are the limiting factors the processes have to compete for in order to fulfill their task, e.g. cpu, buffer, transfer media.
- **Control** elements are abstract objects controlling the processes and carrying information on the data flow, e.g. timers, allocation algorithms.

The main idea of the DSL is to use the smallest indivisible (**"atomic"**) processes that can then be used to build up any DAQ system. A dozen **"atomic"** processes have been defined and make the core of the DSL.

The DSL has a generic level consisting of objects for a generic description of DAQ systems, and a user level where inheritance is used to combine the generic objects with user dependent features. Thus the DSL contains the possibility to refine the objects and to include hardware dependent features.

As an example of an application of the DSL the readout of the combined RD6/RD13 testbeam in November 1993 has been simulated [6]. This setup consisted of a single chain of data flow using a HIPPI link and had a total data rate of 1.5 MByte/s. This example was used as a proof of principle: it showed the easy mapping between reality and simulation and the consistency between the values measured and the values simulated. The simulation could then be used for changes of parameters and extensions of the setup.

4. The Graphical User Interface

A graphical user interface [7] based on the graphical objects in MODSIM II is used to easily configure the simulation model, to initialize each object and to monitor parameters on-line. The GUI has three windows:

- **the library window** displays the objects of the DSL.
- **the configuration window** is a canvas on which the configuration to be simulated is built.
- **the display window** monitors parameters while running the program.

Additional features are available

- for saving and reloading whole configurations and their initialization values.
- for grouping of objects (very useful for copying parts of the configuration).
- for organizing views in a hierarchial way (very useful for complex configurations).

While the GUI can be used to build a configuration and to debug it, there is also a fast version available which can be used to run the program without graphics, thus increasing the performance for time consuming simulations.

5. The Tracing Facility

The tracing facility is a tool that allows each single "atomic" process to

report on its activity by writing a trace record in a file. This facility can be switched on and off for each individual process. The format of the trace record can be extended by the user.

The trace file can have binary or ascii format and can be processed off-line (i.e. after running the simulation) by a tool which is implemented as a C program. This tool can:

- reproduce each individual trace record.
- produce general statistics,e.g. number of events generated, size of the events, etc.
- produce statistics on each type of trace record, e.g. event generation frequency, buffer usage over time, etc.
- can order the records on an event-by-event basis, e.g. latencies, lifetime of event, etc.

The results of the various analysis are written in ntuple format and can be visualized with the help of PAW [8].

6. Conclusions

The DSL (together with the GUI and the tracing facility) is a high-level description language for simulations of DAQ systems. It can be used for simulation of any kind of DAQ system and has the possibility to include lower level hardware simulations. The GUI allows an easy configuration, initialization and on-line monitoring of a simulation program. The tracing facility allows a highly flexible analysis of the output.

The part interfacing from the detector simulations to the DAQ simulations delivering the information on size and distribution of the data in the front-end buffers (the physics interface) is already foreseen, but not yet implemented.

The DSL has been successfully used for simple examples. In the RD13 project it is used for studies of the event building, the event distribution and the interfaces to the upstream and downstream parts of the DAQ system. It is being discussed for use in simulations of functional models of the whole ATLAS DAQ including read-out, Level-2 triggering, event building and Level-3 triggering.

A version of the software is publicly available [4] and documentation can be found on WWW [9].

7. References

[1]L. Mapelli et al., A Scalable Data Taking System at a Testbeam for LHC, CERN/DRDC 90-64, CERN/DRDC 91-23, CERN/DRDC 92-13, CERN-DRDC 93-25.

[2]CACI Products Company, MODSIM II The Language for Object-Oriented Programming, La Jolla, California, 1991.

[3]L. Mapelli, The challenge of Triggering and Data Acquisition at Supercollider Experiments, NIM A315 (1992) 460.

[4]anonymous ftp from sunsci.cern.ch, directory simulation/dsl/pro.

[5]The ATLAS Collaboration, Letter of Intent for a General Purpose pp Experiment at the LHC, CERN/LHCC 92-

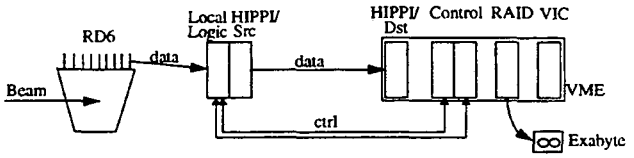
[6]R. Spiwoks, RD13 Technical Note 97, Modelling of the RD6/RD13 Testbeam Setup, February 1994.

[7]K. Djidi, RD13 Technical Note 107, A General Graphical User Interface with MODSIM II, February 1994.

[8]R. Brun et al., PAW - Physics Analysis Workstation, CERN Program Library Q121, Geneva, 1991.

[9]URL: http://rd13doc/welcome.html.

8. Figures



picture 1: The RD6 Testbeam Setup

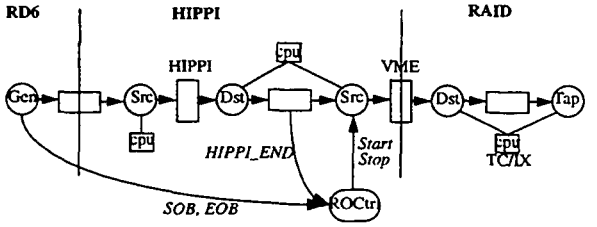


figure 2: The Simulation Model of the RD6/RD13 Testbeam Setup expressed in objects of the DSL

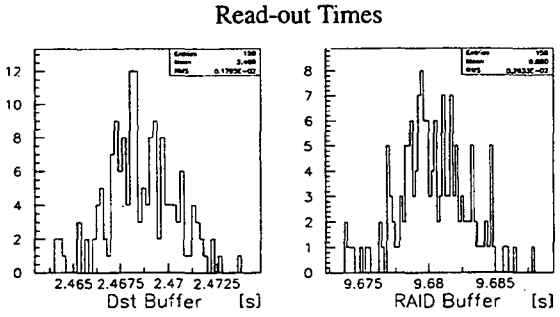


figure 3: Results from the Simulation:
The two plots are obtained using the tracing facility. They show the time to read out the Dst buffer and the time to store all data on tape.
The simulated and the measured values agree.

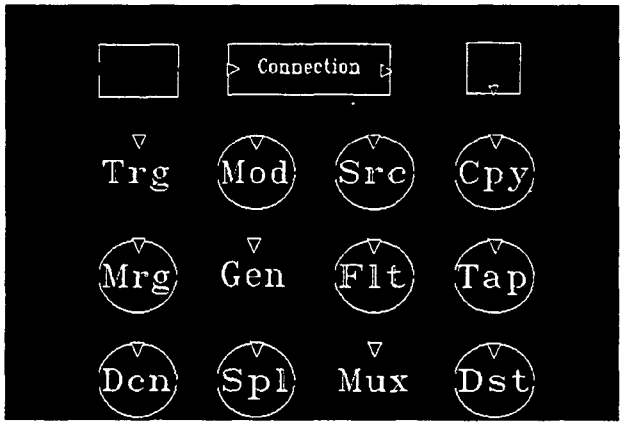


figure 4: The Library Window of the GUI (processes are drawn as circles, resources as rectangles)

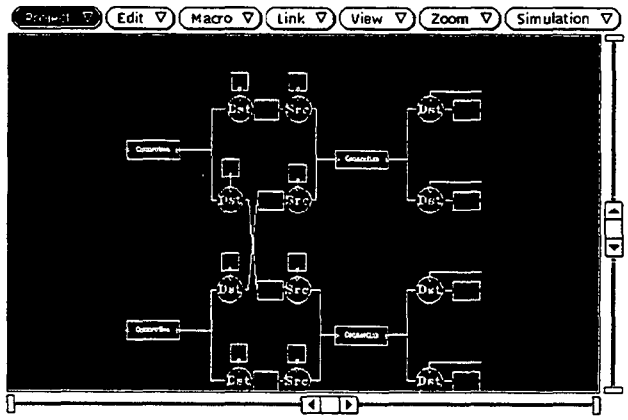


figure 5: The Configuration Window of the GUI (showing a part of a complex example)

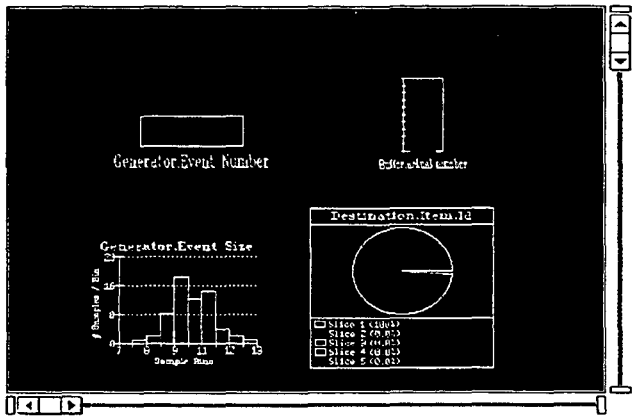


figure 6: The Display Window of the GUI

UNIDAQ

M.Nomachi, Y.Yasu
KEK, Oho 1-1, Tsukuba, Japan 305

R.Ball
U.Michigan, The Harrison M.Randall Laboratory of physics,
Ann Arbor, Michigan 48109-1120

A.Fry, C.Erbas
SSCL, 2550 Beckleymeade Avenue, Dallas, Texas 75237

Y.Takeuchi
Tokyo Institute of Technology, Meguro, Tokyo, Japan 152

C. Timmermans
148 Tate Lab. of Physics, U. Minnesota, Minneapolis, MN 55455

Abstract

UNIDAQ is a UNIX based data-acquisition system designed for the portable data-acquisition system primarily for the Solenoidal Detector Collaboration (SDC) testing and development. The UNIDAQ group, which is KEK, LBL, U.Michigan, SSCL and Tokyo Inst. of Tech., defined the UNIDAQ base-line.

UNIDAQ is highly portable data-acquisition system. It runs on SUN, DEC, SGI and HP platforms. VxWorks is also supported.

I. Introduction

UNIDAQ is a UNIX based data-acquisition system. The UNIDAQ group attempts to define the base-line for the portable data-acquisition system primarily for the Solenoidal Detector Collaboration (SDC) testing and development at various locations throughout the collaboration. The UNIDAQ group, which is KEK, LBL, U.Michigan, SSCL and Tokyo Inst. of Tech., defines the base-line for the system and shares the UNIDAQ development. [1]

II. Philosophy

The SDC collaboration requires data acquisition capabilities at many locations. The computer environment on those locations are very much different from each other. We had to define the base-line so as we can provide uniform capabilities in reduced costs. It requires scalability from small system to high performance system and portability.

UNIDAQ runs on standard UNIX. It does not require any special modification on an operating system. Therefore, UNIDAQ is highly portable data-acquisition system. It runs on SUN under SUNOS and Solaris, DEC under ultrix and Alpha OSF-1, SGI and HP platforms. VxWorks and LynxOS are also supported for the users who needs real-time capability.

UNIDAQ is developed by the collaboration. The UNIDAQ group had to define the interface among the data acquisition modules so as the collaborators could share the effort of the development. As the consequence, UNIDAQ is highly modular system. It is easy to pick-up a part of the system for one's application from UNIDAQ. It is also easy to add or replace pieces of data acquisition parts.

III. Open platform

We chose VME as a hardware platform of UNIDAQ. Because VME is an industrial standard. Many of work-stations have the interface to VME. Many of peripherals like CAMAC have the interface on VME. UNIDAQ supports CAMAC access through VME-bus. It supports Kinetics system's K2917 VME interface and K3922 CAMAC crate controller.[2,3,4] Figure 1 shows CAMAC

access layers on UNIDAQ. The VME interfaces supported by UNIDAQ are Solflower's SFVME, Performance technologies PT-SBS-915, DEC's Turbo-channel VME interface and SPARC VME boards. CES's VIC-bus and BIT3's interface is optionally supported. CAMAC library on DEC, SUN, HP and VxWorks is provided with the common user interface

<table>
<tr><td colspan="5">User program</td></tr>
<tr><td colspan="5">CAMAC Library</td></tr>
<tr><td>Ultrix driver</td><td colspan="3">SUNOS driver</td><td>VxWorks driver</td></tr>
<tr><td>Turbo Channel VME</td><td>SFVME 100</td><td>PTISB 915</td><td>SPARC 2CE</td><td>MVME 167</td></tr>
<tr><td colspan="5">VME</td></tr>
<tr><td colspan="5">K2917 VME interface</td></tr>
<tr><td colspan="5">K3922 Crate controller</td></tr>
<tr><td colspan="5">CAMAC</td></tr>
</table>

Figure 1. CAMAC access layers

IV. Real-time responses

Data acquisition system requires good interrupt response. UNIX had been classified that it has no real-time capability. However, high performance processors show good inter process communication response as shown in table 1. The other real-time capabilities for data acquisition system is also measured in the reference 5 and 6.

Table 1. IPC Response

Platform	IPC Response (μsec)
HP742rt	12
Alpha OSF1	13
DEC2000/125	67
Sparc2 SunOS	114
Sparc2 Solaris	79

UNIX has good real-time response in average value. However, there is no guaranteed response time that real-time operating system has. The measurements of the distribution of real-time response time were shown in the reference 7. Do we need guaranteed response time on data acquisition system? If the delayed response does not cause fatal problem, we don't need the guaranteed response. We need faster response in average. UNIX has no guaranteed response but faster response. It is one of the reasons we could chose UNIX as software platform.

V. Overview

The long write-up describing UNIDAQ is available by SDC note [8] or by anonymous ftp.

UNIDAQ data-acquisition processes, which are "collector", "recorder", analyzer" and so on, communicate with each other using system V message queue.

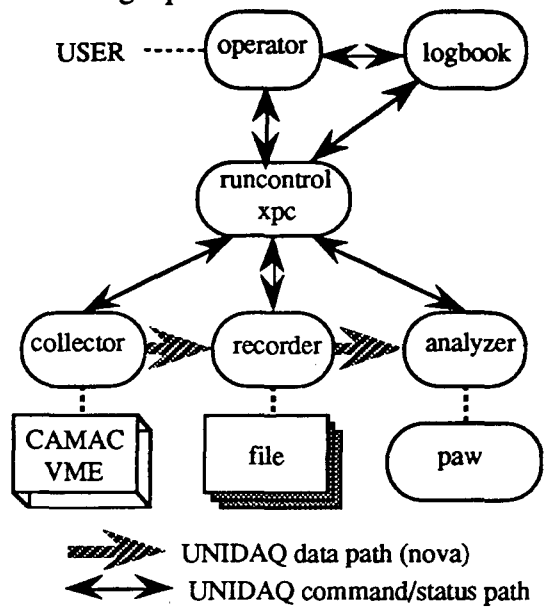


Figure 2. Typical UNIDAQ processes

Three types of communication path are provided on UNIDAQ.

1) Data path provides main data stream through the processes. NOVA buffer manager is developed at KEK. NOVA handles data pointer to the shared data memory. There is no data copies on one machine. NOVA transfers the data with TCP/IP socket over another machine.

2) The command path is used to send commands like begin and stop to processes. The commands are passed over the network if the process is on another machine. Run-control process controls the processes with the command path. One can export local variables so as the other process can access the variables. Error messages are passed along the command path. Commands SET and SHOW are provided to access the variables.

3) Status path provides the way to report statistics. The information for error recovery is also provided by status path.

The figure 2 shows typical example of the process relationship. The collector process for getting data from VME /CAMAC and put the data on NOVA buffer manager. The recorder process gets the data buffer from NOVA, and puts them onto a permanent storage. The analyzer process analyze the data. The analyzed results are displayed by HBOOK library and PAW. Templates of those processes are available on the distribution package.

Several processes, which are developed by Univ. of Michigan, make the complex commands to easy to handle. Runcontrol process execute command sequences. XPC process takes care the termination of the other processes. A simple graphical user interface is also provided. It runs just on the Xlib. Therefore, it is very portable system. Of cause, one can plug in one's own user interface. Tk/Tcl[9] is one of the optional graphical user interface on UNIDAQ.

Errors and statistics logging is provided with a simple "logbook" process. The interface to MURMUR, which is developed at Fermilab, is also provided. One can choose the logging process.

Data acquisition processes and control/logging processes may be spread across several machines over network.

VI. Present applications

UNIDAQ is installed and used for SDC beam tests and detector tests at KEK, BNL, CERN, LBL, the University of Michigan, Indiana University, the University of Toronto, the Tokyo Institute of Technology and SSCL. UNIDAQ is also used at many institutes not only for the SDC but for many fields. KEK chooses UNIDAQ as a standard data acquisition system. It is also used for detector and beam tests for g-2 experiment at BNL(AGS821).

VII. Summary and Conclusion

UNIX based data acquisition system UNIDAQ is successfully developed. UNIDAQ requires minimum setup on the hardware and the operating system. It makes UNIDAQ very portable system and easy to be used as a small data-acquisition system. Many option is also provided for the data acquisition system which require very high performance. UNIDAQ may meet very wide variety of the requirements. It keeps growing by replacing a part of the data acquisition modules to better ones.

VIII. References

[1] A.Fry and M.Nomachi, UNIDAQ A Portable Data-Acquisition System for SSC Detector R&D, Eighth conference on real-time computer applications in nuclear, particle and plasma physics, Vancouver,1993,ed. D.Axen and R.Poutissou,p326.

[2] Y.Yasu, et al., POSIX realtime extension and RISC/UNIX data acquisition, Proc. Int. Conf. Computing in High Energy Physics, Annecy, 1992, ed. C.Verkerk and W.Wojcik (CERN92-07,1992)p.627.

[3] Y.Takeuchi, T.Tanimori and Y.Yasu, Development of data acquisition system using RISC/UNIX workstation, Nucl. Instrum. & Methods A328 (1993) 526

[4] C.Erbas, M.Botlo and A.Fry, "Writing Device Drivers for the VxWorks Operating System", SSCL-N-800,Superconducting Super Collider Laboratory, 1992.

[5] Y.Yasu et al., VMEbus based computer and UNIX as infrastructure of DAQ system, CHEP94

[6] Y.Yasu and Y.Tajima, DAQ Performance of Real-time UNIX on New HP VME Board,1993,SDC-93-600,KEK Internal 93-11

[7] Y.Yasu et al., CAMAC Device Driver for DECStation with TURBOchannel-VME adapter., KEK Internal 91-7,1991, (In Japanese)

[8] UNIDAQ collaboration, UNIDAQ Document Set,SDC-93-573, Superconducting Super Collider Laboratory, 1993, or UM-HE-93-29, Univ. of Michigan, 1993.

[9] J.K Ousterhout, "An Introduction to Tk and Tcl", University of California, Berkeley, CA 94720.

Fermilab's DART DA System*

R. Pordes, J. Anderson, D. Berg, D. Black, R. Forster, J. Franzen, S. Kent,
R. Kwarciany, J. Meadows, C. Moore, G. Oleynik, V. O'Dell, D. Slimmer,
J. Streets, O. Trevizo, L. Udumula, M. Vittone, M. Votava, N. Wilcer
Online Systems Department, Fermilab
V. White, Computing Division, Fermilab
Jürgen Engelfried, E781, Physics Section, Fermilab
Taku Yamanaka, E832, Osaka University
Cedric M. Guss, E811, Cornell University
Eric Stern, John Kim, E815, Columbia University
George Zioulas, E835, University of California at Irvine
Anna Majewska, E835, Pennsylvania State University
Art Kreymer, E831, Physics Analysis Tools, Fermilab

Abstract

DART is the new data acquisition system designed and implemented for six Fermilab experiments by the Fermilab Computing Division and the experiments themselves.

The complexity of the experiments varies greatly. Their data taking throughput and event filtering requirements range from a few (2-5) to tens (80) of CAMAC, FASTBUS and home built front end crates; from a few 100 KByte/sec to 160 MByte/sec front end data collection rates; and from 0-3000 Mips of level 3 processing.

We report on the architecture and implementation of DART to this date, and the hardware and software components that are being developed and supported.

1 Introduction

DART has been established as a collaborative project between six Fermilab experiments and the Fermilab Computing Division to develop and deploy the experiments' data acquisition systems [1]. The system hardware and software architecture must be simple enough for the small experiments, yet extensible and fast enough for the large.

This paper presents a summary of the hardware and software components being supported, the current status of the project, together with details of some of the project strategies and issues.

2 DART Architecture

The DART system architecture is "parallelized" "extensible" "networked" and "distributed". In terms of hardware components, this means that sub-systems and readout are independent and in parallel, the event building architecture is modular and extensible, and

DART DA Parameters

	Small Expts	Large Expts
Trigger rate (KHz)	<.1	10-20
Event size (KByte)	1-12 (up to 200)	5-8 (up to 200)
Rate to event builder (MByte/sec)	1-3	30-160
Event building (MByte/sec)		50-160
# parallel streams	4-6	4-12
# parallel event building VME crates	1	1-4
Max. rate per stream (MByte/sec)		20-40
CPU power for event filter (Mips)	None	1000-3000
Logging (MByte/sec)	1	6-8

* This work is sponsored by DOE contract No. DE-AC02-76CH03000

Ethernet is used for control.

For the software architecture, support is given for stand-alone use of sub-systems and embedded processors for commissioning, and for integration of multiple copies of DA components as a tightly coupled system during data taking.

3 DART Hardware Summary

As far as possible all DART hardware modules are commercially available. KTeV has the most challenging data acquisition requirements; its data flow architecture is shown below [2]. DDDs consist of a triumvirate of modules to support the needed flexibility -- the DM115, DC2 and Dual-ported VSB/VME memory. The DM115 [3] provides for input from RS485 at up to 40 MByte/sec to a 4 KByte data FIFO, and for receipt of data in different VME crates based on the value of an address word in the data stream; the DC2 [4] controls data flow at up to 22 MByte/sec from the FIFO over VSB to commercial dual-ported VSB/VME memories (DPMs) and handles their memory management and flow control. The DC2's embedded 68340 processor gives it flexibility at the expense of simplicity, but this was a trade-off we accepted in order to use an already commercially available design.

The dual ported memories can be configured in size and number to meet the individual experiment's needs. Data is also delivered to the VME backplane through other supported interfaces such as the CAMAC drivers listed in the table below.

Event data is read directly by a commercial 68040 processor board (Motorola MVME167) and written to tape, or written through high speed interconnects into commercial UNIX™ workstations for physics analysis before a selected sample of events are written to tape.

4 DART Software Summary

DART software products include all the traditional components of data acquisition systems. Two of these products — the data flow manager (dfm) and distributed system bootstrap and monitoring software (dbs) — are

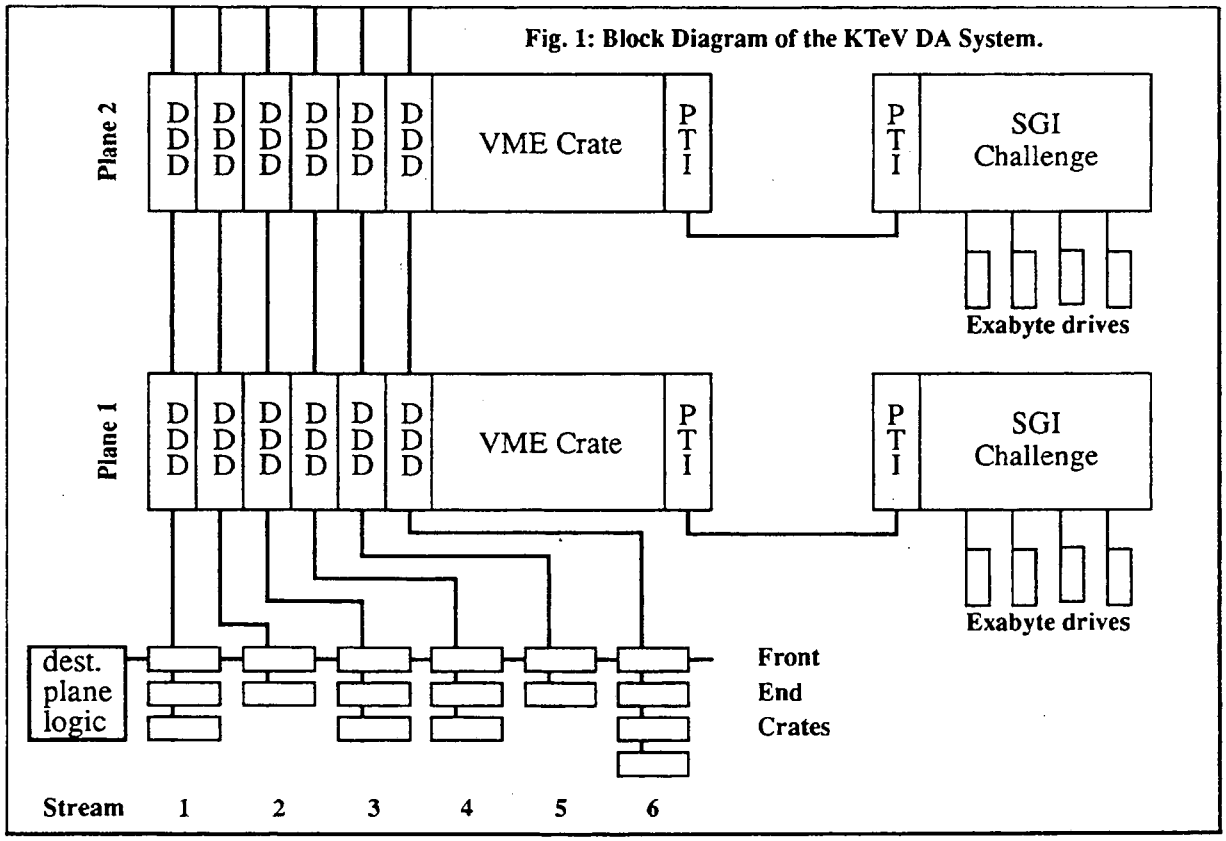


Fig. 1: Block Diagram of the KTeV DA System.

Supported Hardware Modules [12]

Description	Module	Software
68040 VME processor	MVME167	VxWorks, user libraries
FASTBUS Readout Controller	FSCC	VxWorks, FB IEEE routines
FERA Readout controller	DYC+	—
RS485/Fiber adaptor	FOXR/FOXT	—
VME/CAMAC Parallel BHD	CES 8210	VxWorks CAMAC IEEE routines
VME/CAMAC Serial BHD	HYTEC 2992	VxWorks CAMAC IEEE routines
VME/ SCSI adaptor	RIMFIRE 3513	VxWorks 8mm driver
RS485->VSB adaptor	DC2/DM115	Embedded s/w
VSB/VME memories	MMI 6390D	—
VME/VME or VME/processor bus	BIT3	IRIX device driver
VME-VME adaptor	PTI 940	IRIX/VxWorks I/O drivers
UNIX Workstation for event filtering	SGI	Drivers for VME/tape
Host/DA Monitoring Computers	IRIX, SunOS, HPUX?, AIX, VMS	I/O drivers to DA buses, VME

described in other papers at this conference [10]. Conceptually, dfm provides extensions to the operating system in areas of memory management and process queueing specifically for data acquisition needs. DART software packages are designed either as libraries to be embedded in experiment applications, or support user hooks for inclusion of experiment specific code.

DART uses tcl [5] as a common user and program interface. Run control commands are distributed as tcl command strings which are interpreted and dispatched by the receiving program. Graphical interfaces are layered on top of the tcl command line interface with Tk and wish. DART run control and configuration management software addresses DA components logically with named groups rather than physically by node address and process ID, and sends tcl command strings input from scripts or generated internally and then multicast.

Three complementary diagnostic and monitoring tools are used: a message reporter and display program [6]; trace — a continuous recording in time order of diagnostic information; and snapshot — a one time dump of information.

Some Strategies

5 VME as Data Acquisition "Hub"

VME backplanes form the data acquisition hub of the DART data flow architecture. As shown in Fig. 1, data is sent to the VME backplane from many different sources and can then be delivered to any processor connected to VME. Multiple VME planes act as parallel event builders to deliver the maximum required data throughput, 160 MByte/sec.

6 Use of C++ Wherever Suitable

The real time components of DART, and those user libraries that must be easily portable are being coded in C. However, with its advantages in strong typing, and emphasis on proper modulization and encapsulation of code, we are writing DART in C++ [7] wherever it can be justified. It remains to be seen whether this will cause problems under fire: but the experience so far indicates the code is written is more quickly and correctly, and is robust and maintainable. So far, our choices have led to no compromise in data throughput.

Interfaces to FORTRAN are included, to allow the experimenters to integrate in their physics analysis and monitoring code; the

strategy, however, is to gently lead them to C and C++ for their time critical applications.

7 Portability across Operating Systems

DART sees the first serious use of UNIX for the main DA by Fermilab Fixed Target experiments. While VxWorks™ [8] is supported on the embedded processor boards, a single flavor of UNIX is supported for the critical level-3 filter processors (currently SGI/IRIX), and several UNIX platforms and VMS for the back end monitoring and host computers. To meet the requirements of both the small and the large experiments, most applications are supported for both VxWorks and IRIX. Coding standards are defined with an eye to portability and POSIX as a reality in the not so distant future.

8 Tcl/Tk as the User Interface and Command Structure

The user interface is based on public domain command line and GUI software, tcl and Tk, with extensions made by us for our environment. The windowing environment, Tk, is layered above this and can be tailored to the experimenters' individual taste.

9 Leveraging

There is lively cross-fertilization of ideas and implementation of tool-kits across DART and the project for delivery of the Sloan Digital Sky Survey data systems [11]. The overlap in general requirements is leading to benefits in terms of new ideas and designs in both.

CDF and D0 are planning major upgrades in their DA systems over the next few years. Both experiments are actively looking at the underpinnings of DART (e.g., VxWorks and TCL). It will be an interesting challenge to see if three such massive bodies as CDF, D0 and the Fermilab Computing Division can leverage their efforts efficiently and effectively.

Some Remaining Issues

10 Level 3 Processor Integration (Data Input and Logging)

KTeV requirements dictate writing to 8mm tape at 3 MByte/sec from a UNIX workstation for data taking. Our benchmarks show that an SGI Challenge can log to 3 Exabytes in parallel on a single SCSI bus at an integrated rate of 1.2 MByte/sec only if very large tape records are written. (The CPU usage is then a few percent.) We need to support input data rates of 40 MByte/sec and logging of 3 MByte/sec simultaneous with data taking.

11 Run Control and DA Management

Fast, reliable, "push button" run start, stop and restart must be provided for systems with 4-6 UNIX stations, and 20-30 front end processors. Our experiences tell us that we must be very careful to understand the interdependencies of the components of the DA, and provide for accurate, simple reporting of where problems occur and what actions should be taken.

12 Robustness of UNIX for Real Time DA

We will be supporting UNIX workstations as an integral part of real time high throughput DA systems. We have developed significant expertise in analyzing VMS system issues, and must gain this experience with UNIX. Our programmers are still legitimately putting "a bug here, a bug there, and pretty soon you have UNIX" on their mail headers.

13 Integration, Monitoring and Analysis

Successful experiments require well integrated "on-spill" and "off-spill" data taking and monitoring systems. The delay in the Fermilab run schedule should stimulate us to expend effort towards making the overall acquisition of data as seamless and robust as possible, with a goal of reducing the time between data taking and writing of physics papers.

14 Status

DART V1.0 [13] was released in September 1993 and was integrated to allow E811 to take data this collider run.

Other experiments are using parts of the system for detector testing. DART uses ups [9] and sccs for product management, and all

software is available through the main Fermilab distribution mechanisms.

Most components of DART V2.0 are under test and an integrated system will be released early this summer. This will allow the experiments to take, log and analyze data, but lacks full multi-stream throughput, fully functional run control, level 3 filtering, configuration management, and system monitoring capabilities.

Once we have fully integrated small DART DA systems we will be addressing the robustness, and diagnostic features of the system in earnest.

References

[1] DART - Data acquisition for the next Generation Fermilab Fixed Target Experiments, G. Oleynik, J. Anderson, L. Appleton, D. Berg, D. Black, J. Engelfried, B. Forster, J. Franzen, S. Kent, R. Kwarciany, J. Meadows, C. Moore, R. Pordes, D. Slimmer, J. Streets, O. Trevizo, L. Udumula, M. Vittone, M. Votava, V. White.G. Oleynik et al., IEEE Transactions on Nuclear Science, Vol 41, No 1.

[2] KTeV Data Acquisition System, Internal memo, V. O'Dell Fermilab, T. Yamanaka, Osaka University.

[3] DART Specification Document for DM115 Receiving Board, Oscar Trevizo, Fermilab.

[4] DC2 VSB Input Controller User Manual, Access Dynamics Inc.

[5] Tcl and the Tk Toolkit, J. Ousterhout, Addison Wesley Computing Series.

[6] MURMUR - A message Generator and Reporter for UNIX, VMS and VxWorks, G. Oleynik, L. Appleton, B. Mackinnon, C. Moore, G. Sergey, L. Udumula, FERMILAB-PUB-93, Jun 1993, Submitted to IEEE Trans. Nucl. Sci.

[7] Programming in C++, Rules and Recommendations, Ellemtel Telecommunication Systems Laboratory.

[8] VxWorks is a registered trademark of Wind River Systems, Inc.

[9] ups - UNIX Product Support, M. Votava et al. Conference Proceedings, Real Time '91.

[10] Data Flow Manager for DART, D. Berg et al.; DBS, an rlogin Multiplexor and Output Logger for DA Systems, G. Oleynik et al., Fermilab, presented at Chep '94.

[11] The Distributed Development Environment for SDSS Software, E. Berman et al., presented at Chep '94.

[12] MVME167, CES, Hytec, MMI, SGI, IRIX, SunOS, HPUX, AIX, VMS, Bit3, PTI are all registered trademarks.

[13] DART documents are available at url http://fndaub.fnal.gov:8000.

The CEBAF on-line data acquisition system

Graham Heyes, William A. Watson III, Ed Jastrzembski, Jie Chen,
David Abbott, David Barker

Abstract

The CEBAF On-line Data Acquisition system, CODA, is a flexible modular data acquisition system which has been designed to provide a common platform for data acquisition for the three experimental halls at CEBAF. CODA has been in use in detector tests at CEBAF and on-line in experiments at other laboratories (LSND, CHAOS). CODA runs on a distributed network of machines running the UNIX and VxWorks operating systems. The VxWorks systems may be embedded processors in VME, CAMAC or FASTBUS. The UNIX systems currently supported are ULTRIX and HP-UX. The system is modular and scalable and is designed to easily ported to any future operating systems.

This paper describes the main features of the current release of CODA and enhancements to CODA currently under development in preparation for the commissioning of the CEBAF detectors.

I. Introduction

The Continuous Electron Beam Accelerator Facility is a nuclear physics facility in Newport News Virginia USA. The accelerator configuration is two parallel 0.4 GeV linacs with recirculation arcs allowing up to five passes through the linacs. There are three experimental halls (A, B and C) which may receive beam simultaneously at different energies. The beam is continuous wave with no spill structure.

The data acquisition group was formed to provide a common data acquisition system for the three halls. The CEBAF on-line data acquisition system CODA is the result of these efforts.

II. What is CODA?

CODA[1] is a software toolkit from which data acquisition systems with varying degrees of complexity can be built. It is based upon the UNIX and VxWorks operating systems and has been written in such a way as to be as portable as possible between different hardware architectures.

A typical small scale data acquisition system is shown in figure 1. The detector electronic is interfaced to an embedded readout controller, a single board computer running the VxWorks operating system. The readout controllers communicate with an event builder which correlates event fragments. Completed events are then passed to an analysis/filter process where events may be rejected by an reconstruction algorithms and/or histogrammed for monitoring. In such a system the CODA user provides three components, a description of how to read out the detector electronics, FORTRAN or C filter algorithms

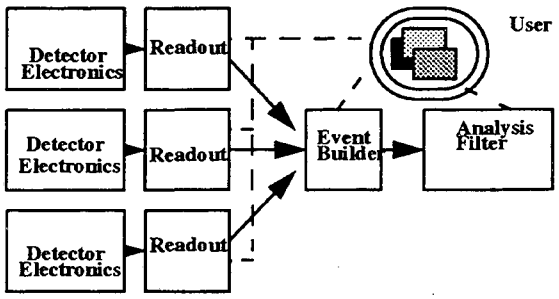


Figure 1.

and a database describing which processes will run on which physical machines, which filter algorithm to run and where to find the readout description. The current release of CODA (1.3) provides the following basic data acquisition features:

- A run control user interface to manage a network of processes.
- A readout controller shell (ROC) provides communication with run control, data formatting, buffer management. Standard interface libraries for CAMAC, VME and FASTBUS.
- A readout description language for customizing the ROC.
- Event insertion and data stream spy library.
- Software event builder to merge up to 32 data streams.
- Analysis/filter shell provides support for user written FORTRAN or C and event logging to disk file or tape.
- Error/Status message logging system.
- Event dump utility. Provides, hexadecimal or decimal dump of events CODA event format from disk or using the event spy library.

III. Run control

Run control[2] provides control of the distributed components of a CODA system. The run control process maintains a state diagram which is mirrored in all the components of the system. In the current release run control communicates with all components using RPC. In response to a user command run control calls an "action routine" in each component which causes the component to make a transition from one state to another, for example paused to active. Run control periodically checks that all processes are in the expected state. The state diagram is shown in figure 2. Run control can run a shell script before each transition. This allows for a degree of customization. For example, using Tcl/Tk the one could pop up a dialog box asking for information from the operator.

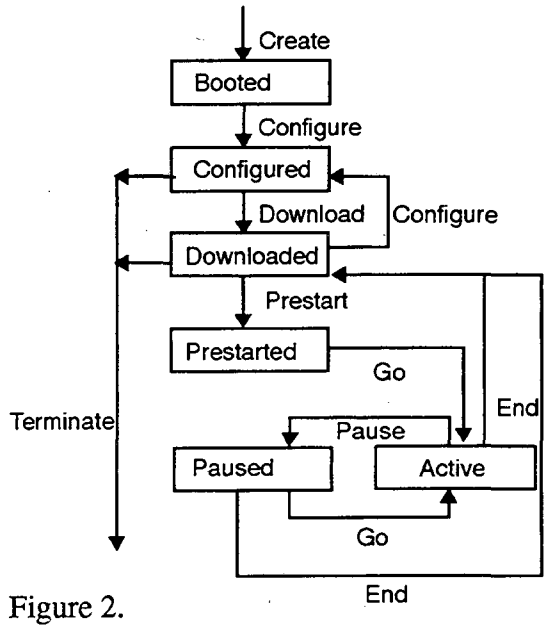


Figure 2.

Run control is configured by a flat file database. An rcNetwork file contains a description of all the components on the network, components are referred to by name and rcNetwork provides a name to physical device translation. The rcRunTypes file contains a list of names of run configurations, for example calibration, test, physics. For each name in rcRunTypes there exists a <name>.options and <name>.config file. The first file specifies special options such as the names of shell scripts to run on particular state transitions. The second file specifies which components from the group defined in rcNetwork will be active during this data taking run. This allows a subset of the hardware to be used during calibration and testing.

Run control is written in Eiffel and has a Motif interface. Push-buttons represent legal state transitions from the current state and a status display shows other useful information such as the number of events. Optionally this interface may be replaced by a graphical representation of the state diagram with push-buttons behind the state transition labels. Valid transitions are highlighted in red and the current state in green.

IV. CODA readout language (crl)

The VxWorks readout controller (ROC) is configured using a readout description written in CODA readout language, crl[3]. The crl scripts are compiled using a compiler ccrl which generates C code. The C code is compiled into an object module which may be dynamically linked to the already executing ROC. The crl compiler inserts C code to perform initialization and perform common actions such as handling trigger sources and buffer management. For advanced users a syntax exists for embedded C code.

The language is english like and is similar to scripting languages like Tcl, AppleScript and HyperTalk. For example, to read data from sub address 2 of a FASTBUS module at geographic address 12 one would write:

```
fastbus readout
...
geographic address 12
sub address 2
block read
release address
```

crl is context sensitive so that the syntax "geographic address" is not allowed if one has already declared camac readout while the syntax block read would generate the appropriate CAMAC commands. The crl language is layered on top of the "standard" subroutine libraries for CAMAC[4] and FASTBUS[5] and is to that extent hardware independent.

V. CODA 2.0

Version 2.0 of CODA has not yet been released but it's major features are already coded and functioning. The differences between CODA 1.3 and 2.0 are large and are due to the difficulties encountered in the transition from small-scale data acquisition to a large on-line system. Most of these difficulties may be traced to two causes:

- Increased Complexity
- Many Operators

In its incarnation as a detector test system a typical CODA configuration is one ROC, one event builder and one analysis program. The simplest of the three CEBAF halls, hall C, has a configuration of six ROCs, three event builders and one or more analysis programs. The need for three event builders occurs because the ROCs are in three physical locations. The hall B experiment has at least twenty readout controllers. This leads to a complex network of CODA processes rather than the simple ROC talks to event builder talks to analysis program model of coda 1.3. A consequence of the complexity issue is that during detector commissioning and testing subsets of the hardware may need to be tested at the same time. In this situation not one but several independent "experiments" are running on the same network of machines. This leads naturally to the multi-operator problem. It may be necessary so have several copies of run control managing several experiments on the same machine or to have several graphical interfaces to the same experiment.

Version 2.0 of CODA contains the following enhancements to resolve the issues discussed above:

- Replace RPC with direct TCP streams. This replaces RPC program numbers with dynamically allocated ports allowing more than one copy of a process on the same machine.
- Create an information server, CODA database manager (cdm), to give any process access to named information about any other process. This allows any process to find out which TCP port another process is listening on.
- Extend the run control database format to support more complex networks.
- Create a graphical tool for generating the new run control database format.
- Split run control into a client (GUI) and

server (manager) to allow several interfaces to same experiment.

- Blur the distinction between the ROC, event builder and analysis program software. This makes it easier to handle custom systems which do not fit the ROC, EB, ANA model.

VI. CODA 2.0 run control

In CODA 2.0 run control exists as one or more client Motif interfaces communicating with one server per experiment. The servers communicate between themselves to dynamically allocate experiment ID numbers which are passed to other CODA processes to tell them which experiment they belong to. When a user starts a run control client a motif interface appears which contains a push-button for each server which already exists on the network and a button to create a new server. Pressing one of these buttons brings up the normal run control interface connected to the appropriate server. Once created a server can run without a client. Only one client may control a particular server at any time. The clients arbitrate between themselves for mastership.

VII. CODA database manager (cdm)

The database manager cdm was designed to resolve the mapping between many distributed named processes and physical devices, port numbers and addresses. The program also acts as in information broker between CODA processes and maintains copies of any named parameters which a CODA process may want to register. A master copy of cdm must exist on any network, it maintains a list of known experiments and where to find the copy of cdm belonging to any particular experiment. When the run control client starts a run control server it also starts a local copy of cdm which registers itself with the master copy. Run control then downloads to the local cdm all the information it knows about the CODA processes it expects to see on it's network. As each CODA process is started it obtains the address of the cdm belonging to it's experiment from the master. It then loads information about itself and other CODA processes from cdm. Communication with cdm is via TCP streams.

VIII. Readout language enhancements

The crl language has been enhanced in CODA 2.0 to add syntaxes for direct manipulation of CODA event format data structures and for multiple trigger sources in the same ROC. The syntax to link a trigger to a routine is ...

link trig source <type> <id> to <rtn>
i.e.
link trig source LAM 3 to my_trigger
link trig source TIME 1 to every_sec
link trig source NET fd to net_handler

In CODA the event type is read from hardware. Depending upon the event type the user can insist that a trigger is present from one or more trigger sources before an event is processed. For Example:

event type 7 then read LAM 2
event type 7 then read LAM 5

would insist on triggers from both LAM 2 and 5. When dealing with data from multiple sources it is often useful to wrap the data blocks in a bank structure which gives more information about the contents. This can be achieved with the syntax,

wrap event with <routine>

here routine is a user written crl subroutine which generates CODA bank headers.

IX. Blurring the edges.

CODA 1.3 and it's predecessors were designed with a semi-rigid architecture in which components such as ROC, event builder and analysis program were identified. As the code has evolved the dis-

tinctions between the different classes of component have decreased. With the crl enhancements mentioned above it is possible to write an event builder using crl. The structure of an analysis program is identical with the "readout" code written in FORTRAN. CODA 2.0 contains only a generic CODA process which is configured by the user to have the desired properties but still maintains all of the supporting features of it's predecessors.

X. Conclusions

For small events (18 x 32-bit words) a 20 MHz 68020 based FSCC[6] running CODA 1.3 can achieve a throughput of over 30,000 events per second if the events are not transmitted over a network. The network protocol is the main limiting factor limiting overall performance. We have measured that 6 MIPS/(MB/S) is required to run the TCP protocol. For an FSCC this translates into a maximum data throughput over Ethernet of about 250 kB/S. We have operated the CODA 2.0 TCP stream based data transport system over FDDI and have achieved 2.8 MB/S between an MV167 VME system and a HP 9000/735 workstation. The next generation of FDDI controllers off-load much of the TCP protocol processing to RISC processors running on the FDDI controller. This should dramatically increase throughput. Higher bandwidths will be achieved either by moving to ATM of by having many FDDI links operating in parallel through a high speed switch.

CODA version 1.3 has been in general use for about one year. The CHAOS experiment at TRIUMF is using CODA 1.3 for its data acquisition. The LSND experiment at Los Alamos is using CODA 1.1. Both experiments have given favorable reports. CODA is in general use by CEBAF users for detector tests and development. Feedback from the users has driven the development of CODA 2.0.

Hall C will be the first CEBAF experiment to receive beam in June 1994. Due to our experience with using CODA 1.3 we will not release CODA 2.0 until after commissioning of hall C is complete. During this time CODA 1.3 will be used with data transport over Ethernet since beam luminosity is expected to be low. Hall C physics runs will begin during the fall and early 1995. Hall B will take beam in the fall of 1995 and expect to use R3000 based FRC FASTBUS controllers possibly with an ATM network when the hardware exists. Hall A will come on-line some time in 1995.

Considering the rapid increase in performance of commercially available hardware we believe that the approach of producing a software data acquisition system which is hardware independent is a valid one. Such an approach allows for flexible and scalable systems to be built and will ease any future hardware upgrades.

References

[1] William A. Watson III, Jie Chen, Graham Heyes, Edward Jastrzembski, David Quarrie, "CODA: A Scalable, Distributed Data Acquisition System", IEEE Trans.Nuc.Sci., Vol. 41, p61

[2] Graham Heyes, Edward Jastrzembski, David R. Quarrie, William A. Watson III, "An Object Oriented Run Control Environment for the CEBAF Data Acquisition System," *Proceedings of the TOOLS 8 Conference* (July 1992), pp. 171-183.

[3] *Graham Heyes, William A. Watson III, Edward Jastrzembski, David Quarrie,"*Real-time Front End Software For CODA", Proceedings eigth Conference on Real-time computing applications (1993).

[4] "Subrutines for CAMAC" ANSI/IEEE Std 758-1979.

[5] J. Pangburn, et. al. "FASTBUS Standard Routines Implementation for Fermilab Embedded Processor Boards," *Conference Record of the 1992 IEEE Nuclear Science Symposium*, vol. 1, pp 528-530.

[6] Gustavo Cancelo, Mark Bowden, Rick Kwarciany, John Urish, "An Intelligent Readout Controller for FASTBUS, the Fermilab FSCC," *Conference Record of the 1990 IEEE Nuclear Science Symposium*, vol. 1, pp.292-297.

An Object-Oriented Experiment Control environment for the Three-Spectrometer Setup at MAMI *

H.Kramer‡ A.Hake‡ V.Kunde‡ C.Martin‡ K.Merle§ and S.Steffens‡
‡Institut für Kernphysik
§Zentrum für Datenverarbeitung
D-55099 Mainz, Germany

INTRODUCTION

The Three-Spectrometer Setup has been built to perform electron scattering experiments at the 855 MeV electron accelerator MAMI in Mainz. Each of the three magnetic spectrometers is equipped with a detector system which consists of two pairs of vertical drift chambers, two sets of fast plastic scintillators, and a segmented Cerenkov detector. A computerized experiment control system (ECS) has been developed to operate the Three-Spectrometer Setup. It is responsible for monitoring and controlling all technical aspects of the setup, commonly referred to as slow control. In addition software is developed to monitor the data quality online. A detailed description of the former part is already given in [1].

SLOW CONTROL OVERVIEW

To cope with the local separation, and the complexity of the apparatus, and also to provide a high degree of independence to the individual sub-systems, our control system has been split into various processes running on different hosts. While MC68030 based VMEbus machines are used for direct hardware access, workstations provide the environment for all central services and user interaction. All machines operate under UNIX. The lowest software layer is comprised of objects which are directly related to I/O-hardware modules (VMEbus, RS232C, CAMAC, ...). On the same layer are objects which have knowledge of the behavior and characteristics of signal transferring devices (adc, dac, ...). The detailed structure of specific experiment devices (e.g. magnets, hv-crates) is modeled by objects on the control layer. These objects provide device specific realizations of control and monitoring algorithms. Each control object is identified by a unique name. This name is e.g. used to get configuration parameters out of a database, which is organized in such a way that a parameter entry is searched first for the specific instance of a class, and in case it is not found the search looks for an entry according to its class type.

Commands are entered on the top layer, either on command-level, or better suited for non-specialized shift crews, via an interactive X-Window program based on the OSF/Motif toolkit. The X-Window interface program is completely table driven. There is no need to recompile it if a new control object is added to the system, since the interface of each control object is stored in a uniform way in our online-database.

The software organization has been standardized up to a high degree ([1]).

*work supported by Deutsche Forschungsgemeinschaft (SFB 201)

A C++ class library, which contains all general code needed to add new control objects, is provided. Specific devices are represented by a set of C++ classes.

If two communicating control objects are not within the same process they exchange messages via the message-passing system MUPIX, which has been developed here in Mainz (for a detailed summary see [2]).

STATUS BROKER

All control objects send status information to a central status server/broker called *opcon*. Status messages are sent whenever a status parameter changes or on a regular basis depending on the object's configuration stored in the database. The status server always keeps the latest status information in its internal database. Clients may query this database. Of greater importance is the possibility that *opcon*-clients request continually status information from specific devices. Such clients are called *monitor clients*. New status messages from these devices are forwarded to the monitor clients by the status broker. Since all control objects are identified by strings, regular expressions are allowed as a selection criterion. There are three standard monitor clients in the system:

- `opterm` prints selected information on the screen.
- `opconsole` prints debug, warning and alarm messages.
- `optrans` pipes all status information in the event-builder in order to save these data together with event data on tape.

`opterm` and `opconsole` normally run on one of the screens in the control room.

ONLINE MONITORING

The quality of any scientific experiment depends on the proper functionality of the apparatus throughout the data taking period. In order to avoid loss of data it is desirable to detect problems and malfunctions as early as possible. The slow control system outlined above helps to achieve this goal, since it controls the technical parameters of the setup - a necessary but not sufficient prerequisite to perform succesful experiments. In addition it is useful to monitor the data itself online. Therefore software was added to provide shift crews with the necessary tools to gain information about the data quality.

Data flow

Subevents from the different subsystems (spectrometers) are assembled into a full event by an event-builder process, which is basically a buffer management package for event buffers located in a shared memory area. The event-builder offers filled event buffers to client programs via three channels: tape-, diagnostic, and spy channel. The event-builder software guarantees that each event is presented to the tape and diagnostic clients. Connected to the diagnostic channel is a monitoring task, which is invoked when a new run is started. The monitoring task collects the statistical information necessary for data quality control: histograms and *monitor fields*. While histograms are updated on event by event base, the contents of a monitor field is calculated in regular time intervals

as defined in the configuration database. Typical monitor fields are rates, efficiencies, multiplicities, etc, together with an error and time average. Both types of objects are related to C++ classes. Histograms and fields are stored in shared memory, so that they are accessible without interference by other applications, e.g. a histogram presenter. At initialization the monitoring task forks itself and starts an "expert"-task as a child process. Each time the set of monitor fields is updated, this "expert" performs a series of tests (threshold test, drifts, etc.). In addition the current status of all fields is written to a log file. This log file is used e.g. to produce plots of field values versus time. If a test fails, the shift crew is informed with an appropriate error message. At the end of a run all histograms are stored on disk.

Histograms

The tasks described above are layered above two basic package: a histogramming package called Tasty and a library which contains all general code necessary for the handling of the monitor fields. Both packages are written in the C++ language.

Tasty, originally started as a project to study the C++ programming language, evolved to the standard histogramming package in our online environment. It introduces histograms as a data type (class) together with a set of operations on or between them (filling, saving, integration, addition ...). Tasty histograms might "live" in shared or non-shared memory which amounts to two different classes. Both inherit interface and code from common base classes. A piece of sample code is given in figure 1.

```
// declare two histograms
HcsFastHisto a (0, 10, 100);
HcsFastHisto b (0, 10, 100);

// fill with a random variable
for (int i = 0; i < 1000; ++i) {
  a += randomGauss(5, 2.0);
  b += randomGauss(5, 2.0);
}

HcsFastHisto c = a + b; // add them.

// save it to disk.
c.write("out.dat");
```

figure 1: a C++ example using Tasty

Histograms stored in files are encoded in XDR-format.

In addition to these basic histogramming operations a variety of methods have been implemented to analyze histograms for automatic error detection, such as simple calculation of moments, channel by channel comparison with reference histograms, Kolmogorov test. This methods are to be used by the expert task mentioned above.

REFERENCES

1. H.Kramer, A.Hake, V.Kunde, C.Martin, K.Merle and S.Steffens, "The Experiment Control System of the Three-Spectrometer Setup at MAMI", New Computing Techniques in Physics Research 3, 131, World Scientific 1994
2. H.Kramer and K.Merle, "MUPIX - a Portable Message-System for Distributed Experiment Control Systems", 613, Proceedings of the CHEP'92, Annecy, France

BIG BROTHER – A FULLY AUTOMATED CONTROL SYSTEM FOR THE DELPHI EXPERIMENT

B. Franek,[3] Ph. Charpentier,[2] M. Dönszelmann,[1] C. Gaspar,[1] M. Jonker[2]
and R. Sekulin[3]
[1]CERN, ECP Division, 1211 Geneva 23, Switzerland
[2]CERN, PPE Division, 1211 Geneva 23, Switzerland
[3]Rutherford Appleton Laboratory,
Chilton, Didcot, Oxon, OX11 0QX, United Kingdom

Abstract

The integrated control system of the DELPHI experiment is described. It allows the data to be taken in an almost automatic fashion. This is achieved by computer monitoring of the states of the LEP machine, the DELPHI subdetectors (SC) and the Data Acquisition System (DAS). Depending on these states, computer initiated actions are then taken. It has been designed using the 'State Manager' concept already used for local and central controls of DAS and SC.

INTRODUCTION

DELPHI (DEtector with Lepton, Photon and Hadron Identification) [1] is one of the four experiments built for LEP (Large Electron-Positron collider) at CERN. It consists of a central cylindrical section and two end-caps. The overall length and the diameter are over 10 meters and the total weight is 2500 tons.

The electron-positron collisions take place inside the vacuum pipe in the centre of DELPHI and the products of the annihilations fly radially outwards. The products of the annihilations are "tracked" by several layers of detectors and read out via some 200,000 electronic channels. A typical event requires about 1 million bits of information.

DELPHI ONLINE SYSTEM

The Online System of the experiment is divided into four main parts. The Data Acquisition System (DAS) [2] reads event data from the 19 sub-detectors composing DELPHI and writes it onto tape. The Trigger System (TRIGGER) [3] provides the DAS system with information on whether or not to write the event to tape. The Slow-control System (SC) [4] controls and monitors slowly moving technical parameters and settings, like temperatures and high voltages. The Lep Communication System (LEP) [5] controls the flow of data between LEP machine and DELPHI detector.

SOFTWARE ORGANISATION

To provide a high degree of independence to the individual sub-detectors, the data acquisition and control system has been split into 19 autonomous partitions. The software organisation of these partitions has been standardised to a high degree. Apart from the large number of Fastbus embedded processors, the main

online computer capacity is provided by a Local Area VAX Cluster running VMS. VAX stations are used to control the individual partitions. These provide a fully independent data acquisition environment to the detector partitions when they are running in stand-alone mode, allowing the detectors to calibrate and test their equipment without interference with other partitions.

The data acquisition control has a highly decentralized organisation. To cope with the complexity of the control we developed, in collaboration with the OC group of CERN Data Handling Division, a new concept for the coding of the control logic [6]. In this concept, the components of the system are represented by objects. Objects can be abstract, representing a logical subsystem, they can be remote, representing objects in other control domains, or they can be 'elementary', in which case they represent an associated software task.

The interaction between the various objects is specified using a formal State Manager Language (SML). The logic specified in SML is translated by a special compiler to create the State Managers (SM). These run as independent processes and communicate with their environments using the Distributed Information Management System (DIM) [7] which runs on DECNET. There are 4 special SMs in which the other SMs are seen as remote objects: DAS - controlling the Data Acquisition; SC - controlling the HV tension, gas flows etc.; TRIGGER - controlling the trigger conditions and LEP - reflecting the state of the LEP machine.

Top Level Control - past

Fig.1 shows the top level control of the experiment as it existed until now. The Shift coordinator was in contact (by telephone) with the LEP operator and was monitoring various displays showing the background conditions. When the conditions for data taking were right, the Shift coordinator instructed the SC Maestro (the operator in charge of SC) to prepare the SC systems such as high voltages for data taking. When the SC were

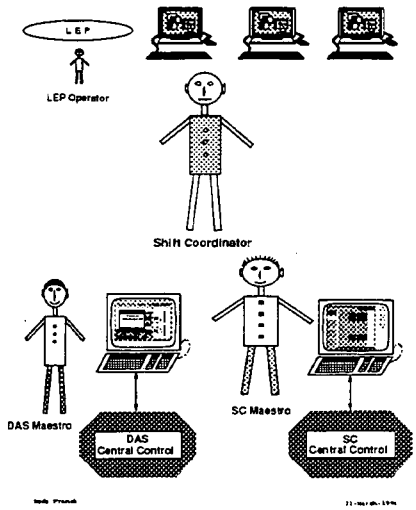


Fig. 1. Control Personnel

ready, the DAS Maestro (the operator in charge of DAS), after setting various parameters for physics conditions, started taking data. A similar procedure was followed when either the background conditions became unacceptable or when the LEP machine stopped providing beams. Fig.2 is an expansion of Fig.1 for the case of DAS. In all the figures the octagons represent SMs. The little bubbles inside represent various SM objects. The DAS Maestro controls DAS from a workstation and communicates with the DAS Central control SM which in turn coordinates the Local SMs for the individual partitions. The Local SMs send commands to the associated elementary processes which then

control their hardware subsystems. The SC has a similar structure.

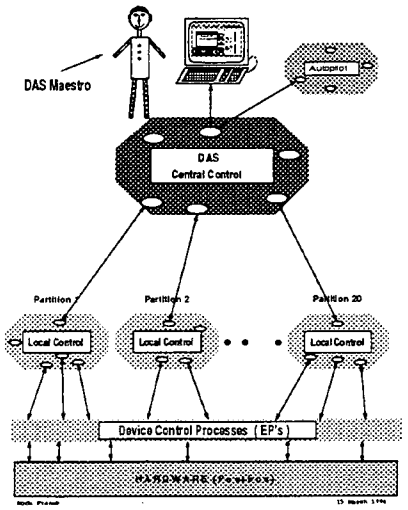


Fig. 2. DAS Control

Top Level Control - present

The Top Level control as described above suffers from the human element. It can for example happen that the Shift coordinator misses the start of acceptable conditions for the data taking, resulting in a loss of data. Or sometimes incorrect parameters may be set for data taking. To minimise such mistakes, we have developed the top SM called Big Brother (BB) which emulates the control described above. It is an example of the suitability of SML to model an existing situation in terms of objects and actions (see Fig.3). In this solution, BB replaces the Shift coordinator. The DAS and SC maestri are replaced by abstract SM objects to which BB sends commands. These objects then send the necessary commands to the DAS and SC Central controls. These commands are exactly the same as in the previous scenario. In fact the systems controled by BB hardly required any modifications. There is an additional 'elementary process' monitoring the status of the LEP machine and also the background conditions. This process is represented again by an object in the BB process. BB also monitors all its sub-systems during the data taking and in case of problems it issues audio warnings to the maestri who are there now only in an advisory role.

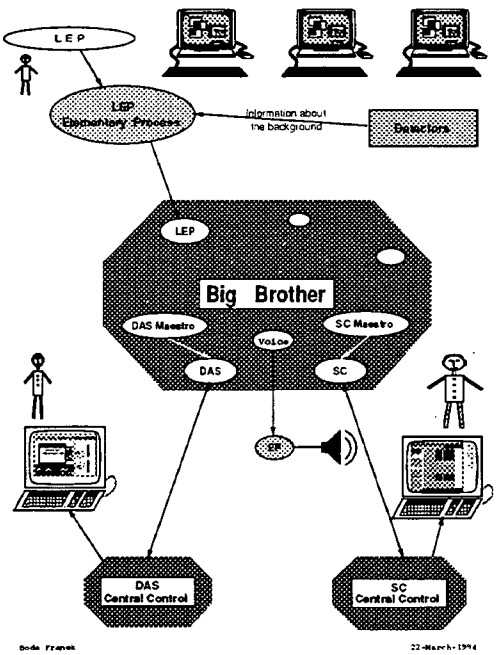


Fig. 3. Big Brother Solution

CONCLUSIONS

The first version of the DELPHI experiment control, based on the State Manager concept, was put into operation in spring 1990. Its inherent modularity made it possible to introduce new operation procedures in stages. Big Brother, the latest development, is an example of this modularity. It links DAS, SC and LEP domains into a coherent system, thus providing a fully automated control system for the experiment with a minimal need of an operator.

REFERENCES

1. DELPHI Collaboration, Aarnio, P. et al.,"The DELPHI Detector at LEP" in **Nuclear Instruments and Methods in Physics Research A303** (1991) pp.233–276

2. Adye, T. et al., "Architecture and Performance of the DELPHI Data Acquisition and Control System", in **Proceedings of the International Conference on Computing in High Energy Physics '91** (Tsukuba, Japan, March 1991)

3. Fuster, J.A. et al., "Architecture and Performance of the DELPHI Trigger system", in **Proceedings of the IEEE 1992 Nuclear Science Symposium** (Orlando, Florida, October 1992)

4. Adye, T. et al., "The Slow Controls of the DELPHI Experiment at LEP", in **Proceedings of the International Conference on Computing in High Energy Physics '92** (Annecy, France, September 1992)

5. Dönszelmann, M. and Gaspar, "The DELPHI Distributed Information System for exchanging LEP Machine related information", in **Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems** (Berlin, October 10-22, 1993) to be published in **Nuclear Instruments and Methods in Physics Research**

6. Barlow, J. et al., "Run Control in MODEL: The State Manager", in **IEEE trans. nucl. sci. 36** (1989) pp.1549–1553

7. Gaspar, C. and Dönszelmann, M. "A distributed Information Managment System for the DELPHI Experiment at CERN" in **Proceedings of the IEEE Eight Conference REAL TIME '93 on Computer Applications in Nuclear, Particle and Plasma Physics** (Vancouver, June 8-11 1993)

Diagnosing the ZEUS Experiment with the ZEX Expert System

Ulf Behrens, Mariusz Flasiński, Lars Hagge, Kars Ohrenberg
Deutsches Elektronen-Synchrotron (DESY)
22603 Hamburg, Germany

Abstract

ZEX is an expert system being developed to support operation of the ZEUS experiment. Based on the experience gained with a prototype version of the expert system, it was decided to implement the final system using a commercial expert system shell. Task of the expert system is to monitor all fields of information available online to achieve the highest efficiency of data taking by minimizing dead time of the detector and ensuring high quality of the data written to tape.

INTRODUCTION

The ZEUS detector is controlled and read out by a highly parallel distributed online system. The components of the system have been constructed independently at many different places and expert knowledge is spread over many human experts at several institutes. Because of the complexity of the system, it was decided to construct an expert system which allows to increase both efficiency and reliability of experiment operation while reducing the required expertness of the human operator. The goal is to combine the knowledge of all the human experts and make it available to the operators on shift. The final system is aimed to detect anomalous behavior in any part of the experiment, to trace errors to their origin and to recover or help to recover the system as quick as possible.

PROTOTYPE ZEX-P

A prototype version of ZEX was implemented to study methodologies and algorithms and to estimate feasibility and maintainability of the final system [1].

ZEX-P was embedded in the online DAQ system of the ZEUS experiment and diagnosing was based on monitoring information collected by the Eventbuilder [2]. The Eventbuilder is a central part of the DAQ system which has connections to nearly all components of the DAQ system. Its powerful monitoring capabilities enabled ZEX-P to explain many of the rate, bandwidth or synchronization problems, or component subsystem failures and blockages of the main data stream.

ZEX-P was based on pattern recognition methods, and all analysis processes were implemented as finite state machines [3]. ZEX-P was written in C, using Motif for graphical user interfaces.

For training and testing ZEX-P, ex-

tensive logfiles of monitoring information from the first run period of the experiment were available. Thus ZEX-P could be implemented and tuned in quasi-online mode by replaying these logfiles.

ZEX-P Experience

ZEX-P was operating reliably during the 1993 data taking period. The real-time requirements were met, but because of the limited scope of the system ZEX-P was not a real breakthrough in supporting the online operator.

The implementation method was chosen in order to test the capabilities of finite state machines in real-time expert systems. They turned out to be reliable and fast, but as the addition of knowledge usually results in a multiplication of system states, they are difficult to expand and can be modified by trained experts only.

The knowledge about a high energy physics detector and its behavior is permanently growing and changing. This requires to give highest priority to maintainability of the knowledge base: the knowledge has to be coded in an easily readable and expandable way. Unfortunately, finite state machine systems do not fulfill this requirement. Therefore, it was decided to switch to a rule based system for further development of ZEX.

RTWORKS

There is a long list of tools [G2, NexpertObject, OPS5, CLIPS, ART-IM, ...] which allow for rule based knowledge storage, but the boundary conditions in the ZEUS environment reduced the number of alternatives. Finally, RTworks [4] was seen as a tool which fulfills our requirements to a high degree:

- real-time performance: monitoring a complex real-time system requires to be able to detect correlations of symptoms in time. This is only possible with fast quasi real-time reaction of the tool.
- powerful reasoning mechanisms: forward and backward chaining, firing rules according to a context, a priority or a test interval, a library for reasoning over time is mandatory.
- a distributed, modular and flexible architecture: if the expert system has to be combined with an existing distributed system, the expert system has to be fit to the system to be surveyed and not the other way round. This means that the tool has to be able to reflect the distributed nature of the system and that existing interfaces and preprocessing methods have to be easily linkable to the tool.
- user-friendliness: the tool has to be easy to use without having to learn any specific language.
- powerful interfacing and visualization mechanism to allow the developer to concentrate on knowledge acquisition and implementation. An integrated tool is preferred which offers an easy-to-use high level graphical interface.
- maintainability: the knowledge has to be stored in a way which is readable and understandable by untrained human experts. Because of the large number of human experts

in our environment and the not yet final system, this is the only possible way of keeping the knowledge up-to-date without a large number of expert system experts.

RTworks has proven within a test installation that it fulfills these requirements.

ZEX

Fig. 1 shows the layout of the ZEUS expert system which is being set up for the data taking period 1994.

ZEX is split up into four sub-expert-systems covering the different parts of the experiment: Data Acquisition, Run Control, Slow Control and Data Quality Monitoring. Each of the sub-expert-systems has an individual knowledge base and can be operated independently. The top level part of ZEX will be driven by the output of the sub-expert-systems. It will combine the knowledge about the subsystems to an overall diagnosis and control of the system. The design of the internal architecture is based on the Blackboard approach ([5] and references therein).

The Slow Control expert system was the first sub-expert-system in commission. More then 1300 rules are needed to cover the knowledge about the basic hardware of the experiment. The expert system informs the operator about the overall status, it provides an explanation of the most severe problems and their impact, and proposes actions for the operator to take. For a detailed diagnosis, the operator is guided through hierachically layered, point-and-click colour graphics views.

Before the frame of the top level expert system will be set up, the main reasoning of the data acquisition expert system and major parts of the data quality expert system will be available.

CONCLUSION

The prototype of the ZEUS expert system, ZEX-P, has shown that the applied paradigm (formal grammars and automata) can be used successfully, but the lack of maintainability of the knowledge base has proven to be a major disadvantage in our environment. Therefore it was decided to implement the final system as a rule based system and to use the commercial expert system tool RTworks.

In a test installation, most of the functionality of ZEX-P has been reimplemented. It was obvious that the use of a commercial tool drastically reduces development efforts by supporting knowledge acquisition and maintenance and providing powerful interface environments. On the other hand, the flexible architecture of RTworks allows to reuse parts of the prototype.

As the first part of ZEX, the slow control sub-expert-system of ZEX was put into operation in May 1994.

REFERENCES

1. Ulf Behrens, Mariusz Flasiński, Lars Hagge, and Kars Ohrenberg. ZEX – An Expert System for ZEUS. In *Proc. Eighth Conference on Real-Time Applications in Nuclear, Particle and Plasma Physics*, 1993.
2. Ulf Behrens, Lars Hagge, and Wolfgang O. Vogel. The Eventbuilder of the ZEUS Experiment. *Nucl. Instr. & Meth. A 332 (1993) 253-262*, 1993.
3. Ulf Behrens, Mariusz Flasiński, and Lars Hagge. ZEXP - The ZEUS Expert

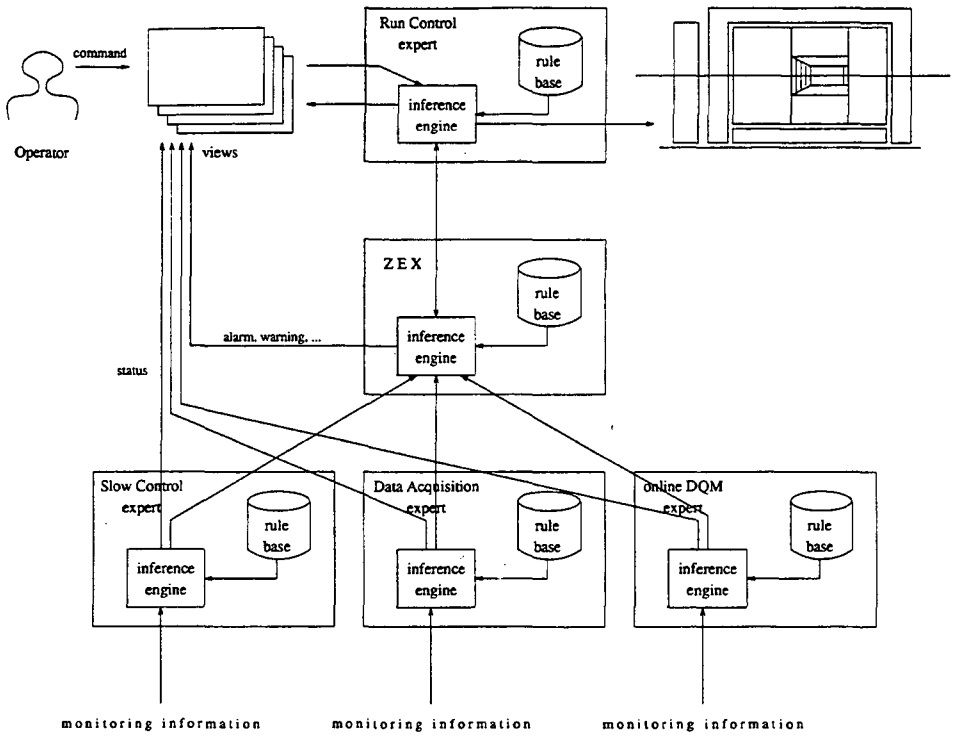


Figure 1. Layout of ZEX. Sub-expert-systems with individual knowledge bases are surveying encapsulated parts of the experiment. Their output drive the central inference engine which is responsible for the overall diagnosis and control of the experiment.

System. DESY–Report 92-141, DESY, Notkestr. 85, D-2000 Hamburg 52, 1992.

4. Talarian Corporation, Mountain View, USA. *RTworks V2.1 Users's Manual*, December 1992.

5. Mariusz Flasiński. Further Development of the ZEUS Expert System: Computer Science Foundations of Design. DESY–Report 94-048, DESY, Notkestr. 85, D-2000 Hamburg 52, 1994.

SCI Data Acquisition Systems: Doing more with less

A. Bogaerts, R. Keyser, G. Mugnai, H. Müller, P. Werner
CERN, 1211 Geneva 23 Switzerland
B. Wu, B. Skaali
Physics Department, University of Oslo, POB 1048, Blindern, 0316 Oslo, Norway
J. Ferrer-Prieto, IFIC University of Valencia, Avda Dr. Moliner 50, 46100 Burjasot, Spain
RD24 Collaboration, ECP Division, CERN

Abstract

The RD24 Collaboration at CERN investigates the use of the Scalable Coherent Interface (SCI) for DAQ Systems at the LHC. We present the status of the hardware and software, simulations, switches and their use for event building and a pilot project for early SCI applications.

INTRODUCTION

The SCI (IEEE Std.1596-1992) is a new technology that "simulates a bus but uses point-to-point links to achieve higher speed". It may be considered as a bus or a network. Like buses, it provides a shared memory environment with no software involvement for data access. It offers the connectivity of a network with reliable data transmission over optical fibres.

HEP Data Acquisition Systems exploit the ease of programming and low hardware and software latencies of buses for the readout of detector data. LANs with reliable communication protocols such as TCP/IP are used for experimental control, monitoring and, sometimes, synchronisation.

The absence of industrial standards for inter-crate connections has led the HEP community to develop their own (Fastbus, VICbus, extensions to VSB) or to resort to point-to-point links such as Transputer links, HIPPI and recently Fibre Channel and ATM. This complicates the software since communication links require drivers (and often client and server programs) for both sending and receiving of data. In addition, buffer managers are used to de-synchronize the connection between bus based memories or processors and point-to-point links.

The SCI which is a network but behaves like a bus looks therefore very promising from the software viewpoint. Its high bandwidth (the goal is 1 Gbyte/s), the intrinsic parallelism and scalability inherent to switch based networks, the cache coherency and the involvement of industry in its standardisation motivated the RD24 collaboration to investigate its use for future DAQ systems at the LHC.

STATUS OF THE HARDWARE

SCI interfaces (see Figure 1) may provide different levels of functionality. The Link Controller (LC) must be implemented in hardware. More functions require additional chips, typically FPGAs or ASICs, or may be emulated by software. RD24 uses LCs, called SCI NodeChips™, from Dolphin Interconnect Solutions (Dolphin), Oslo, Norway.

Commercial products

The first generation of LCs, implemented in GaAs (by Vitesse, Camarillo, Ca), operated at a link speed of 500 Mbytes/s with a maximum throughput per node of ~ 115 Mbytes/s [1, 2].

We are now using the second generation, in CMOS (from LSI Logic Corporation, Milpitas, Ca) with a link speed of 125 Mbytes/s. Two boards are commercially available: an SBUS to SCI Bridge, from Dolphin, and a DMA Block mover, the SCI 8224 from Creative Electronics Systems (CES), Geneva, Switzerland. The latter is a mezzanine board to the CES FIC 8234 (a MC68040 based VME board) that provides a simple connection between SCI and VME using a fast Dual Port Memory. It is based on a design for the GaAs interface from RD24 [1, 2]. VME and ATM to SCI Bridges are expected from Dolphin by May 94.

RD24 developments

We are interfacing the CMOS NodeChip to an optical fibre for serial data transmission at one Gbit/s, together with the CERN SL Controls Group who investigates SCI supported shared memory for accelerator control. We use two "Calliope" boards from Lasertron, Burlington, Ma, equipped with a pair of HP "Gigachips" and a laser and expect to demonstrate optical transmission of SCI packets by May 94.

A transparent SCI to SCI Bridge is the first step towards larger systems. A Bridge based on two CMOS NodeChips connected "back to back" using a simple on-chip routing algorithm, is expected to be ready by May 94. A more complex Bridge linking CMOS speed rings to a high speed GaAs ring acting as a data concentrator suitable for event building up to 500 Mbytes/s is also under development.

SCI (CMOS) interfaces to DSP, Turbochannel and PCI are being designed by RD24 collaborators from different institutes (University of Valencia, INFN of Rome, RAL and University of Manchester) for a pilot implementation of a Second Level Trigger System for the ATLAS experiment.

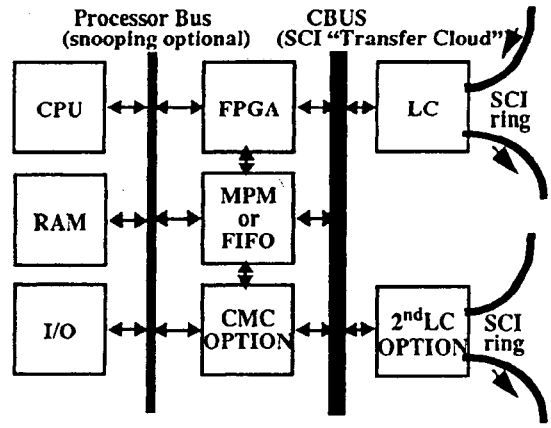


Figure 1: A variety of SCI interfaces are constructed from a chipset with at least one LC. A second LC adds a Bridge function, a CMC Cache Coherency.

FROM PACKETS TO COHERENCY

The NodeChip supports a proprietary 64 bits wide bus-like interface (CBUS) to the application logic. A typical SCI packet containing a 16 byte header (address and control information) and followed by 16 or 64 bytes of data corresponds to 4 or 10 CBUS cycles.

The simplest interfaces provide only *Packet Mode*, using software to build "CBUS packets" in an intermediate fifo or Dual Port Memory.

The SBUS and VME to SCI Bridges use FPGAs to convert CBUS packets into SBUS or VME cycles (see Figure 1). The CES FIC 8224 does this conversion on the fly. To implement DMA transfers, blocks of data of arbitrary length and alignment must be broken up into SCI packets of a suitable size. This, together with two address translation tables (for incoming and outgoing data) provides *Data Mode* allowing memory to memory copying of blocks of data over SCI links.

In *Transparent Mode* SCI behaves like a memory mapped bus eliminating all software intervention for data transfers. The

SCI address space is mapped in the address space of the local processor bus. Address and data of bus read or write cycles initiated by the processor are translated into SCI packets. This may be inefficient because of a mismatch between the data size of a host bus cycle (often 32 or 64 bits) and an SCI packet (minimum 16, preferably 64 bytes). Transparent access is implemented for the SCI to Turbochannel interface (INFN Rome, RAL) as well as the SCI to SBUS and VMEbus interfaces.

Efficiency can be improved by interfacing to processor caches which transfer bursts of data (cache lines, usually 32 bytes long). Caches also avoid repetitive access over SCI reducing latency and contention. Eventually, this will lead to Cache Coherent interfaces requiring an SCI Cache and Memory Controller. Such an interface becomes complex because it links two coherency domains: the "snooping" processor bus and the distributed SCI caches.

SOFTWARE

The two software paradigms, message passing (simplifies hardware) and shared memory (simplifies software), may co-exist.

Packet Mode

If used by software this should be considered as firmware. Its main use is for read-out controllers (e.g. the DSP interface). Like other point-to-point links, packet mode requires software for both sending and receiving data.

Data Mode

The SCI 8224 is a simple DMA block mover for DAQ applications in a VME environment. Conventions to map local, VMEbus and VICbus memory have been extended to SCI. The software is a trivial extension to an existing single-user DMA data transfer routine. No low level software is required to receive data.

The SBUS and VME to SCI Bridges are more versatile. Software being developed by Dolphin for both SUN-OS and Solaris 2.0 comprises:

- UNIX driver for DMA transfer to and from SCI memory, mapped into the user address space
- Read/Write message passing using a Remote IPC API Library
- TCP/IP network driver

The CERN SL Controls Group envisages to port the UNIX software to the LynxOs environment for the VMEbus to SCI Bridge.

Transparent Mode

This mode, implemented for the SBUS and VME to SCI Bridges, requires only initialisation software for the mapping of SCI memory in the user address space.

SIMULATIONS

SCILab [3] is a modelling tool with the aim of simulating accurately SCI protocols for large systems. It uses MODSIM II^{TM} (CACI Products Company, La Jolla, Ca) for the transport layer, the IEEE C-code for the cache coherency [4] and PAW for output. The "events" driving the simulation are packets rather than symbols to make the simulations fast. Early results showed scaling properties of single rings, the behaviour of Bridge based networks and a simple "1000 node" DAQ System [5]. Preliminary results for an ATLAS Second Level Trigger implementation have been obtained [6]. A new project has been launched to obtain more accurate results for the ATLAS Technical Proposal. This will provide a high-level model of a Trigger and DAQ System using parameters derived from detector and physics simulations, integrated with technology dependent simulation code written in MODSIM II.

EVENT BUILDING AND SWITCHES

TOPSCI is a EUREKA project (lead

partners Thomson-CSF, Paris, France and Dolphin) to produce a high performance SCI switch. The four ports are GaAs Link Controllers (at 1 Gbyte/s) on a Silicon substrate, see Figure 2.

RD24 participates in the design of the switch architecture. Simulations have been carried out to determine the switch parameters such as the amount of internal buffering, pipelining, bandwidth of the cross-connection and routing algorithms [7, 8]. Switch based networks for event building have been investigated [9, 10]. Simulations show that the throughput of multi-stage networks constructed from TOPSCI switches (see Figure 3) scales with the number of input and output rings [1, 10].

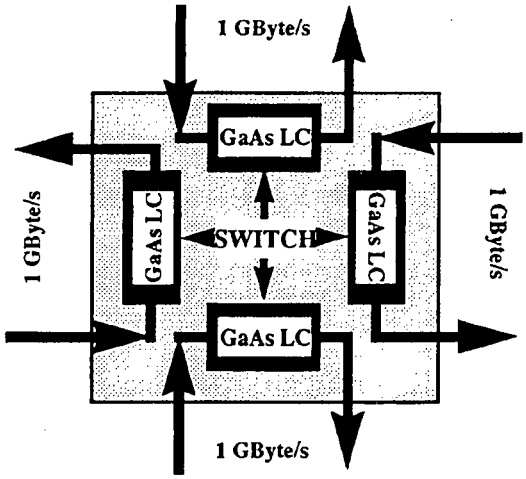


Figure 2: A "4-switch" routes traffic between 4 bi-directional SCI rings

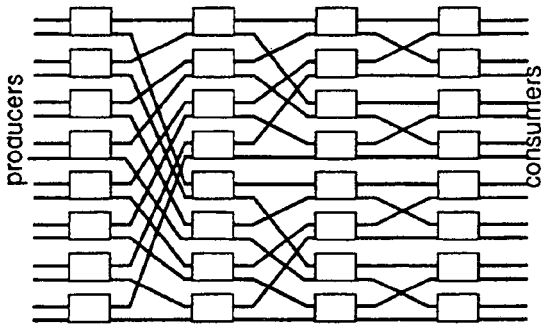


Figure 3: $16_R x 16_R$ multi-stage network constructed from 32 TOPSCI switches. Each line is a bi-directional SCI ring.

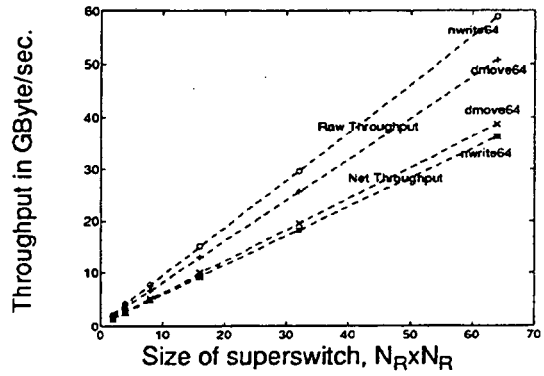


Figure 4: $N_R x N_R$ multi-stage networks switching data between N_R producer and consumer rings (1 Gbyte/s per ring) have excellent scaling properties. The number of 4-switch chips scales as $(N_R \log_2 N_R)/2$.

Figure 4 shows both the total link traffic ("Raw Throughput", this includes headers and echo packets) and the amount of useful data ("Net Throughput") traversing the switch for two different SCI transaction types (pipelined dmove64 and non-coherent nwrite64). Shown in Figure 5 is an $8_R x 8_R$ multi-stage network that was used to show the effect of imposing a burst pattern on the data, see Figure 6.

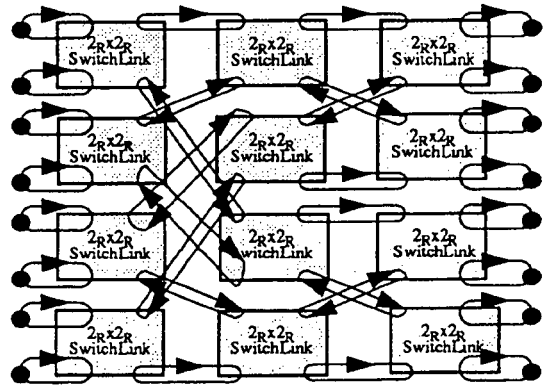


Figure 5: $8_R x 8_R$ multi-stage network constructed from 12 TOPSCI switches.

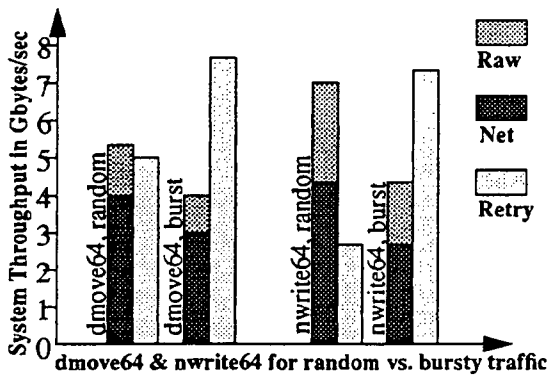


Figure 6: Comparison of switch throughput for random and bursty traffic, $8_R x 8_R$ multi-stage network.

EARLY DAQ APPLICATIONS

With the availability of commercial interfaces to VME and SBUS, DAQ applications for experiments with total data rates up to ~ 100 Mbytes/s and ~ 30 Mbytes/s per node become feasible.

Investigations to integrate SCI in the CERN CASCADE DAQ System have started, initially concentrating on connecting SUN Workstations to VME based DAQ hardware. Other possibilities are event building (up to 100 Mbytes/s) and long distance data transmission. With equivalent I/O capabilities, Work Stations are becoming a competitive alternative to VME based processors for CPU intensive tasks.

ACKNOWLEDGEMENTS

The RD24 project is funded by the DRDC at CERN, the Norwegian Research Council, the Physics Department of the University of Oslo, Creative Electronics Systems (CES) in Geneva, the DEC Joint Project at CERN, Apple Computer in Cupertino, and Thomson-CSF Semiconducteurs Spécifiques in Paris.

The ECP Division at CERN has financed the computer infrastructure for the simulations.

REFERENCES

1. RD 24 Collaboration, "RD 24 Status Report", CERN/DRDC93-20, May 1993
2. H. Müller et al., "First Experience with the Scalable Coherent Interface", RT93, Vancouver, Canada, June 1993
3. A. Bogaerts and B. Wu, "The SCILab Cook Book", internal note, CERN Geneva, Switzerland, July 1993
4. "IEEE Standard for SCI, IEEE New York, August 1993
5. A. Bogaerts, R. Divià, H. Müller, J.F. Renardy, "SCI based Data Acquisition Architectures", IEEE Trans. on Nucl. Sci. Vol. 39, No.2, April 1992
6. R.K. Böck, J. Carter, I.C. Legrand, M. Novak, "Modelling of L2 global decision structures", CERN/EAST note 93-03, April 93
7. B. Wu, A. Bogaerts, E. Kristiansen, B. Skaali, "A Study of Routing Algorithms for SCI-based Multistage Networks", technical report UiO/PHYS/94-06, University of Oslo, Norway, March 1994
8. B. Wu, A. Bogaerts, E. Kristiansen, H. Müller, E. Perea, B. Skaali, "Applications of the Scalable Coherent Interface in Multistage Networks", submitted to IEEE TENCON'94, "Frontiers of Computer Technology", Aug. 22-26, 1994, Singapore
9. B. Wu, A. Bogaerts, R. Divià, E. Kristiansen, H. Müller, B. Skaali, "Constructing Large Scale SCI-based Processing Systems by Switch Elements", Report UIO/PHYS/93-12, University of Oslo, Norway, May 1993
10. B. Wu, A. Bogaerts, R. Divia, E. Kristiansen, H. Müller, E. Perea, B. Skaali, "Distributed SCI-based Data Acquisition Systems constructed from SCI bridges and SCI switches", 10th Int. Symp. on Problems of Modular Information Systems and Networks, St. Petersburg, Russia, Sept. 13-18, 1993, also as Report UIO/PHYS/94-02, University of Oslo, Norway, Jan. 1994

OODBMS for a DAQ system

Ambrosini G[1], Buono S[2], Duval P.Y. [3], Fumagalli G[1], Jones R[2], Mapelli L[2],
Mornacchi G[2], Polesello G[1], Prigent D[2], Qian Z[3], Sanchez-Coral E[2], Skiadelli M[2], Spiwoks R[2]

1: Dipartimento di Fisica dell'Universita'e Sezione INFN di Pavia, Italy
2: CERN, Geneva, Switzerland
3: Centre de Physique des Particules de Marseille, IN2P3, France

The data acquisition of a HEP experiment requires the setting, querying and manipulation of many parameters to be accessed by different parts of the DAQ system. Object-Oriented DataBase Management Systems (OODBMS) appear to fit the requirements of such applications quite successfully. In this paper we present our experiences and impressions while evaluating two commercial OODBMS, GemStone and ITASCA, as part of the RD13 project to study DAQ systems in view of LHC-like experiments. The goal of the evaluation was to compare the two systems and study the applicability of the Object-Oriented approach with respect to the relational systems already in use.

1. INTRODUCTION

A data acquisition system needs a large number of parameters to describe its hardware and software components. The RD13 DAQ [1] system currently uses four databases to store such information: Hardware configuration (describes the layout of the hardware in terms of crates, modules, processors and interconnects); Software configuration (describes the layout of the software in terms of processes, services provided, connections and host machines); Run parameters (e.g. run number, recording device, Level 2 trigger state etc.); Detector parameters (information pertaining to the detector and defined by the detector group themselves within a fixed framework)

2. EXISTING DATABASE SYSTEM

The QUID database system [2] is currently used to store this data. QUID allows the modelling, storing and handling of the data but it is not a full database management system since it relies on the host's file system for storing data and does not provide a multiuser environment. It has some other important deficiencies:

- all the database is in the memory of the application which sets strict limitations to its size
- no referential integrity or concurrency control is provided
- no schema evolution facilities are provided. If a schema change takes place then the data that correspond to the old schema are no longer valid.

3. EVALUATION OF OODBMS

In view of these limitations we have evaluated alternative DBMS with the intention of replacing our existing databases and extending their use to other areas of the DAQ. We chose to investigate OODBMS and after a market-survey selected two commercial systems.

3.1. GemStone

GemStone [3] merges Object-oriented concepts with those of database systems and provides an object-oriented database language called OPAL (an extension of Smalltalk-80) which is used for data definition, data manipulation and general computation. Application programming interfaces (API) for Smalltalk, C, C++,

Fortran, Pascal and Ada exist. It is one of the most mature OODBMS on the market.

3.2. ITASCA

ITASCA [4] is a recent commercial extension of the ORION-2 research prototype from MCC (Microelectronics and Computer Technology Corporation). It runs on UNIX platforms and is implemented in Common LISP. APIs exist for LISP, C and C++.

4. H/W DATABASE

To evaluate these products we chose to re-implement the H/W configuration database and its applications using the tools and techniques provided by the OODBMS systems.

4.1. VIC Initialiser application

The initialization of the DAQ system requires the initialization of the VIC bus by mapping the VIC modules memory space and the initialization of some other modules. We implemented this application in GemStone by creating basic object classes that represent the various hardware components. Relationships were made between these classes and instances created that correspond to the H/W setup in our laboratory. The GemStone version of this application proved smaller, simpler and more easily modified than the QUID implementation.

4.2. System Testing and Diagnostics application

This second application is used to check if the configuration contained in the database is consistent with the real system. This involves executing test programs on either the front-end processors or workstations that are connected to the VIC bus of the system. It is necessary to execute user code on the client machines and this capability is provided in GemStone by loading C code in the object methods.

4.3. Integration of database access with the data flow protocol

As a test of the applicability of the ITASCA OODBMS, we modified the DAQ so that an access to the database was included in the data flow protocol which transports the physics data. The idea was to measure the delay introduced by making a database access on a per event basis. This was possible because the client part of ITASCA had been ported to the front-end processors. The results showed only a minor degradation in the event rate when the client-side object caching capability of ITASCA was used.

5. OODBMS COMPARISON

GemStone lacks some of the advanced features that newer systems provide such as composite objects, versions of objects and sophisticated integrity constraints. The schema evolution and authorization control facilities are limited and the notification mechanisms are not very efficient. However, GemStone successfully supports:

- client/server architecture for distributed applications on heterogeneous platforms and CASE tools for complete application building
- dynamic loading of C functions and execution of code on the client machine
- concurrency control with very efficient indexing mechanisms

ITASCA is a much younger system with some fairly advanced characteristics but it lacks a graphical CASE tool for schema design. The C code loaded in the LISP methods can only execute on the server machine. We expect these problems to be solved in future releases but the current version supports some very useful features:

- complete data model, including composite objects, versioning and integrity constraints.

- complete set of schema evolution facilities that allow the dynamic modification of the schema
- large object storing and retrieving techniques with a complete and mathematically founded query language
- flexible authorization control
- private/shared policy for cooperative multiuser development of applications
- schema-integrated daemon-based notification mechanism, unique to ITASCA

6. FUTURE WORK

Driven by the favorable results of the evaluation, we can foresee other areas in which the use of an OODBMS would be beneficial to the DAQ system. The existing databases of the DAQ system can be consolidated into a single data model and implemented using the OODBMS.

6.1. Error Message database

Currently the error messages are stored in an ascii file that is maintained by hand. A database would ease the maintenance and searching of these message definitions as well as allowing multiple versions to be stored.

6.2. Bookkeeping log

This database contains information about the run of the system e.g. how many events were recorded, on which tape etc.

6.3. Storage of event data

An OODBMS could be used for storing physics data rather than writing it to a tape thus allowing analysis to start during a run. The events could be inserted in the database and used by the monitoring processes or any other kind of analysis tasks. The advantages of doing this are:

- events would acquire a structure based on the sub-events composition [5]
- analysis of the data would be easier and more efficient if it were based on a query language. This would allow retrieval on a per event or sub-event basis.
- the statistics of the events collected could be stored with the events to provide a complete view of the run
- interfacing the database with graphical tools could automate the visualization of the events

However there are three major issues that have to be faced.

6.3.1. Unstructured data

Most of the objects contained (i.e. parts of the events) in the database are going to be long and unstructured and thus special mechanisms should be provided by the system for their efficient retrieval. Today's OODBMS are capable of handling such data (multimedia data like image, sound, text etc.) in contrast with the relational systems that can only contain normalized tables.

6.3.2. Space Requirements

A very large amount of data needs to be stored in the database for each run implying the database may extend over more than one physical device. This issue has already be solved and most of the commercial DBMS provide transparency in case the database is physically on more than one device.

6.3.3. Performance

The performance of the system has to conform to real-time applications and so special mechanisms for caching and clustering of the data should be provided by the DBMS. Some of the OODBMS cache objects in the memory of the applications but consistency problems can occur. However there are good prospects for enhancing these mechanisms in the near future and it is clear that OODBMS are better placed

since they use OO programming languages as database languages.

6.4. Histogram database

The histograms that occur from the analysis of the data can be stored in a database as well. The advanced features like indexing, caching and long data management make the manipulation of such a database not only technically feasible but also efficient.

7. CONCLUSIONS

We have briefly presented our evaluation of the GemStone and ITASCA OODBMS. We have seen some important advantages over relational systems:

- schema design is more direct since the object oriented model is closer to the real world.
- database maintenance is easier due to the schema evolution facilities and modular design
- the identity concept that gives one internal pointer to each object throughout its life protects the consistency of the database an help model similar objects
- not only data but also pieces of code (methods) that run on the data are stored. Consequently a whole application can be stored and executed which also eases maintenance
- the inheritance concept makes code more easily reusable
- the expensive join operations of the relational systems have been substituted by the composite object notion, which combined with the clustering mechanism can improve the performance of composite object retrieval

Encouraged by the positive results of our evaluation, we have indicated how the use of an OODBMS can be extended to other areas of the DAQ and possibly event recording. Further details of this work are available through the World Wide Web [6].

8. ACKNOWLEDGEMENTS

We wish to thank the CORTEX project at CERN for giving us access to the GemStone system for the purpose of the evaluation.

9. REFERENCES

[1] L.Mapllei et al., A Scalable Data Taking System at a Teatbeam for LHC, CERN/DRDC 90-64, CERN/DRDC 91-23,CERN/DRDC 92-13, CERN-DRDC 93-25.

[2] Quid version 2.0 User Manual, Artis srl, November 1992

[3] An Introduction to GemStone V3.0. Servio Corporation 1992.

[4] ITASCA Distributed Object Database Management System. Technical Summary for Release 2.1. Itasca Systems, Inc. 1992.

[5] G. Ambrosini et al. Event Format in the Read-Out Module, RD13 Technical Note 109. Available through WWW at "http://rd13doc.cern.ch/welcome.html".

[6] M. Skiadelli, Object Oriented database system evaluation for the DAQ system. RD13 Technical Note 108. Available through WWW at "http://rd13doc.cern.ch/welcome.html".

Session 2
Hardware Architectures

Using a field programmable gate array as a dedicated coprocessor for the Texas Instruments' TMS320C40 DSP

A.J. Borgers, F.B.T. Golbach, A.W. Lodder, W. Lourens, A.H.D. Ockeloen, A.J.M. de Vries,
Faculty of Physics and Astronomy, Utrecht University,
P.O Box 80000, NL-3508TA Utrecht, The Netherlands.

(presented by: C.Th.A.M. de Laat)

Abstract

For real-time feature extraction, in the future Large Hadron Collider experiments very fast data processing will be needed. In this paper we look at the possibility to speed up a TMS320C40 DSP by means of a user-definable coprocessor that can operate in parallel with the C40. This combination will be able to decrease the total execution time of differnt kinds of algorithms considerably.

REDUCING EXECUTION TIME OF COMMON-TYPE ALGORITHMS

For the future 'Large Hadron Collider' (LHC) experiments at CERN, new trigger algorithms are under development. These algorithms are generally based on simple threshold equations, sensitive for peaks or specific gradients in the energy deposition in a calorimeter, and have to be executed in real time by distributed processing. We are studying whether a network based on some hundreds of digital signal processors can meet the requirements for data processing. The work described is done within the context of the CERN-EAST/RD-11 collaboration [1].

Crucial in the design of a multiprocessor based system is the balance between data processing and interprocessor communication [2, 3, 4]. To realize a maximum throughput in the DSP network we put the emphasis on an optimum combined action of hardware and software. Some algorithms are very well suited to be accelerated by dedicated hardware. Therefore, we are working on adapting the possibilities of a commercial processor (Texas Instruments' TMS320C40 [5]) to the demands of the application by combining each C40 with a user-definable hardware coprocessor implemented in a Xilinx LCA (Logic Cell Array). We want to use the LCA as a kind of 'accelerator' for the C40 by implementing functions of the trigger algorithms. The functions added run in parallel with the instructions executed in the C40. In this way algorithms that would take several instructions using a C40 (or any other processor) can be hardware implemented in the LCA, reducing the total execution time considerably. As the hardware within the LCA is reprogrammable, its use can be adapted to many kinds of problems.

To give an impression of the possibilities, we will give some simple examples. First we notice that many algorithms use some kind of histogramming to

count events. After some time these histograms need to be cleared. This means clearing all elements, one after another of the array containing the histogram. The amount of time to do this linearly increases with the size of the histogram. Using a dedicated hardware solution, the memory elements forming the array can be located in the LCA, thus having the possibility of clearing all of them in one single instruction.

Many algorithms need to find an extremum of a specified number of inputs. Using the classical software solution this means comparing all inputs in a loop, thus taking a relative long time. When all data is stored using the LCA as an active memory component, finding a maximum or minimum can be implemented in hardware, thus reducing the time to find it to a single C40 cycle.

The variety of functions that can be performed by the LCA-coprocessor is almost unlimited, but it shows that the best performance is achieved in systolic and vector processing tasks that take a lot of time using software solutions.

Some complete examples of how the execution time of algorithms of interest for feature extraction can be reduced, are presented in [6]. The execution time of these programs is reduced by about a factor 3, but some parts of it are reduced by a factor 65. This indicates that the reduction of execution time using the LCA-coprocessor (highly) depends on the algorithm involved. There will be algorithms that can be enhanced by a factor 10 or more, but others are not suited for hardware implementation at all. In our opinion it is recommendable to keep the possibilities of this kind of hardware in mind when developing new algorithms.

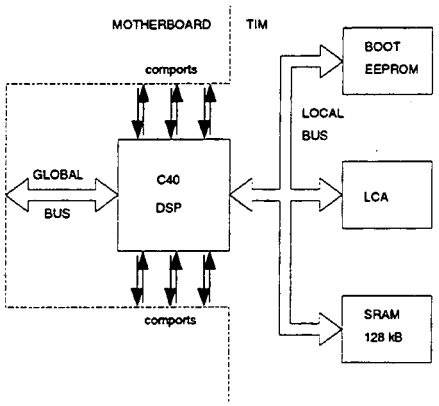


Figure 1. Block diagram of the new TIM design.

HARDWARE UNDER DEVELOPMENT

Until now, the concepts of the coprocessor were developed and analysed using digital simulation software. To be able to test the real situation by measurements , we are developing our own TIM (Texas Instruments Module [7]) containing a C40-LCA combination. A TIM is an universal module, which can contain a C40, memory, ADCs and other hardware. TIMs are frequently used because they are compatible with each other, offering the possibility to change the processor modules without changing the motherboard. The block diagram of the TIM under development is outlined in Figure 1.

First of all we want to notice that all six communication links [5] of the C40 will be kept available on the TIM connectors. This is conform the TIM specification, and of crucial importance to the future motherboard we plan to use [6].

The LCA is connected to the C40 using the local bus. This way the global bus is completely free for other purposes. In the TIM concept, the local bus is

only available on the TIM. The global bus, offering the same functionality, can be made available to the motherboard through the TIM connectors.

The LCA is configured by a serial PROM. This PROM will be connected using a LIF-socket, so it can be changed easily to give the LCA-coprocessor another functionality.

We plan to place at least 128 Kb RAM on the TIM. Although our strategy is to use internal C40-RAM as much as possible [4], external RAM can be useful for data storage, or, in our case, for a look-up table. Furthermore, we want the TIM to be suitable for universal use, so external RAM is desirable.

Finally, we want to include a EEPROM (Electrical Erasable PROM) on the TIM. This EEPROM will be free available to the user. In our applications the kernel of our C40 communication software will be stored in it [4]. It will be used further to give the C40 a unique ID code, which will be of interest in multiple-C40 based networks.

CONCLUSIONS

The concept of an LCA as a programmable coprocessor can reduce the execution time of common used type algorithms. Programming the coprocessor requires extra effort, but it will reduce the number of C40s needed by a factor 2 to 5. The acceleration that can be achieved depends mainly on the algorithms used. As soon as we have the completely functional C40-LCA TIM we will do measurements executing a collection of different programs. We expect the results of the future measurements will support our present estimations.

A TIM is rather expensive at the time. Reducing the number of C40s needed would reduce the costs of hardware considerably. Keeping in mind the number of processors that might be needed in the LHC experiment, this can be a serious advantage. Less processors also means less complexity and higher reliability.

REFERENCES

1. EAST (RD-11), Proposal and Status Report: CERN/DRDC 90-56 and CERN/DRDC 93-12
2. A.J. Borgers, F.B.T. Golbach, W. Lourens, *Data transfer rates using the communication ports of the Texas Instruments' TMS320 C40 Digital Signal Processor*, CERN/EAST note 93-14.
3. J.C. Vermeulen, *Data transport with the Texas Instruments' TMS320 C40 DSP*, CERN/EAST note 93-21.
4. A.J. Borgers, F.B.T. Golbach, W. Lourens, A.H.D. Ockeloen, *Interactive communication and debugging tools for a multiple TMS320 C40 system architecture*, CERN/EAST note 94-01.
5. Texas Instruments, *TMS320C4x User's Guide*, 1992.
6. A.J. Borgers, F.B.T. Golbach, W. Lourens, A.H.D. Ockeloen, A.J.M. de Vries, R. K. Bock, I.C. Legrand, *Using a field programmable gate array as a dedicated coprocessor for the Texas Instruments' TMS320 C40 DSP*, CERN/EAST note 94-xx.
7. Texas Instruments, *TIM-40*, TMS320 C4x Module Specification, 1992.

D0 FARM PRODUCTION SYSTEM *

Kirill Denisenko, Pushpalatha Bhat,
David Fagan, Lee Lueking, Adam Para,
Kurt Ruthmansdorfer, and Matt Wicks
Fermi National Accelerator Laboratory
Batavia, IL 60510 USA

Abstract

A solution for a highly I/O intensive production system for a Fnal Collider experiment (D0) is presented. The implemented system is based on a parallel processing of events on a collection of Unix processors and supports flexible allocation of resources between various production projects. The hardware architecture is supported by FNAL Farms group and utilizes a loosely coupled parallel array of workstations. The peak capacity of the system was around 1500 MIPS. The exact allocation of computing power was based on the amount of data recorded by the D0 detector. All reconstruction was done in a near real-time mode.

INTRODUCTION

The specifics of D0 collider detector data production especially during 1992-1993 Collider Run was that the ratio of the event size to CPU time necessary to reconstruct this event was atypically large. An average non-reconstructed event size was around 500 KB, whereas CPU time to process it was only around 20 sec. This presented us with a challenge to create and implement a system that would utilize fully the capacity of Ethernet-based network and at the same time allow us to use FNAL standard hardware configuration.

Production Environment

This system consists logically of two parts: a fully automated batch job processing and book-keeping tightly coupled to FATMEN [1] and RDB production databases, and a customized network package based on BSD sockets that allowed us to achieve rates around 700 KB/sec and processor nodes utilization around 70% on average (including downtimes).

A Hardware Organization

The hardware organization of the D0 farm eventually converged to a standard Fnal farm-architecture [2], see fig. 1. It consists of two I/O servers (SGI 420) each supporting 8 Exabyte-8500 tape drives, 12 GB of disk storage, and three Ethernet controllers. A string of maximum of 8 SGI Indigos R3000 was attached to each of the Ethernet controllers as an isolated Ethernet segment. The disks that

*This work is supported by the U.S. Department of Energy

contain the production manager and reconstruction codes as well as production manager databases were served by a separate SGI 4D/35 computer. Fig.1 shows the hardware organization of one of the I/O servers.

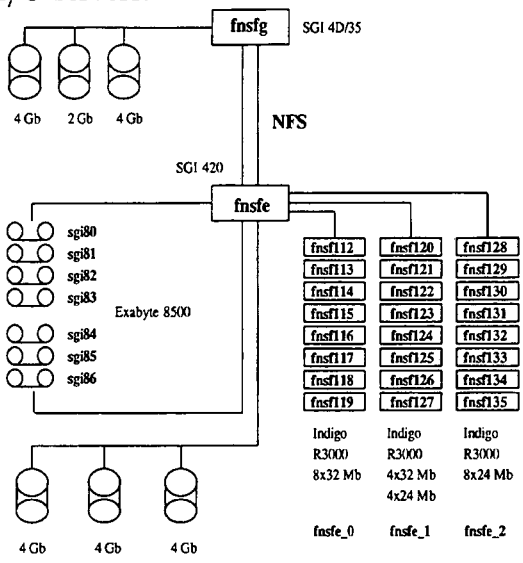


Fig.1

A Batch System

A batch processing sub-system allows on-line reallocation of worker nodes between projects and supports several I/O options (8500 exabyte tape drives, direct network transfers). Together with a parallel mode (when events from the same dataset are shipped to different worker nodes) it allows sequential (each node processes its own dataset) and standalone processing (each processor serves as I/O node for itself; need to be equipped with all peripherals). The core of the system is TPM production manager [2], which controls the data flow through the system. The I/O spoolers based on FNAL RBIO package [3] read datasets from input 8mm tapes and store them on disk. As soon as the previous dataset has been processed, a new one is picked-up by the parallel server which start to serve events over the network to the worker nodes. Another server collects the events processed by the worker nodes and stores them in the output disk areas, where they are picked-up by the I/O server and written to tape. This production manager also controls the allocation of hardware resources (tapes and disk) dynamically redistributing available resources between farm processes. An experimenter running a farm has to supply only a file which dedicates certain amount of CPU power (worker processors) to a given project (by project we denote a group of parameters accepted by the D0 Reconstruction Code). Fig.2 shows the simplified TPM flow-chart.

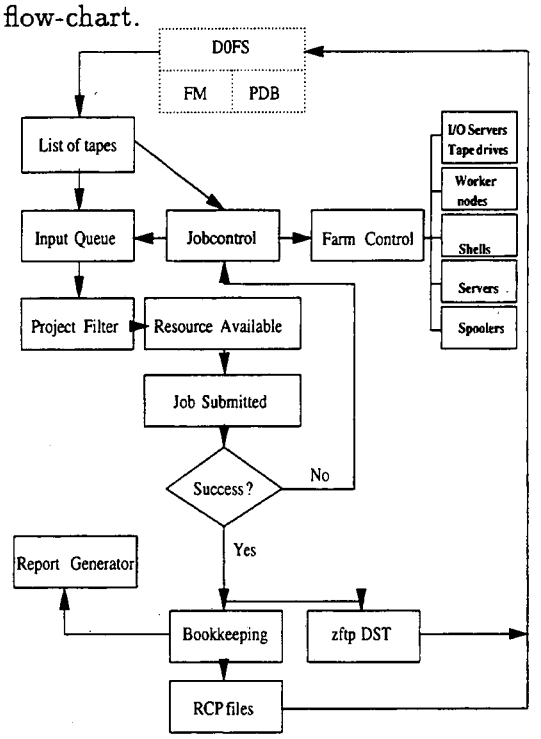


Fig.2

Network Issues

It was extremely important to get all available performance out of the network since it defined the number of worker nodes attached to an Ethernet controller on the I/O server. We wrote a package

that allowed any processor on the network to connect to the given I/O server using a predefined port via BSD socket mechanism and receive/send events. A custom message-passing protocol to minimize network transfers has been developed. This scheme allows for additional flexibility in dynamic distributing of worker processors between projects - you can do it any time without interrupting the execution of any project. The worker processor code includes the bufferization of the events via the shared memory facility - an event received by the worker processor is stored in the shared memory area and is queued to the reconstruction program, which, when finished with an event, queues the next one. The reconstruction output is also queued to the network.

Communication With D0 Databases

One of the goals of the D0 production system was to seamlessly integrate the data flow with the rest of the D0 environment, most prominently with Fatmen [1] and PDB (VAX RDB-based production database). In order to achieve this for each dataset a so-called RCP-file was created which contained all information related to a derivative dataset. These RCP-files were then shipped together with the datasets themselves to an appropriate database input queue where they were picked-up by database servers and subsequently catalogued.

Exploitation of the system

Most of the time this system was run by one computer professional providing technical support and one graduate student watching the system condition. When system matured and grew more robust, the time to maintain it for a person who is familiar with the system won't exceed 2-3 hours a day. D0 production shifts are considered to have the same status as general detector shifts.

Accomplishments

This system was used by D0 experiment for its farm-reconstruction since the beginning of the data taking and is continued to be used. At peak times, when there were 42 worker processors attached to two I/O servers, the farm was able to process about 130,000 D0 events a day. A total amount of events processed through Run 1A was close to 14,000,000.

ACKNOWLEDGEMENTS

The authors thank Steve Wolbers and Frank Rinaldo for their continuous support of this project, and Data Center Services personnel for their help in carrying out production tasks. One of us (KD) is grateful to Steve Bracker and Krish Gounder (UMISS) for help with learning TCP/IP programming.

REFERENCES

1. K.Genser, FATMEN as the D0 File and Management System, CHEP'92, Proceedings, CERN 92-07, Geneva 1992, pp. 759-762.
2. K. Denisenko, TPM Production Manager Operator's Manual and User's Guide, D0 Note 2024, January, 1994.
3. C. DeBaun and S. Rojak, Using the Raw Buffered I/O Interface in FORTRAN Programs, FNAL SU0033-S1, 1993.

PRODUCTION FARMS AT FERMILAB *

Mark Fischler, Frank Rinaldo, Stephen Wolbers
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia IL 60510 USA

Abstract

UNIX Farms at Fermilab have been used for more than three years to solve the problem of providing massive amounts of CPU processing power for event reconstruction. System configurations, parallel processing software, administration and allocation issues, production issues and other experiences and plans are discussed.

INTRODUCTION

The UNIX Farms Systems at Fermilab are a large and growing example of loosely-coupled parallel computing. There currently are over 300 UNIX workstations (SGI and IBM) in the Fermilab Farms in full production, with a rated computing power of over 10000 MIPS. In addition to the CPU's there are quite a few peripherals (disk and tape) attached to the systems to allow access to the large datasets that are to be processed. The Farms are utilized for reconstruction (pattern-finding and fitting) of the raw data coming from high-energy physics experiments at Fermilab and for CPU-intensive Monte Carlo simulations. Many experiments have used the Farms to successfully do event reconstruction for very large datasets.

*This work is supported by the U.S. Department of Energy under Contract No. DE-AC02-76CH03000.

The software for parallel computing (**cps**) and for batch scheduling (**cps-batch**) is maturing and improving in capability and robustness. Additional tools for debugging and performance as well as for computer operations (tape mounts) are being developed and delivered in order to make maximum use of these computing cycles. This paper discusses experiences in making large systems work, including efforts put into improving those tools.

The systems are currently being used to complete the reconstruction of the multi-Terabyte datasets from the last fixed-target runs at Fermilab and for the essentially real-time reconstruction of data from the 1994 Collider run.

FARM HARDWARE

The farms consist of UNIX workstations and servers configured logically in two different sets. The vast majority of the workstations (over 300) are "worker

nodes": Each is a UNIX workstation with 16-24 MB of memory, a local system disk, and ethernet and power connections. These workstations are a mix of SGI 4D/25, 4D/35, and R3000 Indigo's, as well as IBM RS6000/320, 320H and 220's. The combined CPU power is approximately 10,000 MIPS. The other set of workstations and servers are designated as I/O nodes; these provide connectivity to the datasets (via SCSI-connected 8mm tape) as well as additional disk space and memory to handle other job functions. There are 17 I/O nodes: SGI 4D/420's and Challenge XL, and IBM RS6000/580, 590, 530, 530H, 320 and 320H workstations.

Configuration

The balance of I/O and worker nodes must be tuned to obtain maximum utilization of the available compute power. The average ratio of worker nodes to an I/O node is approximately 16—the ratio used for a given application is determined by experience and by a knowledge of its CPU/IO needs. To avoid saturating ethernet segments, the worker nodes are divided into subnets attached to routers, with each subnet consisting of between 8 and 20 workstations. The users of the farms are assigned production systems which contain an I/O node, tape drives on the I/O node, and multiple worker nodes. Each production system is typically assigned to only one experiment and each experiment is assigned multiple production systems depending on priority and need.

FARM SOFTWARE

The success of the UNIX Farms depends to a large extent on the software provided to utilize the available CPU power.

cps

cps (Cooperative Processes Software)[1] was written to allow an arbitrary program to share data and computing across many processors. The package allows many different types of parallel computing but typically in event reconstruction the structure used is to have an input and output process and many reconstruction processes. Events are read from tape or disk, passed to a process on a worker node, reconstructed there, passed to an output process and written onto disk or tape. The changes to a typical reconstruction package to allow it to run with cps are fairly modest and do not require fundamental restructuring of the program. Performance tuning of the components of **cps** has been done as differing needs were identified and bottlenecks were found and removed. **cps** relies on a single job manager process, and thus is not indefinitely scalable. But in our experience the relevant size limit is imposed by I/O requirements, not by **cps**-induced bottlenecks.

cps-batch

Due to the nature of farm computing at Fermilab (hundreds or thousands of tapes to be processed) a batch system is necessary to allow queuing of jobs. There is no ubiquitous batch management tool in the UNIX environment. **cps-batch** is a simple queueing mechanism which allows users to submit multiple jobs to be executed one at a time on each production system. Each experiment is assigned mul-

tiple production systems to use as part of their overall farms allocation. Enforcement of resource control is primitive: A given system is assigned to only one experiment.

OCS

As farm usage began to grow, it became clear that tape-mounting by operators was an extremely important part of the overall processing problem. Reliable and fast tape mounts are needed to handle the hundreds of tapes that are processed each day on the many I/O nodes of the systems. OCS (Operator Communications Software)[2] allows robust tapedrive allocation and tape-mounting in a distributed UNIX environment such as the farms. It handles the bookkeeping needed to tell operators what tapes to mount and where, and to cope with operator replies indicating anomalous situations. This system is invaluable in providing a successful production environment.

Utilities

Many utilities have been written to manage, debug, operate, and tune the farms. The **jobview** log file tool allows post-mortem analysis of the flow of data and control, to help tune the system. A distributed debugger, **jmdb**, lets the user debug the parallel aspects of his code. Activity on entire farms can be monitored coherently using the graphical **cps_xpsmon** and **xcps_sysmon** tools, and **xfalive** watches for hardware failures by probing the worker nodes in various ways.

CONCLUSIONS

The UNIX Farms at Fermilab have been extremely effective in solving the needs of event reconstruction of high-energy physics data. The Farms have allowed both collider detectors to reconstruct their data in real-time and to handle special reconstruction needs. The success of the farms has depended on large efforts from many people to produce a production system which handles many hundreds of tapes each and every day. Many issues which were difficult enough to handle in the context of dedicated systems would present insurmountable problems in the context of scavenging time on diverse under-utilized workstations.

The current farms are being modified to handle the increased data-taking of the current collider run, and to expand the range of HEP data processing activities to which they can be applied. Further improvements and enhancements will be made as technology allows and as demands grow.

ACKNOWLEDGEMENTS

The development and maintenance of the hardware integration and software products make the farms a valuable resource. Credit for the success of this effort is due the Fermilab Farms Group and UNIX Systems Support Group.

REFERENCES

1. M. Fausey, *et al.*, "CPS User's Guide, CPS Version 2.9", Fermilab Computing Division Library GA009, June 24, 1993.
2. M. Fausey, M. Schweitzer, "Operator Communications Software Reference Guide V2.2", Fermilab Computing Division Library GA0012, April 1, 1994.

A High Performance Computing Environment for Physics Analysis

Karsten Künne,
Deutsches Elektronen-Synchrotron (DESY)
Notkestraße 85
D-22607 Hamburg, Germany

Abstract

The needs for high performance computing systems in terms of CPU- and I/O-Performance in High Energy Physics are constantly growing. This paper presents an overview about the High Performance Computing Farm at DESY. Also presented are experiences with that Computing Farm and future developments.

Introduction

The DESY experiments H1 and ZEUS, in the future also HERMES, have big requirements for processing power combined with large I/O handling capabilities.

In 1992 it was decided to build up a UNIX based High Performance Computing Platform in order to run Physics Analysis at DESY.

Requirements

On the hardware side such a system needs enough CPU performance, and here especially integer performance, and a high I/O throughput. Another requirement is a fast interconnection between the machines. This is needed because most of the data is typically held on tapes and needs to be staged via this interconnection to disks and there is also a need to access data on disks which are connected to another machine. Other requirements are that the hardware should be scalable and extendable and it should be easy to use it.

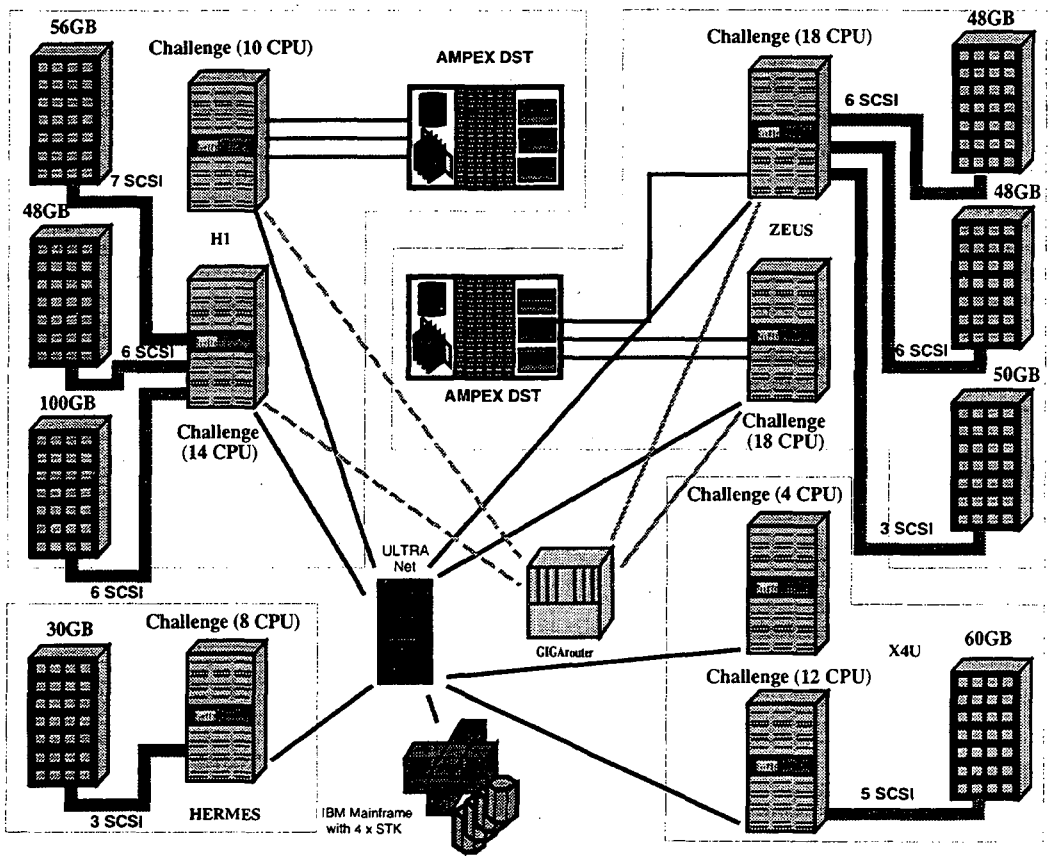
The software for that system needs to be able to manage large amounts of batch jobs, to manage the diskspace connected to the system and to transfer data between tapes and disks (tape staging).

Hardware

Figure 1 gives an overview about the currently installed hardware configuration.

CPU and Disk Servers

The currently used CPU and disk servers are 7 Silicon Graphics Challenge machines. These machines are multiprocessor machines and contain together 84 R4400 CPU's running at 150 MHz. This gives a CPU performance of approx. 10000 MIPS or 120 IBM/ES9000 CPU's. The main memory of all machines together has a capacity of 6 GB and there are approx. 440 GB diskspace connected



System Overview

to these machines.

What are the advantages of these machines. First, the Challenge machines are Symmetric Multiprocessor (SMP) machines with shared memory. Every processor in a machine has access to all data, so it is in general easy to write programs. Tests at DESY also showed that these machines scale up well to 30 CPU's in one machine. If 30 identical programs are running on a 30 CPU machine every program performs with approx. 95 % of the performance which was measured with only one program running. Another very important advantage of the Challenge machines is their superior I/O performance. Measurements at DESY showed a transferrate of more than 14 MB/s for a single stream reading from a filesystem consisting of 4 disks which is 4-way striped.

Interconnection

The high speed interconnection between the machines is currently done with Ultranet.

There is a new interconnection scheme between the Challenge machines based on HIPPI under test which will render Ultranet superfluous [1].

Software

The software described in this section is the piece of software which resides between the pure operating system and the physics application software.

Queueing System

There exists a widely used queueing system for UNIX, called NQS [2]. NQS has most of the abilities which were needed for a Physics Analysis batch system and it is also used on a lot of similar systems in the HEP community, so it was chosen as the queueing system. Over the time of its usage NQS showed that it is in general sufficient for the task but there are some shortcomings. One important point which is missing in NQS is, by example, checkpointing.

Some tests with other queueing systems, Loadleveler and Codine, are currently running.

Diskspace Management and Remote File Access

There are packages existing for these purposes, the Disk Pool Manager (DPM) and the Remote File Access System (RFAS or RFIO), written at CERN [3]. These packages were installed at DESY. In general, both packages fulfil the requirements but, especially with the Disk Pool Manager, there are some deficiencies. The main limitation here is that the Disk Pool Manager doesn't keep track of allocated diskspace so that more disk space can be allocated than is available.

Tape Staging

The main tape server for the Physics Analysis system is still the IBM/MVS mainframe with the four connected STK robots. In order to stage tapes between IBM cartridges and the disks on the Challenge machines, a new software package had to be written, called tpread/tpwrite. This package is partly based on the tpread/tpwrite package written at CERN, but because of the way the tape access is handled on the MVS system, most parts had to be rewritten. The transferrate of tpread/tpwrite is in between 1 and 1.5 MB/s. That means, one typical tapedataset of 200 MB is transfered in about 5 minutes.

Experiences

It turned out that the hardware was capable of delivering a very large amount of disk I/O bandwidth. The peak rate which could be observed was far above 20 MB/s during normal production periods (no special benchmark runs).

The usage of the whole system grew up continiously since it went into production and now reached a remarkable high level. All the disk space is filled up to more than 90 % and the cpu's show only small idle times. The following is an usage example from the 14-CPU machine on April 11th, afternoon:

- 1124 processes, 800 MB userdata
- 126 users logged in, 80 active
- 13 batch jobs
- 17 PAW sessions
- 194 xterms, 60 window managers

Future Plans

The first change which will happen is the high performance network integration based on HIPPI connections [1].

Another important change will be the introduction of a new mass storage system that will also help to tightly integrate the STK robots, which are currently connected to the IBM/MVS mainframe. One of the goals of that project is also to

present a more unique view of all the data to the user and to more or less hide the tape staging process [4].

It is also foreseen to introduce AFS into the system. The main advantage of AFS here is the unique global namespace. As a perspective the catalog of the mass storage system could somehow be integrated into AFS. That would be a major step on the way to a global catalog of all the files in the system.

Conclusions

Over the time of usage of the system it turned out that the chosen hardware is an excellent base in order to run Physics Analysis jobs [5]. But, without some software between the operating system and the physics software itself it would be very difficult to use the system, perhaps impossible. The chosen software here is sufficient for the purpose but there are some deficiencies. It is necessary to integrate disk pool management and tape staging into a common, more sophisticated, mass storage management system, which keeps track of the location of the data and automatically allocates diskspace and does tape staging.

REFERENCES

1. M. Ernst, "Using High Performance Interconnects in a Distributed Computing and Mass Storage Environment", these proceedings
2. B. Kingsbury, "The Network Queueing System", (1992)
3. J. Baud *et al,* "SHIFT - User Guide and Reference Manual", CERN (1993)
4. M. Gasthuber, "Distributed Mass Storage and Management Systems at DESY", these proceedings
5. R. Gerhards *et al,* "Experience with a UNIX based Batch Computing Facility for H1", these proceedings

EXPERIENCE WITH A UNIX BASED BATCH COMPUTING FACILITY FOR H1

Ralf Gerhards, Uwe Krüner-Marquis, and Zbigniew Szkutnik
Deutsches Elektronen Synchrotron, Notkestraße 85, D-22603 Hamburg

Abstract

A UNIX based batch computing facility for the H1 experiment at DESY is described. The ultimate goal is to replace the DESY IBM mainframe by a multiprocessor SGI Challenge series computer, using the UNIX operating system, for most of the computing tasks in H1.

Introduction

With the advent of HERA, the first electron proton collider ever built, and its two large experiments H1 and ZEUS, the requirements for computing at the DESY laboratory in Hamburg increased considerably. These needs could not be fulfilled anymore with conventional mainframe computers. DESY, and the H1 experiment in particular, therefore started a project called DICE (DESY Integrated Computing Environment), closely following the SHIFT project at CERN [1]. The basic idea is to use the less expensive, high performance products provided by workstation industry, interconnected by a high performance network.

DICE has been started about two years ago and, thanks to a close cooperation between the DESY computer center groups and the H1 collaboration, became a useful and powerful platform for H1 computing, increasing the available CPU power for H1 by an order of magnitude [2].

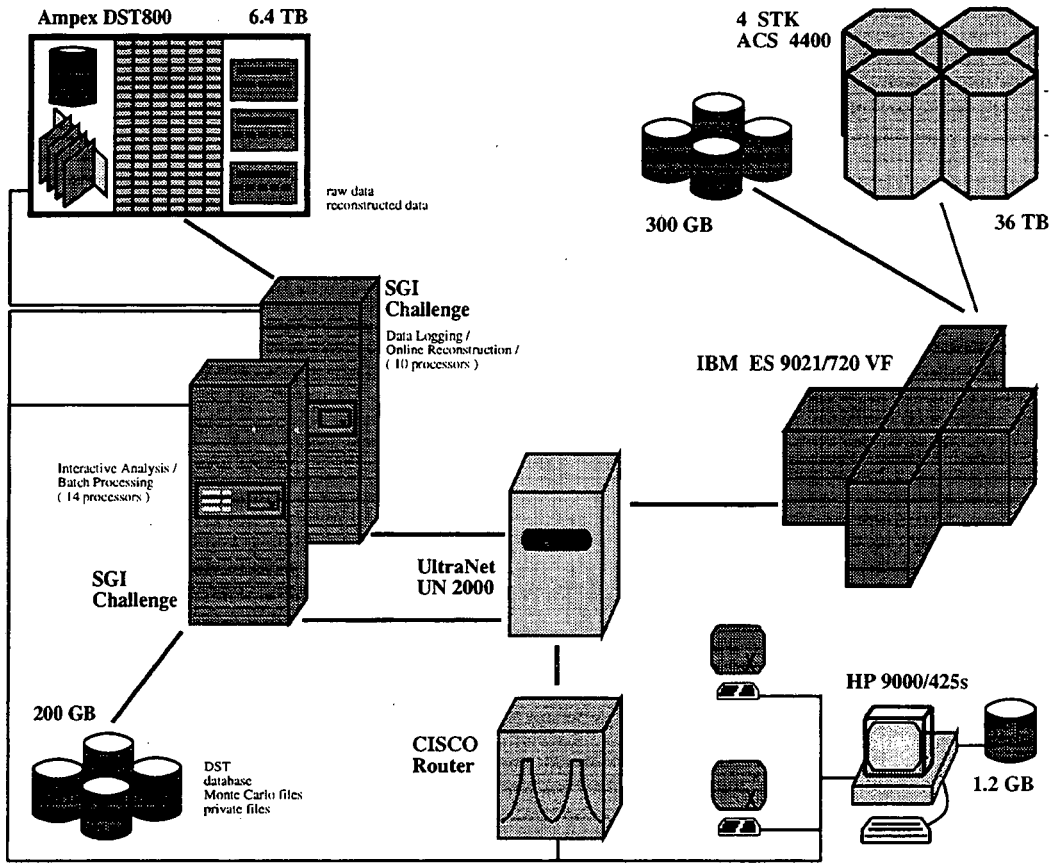
Unlike the original CERN project, however, the DICE hardware is also largely used for interactive analysis by H1 physicists. Therefore, the requirements on DICE differ in some parts considerably from those on SHIFT.

System Overview

Figure 1 shows an overview of the system architecture setup for H1. The H1 experiment finally decided on a single multiprocessor SGI Challenge series computer[1]. This SGI is supposed to take over the role of the IBM mainframe for H1 computing[2]. It is equipped with 14 150 MHz MIPS R4400 processors and 1.5 GB physical memory, thus increasing the available CPU power for H1 by about a factor of 10. It is connected to the DESY world via fast UltraNet and FDDI links as well as Ethernet. Fast access to data both on tapes and disks is provided. The disk space amounts to about 200 GB

[1] A second SGI Challenge computer is used for data logging and event reconstruction[3]

[2] About one third of the IBM resources are presently assigned to the H1 exeriment



H1 Offline Computing Environment

connected via 19 SCSI interfaces.

Major tape service is presently provided by four StorageTek Automatic Cartridge Systems (ACS 4400) with a total capacity of about 36 TB, using compactification, double density (36 tracks), and double length (3490) cartridges. Mass data are also kept on tapes in an Ampex TeraStore mass storage system, equipped with 3 DST 600 helical scan D2 recorders and connected to the SGI Challenge via SCSI interfaces [4]. The total capacity of the Ampex system amounts to 6.4 TB.

Login to the SGI is provided from a variety of terminals both for batch and interactive work. About 500 Xterminals are spread around the DESY site that allow direct login to DICE. Furthermore, personal computers (mainly Apple Macintosh) and workstations can be used for login. A uniform user interface based on X11/Motif is provided by the computer center.

Batch Computing

The first and major application for DICE is to free the DESY IBM from time consuming batch applications. As in SHIFT at CERN, H1 chose the classical UNIX batch system, NQS (Network Queuing System) [5], for scheduling of batch jobs on DICE. In the present configuration, four different queues are available, ranging in CPU time limits from 20 minutes to several days. About 10 batch jobs are running on average, and about 50 percent of the available CPU time is used for batch applications. Ac-

cess to the batch resources from other DESY computers is possible, in particular from the IBM mainframe, without interactive login to DICE [2]. However, NQS has some essential deficiencies in the job scheduling. Therefore, another (commercial) batch system, the LoadLeveler, is being tested right now.

Access to data is implemented in a machine independent way using the H1 standard input/output package, thus providing similar user interfaces on all platforms in H1 [6].

The disk space is managed by a *Disk Pool Manager*, which is also used to access disk files, and divided into a large pool used for general datasets, dedicated pools for the different physics working groups, and a pool for tape staging. Access to disk files is simplified by using the concept of index files or event directories [6].

Presently, tape access to the ACS is only possible via the IBM mainframe as the tape server, thus causing the main bottleneck of the batch facility. About 1000 tape requests are on average serviced per day by the IBM. The direct access from the SGI to the ACS is in preparation, using commercial software e.g. for hierarchical storage management, file catalogues etc [4]. The actual data transfer uses UltraNet with a minimal CPU requirement for protocol evaluation on both hosts. The *Remote File Access System* (RFAS) is used to access remote files on tape and disk servers in a very convenient way. A very promising new concept for data storage, based on the *IEEE mass storage model* is also discussed at DESY [7].

Interactive Computing

Unlike the SHIFT project at CERN, the H1 collaboration is using the DICE hardware also for interactive purposes, e.g. N-tuple analyses (PAW, LOOK), or code development. On average, more than 100 users are logged in to DICE simultaneously. It was seen rather early that the main limitation for the number of interactive sessions is determined by the physical memory of the machine. Presently, stable and convenient operation is possible with 1.5 GB memory. In addition, large improvement could be achieved by employing the concept of shared libraries to reduce the in-memory size of executables. In particular memory intense programs like X-applications, event display programs or programs for interactive analysis, profit considerably. Replacing an conventional mainframe of course also requires providing services like backup, dataset migration, accounting,and information services (mail, news etc.), which are presently just emerging. The H1 experiment is closely following recommendations from the DESY computer center groups for application programs (e.g. tin, pine, mosaic, ...).

Conclusion and prospects

An SGI Challenge series computer will replace the IBM mainframe computer in H1 both for interactive and batch usage. This implies a mainframe quality service in a UNIX environment and the porting of many tools and utilities. The "UNIX mainframe" is certainly able to serve the increased demands for batch processing. It is also well suited to serve the interactive requirements for a large High Energy Physics experiment like H1. It should, however, be emphasized that a full mainframe quality service is not yet available.

Hence, and because UNIX is an open system without too many restrictions, solid rules for users are mandatory. It is of considerable interest to follow the evolution towards the final goal of all projects like DICE, i.e. to end up with a UNIX based, mainframe-like computing facility.

Acknowledgements

The authors gratefully acknowledge the tremendous efforts by the DESY computer center groups in providing a convenient and usable computing facility for H1. The authors also express their gratitude to their colleagues in the H1 collaboration for many discussions and suggestions which helped developing and improving this challenging computing project.

REFERENCES

1. J. Baud *et al.*, "SHIFT - User Guide and Reference Manual", CERN (1992)
2. K. Künne, "A high performance computing environment for physics analysis", these proceedings
3. P. Fuhrmann *et al.*, "Data Logging and Online Reconstruction in H1", these proceedings
4. M. Gasthuber, "Distributed Mass Storage and Management Systems at DESY", these proceedings
5. B. Kingsbury, "NQS Manual" (1992)
6. V. Blobel, "The F-package for Input/Output", Proc. Int. Conf. Comp. in High Energy Physics, Annecy, France (1992)
7. M. Ernst, "Using High Performance Interconnects in a Distributed Computing and Mass Storage Environment", these proceedings

How to survive the interim, or making (users of) mainframes and farms mutually coexist

W. Wojcik, Y. Fouilhé, J. O'Neall

Centre de Calcul de l'IN2P3, F-69622, Villeurbanne, France

Abstract

The IN2P3 Computing Center (CCIN2P3) in Lyon/Villeurbanne recently inherited an entire new group of users. Altho we were already launched upon the gradual transfer of computing power from an IBM ES9000 mainframe to workstation farms, the abrupt arrival of these new clients, on a time scale incommensurable with their conversion to a UNIX–farm environment, obliged us to increase the capacity of our mainframe at a time when other Computing Centers are cutting back drastically on such machines. The purpose of this paper it to present our plans for resolving this conflict and to explain the current status of their implementation.

Introduction

The IN2P3 Computing Center (CCIN2P3) in Villeurbanne offers a centralized computing facility to French nuclear and high–energy physicists in 17 IN2P3 laboratories distributed across all of France. This service was based previously on an IBM mainframe (ES9000–820) running VM/CMS with HEPVM [1] software. Since the beginning of 1991, user requirements have exceeded the capacity of the mainframe. An important part of this demand was for "CPU–bound" HEP simulation jobs, typically running for ten hours and writing a cartridge (3480) of almost 200 megabytes of data.

The idea of using "cheap" workstations to absorb this load was retained after discussions between IN2P3 physicists and Computing Center engineers. The main points which drove the BASTA [3] (BAtch STAtions) working team were heterogeneity of the cluster, environment transparency, use of existing or standard tools where possible and overall system reliability.

Another important consideration, born from our HEPVM experience, was to compare projects with the CORE project at CERN, so as to maintain as much compatibility as possible for those of our users who might also use CORE (SHIFT [2], CSF) or other such facilities at CERN. The resulting BASTA cluster is currently producing 25,000 hours[1] of CPU time/month.

The success of BASTA encouraged us to consider an integrated VM/UNIX (mainframe/farm) computing environment at the central site of the CCIN2P3, structured in such a way as to make appropriate components exportable to IN2P3 laboratories. The abrupt influx of an entire laboratory (DAPNIA/Saclay) of users, obliging us to upgrade the ES9000 to a model 952, lends weight to this plan.

File access services

User data can be stored on the following storage media (see Fig. 1):

- StorageTek Automated Cartridge System – 5 silos with cartridges model 4480 and 1 silo with model 4490.
- 365 GB of mainframe disk under VM.

[1] An "IN2P3 unit", or UI, = 1 IBM 3090S cpu time

- UNIX "Data server" (IBM RS/6000 98B plus 2 370), with 30 GB of RAID disks and 70 GB of SCSI–2 disks, with direct access to StorageTek controller via BMC channel emulator.

The data from the StorageTek robot can be accessed directly by VM or UNIX applications (using Tape Server) or can be "staged" on the stage disks on both systems. The "Data server" is organised in the following way:

	RAID	SCSI–2
AFS	home, libs	group disks
		scratch + stage

Access to the "stage" and "scratch" areas is via the RFIO package (from the SHIFT facility [2]). The choice of AFS and RFIO has been taken after study of transfer rates between the processors and the "Data server" disks[2]:

	transfer rate in MB/s
NFS write	0.7
NFS read	2.4
AFS write	1.4–2.0
AFS read	1.2–1.4 (disk cache)
AFS read	2.0–2.4 (memory)
RFIO write	3.5–5.0
RFIO read	2.4–3.2

Clusterization – cpu services

The cpu services are provided by the following processors/clusters (see Fig. 1):

- VM mainframe (9,000 UI/month).
- BASTA – the simulation farm (see [3]), currently producing 25,000 UI and devoted to the highly cpu–oriented batch jobs. The BQS batch queueing system has been developed for job submission and control.
- ANASTASIE – the analysis farm, designed for I/O–oriented batch jobs. Enhancements to BQS will take into account jobs I/O requirements.
- BAHIA – the frontal cluster for job preparation and submission to the BASTA and ANASTASIE batch farms (using the BQS client).
- SIOUX – the interactive service cluster, running X11 clients, event servers, etc.

[2]With 1 MB chunks and disk cache of 120 MB

Current Status

The BASTA and BAHIA stations are running in production and ANASTASIE is under beta–test by first users. Tape access thru Data Server will be available in May.

Conclusions

Our immediate goal is to furnish an integrated VM/UNIX (mainframe/farm) computing environment at the central site of the CCIN2P3, structured in such a way as to make appropriate components exportable to IN2P3 laboratories. Our proposed solutions are based on:

- A central UNIX–based data service. Key requirements are direct access to tape cartridges in our StorageTek libraries and access to all data from VM.
- Grouping of stations according to their functions, including the user–access functions.
- Use of AFS for home directories and libraries.

References

[1] N.A. McCubbin
VM/CMS in the HEP community (HEPVM)
Proceedings of the International Conference on Computing in High Energy Physics, Asilomar, 2–6 February 1987, published in Computer Physics Communications, **45**(1987)61–66.

[2] J.–P. Baud, J. Bunn, F. Cane, D. Foster, F. Hemmer, D. Jagel, G. Lee, L. Robertson, B. Segal, A. Trannoy and I. Zacharov
SHIFT – The Scalable Heterogeneous Integrated Facility for HEP Computing
Proceedings of the International Conference on Computing in High Energy Physics '91, Tsukuba, March 11–15, 1991, Universal Academy Press Inc., Tokyo, Japan.

[3] Y. Fouilhé, J. Furet, M. Gaillard, N. Giraud, P. Larrieu, J. O'Neall, A. C. Pardo, W. Wojcik, G. Grosdidier
BASTA, the IN2P3 heterogeneous simulation cluster
Proceedings of the International Conference on Computing in High Energy Physics '92, Annecy, September 21–25, 1992.

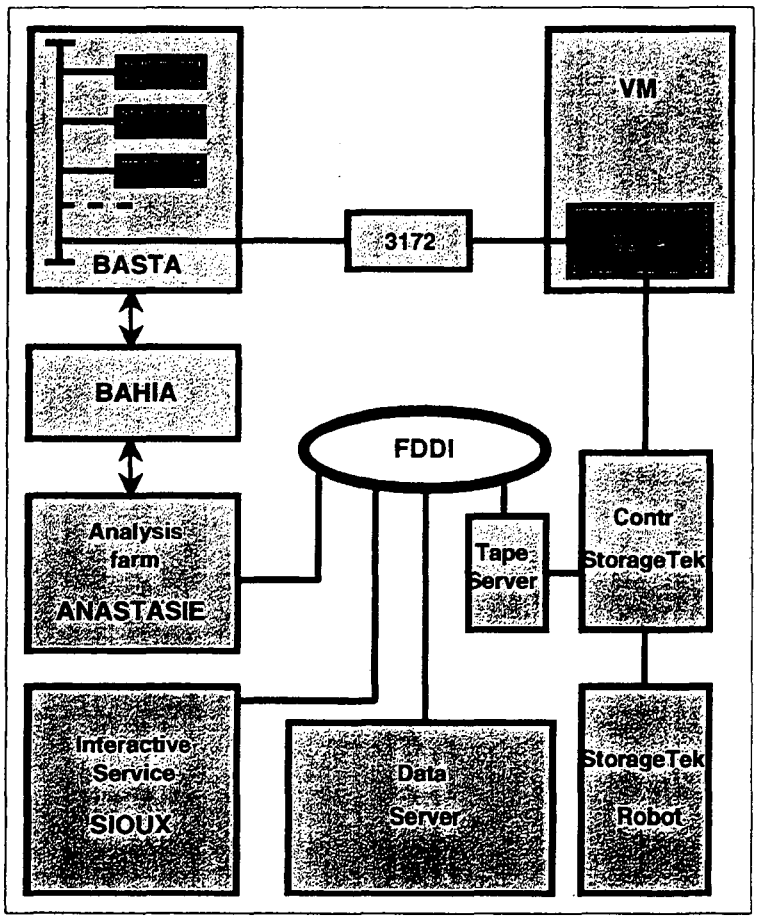


Figure 1: CCIN2P3: Configuration

Parallel I/O measurements on a SparcCenter and a MEIKO CS2

Bernd Panzer-Steindel, CN-GPM
CERN, Geneva, Switzerland

CERN is participating in a European ESPRIT project (GPMIMD2) to investigate the benefits and future prospects of parallel processing to HEP computing.

One of the focal points at CERN is the parallelisation of the Monte-Carlo program from the NA48 experiment. This experiment aims to measure the CP-Violation parameters in the neutral kaon system to a very high precision starting in 1996.

The Monte-Carlo program (NMC) from the NA48 collaboration consists of 102 subroutines with 17500 lines of FORTRAN code. It simulates the decay of neutral kaons into charged / neutral pions in their detector setup. Their main detector is a liquid krypton calorimeter with about 13000 channels. As the shower simulation of photons and pions in this detector take a long time per event (GEANT), the NMC uses a lookup table (showerlibrary) of previously generated showers.

First of all the influence of message passing overhead onto the scalability was investigated. For that purpose the Monte-Carlo was modified to do taskfarming using CHIMP, a message passing model from the university of Edinburgh. One could as well have used PVM, the choice of CHIMP has purely historical reasons. A SOURCE program regularly sends event packets to N WORKER which do the actual decay simulation and finally the results are collected by a SINK program. For a given number of total events the event packet size was varied from very few events (high message passing overhead) to very large event numbers (low messsage passing overhead). This game was played on several cluster of workstations (SUN ,HP, SGI) and on a SGI Challenge multiprozessor machine. Simulation of an event took about 30 ms on a Sparc CPU and the communication (send-receive) with CHIMP on ethernet took about 20 ms (only determined by protocol overhead).

The result was that scalability up to 14 processors can be reached if the message passing overhead is below 2-3 percent.

The actual I/O tests were performed on the following two platforms :

1. shared memory SparcCenter

   8 CPU, 50 MHz, 2 Mbyte 2nd level cache, 512 Mbyte main memory, SOLARIS 2.3

   14 Gbyte of diskspace in different configurations

2. distributed memory CS2 from MEIKO

   8 nodes, per node 40 MHz CPU, 1 Mbyte 2nd level cache, 32 Mbyte main memory, 1 Gbyte disk, SOLARIS 2.1

The NMC needs per event random access to a large database (2 Gbytes in this test). The limiting factor therefore is of course the seek time of the disks. Thus to the CPU time of 26 ms (SparcCenter) per event the wait time for the disk has to be

added to get the real time per event. In this case it was about 70 ms with 4 effective random accesses to the database (11 ms per access = seek time).

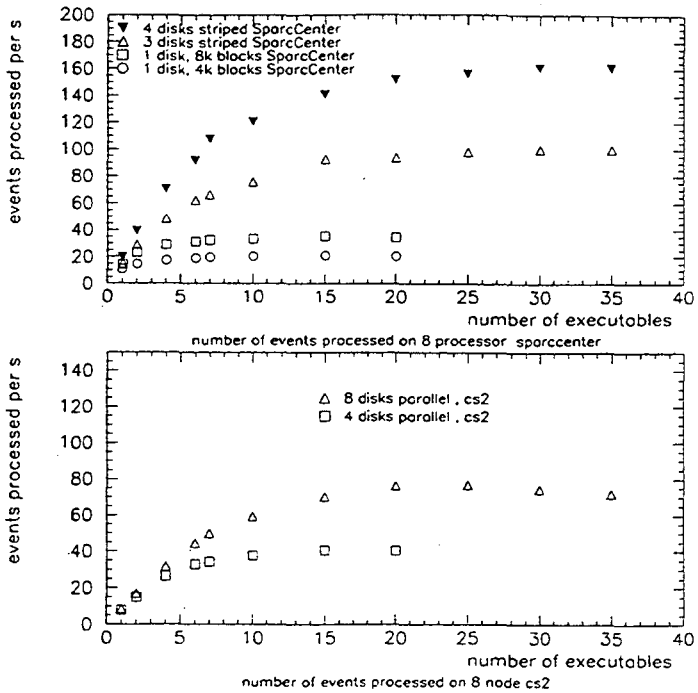
The measurement of I/O speed for sequential writing a 2 Gbyte file (8 Kbyte blocks)to the disk arrangements were : 2.9 Mbytes/s (max. 5 Mbytes/s) for a 4 disk parallel filesystem on the cs2 and 3.9 Mbytes/s for 4 disks striped (2 SCSI channels) on the SparcCenter.

For the I/O performance test the sequential version of the NMC was used. Just N executables were run in parallel accessing the same showerlibrary, with each of the jobs starting with a different random number seed. On the CS2 the jobs were always equally distributed over 7 out of the 8 available nodes. The showerlibrary itself was placed on disk using several different disk arrangements on the SparcCenter and the CS2 :

1. SparcCenter
   (a) 1 disk
   (b) 3 disks, striped with a 16K blocking factor, 1 fast SCSI-2 channel
   (c) 4 disks, striped with a 16K blocking factor, 2 fast and wide SCSI-2 channels
2. CS2
   (a) using the parallel filesystem with 4 disks, striped with a 16K blocking factor, each disk on a separate node
   (b) using the parallel filesystem with 8 disks, striped with a 16K blocking factor, each disk on a separate node

Figure 1 shows the results of the tests.

With no surprise it was realized that an 8 CPU shared memory machine performs better than an 8 node distributed memory machine. The advantage of distributed memory machines will only show up at higher node numbers. Still scalability of the random access I/O application was seen on both machines.



Session 3
Networks and Interconnections

A PACKET-SWITCHED NETWORK FOR DATA READOUT FROM THE LHC INNER DETECTOR. *

Joar Martin Østby, Oddvar Søråsen
Department of Informatics, University of Oslo,
P.O.Box 1080, Blindern, N-0316 OSLO, NORWAY,
Internet: joar@ifi.uio.no,
Phone: + 47 22 85 24 52, Fax: + 47 22 85 24 01.

Abstract

This paper presents a network which can be used for data fusion in the planned LHC (Large Hadron Collider) ATLAS experiment at CERN. The network named SWIPP (SWitched Interconnection of Parallel Processors), is built of 16 × 16 star switches and protocol engines connected by pairs of optical fibers. Each channel is designed to carry up to 1 Gbps in each direction. The paper will give a brief introduction to SWIPP principles and ASIC implementation. Some of the advantages offered by the network will be mentioned.

INTRODUCTION

In the planned LHC ATLAS experiment at CERN[1] particle collisions ('events') will generate data at an estimated rate up to several hundred TBytes/s. Through three selection (trigger) levels uninteresting data will be thrown away while the remaining will be passed to the next level.[1-3]

An interconnection structure used for massive data fusion could be based on SWIPP. The network offers high capacity and flexible scalable interconnection for heterogenous nodes.

*This work is supported by The Royal Norwegian Council for Scientific and Industrial Research (NTNF), projects IT0111.21609 and STP Microelectronics.

[1]Conceil Européenne pour la Recherche Nucléaire. The European Organization for Nuclear Research, Geneva.

THE SWIPP NETWORK ELEMENTS.

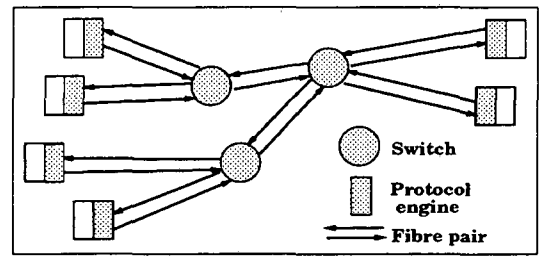


Figure 1. The SWIPP hardware elements.

The SWIPP hardware consists of protocol engines and switches interconnected by pairs of optical fibres or electrical cables. Various topologies can be realized according to the switching requirements

The protocol engines constitute the interface between the network and the environmental nodes being transducers, storage nodes or analyzing computers. The main functions of the protocol engines are to convert between external and internal address and dataformats, to assemble and deassemble packets, thereby

relieving the connected nodes from communication tasks.

The switches have two functions: to establish connections between inputs and outputs through a crossbar unit, and to store data temporarily at the inputs when the output channels are busy.

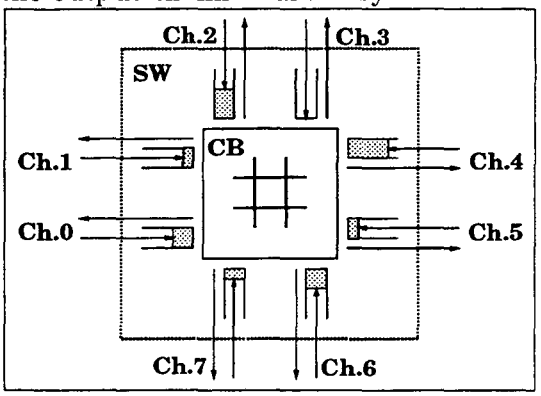


Figure 2. A switch unit.

The network elements are connected by pairs of fibers transmitting in opposite directions. Thus full duplex connections up to 1 GHz are supported. The fibres also facilitate connections to noisy regions.

The packet format

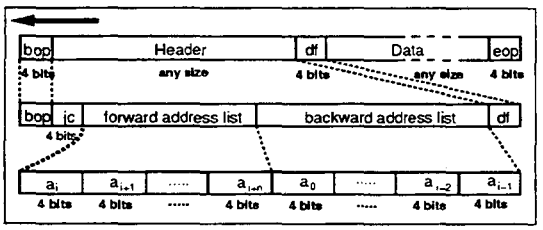


Figure 3. SWIPP's packetformat.

Data are packed into packets each having a **header** field which contains routing information for the switches and a **data** field of variable length.

The header and data fields are enclosed and separated by three unique control symbols: *(bop) beginning-of-packet*, *(df) data-follows* and *(eop) end-of-packet.*

"Source routing" address format

The header field contains two address fields: the forward address list (*fal*) and the backward address list (*bal*). At the source node the forward address list will contain a description of the entire path all the way to the destination ("Source-routing"). The backward address list will initially be empty.

As the packet advances through the network the forward address list will shrink so that it always gives a description of the path from the current position to the destination. Simultaneously the backward address list will increase and give the return path. Thus at the destination the backward address list can be put in opposite order to create the address list back to the source node.

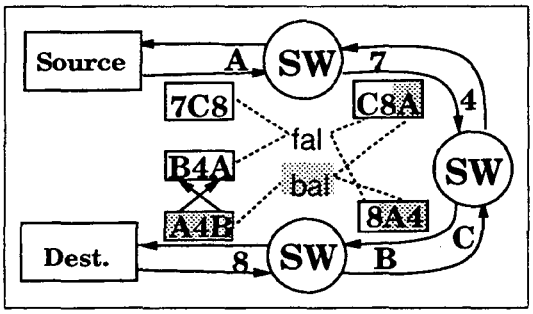


Figure 4: Example of source routing.

Worm-hole

Packet sections are forwarded as soon as they arrive ("worm-hole"-routing). [4] The destination address is decoded and the data switched "on-the-fly". This will give a shorter transmission delay than "store-and-forward" where the entire packet is buffered before it is forwarded.

Only if contention occurs the data will have to be buffered. No address tables need to be stored in the switches, reducing the hardware complexity.

Flow control and input buffers.

All channels have a FIFO buffer at the receiving end. If the amount of data in a buffer passes an 'almost full' limit, a flow control signal will be transmitted upstream to the previous buffer in the transmission path. This buffer will stop sending data and start temporary storage.

With this scheme the remaining part of the buffer when the limit has been passed, has to be large enough to store the incoming data until the data stream changes according to the transmitted signal. Thus, if the reaction time of the flow control is short, a switch may have small buffers.

A packet may be divided into sections stored in many switches. On the other hand one buffer may store a number of packets. The buffer size should be selected from queueing estimates built on simulations or analytic analyses.

Elastic buffers

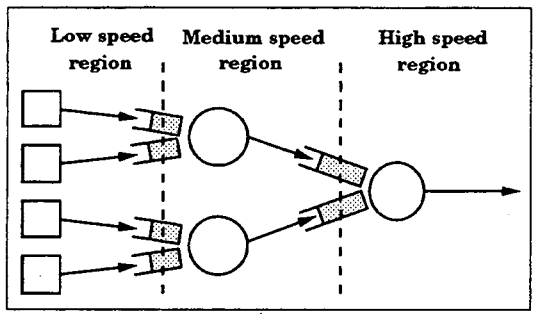


Figure 5. Clock speed regions.

The input and output part of each buffer operate on different clocks. This allowes regions of the network to have lower clockspeeds than other parts. This may be desirable in areas where low power consumption is an aim.

REALIZATION

Two prototypes (one CMOS and one ECL) of a crossbar circuits (4×4) have been designed and are now being tested. They operate at clock rates up to 100 MHz giving a total of 1 Gbit pr. channel.[5][6]

Also an input (asynchronous) FIFO has been designed. It operates at clockrates up to 100 MHz giving a total of 900Mbit pr. sec. pr. channel. The difference in clock rate over a buffer can be up to 10 times.[7]

ADVANTAGES FOR LHC READOUT.

Some primary SWIPP features are beneficial in the LHC data fusion: The variable packet size adapts well to different event sizes. Various kinds of nodes can be connected, LHC inner detectors can use a reduced version of the protocol engine and use their optimal speed for minimum power consumption. Both test and monitoring traffic can be mixed with data using a simple protocol. Bursty trafic can be smooted without loosing data.

CONCLUSION

This paper has given a brief impression of a network consisting of star switches, protocol engines and optical fibers which may be suitable for massive readout and for collection of data from an LHC inner detector. The concept allows different buffer sizes and clock speeds in different regions. Buffer dimensions and clock rates can be selected to reduce queueing, latency, and power consumption.

REFERENCES

1. CERN, "LHC - Large Hadron Col-

lider," CERN Publications, European Laboratory for Particle Physics, Switzerland, June, 1990

2. CERN, ECFA, "Towards the LHC Experimental Programme - General Meeting on LHC Physics & Detectors", Evian-les-Bains, France, March 1992.

3. Ellis, N. et al., "First-Level Trigger Systems for LHC Experiments", **Technical Report CERN/DRDC 92-17**, CERN, March 1992.

4. Ni, L.M. and McKinley, P.K., "A Survey of Wormhole Routing Techniques in Direct Networks", IEEE Computer, February 1993, pp. 62 - 76.

5. Moe, M.,"Konstruksjon av crossbar-svitsj i ECL for multiprosessor nettverk", Master thesis (in Norwegian), Dep of Informatics, Univ of Oslo, Aug 1993.

6. Kvarme, J.E.,"Konstruksjon av crossbar-svitsj i høyhastighets CMOS for multiprosess nettverk", Master thesis (in Norwegian), Dep of Informatics, Univ of Oslo, Nov 1993.

7. Røine, P.T.,"Asynchronous FIFO Buffer for Multicomputer Applications", Master thesis, Dep of Informatics, Univ of Oslo, Jun 1994.

The ZEUS message-passing system*

Jaroslaw Milewski[1)2)] *milewski@vxdesy.desy.de*
Christopher Youngman[1)] *youngman@vxdesy.desy.de*
Andrzej Kotanski[1)3)] *kotanski@vxdesy.desy.de*

[1)] Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.
[2)] II Institute of Experimental Physics, Hamburg University, Germany.
[3)] Department of Physics, Jagellonian University, Cracow, Poland.

Abstract

The ZEUS message-passing system (ZMP) is a general-purpose communication package designed primarily for synchronisation, control and monitoring of processes in real-time environment. The highlights of ZMP are: ease of programming and installation, the dynamic name service, fully distributed communication, portability of code and data.

INTRODUCTION

A message-passing system, in our sense, refers to network communication software serving primarily for **synchronisation, control** and **monitoring** of processes in real-time environment. More generally, the message-passing system enables **on-line data transfer** of any kind between distributed real-time applications. The examples of messages passed over such a communication system are:

- error and status messages,
- commands and acknowledgements of their execution,
- queries and responses to them,
- data tokens of arbitrary size.

In order to reach a satisfactory level of flexibility and ease of use, the message-passing system should meet, at least, the following criteria:

- It must be available on **different computing platforms,** and the applications developed on different machines must be easily portable from one platform to another.
- The applications running on different machines may cooperate easily only if the message-passing system supports **platform-indepentent representation of data.**
- A **name service** which allows for symbolic identification of cooperating processes is *essential.* Applications need not be permanently attached to definite hosts or network addresses.

*The full specification of ZMP [1] is available from the authors at any of the given e-mail addresses.

- Message passing should be **fully distributed.** A real-time environment cannot tolerate a single point of failure like that of the central message router.

So far, ZEUS online applications [2] have relied on OSP [3], ICT [4], ZIP [5] and direct TCP, UDP and DECnet solutions. None of these meet all the required criteria; some are no longer supported and others are not scalable due to design limitations or platform dependencies.

Another seriously considered option was the well-known Sun RPC standard [6]. However, RPC is strongly biased towards C, and, more importantly, it provides no general name service.

The search for uniformity and maintability has ultimately led the ZEUS Central Data Acquisition group to develop its own communication facility, ZMP. The initial aim for ZMP is to replace all the communication systems currently used in the ZEUS online environment, thus providing a **uniform networking platform** for all the ZEUS online applications.

However, implemetation of ZMP is more ambitious. An important design issue was to provide non-experienced users with a tool for creating efficient and portable network applications in a simple and natural way. Due to its flexibility and ease of use, ZMP could be successfully used also by other experiments in High Energy Physics, and probably outside the HEP community as well.

ZMP FEATURES

A ZMP agent is any process using ZMP as the network communication system.

Every agent may use one or more **network interfaces.** The network address of an interface, and therefore one of the available agent addresses, consists of the IP address of the host the agent is running on and of the port number the agent interface is bound to, i.e. *agent_address = (host_address, port_number).*

Naming

ZMP provides a **dynamic name service** for mapping symbolic agent names to agent addresses. 'Dynamic' means that the Name Server keeps track of all the *active* ZMP applications, i.e. those which are *currently* running.

Named vs. anonymous agents

There exist two classes of agents with regard to naming. Any agent may either register itself under a public name or remain anonymous.

Named agents may be freely accessed by any other agents (both named and anonymous). They play the role of 'known network applications' or 'known network services'. Agent names must be *unique* and it is the applications which choose the name they want to bear. Consequently, most network applications running as named ZMP agents exist in *one instance only* at a given time.

Typical candidates for known ZMP agents in the ZEUS online environment are:

- the central ZEUS online tasks like the First, Second and Third Level Trigger, central Run Control and Slow Control Manager,
- Run Control and Slow Control systems of individual ZEUS components,
- various histogram producers,

- other sources of information like the status of the interlock system (BBL3), beam information from the HERA accelerator, etc.

Anonymous agents may act only as clients of known services. In contrast to named agents, there are usually *many instances* of such a program running at a time. The candidates for anonymous ZMP agents in the ZEUS online environment are various display and analyse programs connecting to public data sources.

Mirror agents

As previously noted, named agents usually run as single instances of a given program.

On the other hand, there exist situations when more than one instance of a known network service is desirable. For example, there may exist several servers of a given kind offering exactly the same service, or for the sake of fault tolerance one may arrange a set of identical programs doubling each other's job.

Note that in each case all the instances of the named agent are not distinguishable with regard to their behaviour, i.e. for the possible clients it does not matter which particular instance will be picked up for communication. Therefore, all the instances may (and should) bear the same ZMP name.

The identical instances of a network application running under a common name are called **mirror agents.**

Domains

An agent always specifies to which **ZMP domain** it wishes to belong. The domain is principally a unit of protection which logically separates the message traffic into different application areas. A domain is a *dynamic* entity, which exists 'logically' only as long as at least one named agent belonging to it is running. Any named agent may propose a new domain name.

ZMP functions

The following ZMP functions are available for application programs:

ZMP_open - open a ZMP interface,
ZMP_close - close a ZMP interface,
ZMP_map - get the address of a named agent,
ZMP_xmap - get all the mirror addresses of a named agent,
ZMP_send - standard send,
ZMP_receive - standard receive,
ZMP_qsend - queued send,
ZMP_qreceive - queued receive.

With the **standard send** operation the process waits until the message gets delivered or communication times out. The **standard receive** operation enables the process to wait for a message for a specified period of time. With the **queued send** operation the process only initialises sending; a system signal is raised when sending completes. With the **queued receive** operation the process requests a system signal to be raised when a message arrives.

The standard send and receive operations are sufficient for the majority of applications. The queued versions are intended for more complicated tasks. The same holds for the extended, *xmap,* version of the map function.

It can be seen from the above list of functions that ZMP promotes message passing on the *peer-to-peer* basis. The client-server model of communication may be implemented above the ZMP

level by message exchange in both directions.

IMPLEMENTATION

The network transport

The TCP/IP protocol suite is the only one supported on all the required ZEUS platforms, so it is was the only sensible choice for the implementation of the ZMP transport layer.

TCP as a connection-oriented protocol is not suitable for implementing message-passing systems. The ZMP protocol is therefore based on UDP [7][9, Ch. 11] with added-on *reliable message delivery.*

ZMP may also be used for connecting processes which run on the same host. The overhead of such communication is comparable with the usage of UNIX pipes or VMS mailboxes. Consequently, a ZMP application may treat all its communication partners in a *uniform way;* it does not matter where they are physically located.

Message format

ZMP messages are **data structures** which may contain integers, floating point numbers and strings. All the data sent over the network should have a standard network representation; the anticipeted format is XDR, the External Data Representation [8][6, Ch. 2].

In the present release of ZMP the user has to convert data *manually* from host-specific to network format and vice versa. It is planned to augment ZMP with a tool which will *automate* this task by generating appropriate buffer structures and data conversion functions from representation-independent message descriptions.

There is no a priori limit on ZMP message size. Therefore, ZMP may be used not only as a message-passing system in the sense defined in the Introduction, but also as a vehicle for *more intensive data transfer* between cooperating processes. The maximum (but easily reachable) ZMP throughput seen at the user level is **half of the underlying network media bandwidth.**

The Name Server

Logically, there exists one **Name Server** for the whole ZMP system. It is queried by the agents only when the actual address of a named agent is needed. Apart from that, the message transfer between agents is *fully distributed.*

Internally, the ZMP Name Server program (ZMS) runs in several concurrent instances which maintain the integrity of the distributed name database. This provides for a very high level of availability of the name service.

The ZMP Name Server is accessible from any host on the Internet.

Supported platforms

ZMP is available on **VMS, UNIX-like** (ULTRIX, IRIX, OSF-1), and **OS-9** system platforms.

The most popular programming languages of the ZEUS experiment are **C** and **FORTRAN.** Therefore, ZMP provides bindings for these two.

ZMP installation

ZMP is essentially a library of callable functions layered upon the standard socket library. It is therefore *very easy to*

install. Only the ZMP header file (to have your ZMP application compiled) and the ZMP library (to have it linked), are required. No system privileges, no demons, etc.

Of course, the ZMP Name Server also must be running, but this installation can be done just once by the local network administrator.

IMPACT ON ZEUS ONLINE

We plan to move the whole ZEUS online environment to ZMP during the next year.

All the ZEUS online information of 'general interest' will be covered by ZMP-based data servers. The ones available at the time of writing are: the Global First Level Trigger (GFLT), the Global Second Level Trigger (GSLT) and the HERA Luminosity (LUMI) public data servers.

Another already available ZMP application is the 'sound server' in the ZEUS Control Room, through which the voice signals and pre-recorded announcements are emitted.

The entire ZEUS Slow Control system will be moved to ZMP by November 1994 and the Run Control system by April 1995.

All the ZEUS Expert System [10] interfaces to the data sources will be based on ZMP. The connection to the ZEUS central Slow Control Manager is already operational.

ACKNOWLEDGEMENTS

Tim Short and Thomas Voss participated in testing of the Beta Release of ZMP and wrote many of the ZMP applications existing insofar.

REFERENCES

1. Kotanski, A., Milewski, J. and Youngman, C., *Specification of ZMP, the ZEUS Message-Passing System (Release 1),* DESY ZEUS/CDAQ group, 1994, 38 pp.
2. Youngman, C., *The ZEUS Data Acquisition System,* Proc. of Computing in High Energy Phisics '92, Geneva, 1992, pp. 145-150.
3. Jones, R., Mornacchi, G. and Russell, R., *OSP User's Guide, Version 3.0,* CERN DD/OC Group, 1990.
4. Deffendini, A., Vande Vyvre, P. and Vascotto, A., *The MODEL State Manager User Manual (Appendix C: Inter-process control transactions), Version 2.0,* CERN DD/OC Group, 1990.
5. Park, I., *The Interprocess Communication in ZEUS Central Data Acquisition System (ZIP),* ZEUS Note 89-124, DESY, 1989.
6. Corbin, J. R., *The Art of Distributed Applications. Programming Techniques for Remote Procedure Calls,* The Sun Technical Reference Library, Springer 1991.
7. Internet RFC 768, *User Datagram Protocol.*
8. Internet RFC 1014, *XDR: External Data Representation Standard.*
9. Comer, D. E., *Internetworking with TCP/IP, Vol. I; Principles, Protocols and Architecture,* Prentice-Hall 1991.
10. Behrens, U., Flasinski, M., Hagge, L. and Ohrenberg, K., *ZEX - an Expert System for ZEUS,* Proc. 8th Conf. on Real-Time Computer App. in Nuclear, Particle and Plasma Physics, Vancouver, 1993, pp. 168-173.

USING HIGH PERFORMANCE INTERCONNECTS IN A DISTRIBUTED COMPUTING AND MASS STORAGE ENVIRONMENT

Michael Ernst
Deutsches Elektronen Synchrotron, Notkestraße 85, D-22603 Hamburg

Abstract

Detector Collaborations of the HERA Experiments typically involve more than 500 physicists from a few dozen institutes. These physicists require access to large amounts of data in a fully transparent manner. Important issues include Distributed Mass Storage Management Systems in a Distributed and Heterogeneous Computing Environment. At the very center of a distributed system, including tens of CPUs and network attached mass storage peripherals are the communication links. We are witnessing an integration of computing and communication technology with the 'network' becoming the computer. This contribution reports on a centrally operated computing facility for the HERA Experiments at DESY, including Symmetric Multiprocessor Machines (84 Processors), presently more than 400 GByte of magnetic disk and 40 TB of automated tape storage, tied together by a HIPPI 'network'. Focussing on the High Performance Interconnect technology, details will be provided about the HIPPI based 'Backplane' configured around a 20 Gigabit/s Multi Media Router and the performance and efficiency of the related computer interfaces.

Introduction

The DESY Computing and Networking Devision is developing a Distributed Computing and Mass Storage System to meet the high-end data storage and data-access requirements for High Energy Physics Experiments currently performed at DESY. Current computational resources include 7 SGI Symmetric Multiprocessor computers that have a total of 84 150 MHz CPUs and more than 2500 workstations. A hierarchical network infrastructure based on HIPPI at the top end, FDDI in the medium range and Ethernet at the low end interconnects Mass Storage elements and computational machines.

The following article briefly describes current and planned interconnect technologies used in a forth-generation data-storage model [1].

Data-storage Services

Currently, a spectrum of data-storage

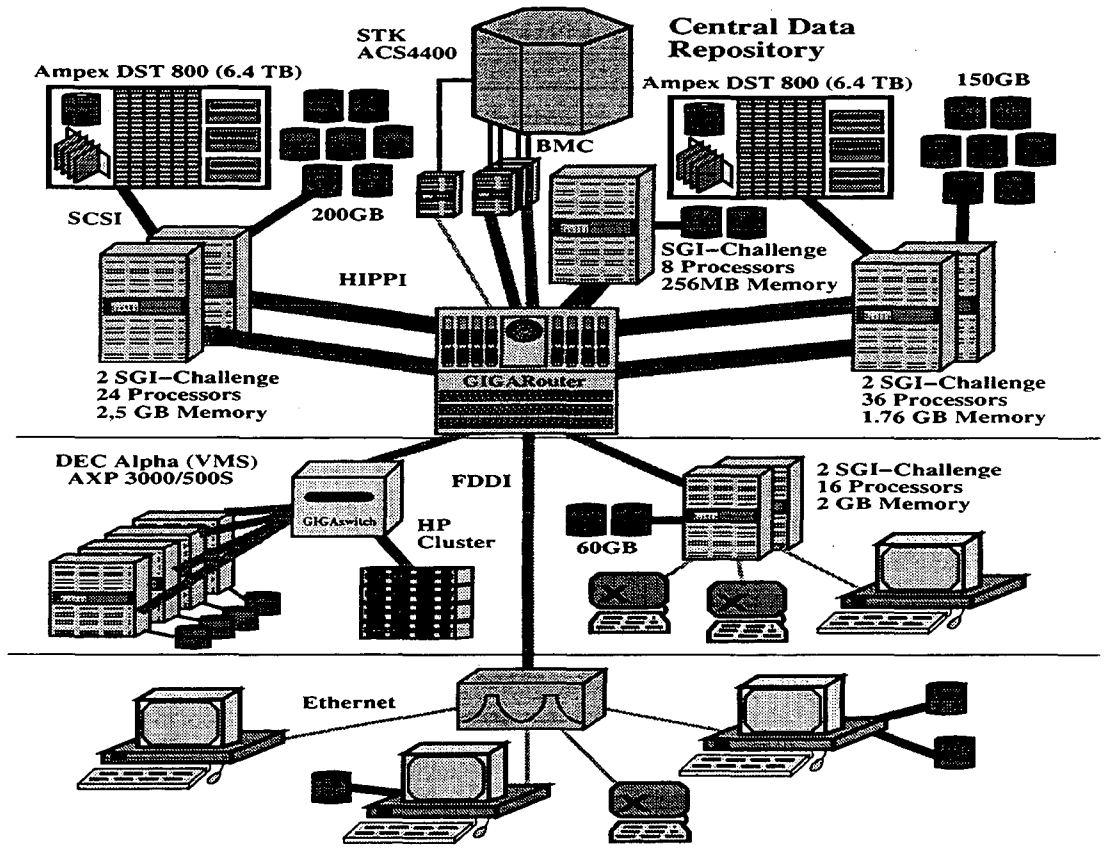


Figure 1. Distributed Computing and Mass Storage Model at DESY

services is used at DESY. Local disk storage is used on supercomputers and workstations. Network File System (NFS) servers that provide mountable file systems having the appearance of local storage are used for workstations. However, there was no common solution providing archival and backup storage for local storage and backup storage for local storage and NFS servers. Projects are underway to upgrade data-storage services and introduce a Hierarchical Storage Management system. DESY has aquired a commercially available generic Management System [2] that is able to meet the demanding storage and access requirements for files that may be as large as hundreds of megabytes, and stored on a growing variety of different storage technologies.

A fourth-generation data-storage model

Traditional network storage systems use a front-end computer that provides network connectivity for storage devices along with storage management, device management, and data transfer capabilities. This approach can result in a solid, more or less flexible system, however, depending on the front-end's architecture, transferring data through it can make the data-storage system very expensive and limit the performance. To meet high-end data storage and data handling requirements following a traditional scheme, a large mainframe must be used for the front-end.

An alternate method is to directly attach storage devices to the network and transfer data directly between storage devices and client machines. Higher data transfer rates and reduced hardware costs are realized by this method, which allows more powerful data-storage systems to be implemented.

The recent availability of HIPPI network technology and HIPPI-attached storage devices has generated widespread interest in a new generation of data-storage systems. The term *forth-generation* has been given to this type of data-storage system in which data is transferred directly between storage devices and client machines, workstation-class machines are used for control and management. Disk arrays and very high-performance tape systems based on DD2-technology are commercially available using a highly specialized controller to get them attached to the HIPPI network. However, the drawback of that solution is not only the very high cost, but also the lack of freedom for an individual integration into an overall system. Today very high-performance workstations, coming at moderate cost, start to appear on the market, including HIPPI interface options tightly coupled to the machines internal high-speed resource-interconnect. The advantage of using them is the possibility to implement the relevant components belonging to a distributed mass storage system, taking advantage of a vendor-optimized communication protocol stack and/or a communication API, respectively.

The HIPPI standard has reasonably matured, HIPPI switching components are commercially available, necessary to allow making a channel based architecture to form a network.

Data Transfer in a HIPPI Network

Reliable, high-speed data transfer in a HIPPI network is important for any high-performance mass storage servers and the computational clients. A data transfer protocol for all kinds of network media, including HIPPI, FDDI and Ethernet, must provide routing, flow control, reliable data delivery, identification and peer-to-peer connectivity. Without significant performance degradation, the protocol must be supported by the client computers and the storage systems.

DESY's implementation uses the de facto standard TCP/IP protocol suite for network data transfers because it is flexible and cost-effective today. The use of a peer-to-peer protocol allows a network node to exchange data with all other network nodes, not just with a particular type of node. It comes with a very well known API allowing to implement the application easily and without any kernel modifications.

At the hardware level, the HIPPI standard provides a Framing Protocol (FP) for HIPPI packets [3]. The HIPPI FP consists of a header that contains an identification of the upper-level protocol (ULP-ID) being used, a D1 field that contains upper-level protocol information, and a D2 field that contains packet data. Link Encapsulation (LE) and IPI-3 are defined upper-level protocols.

The IPI-3 protocol has been implemented for HIPPI-attached storage devices, but IPI-3 is not a peer-to-peer network protocol. IPI-3 is a master/slave protocol, where the storage device is the slave, and the client computer is the mas-

ter. As a result, IPI-3 cannot be used for data transfers between two masters or between two slaves.

LE [4] provides IEEE routing information and is the foundation for TCP/IP. The TCP/IP protocol meets all of our requirements described above. Though TCP/IP support for HIPPI-attached storage devices is not available, we believe in a better approach by choosing whatever mass storage system is required and making it available as a network resource by using an appropriate and cost-effective workstation in between.

Also very importantly, we got to make sure that we don't get tight up to a single network technology. Both different media and new emerging network standards have to be integrated smoothly as soon as they are considered necessary, cost-effective and available. It is not obvious to us that HIPPI will be the ultimate choice for an even foreseeable future. Having chosen it today has to be understood as a pragmatic solution. The point is that we can introduce new media or phase out outdated technology without being forced to rewrite parts, or even worse, the whole communication protocol.

Implementation

We understand that all different kinds of network interfaces for the machines we intend to use are commercially available and well supported by the manufacturers. Pratically, we are looking for HIPPI interfaces for the SGI Multiprocessor computers and FDDI for most the rest. Now, the question is, how to allow packets to traverse different media fully transparently ? The answer until recently was to use a powerful computer being equipped with both media interfaces and running the TCP/IP protocol suite, acting as a router. Though the full functionality can be achieved, however, the efficiency both concerning CPU utilization on the 'router' and the network throughput is by far not satisfactory. Specialized hardware boxes for Ethernet/FDDI are available from a great number of manufacturers, but is not the case for HIPPI/FDDI today. NetStar is the only vendor I know of who is making exactly what we need. Not limiting their product both on the media side and concerning supported communication protocols, they offer or will offer in the foreseeable future besides HIPPI and FDDI, ATM and probably Fiber Channel interface cards. Having a look at the protocols, NetStar not only offers *frame-switching* but also *IP routing*. The latter is extremely important when packets have to traverse differnt media. As an example, our STK Controllers are network attached using an IBM RS/6000 workstation with a Block Multiplexer Channel (BMC), connecting the STK Controller, and an FDDI interface connecting it to the network. This is giving us a perfect performance match concerning both sides, the data path to the STK is limited by the channel speed, running at a maximum of 4,5 MByte/s which fits into the throughput range of FDDI ($<$ 10 MByte/s).

NetStar's GigaRouter design is a distributed control, busless architecture in which mixed-media, multiple-port modules are cabled to a central crosspoint switch. Crosspoint switch capacities are matched to the number of 16-port modules in a system, and serve 16, 32, or 64 ports.

A port card is media-specific, that

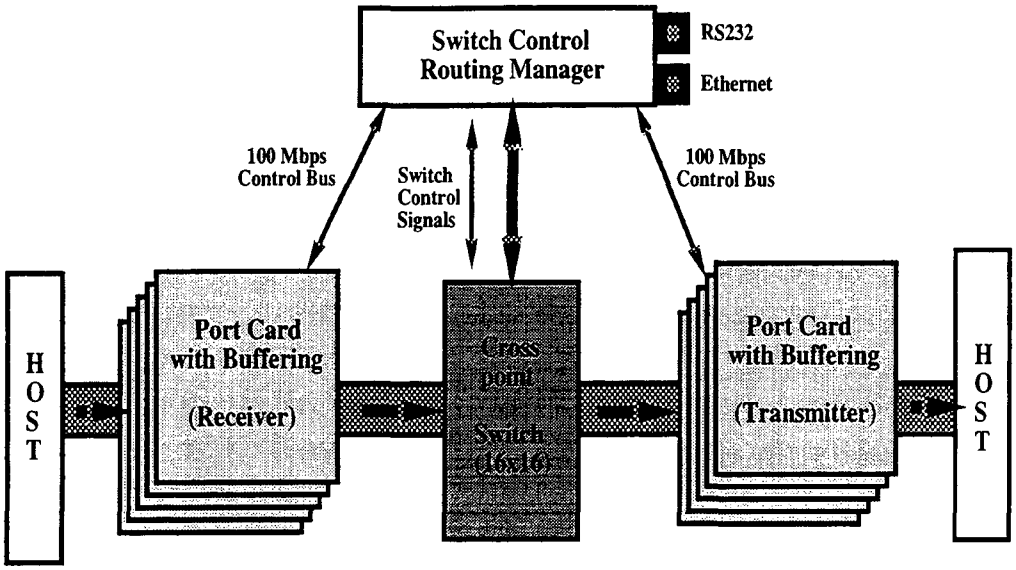


Figure 2. Data and Communications Architecture of the GigaRouter

is, it supports HIPPI, FDDI, or other *standard* communications connections. A port card has an input port (receive channel) and an output port (transmit channel), both governed by an intelligent processor.

The GigaRouter is designed in a *non-blocking* architecture. Our 16-node switch provides 256 different and separate data paths accross the switch. Network congestion is not caused by the switch, but can occur when a large number of input ports are requesting access to a single destination port.

The raw bit rate of the serial links is one gigabit/second. The data encoding scheme used on the links transfer 40 encoded bits for every 32 bits of unencoded data.

Individual port cards maintain their own routing table, perform lookups, and autonomously handle passing of datagrams to other port cards for export, without the intervention of the main routing processor. When possible, overhead on the transfer of packets is minimized by the use of a cut through process. As soon as it is received, the packet frame (raw mode) or IP header is read, the routing decision is made, and the first buffers of data are transmitted accross the router before the rest of the packet may even be received.

Performance

Figure 3 clearly illustrates the tremendous performance gain when compared to the proprietary Ultranet. Especially with relatively small buffers, like the NFS style 8 kByte buffersize, more than 18 MByte/s can be achieved. At the same size Ultranet is running at 3 MByte/s and approximately 1/3 CPU-utilization.

Summary

Data-storage and data-access requirements in the demanding area of HEP-computing can be met in a cost-effective manner by using a fourth-generation data-storage system in which storage devices are directly connected to a Multi-

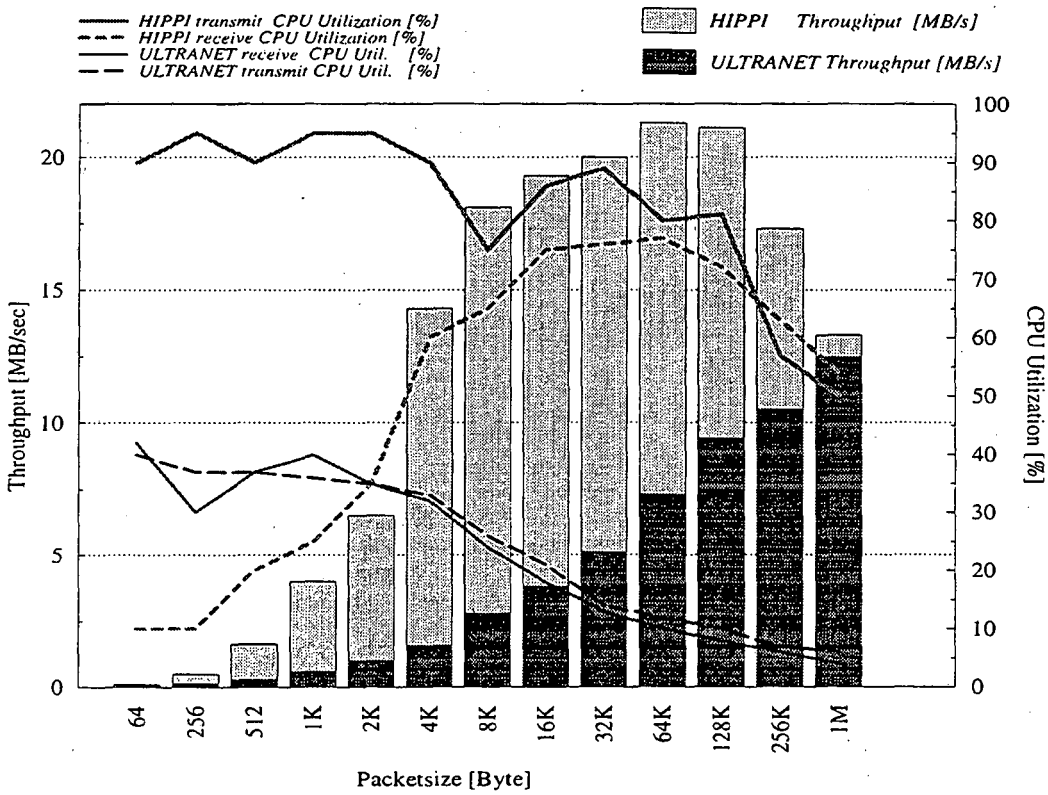


Figure 3. HIPPI versus Ultranet performance

media hierarchical network infrastructure and are managed by workstation-class machines. By sharing conceptual ideas, experience and work, DESY has established a very close collaboration with vendors in order to define and build a scalable and fexible computing and mass storage environment that perfectly matches with HEP demands today and for the medium range future.

Acknowledgements

The author gratefully acknowledges the help of many colleagues from the DESY computing and networking devision taking heavily part in many fruitful discussions which helped developing and improving this challenging project.

REFERENCES

1. M. Gasthuber, "Distributed Mass Storage and Management Systems at DESY," these procedings
2. Lachman Technology Inc; "Open Storage Manager (OSM)," Company Product Manuals
3. "High-Performance Parallel Interface Framing Protocol Specification (HIPPI-FP)," ANSI X3.210-199X
4. "High-Performance Parallel Interface Encapsulation of ISO 8802-2 (IEEE Std 802.2) Logical Link Protocol Data Units (HIPPI-LE)," ANSI X3.218-199X

"SHIFT-BETEL" : A (VERY) DISTRIBUTED MAINFRAME

B. Segal, O. Martin, F. Hassine, F. Hemmer, J-M. Jouanigot
CERN, 1211 Geneva 23, Switzerland
J. Bernier, S. Ohlsson
IN2P3, Centre de Calcul, Villeurbanne, France

1 Introduction

Over the last four years, CERN has progressively converted its central batch production facilities from classic mainframe platforms (Cray XMP, IBM ESA, Vax 9000) to distributed RISC based facilities, which have now attained a very large size. Both a CPU-intensive system ("CSF", the Central Simulation Facility [1]) and an I/O-intensive system ("SHIFT", the Scaleable Heterogeneous Integrated FaciliTy [2]) have been developed, plus a distributed data management subsystem allowing seamless access to CERN's central tape store and to large amounts of economical disk space. The full system is known as "CORE", the Centrally Operated Risc Environment; at the time of writing CORE comprises around 2000 CERN Units of Computing (about 8000 MIPs) and over a TeraByte of online disk space. This distributed system is connected using standard networking technologies (IP protocols over Ethernet, FDDI and UltraNet), but which until quite recently were only implemented at sufficiently high speed in the Local Area.

In the second half of 1993, the first international trial took place involving a Wide Area network technology with comparable performance (ATM over a link running at 34 Mbit/s). This trial was called "BETEL" - Broadband Exchange over Trans-European Links [3] - and was funded by the EEC on the initiative of the European Parliament. BETEL had the goal of making convincing demonstrations of "real" applications over ATM in a very short timescale - one year from start to finish. BETEL collaborators were France Telecom/Expertel and the Telecom PTT Switzerland (Operators), Alcatel CIT and Cisco (Manufacturers) and four research institutes (Users).

The installed BETEL platform interconnected FDDI rings on the four BETEL user sites: CERN (Geneva, Switzerland), EPFL (Lausanne, Switzerland), IN2P3 (Lyon, France) and EUROCOM (Nice, France) in a fully meshed manner, via Cisco routers and ATM Terminal Adapters located at each site, 34 Mb/s optical fiber circuits, and an ATM Cross-connect located in France Telecom premises in Lyon.

One pair of BETEL users (EPFL and EUROCOM) worked on a multimedia "teleteaching" application while the two High Energy Physics institutes (CERN and IN2P3) demonstrated two HEP applications. The first was "PAW", the interactive Physics Analysis Workstation [4] system which had already been well tested in both LAN and WAN environments. The other was SHIFT, whose extension to a WAN environment is the subject of this paper.

2 SHIFT-BETEL

The SHIFT portion of the BETEL demonstration took place between CERN, in Geneva, and the IN2P3 computing center in Lyon (200 kilometers away). A representative SHIFT CPU Server, Disk Server and Tape Server was set up at IN2P3 and connected to the IN2P3 central tape store. The IN2P3 CPU and Disk Servers were both implemented using standard SHIFT software in an SGI Power Series 4D/340 machine, while the Tape Server was implemented as a SHIFT-compatible interface to the existing IN2P3 "zstage" facility.

The major goal was to show that when the CERN and IN2P3 systems were interconnected by the ATM line, all SHIFT disk and tape resources at both sites (including data from either the CERN or the IN2P3 tape robots) became transparently accessible to batch jobs running at either site, with performance comparable to that in the Local Area.

3 Problems Encountered

3.1 Basic TCP/IP Limitations

There are particular problems related to the use of TCP on high-speed and long-delay links ("Long Fat Networks") which have been analyzed by Van Jacobson and others [5]. Improved "sliding window" and "slow start" algorithms have been developed for which the "Round Trip Time" (RTT) of the path between the source and the destination hosts is measured and monitored so that the optimum size of the actual TCP "window" (i.e. the number of unacknowledged bytes) can be determined and used. Many other refinements have also been proposed like fast retransmit, fast recovery, as well as an increase in the maximum TCP window size of 64KBytes. Unfortunately not all host TCP implementations support these yet.

In BETEL, we did not really enter the "Long Fat Network" regime as the product of line-speed (34Mb/s) times round-trip delay time (about 2msec for a 400km round trip) corresponded to a window occupancy of about 10 KBytes. Also, the maximum line speed was not fully attained due to equipment limitations. However, several factors could rapidly degrade this situation in future cases. The actual RTT is increased in practice due to store-and-forward delays (in communications adapters as well as in routers); it was already observed to be 2-3 times the theoretical value in our very simple case. One known reason for the extra delays measured on BETEL was the fact that the ATM Terminal Adapters were functioning in "store and forward" rather than in "pipe-lining" mode, i.e. with no overlap between the receiving of an IP packet and its segmentation into ATM cells.

It is feared that this problem may become much worse on public infrastructures, e.g. that of the European ATM pilot, where the network topology is not directly derived from the needs of the users (e.g. the "distance" between Geneva and Lyon will become 1500 km instead of 200 km because of detours via Zurich and Paris) and where many IP gateways may be involved, increasing the number of hops and therefore the real RTT.

A paradox of "sliding window" algorithms is that, in order to minimize the round-trip time and therefore the window size, short TCP segments (and MTU sizes) are preferable to long TCP segments which take longer to transmit. However, it is also well known that the resulting load on the end systems of generating many small segments may be unacceptably high, hence the need for a trade off between long segments which are acknowledged slowly and small segments which put too much load on the end systems (and on the routers).

In the end, it seems that the MTU size of approximately 4 KBytes used on both FDDI and SMDS is a suitable compromise. A serious problem, however, is that standard TCP/IP implementations use a default IP packet size of 576 bytes when the source and destination hosts are not located on the same IP network, which is the normal case on wide area networks. This leads to a serious performance loss. Fortunately, some implementations allow subnets (or all nets) to be defined as "locals" and/or to specify the MTU size for each destination, which is an inconvenient but required way to bypass this problem until "dynamic MTU path discovery" mechanisms are more widely implemented. This was quite important in BETEL.

3.2 Lower Level Problems

On September 1st 1993, the 34 Mb/s link between CERN and IN2P3 became operational. Using T3 adapters and running only HDLC between the Cisco's across this link, initial TCP memory to memory transfers from an HP 735 on FDDI at CERN to another HP 735 on FDDI at IN2P3 were measured at 3.5 MB/s using the maximum window size (55 KBytes) and the default MTU of 576 bytes. This exceeded 80 percent of the available 34 Mb/s bitrate. The first SHIFT tape transfers were then made from IN2P3 to CERN and obtained about 600 KB/s, similar to LAN (Ethernet) performance.

On September 23rd, the SMDS/ATM Terminal Adapters were ready, and the Cisco's were reconfigured - initially using pure SMDS. The TCP memory to memory transfer rates between HP 735's dropped surprisingly to 800 KB/s, and to 400 KB/s between two SGI machines. This was finally traced to packet losses in the Cisco boxes because of the high packet rate produced while the MTU was set at 576 bytes. The problem diagnosis was not eased by the fact that Cisco updated their SNMP variables at only 10 second intervals when using autonomous switching. The problem could be worked around (using SGI

machines) by resetting the MTU to the standard FDDI value of 4352 bytes, giving a TCP transfer rate of 3.7 MB/s for pure SMDS. Cisco did fix the packet loss problem shortly afterwards. However HP did not succeed in providing an MTU resize option (supported by both SGI and IBM) during the experiment.

The next step was to introduce ATM under SMDS on the line. This reduced the TCP transfer rate from 3.7 MB/s to 1.25 MB/s, and was found to be due to limitations of the Alcatel SMDS/ATM Terminal Adapters: these adapters were never made to run faster than 11 Mbit/s during the experiment.

Finally extensive tests were carried out with a mixture of HP, IBM and SGI machines and tape, disk and memory transfers with both simple applications and with the SHIFT software. These showed up an SGI FDDI bug, causing asymmetric transmit/receive performance, and an IBM bug causing FDDI hangs. When these were fixed, and the MTU, TCP window and socket buffering parameters were tuned, performance of simple applications like FTP and TTCP were acceptable (about 800 KB/s to 1 MB/s over SMDS/ATM), but that of the SHIFT Remote File I/O protocol "RFIO" was less than expected (about 300 KB/s).

3.3 Higher Level Problems

3.3.1 SHIFT-RFIO

To study the RFIO performance problem, an FDDI analyzer was put on the FDDI ring at CERN and quite long delays (about 135 msec) were observed between bursts of packet activity (themselves lasting about 66 ms). It was discovered that these delays were due to the fact that the SHIFT transfer protocol RFIO emulates open/read/write/close functions remotely, in the style of an RPC. It sends a command and/or a block of data over the network and then waits for the other end to confirm that the transaction was completed. The standard TCP algorithms used to optimize data transfers then become inefficient as the TCP layer does not receive a continuous flow of data. A new RFIO algorithm was developed using a read-ahead mechanism, to avoid waiting for the other end to confirm. Implementation and tests of this new algorithm showed the predicted performance improvement (to between 800 KB/s and 1 MB/s for disk to memory transfers).

3.3.2 User Mapping

A solution had to be found for the "user registration problem", typically encountered when interworking between user communities. A user, defined as the triple (username, UID, GID), must be identified correctly at CERN and IN2P3 for both client and server functions. User mapping functions were incorporated, together with a secure token passing scheme to authorize tape and disk space requests on behalf of a given user. Incidentally, this approach was also found to be perfectly suited for the Local Area extension of SHIFT functionality to VMS and other non-Unix systems such as OS-9.

3.3.3 Lack of NFS

For various reasons, including security and protocol efficiency, CERN does not permit NFS mounting of its file systems from external sites. This limited the operational flexibility of SHIFT-BETEL: typical batch jobs require access to the users' home directories and often to other standard directories or libraries, particularly at compile and link time. Without NFS, this could only be provided by duplicating the required directories as required. In future, AFS will solve this sort of problem.

4 Conclusions and Future Plans

In spite of the short time available for system development and testing (4 months between September and December 1993), full production running of SHIFT-BETEL was achieved before the end of 1993 and continued while the ATM link remained available (until February 1994).

Plans for further development of the system are proceeding, as we have been promised a new ATM linkup to begin in mid-1994. The major system improvements will be to RFIO efficiency: separate TCP control and data streams will be used to permit more continuous data flow, and RFIO write performance will be enhanced by use of a "write-behind" technique. It is also hoped to have successfully incorporated AFS into SHIFT by the time the next BETEL trial takes place.

REFERENCES

1. The Central Simulation Facility at CERN, E. Jagel, CERN CN Report, February 1992.
2. Mainframe Services from Gigabit-Networked Workstations, J-P. Baud et al., CERN CN Report 93/8.
3. Broadband Exchange over Trans-European Links,
   O.H. Martin, Proc. SMDS Conference, Amsterdam, November 1993 (CommEd).
4. PAW, Complete Reference, R. Brun et al., CERN Program Library Long Writeup Q121.
5. TCP Extensions for High Performance, RFC-1323, V. Jacobson, R. Braden, D. Borman, May 1992.

Networking with China

R. L. A. Cottrell, Charles Granieri

Stanford Linear Accelerator Center, Stanford University, California, U.S.A.

Lan Fan, Rongsheng Xu

Institute of High Energy Physics, Beijing, China

Yukio Karita

KEK National Laboratory for High Energy Physics, Tsukuba, Japan

Abstract

This paper presents the history and current status of computer networking between IHEP in Beijing, China and the rest of the world, starting with no links at the beginning of 1987 thru X.25 public networks and dial up links, to the installing, in March 1993, of one of the first dedicated 64 kbps satellite computer links between China and the outside world. In May 1994, IHEP became the first Chinese institution to have a fully operational world-wide Internet connection. Experience with this dedicated link between SLAC and IHEP will be presented together with future plans to add a land line between KEK and IHEP and to extend the links within China.

Early History

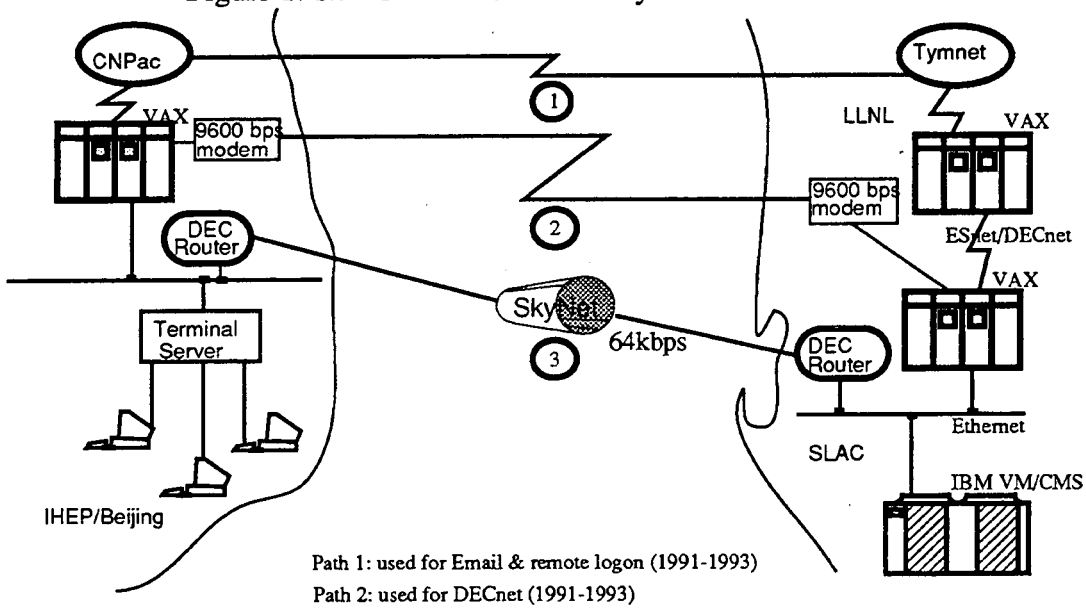
Computer networking at the Institute of High Energy Physics (IHEP) in Beijing, China was started in 1987. One computer, named BEPC2 (VAX785) in IHEP, was remotely linked with CERN via the PSDN first, then in May of 1990 changed to CNPac (X.25 with a speed of 4.8 kbps) provided by the Ministry of Telecommunication of China. CERN worked with IHEP to provide PSI and EAN (X.400) Email connectivity to IHEP via CERN. Scientists in IHEP then could send and receive E-mail with the outside world via the VXCERN node at CERN, Geneva, Switzerland.

The decision to set up extensive network communications between IHEP and SLAC (Stanford Linear Accelerator Center in California, United States) was made in 1991 by the Beijing Electron-Positron Spectrometer (BES) collaboration group. At this time, IHEP had one phone line capable of making international calls. This line was in the International Relations department and was mainly used for FAXes.

In March 1991, 3 direct dial phone lines with international access were installed at the IHEP computer center. They were used with Telebit T2500 modems to provide logon connectivity between SLAC and IHEP (Figure 1, Path 2). The effective transmission rate was 700-900bps. Cost was $1/min if initiated from SLAC and $3/min if initiated from IHEP. For 1991 the typical bill was about $4K/month. After some experimenting the use of these dial up links was extended to supporting asynchronous dynamic DECnet between SLAC & IHEP, and IHEP was moved into the SLAC HEP DECnet area. The effective transmission rate was 400bps. The line was noisy and difficult to use, often connections could not be made when no international lines were available, and there were frequent disconnects in mid-session.

Concurrently, X.25 connectivity between SLAC & IHEP was arranged via LLNL/Tymnet and CNPac (Figure 1, Path 1). The connection cost was about $100/hour. It was used for Email, which cost about $1/mail item, and for remote logon. By September 1991 it was being used to send about 5-10 mail items/day, and between 5 and 20 mins for remote logon. This connection was satisfactory for Email, but very sluggish for interactive use with typically a 1.5 sec response time and transfer rates of a few hundred bits/sec. The costs of these services during 1991 was about $3-5K/month for the US end and $7K/month for the Chinese end.

Figure 1: SLAC-IHEP Connectivity Paths 1991-1994



Path 1: used for Email & remote logon (1991-1993)
Path 2: used for DECnet (1991-1993)
Path 3: used for Email, remote logon & DECnet (1993-1994)

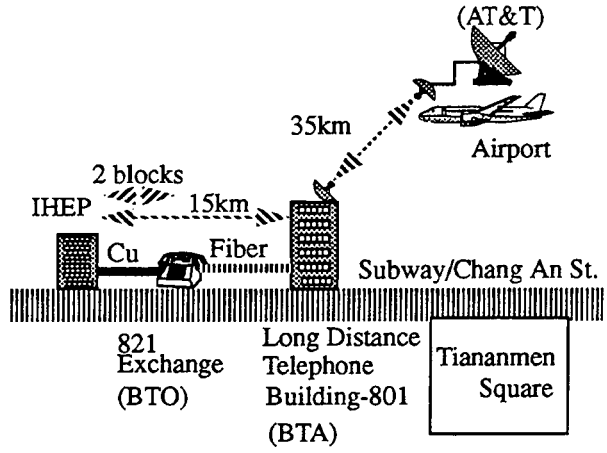
SLAC-IHEP Satellite Link

Starting in 1991 SLAC & IHEP started pursuing the feasibility of installing a dedicated link. Interest in the DoE HEP community was solicited and expressed by several High Energy Physics, Nuclear Physics, and Basic Energy Science communities, and the National Science Foundation (NSF) expressed interest. Various options were investigated including services via KEK, Cable & Wireless and AT&T SkyNet.

Approval for installation of a dedicated 64kbps link, to be funded in the US by Department of Energy (DoE), SLAC and the Super Conducting Collider with SLAC taking the lead, was received from DoE in late 1991. Based on costs and schedules proposed, an AT&T SkyNet satellite link was chosen (Figure 1, Path 3). The contract with AT&T was signed in January 1992. The US side cost was about $5.5K installation and $5K/month. IHEP pays a similar amount in Chinese currency for the Beijing side of the link.

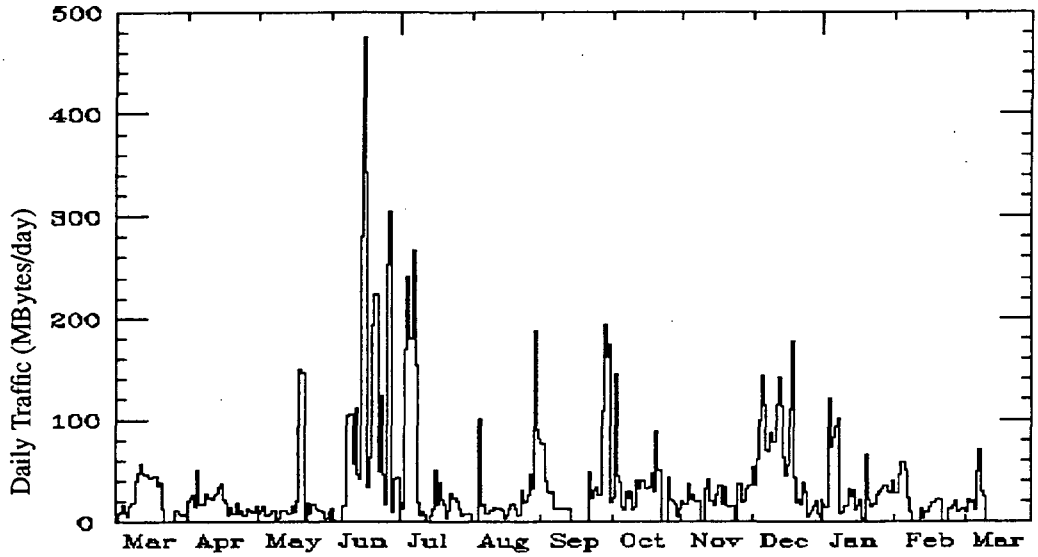
Considerable problems were encountered getting the link from the Beijing airport to IHEP between January 1992 and March 1993. The paths involved in this link are shown in Figure 2. The original plan was to use 64kbps microwave modems between IHEP and the local 821 phone exchange. However, these were not able to deliver satisfactory service. Instead it was decided to try existing copper links. Then there were problems interfacing between the copper and optical fibers running from the local phone exchange to the satellite earth station located at the Beijing airport. Early in 1993 the Beijing telephone companies succeeded in getting acceptable error rates and the link was handed over to IHEP on March 1st 1993 at 15:19hr PST. Seconds later a monitor program showed node 44.393 (the SLAC DEC-router) adjacent to IHEP.

Figure 2: Link from IHEP to Beijing Airport



The file copy rate was measured at ~42kbps (VAXstation 2000 - VAX 780) which was 10 times better than before, the echo time was better than 1 second, and the error rate was ~ 2.3E-7. The daily utilization is seen in Figure 3. It has peaked at nearly 100% of capacity over a 24 hour period during a conference held at IHEP. Typically about 40MBytes/day (see Figure 3) are transferred. Typical hourly peaks are 6MBytes/hour. There

Figure 3: Daily Utilization of the SLAC-IHEP Satellite Link (1993-1994)



are about 1 to 2 unscheduled outages of several hours/ month in addition to some scheduled outages twice per year when the sun is directly in line with the satellite. AT&T switched to a new satellite in Jan 1994 and the connection from IHEP to the local 821 exchange was changed to DDN (Digital Data Network) in March 1994 in order to increase reliability of the link.

The link is currently mainly used for:

1. Transferring samples of the detector data each day to check remotely that the detector is functioning correctly. When the experiment is taking data, this consists of one file/day of up to 200 Mbytes (the contents of a 3480 cartridge) taking about 6 hours to transfer;
2. About 2500 Email items are exchanged daily with over 400 sites in over 21 countries. About 10% of the Email is via DECnet, the remainder is gatewayed at SLAC between China and the Internet via PMDF/MX mailers. Typically there are about 20 Email messages/day destined for China that need manual intervention due to incorrect addresses being supplied;
3. Usenet for local SLAC and IHEP news groups;
4. Coordinating collaboration activities, including travel, scheduling conference and regular phone calls.
5. Code management to keep BES software developed at multiple sites in the US and at IHEP in synchronization;
6. Copying files, remote login and real-time communication (e.g. VMS PHONE).

Connecting to the Internet

Since the 64kbps IHEP-SLAC line was established, several institutions in China have expressed interest in connecting with IHEP and getting international access to the Internet. The Institute of Scientific and Technical Information of China, the Computer Network Center of Academica Sinica and the Institute of Modern Physics of Academica Sinica have connected with IHEP by leased CNPAC or CHINAPAC (X.25) lines. In addition about 300 top-scientists and professors or Ph.Ds in China are allowed to access IHEP computers by dial-up phone line to communicate with collaborators around the world. Universities, like Tsinghua and Beijing in Beijing and some very important universities outside Beijing, including Fudan University in Shanghai and Nanjing University are considering leasing lines to IHEP. All these sites are especially interested in getting access to the Internet via the IHEP-SLAC line. Thus IHEP could soon become the first Chinese gateway to the Internet.

We have been working on adding IP connectivity to the existing DECnet connectivity between SLAC and IHEP. To do this we have replaced the DEC routers with Cisco routers at SLAC and IHEP. A Management Plan for Energy Sciences network (ESnet) Network Access to the People's Republic of China* was created by a task force of the ESnet Steering Committee, and as a result export licenses for Cisco routers to go to IHEP were ob-

* "Management Plan For ESnet Network Access To People's Republic of China," The China Connectivity Task Force of the ESSC, 7/19/93

tained from the U.S. Department of Commerce (DoC) at the end of 1993, and the routers were received in Beijing in February 1994. The routers were installed in March replacing the DECnet routers, and DECnet connectivity was opened up to the worldwide HEPnet DECnet. At this time management of the link was turned over to ESnet. A visit to IHEP by U.S. Congressman George Brown in December 1993 also increased interest in the link at higher U.S. government levels. A meeting was held in January 1994, with attendees from universities and computing institutes in Beijing, to discuss and come up with a recommendation for domain names in China. It was decided that IHEP nodes will appear as node.ihep.ac.cn. IHEP has been assigned two class C Internet address blocks by the Internet's regional registrar (APNIC). Agreement was obtained to allow the Internet to carry Chinese traffic contingent on some conditions being met. One major condition was to make a wide-area electronic mail announcement that the Internet will be carrying Chinese IP traffic. This announcement was made to ESnet sites on April 18, 1994 and stated that ESnet would begin to carry Chinese IP traffic on April 25, 1994. Opening of the link to full Internet connectivity happened on May 17, 1994 when a connection was established via ESnet from SLAC to FIX-West, which is the West Coast interconnection point for all of the major TCP/IP networks. IHEP thus became the first Chinese institution to have a fully operational world-wide networking connection.

Currently the cost of the U.S. end of the link is about $50K/year. This is borne by SLAC and the DoE High Energy and Nuclear Physics Office. This will be transferred to ESnet. Plans to increase the speed of the link to 128 kbps are currently deferred until the traffic warrants it. This is expected to happen shortly after the link is opened up to the Internet.

There are also needs for a link to China from KEK in Japan to support many Chinese collaborators in TRISTAN experiments at KEK and for KEK-IHEP collaborations in R&D in constructing accelerators. There already exists a KEK-ESnet (U.S.) 192kbps terrestrial link, so work is in progress between KEK and IHEP to create a new IHEP-ESnet path via a 64kbps terrestrial link from KEK to IHEP. This would replace or the existing IHEP-SLAC satellite link. An optical cable between Japan and China was completed in December 1993 from Miyazaki in Japan to Shanghai in China. Domestic optical links within China including a link from Shanghai to Beijing was scheduled to be completed at the same time frame but is being delayed. A proposal was made to put in a temporary satellite link between KEK and IHEP until the land line cable is completed. This has been done and the cut-over from the SLAC-IHEP to the KEK-IHEP link awaits a convenient time for the BES experiment at IHEP. At the completion of the link, multiprotocol routers will be located at IHEP and KEK, and the link will be configured to carry both DECnet and IP traffic.

China is a huge country with enormous potential and ambitious plans to open up electronic communications. The HEP community should be proud to have pioneered one of the first direct electronic links from the outside world into China, and helping to open it up to the global internet community.

Acknowledgments

We would like to acknowledge the help we have received in writing this paper from Ray Cowan, Bill Dunwoodie, Mike Sullenberger and Teresa Downey of SLAC, Xin Hao and Yang Dajian of IHEP, Jim Leighton, Tony Hain, Joe Burrescia and Rebecca Bostwick of ESnet, Mike Kelsey of Caltech, Joe Izen of UT Dallas, and Julie Hennessey-Niland and Robert Luke of AT&T.

This work was supported in part by the Department of Energy Contract DE-AC03-76SF00515. This report is SLAC Pub 6478, published June 1994.

THE RADIO-MSU NETWORK

Hans Frese
Deutsches Elektronen-Synchrotron DESY
Notkestraβe 85 22603 Hamburg, Germany

Sergey F. Berezhnev, Dmitry A. Avdeyev
Nuclear Physics Institute, Moscow State University
119899 Moscow, Russia

Abstract

A combined satellite/microwave network has been set up between three High Energy Physics institutes in the Moscow region and DESY. 2 Mbps microwave links are used for Moscow local loops. The hub is connected to DESY via a 256 Kbps satellite channel .

INTRODUCTION

A combined satellite/microwave network has been set up between three High Energy Physics institutes in the Moscow region and DESY.

Previously, access from these institutes (ITEP - Institute of Theoretical and Experimental Physics, the Lebedev Physics Institute of the Russian Academy of Science, and NPI MSU - the Nuclear Physics Institute of Moscow State University) was only possible through a terrestrial line running at 12 kbps.

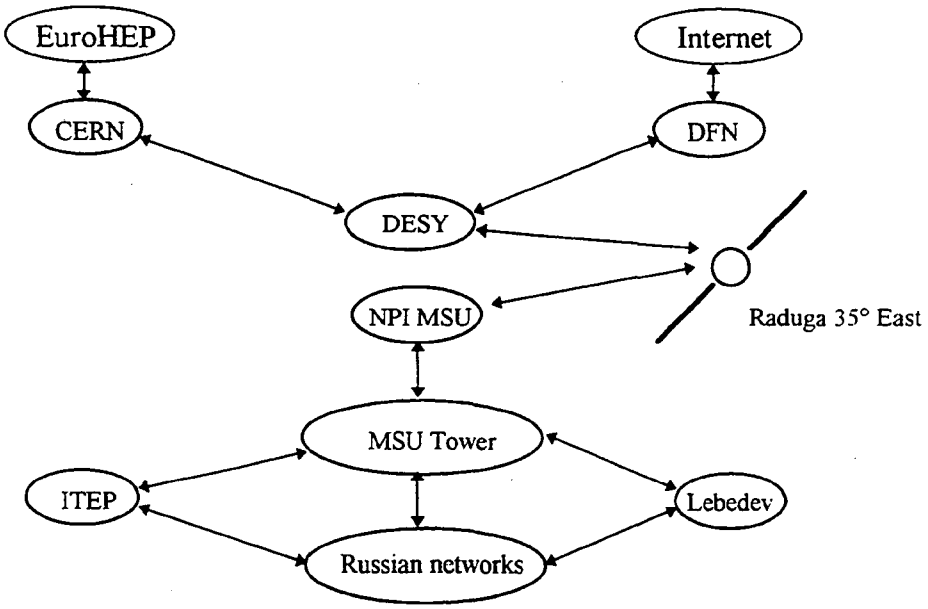


Figure 1. Radio-MSU Network Topology

TOPOLOGY

The Radio-MSU network (Fig. 1) uses a Russian satellite to link up DESY and NPI MSU at 256 kbps. The tower of Moscow State University is used as a radio tower for three microwave links running at 2 Mbps to ITEP, Lebedev, and the Lebedev campus at Troitsk. The end points of the microwave links were chosen in such a way that they act as distribution centers of middle and low speed terrestrial connections. This hierarchical scheme avoids the "last mile problem" since terrestrial connections can be implemented as direct electrical connections within one telephone exchange which permits line speeds up to 38.4 kbps using cheap Russian modems.

At the DESY end, traffic is split between DESY, European HEPnet via CERN, and the rest of the world via DFN (the German Research Network).

EQUIPMENT

All backbones (satellite and microwave) are connected via cisco AGS+ routers (there are 5 of them). Terrestrial lines are linked via Telebit Netblazer routers. The TCP/IP stack is the main network protocol, with RIP, OSPF, and BGP as routing protocols. The network registered domain is Radio-MSU.NET.

Microwave Equipment

Two types of Russian made microwave equipment are used:

- "Radan-MS" digital microwave stations (11 GHz carrier frequency, 2 Mbps data rate, G.703 interface, AMI coding system) for intra-Moscow links. AMI requires an external scrambler to avoid synchronization loss in the presence of long sequences of ones or zeroes.
- "Kompleks-MG" digital microwave stations (11 GHz carrier frequency, 2 Mbps data rate, G.703 interface, HDB-3 coding system) for long distance links.

Satellite Equipment

Except for the US made satellite modems (Fairchild 2900), Russian made equipment is used:

- a "Raduga" geostationary satellite positioned at 35° East operating in C-band with an uplink frequency of 6.2 GHz and a downlink frequency of 3.875 GHz,
- a "Kalinka" ground station with a 3.5 meter dish at DESY, and
- a "Kedr-M" ground station with a 4.8 meter dish on the roof of NPI MSU.

OPERATIONS

The Moscow microwave links became operational on October 25, 1993. Satellite channel tests were started on December 14, 1993. Regular operation of the satellite link commenced on March 25, 1994, after the German certification procedures for the DESY ground station were completed.

The satellite modems use Viterbi encoding which produces corrected bit error rates normally below 10^{-9}. This is an improvement of at least 3 orders of magnitude over the raw bit error rates. The error rates of the digital microwave links are already below 10^{-11} without forward error correction.

Satellite and microwave link quality as well as IP connectivity between selected networks is monitored continuously by a management station at NPI MSU.

APPLICATIONS

Geostationary satellite links exhibit a minimum round trip time of 500

milliseconds. This should be taken into account by all applications which are normally written/configured for terrestrial environments. Touch typing skills are very helpful for interactive work, especially since round trip times show very little variation.

FTP accounts for the third largest number of bytes transferred, right after netnews and mail. It should be noted that the maximum 16 bit TCP window size introduces a throughput limit for FTP transfers independently of the bandwidth: with a round trip time of half a second, a single TCP connection cannot use up more than roughly 1 Mbps.

At the current link speed of 256 kbps, ftp transfer rates of 16..18 Kbytes/s are observed. This is due to hosts using less than the maximum window size.

FUTURE DEVELOPMENT

The project consists of two stages:

1. Improving network connections for HERA collaborators and other HEP institutes.
2. Using stage one as the basis for extending networking to Russian scientists in fields outside High Energy Physics.

Further development will be pursued in three directions:

1. Extending the network in the Moscow region using dedicated links and the Moscow Internet eXchange (MIX), a planned FDDI backbone.
2. Upgrading the speed of the satellite backbone DESY - NPI MSU.
3. Adding 64 kbps satellite connections to the satellite backbone to reach other areas in the former Soviet Union.

ACKNOWLEDGMENTS

This project* was funded by the German Ministry of Research and Technology. DESY and DFN (the German Research Network) were tasked with the implementation, with NPI MSU acting as subcontractor in Russia.

Special thanks go to the administration of Moscow State University for making possible the use of the university tower for the microwave distribution system.

* "Pilotvorhaben zur Verbindung von Wissenschaftseinrichtungen im Grossraum Moskau mit DESY und WiN", TK596-001.

DECnet Routing Transition from PhaseIV to PhaseV/OSI in Japan

F.YUASA, H.HIROSE, S.ICHII, Y.KARITA, T.MIYAJIMA, Y.MORITA and T.NAKAMURA

Network Group, Computing Center, KEK

Abstract

DECnet, one of network protocols used in world-wide HEP community, is under the way of transition to PhaseV/OSI because of lack of address space. This affects to DECnet naming, addressing and routing. We will present the current status of the routing transition in Japan.

Introduction

The growth of DECnet nodes in world-wide HEP/SPAN DECnet is the major driving force behind the transition to DECnet PhaseV/OSI network. HEP/SPAN DECnet network has reached practical limitations of PhaseIV naming, addressing, and routing such as:

- node name is restricted to 6 alphanumeric characters,
- no distributed name service,
- only 63 areas and only 1023 nodes in a area (2 bytes for address space),
- complex network management due to boundless routing domain.

Alternatively, DECnet PhaseV/OSI provides us hierarchical naming scheme, distributed name service, 20 bytes of address space, concept of routing domain, and new routing technique (link state routing). Transition has several aspects, especially, naming changing and routing changing affects to users and managers so much. In order to coordinate world-wide transition, HSDCG (HEP SPAN DECnet Coordination Group) prepared transition plan for naming, addressing, and routing[1][2]. According to the rule decided by HSDCG, in November 1992 distributed name service was open to the world-wide HEP/SPAN and in autumn of 1993 a part of Europe HEP/SPAN DECnet started routing transition.

Status of Japan HEP/SPAN DECnet

We Japan HEP/SPAN DECnet (JP-DECnet) are allocated area40 in PhaseIV network. In this area two networks, HEPnet-J[3] and TISN(Todai International Science Network), are interconnected. HEPnet-J is managed by KEK and links university HEP groups in Japan. On the other hand, TISN links various Japanese institutes and operated by Faculty of Science, University of Tokyo. Each network has an international link to US independently. HEPnet-J has a link to ESnet and TISN has a link to NSI. Now total number of DECnet sites is 44 and that of nodes is over 620 in the area40. Number of nodes increases over 100/year. This year, a new international link from KEK to IHEP (Beijing, China) will be opened and they will join the area40. After IHEP's joining us, lack of address space will be a very serious problem in near future.

We made our transition plan in JP-DECnet committee within the framework of HSDCG's agreement, considering HEPnet-J and TISN's circumstances. For example, in that committee we agreed that we use reverse order of domain-name defined in Internet as the PhaseV hierarchical nodename. As for naming transition, we had already set up the Japan country name-server and some site name-servers[4][5]. Several endnodes started transition and joined name-service managed

by HSDCG. As for address transition, now we are using compatible address, 47:0020:0028 to area40 assigned by HSDCG. Japanese Government is preparing to distribute OSI address to us. After transition will be over, we will be able to use OSI address. In this way, we have been working on naming and addressing transition in past 3 years. In this environment, some endnodes finished transition to PhaseV.

Next target of the transition is routing. Even if all endnodes do not finish transition to PhaseV, we can proceed routing transition as long as we use compatible address.

Difficulties in Routing Transition in Japan

Routing transition has three aspects: L1(intra area) routing transition, L2(inter area) routing transition, and inter-domain setting if necessary. Especially for L1 routing, the transition should be done at once because all L1 routers in one area must use the same routing protocol to keep connectivity. However, in Japan there are several difficulties as follows:

- a lot of sites in one area (HEP and non-HEP sites),
- 2 groups (HEPnet-J and TISN),
- various vendor's routers,
- various software versions on routers,
- many host-based routers,
- no extra area for transition preparation.

In the above circumstances, it is a very hard condition that we change L1 routing at once.

Routing Transition Plan

Fortunately, in HEPnet-J, all dedicated routers used are Cisco's. Also in PhaseV/OSI no host-based routers are allowed . We had already upgraded all Ciscos so as to handle PhaseV. Taking this advantage, we proposed to the JP-DECnet committee 3-stage transition plan:

1) We will set both PhaseIV and PhaseV (CLNS IS-IS) on HEPnet-J links using Cisco's capability, DECnet conversion . By using this capability, routers can handle not only PhaseIV packets but also PhaseV packets without losing connection between endnodes. This is only pre-introduction of L1 PhaseV, however we will start without waiting for TISN site's preparations. We will be able to expand this pre-introduction gradually. This trial will give us rich experience on IS-IS routing, which is very useful for the following stages.

2) After all TISN sites will be ready, we will change all L1 routers to PhaseV at once. In this stage, real L1 PhaseV environment will be established.

3) L2 routing on US links will be changed to PhaseV and inter domain routing will be started between routing domains if necessary.

Actually, we have tested the stage-1 plan in a pilot environment at KEK(Fig.1). In the pilot environment, several ethernet segments are inter-connected by using Cisco routers where we set PhaseIV, PhaseV(IS-IS), and DECnet conversion in both serial and ethernet interfaces at the same time. We used two kinds of Cisco routers whose specifications are shown in Table 1.

Table 1 Routers summary

Hardware	Software	I/O Memory	Memory used
Cisco 3000	9.1(6)	4MB	about 3.6MB
DEC brouter90*	9.14(1)	1MB	about 0.8MB

We got the following results:

* DECbrouter90 is Digital's product and is developed in cooperation with Cisco Systems.

• without DECnet conversion ON, connectivity between PhaseIV EndNode and PhaseV EndSystem was lost,

• connectivity between PhaseIV EndNode and PhaseV EndSystem which are located in a remote segments was guaranteed with DECnet conversion ON,

• it is confirmed by using CLNS PING that OSI packets were received and forwarded through multi-hopped routers ,

• routing information was updated normally.

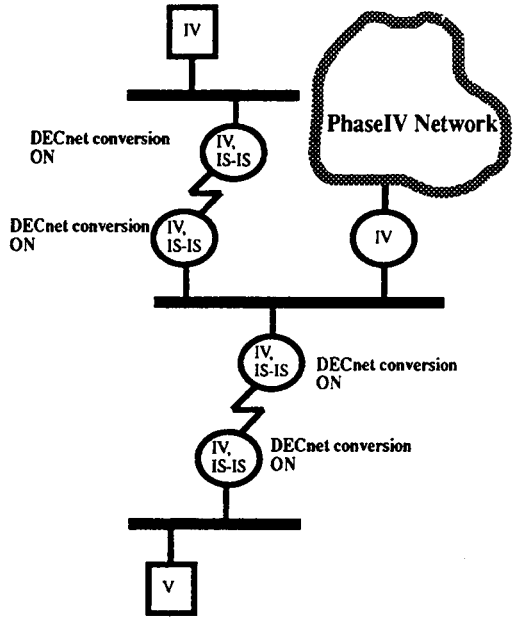


Fig.1 KEK pilot environment

From the experiences we got in the pilot environment, we confirmed that we can apply our plan to JP-DECnet safely. Our staged routing transition plan gives us the experiences on OSI protocol and the possibility of using OSI applications if we hope.

Summary

Specific routing transition plan for Japan HEP/SPAN DECnet is presented in this paper. Also we presented that our routing transition plan was started in a pilot environment at KEK. From the result in the pilot, we confirmed that we could expand PhaseV routing transition incrementally in our HEPnet-J. By using our plan, sites which are ready can go to PhaseV quickly and enjoy advantages of it. On the other hand, sites which are not ready can stay PhaseIV for a while. After all sites in JP-DECnet will be ready to PhaseV, we will change L1 routing to PhaseV simultaneously . Experiences stored at the first stage are very useful to make risk minimum.

This transition plan was approved by the JP-DECnet committee held in this January.

References

[1] W. Van Camp. Minutes of HSDCG Meeting, September 19-20 1991, Bologna, Italy.
[2] P. DeMar and L. Porter. *NASA/NSI-DECnet and DOE/ESNET-DECnet Transition Strategy for Phase V/OSI.*
[3] S.Ichii et al. KEK Proceedings 92-19, Feb. 1992. Proc. of *the Workshop on Distributed Computing and Network*, p. 280, 27-28 Feb. 1992, KEK.
[4] F.Yuasa. *Migration to DECnet Phase V/OSI I* [in Japanese]. KEK Internal 93-7, Oct. 1993.
[5] F.Yuasa et al. KEK Proceedings 92-19, February 1993. Proc. of *the Workshop on Distributed Computing and Network*, p.221, 27-28 Feb. 1992, KEK.

The Effects of X Window HEP Graphics Applications on ESnet*

Farhad A. Abar
Fermi National Accelerator Laboratory, Batavia IL
Illinois Institute of Technology, Chicago IL

William P. Lidinsky
HEP Network Resource Center at Fermi National Accelerator Laboratory, Batavia IL
Illinois Institute of Technology, Chicago IL

Abstract

Wide area networking is the next evolutionary step toward distributed computing. Many applications that were found useful in local area networks are beginning to show their presence in wide area networks (WANs). A question is: Given today's typical WAN infrastructure, what are the effects of the presence of distributed applications in the WANs and what can be done to facilitate full deployment of such services across wide area network? A simulation model for X window distributed graphial applications in high energy physics communities interacting across the DOE Energy Science wide area network (ESnet) was created to examine X service resource requirements and ESnet resource limitations. Through simulation analysis the effects of the incremental introduction of X traffic to ESnet was determined as was the load level at which ESnet became unstable. Proposals for improving ESnet performance by upgrading to T3 links and also by introducing a service-based packet priority scheme at the network layer were also examined.

INTRODUCTION

Typical wide area research and educational networks consist of owned or leased communications lines terminating at routers to form an irregular mesh topology. Given today's technology and leased line costs, the links are frequently T1 lines with bit rates of approximately 1.5 Mbps. Routers are matched to these link bit rates. High traffic links are sometimes of the T3 variety, having bit rates of approximately 45 Mbps. As usage of these networks increases and link costs decrease, more T3 links are being employed. Host computers are seldom attached directly to these WANs. Instead, networking at sites "on" a wide area network have site-wide arrangements of local area networks (LANs) interconnected by either bridges or routers. It is these arrangements of LANs to which the host computers are attached. Site-wide internetworks of LANs are, in turn, attached to the wide area network. The entire communications is referred to as an internetwork.

Examples of wide area networks in operation today are the National Science Foundation Network (NSFnet), the Energy Sciences Network (ESnet), and the NASA Science Internet (NSI). Figures 1 shows the topology of ESnet.

ESnet consists primarily of T1 lines and Cisco™ routers at the sites shown in Figure 1. ESnet was chosen for the research because (1) it is typical of wide area networks, (2) it is a WAN over which much high energy physics data traffic passes in the United States, (3) the network is homogeneous in both its links and routers

* This work was partially supported by the U.S. Department of Energy under Contract No. DE-AC02-76CH03000

thus easing the analysis, and (4) it was readily accessable to the authors.

ESnet is untypical in that it is rather tightly controlled by one organization and is homogeneous in its components (Cisco™ routers and T1 lines).

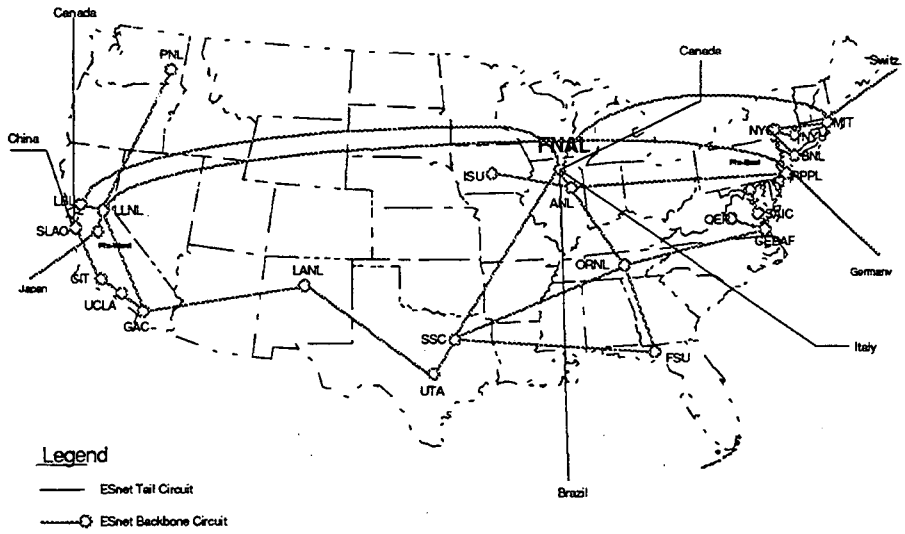


Figure 1: ESnet and International Circuits

PERFORMANCE RESULTS

T1 Links

No Added X Traffic. The entire ESnet network was the subject of this simulation. Each ESnet node was connected to its neighboring node via terrestrial T1 links. There were also firewall routers (AGS+) which attached their corresponding ESnet node to the LANs via Ethernet links. The present ESnet network was simulated based on the established work-load model. Based on this configuration, the present ESnet network was simulated for 110 seconds with sampling starting at 10 seconds to eliminate transient effects. Statistics were collected at 10 seconds interval.

Table 1 presents the average simulation link utilization (with 95% confidance interval) between Fermilab's ESnet node (FNAL) and its neighboring nodes during this run. X window traffic was not added.

Source to Destination	Link Type	Average Link Utilization - [%]
FNAL to firewall router	Ether-net	6.128 ± 0.038
FNAL to LBL	T1	10.048 ± 0.164
FNAL to MIT	T1	7.421 ± 0.294
FNAL to ANL	T1	13.291 ± 0.094
FNAL to SSCL	T1	4.943 ± 0.136

Table 1: ESnet Average Link Utilization - T1 Links, no added X Traffic

Table 2 presents the average simulation link throughput (with 95% confidance interval) between Fermilab's ESnet node (FNAL) and its neighboring nodes during this run.

Source to Destination	Link Type	Average Link Throughput - [pkts/sec]
FNAL to firewall router	Ether-net	397.2 ± 0.910
FNAL to LBL	T1	176.7 ± 0.547
FNAL to MIT	T1	98.3 ± 0.771
FNAL to ANL	T1	96.1 ± 0.646
FNAL to SSCL	T1	59.0 ± 0.634

Table 2: ESnet Average Link Throughput - T1 Links, no added X Traffic

Table 3 presents the average simulation delay (with 95% confidance interval) experienced by a packet traveling between the firewall of Fermilab's ESnet node (FNAL) and the above HEP sites firewall routers.

Source to Destination	Average Packet Delay - [msec]
FNAL to firewall router	0.166 ± 0.001
FNAL to LBL	19.873 ± 0.053
FNAL to MIT	9.307 ± 0.036
FNAL to ANL	1.628 ± 0.013
FNAL to SSCL	9.062 ± 0.045

Table 3: ESnet Average Packet Delay - T1 Links, no added X Traffic

As is shown the average link utilization between Fermilab and other high energy sites is less than 15 percent, indicating that present ESnet infrastructure has met the demand of today's network traffic.

X Traffic Added. The ESnet model was extended to include traffic of selected HEP X applications (PAW, HistoScope, and Explorer). Sessions of X client and server communication were established by adding a set of the application modules representing the X clients of the HEP X applications to the firewall of Fermilab site for each prospective HEP site. Also, for every X client application module, a server module was added to its corresponding ESnet site.

The network load of the above X application set between Fermilab and other HEP sites was incrementally added to the point of network performance degradation. A total of 24 concurrent X sessions (Three X applications (PAW, HistoScope, and Explorer) running between Fermilab and eight other HEP sites) between Fermilab and ANL, LBL, MIT, SSC, PPPL, AMES, NEVIS, and FSU sites predicted an stable condition (i.e., no packet loss or memory exhaustion observed).

Table 4 shows the average simulated delay (with 95% confidance interval) experienced by X packets traveling between the firewall of Fermilab's ESnet node and its neighboring nodes; and the average simulation link utilization (with 95% confidance interval) between Fermilab's ESnet node (FNAL) and its neighboring nodes.

Source to Destination	Ave. Packet Delay - [msec]	Average Link Utilization - [%]
FNAL to firewall router	0.554 ± 0.004	45.853 ± 0.330
FNAL to LBL	56.421 ± 12.053	75.788 ± 1.365
FNAL to MIT	29.700 ± 1.604	70.260 ± 0.990
FNAL to ANL	22.847 ± 1.488	76.064 ± 1.158
FNAL to SSCL	29.489 ± 3.430	68.471 ± 1.508

Table 4: ESnet Ave. Packet Delay - T1 Links + X Traffic

The above tables showed the results of adding distributed high energy physics X applications to the ESnet. Only 24 concurrent X applications could be applied to the net. The bottleneck was the T1 links between FNAL and LBL and between FNAL and ANL with average link utilizations of above 75%.

T3 Links

Ethernet Link to LANs. The ESnet model was modified by replacing its existing T1 links with T3 links and was subjected to HEP X traffic between Fermilab (X clients) and other HEP sites (X server). A simualtion run with stable outcome which includes a total of 90 concurrent X applications (six X applications running between Fermilab and 15 other HEP sites) was established. As expected, Fermilab site experienced packet-drops as a result of above configuration. The bottleneck appeared to be the Ethernet link between FNAL ESnet and its firewall with average utilization of 98% and collision rate of about 300 incidents per second.

T3 Link to LANs. The ESnet model was furthermore modified by replacing the existing Ethernet links to firewalls with T3 links. The maximum level of loading in a stable condition was the simulation run of 180 concurrent X sessions. Table 5 shows the average simulated delay (with 95% confidance interval) experienced by X packets traveling between the firewall of Fermilab's ESnet node and its neighboring nodes; and also the average simulation link utilization (with 95% confidance interval) between Fermilab's ESnet node (FNAL) and its neighboring nodes.

Source to Destination	Ave. Packet Delay - [msec]	Ave. Link Util. - [%]
FNAL to firewall router	0.007	3.711 ± 0.013
Firewall router to FNAL	0.133	64.766 ± 0.185
FNAL to LBL	63.137 ± 9.517	42.527 ± 0.890
FNAL to MIT	10.702 ± 0.152	14.620 ± 0.372
FNAL to ANL	1.297 ± 0.006	17.410 ± 0.052
FNAL to SSCL	10.256 ± 0.140	10.349 ± 0.647

Table 5: ESnet Average Packet Delay - T3 Links + X Traffic with 10^{-6} Error Rate

Lower Error Rate. The above ESnet model was furthermore modified by reducing the average serial link bit error rate from 10^{-6} to 10^{-7}. Based on the above modification, loading of ESnet with HEP X applications was increased from 180 concurrent sessions to 270 concurrent X sessions. Table 6 outlines the average simulation delay (with 95% confidance interval) experienced by X packets traveling between the firewall of Fermilab's ESnet node and designated HEP sites and also the average simulated utilization of the link between the ESnet node at FNAL and its neighboring nodes.

Source to Destination	Ave. Packet Delay - [msec]	Average Link Utilization - [%]
FNAL to firewall router	0.007	5.387 ± 0.014
Firewall router to FNAL	0.183 ± 0.001	95.276 ± 0.196
FNAL to LBL	36.127 ± 9.948	76.818 ± 1.593
FNAL to MIT	11.079 ± 0.130	23.229 ± 3.882
FNAL to ANL	3.000 ± 0.139	25.707 ± 0.115
FNAL to SSCL	11.049 ± 0.179	13.100 ± 0.192

Table 6: ESnet Average Packet Delay - T3 Links + X Traffic With 10^{-7} Error Rate

Routing Priority. Based on NSFnet traffic statistics the following services were considered to have high priority which make up about 38% of the total traffic: Telnet, Domain, Gopher, ICMP, X, Login/who, Talk, Finger, and SNMP.

The remaining services, FTP-data, NNTP, SMTP, and UUCP were classified and low priority which make up about 62% of the total load on the network. The ESnet model with T3 links was used with 38% of the ESnet backbone traffic generated and labeled as high priority class and 62% of the backbone traffic generated and labeled as low priority class. All 180 concurrent X application packets were generated as high priority class.

Table 7 outlines the average simulation delay (with 95% confidance interval) experienced by X packets with high priority class traveling between the firewall of Fermilab's ESnet node and designated HEP sites for 180 concurrent X sessions.

Source to Destination	Ave. Packet Delay - [msec]
FNAL to LBL	48.620 ± 8.227
FNAL to MIT	10.588 ± 0.189
FNAL to ANL	1.302 ± 0.017
FNAL to SSCL	10.226 ± 0.102

Table 7: ESnet Average Packet Delay - T1 links + X Traffic + Priority

CONCLUSIONS

The question posed ealier was: Given today's typical WAN infrastructure, (ESnet as an example) what are the effects of the presence of distributed applications (such as X) in the WANs and what can be done to facilitate full deployment of such services across wide area network.

One strategy for improved performance is resource upgrade. The T1 links in the model were replaced by T3 lines with bandwidth increase from 1.4 to 44.7 Mbps. As presented in previous sections, performance was improved three-fold from 24 concurrent X sessions to 90 concurrent sessions. The bottleneck moved to the Ethernet link between FNAL's ESnet node and its firewall router with average link utilization of 98%. Ethernet connectivity between the ESnet router and its corresponding enterprise firewall router is clearly not the optimum configuration for this type of services.

The model was then modified by replacing the Ethernet links between ESnet nodes and firewall routers with T3 lines. Performance doubled to about 180 concurrent X sessions.

As expected the average delay per packet decreases in all cases. The delay ratios (T1 to T3) is close to the bit rate ratio (28:1) only in the case of X session between FNAL and ANL, the site pair with the shortest distance (i.e., shortest propagation delay). In the other site pairs with long distances between them, the delay ratio is close to three. The effect of increase in bandwidth on average delay decreases as the propagation dominates the term. The effect of T1 to T3 resource upgrade, however, is more profound in terms of the throughput across the backbone, with an increase in eight concurrent X sessions to 180 ones (a 1:23 ratio).

The effect of link quality improvement was also investigated. By improving the link bit error rate from 10e-6 to 10-7 total concurrent X sessions is increased from 180 sessions to 270 ones. An improvement in the link bit error rate reduces the possibility of packet retransmission and consequently increases the link throughput. The effect of link quality improvement on packet throughput is more evident in long distance links (e.g., FNAL to LBL) where packet acknowledgement delay becomes longer.

Another strategy considered for improved performance was service-based priority routing at the network layer. Presently all packets are routed on the first-come first-served basis and since good percentage of packets on the network (63%) are not considered time critical those that are time critical could be given higher priorities and consequently faster responses.

In a simulation run with 180 concurrent X sessions, average packet delays for some of the sessions during the high priority run (e.g., FNAL to LBL: 48.620 msecs) was considerably lower than its counterpart in the no priority scheme (FNAL to LBL: 63.137 msecs). There was a reduction in average delay for high priority packets of about 23%.

The reason for such decrease in delay is due to the following: Packets arriving at the routing service center will be placed in the

appropriate queues based on their priority levels if the routing server is busy. The routing server processes packets in its high priority queue before its medium and low priority ones. Processed packets are then handed to the Network Interface Module (NIM) server for transmission in a first-come first-serve basis. Since high priority packets are serviced by the routing server before the low priority ones, they will be queued up together in groups at the NIM service center without any low priority packet between them. Low priority packets will be placed at the end of the transmission queue.

The effect of priority scheme is more pronounced when:

- Outbound packet traffic is high, such as FNAL to LBL link. As traffic increases so does the possibility of time critical packets waiting behind other packets for transmission,
- average size of low priority packets is high as is the case with FTP and SMTP packets. (Longer packet length for low priority packets requires longer transmission time), and
- transmission link has lower bandwidth and hence it would take longer to transmit low priority packets.

ESnet is in the process of upgrading its links to T3 service. The preceding analysis was for one class of distributed applications between Fermilab and 15 other HEP sites. There still many other services such as packetized video, AFS, etc. that could impose major load on the network. Upgrading to T3 lines in the ESnet backbone seems to be a good idea for very near future. However, concurrent application of distributed services over the network seems to indicate that a T3 link might not be a sufficient long term resource.

REFERENCES

[1]NERSC. ESNET IP ROUTING, Version 3.2. UID=file://nic.es.net/.../ESNET-IP-ROUTING.TXT

[1]F. A. Abar, *Analysis of X Protocol and the Underlying Network Interface*, Proc. of the Second Int'l. Workshop on Software Engineering, Artificial Intelligence and ExpertSystems in High Energy Physics, Agelonde La-Londe Les Maures France, pp. 141-147.

[1]F. A. Abar, C. O'Reilly, E. Wicklund. *Network Load of X Terminal at CDF.* FERMILAB-TM-1773, March '92.

[1]R. Droms, W. Dyksen. *Performance Measures of the X Window System Communications Protocol.* Software Practice and Experience 20(S2) (October 1990), pp. 120-136.

[1]D. Mirchandani, P. Biswas. *Ethernet Performance of Remote DECWindows Applications.* Digital Technical Journal 2,3 (July 1990).

[1]W. Bux, K. Kummerle, H. Truong. *Balanced HDLC Procedures: A Performance Analysis.* IEEE Transactions on Communications COM-28, 11 (November 1980), pp. 1889-1898.

[1]S. S. Lam. *Carrier Sense Multiple Access Protocol for Local Networks.* Computer Networks 4, (February 1980), pp. 21-32.

Session 4

Data Handling and Storage Systems

Distributed Mass Storage and Management Systems at DESY

Martin Gasthuber,
Deutsches Elektronen Synchrotron DESY
Notkestraße 85
D-22603 Hamburg

Abstract

DESY is on the way of massive change in connecting and accessing Mass Storage systems. The main goal is to have full network connected storage devices combined with central network services which should be connected in case of store or retrieve of data.

Introduction

Todays HERA experiments show a huge demand of data access and capacity which could be seen as on the half way of that expected for future experiments like LHC. The online and offline reconstruction has such high IO demands in case of capacity, availability and bandwith that could not be satisfied with todays, mostly home grown solutions.

On the way of looking for alternatives, which should be designed according to the IEEE model, we discover a still increasing gap between given Hardware and operating systems compared to state of the art HSM products.

Current situation and penalties

DESY makes use of two storage device technologies to handle physics data. One is the IBM 34X0 and on the other hand the Ampex DD2 helical scan technology.

Both technologies have there inherent pros and cons. The DD2 system is our primary choice for physics data requiring high bandwidth and capacity, as long as there are small numbers of compute systems trying to access these systems. These constraints has its root in the fact of high capacity volumes and small number of drives. The 34X0 systems runs more efficiently in case of multiple independant storage requests of moderate data size.

This leads to the need for a more efficient management and administration system to overcome this data-type and location dependency which must be known by every user of these systems.

Both systems are mainly used for staging purposes to local disks where the real data access is done, although the direct access is still possible. Mostly this is done for some historical reasons and located in the, over years created reconstruction and analysis software. Direct access to tape data still comprise a big potential for increasing efficiency.

Looking for Alternatives

After completing the requirements we

had to look for available software packages. As mentioned above, we were really disappointed realizing that we will have no big choice. It should be pointed out that the pure performance is not the biggest issue on such a system. Criteria like heterogenious capabilities and simple procedures to add new devices are definetly more important.

By the end of 1993 DESY bought a source code license of the Open Storage Manager (OSM) from Lachman Technologies, Inc. This packages provides all features we initially expected, and will have additional features like virtual libraries etc. in the next release.

The Open Storage Manager

The product consist of three separate entities called Transmigrator, a migration capable UNIX filesystem, the Conservator where the main services of the HSM are located, and the Mediator including all hardware dependencies. The whole hierarchical Mass Storage Management is done by the Conservator and Mediator. The Conservator acts as a file based put/get repository with internal migration capabilities to support multi level device hierarchies. The Transmigrator uses the Conservator API to implement the migration capabilities. Other clients to the Conservator which are more appropriate for physics data, could easily be built and used in parallel to the Transmigrator. With todays UNIX filesystems it seems to be a bad choice to use the Transmigrator as an interface to the underlaying HSM, because all limitations on the disk filesystems are still valid for the Transmigrator. Future UNIX filesystems with better performance on sequential IO and bigger sizes will make the Transmigrator more attractive for physics data.

Network issues

All modules (processes) of the OSM could be run on one single host but more important, also on different hosts using TCP and/or RPC communication protocols. The data transport from the device controller to the requesting host is based on a standard TCP stream, thus allowing to have multiple different network types connected by IP routers. The data connection is directly between the controller and host and no management or controlling machine is involved in the data transfer.

Configuration example

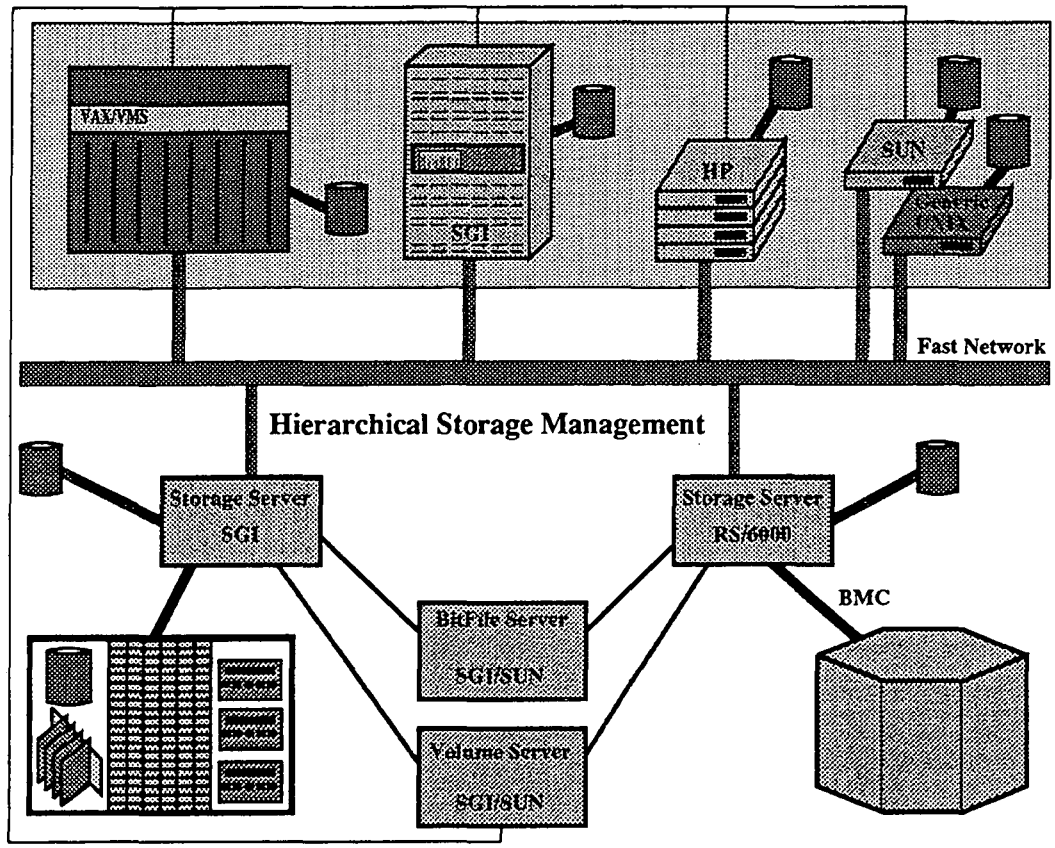
Figure 1 shows the HSM configuration which will be used at DESY

Clients

As mentioned above, there is still the need for a different client to the Conservator instead of the Transmigrator. A short term solution will be a simple database based mapping of filenames to bitfile ids. This could be built as a simple RPC service which could be queried by all relevant hosts. This scheme is only usefull when many hosts will only read the data and a small number of hosts doing the writes.

This *physics data* client implementation is not completed but could be done fairly easy within a short amount of time. It's clear that this approach is not the best one and it would be interesting to combine other ideas of that issue.

Tied together with the item *physics*



Configuration

data client is the item of the unit which you can put and get from the HSM service. Together with Lachman we have discussed the ability to specify not only the file (bitfile id) but also a byte offset and number of bytes you want to get from the HSM service.

Including this feature it will be an excellent combination with database based *event directories* to dramatically reduce the amount of data to copy.

Status

The first area of application is the connection of the ACS-4400 StorageTek silos to the UNIX based reconstruction machines. The control path for mount and dismount activities will be a separate Ethernet based connection to the existing IBM mainframe where the STK *Station-Platform* software is running. The intelligent controller in this game is performed by an RS/6000 workstation equipped with the BMC channel adaptor and an FDDI board for the network part of the data transfer. This machine runs the OSM mediator and Conservator's Storage-Server to serve the new device type **34X0**.

The port of STK client software to the *Station-Platform* is done and running. The OSM modules which needs to be run here are ported, merged together with the STK client code and runs.

The port of the remaining parts of the Conservator, which runs on SGIs is completed and running. This includes the main HSM server; the *BitFile* server in-

cluding the main database for bitfile ids.

All together composes a network attached 34X0 Mass Storage device, which still lacks the appropriate client for physics data (see item *Client* above).

The port of the Transmigrator will be finished by mid of 1994 and implies the most kernel (operating system) dependent work.

The next following project will be the integration of the existing Ampex DD2 library systems into OSM.

Conclusions

With the OSM we believe to go in the right direction and are well equipped for the next generation of physics data demands. We got the maximum flexibility we can get to adapt new client software on top and new hardware on the bottom side.

Centrally Managed Tapes at DESY

Otto Hell
DESY, Abt. R1, Notkestrasse 85, D 22603 Hamburg
Email: R01HLL@dsyibm.desy.de

Abstract

Tapes at DESY are managed by the computer center and thus offer virtually unlimited storage space to the DESY computer center users. This paper describes the most recent version of the DESY tape management system, its predecessors having been in service for many years.

Introduction

Tapes at DESY are managed by the computer center and thus offer virtually unlimited storage space to the DESY computer center users. This paper describes the most recent versions of the Archive and Retrieve Environment, of the system to reclaim free tapes, and of the "tape factory". My experience is with "round" tapes and cartridges only, therefore I shall not talk about Ampex or Exabyte tapes.

Tapes (36 tracks, 32000 bpi)

Sort of tape	total	robot	GByte
H1 (B-cartridges)	14000	1200	~ 1
ZEUS (D-cartridges)	14000	500	~ 1
centrally managed K	50000	12000	~ 1
centrally managed L	8000	5500	~ 1.6
HSM and others	3300	3300	~ 1
"round" A-tapes	~ 10200	0	< 0.18

The DESY tape robot consists of 4 STK silos, with 22480 slots for cartridges (currently totalling roughly 27 TByte), 36 I/O units operating at 4.5 MByte/sec, and 4 caps for injection / ejection.
A-tapes are no longer being written.

Cartridge Archive and Retrieve Environment

CARE – Movement of tapes between a tape robot and archive rooms

User Aspect

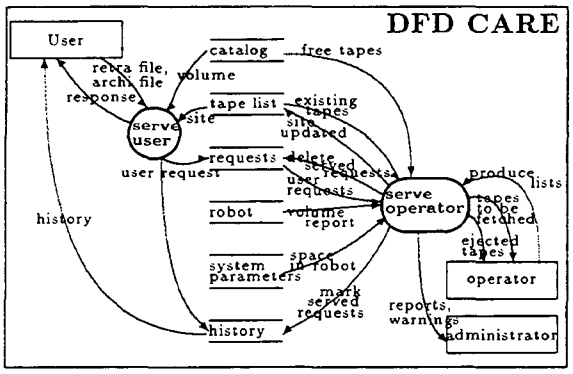
When the users list their catalog they see volumes on which their files currently reside and sites where the volumes are currently located. Both of these are subject to sudden change. Users also "see" an unlimited number of scratch tapes. they never touch centrally managed tapes, and they have no control over which volumes their data sets reside on. Their commands for having a file change site are:
retra = RETRieve tape data set from Archive room
archi = move tape data set into ARCHIve room

Various responses from the system are possible:

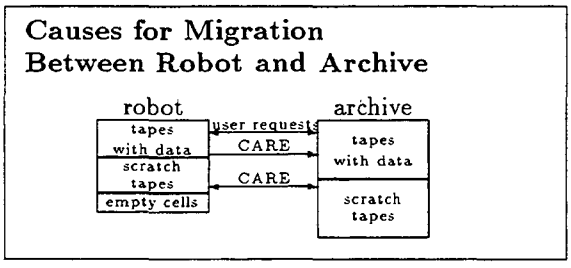
Will stay in the archive	**archi** after **retra**
Added to retra list	"normal" answer
Present in retra list	repeated **retra**
Present in the M-room	unnecessary **retra**
Please use "recall "	under HSM control

Users can inspect the "history file" of requests. It contains e.g. up to three dates: of the request and of the first and latest attempts to move the tape.

Administrator Aspect



Several parameters limit the volume numbers in the robot: minimum and maximum number of scratch tapes, minimum number of empty slots, and the number of days user data tapes may reside unused before becoming eligible for automatic archiving.



Operators' Aspect

For each sort of tape up to 3 lists of tapes may appear: data tapes to be retrieved, data tapes to be archived, and scratch tapes either to be retrieved *or* to be archived. Twice a day the operators start jobs to produce CARE lists for each sort of cartridge and fetch the tapes in the – printed – retrieve lists. Cartridges to be taken away come as jobs that eject a certain number of cartridges each. The operators run these jobs and take away the ejected cartridges. Printed lists for the "round" A-tapes come automatically once per night.

Experience and Outlook

CARE in its modern form runs quite stably since about September 1993. Although it was not initially foreseen to support the H1 and ZEUS raw data tapes, it was no problem to provide service for these when the experimenters asked for it.

The numbers of moved tapes vary enormously from day to day. The following are average daily numbers, compiled over the first 3 months of 1994:

Tape type	archive data	fetch	archive scratch	fetch	total moved
K	51	32	33	36	152
L	51	39	12	37	139
B	86	57	0	22	165
D	19	18	0	0	37
A	7	19	13	0	39

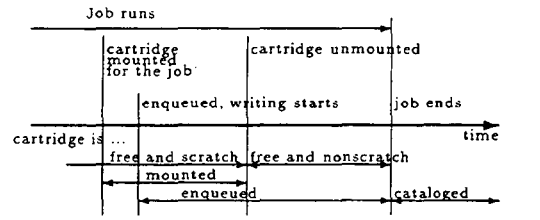
Future possibilities:
Currently the SGI UNIX machines are entering the robot stage and start handling tapes in the robot. Maybe CARE will soon handle U-tapes? Maybe one day CARE will run under UNIX and move Ampex tapes around?

BANDFREI – Reclaiming free tapes

Definition: **free tapes** are those that are *not* mentioned in the central catalog.

The procedure for putting free cartridges that reside in the robot into SCRATCH status is a complicated one and performed once per night: Data sets newly created are entered into the catalog at the end of the jobstep. A cartridge must not be made SCRATCH after having been mounted for writing. The various tools to determine free cartridges cannot be used all at once. Each one will run for a few minutes.

States of a cartridge being written in the robot



Available tools are a) The volume report of the robot: SCRATCH or NONSCRATCH, b) Enqueue SYSZVOLS lists all tapes with an OPEN by a job, c) Catalog listing for all occupied volumes. A cartridge is to be made scratch if the following holds in this order: It is 1. NONSCRATCH in the volume report, 2. not in Enqueue SYSZVOLS, 3. not mentioned in the catalog.

When a user deletes all her data sets on one of "her" tapes, this tape will be marked *free* during the following night. If it happens to be a cartridge and present in the robot, it will also be made SCRATCH. If it is not present in the robot, CARE may at any time put it on the list of scratch cartridges to be fetched into the robot. It will then be made SCRATCH during the following night.

Conclusion:

*Free*ing tapes is a part of central management of tapes. Thus we have been doing this for many years. The robot relieved the operators of a stupid procedure that had to be performed **very** carefully.

The present version of BANDFREI runs smoothly since we have the robot.

BBKOPIE – Using tape space effectively

Catering for tapes centrally makes it fairly easy to fill them to their capacity. Therefore we have done this for many years, and we have found it to be a "tape factory". Nowadays we have the "Improved Data Recording Capability" (IDRC) which put an end to filling tapes with interblock gaps. But the short files remain.

We offer only the (1 GByte) K-tapes to the users and copy all single volume files to the (1.6 Gbyte) L-tapes with the program BBKOPIE. This program writes multiple files to its output tapes, trying to fill them to their capacity.

At present this system runs in 4 to 5 instances in parallel:
- Two jobs move K to L as mentioned above and detailed below. We try to do this at a rate that keeps up with the production of files on K-tapes.
- One job empties old "round" tapes into the robot.
- The HERA experiments H1 and ZEUS have this tool at their disposal to move raw data tape files.

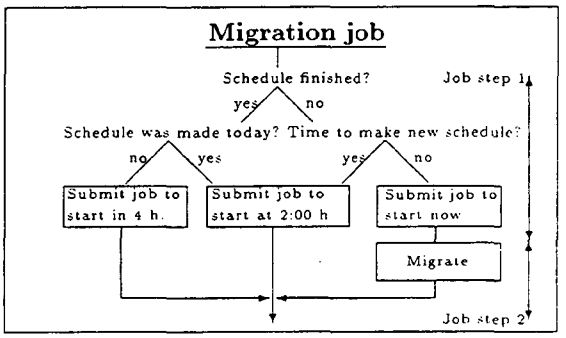
The BBKOPIE system works in two steps: 1. Creation of a schedule, i.e. a list of data sets to be moved, together with

their current location {tape, file number}, 2. Movement of files according to this schedule. Both activities will be considered briefly:

A new schedule is made every night. There are quite a few parameters controlling the planning, for instance: maximum number of files on an input tape, maximum sum of file lengths on an input tape (used for "compression" of tapes), date of latest usage of files on input tapes (in order to exclude very short-lived files), tapes in the robot to be considered: *only/no/don't care*. Of course a schedule can also be created from an explicit list of data set names.

During migration the first action after reading a record from the schedule consists of marking this record "in operation" and writing it back. This allows us to have "generations" of jobs working on the same schedule, each one starting after the previous one has stopped, thus avoiding "dead loops" of jobs if the move program "dies" in the attempt of moving a file. Alas such "deaths" happen quite frequently, e.g. if a file does not exist at the location where the catalog said or if a file gives rise to serious I/O errors, etc.

Next the data set location from the schedule record is verified against the catalog. If this fails, the schedule record is ignored. Now the file is copied and, upon successful termination, re-cataloged to its new location. The jobs that move files from K- to L-tapes have two job steps each, the first one submits a new job and the second one migrates.



Experience and Outlook

One of these jobs now reads up to 100 K-tapes per day, about two of these jobs running more or less continually. At the moment this keeps us "winning" free K-tapes, but this depends largely on user activities ...

The program BBKOPIE has been used for many years, e.g. many of the once 72000 A-tapes were written with it. It was used to empty our first 6000 cartridges which had only 80% of the capacity of the present K-tapes. It was also used to compact all K-tapes when IDRC came (and the new L-tapes kept us waiting). Nevertheless the current usage has revealed several weeknesses. In particular, a BBKOPIE-job "holds" all tapes it has written. Thus a job must be stopped when a user wants to access one of the files that were moved by it. We try to minimize this by only moving files that were not touched by the user for at least 7 days.
There are also efficiency problems ...

The future ? Working on the problems, more automation.

ID-1 Mass Storage System for Mainframe by using FDDI network

Y. Morita, H. Fujii, E. Inoue, H. Kodama, A. Manabe, A. Miyamoto,
M. Nomachi, Y. Watase and Y. Yasu

National Laboratory for High Energy Physics (KEK)
1-1 Oho, Tsukuba, Ibaraki 305 JAPAN

Abstract

We have developed an ID-1 mass storage system as a distributed data server for Fujitsu mainframe computers. The system consists of a SONY ID-1 recorder DIR-1000, a tape robot system DMS-24 and a SCSI-II interface DFC-1500, which are connected to Sparc Station 10 with an FDDI interface. The maximum speed of 7.5 Mbytes/sec is achieved for data transfer between Sparc Station 10 memory and DIR-1000 with a buffer size of 1 Mbytes. The system has been used successfully since last October to migrate more than 1 Tbytes data.

Introduction

We have been using the Fujitsu mainframe M1800/30 and M1800/10S (FACOM), and tape library systems F6453 (VHS) and F6455 (IBM 3490) for 7 years. At present, the total capacity of the system is 7.8 TB. This storage system has been fully utilized by three TRISTAN experimental groups, VENUS, TOPAZ and AMY. Due to a lack of room to expand, we need a more space effective storage system to migrate these data sets. In addition, as the second phase of the TRISTAN project has been approved, the BELLE collaboration will require a storage capacity of 30 TB/year and data I/O speed of at least 12 MB/sec by 1997[1].

To migrate the data of the current TRISTAN experiments, and to meet the requirements for the future B-factory experiment, we have been testing SONY ANSI ID-1 format tape as one of the candidate storage media[2]. We have developed a distributed data archival system for this purpose. The system has been used since October 1993, for the migration of a part of TOPAZ data sets which now reside in F6453 and F6455 tape library system.

The system consists of a SONY ID-1 recorder DIR-1000 with SCSI-II interface DFC-1500, SONY DMS-24 Tape Robot, and a Fujitsu S4/10 workstation (Sparc Station 10) for the drive server and an S4/2 for the robot control. The system is connected to the mainframe via FDDI. The software protocol for data transfer is TCP/IP.

DFC-1500 is a SCSI interface for DIR-1000. It is designed to work as a standard cassette streamer tape drive, except for a few vendor unique commands to initialize, mount, and take a directory of a tape. SONY provides a unique set of UNIX commands to handle functions for SONY NEWS workstations. However, for distributed and multi-vendor environment, we need to connect the drive to other flavors of workstations as well. Therefore SONY cooperated with us to modify the DFC-1500 interface.

The revised DFC-1500 can accept the control commands from both SCSI and

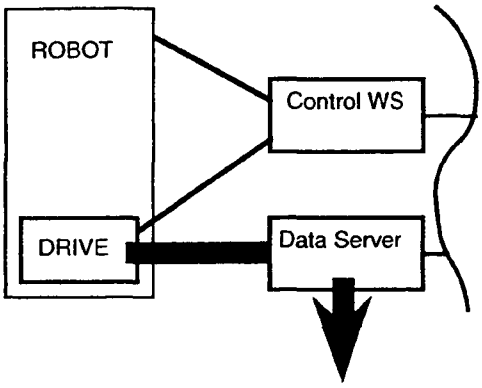


Fig. 1 Data path and Control path

RS232C (Fig. 1). While in SCSI execution, the command from RS232C is ignored. The acceptable commands from RS232C are shown in Table 1. The command through RS232C is sent as a packet and the reply is also returned as a packet (Fig. 2). The datagram of the packet is ASCII-encoded string.

Tape Management Software

The RS232C ports of DMS-24 and DFC-1500 are connected to a Fujitsu S4/2 workstation. Client-server tape management software[3] is running on the

Table 1 DFC1500 RS232C Commands

MOUNT	load tape and read directory block from tape to DFC-1500 memory
UNMOUNT	write back the directory information from DFC-1500 to tape and unload
EJECT	eject tape from drive
ERASE	initialize tape
RECOVER	rebuild directory information from tape
REQUEST SENSE	request drive status information

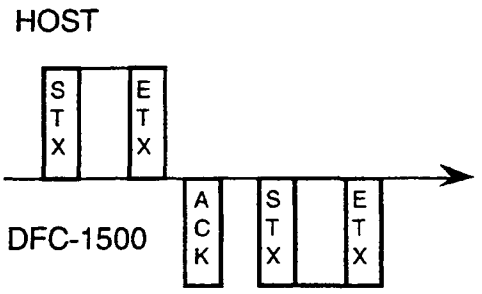


Fig. 2 RS232C Command Sequence

S4/2. A server daemon program *chsd* runs on the S4/2, and it controls the DMS-24 and DFC-1500 RS232C ports. The daemon receives tape mount requests from client workstations *(chsclerk)* via TCP/IP socket. This pair of program enables a remote access to the tapes in DMS-24 from the registered workstations and the system serves as a network data server for distributed environment as well as for FACOM. Files on the DIR-1000 drive are accessed with a combination of *tar* and *dd* commands. Files can also be staged to the S4/10 disks for the access from FACOM. Further development of the management software is planned in collaboration with SONY.

Transfer Speed

We have tested the recording performance of the DIR-1000 by directly writing to or reading from the S4/10 memory. Fig. 3 shows the data transfer speed with various buffer size ranging from 512 bytes to 1 Mbytes. It is clear from the figure that the performance of DIR-1000 strongly depends on the buffer size. The measured maximum speed is 7.5 Mbytes/sec with a buffer size of 1 Mbytes.

We have also measured the data transfer time from Fujitsu F6455 tape to the DIR-1000. On FACOM, a data file on F6455 tape is staged to the mainframe disk with a JCL script (step 1). The file is then transferred to S4/10 disk with FTP over

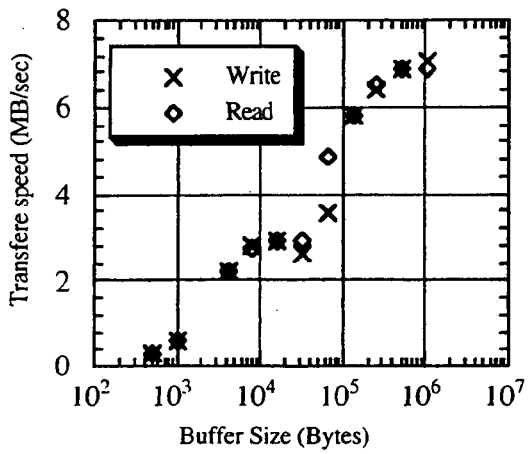


Fig. 3 DIR-1000 Data Transfer Speed

FDDI (step 2). Finally the file is recorded on the ID-1 tape from the S4/10 disk (step 3). Fig. 4 shows the elapsed time of each step with various size of data file. Average throughput of this system is 513 Kbytes/sec with overhead of 145 seconds.

Summary

SONY DIR-1000 ID-1 drive has been tested with a SCSI-II interface DFC-1500, which is connected to a Sparc Station 10. The drive works as a standard cassette tape streamer except for a few vendor unique commands. Performance of the DIR-1000 drive was measured to be 7.5 Mbytes/sec when it is directly written to or read from Sparc Station 10 memory via SCSI-II interface DFC-1500. System has been developed to utilize this drive and DMS-24 tape robot as a distributed network data server. Effective throughput of the system from FACOM tape library to this system is 513 Kbytes/sec with average overhead of 145 seconds. The throughput is degraded due to a combination of several effects: access time of the FACOM tape library, IP(FTP) overhead of the FDDI network, and staging overhead to the SCSI disk. Direct access to the DIR-1000 drive from the network tape server software would be necessary to utilize the potential speed of the DIR-1000 drive.

The system has been used by TOPAZ since October 1993, and so far 1 TB of TOPAZ data set has been migrated from the FACOM tape library as a backup data server.

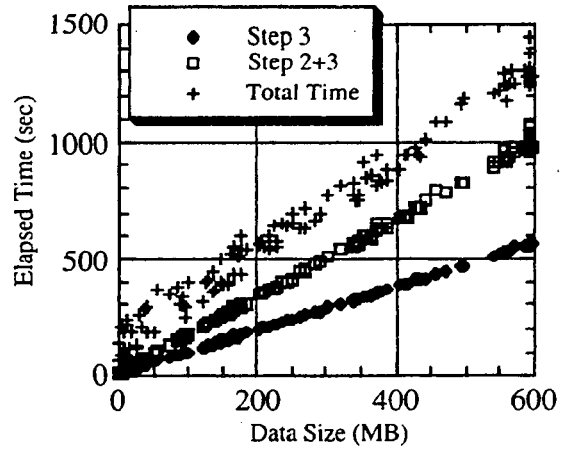


Fig. 4 Transfer elapsed time from F6455 to DIR-1000

Step 1: F6455 to FACOM disk
Step 2: FACOM to WS disk
Step 3: WS disk to DIR1000

References

[1] The BELLE Collaboration, *Letter of Intent,* February 1994, to be submitted to the TRISTAN Program Advisory Committee.

[2] H. Fujii *et al., An ID-1 Mass Storage Project at KEK,* Proceedings of the International Conference on Computing in High Energy Physics '92, Annecy, France, 21-25 September 1992, CERN.

[3] Y. Morita *et al., Robotic Tape Server for Distributed UNIX Computing,* Proceedings of the International Conference on Computing in High Energy Physics '92, Annecy, France, 21-25 September 1992, CERN.

Data Organization, Management and Serving in D0 Collider Experiment

Krzysztof Genser
Fermilab
Batavia IL 60510-0500 USA

for D0 Collaboration

Abstract

D0 Experiment at Fermilab has collected some 20 Million events in Collider Run Ia and is in the process of run Ib data taking. The total data sample includes about 40 TBytes of data organized in 750,000 files. Compressed data sets are residing on disk, whereas the full set of data resides on 8mm tapes. All the data is served from the D0 File Server, cluster of VMS workstations, to the remote analysis clusters.

HARDWARE CONFIGURATION AND ANALYSIS MODEL

D0 physics analysis and data serving is based mainly on DEC VMS machines. There are two VMS analysis clusters (FNALD0 and D0SFT consisting of $\sim$ 140 nodes in total), and one File Server VAX Cluster (D0FS). The analysis clusters are located on the D0 site which is about two kilometers from the Feynman Computing Center (FCC) where D0FS is located. All the clusters are internally connected with multiple Ethernet segments, while the D0FS to analysis Clusters communication is based on FDDI. The FDDI connection provides a sustained rate of 4MB/sec, with peaks reaching 8MB/sec.

D0FS has the total capacity of 34 nodes: Three VAX 4000–90 used as the bootnodes, 17 VAX 4000–60 used as Tape Server nodes and 13 VAX 3100–76 used as File Servers. Each Tape Server node is equipped with two 8mm tape drives and a disk stripe set for the tape staging. Each File Server node has three disk stripe sets connected to it. At present the VAX 3100–76 nodes are being replaced by Alpha's AXP 3000–400 with one AXP machine already installed and three more to be installed within the next three months. The total mass storage available consists of 250 disks ($\sim$ 500GB) and 38 Exabyte tape drives. The disk storage is used primarily for DST data sets ($\sim$ 300GB) and for staging data from tapes ($\sim$ 90GB).

D0FS disks are accessible from the analysis clusters using DEC DFS, for striped sets residing on VAX'es, or DECNET for disks mounted on Alpha's.

The D0 Analysis Model is based on "sending data to the jobs" strategy. Most of the I/O intensive analysis jobs run on several FNALD0 VAX 4000–90's which

are directly connected to the FDDI ring. The output of such jobs is directed onto some of the disks (300GB) located on the analysis clusters. Given the fact that most of the data resides on tapes on D0FS, whereas most of the analysis is done on analysis clusters, a client–server scheme was developed and implemented using FATMEN/VAXTAP packages [1, 2].

DATA VOLUME AND CHARACTERISTICS

During the run Ia (1992/93) D0 has collected 20M events resulting in about 10TB of RAW detector data. Fourteen Million events were written to the two main physics streams, ALL and EXP, the remaining 6M events were taken for various calibration purposes. Physics data were reconstructed almost in real time on the Fermilab UNIX Farm [3, 4]. The offline reconstruction program reconstructs all primary partons (jets, leptons, missing energy) for every event and writes the results to an STA file (full information, $\sim$ 500kB/event) and a DST file (compressed information, $\sim$ 20kB/event). Later, the STA and DST files are split into several physics–oriented streams. To minimize the amount of I/O volume during the analysis phase a micro–DST file (super–compressed information, $\sim$ 5kB/event) was introduced recently. All these files are stored on 8mm tapes. The file size ranges from 5 to 400MB with 10−800 files per tape. Selected samples of DST's and complete set of micro–DST's are placed on disks on D0FS.

At the end of the run Ia D0 had some 65k of RAW data files, 420k STA files and 190k DST files. The expected number of events during run Ib (1993/94) should be around 50 Million. At the time of this conference D0 had a total of 750,000 catalogued files including 36,000 disk files. The number of catalogued 8mm tapes was more than 15,000.

MANGEMENT AND ACCESS TOOLS

All data files produced by the offline production systems are catalogued in the Production Data Base (PDB) and in FATMEN [1].

PDB is a relational database based on DEC RdB. It keeps track of the file content and the file history. In addition it provides the experiment–related (triggers) and accelerator–related (luminosity) information on a file–by–file basis.

FATMEN is used to do the physical management of the files. It is the primary source of the information on the physical location of the data sets. The FATMEN/VAXTAP combination provides the only access to the data sets residing on magnetic tapes.

FATMEN SETUP AND FUNCTIONS

There are three FATMEN catalogues at D0. Each cluster has its own copy of the catalogue. This choice was dictated by the difficulties with the lock manager on the DFS–mounted disks. The master version of the catalogue resides on D0FS and it receives all updates. Copies of the catalogue on the analysis clusters are updated with a typical delay of 10 minutes. The catalogue has currently 750,000 entries organized into 7,500 directories. Directory structure reflects the source of the data (D0 Detector, Test Beam, Monte Carlo), level of the reconstruction (e.g. RAW, STA, DST...) and physics content of the data files (e.g. Top/WZ/QCD

etc...). We have found that performance of the ZEBRA RZ package (the underlying Memory Management package used by FTAMEN) degrades considerably with more than $\sim$ 1,000 entries in a single directory, therefore we have split our data set into some artificial subdirectories based on the run number.

Almost all access to the data by users is done from the analysis clusters. Disk–resident files are mostly used "directly" via DFS and DECNET, whereas tape resident files are used via FATMEN/VAXTAP interface.

The monitoring of the file usage is done on the analysis clusters by analysing the FATMEN server log files. The file access history is stored in an ORACLE database.

Given the fact that the FATMEN catalogue provides the only access to the data sets residing on tapes it must be kept up to date and be accessible at all times. The integrity check of the catalogues is done each night on all three clusters to make sure that any possible corruption of the database is promptly addressed. In addition to the formal catalogue checks various tests of the catalogue content consistency are performed. The latter is done on D0FS together with all the possible data file management (e.g. backup of the disk files to tapes etc...).

Multi File Tape Server Mode Staging

Storage of files on a large capacity tapes creates new problems with staging. If entire tapes are copied to staging disks then the staging disks are filled out instantly. If one file at a time is staged from the tape then the same tape is mounted multiple times as the requests come through. To optimize the access to data from the magnetic tapes a new staging mode was added to the FATMEN package [1, 2] and was called the Tape Server Mode. In this scheme a request for a tape resident data set creates a process on D0FS which stages the file from the tape. In addition this process satisfies any other requests for files from the same tape. The requests are being received till the moment the tape is dismounted which happens a while after all the present requests are fulfilled.

The Tape Server Staging Mode is used in D0 as the first step of analysis jobs to request prestaging of files to be analyzed. Most of the file staging is done asynchronously and the analysis jobs do not wait for each specific file.

DATA ACCESS STATISTICS

FATMEN server log files contain complete records of the data usage, including file names, access modes (from disk or tape, from which cluster/node), access patterns, etc... Analysis of these statistics shows that the disk–resident data sets are used mostly via direct access using DFS or DECNET. The fraction of disk files accessed via FATMEN grows steadily, especially because DST data sets are being archived to tapes. File access via FATMEN generic names guarantees a transparent access to the data, irrespective of its location.

A typical analysis job reads several hundreds files. For disk resident data the average access time (i.e. time between the file request and opening of the file) is of the order of 30 seconds, and it is entirely dominated by the D0FS DECNET/DFS server response time. For tape resident files the average time to open a single file is about 1h and is primarily due to con-

tention over the tape drives and/or operator mount time. Since the use of prestaging the average tape file access time was reduced to only $\sim$ 100 seconds as most of the files were staged in groups.

REMARKS ON PERFORMANCE

The File Serving system performed generally quite well, providing access to data with the bandwidth exceeding the specification by a factor of two. FATMEN, after some initial debugging period, was found to be an extremely useful package. The outstanding problem appears to be the unreliability of the 8mm tapes and tape drives. An average lifetime of the tape drive head is of the order of few months and the I/O error rate is high.

NEAR FUTURE PLANS

We are in the process of implementing the event directory approach to access our disk residing data. We are commissioning our Alpha/VMS cluster as the third analysis cluster. We are working on providing FATMEN access from our small UNIX cluster. We are designing an analysis server model where the majority of the jobs will be sent to data.

SUMMARY

The current experience shows that it is possible to manage data samples of the order of 10^6 files and provide transparent access to them. This possibility was utilized by D0 to analyze the Collider data from run Ia. This in turn resulted in a remarkable achievement of several published physics papers in less than just two years since the commissioning of the detector in mid 1991. We are looking forward for the challenges provided by the expected amount of data from run Ib.

ACKNOWLEDGEMENTS

We would like to thank Jamie Shiers, CERN/CN for the help provided while using his packages [1, 2].

REFERENCES

1. J. Shiers, "FATMEN — Distributed File and Tape Management", CERN Program Library Q123, CERN, 1994.
2. J. Shiers, "VAXTAP — VAX/VMS Tape Handling Package", CERN Program Library Z312, CERN, 1992.
3. K. Denisenko, "D0 (FNAL) Farm Production System" in this Proceedings.
4. S. Wolbers, "Analyzing Terabytes at Fermilab: Present and Future" in this Proceedings.

A Data Base for Tracking File Processing History*

Paul S. GEE, Information and Computing Sciences Division
Matthew A. BLOOMER, Nuclear Science Division
Douglas L. OLSON, Nuclear Science Division
Lawrence Berkeley Laboratory, University of California
Berkeley, California, 94720, USA

The STAR Data File Manager, a data base for tracking file processing history, records the program and inputs used to generate each data file. It is used in a large physics collaboration to help organize and support the data file processing needs of the experiment.

1. Introduction

A Data File Manager is being developed for tracking all data files and their processing history for the STAR experiment at RHIC. It records the program and inputs used to generate each file, and has utilities to manage the possibly multiple file storage locations such as on disk and on tape.

The processing history of a data file helps to identify the initial conditions and program name and versions used to create the file. The initial conditions can include input files and job invocation script files. This information classifies the data files for retrieval purposes. For example, when new versions of software or new calibrations become available, the processing history helps to identify the data files that need to be reprocessed.

The information on the locations of a file facilitates efficient file retrieval. Possible locations for a file may include multiple locations on the same file system or different file systems. Other locations may include various kinds of near-line storage such as automatic tape libraries, or off-line storage such as removable tapes or disks. Knowing the location of the nearest copy of the file makes file retrieval faster and reduces load on the overall system.

The Data File Manager (DFM) tracks data files which result from data acquisition, detector calibration, simulation and off-line event reconstruction. The run configurations and simulation configurations, along with other parameters associated with the production of a data file, are also recorded.

This paper presents a prototype implementation of the STAR Data File Manager using the Informix data base. We describe our data flow model for representing processing history of files.

2. Design Goals

The design goal of the Data File Manager is to maintain adequate information to answer the following types of questions:

1. What data files were generated based on a certain run configuration?
2. What was the run configuration used to generate a certain data file?

* This work was supported in part by the Director, Office of Energy Research, Office of High Energy and Nuclear Physics, Division of Nuclear Physics and by the Office of Basic Energy Sciences, Division of Nuclear Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098

3. What are the possible calibration files to use for analysis of a group of events, using a given processing module?
4. What are all the ancestor files of a given file?
5. What are the locations of a file on disk and on tape?

The data base design must be consistent with the information being recorded. The DFM must provide interactive interfaces to query the data base. There must be a convenient way to populate the data base with accurate information.

3. Logical Data Model Design

The heart of DFM is a data base to track the file processing history. The logical data model (LDM) design of the data base is illustrated in figure 1. A LDM is concerned with the objects or entities and the relationship between the entities. The entities, shown as labeled rectangular boxes in figure 1, are **job**, **file**, **disk_file**, and **tape_file**. Attributes of each entity are written in the boxes, with the primary key attribute marked by an asterisk. The relationship between boxes is indicated by labeled arrows, with the

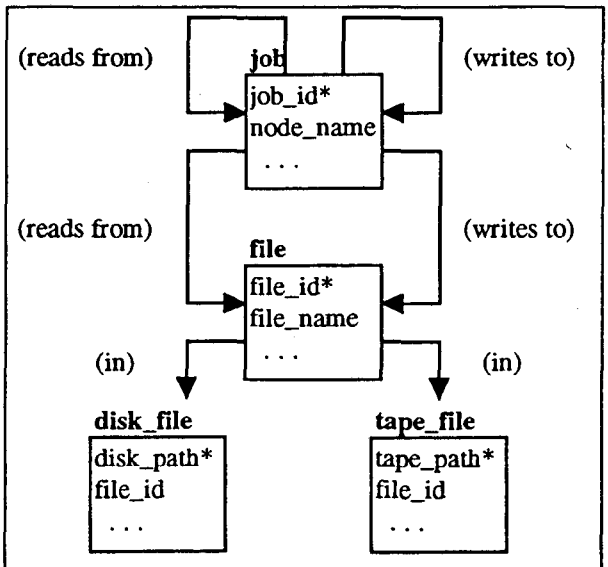


Figure 1: LDM of File Processing History

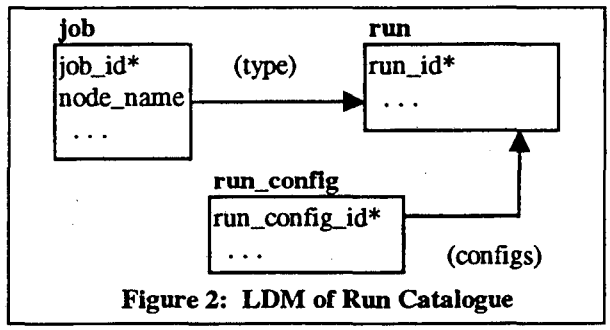


Figure 2: LDM of Run Catalogue

label signifying the relationship. This LDM is designed to be completely general. It can accommodate a job reading from multiple files and jobs, and writing to multiple files and jobs. Furthermore, a file may be read from or written to by multiple jobs. Also, a file may be stored in any number of disk or tape locations. The generality permits modeling of most real world situations. However, we may apply constraints to the model to mirror how we actually wish to use the data base.

With the core of the LDM design in place, application specific information may be added. For example, figure 2 illustrates how information about experimental runs may be added. In this design, a **run** entity is a type of **job**, and a **run** is configured by a **run_config** entity. We have also designed the LDM to track information on simulation and calibration.

4. Relational Data Base Tables

A prototype of the relational data base tables corresponding to the above LDM is shown in figures 3 and 4. In these diagrams, the rectangular boxes represent relational data base tables. Arrows between boxes indicate references from one table to another. Fields of the table are written in the boxes. The '#' symbol after a field name indicates a non-unique foreign key, that is, a reference to another table with duplicates allowed. The '@' symbol similarly indicates a unique foreign key. For example, the file_id field in

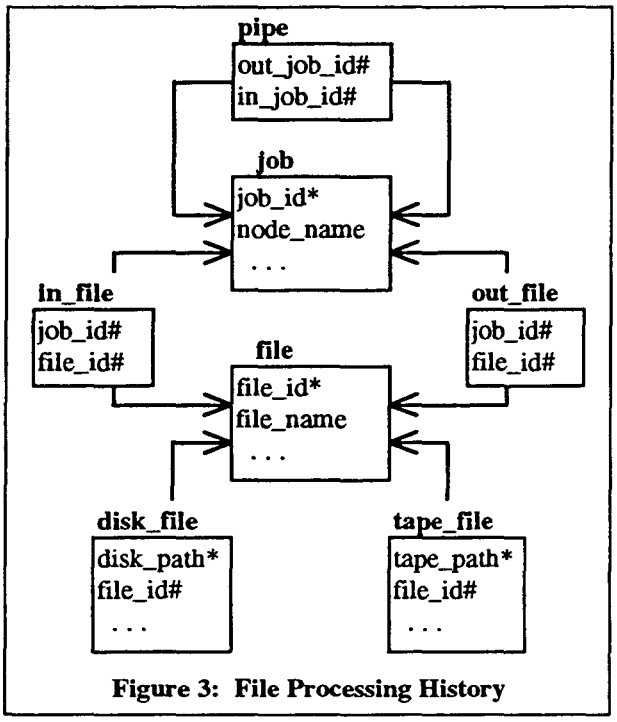


Figure 3: File Processing History

tape_file is marked with '#' to indicate that it is a non-unique foreign key, implying that a file may be in multiple tape locations. The job_id field in **run** is marked with '@' to indicate that it is a unique foreign key, implying that a job can only be associated with one **run**.

5. Data Base Interfaces

Two types of interfaces are provided for effective usage of DFM. The first is an interactive interface to allow query and modification of the data base by an interactive user. In the prototype, we have provided data entry forms for accessing the history and locations of a file. Application level data entry forms include those for tracking runs, simulations and calibrations.

An important aspect of the DFM is to populate the data base with accurate and complete information on all data files generated. It must be relatively automatic and easy to enter the data into DFM. We have provided DFM commands that can be

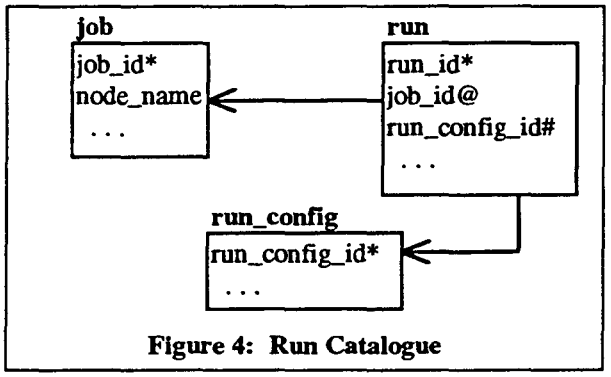


Figure 4: Run Catalogue

included in job invocation scripts to take the relevant information to generate an SQL script. The SQL script can then be used to update the DFM data base. Separately generating and then running the script has the benefit of not requiring that the DFM data base be available at the time that the data file processing occurs. Furthermore, it allows an additional opportunity to decide if the set of generated file is worth recording in the DFM data base.

6. Conclusion

In the field of high energy physics today, many experiments are performed by a large collaboration comprising of many researchers from a multitude of sites and countries. Many files are generated in the course of the experiment, from data acquisition to on-line and off-line analysis. The ability to keep track of data file processing becomes increasing important in helping to organize and support the experiment. It will also be valuable in providing a trace-back capability to help validate conclusions drawn from the data.

The DFM is an attempt at providing a tool to track the processing history. It has a user interface for interactive query and modification of the data base. It also has tools to support data gathering from job invocation scripts.

The PASS Project: A Progress Report[1]

D. R. Quarrie[2], C. T. Day, S. Loken, J. F. Macfarlane
Information and Computing Science Division
Lawrence Berkeley Laboratory

D. Lifka, E. Lusk, D. Malon, E. May, L. E. Price
High Energy Physics, Mathematics and Computer Science,
and Decision and Information Science Divisions
Argonne National Laboratory

L. Cormell, A. Gauthier, P. Liebold, J. Hilgart,
D. Liu, J. Marstaller, U. Nixdorf, T. Song
Physics Research Division
Superconducting Supercollider Laboratory

R. Grossman, X. Qin, D. Valsamis, M. Wu, W. Xu
Laboratory for Advanced Computing and
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago

A. Baden
Department of Physics
University of Maryland

ABSTRACT

The PASS project has as its goal the implementation of solutions to the foreseen data access problems of the next generation of scientific experiments. It is in the process of transitioning from an exploratory phase, where the focus has been on understanding the requirements and available technologies to an implementation phase, where detailed design work is commencing on a common framework for scientific applications.

INTRODUCTION

The Petabyte Access and Storage Solutions (PASS) project [1] has as its goal the implementation of solutions to the foreseen data access problems of the next generation of scientific experiments. These are characterized by a very large sample of complex event data (~10^{15} bytes), a dilute signal and a large (~1000) and geographically dispersed user community. Although the original focus of the PASS project was the experiments at the SSCL, its approach is broad enough to encompass many areas of scientific computing. Target customers now include experiments at RHIC, CEBAF, the B-Factory at SLAC, NASA and governmental projects.

The general approach has been to investigate the feasibility of using distributed database technology in conjunction with

[1] Supported by the U.S. Department of Energy under Contract No. DE-AC03-76FS00096.

[2] Address: LBL MS50B-3238, 1 Cyclotron Road, Berkeley, CA 94720
Email: DRQuarrie@LBL.Gov

hierarchical mass stores to handle the conflicting requirements for the storage of large amounts of experimental data with the desire to provide rapid access to selected components of that data from a dispersed user community.

ORGANIZATION

PASS has been organized in two distinct phases, where the initial phase has been mainly an exploratory one, with the focus being on benchmarks and technology demonstrations to demonstrate proof of principle and understanding of the available hardware and software technology. The culmination of this initial phase has been the development of an Architectural Model that forms the basis for the second, implementation, phase. The second phase is focused on the detailed design and implementation of components identified by the Architectural Model. The goal is this phase is the creation of second-generation prototypes, eventually leading to systems capable of handling the data access demands for a wide variety of scientific disciplines.

The exploratory phase has involved two generations of testbeds, the first of which demonstrated that the database approach was feasible. The second generation testbeds have extended the size of the data sample, the complexity of the queries and have embarked on an investigation of access to distributed data and distributed queries.

MARK 0 TESTBED

The first generation testbed was described at an earlier CHEP Conference [2]. It demonstrated the feasibility of the database approach, and indicated that an object oriented approach using either an object oriented database manager or persistent object store (or a combination of the two) was the preferred approach. However, it was limited in the size of the data sample and complexity of the physics queries that could be performed.

The Mark 0 testbed indicated that either a full-scale object oriented database or an object persistence manager with lower overheads were suitable candidates for further investigation. PTool [3] is a persistent object manager developed at the University of Illinois at Chicago. A 32-bit version was used during the Mark 0 tests, whereas a 64-bit version with significantly enhanced capabilities has since been developed.

MARK 1 TESTBEDS

Several Mark 1 testbeds are underway. A testbed at the SSCL was designed to demonstrate the use of 19mm helical scan tape technology within a database environment and to increase the data sample to 10GB. The configuration is shown in Fig. 1. Two

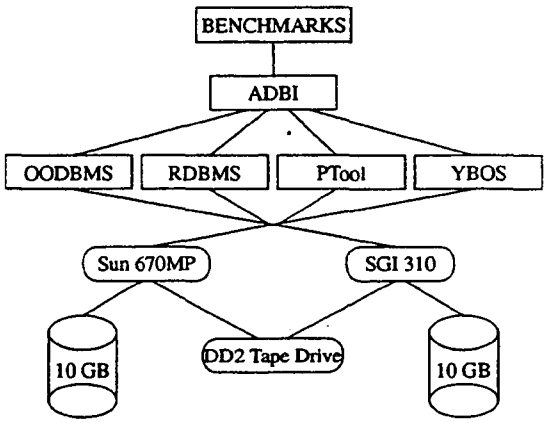


Figure 1. SSCL Configuration

different computer architectures were investigated as well as the same set of databases as for the Mark 0 tests, but with a significantly enlarged data sample. The demise of the SSCL prevented completion of these tests.

Another testbed has focused on the use of the 128-processor IBM SP-1 computer system at ANL and the development of parallel query processing techniques using the PTool persistent object manager. Several replicated query strategies have been investigated as well as techniques for the movement of data within a distributed database

environment. This project is the subject of a paper to be presented at this Conference [5].

Work at UIC has been targeted towards the further development of the PTool persistent object store into a fully distributed environment. This has allowed tests to be made with different caching, migration, and replication algorithms for interfacing low overhead, high performance persistent object managers to hierarchical storage systems. This work is described in Ref. [6].

Technical difficulties limited the complexity of the queries that were possible on all the above testbeds, so another testbed at LBL has attempted to integrate an existing physics analysis framework with a distributed OODBMS so that a larger sample of data may be examined and typical physics queries may be run. The CDF analysis framework has been modified to allow the OODBMS to become the source of event data, whilst allowing the user code to remain unmodified. This framework has further been enhanced to act as a testbed for the use of a distributed database based on the concepts of the Object Management Group [7] and the Common Object Request Broker Architecture (CORBA) which forms the basis for the Architectural Model described in the next section. This testbed is described in Ref. [8].

ARCHITECTURAL MODEL

Focal point of the exploratory phase of the project has been the development of an Architectural Model [9]. It is comprised of four major components:

(a) The operational and technical requirements. Operational requirements are broad capabilities that result from the environment within which the system must operate. Characteristics which drive the operational requirements are the high input bandwidth, the very dilute signal and the widely dispersed scientific community. Technical requirements a specific capabilities that the system must exhibit in order to match the operational requirements. These include the input bandwidth, uniformity and scalability constraints, flexibility and extensibility in the data organization, modes and patterns of access, concurrency and access controls, and query language.

(b) An abstract reference model. This describes a system that matches the requirements in terms of its components and the mechanisms by which they communicate, but does not discuss policy or management issues that would be necessary to match the model to an actual implementation. This reference model builds upon the concepts and terminology of several standards organizations, including the Object Management Group and the Object Database Management Group [10].

(c) An implementation model. This describes a conceptual implementation, matched to the requirements of a HEP collider experiment. It consists of a set of hierarchical data servers.

(d) A discussion of some design and policy issues. The reference model lacks many of the characteristics of a final implementation that accommodate technological constraints to optimize the available capacities. For example, the reference model states that data must be movable amongst a hierarchy of data stores in a manner that optimizes response times to the most frequent access patterns. How best to achieve this caching and migration, and whether to replicate or move the data, is a detailed policy and design issue that lies beyond the scope of the reference model. We have identified several such issues that are worthy of more discussion and have presented aspects of their impact, even though at this stage in the design process we cannot necessarily identify the correct design choice.

SUMMARY

This paper presents an overview of the PASS project, summarizing the results from the various testbeds and presenting a brief description of the Architectural Model. The Mark 0 tests showed the desirability of analyzing events using distributed database and distributed object computing technologies. The Mark 1 tests showed how this technology could be interfaced to hierarchical storage systems resulting in our current architectural model. We are now ready to scale this technology up to the production level.

We have embarked on a detailed design of an implementation of the Architectural Model and the software products developed thus far are being retrofitted to conform. In the short time frame, work is underway to populate object stores with D0 data at Fermilab and the longer term plan encompasses experiments at both SLAC and RHIC with the goal being the availability of production systems in the LHC time frame.

REFERENCES

[1] A. Baden and R. Grossman, *A Model for Computing at the SSC*, Superconducting Super Collider Laboratory Technical Report No. 288, 1990.

[2] C. T. Day et al., *Database Computing in HEP -- Progress Report*, Computing in High Energy Physics, 1992.

[3] R. Grossman and X. Qin, *PTool: A Software Tool for Working with Persistent Data*, Laboratory for Advanced Computing Technical Report Number 92-11, University of Illinois at Chicago, 1992.

[4] A. Gauthier et al., *PASS Project at the SSC: Test Plan for 10 Gb Database Comparison*, PASS Project Note 93-04, April 1993.

[5] D. Malon et al., *Parallel Query Processing for Event Store Data*, to be presented at CHEP '94.

[6] E. N. May et al., *A Demonstration of a Multi-level Object Store and its application to the Analysis of High Energy Physics Data*, to be presented at CHEP '94.

[7] Object Management Group, *The Common Object Request Broker: Architecture and Specification, Revision 1.1*, OMG TC Document 91.12.1, 1991.

[8] D. R. Quarrie and C. T. Day, *Using a Distributed OODBMS as a Source of Events for CDF Physics Analysis*, to be presented at CHEP '94.

[9] D. Lifka et al. (the PASS Collaboration), *The PASS Project Architectural Model*, to be published.

[10] R. G. G. Cattell et al., *The Object Database Standard: ODMG-93*, Morgan Kaufmann, 1994.

The PASS Project Architectural Model*

C. T. Day†, S. Loken, J. F. Macfarlane and D. R. Quarrie
Software Technologies and Applications Group
Information and Computing Sciences Division
Lawrence Berkeley Laboratory

D. Lifka, E. Lusk, D. Malon, E. May, and L. E. Price
High Energy Physics,
Mathematics and Computer Science,
and Decision and Information Science Divisions
Argonne National Laboratory

L. Cormell, A. Gauthier, P. Liebold, J. Hilgart,
D. Liu, J. Marstaller, U. Nixdorf, T. Song
Physics Research Division
Superconducting Supercollider Laboratory

R. Grossman, X. Qin, D. Valsamis, M. Wu, W. Xu
Laboratory for Advanced Computing
and Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago

A. Baden
Department of Physics
University of Maryland

ABSTRACT

The PASS project has as its goal the implementation of solutions to the foreseen data access problems of the next generation of scientific experiments. The architectural model results from an evaluation of the operational and technical requirements and is described in terms of an abstract reference model, an implementation model and a discussion of some design aspects. The abstract reference model describes a system that matches the requirements in terms of its components and the mechanisms by which they communicate, but does not discuss policy or design issues that would be necessary to match the model to an actual implementation. Some of these issues are discussed, but more detailed design and simulation work will be necessary before choices can be made.

---

* Supported by the U. S. Department of Energy under Contract No. DE-AC03-76FS00096.

† Address: Lawrence Berkeley Laboratory, MS 50B-3238, 1 Cyclotron Road, Berkeley, CA 94720.
Email: CTDay@LBL.Gov.

INTRODUCTION

The Petabyte Access and Storage Solutions (PASS) project [1], [2] has as its goal the implementation of solutions to the foreseen data access problems of the next generations of scientific experiments. These are characterized by a very large sample of complex event data, a dilute signal and a large and geographically dispersed user community.

We describe an Architectural Model that is comprised of four major components: the operational and technical requirements, an abstract reference model, an implementation model, and some design issues.

The architectural model is the basis of the transformation of the PASS project from its initial exploratory phase to an implementation phase where components of the model are being designed and implemented.

REQUIREMENTS

The requirements are presented as both operational and technical. Operational requirements are broad capabilities that result from the environment within which the system must operate. We have used the model of an HEP collider detector as the basis for these, although most would be similar in other areas of scientific research such as nuclear physics, satellite telemetry and high resolution astronomical surveys. Of prime importance for the operational requirements are that the input bandwidth is high, that the signal is very dilute and that the scientific community is widely dispersed.

Technical requirements are specific capabilities that the system must exhibit in order to match the operational requirements. These include the input bandwidth, uniformity and scalability constraints, flexibility and extensibility in the data organization, concurrency and access controls, and a query language.

THE REFERENCE MODEL

We have based the reference model on the concepts and terminology from commercial organizations [3, 4, 5]. By this means we hope to achieve the maximum leverage to be able to incorporate components from a large variety of sources. Within different application domains different sets of components might need to be present, but they would share significant commonality and would be easily replaceable within the same common framework.

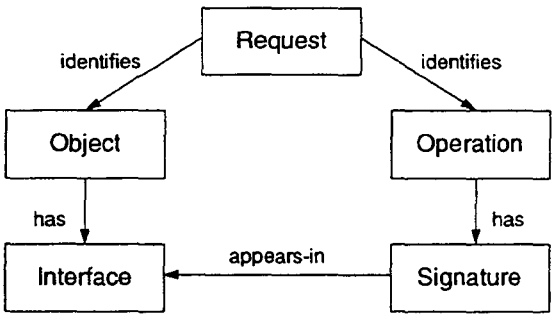


Figure 1. Object Semantics

The reference model is presented from a variety of different viewpoints that reflect the overall complexity of the system and highlight different aspects of it. The information viewpoint concentrates on information modeling, structure and flow. It is primarily based on the Object Management Group Object Model [3]. Figure 1 is a representative diagram from this level.

The computational viewpoint focuses on interactions between the entities identified within the information viewpoint and their exchange of data and control. It is expressed in terms of Object Services as in the OMG's Service Architecture [5].

The engineering viewpoint deals with the communication and support mechanisms; objects outside of the nominal application domain make their appearance here. Much of this viewpoint follows the OMG's CORBA [4]. However, Figure 2 shows an additional abstract mechanism we have identified to make distributed services transparent to client objects [6].

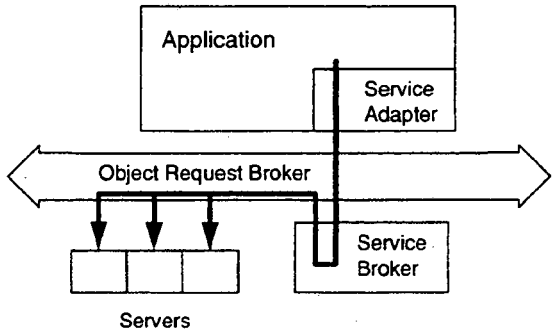


Figure 2: A Brokered Service

The technology viewpoint concentrates on the impact of available technology on components of the complete model. An example of this viewpoint is presented which connects our model with the IEEE Mass Storage Model.

AN IMPLEMENTATION MODEL

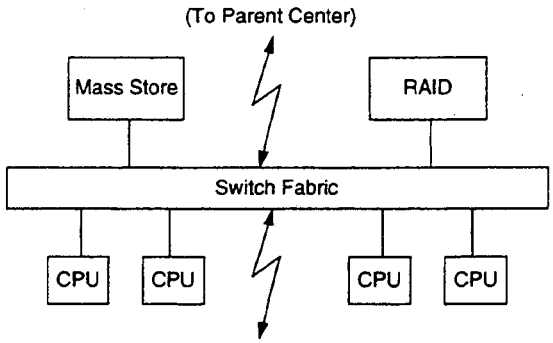


Figure 3. Generic Data Center

The implementation model focuses mainly on hardware aspects of a solution, including capabilities such as network bandwidth and mass store capacities and bandwidth. We present an implementation model made of a geographically dispersed hierarchy of Computing Center Nodes. A typical node is shown in Figure 3.

DESIGN ISSUES

A reference model lacks many of the characteristics of a final implementation that accommodates technological constraints to optimize the available capabilities. For example, the reference model states that data must be moveable amongst a hierarchy of data stores in a manner that optimizes response times to the most frequent access patterns. How best to achieve this caching and migration, and whether to replicate or move data, is a detailed policy and design issue that lies beyond the scope of the reference model. We have identified several such issues that are worthy of more discussion and presented aspects of their impact.

REFERENCES

[1] A. Baden and R. Grossman, *A Model for Computing at the SSC*, Superconducting Super Collider Laboratory Technical Report No. 288, 1990.

[2] C. T. Day *et al.*, *Database Computing in HEP—Progress Report*, Computing in High Energy Physics, 1992.

[3] Object Management Group, *Object Management Architecture Guide, Revision 1.0*, OMG TC Document 90.9.1, 1990.

[4] Object Management Group, *The Common Object Request Broker: Architecture and Specification, Revision 1.1*, OMG TC Document 91.12.1, 1991.

[5] Object Management Group, *Object Services Architecture, Revision 6.0*, OMG TC Document 92.8.4, 1992.

[6] D. Lifka *et al.*, (the PASS Collaboration), *The PASS Project Architectural Model*, to be published.

A Multi-level Object Store and its Application to HEP Data Analysis

Edward May, David Lifka, David Malon, R.L. Grossman, X. Qin, D. Valsamis and W. Xu

Abstract

We present a design and demonstration of a scientific data manager consisting of a low overhead, high performance object store interfaced to a hierarchical storage system. This was done with the framework of the Mark1 testbeds of the PASS project.

Introduction

The work described in this report is part of the PASS project. The motivation and a status report for the PASS project are given in a paper by Quarrie [1]. The basic notion of the present work is to use an Object Oriented (OO) approach to the modeling, storage, and accessing of HEP data. Thus the user should only be concerned with representation of HEP data in-memory (as C++ classes), while the storage and access of the HEP data structures are handled transparently in a multilevel cache based hierarchical storage environment. We use the PTool system [2], which provides an efficient and flexible method of making C++ classes persistent. It is, in effect, an object manager which manages physical collections of objects between memory and disk.

We have extended this model with separate managers to manage the physical collections between disk and tertiary storage devices. This follows the basic design described by the IEEE Mass Storage Model [3].

Ptool: Design And Architecture

This section is based in part upon reference [4]. The physical design of PTool is based upon three concepts: objects, folios, and stores. A {\it folioV} is a contiguous range of virtual memory that is managed by PTool. A folio may contain one or more objects or a piece of a large object. A {\it storeV} is a physical collection of folios. A folio is implemented as a UNIX file.

The PTool Management System consists of several software tools which the user sees through well defined APIs. The most important are the Persistent Object Manager which creates, accesses, and updates persistent objects; and the Persistent Folio Manager which manages folios between virtual memory and local disk, and between local disk and networked or hierarchical storage.

The user of PTool works in a C++ programming environment. The user may declare any C++ class (the objects) as persistent in a full 64bit addressable virtual memory space. The user accesses the objects via a "overloaded" pointer. When this pointer is dereferenced during program execution, the PTool system determines if the object is in the local system real memory. If it is not PTool consults the Persistent Folio Server to determine the location of the object within the multilevel cache storage system, then moving the object through the hierarchy by the fastest means available. For local or NFS-mounted disks this can be memory-mapped I/O, for network attached RAID or tape robots this could be lightweight message-passing protocols to remote servers, etc.

The Mark1 Evaluation Tests

The Mark 1 evaluation tests are an extension of the Mark 0 tests reported at the CHEP'92 meeting [5] designed to evaluate the use of various database like technologies for HEP data analysis. A set of test suites using typical HEP data were constructed for use in a hierarchical storage environment. Testbeds at the scale of 10 GB have been developed as an

initial point to study the scaling of the performance of database technology over the range of 10 GB to 100GB with the later extension to 1TB. The plans included the testing and evaluation of a commercial relational database, a commercial object oriented database, the PTool object manager, and a typical HEP Fortran-based access method.

The ANL testbed was used for both development work with the Sun Fortran and C++ compilers; and for initial tests. It consisted of a SparcStation 2 (4/75) with local SCSI attached disks (1 GB for data storage) and a 5 GB 8mm tape drive. A SGI 4D/35 server was available via 10 Mb/s ethernet for data storage and access via FTP and NFS remote file system mounts.

The SSCL testbed consisted of a Sun 670MP server with 10 GB of SCSI-2 attached disk and IPI3 attached Ampex DD2 tape drive, capable of using the 25GB cartridges.

We obtained a CDF 1989 J/Psi DST data sample in YBOS format stored on 8mm tapes. It contained muon, electron, photon, jet, central drift chamber tracks, and digitized hit information. The YBOS Fortran-based access library was obtained and compiled on the Sun architecture.

- **Data Model**

An object-oriented data model was constructed for the data using the Rumbaugh design notation[6] and an X11-GUI based design tool (OMTool from GE Research, Inc). This tool can output the data model in either SQL or C++ header files, which can be used as the schema for relational or (C++ based) object-oriented databases. In all three cases, the schema produced required some hand editing to fit the specific requirements of the target database.

- **Data Sample**

A group of codes (in Fortran, C and C++) were written to read the CDF YBOS data, and to reorder and populate the test evaluation databases according to the schema obtained for the data model. Due to the cancellation of the SSC project, the building and testing of the commercial relational and OO databases have not been completed by the time of this report. The remainder of this paper will thus describe experience and test results for the YBOS and PTool-based access methods. Three data set sizes were prepared: 805 MB, 1.8 GB, and 6.5 GB in the case of the PTool-based stores. These sizes were set to conveniently fit on local disks, 8mm tapes, and the DD2 tape cartridge.

- **Test Results**

We comment here only briefly on the test results; a full report[4] is in preparation for publication. A comparison of the YBOS and PTool access for typical physics-based queries shows a speedup of approximately four in the wall clock elapsed time. From the technical queries, we note that a study of the effect of accessing objects of increasing complexity (higher multiplicity of instances) shows an elapsed time dependence which is approximately a power law on the complexity with exponent of 1/3. The access to PTool stores which are remotely NFS mounted (via ethernet) are only a few per-cent different from locally mounted disks. Access times to folios stored in a simple hierarchy based upon 8mm or DD2 tape drives, are only limited by the speed of large block transfers from the media to a local disk. These and other results for the OO approach and its implementation in the persistence model provided by PTool within a caching hierarchical storage environment are very encouraging.

Current Activities -- Future Plans

The PTool code system has been moved to the GNU g++ compiler for more portability. A cooperative R\&D program between ANL and IBM has recently resulted in the siting at Argonne of an IBM 128 node SP1 computer and advanced high capacity I/O system for file systems. The I/O system includes a hierarchical storage system based on a HIPPI connected 220 GB RAID, a 6 TB three read-head tape robot for DD2 tape cartridges, and an NSL Unitree hierarchical storage manager. We have extended the PTool code system to work in this environment and are currently

preparing the test suites described above for a detailed evaluation.

During the same time frame a similar R\&D project has sited a 24 node SP1 at Fermilab as part of the Fermilab computing division CAP project. Their strategy for high performance I/O and file systems is different and complementary to that of the ANL facility. It is based on multiple directly node-attached SCSI disks and the use of the IBM Research Vesta Parallel File System. We have begun a series of cooperative implementations and tests with the CAP and D0 team that is exploring the use of the SP1 for D0 DST analysis.

We intend to extend the tests of the PTool-based access methods to larger size data stores at a scale typically of 100 GB on the RAID disk and 1 TB in the tape robot at ANL. In addition, the use of NSL Unitree (and later the HPSS) hierarchical storage manager will be explored. Some of the important issues to be studied are: the effect of varying the caching size to match the physical devices in a more optimal manner; the use of different clustering models of which objects (or sets of objects) to store together in a physical storage volume; techniques for reclustering existing stores to better match typical HEP query strategies.

Acknowledgements

The authors thank A. Gauthier, U. Nixdorf, J. Marstellar of the Superconducting Super Collider Laboratory for their help with the data model design, manipulation of the CDF data, and use of the SSCL testbed.

Bibliography

1. D. Quarrie, "The PASS project, A Status Report," CHEP 94, this publication.

2. R.L. Grossman and X. Qin, "PTool: a low overhead, scalable object manager," Proceedings of Sigmod 94, to appear.

3. R.A. Coyne and H. Hulen, "An Introduction to the Mass Storage Reference Model, Version 5," Proceedings: 12th IEEE Symposium on Mass Storage Systems, S. Colman, ed., IEEE Computer Society Press, 1993.

4. R.L. Grossman X. Qin, Valsamis, W. Xu, D. Lifka, E. May, D. Malon, and L. Price, "The Architecture of a Multi-level Object Store and it Application to the Analysis of High Energy Physics Data," in preparation, May, 1994.

5. C.T. Day, et. al., "Database Computing in HEP---Progress Report", Proceedings of the International Conference on Computing in High Energy Physics '92," CERN Report CERN-92-07, 1992.

6. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, "Object-Oriented Modeling and Design," Prentice Hall, 1991.

Parallel Query Processing For Event Store Data

D. Malon, D. Lifka, E. May, R. Grossman, X. Qin, and W. Xu

Abstract

Enormous data volumes and large, geographically dispersed user communities characterize the next generation of experiments in high energy physics and other scientific disciplines. Parallel processing will be integral to the solution of the information storage and retrieval problems that these experiments will engender. \hfil\vfil\vskip\baselineskip We describe several approaches to parallel query processing that have been implemented in the early stages of the PASS (Petabyte Access and Storage Solutions) project. These have been tested on an object-oriented persistent event store built from Fermilab CDF data, and evaluated on the 128-processor IBM SP-1 at Argonne National Laboratory, as well as on networks of workstations. %\hfil\vfil\vskip\baselineskip %We conclude with a discussion of scalability issues, and with a description %of ongoing parallel query processing research.

Introduction

Responding to queries directed to petabyte-scale scientific databases in large multiuser environments will require significant parallel processing capabilities. The nature of high energy physics data makes effective parallelism possible at a number of levels—individual queries may be parallelized, data servers and file systems may be parallelized, and even straightforward multiuser parallelism is relatively free of degradation due to lock contention since the preponderance of data are accessed by most users in read-only fashion. The following sections describe ongoing research in the PASS project in the area of query parallelization for large object-oriented scientific databases.

The focus of our parallel query processing work to date has been upon replicated query strategies, and upon parallel strategies for caching and migration of physically contiguous units of persistent storage, known as folios, in which collections of persistent objects reside. %Preliminary work on parallel object server strategies will also be described.

Replicated Query Strategies

Replicated query strategies follow the SPMD (single program, multiple data) model of parallel programming. Such a model applies when typical queries have the form ``for each event that satisfies criterion A, return the derived data produced by computation B.'' Such queries can be parallelized readily by replication into queries against disjoint database subsets if events are essentially independent, or if events can be partitioned into essentially independent collections.

Two replicated query strategies have been studied in various guises. In the first, queries are sent to worker processes, who each satisfy the query against their local data and send results back to the master process. In the second, a workload queue is constructed, and workers in turn remove the first workload chunk that they can handle (e.g., for which they have access to the requisite database folios). When a worker completes a chunk of work, it sends results back to the master, and checks the queue for another chunk of work for which it has the resources. The process continues until the queue is empty, or until no worker can handle any of the remaining workload chunks.

Parallel Folio Servers

Work has also been undertaken on a different approach, in which references to nonlocal data generate requests to a folio manager, who arranges delivery of the physical unit of persistent storage (folio) in

which the desired object resides from one or more of a collection of parallel data servers.

Implementations

Trial implementations of these approaches have been tested on the Argonne and Fermilab IBM SP-1 PowerParallel systems, and on heterogeneous networks of UNIX workstations. The underlying database was built from Fermilab CDF data, and follows an object-oriented model formulated as part of the PASS project. An Argonne-enhanced version of PTool64, developed at the University of Illinois at Chicago, was used as the persistence manager. Interprocess communication in the parallel query tests was implemented using the Argonne-developed p4 package. To date we have developed an FTP-based implementation of the parallel folio server strategy, and a mechanism to move database folios directly over socket connections. We have also used IBM's Vesta parallel file system on the Fermilab SP-1 to parallelize delivery of database folios to single-user queries.

Comparisons

These approaches and their various implementations differ in both theoretical and practical ways. Differences include single-user and multi-user speedup, whether user code, database code, or storage system interface code needs to be parallelized, potential for load balancing, adjudication of access to shared data, levels of data communication traffic, parallel work queue management, data caching, and satisfying queries that cannot be subdivided into pieces that can be handled by a single node.

Both replicated query strategies are relatively easy to parallelize, and do not require parallelization of (possibly proprietary) database packages. In our test implementations, parallel queries were constructed by hand from serial queries written in C++, but with a modicum of control at an Object Query Language (OQL) interface, automatic parallelization should be possible in many cases. The first query replication strategy requires only the ability to initiate the parallel queries and to aggregate the returned data; it does, however, rely upon the assumption that parallel workers have access to disjoint portions of the database. The workload queue implementation correctly handles the problem of shared data and implicitly provides dynamic load balancing when data are in fact shared, but the workload queue introduces the potential for a serial bottleneck when the number of parallel processes is large. Such a bottleneck should be avoidable by making workload chunks sufficiently large.

Parallel folio server strategies rely on the ability of the database package to connect to parallel and high-performance data servers, and may result in high data traffic. Parallel speedups derive from parallel data paths to multiple queriers, parallelism implicit in the data server (e.g., the Vesta parallel file system), and the potential for parallel prefetching.

It is likely that in a very large database environment, queries will benefit from parallelism both in the query processing and in the data services. It is also likely that a hybrid strategy that partitions the query workload between the querying node and the database host nodes may prove the most promising. We have done some preliminary theoretical performance analysis along these lines, and we plan to investigate such strategies in the coming year.

Bibliography

1. R.G.G. Cattell, editor, *The Object Database Standard: ODMG--93,* Morgan Kaufmann Publishers, San Mateo, CA, 1994.

2. C.T. Day, R. Grossman, D. Malon, E. May, L.E. Price, D.R. Quarrie et al, "The PASS Project Architectural Model," Draft, January, 1994.

3. R. Grossman, X. Qin, "PTool: A Low Overhead, Scalable Object Manager," Proceedings of SIGMOD 94, to appear.

4. E. Lusk, R. Overbeek, et al, *Portable Programs for Parallel Processors,* Holt, Rinehart, and Winston, Inc., New York, 1987.

Using a Distributed OODBMS as a Source of Events for CDF Physics Analysis[1]

David R. Quarrie and Christopher T. Day[2]
Software Technologies and Applications Group
Information and Computing Sciences Division
Lawrence Berkeley Laboratory

ABSTRACT

We report on a project to use an object-oriented database in conjunction with an implementation of the CORBA distributed object request broker as the source of events for physics analysis using an enhanced version of the CDF Analysis Framework. The goals are to demonstrate that this is feasible with no changes to the user code, to show that significant performance advantages are possible relative to conventional sequential processing, and to gain a better understanding of the access patterns and optimal data organization for complex physics analysis queries.

INTRODUCTION

This project extends earlier work performed by the PASS Collaboration [1], [2] to encompass a much larger data sample (>10GB) and to allow very complex queries to be made on the data and hence be more representative of queries that are typically made during HEP physics analysis. The analysis framework for the CDF experiment [3] at Fermilab has been extended to use an OODBMS as the source of events. In addition, we report on a further enhancement of this framework to support a distributed environment by the incorporation of a CORBA implementation.

The basis for the project is to access events from the database instead of from a file. In particular, only those portions of the event actually required by the analysis are accessed on demand. One advantage of our implementation is that there is no need to create a detailed data model for the CDF event data. Neither is it necessary to rewrite queries in the query language of the database. There already exists a data model, but much of it is embodied within the analysis code rather than within the schema. The query language is the application code itself. The data within an event can be treated as a set of untyped "blobs" by the access routines, being decoded by the existing application code. We need only to provide access mechanisms to the blobs.

THE CDF ANALYSIS FRAMEWORK

The Analysis_Control framework [4] provides a highly modular environment where the user may specify multiple analysis paths involving sequences of analysis modules. The characteristics of the framework that make it suitable for this project are that the data for each event are organized into *banks*, being identified by a name and number; that access to the event data is through a managed region of program memory via a few YBOS [5] routines; that arbitrarily complex queries may be performed by combining user modules into *paths* and that multiple input modules may be linked, one being selected at run-time. The standard input module reads events sequentially from a file, the complete data for each event being read.

1. Supported by the U.S. Department of Energy under Contract No. DE-AC03-76FS00096.

2. Address: LBL MS50B-3238, 1 Cyclotron Road, Berkeley, CA 94720
Email: CTDay@LBL.Gov & DRQuarrie@LBL.Gov

THE EXTENDED FRAMEWORK

The extended framework is illustrated in Fig. 1. New components are shaded and comprise the OODBMS input module and data access layer. The input module main-

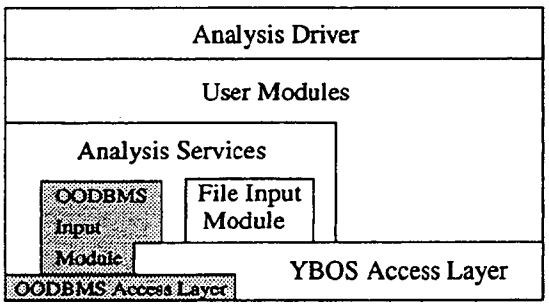


Figure 1. The CDF Analysis Framework

tains the concept of "reading" an event but only advances a cursor to the next event in the sample. The data access layer provides demand-driven access to event data in the database from the YBOS data access layer. A caching scheme is used, whereby a request from a user module to locate a bank first looks to see whether the bank already resides in the managed memory region before the database is interrogated.

In order to simplify the project, the database has been used for input only, any output being to a conventional sequential file. Thus access to the database is in principle read-only.

DATA MODEL AND CLUSTERING

One problem with earlier PASS studies was that associated with converting the YBOS banks into C++ objects. This involved a detailed understanding of the original data model, and a conversion process whereby a function had to be created for each bank type in order to convert the input data into a C++ object. This was a time-consuming process and only 10-12 of the most important ones were converted.

The data model for this project is described in Fig. 2. The *Extent* is a collection object containing all events in the current sample, ordered by event number. The act

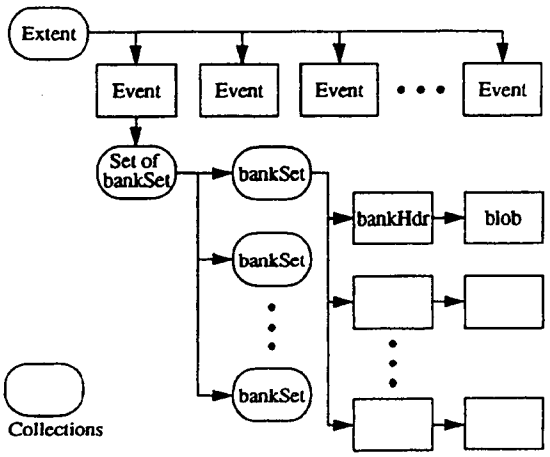


Figure 2. Data Model

of requesting a new event updates a cursor to reference the next *Event* object. Location of a bank within the event involves a query on a collection object containing *bankSet* objects to locate that which corresponds to the given name, followed by a query on that bankSet to locate the *bankHdr* object corresponding to the given bank number. Each bankHdr object contains a reference to an untyped *blob* object that is the bank data.

Conventionally all the data for a single event is held together on a sequential file. In contrast, the chosen clustering scheme locates all banks with the same name in the same database, the data for any particular event being distributed across many databases. In one particular data sample, 71 databases were created, corresponding to the different bank names. If a particular analysis only accesses a small subset of the available classes for the event sample, significant performance improvements should result since only the desired information need be accessed. Furthermore, such clustering of data also takes advantage of caching and pre-fetching mechanisms within the OODBMS itself.

DATABASE LOADING

We have provided a mechanism for taking an input YBOS disk or tape file and loading it into an OODBMS. It is totally data driven, there being no bank-specific code, and can

therefore accommodate arbitrary YBOS input data samples. It operates in a single pass over the input data, no intermediate files being created during the loading process.

Efficient loading of the database is an important aspect of the PASS project, since the loading bandwidth must be comparable to the output bandwidth from the experimental apparatus. Present load times are impacted by the fact that data on the input files are in VAX format and must be converted and that creating indices on collections and inverse relationships between objects significantly degrades the performance. This is still under investigation.

APPLICATION PROGRAMS

Three existing CDF physics analysis programs have been used during this project. They are a generic bank dumping application, a W/Z-finding program having 5 user modules and a J/Psi and B lifetime program. This has 18 user modules, some being executed multiple times with different parameters within the same execution path.

Despite the fact that these programs include code written by a large body of physicists, they have been successfully converted to run with the OODBMS input event source with no changes to the user application code, and therefore exhibit a high degree of conformance to the CDF programming standards. However, several unexpected features of these programs impacted the conversion process.

We had assumed that the applications were read-only and that no attempt would be made to modify the input data, whereas in fact two applications either renamed or deleted input banks. This impacted the caching scheme that had been devised and required the use of aborted update transactions on the database in order to prevent permanent changes. Furthermore, alternative bank location techniques were being used which again required modifications to the caching and access schemes in order to maintain the application programmers simple view of the YBOS data structures.

The above factors meant that, instead of the 3 YBOS access routines that we expected to have to modify, a total of 25 routines eventually needed modification. Most of these were trivial name changes, however.

RELATIVE PERFORMANCE

The performance of the extended analysis programs has so far been comparable to that of the originals. We believe that this disappointing result is due to a number of different factors:

- One application is cpu limited.
- The present data files are highly condensed and almost all banks are accessed for every event. A better test would be to use a more dilute file from upstream in the conventional processing hierarchy.
- The extended applications incur a slight performance penalty since any request to access a bank must first interrogate the local cache before interrogating the database. This penalty is particularly significant if the requested bank does not exist.
- The present data model does not supply indices on the collection objects within an event due to performance difficulties in loading of the databases.
- The extended applications have significantly larger virtual address space requirements because of the OODBMS caching and client-server interactions. This does not impact the cpu time significantly, but does adversely impact the elapsed times through increased paging.

A DISTRIBUTED VERSION

The PASS Architectural Model [6] is based on the OMG [7] distributed object model. We have extended this project to investigate the suitability of this approach by the use of a version of an Object Request Broker (ORB) that supports a variety of hardware platforms and includes an Interface Defini-

tion Language (IDL) to C++ binding. The IDL only specifies the interface for the classes, it does not specify an implementation. Thus the client application is shielded from the implementation details and has a very simple viewpoint.

A simplified view of the relationship between the non-distributed and distributed versions is shown in Figs 3a and 3b.

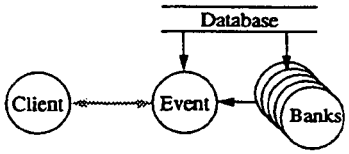


Figure 3a. Non-distributed Implementation

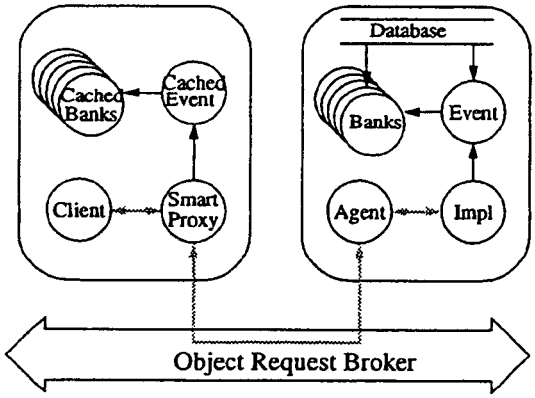


Figure 3b. Distributed Implementation

In the non-distributed implementation, the client, using the interface specified by the IDL file, directly queries the local database. In the distributed implementation, the application is split into two components, the client and server, with the ORB forming the communication medium between them. The client, using the same IDL-specified interfaces, actually queries the smart proxy. If the proxy has the requested data in its local cache, it returns it directly to the client. Otherwise, the proxy forwards the query to the remote agent object which, in turn, queries its local database. The result is returned to the smart proxy, which both caches it and returns it to the client.

CONCLUSIONS

We have successfully demonstrated the integration of an OODBMS as the source of events for the CDF analysis framework. No user code needed to be modified to accommodate this. However, the expected performance improvements have not so far been achieved and work is continuing on this.

The extension of this to a fully distributed system is only partially completed, not all software components not yet being available on the same hardware platforms.

ACKNOWLEDGEMENTS

This work has been performed under the auspices of the PASS project. We would like to acknowledge the fruitful discussions with other project members and to thank the DOE for funding the project.

REFERENCES

[1] A. Baden and R. Grossman, *A Model for Computing at the SSC*, Superconducting Super Collider Laboratory Technical Report No. 288, 1990.

[2] C. T. Day et al., *Database Computing in HEP— Progress Report*, Computing in High Energy Physics, 1992.

[3] CDF Collaboration (F. Abe et al.), *The CDF Detector: An Overview*, Nucl. Instr. Meth. A271 (1988) 387.

[4] M. D. Shapiro and D. R. Quarrie, *A Beginner's Guide to Analysis_Control and Build_Job*, CDF Note No. 384, 1991.

[5] D.R. Quarrie and B Troemel, *YBOS Programmer's Reference Manual*, CDF Note No. 156, 1989.

[6] D. Lifka et al. (the PASS Collaboration), *The PASS Project Architectural Model*, to be published.

[7] Object Management Group, *The Common Object Request Broker: Architecture and Specification, Revision 1.1*, OMG TC Document 91.12.1, 1991.

DATA HANDLING AT CERN FOR LEP AND FUTURE EXPERIMENTS

Richard P. Mount
California Institute of Technology
Pasadena, CA 91125 USA

Abstract

The current status of hardware and software for data handling at CERN is examined with emphasis on the services offered to LEP experiments. Planned or possible near-future developments are presented. Extrapolations from current storage technologies and costs are used to make reasonable, but perhaps conservative, estimates of what will be possible for LHC experiments.

DATA HANDLING AT CERN: CURRENT HARDWARE STATUS

Since the beginning of the LEP programme, data handling at CERN has been dominated by IBM 3480 tape cartridges. The CERN tape vault holds over 200,000 3480 cartridges and a further 200,000 are kept in archive storage in other buildings. The tape drives in the CERN tape vault include about 50 manual 3480/3490 drives plus two IBM 3495 tape robots each equipped with 16 drives and each containing about 18,000 cartridges. One tape robot is equipped with 3490E drives and can store up to 1 Gbyte on each cartridge. The rate of manual tape mounting exceeded 1,000,000 per year in 1993 and is still increasing.

LEP physics analysis requires good access to stored data. As an example consider the measurement of b-quark lifetime using the L3 1992 data. This measurement required:

- 1700 cartridges of $q\bar{q}$ data. (These cartridges contained the full output of the reconstruction, about 250 kbytes/event. Greater foresight in writing the 89 $q\bar{q}$ data summary tapes would have allowed these to have been used instead.)
- 252 cartridges of e^+e^- and $\mu^+\mu^-$ data.
- 26 cartridges of $\tau^+\tau^-$ data.
- 304 cartridges of $q\bar{q}$ Monte Carlo.
- 36 cartridges of $b\bar{b}$ Monte Carlo.

Within the b-lifetime analysis there were several iterative calibration steps requiring repeated reading of large numbers of tapes, and the whole analysis was repeated several times as calibrations and understanding of the data improved.

The 3480/3490-based systems at CERN have proved reliable, but physics analysis reading hundreds or thousands

of tapes is inevitably slow and inflexible. Over the last year, CERN and the LEP collaborations have accelerated their efforts to provide sufficient disk space to support a large fraction of physics analysis. As of this conference, while no experiment is satisfied, several would admit that revolutionary improvements have occurred.

The focal point for these revolutionary improvements has been the SHIFT system[1] which has grown from a CN-OPAL project in 1991 to being the principal resource for data-intensive physics analysis at CERN[1]. As shown in table 1, the CPU and disk components of shift are rigidly allocated to (or bought by) individual experiments. These components are complemented by shared networks, workstations and tape drives which give access to 3480 cartridge tapes. Figure 1 shows the components of the SHIFT system from the viewpoint of the L3 experiment.

The SHIFT system has access to five RS6000-370s connected by IBM block multiplexor interfaces to eight manual and eight robotic tape drives. Most of the remaining tape drives are still connected to mainframe systems. As mainframe services are run down, tape drives will be progressively transferred to the SHIFT system using additional IBM RS6000 workstations with mainframe-compatible block multiplexor and ESCON channel interfaces. Although the days of the mainframes are numbered, the importance of 'mainframe-quality' tape equipment remains.

In its original form, the SHIFT project used Ultranet to interconnect

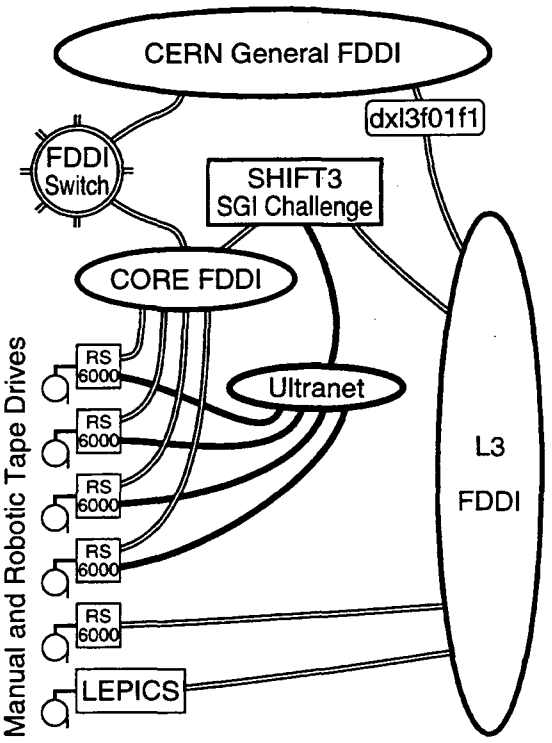


Figure 1: The SHIFT system and its access to tapes and networks as viewed by the L3 experiment

separate machines providing CPU-server, disk-server and tape-server functions. Such decoupling remains desirable in principle, but the performance of a shared Ultranet (or FDDI) is no longer adequate for the total CPU-disk traffic of a LEP experiment. The L3 solution to this problem is to attempt to confine all data-intensive analysis to a single SGI-Challenge system offering a total CPU-disk bandwidth of hundreds of Mbytes/s. The current configuration of the L3 system SHIFT3 is shown in figure 2. The 300 Gbytes of disk space is already insufficient since the total volume of compressed summary files for data and the relevant Monte Carlo is now over 600 Gbytes. It is hoped that the system will continue to function efficiently when expanded in CPU power

[1]The Central Simulation Facility (CSF) complements SHIFT for CPU-intensive work.

Table 1: SHIFT Status; equipment installed April 1994

CERN Group	Configuration	CPU (CU*)	Disk Gbytes
OPAL	SGI Challenge 4-cpu + 8-cpu; 2 × SGI 340S 4-cpu	290	590
L3	SGI Challenge 10-cpu	200	300
ALEPH	SGI Challenge 4-cpu; 8 × DEC 9000-400	216	200
DELPHI	2 × HP 9000/735	52	200
ATLAS	HP 9000/735	26	23
CMS	HP 9000/735	26	23
SMC	SUN SPARCserver10, 4/630	22	4
CPLEAR	DEC 3000-300AXP, 500AXP	29	10
CHORUS	IBM RS6000-370	15	15
NOMAD	DEC 3000-500 AXP	19	15

* CU=Cern Unit, approximately 4 SPECint

and disk space by a factor of three.

DATA HANDLING AT CERN: SOFTWARE STATUS AND PLANS

This section will concentrate on software[2] giving access to data which may be resident on cartridge tapes. The principal components of CERN data handling software are:

FATMEN, which maintains a database of datasets with such information as format, creator process, possible multiple locations, name and volume-id at each location. FATMEN stores much more information about datasets than is stored in the I-nodes of a Unix file system.

[2]Software, such as RFIO, which offers efficient network file access, is also important in the SHIFT environment.

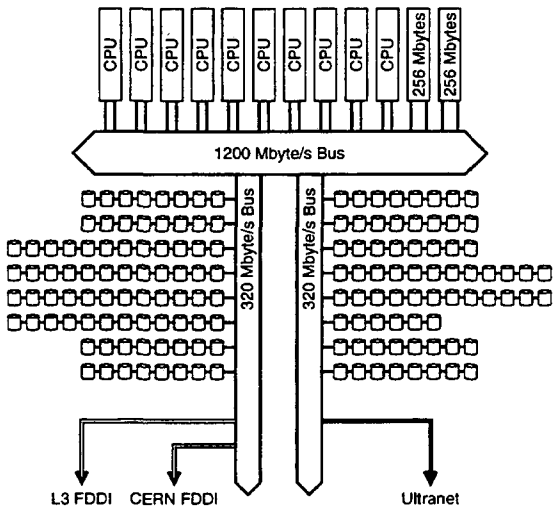


Figure 2: The SHIFT3 SGI Challenge system showing its bus structure. Disks are connected via sixteen 20-Mbyte/s SCSI buses

STAGER, which copies the contents of tapes or tape files to and from disk. STAGER manages disk pools and acts sensibly in response to concurrent requests for the same tape.

TMS, the tape management system records the ownership of tapes, whether a tape is used or free, where a tape is located and whether a tape is already in use (perhaps on another machine).

The FATMEN, STAGER plus TMS system provides reasonably transparent semi-automatic storage management, particularly for users of Unix systems where it is possible for file names and FATMEN names to be identical. However, migration of datasets to tape is not automatic and recall of datasets is only transparent when they are accessed via FATMEN-based utilities.

There has long been an acknowledged need at CERN for a hierarchical storage management system suitable for a wide range of files, from source code in home directories to large data summary files. Neither proprietary systems limiting the choice of hardware, nor home-made solutions beyond the existing tools, have been considered attractive. The Open Storage Management software from Lachman is now under very serious consideration and may be given a serious trial by implementing a pilot OSM system serving the PIAF Parallel Interactive Analysis Facility. The Lachman software has three components: Transmigrator, a transparent algorithm-based migrator which adds to, but does not change, Unix Kernel subroutines; Conservator, a 'bitfile server' which stores and retrieves files; Mediator, an interface package which can be adapted for any robot, jukebox or new mass-storage device. All the OSM code is available as source code only, ensuring that it can be readily adapted to work with any hardware configuration. A key architectural feature of the software is that it is designed to work with a physically distributed ensemble of heterogeneous mass-storage devices.

EXTRAPOLATIONS TO THE LHC

There is general acceptance that physics at the LHC will involve storage and access to Petabytes of data. While I do not disagree with current estimates of LHC data volumes, it should be stressed that LHC physics should drive us to store and get access to as much data as we can at tolerable cost. A quote from the ATLAS Letter of Intent — "whether or not the ... trigger criteria would also be efficient for any unexpected process is, of course, impossible to estimate" — confirms the value an intelligent widening of trigger acceptance if the data-handling load is tolerable. The relevant question is thus "how much will we be able to store and access".

Magnetic random-access and sequential-access storage continues to increase in density and decrease in price. Today's 'commodity' 2-Gbyte disks store 0.35 bits per micron^2 but densities 10–20 times greater are achieved in development laboratories. There is no clear limit to this progress. Magnetic sequential storage currently achieves densities in the range 0.025 bits per micron^2 (DEC TZ87) to 0.07 bits per micron^2 (Ampex DD2). Again, products under development encourage the belief that progress will continue through the LHC era. Op-

tical storage seems a dead end. Almost all current devices reach storage densities close to 1 bit per micron^2, limited primarily by the wavelength of light. Shorter wavelength lasers could bring some improvement, but only a totally new technology which could record in a volume rather than on a surface would give optical storage a serious future.

While revolutionary advances would be most welcome, today's plans for LHC computing must be based on extrapolations from current progress with existing technologies. My personal guess is that the smooth evolution of existing magnetic technologies will result in disk space costing \$30 to \$70 per Gbyte in 2003. Current (LEP) experiments are aiming for 1 Tbyte disk systems for their principal analysis facilities before the end of 1994. In 2003 LHC experiments will be installing systems in the range 15 to 100 Tbytes. Sequential access magnetic storage will use cartridges holding 1Tbyte or more, making 10 Pbyte robotic systems commonplace but not cheap. Media costs for a 10 Pbyte system are unlikely to be less than \$1M. State-of-the-art storage systems will almost certainly be 'closed shop', mainly due to the impossibility of standardisation of media and drives in a continuously evolving marketplace. Sadly (perhaps) The days of the 2,400-foot tape are gone forever.

We can be confident of continuous progress in storage hardware and can even hope for revolutionary advances. Can we expect progress in storage software? Do we even need it? A strong case can be made that physics analysis based on access to data files, complemented by a 'FATMEN-like' catalogue to supplement the information stored in the UNIX I-nodes, and supported by an efficient hierarchical storage manager, would be a perfectly acceptable solution for LHC physics, just as it is a perfectly acceptable solution now. Such an environment would imply progress, but no revolution. Revolutionary improvements in physics analysis may be possible by exploiting the database technology of 2003 to handle Petabytes of data. If this becomes possible, it will be mainly through advances motivated by commercial applications far from HEP. Nevertheless HEP should exploit all opportunities to explore and advance this technology.

REFERENCES

1. J-P. Baud *et al.*, Proceedings of the International Conference on Computing in High Energy Physics '91; editors Y. Watase and F. Abe, Universal Academy Press Inc., Tokyo, 1991.

Session 5
Software Methodologies

HYPERDEV: HYPERTEXT TOOL TO SUPPORT OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Carmen Maidantchik
Federal University of Rio de Janeiro - Brazil
CERN - Switzerland
and
Maria Celia Perillo Isaac
Institut National de Physique Nucleaire et de Physique des Particules - France

Abstract

We propose a software tool, based on hypertext techniques, to support the object-oriented development of scientific applications. Within HyperDev, all kinds of software information such as plain text, formatted text, graphics and code are connected through links allowing for different views of the same object and, consequently, achieving a better understanding of the software components.

INTRODUCTION

HEP experiments make extensive use of large and complex software that extracts physics from information generated by the various detectors. All these data are manipulated by a large number of users. The codified physics knowledge must be correct, quickly developed and maintainable.

Object oriented techniques promise to simplify and speed up the development of software systems. The benefits of an o-o software construction, as improved system structure, maintainability and reuse, are evident. The start of an o-o development process, presented in almost all methods, is the identification of the entities of a problem that are then, formalized in objects and attributes. The misunderstanding of the system components can grow from phase to phase and can lead to either an inappropriate computer solution or to difficult maintenance.

The main objectives of the HyperDev are to promote better understanding of the software requirements, to avoid postponing decisions, to allow for the identification of the essential software features, and to be used as a base for maintenance and future extensions. These aims are reached by hypertext links among different abstract levels of the system components and by managemental features.

METHODOLOGY

The methodology under HyperDev is an extension of the one proposed by Rumbaugh [1]. The software information are organized in four sets: (1) *problem domain*, summarization of the problem to be solved and the role of the system in the environment; (2) *software system*, establishment of the services that the system should provide, identification of the constraints under which the system must operate; (3) *environment*, description of the hardware and its interfaces where the

system is to be implemented, identification of the end users and their current tasks; (4) *development process*, recording of the decisions that were taken or that have to be taken, description of the development strategy and the acceptance test plan, foreseeable modifications and enhancements.

Each of these sets is split in other sets of information, generating a hierarchical structure of the software development process. The user can have either an overview or a detailed view of the software object navigating in the hyperdocument up or down, respectively.

Technical terms, presented in any page of the hyperdocument, are linked to its description that is placed in the *Glossary of Terms*.

TOOLS

HyperDev uses the WorldWideWeb (W3) [2] that merges the techniques of networked information and hypertext to make an easy but powerful global information system.

HyperDev consists of a set of forms and a set of hypertext and integrating tools. The forms guide the steps to be performed during a software life-cycle. The figure 1 presents the form to describe the Problem Statement.

The hypertext tools are: an *editor*, that presents the forms, a *browser*, to visualize the resulting pages of the hyperdocument, *contents*, that presents a hierarchical list of the hyperdocument pages and the *glossary*, with entries to describe the technical terms.

The integrating tools allow for the insertion of the results from the various o-o

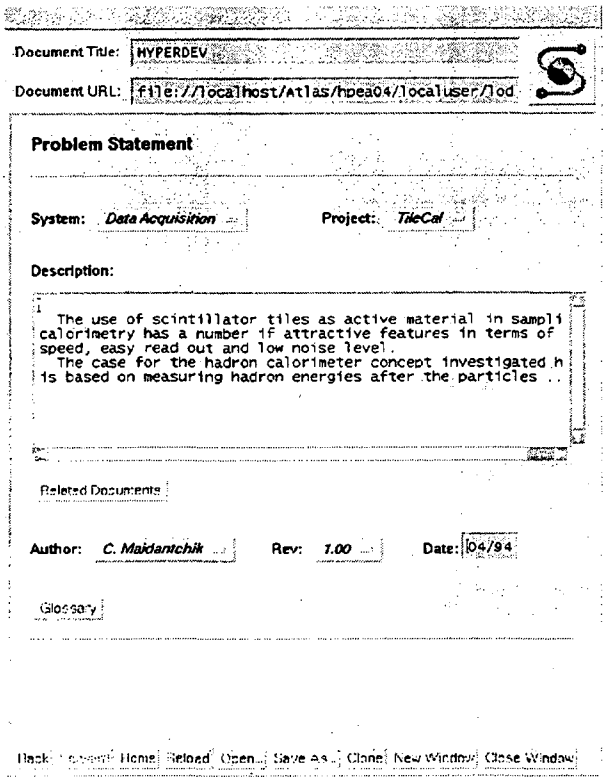


Figure 1. Problem Statement HyperDev form

CASE tools as graphics and formatted or non-formatted text. HyperDev offers conversion tools to generate HTML (*HyperTextMarkup Language*) files and scripts to convert image files into PostScript files.

HOW HYPERDEV WORKS

The user summarizes the problem to be solved and the role of the system in the environment by filling up the "Problem Statement" page. The tool also offers guidelines to describe important requirements and the essential information of the system to be developed such as: the services that the system should provide, constraints under which the system must operate and interfaces with other systems.

The analysis and design can be generated by using any CASE tool. The resulting models and related documenta-

tion are integrated with the requirements definition through links. During these steps, the developer can store incomplete definitions, taken decisions, the solution strategy, foreseeable modifications, enhancements and generic information in the "Development Process" pages that organizes data according to the subject.

To support the implementation phase, the tool provides a documentation scheme in order to describe each piece of code in a standard way. The developer can define a programming style and test plans, both supported by the tool.

During the maintenance phase, the implementation of an object and related information - code documentation, diagrams that represents its design and analysis and the problem statement - can be easily updated.

To validate its efficiency and applicability, HyperDev was chosen as the support tool for the development of a data acquisition system for one of the Atlas detectors, the Tile Calorimeter. The description of the problem, proposed solution, state transition diagram, documentation and development strategy were registered into hypertext pages.

CONCLUSIONS

The planning, design, development and implementation of computer programs represent a considerable investment of human and automated resources. To maximize the return of this investment and to reach a cost effective operation and maintenance, sufficient documentation is needed at each stage of the software development life cycle [3].

Documentation must provide enough information to support the effective design, management, operation, maintenance and transferablity and, to facilitate the interchange of information. It is necessary in order to store technical information and moreover allows for coordination of later development, applications and modifications, being also essential for the understanding between developers and users.

HyperDev, presently in its development phase, can be defined as a set of management features that by integrating the results of each phase of the software construction process, minimizes the misunderstanding of the true meaning of the objects. Within this environment, developers are responsible only for tasks that require intelligence and computers provide guidance, records data and are responsible for manageable and ordinary tasks.

ACKNOWLEDGEMENTS

This work has been supported by CERN/Atlas Experiment and by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brazil). We thank Peter Jenni and Zieli Thomé for their continuous support.

REFERENCES

1. J. Rumbaugh, "Object-oriented modelling and design" Prentice-Hall, Inc., 1991.
2. T.J. Berners-Lee et. al., "World-Wide Web: The Information Universe", "Electronic Networking: Research, Applications and Policy", Meckler Publishing, Westport, CT, USA, 1992.
3. I. Sommerville, "Software Enginnering", Addson-Wesley Publishing Company, Inc., 1992.

OBJECT ORIENTED SOFTWARE DEVELOPMENT IN THE ATLAS COLLABORATION

Arthur Schaffer
Laboratoire de l'Accélérateur Linéaire
Université de Paris Sud
91405 ORSAY, France

Abstract

For more than a year a group within the Atlas Collaboration has been investigating the possibilities of the application of object oriented methodology and program development to the software of Atlas. Recently this group has been joined by members of the CMS Collaboration in the submission of a proposal to the DRDC at CERN to find a common solution for the software development environment for LHC. This talk will discuss the progress achieved so far and the future perspective.

Introduction

Keeping in step with the general trend in high energy physics, the LHC experiments will be larger and more complex than those at present day accelerators. They will involve a larger number of physicists, and almost certainly have a longer experiment lifetime. To meet these challenges it is crucial that all elements of an experiment, including the software, be designed, built and maintained to the highest engineering standards.

To put this into perspective, the requirements in terms of computing power and data volume are several orders of magnitude higher than that for present day experiments: there is an expected rate of 100-1000 event/sec (with 0.5-1 MB/ev) into the event reconstruction and filter farm, which is the final level of the trigger. In order to maintain coherency of the software effort, the filter farm should represent the beginning of what has traditionally been the offline domain where the analysis algorithms are run. This is only part of the large software effort which will be required for each LHC experiment, but it makes clear the need of an engineering approach to coordinate the development effort and maintain the software quality.

The adoption of a methodological, or software engineering, approach has been more or less successfully applied in a number of large high energy physics experiments. The method of choice for experiments begun in the 1980s has typically been 'Structured Analysis - Structured Design' (SASD) [1]. Since this time the object oriented paradigm has been spreading throughout the computing industry, and methodologies, with support from CASE tools, have arisen to support this paradigm.

For more than a year, a group from

the Atlas collaboration has been investigating the object oriented approach to software development. Emphasis has been put on learning to think and program in an object oriented manner, instead of the algorithmically oriented style with which most high energy physicists are familiar. In March 1994 this group, joined by members of the CMS collaboration, submitted a research proposal to the Detector Research and Development Committee for LHC (DRDC) [2] to evaluate the viability of the object oriented approach and look for a common solution for LHC. This paper describes the goals, the context of the problem and the questions being evaluated within this research project.

Goals of Research Proposal

The proposal submitted to the DRDC is expected to be approved at the beginning of June 1994. During the next two years the group hopes to be able to:

i) Establish a well structured software development environment to write the reconstruction software for LHC, trying to do this in an object oriented way.

ii) Adapt the use of an Object Oriented methodology to the high energy physics environment:

   - understand better the components of a spiral software lifecycle:
     - User requirements
     - Analysis
     - Design
     - Implementation
   - learn modeling techniques for the analysis and design of software before the coding is done
   - use CASE tools to support the development

iii) Be able to recommend, or not, the object oriented approach to our colleagues

The Benefits of a Method

A methodological approach to software development should allow a systematic means to step through analysis, design, and implementation phases. The lifecycle can be described as a spiral where the different phases are gone through in an iterative manner and the software increases in complexity on each pass. Using a method, more time is spent in the early phases of analysis and design using different modeling techniques, and code takes longer to appear. The benefits of using a method include the following:

i) *Requirements checking:* The analysis process generates searching questions about the requirements and can clarify ambiguities and catch omissions.

ii) *Clear concepts:* Analysis leads to a better understanding of the problem domain. This makes the models built more robust against changes in the requirements. The resulting models can also be used elsewhere.

iii) *Less redesign work:* The models and notations used in analysis and design allow problems to be explored at different levels of abstraction, before the details of the implementation intrude. Alternatives can

be explored before committing to a single design.

iv) *Better factorising of design work:* Doing analysis and design allows an easier decomposition of the problem into independent parts to be worked on by different groups of a development team.

v) *Improved communication between developers:* The models and notations allow for communication in a precise but abstract way.

vi) *Less effort needed on maintenance:* The LHC software will have a long lifetime and will be maintained and improved by people other than the original developers. The analysis and design documents will provide the abstract framework to allow new people to understand the code.

Why Object-Orientation?

The object-oriented approach provides several features and techniques that reduce the complexity inherent in the software development process. These aspects can help to improve software quality and promote software the reuse necessary for the long life-cycle of the LHC era. These are briefly reviewed:

i) *Real-world Modeling:* Particle physicists have been working forever with events, tracks, vertices, etc. These are embodied directly in software in terms of classes.

ii) *Encapsulation:* Clients, or users, depend on the class interface, i.e. *what* can be done by a class, and are shielded from the modification and evolution of its implementation and from its interaction with the rest of the system. This simplifies a user's view of the system and focuses effort on defining the information and behavior which is important to the user.

iii) *Inheritance:* supports incremental refinement of data types, allowing behavior not foreseen in the original development. In contrast to the subroutine approach where generality is obtained through a high degree of parameterization, which must be foreseen at the beginning, inheritance supports generality through simplification: the more general components have less specific behavior.

iv) *Polymorphism:* is the crucial element which obviates the need to modify the calling environment for new classes related by inheritance.

iv) *Parametric or generic types:* allow "container" classes, eg. List, Vector, Set to be type-safe, i.e. the compiler checks that a set of tracks contains only tracks, or classes which inherit from tracks.

The main difficulty with the object-oriented paradigm within the context of the high energy physics community is that it demands a rethinking of the way in which one approaches writing software in that one thinks in terms of message passing between objects. This is in part the motivation for our research project.

Outline of Program

Although there has been some success in applying software engineering tech-

niques to the most recent experiments, a large fraction of the people interested in constructing the software for LHC do not have much experience in this domain. In fact, most physicists have little formal training in software techniques. Typically, they learn programming 'on the job'. An important aspect of the early part of this project is to obtain the necessary training in software engineering, object-oriented programming and the use of CASE tools.

A true improvement in the level of competance of high energy physicists in these areas will only arise through the work on serious prototypes. The Atlas and CMS subgroups will each work on their respective prototypes. Initially, these prototypes will reproduce the inner tracker reconstruction code which already exists in fortran.

Different members of the present proposal have already been active in various areas. During the past year, the group within Atlas has been studying modeling techniques, for instance many have followed a course on Object-Oriented Modeling and Design (OMT) [3], and have gone through a number of models for parts of the inner tracker reconstruction. As well, a Japanese subgroup have been applying OMT to re-engineer GEANT [4] before they joined the research proposal. Certainly, much will be learned from the efforts invested in GISMO [5].

In parallel with the primary prototype work, various methodologies and their corresponding CASE tools will be evaluated. We are now in a fairly active period where methodologies are appearing which give a more complete coverage of the software cycle. Correspondingly, tools are arriving which implement more fully the different models used by a methodology and provide cross checks between the models and the code.

An important aspect of the distributed computing environment towards which we are evolving is the ability to allow for objects to communicate across platforms. The Object Management Group (OMG) [6], formed by nearly 400 of the leading software and hardware vendors worldwide, is setting standards for object interoperability. In particular, their Common Object Request Broker (CORBA) standard defines the backbone for such platform independence where the broker acts as the agent which handles the messages between client and server objects. The OMG is continuing its work in the definition of standard services for object interaction, eg. creation/deletion, naming, storage of objects. Commercial products are now beginning to appear on the market.

Finally, the questions of input/output and data access and storage are of utmost importance for LHC. Interesting ideas are coming out of the Petabyte Access and Storage Solution (PASS) R&D research project [7]. This group is sketching out an architecture for accessing data using a database approach where there is a hierarchical set of datastores ranging from the main datastore at the experiment to the physicist's desktop.

Current Status

There are currently nearly 50 people from 20 institutes from Europe, Japan, and the USA participating in the object-orient research project. Information concening the group's activities can be

found on WWW:

http : //www.cern.ch/OORD/Home_oord.html

Typically there will be three general meetings per year and a similar number of workshops to concentrate, for example, on the different prototypes of the Atlas and CMS subgroups. There are also biweekly teleconference meetings using mbone. People who are interested in working and/or have experience are certainly welcome to join.

REFERENCES

1. For a description of the software engineering methods used in ALEPH see: J. Knobloch, "Methods of Software Development for the ALEPH Experiment", Proc. of New Computing Techniques in Physics Research, Lyon, France (Editions CNRS 1990), or G. Kellner, "Development of Software in ALEPH using Structured Techniques", Computer Physics Comm. 45 (1987) 229.
2. S. O'Neale, et al., "Object Oriented Approach to Software Development for LHC Experiments", CERN/DRDC/94-9, DRDC/P55.
3. J. Rumbaugh, et al., "Object-Oriented Modeling and Design", Prentice Hall, 1991.
4. See K. Amako, "Object-Oriented Analysis and Design of a GEANT Based Detector Simulator", this conference.
5. See A. Breakstone, "Experience Building Object-Oriented Systems", this conference.
6. There are a number of publications which can be obtained from the Object Management Group, Framingham Corporate Center, 492 Old Connecticut Path, Framingham MA, 01701 USA, email: omg@omg.org, anonymous ftp: omg.org
7. See "The PASS Project Architectural Model", and the serveral presentations at this conference.

Omo - A Tk/tcl based object modelling tool to support Eiffel

S.M. Fisher <*S.M.Fisher@rl.ac.uk*>
Rutherford Appleton Laboratory, Chilton, Didcot, OXON, OX11 0QX, England

D.J.Candlin <*D.J.Candlin@ed.ac.uk*>
Dept. of Physics, Edinburgh University, Edinburgh, Scotland

Abstract

Using free, readily available components, we have constructed omo, a tool for designing and maintaining Eiffel code. Omo is at the same time a design tool and a reverse engineering tool which is able to accept any valid Eiffel code and to generate Eiffel code. By building our own tool we have been able to try variants of the notation and of the model. As much of the code was written in a non-OO language, we have learned to appreciate still more the benefits of OO.

INTRODUCTION

Existing CASE tools for OO development are oriented in the concepts they support towards C++. For example *public*, *private* and *protected* rather than the more general approach which Eiffel[1] takes to visibility.

There is a trend towards customisable tools, with parameterised code generation. Nevertheless the information which is stored must be translatable to the programming language, and to achieve reverse engineering it must be possible to derive the concepts from the code.

Omo is intended to be a tool to be used when it seems useful, rather than a strait-jacket. We want the user to be able to start from nothing, or from some Eiffel files and maybe some files defining the layout of the diagrams. He can work without the tool for a period while debugging algorithms and be able to reintegrate the modified code later. To achieve these aims we hold as much information as possible in standard Eiffel source files. Special omo view-files control the layout of a group of classes on a diagram.

Omo has a consistent, simple user interface with a complete help system and a single source of documentation. See URL http://hepunx.rl.ac.uk/packages/omo.html on WWW for information on getting a copy of omo.

SOME DETAILS OF CONSTRUCTION

The parser for Eiffel input files is constructed using some existing Eiffel classes[2]. This is used both to read whole files and for checking small sections of code, for example a single *feature* (generalisation of a C++ *method*) which the user has edited and wishes to have checked for correct syntax.

Tcl[3] is an interpreted language with a very simple syntax. Tcl is designed such that it is easy to add your own commands written in C. To interface the parser, written in Eiffel, some code inspired by tclish[4] was written which then enabled us to issue Tcl commands to the parser. A GUI was constructed using Tk,

the other part of Tk/tcl which is a set of widgets able to issue Tcl commands.

We chose to use the Eiffel data structures only for holding the class being parsed, while the bulk of the information is stored in Tcl structures. We have implemented Entity Relationship (ER) style structures in Tcl, using a small integer to identify objects of a particular type (a surrogate key).

THE OMO METHODOLOGY

A few windows will be described to give an idea of how omo looks to a user, and to show the characteristics of the methodology which omo currently supports. The example considered is a KEYSTORE which is able to recognise keywords and values.

Each view shows a number of classes with their inheritance and relationships. Fig. 1 shows a part of the view which defines the KEYTEST class. This is simply a class for testing KEYSTORE, which is also related (the purple line[1]) to a WORD_GRABBER. It inherits (the red line) from TRAVERSABLE. The hatching on traversable shows that it is a *deferred* (or *abstract* in C++ terminology) class.

KEYTEST is in dark blue, meaning that it is being defined on this view. KEYSTORE is in light blue meaning that it is defined elsewhere. It will be in light blue in all views except one, its *defining view*, where it is in dark blue. WORD_GRABBER is in an intermediate blue, meaning that it has no defining view.

Relationships in the ER model are symmetric. Omo supports directed relationships because OO is essentially asymmetric. When an object tells another object to do something there is no reason for the latter object to know anything about the object giving the instruction. Eiffel supports one way relationships by allowing an object to have a reference to another object (a feature which is just another object) or to a collection of other objects. Omo denotes a relationship by a broad purple arrow with the name of the feature which will be used to represent it at one end, and the name of the collection class (if any) at the other end.

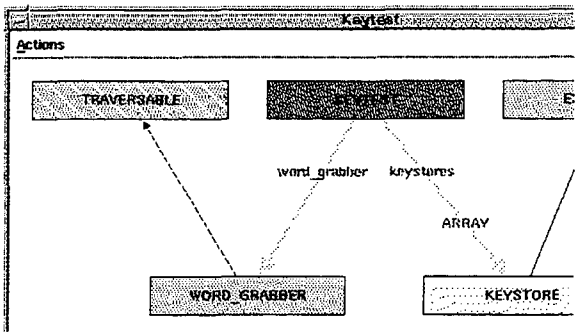


Fig. 1. An omo view

In a good design, an object receives an instruction and then talks to the objects it needs to do the job. It does not generally pass out references to these objects but hides them to simplify the task of changing the system in future. To provide this protection, relationship features should not be *exported* to all classes (i.e. not *public* to C++ people), though some relationships should be exported to cooperating classes. The cases are distinguished on the diagram by the weight of the line: a light line (as in Fig. 1) when the relationships are not exported, a heavy line if they are exported freely, and a line of intermediate weight if the export is restricted to certain classes.

If you click on a class a menu will pop up. This will offer a number of facilities

[1] the reader will have to imagine the colours

for modifying the diagram, to inspect the code, to edit the code with the editor of your choice or to see the dynamic model for that class. For example if you ask to inspect the code of the KEYTEST class the window shown partially in Fig. 2 will appear.

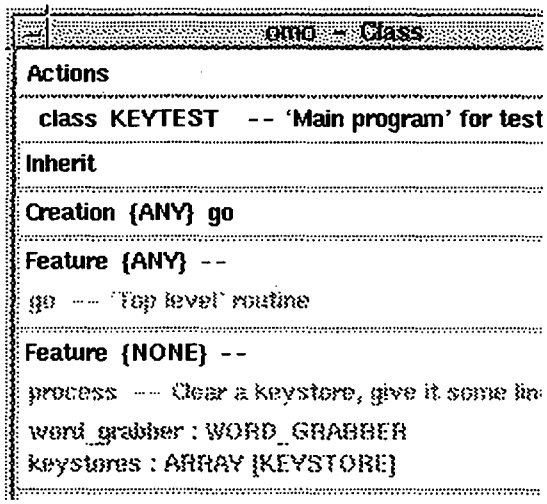


Fig. 2. An omo class - the code display

The relationships are in purple and the inheritances, if there were any, would be in red, corresponding to the colours of the lines on the view window. If you add a new inheritance or relationship on the view window, it will immediately appear here as well. Almost all the text on this window will respond to mouse clicks. The action menu includes facilities to add feature blocks and to navigate to other parts of the model.

If you click on the name of a feature a window will open to give a display of the body of that feature. You can then edit that text in the window, where the bindings for the widget allow many emacs-consistent character sequences. After you edit it, you can parse the code which will either give an error message and a highlighted line where the error was first spotted, or if it is syntactically correct the text will be reformatted to look pleasing.

Either from the class window's action menu or from the menu of one of the classes on a view, one can select the dynamic model. Our model is similar to that of Harel[5] but allows multiple inheritance. This is not shown by drawing one state within another, but by using inheritance arrows between states, just as between classes on the view diagram. The diagram shows transitions between states with triggers, guards and actions. A menu button will generate Eiffel code from a diagram.

CONCLUSIONS

It is not hard to build your own CASE tools, provided you use the right bricks.

Tk/tcl is good — but not Object Oriented. Having been bitten by the OO bug, we found it very clumsy to use old fashioned functions with separate data. It would almost certainly be better to code an application of this size in an OO extension of tcl.

REFERENCES

1. B. Meyer, *Eiffel: The Language*, Prentice Hall, 1992.
2. David Morgan, */pub/eiffel/eiffel-3/sig/tools-v1.3.tar.Z* ftp from ftp.informatik.uni-stuttgart.de.
3. J.K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.
4. S. Herrmann, *tclish*, <stephan@cs.tu-berlin.de>.
5. D. Harel, *Statecharts: a visual formalism for complex systems*, Science of Computer Programming, 8 (1987) 231-274.

The Distributed Development Environment for SDSS Software *

Eileen Berman, Vijay Gurbani, Bryan Mackinnon, Heidi Newberg, Tom Nicinski
Don Petravick, Ruth Pordes, Gary Sergey, and Chris Stoughton
Computing Division
Fermi National Accelerator Laboratory
Batavia IL, 60510

Robert Lupton
Department of Astrophysical Sciences
Princeton University
Princeton, NJ

Abstract

We present an integrated science software development environment, code maintenance and support system for the Sloan Digital Sky Survey (SDSS) now being actively used throughout the collaboration.

INTRODUCTION

The SDSS is a collaborative effort between Fermi National Accelerator Laboratory (Fermilab), the U. of Chicago, Princeton University, the Institute for Advanced Study (at Princeton), The John Hopkins University, U. of Washington, the U.S. Naval Observatory and the Japan Promotion Group. Its main results will be an imaging survey of $10^4 deg^2$ and a red shift spectroscopic survey of 10^6 galaxies and 10^5 quasars producing approximately $1.2x10^{13}$ bytes of data over the 5 year running period (1995 - 2000). This will produce a three dimensional map of the Universe.

Software Development Environment

The Fermilab Computing Division supports a set of standard tools and software for development of the SDSS data processing and analysis code. Survey standards are documented for : portable software environments, ANSI C, C++, FORTRAN, writing scripts, accessing survey databases, *make* methodologies, product structure and source code use. These standards were created to aid in the development and maintenance of a unified software system written by many people at different institutions. The goal of the standards is to meet the requirements of the survey and collaboration, be sufficiently benign as to be universally adopted and followed, be maintained, supported and extended with the available resources, and be expected to allow the needed evolution as the survey progresses.

In order to help accomplish the above goals, the following set of standard development tools were chosen.

RCVS

*Sponsored by DOE Contract number DE-AC02-76CH03000 and the Alfred P. Sloan Foundation.

Remote extension of Concurrent Version System (RCVS) was written by Terry Hung at SLAC[3] and is built on top of CVS (written by Per Cederqvist[4]). It provides for source code management across a wide-area network.

In general RCVS has supplied adequate functionality to support concurrent software development by multiple collaborators at physically distinct sites. We use RCVS for maintenance of all software used by SDSS (which includes approximately 1 million lines of code). Through RCVS any collaborator may obtain a copy of any SDSS software product. Access restrictions limit those who may include their changes back into the software. All modifications made to the software are logged within RCVS. RCVS supports a complete revision control system with script invocation on most of the RCVS commands, tags and branches.

Most of our users operate effectively using only a few commands, and consult with an expert when necessary. RCVS is a complex system and as such has a fairly steep learning curve when trying to understand it at a deeper level (i.e. tags).

Some of RCVS's drawbacks include possible use of much disk space since multiple copies of a product may exist simultaneously, network speed limitations, and dependencies on specific versions of other layered products (gnu diff, rcs). We have always maintained a good working relationship with the RCVS maintainer and any bugs we have encountered have been readily fixed.

UPS

Unix Product Support[5] (UPS) was written at Fermilab. It is a configuration management tool providing support for creating and maintaining multiple versions of software products on different Unix platforms.

UPS provides good support for handling the various products that are part of SDSS software. All software written for the survey is packaged in Fermi product form and is accessed via UPS. UPS includes a mechanism for registering 'dependent' products along with the main product so that specific versions of the dependencies are linked with a specific version of the main product. UPS is easily expandable and several suggestions made by SDSS developers have already been implemented.

UPS shell commands are not intuitive. Ups also depends on Fermi utilities (funame and dropit) that are not part of the UPS product.

UPR

Unix Product Retrieval[6] (UPR) was written at Fermilab. It provides a software distribution environment allowing remote users to copy versions of software to their nodes and performing installation of the software in UPS. If a product has dependent products, those versions are copied too.

UPR has been used successfully by remote collaborators to obtain copies of SDSS software. It is menu driven and requires no detailed knowledge to use. UPR insulates the remote user from having to know the details of UPS.

UPR is not proactive, the remote user must check for new versions of products. It does not inform the user how much space will be needed on the remote disk or check to see if the appropriate amount of space is available.

GCC

GCC is the Gnu C compiler written by Richard Stallman for the Free Software Foundation[7].

GCC has proved to be a useful tool for detecting software bugs. The compiler will give warnings on suspect C lines ignored by C compilers. Examination of these warnings has solved many bugs **before** the code was included in a software release.

In order to make GCC completely ANSI compliant, we have created companion software that is used when building with GCC. The product GCCFIX contains all ANSI supported functions and header files used by the survey software that are missing from the distributed GCC environment.

WWW

World Wide Web (WWW) is a wide-area hypermedia information retrieval system using a hypertext markup language (HTML) developed at Cern[2].

WWW has proved to be an invaluable tool for exchanging information between members of the collaboration. All of the SDSS software documentation is available via WWW. In addition we have made available software development help, user feedback (in the form of bug reports and user wish lists), and software version information. Using WWW, the entire collaboration has access to the most up-to-date documentation concerning SDSS software development.

There are many features of WWW (and HTML) that have aided in the establishment of this documentation environment. The small command set of HTML makes it easy to learn especially by example. The ability to immediately display the HTML files aids in quick debugging.

Writing documentation for display through WWW raises different issues and problems than when writing flat documentation. Since HTML documentation is not meant to be read cover to cover, each 'piece' needs to be written as a coherent whole that includes links to other documentation. We had to take care not to make our documentation too circular or the user would get lost on the web. The small number of HTML commands restricts how documents can be displayed, so one should learn HTML's restrictions before picturing what a document should look like.

TCL and TK

Tool Command Language (Tcl) is an extensible interpretive command language developed at the University of California, Berkeley by John Ousterhaut et. al. [8]. In addition we have integrated in Neosoft's Extended Tcl package[9].

In general we have had very good experience with building our software environment around Tcl. Using Tcl as the standard for our data processing software provides modularity and easy integration of separately written modules into a whole. The data analysis system consists of several hundred SDSS Tcl extensions written by many different people. This set of primitives provides a flexible framework on which the user can build specific analysis software, rapidly prototype, or work interactively.

The command interpreter is easy to learn and use. Procedures can be built up that are complex yet readable. It is good for making quick tests as there is

no compilation step. Many of the SDSS collaborators develop using the Tcl extensions and then rewrite in C functions that run too slowly in Tcl.

The main problem we have had is that Tcl does not provide good support for mathematical operations. It does not do floating point arithmetic with complete accuracy. Therefore most mathematical computations we use need a Tcl extension linked to a C routine.

Tk (ToolKit) is an X Windows interface package integrated with Tcl and developed at Berkeley. Tk gives any user with Tcl competence the capability to build their own GUI.

Tk was used to help create the GUI for displaying system status information for the SDSS prototype, the Drift Scan Camera[10]. It is easier to use and learn than Motif. At the time, there was no interactive interface builder and precise layout was difficult.

Conclusion

We have had positive experiences with all of the tools described. Each is in daily use by many of the collaborators. The whole environment enables the scientists to integrate their code directly into the system. The standards enable testing scripts to be run on code without 'personal review'. Having the concept of a well defined and maintained environment early on has helped the software development process. We estimate that a total of approximately 1 year FTE (full time equivalent) was spent integrating all of the tools into a single environment and developing standards while the cost of developing a similar set of tools would take approximately 15 years FTE.

REFERENCES

1. " A Digital Sky Survey of the Northern Galactic Cap," Proposal, December 20, 1990.
2. T. J. Berners-Lee, R Cailliau, J. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," Electronic Networking: Research, Applications and Policy, Vol. 2 No. 1, pp. 52-58 Spring 1992, Meckler Publishing, Westport, CT, USA.
3. Terry Hung, "RCVS: Remote extension of concurrent Versions System," unpublished document, Stanford Linear Accelerator Center
4. Per Cederqvist, "Version Management with CVS," unpublished manual, Signum Support AB, March 1993.
5. Fermilab Computing Division, "UPS User's Guide," unpublished manual, PN-426, Fermilab, March 1993.
6. V. Gurbani, "Product Distribution Via UPR," unpublished manual, Fermilab, December 1993.
7. R. Stallman, "Using and Porting GNU CC," unpublished manual, Free Software Foundation, May 1993.
8. J. Ousterhaut, et. al., "Tcl: an Embeddable Command Language," Proceedings of the Winter 1990 USENIX Conference.
9. K. Lehenbauer and M. Diekhans, "Extended Tcl Command Set," unpublished manual page, Neosoft, Inc., January 1992.
10. B. Mackinnon et. al., "Data Acquisition Systems for the Sloan Digital Sky Survey," IEEE Transactions on Nuclear Science, Vol. 41, No. 1, pp. 105-110, Feb. 1994.

THE FORMAL DEVELOPMENT METHOD VDM^{++}

W. Lourens, A.C. Balke, J. Haveman (haaf@fys.ruu.nl)
Department of Physics and Astronomy, Utrecht University
PO Box 80.000, 3508 TA Utrecht, The Netherlands

J. Carter
CERN, ECP Division
CH-1211 Geneva 23, Switzerland

Abstract

We present an overview of the object-oriented formal method VDM^{++} and its application to the development of a second level trigger system for LHC experiments.

INTRODUCTION

The growing complexity of computer systems, and their increased use in areas where their correct operation is of vital importance, have spurred research into development methods that increase the level of correctness of these systems. The first step in such a development method should be the unambiguous specification of the intended behaviour of the system.

The use of formal methods is a possible way of reaching the above goal. The ESPRIT III project Afrodite has the objective to make formal methods more acceptable for industrial use. Utrecht and CERN members of the EAST collaboration are participating in Afrodite. During the last year they have developed a number of specifications based on the requirements for a second level trigger for LHC experiments. The experience gained in this way has been reported in [1].

VDM$^{(++)}$: FORMAL METHODS

VDM [6] is a model-oriented formal method that has been in use for two decades already. Its associated specification language VDM-SL supports both implicit (property-oriented) and explicit (algorithmic) specification styles. Complex systems can be described at various levels of abstraction: an abstract specification can be *refined* to a more concrete one and be developed ultimately into a simulation or an implementation of the system. For each step in this process there are rules to check that the specification is consistent and that the more concrete specification is a correct refinement of the abstract one.

The advantages of this approach are formality (which may eventually allow automatic reasoning support), precision (forcing one to gain a deeper insight into the problem) and abstraction (an abstract specification is easier to read than a program text).

VDM^{++} [3] is an extension of VDM that contains most of VDM-SL and adds object-oriented concepts (state is encapsulated in objects and can be accessed through methods), concurrency and real-time constructs. The emphasis in the development of VDM^{++} is not so much on

satisfying all formal proof obligations as on the development method. This makes the method more accessible to the general programmer, since it is performing formal proofs that requires most mathematical sophistication.

REFINEMENT AND CLASS HIERARCHY

The refinement relations of a formal method introduce a second hierarchy next to the inheritance hierarchy of an object-oriented system. Refinement does not fit in the inheritance hierarchy, since it is not a specialisation but involves a complete redefinition of (part of) the state. Therefore the refinement hierarchy is orthogonal to the inheritance hierarchy.

A typical refinement step in an object-oriented design is to move part of the data or of the functionality to a server object. In order to reformulate the abstract class's properties in the refined class one needs access to the (hidden) state of the server object. This is accomplished, for reasoning purposes only, by using a central concept from VDM refinement theory: the retrieve function. This function, which maps the concrete state onto the abstract one, provides a window on the abstract information content of the newly introduced server object, not on its instance variables. This makes it possible to substitute the server object by any of its refinements without having to change the client class. It is foreseen that the VDM^{++} syntax will be extended to support this concept.

This refinement scheme uses traditional polymorphism in a new way. All refined classes are made a subclass of one abstract, stateless, superclass that defines the type of the retrieve function. A reference to the superclass can point to an object instantiated from any refined class.

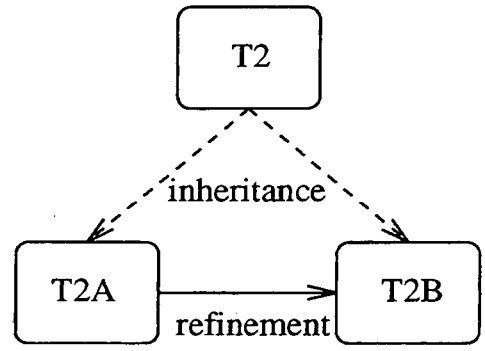


Figure 1

Figure 1 shows an example of a class hierarchy with both inheritance and refinement relations. The abstract stateless class *T2* defines the type of the state and the effect of the methods in terms of the retrieve function. Objects can only be instantiated from its subclasses *T2A* and *T2B*. *T2A* is the start of the refinement chain. *T2B* may move part of *T2A*'s state to server objects, whose retrieve functions can be used by *T2B* to formulate its own retrieve function and to express properties that *T2A* specified for its state.

CONCURRENCY AND REAL-TIME

A system is thought to be constructed from independently operating objects. The concurrency between the objects can be controlled in two ways. *Trace* statements specify the allowed invocation sequences of methods. The use of a trace statement implies mutual exclusion between method executions. A second mechanism, that allows for parallel execution of methods within one object, is the *permission* statement. Here an object accepts or rejects a method invocation depending on the invocation history of its methods and on its current state.

Objects can be made active by giving them a *thread.* The *periodic* thread specifies that a certain method should be executed periodically with a certain period. It is the only real-time concept currently in VDM^{++}. The second mechanism, the *procedural* thread, is a small program that uses select and answer statements to specify which methods are to be executed under certain circumstances.

The specification of concurrent behaviour introduces the obligation to check the satisfiability of the combined concurrency constraints, first on the class level but also on the global specification level.

TOOL SET

The tool set is centred around a Class Library Manager that keeps track of dependencies between classes and of the effects that a change to one class definition has on the status of the other classes. The CLM is coupled to tools to edit and check a class definition, to a pretty printer and to a C^{++} code generator. The latter generates code for the non-concurrent part of VDM^{++} only, using a class library to directly implement VDM's abstract types. Both a text-based and a graphical interface are available.

To allow for the integration of VDM^{++} with existing OO software engineering techniques, in particular the use of a graphical notation during the analysis phase of a project, one of Afrodite's partners is extending their existing OMT [7] tool to generate VDM^{++} class descriptions from an OMT object model. A future development will be to map an OMT state machine description onto VDM^{++} concurrency constructs.

AFRODITE STATUS AND PLANS

The object-oriented and concurrent parts of the language are now stable, little change is expected in these. At the moment the real-time part of the language is being further developed.

The CERN/UU group has used VDM^{++} to specify a feature extractor [4] and the global structure of a second level trigger [2]. Currently we are developing a simulation of the global trigger based on the latter. Another subject of research is the hardware implementation of part of the trigger by translating a VDM^{++} specification into the hardware description languages VHDL or ELLA [5].

REFERENCES

1. A.C. Balke, R. Bock, J. Carter, J. Haveman, W. Lourens, "Evaluation of VDM^{++}," Afrodite report, 1994.
2. A.C. Balke, J. Carter, J. Haveman, "The Specification of Second Level Trigger Systems for LHC Experiments," Afrodite report, 1994.
3. E.H. Dürr, A. Duursma, N. Plat, "VDM^{++}Language Reference Manual," Afrodite report, 1994.
4. J. Haveman, "Specification of a Pipelined Feature Extractor in VDM^{++}," Afrodite report, 1993.
5. C.O. Newton, P.Y.A. Ryan, E.H. Whiting, "The Translation of VDM^{++} into Hardware Description Languages," Afrodite report, 1994.
6. C.B. Jones, "Systematic Software Development using VDM," Prentice Hall International, 1990.
7. J. Rumbaugh et al., "Object-oriented Modeling and Design," Prentice Hall, 1991.

FARFALLA: C++ Data Management That Even FORTRAN People Can Love

Bob Nolty (presenter) and Chris Walter
Caltech/Lauritsen Laboratory of High Energy Physics
Caltech 256-48
Pasadena, CA 91125

Abstract

Many HEP people are devoting considerable resources to developing object-oriented tools, but often find the HEP community reluctant to learn and use them. This paper discusses a few ways of making C++ applications more palatable to FORTRAN programmers. These ideas are discussed in the context of FARFALLA, a new C++ memory-management/data I-O package we have written.

1 Introduction

There is much excitement in the software engineering community about the potentials of object-oriented programming. In this approach, one spends a relatively long time designing data structure units (called *classes*) that "know" about the real-world items they model (detector elements or interacting particles, for example). That work being done, it is relatively faster and easier to write individual programs utilizing the classes to accomplish various tasks (offline reconstructions, for example). In fact, High Energy Physics seems ideally situated to benefit from an object-oriented approach, because in each collaboration a few software gurus can do a good job of defining a *class library* for the experiment. A C++ class library offers much more functionality and flexibility to the collaborators than the subroutine libraries or programming environments which we are accustomed to receiving from our software groups.

In the past few years, many people have been developing both general purpose and collaboration-specific packages using object-oriented programming. However, they have found it difficult to "sell" these packages to their colleagues. The object-oriented programming paradigm is quite different from the procedural programming style we are accustomed to. In addition to the usual problems of learning a new language, one must learn a whole new way of reading and writing programs. Typically after being shown a HEP object-oriented package, the application programmer may feel confused about where the physics gets done, or have no idea where to begin programming.

In creating FARFALLA, our new HEP data management package written in C++ , we gave special attention to issues affecting the acceptance of our package by FORTRAN people.

2 FARFALLA

In order to make the points below clearer, I will give here a very brief introduction to FARFALLA. The basic kernel of FARFALLA is a generic data manager which knows nothing about any particular experiment. It may be thought of as a replacement for ZEBRA. All information is organized into FARFALLA *nodes*, and the nodes are linked together into *trees*. (See Figure 1.) Trees may be output to disk (in an exchangeable binary format based on the XDR standard); trees stored on disk may later be read in and recreated in memory.

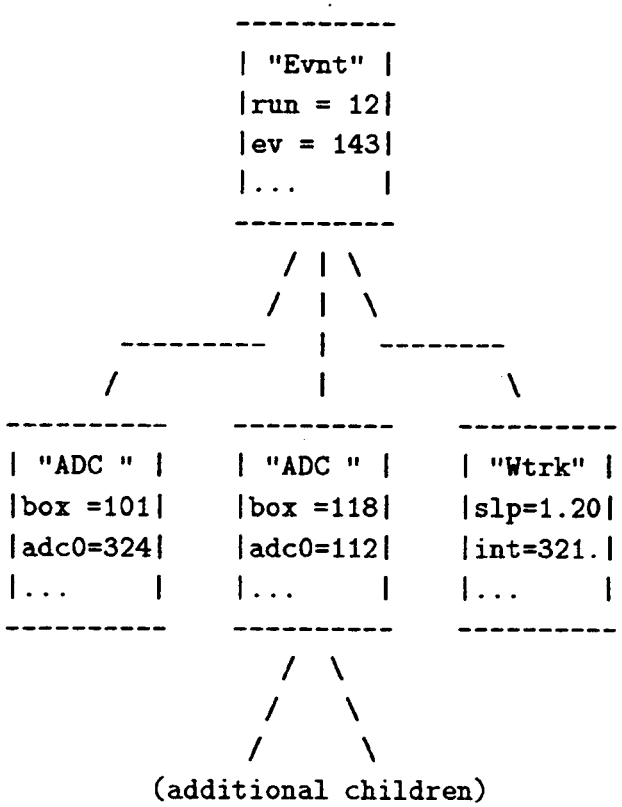


Figure 1. A FARFALLA Tree

Users or collaborations who wish to use FARFALLA extend it by defining *node types* that know about their data – raw data, reconstructed tracks, or geometry constants for example. The node types are packaged in a C++ class definition, which includes the data as well as several functions, including print functions and an I/O function that calls FARFALLA routines to shuttle the data to or from disk. The tree shown in the figure includes an eventNode, two adcNodes and a wireTrackNode. These node definitions are not part of the FARFALLA kernel, but would be provided by a user or collaboration.

3 Considerations for FORTRAN People

• RETAINING PROCEDURAL PROGRAMMING STYLE IN USER INTERFACE

In the internal implementation of FARFALLA we used powerful object-oriented ideas and we require node designers to have an understanding of C++ classes; nevertheless, data analyzers may write easy-to-read procedural programs that access the data through function calls. Thus, using FARFALLA provides a gentle transition into the world of C++ programming.

• PROTECTING USERS FROM C++ TECHNICALITIES

Some of the features of C++ require considerable effort to understand and use intelligently. When possible, we tried to avoid putting features into our interface that would require users to master these details before they could use the package. For example, casts and management of the free store are concepts with

no analogue in FORTRAN. The most straight-forward (from a C++ programmer's viewpoint) implementation of our package would have required users to use casts and manage the free store to access FARFALLA data. However, with some additional thought we were able to design an interface that required neither of these skills from the users.

On the other hand, we considered a mastery of the concepts of pointers and structures to be necessary; in fact, there would be little point in using FARFALLA if one could not take advantage of the power of pointers and structures.

• GIVING SPECIAL ATTENTION TO DOCUMENTATION

One of the advantages of object-oriented programming is its supposedly self-documenting nature. However, this is of little use to someone who is not familiar with the viewpoint of object-oriented programming. Therefore we spent considerable time developing clear, tutorial documentation. We also tried to point out which concepts a casual user could defer understanding.

• ASSUMING READERS HAVE NO KNOWLEDGE OF C/C++

It was tempting to assume readers of our documentation would be willing to read a C++ text before reading our documents. However, we knew that in reality they would not invest the time unless they first became excited about using a C++ application. Therefore our goal was to write documentation that could be understood by someone who knew nothing about C syntax, pointers, data encapsulation or inheritance. We introduced all the important concepts from scratch, and gave special attention to the important concepts of pointers and structures. We packaged the documentation as a short, readable introductory document and a larger, more detailed document including an appendix introducing C++ programming.

4 Results

We have made FARFALLA available to our collaborators in the MACRO experiment, including the documentation, MACRO-specific node type definitions, and code to create FARFALLA DSTs from MACRO run files. Some of the people using it had never written in anything but FORTRAN before, but they report being able to very quickly learn enough C++ programming to begin data analysis. Some of the users have also proved willing to tackle the more demanding task of designing new node types. Most of the users have had a few questions a month for us, more often about C++ or Unix than FARFALLA. The MACRO Offline Committee is giving consideration to making FARFALLA DST production part of the official data distribution system.

Design and implementation of a tracking class library

Nobu Katayama and Michael Smyth*
Wilson Laboratory, Cornell University
Ithaca, N.Y., 14853, U.S.A.

Abstract

We present a design of a tracking class library using object oriented methodology and its implementation in an OOP language C++.

INTRODUCTION

In fall 1994, a new three layer double sided silicon vertex detector will be installed in the CLEO II detector. The new SVD will complement already existing charged particle tracking detectors, the drift chambers. Because the new SVD is different from the drift chambers in many ways, we decided to prototype a track finding/fitting program to learn how to make the best use of the new detector configuration At the same time we decided to explore the application of object oriented techniques, in particular object oriented design techniques. Thus this work is a feasibility study of applying OO[ADP] techniques to one of the basic and important off-line problems in HEP, track finding and fitting. Although we have started from a simple geometry and clean Monte Carlo events, the program must eventually cope with the complexity in the real experiment. This will be a good test of the extendability of an OO application.

The work described here is a design and implementation of a class library for tracking with a silicon vertex detector[1] and drift chambers. Using this library we study different strategies for track finding/fitting in the new detector configuration (or future detector configuration). We would also like to answer questions related to object oriented techniques. In the next section, the design of the classes are presented. This is followed by a discussion of implementation issues.

CLASS DESIGN

Since the size of project is small, on the order of 50 classes and 20,000 lines of code, we did not use any object oriented design tools. We adopted the spiral model [2] as our design methodology. This model is incremental and iterative; we:

1. Identify the classes and objects at a given level of abstraction, algorithm, and/or complexity,
2. Implement these classes and objects,
3. Experiment and test these classes and objects,
4. Repeat until there are no new abstractions.

After a few round trips, we developed a design and implementation of classes

*Present address:Jet Propulsion Laboratory

with which we can study the track finding/fitting problem. We developed the following classes.

Detector description classes to represent the detector configuration. This includes classes that describe the geometry and response of the detectors.

Measurement information classes to represent measurements in the detectors. This includes the drift chamber anode and cathode and SVD hits.

Track information classes to represent the trajectory of a charged particle. There are several representations of a helical trajectory.

Graphics library classes to interface to the PHIGS graphics system.

Miscellaneous/Utility classes to do matrix computations and other basic linear algebra functions, container classes, and iterator classes.

The following sections describe some features of the classes in detail.

Track Information Classes

Track models the trajectory of a particle that is reconstructed from a set of measurements. It has members to:

- Find the intersection of itself and a plane or a circle,
- Calculate the Jacobian and Covariance of that intersection,
- Improve itself by taking a set of measurements into account.

We also designed light weight track parameter classes. By light weight we mean classes that are quick to construct and destroy. **cpar** is a class for track parameters in the canonical representation for a uniform magnetic field along the Z axis. However, other representations of the same track are often useful. For example, when doing a fast 2D fit, a set of circle parameters is chosen. In the case of a vertex fitting, momentum p and position x are used instead of the canonical representation. Several features of C++ help to manage these multiple representations; Constructors, assignment and cast operators are very convenient and reliable ways to go from one representation to another.

These light weight track parameter classes are used extensively in pattern recognition and segment-wise fitting where creation and destruction of the objects is quite frequent.

Measurement classes

Measurement models the SVD and drift chamber measurements associated with an event. It has members to implement track finding/fitting algorithms. It is a container class for MeasureRegions. **MeasureRegion** models measurements in a single layer of the detector. It is a container class for MeasurePoint. **MeasurePoint** is an abstract class that models a single measurement. **SiliconMeasurePoint**, **DriftMeasurePoint** and **CathodeMeasurePoint** classes are subclasses of MeasurePoint.

Detector classes

The detector classes are used to model the SVD and drift chambers. We need to be able to:

- Take geometrical data and build a description of the detectors,
- Supply information to the graphic objects in order to make a graphical

representation of the detectors,

- Give "meanings" to the measurements and improve track information.

Geometry calculates information that is common for the entire detector such as number of layers, value of the magnetic field and size of the whole detector. It is a container of GeomLayer objects.

GeomLayer is an abstract class that calculates information at the level of a single layer. Several concrete classes are derived from GeomLayer. **SiliconGeomLayer** calculates information needed to build each silicon wafer element in the layer. **AxialGeomLayer** calculates information for the axial wire layer of the drift chambers. **StereoGeomLayer** is derived from AxialGeomLayer.

The geometry description of the detector is kept in an ASCII file. These classes have ">>" operator defined. We created the geometry classes to separate the function of "reading" the detector description from the detector classes themselves. If we change the format of the description file, we can create a new subclass of Geometry to handle it, without changing the detector classes at all.

Detector models the SVD and drift chambers as a whole, containing information and members that apply to the entire detector. The detector contains a number of DetectorRegion objects. **DetectorRegion** class models a single layer of the detector. It is a container class of DetectorElement. **DetectorElement** models both an active detector element and passive material. **AxialWire**, **StereoWire**, **CathodeElement** and **SiliconElement** model active detector elements. These classes have members to:

- "Read" input data and create measurements.
- Take the measurements and associate with tracks.
- Improve the tracks, adding covariance matrix due to the multiple scattering.
- Generate graphical representation of themselves.

Graphics classes

Graphic classes provide a clean abstract interface used to produce graphical displays and to handle graphical inputs (e.g. point and click). At the lowest level, graphical functions call PHIGS library routines. PHIGS functions are rather low level, lower than a user is likely to want to use. Therefore, we constructed a hierarchical set of classes to handle graphics at a higher level.

Display handles the initialization of PHIGS as well as global properties of the display. A display object contains a number of GraphicObj objects. **GraphicObj** object is a grouping of Shapes that one wishes to think as a whole. A GraphicObj can be scaled, rotated and translated. GraphicObjs can be nested. **Shape** is an abstract class at the bottom level of the hierarchy and the basic building block of graphics. Circle, FillArea, FillAreaSet, LineSet, Plane, PolyLine, PolyPoint... are derived from Shape. A Shape does not have to be a simple graphics primitive, it can be very complicated as long as it know how to make a graphical representation of itself.

Miscellaneous/Utility classes

List is a container class, and **Iterator** is a class to read a list. The current version is implemented as a doubly linked list, although nothing in the public class definition requires this to be the case. The list and iterator classes are used where frequent additions and deletions are expected.

Matrix and **Vector** provide a set of tools for basic linear algebra operations, taking advantage of the ability to overload operators in C++ [3].

Fit classes provide functionality for a least square fit. It has a constructor that takes initial values of the parameters, "+=" operator to add a measurement, and a member function to minimize χ^2 and update the parameters. Several special types of fitting techniques can be derived from the basic fit class.

MAJOR FUNCTIONS FOR TRACKING

The goal of this project is to experiment with various track finding/fitting techniques using the class library we built. Implemented as member functions of Measurement class we developed functions to:

- Find tracks in the drift chambers in r-ϕ,
- Associate hits on the stereo wires to the 2D tracks,
- Fit tracks in 3D,
- Fit tracks using the Kalman filter,
- Find track segments in the SVD,
- Improve track parameters using the hits in the SVD.

When designing such functions we (1) think carefully how they should work, (2) identify new utility, helper classes, (3) design and implement them, (4) modify Measurement and Detector classes if necessary and (5) test the functions.

For example, a track finding (pattern recognition) function requires the knowledge of neighboring wires and hits. We designed **link(jhits)** class which links wires(hits). The AxialWire and DriftMeasurePoint classes are modified accordingly.

In developing the functions, we observed: (1) Bookkeeping and data handling are much easier in C++, (2) Use of container classes makes the code simple, (3) Recursion makes code simple in many functions, (4) Debugging is much easier than FORTRAN code as well as having fewer bugs in the first place and (5) Code reusability is much better.

Most of the time, the functions work as designed in the very early stage of the testing. However we find that there are always "exceptional" cases which breaks a given algorithm. This turned out to be a bit of a problem and is discussed in the next section.

TOPICS FROM THE DEVELOPMENT

Development cycle

The spiral model worked very well for us. However we found one small problem with it. After testing of a tracking function, the classes sometimes have a lot more member functions than necessary, members are exposed and relationships among the classes become complicated. This is because in the testing stage, we make several decisions/changes. It can

be considered that we made a bad design. However, in coping with the exceptional cases we are in fact learning about the next level of complexity.

For example, say we are working on the track fitting function. We design and implement a class that performs a linear χ^2 fitting. We modify Measurement class to incorporate this new class. While testing we discover a problem; "It needs to repeat the fits in some cases." So we make a subclass of the fit class that keeps a list of MeasurePoints. We then discover another problem; "It needs to find more hits at each stage of the iteration with the new track parameters." This forces us to send the hit list back and forth making the coupling between the fit and measurement classes casual. We thus learned that walking trough the classes and member functions to eliminate such casual couplings after each iteration is very important.

Advantages of C++

We have observed that many C++ features are indeed very useful. They include: Classes and derived classes, constructor, destructor, operator overloading, overloading of "new" and "delete", inline, templates, const, and strong typing. We also learned that some of the "good" features can be misused and become dangerous. They include: default argument value and virtual functions. In particular, an overuse of friends and public members can defeat the goal of object oriented programming.

Exceptions

We find the **exception** mechanism in C++ very useful. For example, member functions of the track parameter classes that calculates various information at radius r know that the track does not reach the given radius but does not know what to do in such a situation. On the other hand, a member function of the fit class which calls such functions knows how to cope with such an exceptional case. We therefore throw exceptions in the member functions of **cpar** class and catch them in member functions of **fit** class.

CONCLUSIONS

C++ can be used in for rapid prototyping; coding is simpler and one faces fewer bugs. C++ is probably usable for solving real problems in HEP. C++ is much better suited for problems that need a lot of bookkeeping.

Designing, implementing and testing classes is an iterative process. However, one has to look at the implementation and eliminate casual coupling of classes. In fact, when a prototyping is done, one should walk through the current implementation thoroughly before writing a production version of the code. Unlike FORTRAN programs, one can hope to do this and will end up with the code that can be maintained by other people.

REFERENCES

1. M. Smyth, "A Tracking Library for a Silicon Vertex Detector", Master's Thesis. Cornell University, Ithaca, NY, 1993.
2. G. Booch, *Object Oriented Design: with Applications*, The Benjamin/Cummings Publishing Company, Inc., 1991.
3. Matrix and Vector classes are submitted to become a part of CLHEP class library.

The Design and Implementation of an Object Oriented Software System for the E771 Fixed Target Experiment at FNAL. *

Shigeki Misawa
Physics Department
University of California
Berkeley, CA 94720 USA

Abstract

This paper gives an overview of an object oriented system that is designed to provide the components needed for the programs required by the E771 fixed target experiment. These programs includes online analysis programs for monitoring detector performance during the running of an experiment, as well off-line programs for creating DSTs (Data Summary Tapes).

INTRODUCTION

Current high energy physics experiment software systems tend to be large and complex programs that are beyond the comprehension of a single user. The goal of this project was to utilize object oriented programming techniques to develop a software system for a high energy physics experiment that was easy enough for a single person to use and understand and that could be used as a "drop in" replacement for the currently used FORTRAN software system.

SYSTEM OVERVIEW

The system, written almost entirely in C++, is separated into two major subsystems which will be called "event delivery" and "event analysis". The event delivery system takes raw data from an experiment, typically from a data tape, and converts it to a format that is suitable for analysis.

The event analysis system is responsible for providing "higher level" information for an event, such as particle trajectories in different sections of the spectrometer. This information is reconstructed from the data provided by the event delivery system.

The E771 Spectrometer

To understand the architecture of the system, familiarity with the E771 spectrometer and its data acquisition system is required. The spectometer consists of four major sets of detector planes designed to record the trajectories of particles resulting from the interaction of protons with a fixed target[1]. These four sets of planes are the silicon microvertex planes, the "front" detector planes,

*This work was partially supported by the National Science Foundation under Contract No. PHY-9121416. The author was also supported by a Department of Education fellowship during part of the development of the system

consisting of those planes that are "upstream" of the analysis magnet, the "rear" detector planes, consisting of those planes that are "downstream" of the analysis magnet, and the muon detector planes. The detector planes in the spectrometer provide either 1D (one transverse coordinate) or 2D (two transverse coordinates) position information about a passing particle. Signals from groups of these planes are collected by readout electronics which in turn send the encoded data to an event processor for consolidation into an event data block. This event block is then sent to a tape drive for storage.

The Event Delivery Subsystem

The classes in the event delivery system and their behavior are modelled after the objects in the E771 spectrometer[2].[1] The objects in the event delivery system work just like those in the E771 spectrometer but in reverse.

Event data are obtained from objects of the Tape and Disk classes in the event delivery system. The behavior of these two classes is defined by their parent class, the Data_Source class. The event data is obtained from the source in the form of a Data_Block object. Information from this object can be in one of two forms, a Tape_Header object or an Event object, depending on the type of data read from the Data_Source object.

In addition to supporting event input from Tape and Disk, a separate class, Mc_Event, which handles event data generated by a Monte Carlo event simulation package, is also provided. The Mc_Event class combines the functionality of the Event and Data_Source classes.

The extraction and decoding of readout information from the Event object or the Mc_Event object is handled by objects of classes derived from the Readout class. The Readout class defines the unpack_event message for extracting readout data from an Event or Mc_Event object. (This is done with function overloading. An additional unpack_event message is provided to allow the user to merge data from an Event object and a Mc_Event object.) The Readout class defines the decode_event message for decoding the data. An update message is also defined for updating any "parameter" information needed for these two processes.

The unpack_event message converts the data into Raw_Hit_1d or Raw_Hit_2d objects depending on whether the plane connected to the readout was a 1D or 2D plane. These objects provide hit information in the form of "wire" or "pad" numbers. The raw hit objects also contain a user writable flag that allows the user to "tag" hits. Finally, the raw hit objects provide "Monte Carlo" hit information for those hits that are generated by a Monte Carlo event simulation package.

The decode_event message converts the raw hit information into Hit_1d and Hit_2d objects. These objects provide position information for a hit in a "reference" coordinate system and in a user definable "local" coordinate system. In addition, the hit objects provide "sigma" and "width" information as well as pointers back to the plane containing the hit and to the raw hit that corresponds to the hit.

Hit and Raw_Hit information are con-

[1] This paper describes an earlier version of the event delivery system.

tained in objects of the Plane classes (Plane_1d and Plane_2d), representing the physical detector plane containing the hits. The plane classes provide array like access to the hits as well as information about the position/geometry of the plane. The plane class also provides messages that allow the user to translate the plane (and its hits) in XYZ space (through the use of a "local" coordinate system provided by the hit and plane classes.)

To simplify handling multiple readout objects, a Spectrometer class is provided that holds readout objects in a list and automates the broadcast of messages to these objects. Group and Group_List classes (for both 1D and 2D planes) are also provided that allow the user to group planes into lists and lists of lists of their own choosing.

The E771_Spectrometer class, derived from the Spectrometer class, utilizes objects of the Group and Group_List classes to provide default lists of planes that are utilized by the analysis code. The E771_Spectrometer class also automates the creation of readouts by allowing the specification of the desired readouts via an array of ASCII strings containing the names of the desired readouts.

The Event Analysis Subsystem

The primary goal of event analysis is to provide particle trajectories in the three sections of the spectrometer in (i) the silicon micro-vertex detector planes, (ii) the front detector planes, and (iii) the rear detector planes. In constructing 3D tracks, i.e., full XYZ lines, the algorithm used in the analysis first constructs 2D tracks, i.e., lines in a plane, in various projections. These two classes of tracks are represented by Track_2d and Track_3d objects. These objects provide slope, intercept, covariance matrix, and chi square information for each track. In addition, messages calculating the position and sigma of a track at a given Z position (i.e., along the beam line) are provided. A reference and a local coordinate system are supported by the classes. Track translation messages automatically calculate track information in the local system. Also, a means of storing and retrieving tracks to a block of memory for archival purposes are provided.

Derived from the Track class is the Hit_Track class which stores the individual hits associated with a track. This class redefines the storage/retrieval message to allow the hit information to be stored and retrieved. Note that the storage/retrieval messages allow stored tracks to be retrieved as objects of base classes of the stored track object's class as well as objects of the stored track object's class. The Hit_Track class also defines member functions that fit lines to the hits as well as eliminate "bad" hits from a track.

Sets of tracks are held in Track_List class objects. This class provides a means of accessing a list of tracks in an array like fashion. The class also provides member functions that store and retrieve tracks contained in the list to/from a block of memory. Derived from the Track_List class are the Track_Array and Simple_Track_List classes. The Track_Array class defines member functions for appending and removing tracks from the list. The primary use of the Track_Array class is to create customized lists of tracks. The Simple_Track_List differs from the Track_Array class in the way track memory management is handled. A Simple_Track_List object is responsi-

ble for deallocating the memory associated with the tracks it contains whereas the Track_Array object is not.

The process of reconstructing tracks from event data is handled by member functions of classes derived from Track and Simple_Track_List. The track reconstruction process involves the interaction between a Track object, a Track_List object and a Track_Cut object. The purpose of the Track_Cut classes is to localize track selection criteria into a single object. The "fine-tuning" associated with track finding is handled either by deriving new classes from the Track_Cut class that implement different selection criteria or by changing parameters in data files used to initialize the Track_Cut objects.

APPLICATION OF THE SYSTEM

The classes in the event delivery and event analysis system have been used to create classes that reconstruct tracks in all the sections of the spectrometer. The algorithms used in the reconstruction are the same as those used by the FORTRAN analysis software. However, the implementations of these algorithms differ significantly from their implementation in the FORTRAN system. Currently implemented are beam track reconstruction in 2D (3D is not possible with the set of available beam planes), silicon 2D track reconstruction (3D reconstruction is possible from the available silicon microvertex planes, but is not considered very robust), front 2D and 3D track reconstruction and rear 2D and 3D track reconstruction. In addition, a class that reconstruct the primary vertex in 3D has also been created. Classes that implement "recursive" vertex/silicon track reconstruction have been built from these track and vertex reconstruction classes. These recursive reconstruction classes utilize the results of non recursive vertex and silicon tracking to "seed" a second attempt at vertex and silicon track reconstruction[3].[2]

CONCLUSIONS

For the end user, the process of building code to accomplish event analysis is reduced to the following steps. (a)"Instantiate" (or create) the desired objects (readouts, data sources, track lists, and track cuts). (b) Write the event loop to cycle through the events. (c) Send the appropriate messages to the instantiated objects. Minor adjustments in track finding are obtained by modifying initialization files or deriving new Track_Cut classes. Major adjustments in the track finding process are made easier by building on the services provided by the base Track and Track_List classes.

REFERENCES

1. T. Alexopoulos *et al.*, "B Physics at FNAL E771," Nuclear Physics B (Proc. Suppl.) Elsevier Science Publishers B.V., 1992.
2. S. Misawa, "The Application of Object Oriented Programming Methods to Event Delivery in Experimental High Energy Physics," FERMILAB-Pub-93/079 Fermi National Accelerator Laboratory, March 1993.
3. A. Boden, "Observation and Reconstruction of B Mesons in p-Si Collisions at 800 GeV/c," Ph.D. thesis, University of California at Los Angeles, 1993.

[2]The FORTRAN implementation is discussed in this thesis.

AN OBJECT-ORIENTED TECHNIQUE FOR DETECTOR MODELS USED BY A SIMULATION PROGRAM *

Miroslaw Dach, Nils Hoimyr, Janne Saarela, and Jouko Vuoskoski
Computing and Networks division, CERN
CH-1211 Geneva 23, Switzerland

Abstract

STEP is a new International Standard to exchange and represent product model data. It uses an object-flavoured data specification language to specify the representation of the data. We present the design and implementation of a program where STEP principles were applied for detector models used by the GEANT simulation package. The program reads GEANT detector models and converts them to object instances into a class library in order to be easily accessible by other applications. We also describe the development of an interface application which will be implemented within this program in order to exchange detector models between GEANT and CAD systems via STEP physical files.

INTRODUCTION

STEP [1,2] (STandard for the Exchange of Product model data) includes a data specification language, EXPRESS. STEP defines also a physical transfer file. STEP provides a system-independent way of implementation. These features offered a possibility to apply the STEP principles to detector models using modern object-oriented design and implementation.

We have implemented a program where STEP principles were applied to detector models used by the GEANT [3] simulation package. The design of the program was carried out using the Object Modeling Technique (OMT) [4]. The C++ programming language and the NIST STEP Toolkit [5] were used in the implementation.

The program reads a GEANT detector model from a ZEBRA RZ file [6] or from a STEP physical file and converts it to object instances into the class library. After this the geometric data of the detector can be accessed. After desired operations the detector model can be written in an RZ file or a STEP file. The program is to be used as a basis to implement different applications. The class library access routines offer an environment to manipulate or use GEANT detector models.

We also describe an interface appli-

*Financial support is acknowledged from the Finnish Ministry of Trade and Education, Institute of Particle Physics Technology (HTI) in Espoo, Finland and from CERN in Geneva, Switzerland.

cation which will be implemented within the program. The interface application, called CADGE [7], is under development and it will be an interface between STEP physical files to be used by CAD systems and GEANT. This application is to be used by engineers and physicists to exchange detector models in order to improve their communication.

STEP is a collection of standards to represent and exchange product information. The objective of STEP is to offer a system-independent mechanism to describe the product information in computer aided systems throughout its lifetime. It separates the representation of product information from the implementation methods. Implementation methods are used for data exchange. The representation offers a definition of product information to many applications. STEP provides also a basis for archiving product information and a methodology for the conformance testing of implementations.

EXPRESS is a data specification language that allows encapsulation of data, implementation of constraints and relationships among objects. EXPRESS is not a programming language, for it does not support input/output elements, information processing, etc. However, EXPRESS can be processed by a computer.

PROGRAM PRINCIPLES

The program consists of the file input/output for STEP files and RZ files. It contains a class library, routines to access the class library, and routines to convert detector models between the program and RZ files. STEP files can be used to store or transfer GEANT detector models. The files are written according to the GEANT geometric representation. The file is a human-readable ASCII-file.

Software used in the program includes
- o C++ steering program,
- o C++ binding for OSF/Motif,
- o CERNLIB software (GEANT,ZEBRA),
- o NIST STEP toolkit.

The C++ steering program is the main program which performs user actions and conversions between RZ files and the class library. A C++ Binding for OSF/Motif [8] is needed in order to create a visual interface to the program. The GEANT Fortran library and the ZEBRA package [6] are linked with the main program. They are needed to access a detector model which is saved in an RZ format. The NIST STEP toolkit provides routines to manage object instances and STEP file I/O. It contains an EXPRESS compiler in order to create a C++ class library from the GEANT EXPRESS schema.

When transferring data from an RZ file to the class library, the C++ main program initializes the data structure via GEANT, which loads the data into ZEBRA banks in memory. The data is then converted and transferred into object instances which are derived from the class library. The same procedure can be applied to reverse direction.

The STEP file I/O is handled by the STEP toolkit. Routines for I/O operations are implemented as member function of the class *STEPfile*.

CADGE

CADGE, a CAD-GEANT interface application, will convert detector models between GEANT and CAD systems. The exchange file format to be used in CAD systems will be the STEP physical file. The conversions will be performed

for geometric representations defined in the STEP part 42 (Geometric and Topological Representation) [1].

CADGE will contain two separate schemas, STEP part 42 and GEANT, which will be compiled into the interface program as separate class libraries. The geometric conversions will be performed between the two class libraries. Each geometric entity in a class library will be converted to the representation of the other library. The conversions will be implemented in each class as member functions. Each entity will know how to convert itself to an entity or entities of the other class library. The member function definitions for conversions will be added to the class definitions which the EXPRESS compiler produces.

CONCLUSIONS

We have designed and implemented a program in order to access and transfer detector models using STEP principles. The program is to be used as a basis to implement different applications.

The implementation of the program was a part of the GEANT-CAD interface project. The program serves as the basis for the project. The interface will be implemented within the program, and it is still under development: the prototype version is expected by this summer.

ACKNOWLEDGEMENTS

We are grateful to our colleagues at CERN for their help. In particular we would like to thank Federico Carminati and Mik Ferran. We are also grateful to the staff of US National Institute of Standards and Technology (NIST) and many others for their help in this project.

REFERENCES

1. International Organization for Standardization. "ISO 10303 Industrial Automation Systems and Integration - Product Data Representation and Exchange," ISO TC 184/SC4, 1992.
2. Bradford M. Smith, "The STEP Project," ISO bulletin, June 1992.
3. R. Brun and F. Carminati, "GEANT Detector Description and Simulation Tool," CERN Program Library, W5013, 1993.
4. J. Rumbaugh et. al., "Object-Oriented Modeling and Design," Prentice-Hall, Inc., 1991.
5. K.C. Morris, D. Sauder, S. Ressler, "Validation Testing System: Reusable Software Component Design," National Institute of Standards and Technology, Gaithersburg, MD, NISTIR 4937, 1992.
6. R. Brun, M. Goossens and J. Zoll, "ZEBRA - Data Structure Management System," CERN Program Library, Q100, 1991.
7. J. Vuoskoski, "CADGE - A Proposal for a General CAD-GEANT Interface Application," Helsinki University of Technology, Report TKK-F-A715, 1993.
8. K. Seetharaman, C. F. Rei, B. R. Montague, "A C++ Binding for OSF/Motif," Univ. of Lowell, 1990.

OBJECT-ORIENTED ANALYSIS AND DESIGN OF A GEANT BASED DETECTOR SIMULATOR

K. Amako, J. Kanzaki, T. Sasaki, Y. Takaiwa
KEK, National Laboratory for High Energy Physics, Tsukuba 305, Japan

Y. Nakagawa and T. Yamagata
International Christian University, Mitaka 181, Japan

Abstract

We give a status report of the project to design a detector simulation program by reengineering GEANT with the object-oriented methodology. We followed the Object Modeling Technique. We explain the object model we constructed. Also problems of the technique we found during our study are discussed.

INTRODUCTION

High energy physics experiments in next generation colliders will face with the following serious problems in managing their software systems: a drastic increase in their scale and complexity comparing to the systems in the present, a long span of the software life cycle, a large number of persons who commit to the development and maintenance of systems, and they are internationally distributed.

In recent years, various software development methodologies have been advocated and used in industry and also in the HEP community. These are for the development of a software system in the similar environment as described above. Among these, the object-oriented analysis and design (OOA and OOD), although it is still evolving rapidly, is considered to be the most promising technique. We decided to initiate a feasibility study of OOA/OOD for HEP off-line software developments, because this has not been well studied yet. The name of the project is called ProdiG which stands for PRoject for Object-Oriented Design and Implementation of Geant.

STRATEGY OF THE PROJECT

The goal of the project is to answer the following question: Is the OO programming with OOA/OOD methodology a right approach for the development and maintenance of large scale HEP software systems in future experiments?

Our basic approach to attack the problem is to reengineer GEANT by OOA/OOD: To analyze and design a detector simulator of which functionality is same as GEANT. This allows us to reuse software assets (both ideas and algorithms) the HEP community accumulated so far. We believe reuse is a critically important issue when we try to shift to a new technology.

METHODOLOGY

The methodology we employed is Ob-

ject Modeling Technique (OMT) by J. Rumbaugh, et al.[1]. OMT divides the development process of a software system into four stages: 1) analysis; 2) system design; 3) object design, and; 4) implementation. Throughout these stages, three models are used to describe the system: 1) the object model; 2) the dynamic model, and; 3) the functional model.

The design phase, the stages from 1) through 3) in the above, terminates when the construction of the three models is completed. In the following sections, we explain the process of the model construction for our detector simulator.

OBJECT-ORIENTED ANALYSIS

Problem Statement

The analysis phase starts from the construction of the problem statement. Because of the vague definition of the problem statement in the Rumbaugh's book, what should be or should not be included in it is always controversial. We referred the GEANT manual [2] to generate the problem statement. We took the following steps: 1) to select essential statements and to avoid ones which are specific to the GEANT design from the manual (we used the version 3.16), and; 2) to update the problem statement as our analysis and design proceed.

Object Modeling

An object model consists of object diagrams and a data dictionary. An object diagram shows relationships, attributes, and operations of classes/objects in a system. A data dictionary textually describes each class, its associations, attributes and operations. We generated an initial version of the model from the problem statement created in the previous step. Basic guides for generating the model are: 1) nouns correspond to classes; 2) verbs correspond to associations, however; 3) don't forget any noun can be verbed and any verb can be nouned [3].

We found that the design of geometry related classes and their associations is relatively straight forward at this stage. We consider this is because the relations of them are mostly static. In contrast, the track related classes and their associations are difficult to design at this stage, because their natures are dynamic. For the the design of track related classes, the dynamic model described below plays an important role. The initial version of the model evolved a lot as we proceeded studies of other models.

Dynamic Modeling

A dynamic model consists of state diagrams and global event flow diagrams. These two diagrams summarize dynamical behavior of a system. To construct a dynamic model, we started from writing scenarios. A scenario describes a sequence of events that occur during a particular execution of a system. We generated scenarios by referring the GEANT algorithm of transporting a particle. We then created event trace diagrams from these scenarios. These diagrams describe the sequences of events in the particle transportation. During the study of the diagrams, we found new classes that are not appeared in the previous stage. We assigned the operations found in this stage to appropriate classes.

We also generated event flow diagrams and state diagrams. However, they do

not play an important role in the analysis, because they only contain summary information of the event trace diagrams.

Functional Modeling

A functional model consists of data flow diagrams. A data flow diagram shows how values are computed, regardless of sequencing, decisions, or object structure. We think a natural way to study the functional (operational) behaviors of a system is to study them in close connection with objects so far found. However, Rumbaugh's approach in the functional model is: 1) to consider data flows by ignoring objects; 2) to break down processes into functional levels, and; 3) then, to assign these functions to classes. Since objects are considered from the beginning in the construction of other models, this approach in the functional model is rather different. We have not understood yet the role of the model in our design.

OBJECT DIAGRAMS FROM THE OBJECT-ORIENTED ANALYSIS

In OMT, the object model plays the most important role. In this section, we explain the object diagrams obtained by the study done so far. Because of the space limitation, we cannot show all diagrams we generated.

Geometry Related Classes

In fig. 1 we show the current version of the object diagram for geometry related classes. This diagram shows only the relations between classes of which roles are relatively generic. In the diagram "Volume" play a major role. It abstracts geometry and medium information necessary for tracking a particle. "Volume" has aggregation relations between "Shape", "Medium" and "BField". The class "Shape" describes objects that have geometrical shapes such as box, sphere, etc. The class "Medium", with the aggregation class "Material", has attributes such as density, A and Z numbers, various control parameters for tracking, etc. The class "BField" abstracts the magnetic field in a volume.

Another important class is "VolumeLink", which is a link attribute of the association between two volumes. It describes geometrical relations between mother and daughter volumes.

In GEANT multiple positioning of a volume to the same or different mothers is possible. The classes "VolLocated" and "VolDivided", inherited from "VolumeLink", describe two ways to place a volume in the mother volume; the

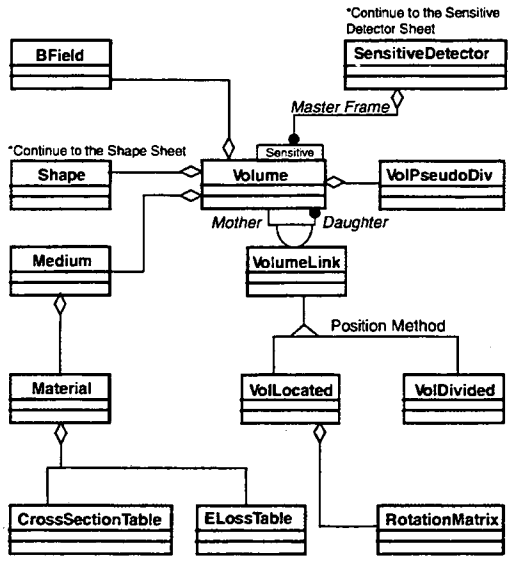


Fig.1. The object diagram of geometry related classes. See the Rumbaugh's book [1] for the diagram notations.

positioning and the division. "Volume" also has an aggregation with the "VolPseudoDiv" that describes the pseudo-division of a volume. In the diagram it is shown another major class "SensitiveDetector". This class abstracts "hit" information of particles in sensitive volumes.

Particle Related Classes

In fig. 2 we show the current version of the object diagram for particle related classes. It again illustrates only the relations between classes of which roles are relatively generic. "Event", with its aggregation class "Vertex", has attributes which keep information of the current event for the simulation. The class "Vertex" has attributes which keep vertex data, such as a vertex position.

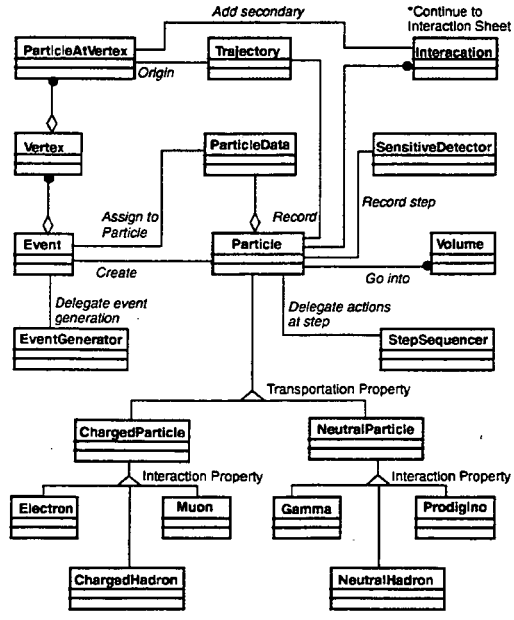


Fig.2. The object diagram of particle related classes. See the Rumbaugh's book [1] for the diagram notations.

Information at a vertex, such as particle types, their four-momenta, etc., are kept in attributes of another aggregation class "ParticleAtVertex". The "Event" class has association with the "Particle" class, which describes the transportation of a particle in the geometry. "Particle" is the super class of "ChargedParticle" and the "NeutralParticle". In this generalization, the transportation property is abstracted. From these two classes, six types of classes ("Electron", "Muon", etc.), which correspond to tracking types in GEANT, are derived. The static property of a particle is described by the aggregation class "ParticleData".

The interaction between a particle and a medium is described by the class "Interaction". When secondary particles are created by an interaction, they are instantiated as "ParticleAtVertex" objects, and are associated with the "Event" object. The classes "Trajectory" and "SensitiveDetector" describe particle trajectories and "hit" in a sensitive detector respectively.

OO DESIGN AND IMPLEMENTATION

In OMT the object design is started after completing the object analysis stage. During the object design phase, the analysis models are elaborated, refined, and then optimized to produce a practical design. There is a shift in emphasis from application concepts towards computer concepts. Unfortunately this definition does not give a clear boundary between the analysis and the design phases. Because of this obscure distinction, we sometimes faced with a difficulty in judging whether a certain class should be considered in the analysis or the design phases. In our current design, classes re-

lated to the following items are to study in the object design phase: 1) user interface; 2) controller to drive the system, and; 3) utility library, etc. For the item 3), we think it is important to have a common HEP class library.

CONCLUSIONS

We studied the feasibility of OOA/OOD by reengineering the GEANT program from the object-oriented view. The OMT methodology was used. Referring the GEANT manual for the generation of the problem statement gave us a good start point of reengineering. We found that the object model and the dynamic model provide effective tools to exchange ideas of the design among our collaborators. Because of its different approach than other two models, the functional model was not used in the present design. We found that the design of geometry related classes are straightforward from the problem statement. For the design of tracking related classes, the dynamic model played a major role. The models we generated give a clear description of the current design of our detector simulator.

ACKNOWLEDGMENTS

We would like to express our special thanks to X. Zhou for his guide to the object-oriented approach. We also acknowledge to M. Asai, A. Breakstone, R. Brun, F. Carminati, S. Giani, S. Kabe, M. Kawaguti, H. Sakamoto, S. Tanaka, Y. Watase and H. Yoshida for their useful discussions.

REFERENCES

1. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design," Prentice Hall International, Englewood Cliffs, NJ, 1991.
2. CERN Program Library Long Writeup W5013, CERN, 1993.
3. G. Booch, "Object-Oriented Analysis and Design with Applications," Benjamin Cummings, Redwood City, CA, 1994.

Quo Vadis Code Optimisation in High Energy Physics

Sverre Jarp
Computing and Networks Division CERN

Although performance tuning and optimisation can be considered less critical than in the past, there are still many High Energy Physics (HEP) applications and application domains that can profit from such an undertaking. In CERN's CORE (Centrally Operated RISC Environment) where all major RISC vendors are present,[1] this implies an understanding of the various computer architectures, instruction sets and performance analysis tools from each of these vendors. This paper discusses some initial observations after having evaluated the situation and makes some recommendations for further progress.

1. Introduction

Recently an explosion of affordable processing-power across the computer market has been observed. CMOS[2] -based RISC microprocessors with power that exceed that of expensive mainframes are now available at a cost well below $1,000 (or $10,000 for a complete workstation). The results have been extremely positive, especially given the fact that no previous efforts (such as vectorisation) had really managed to revolutionize the cost of HEP computing.

The question raised in this paper is the issue of code optimisation. With a 15–20 million dollar CDC 7600 (rated at 10 MIPS) or IBM 370/168 (whose capacity incidentally helped define the CERN unit = 4 MIPS), there was no doubt that code optimisation was justified due to the fact that the acquisition costs (even before adding maintenance) corresponded to hundreds of man-years. As a consequence, even if it took a complete man-year for somebody to improve the annual output of such a machine by a modest 1%—the effort was already cost-justified.

Today, the situation is different. The acquisition cost of the Central Simulation Farm, consisting of 25 high-end HP workstations, with its 675 CERN units would now cost about 500–600 K$. The 10-processor SGI Challenge, equipped with 300 GB that L3 recently decided to acquire, is estimated at a similar cost for about 190 CERN units. Given annual manpower cost in the region of 100 K$ it is not difficult to understand that the situation has changed dramatically. A programmer is now required to make improvements per man-month that have a global effect of several percent. Nevertheless, this is still entirely feasible and should be encouraged in several cases.

2. Why is code optimisation still of interest?

The LEP experiments, which were promised no more than 2 CERN units each by the CERN Directorate in 1982 but now enjoy more than 100 times that tiny capacity, are still not satiated. When the LHC experiments were not yet using CSF, one or two LEP experiments still managed to swamp the whole farm without any problems. This is seen as a clear indication that the need for simulation capacity is semi-infinite.[3] As the accumulated number of Z^0 events increases over the next two years, it is also expected that the capacity for analysis will continue to grow.

Another argument for code optimisation comes from the fact that physicists will always welcome better turn-around. If a job that used to run in a day can now be made to take only one hour, so much the better. Unfortunately, whereas industry can often relate to direct or indirect manpower savings in such cases (to justify the tuning effort), a research organization like CERN tends to categorise such improvements as increased 'comfort' rather than improved efficiency.

New experiments, such as the heavy ion NA49 experiment and the LHC experiments in the next

[1] Alpha from DEC, Power from IBM, Precision Architecture from HP, MIPS from SGI and Sparc from Sun.
[2] Composite Metal-Oxide Semiconductor.

[3] Today several LEP experiments have private simulation farms in addition to using the central cluster.

decade, will have much bigger appetites than LEP. A fully simulated NA49 event is estimated to take about 1 hour on a CSF processor and a full simulation of a LHC event may last 100 times longer than a LEP event.

By gradually changing the way physicists perform data analysis, one can also expect large additional CPU capacities to be required. When hundreds of LEP and LHC participants will perform data mining on huge event loops in quasi-real-time on super-PIAF systems, these will have to be of enormous capacity and extremely well tuned.

Other paradigm shifts may also come along, be it virtual walks inside an operational detector or other 'bright' ideas that will require enormous amounts of well-optimised computing capacity.

Another factor of influence will be the cadence with which manufacturers offer higher CPU-speeds. Whereas it has been relatively straightforward to increase microprocessor chip speeds from say 20 to 40), 80 and 160 MHz, it is very likely that the continued doubling (to 320, 640 and higher) is going to be much more difficult, and the microprocessors may not show up as quickly as the HEP community has anticipated.

When the manufacturers use other 'tricks' to increase their performance ratings, such as the inclusion of more and more parallel execution units, speculative execution, branch prediction, a more complex instruction set, multiple cache levels, etc., it is unavoidable that only with a corresponding tuning effort will we come close to optimal use of these complex additions to the original RISC concept. Of course, sophisticated compilers will try to close the gap, but it is highly unlikely that these compilers will grow in complexity sufficiently quickly.

3. Multiple performance domains

This paper concentrates on CPU optimisation, but this is far from being the only area worthy of interest. One could even argue that the exploitation of affordable RISC power has only created other bottlenecks, such as inadequate networking performance and limited disk throughput.

With the first couple of generations of workstations, compile and link time was a major issue, and it was of limited comfort to know that the application would run well once it had been built if it took forever to get the compilation through! Even today, the compilation and generation of complete libraries is a long and painful exercise and there is some ongoing effort to try to understand how to improve the situation.

For a parallel system like PIAF,[4] RAIDs[5] that allow file striping have become indispensable and the system is critically dependent on high-speed interconnect (currently based on FDDI) and NFS performance.

4. What is the issue of CPU performance tuning?

In an ideal world our programs would contain an ideal set of algorithms describing our problem domain which we would transform using a perfect compiler that translated every algorithm into machine code without any loss of cycles compared to the peak performance of the hardware in question. In High Energy Physics (as in most other areas of society) real life is unfortunately quite different.

Of course, it was a true revolution (similar to the RISC revolution?) when all HEP programs could be written in high-level languages such as FORTRAN IV without the need for assembly-level programming expertise. FORTRAN gave the programmers the freedom to concentrate on the algorithmic part of the problem without the need to worry about low-level details. (Relegate the Important Stuff to Compilers!) Programs were also made portable, and this gave the physics community another 'arrow to its bow' when hunting for adequate CPU resources.

This abstraction, however, still carries a price. Given that the underlying hardware is as 'messy' as it has always been, a programmer may experience all sorts of unpleasant surprises if he/she does not try to understand what really happens in the hardware during program execution. By initially writing code that, for instance, serialises instruction execution (leaving most of the hardware idle), traverses multi-dimensional arrays in a sequence unoptimised for

[4] PAW Interactive Analysis Facility.

[5] Redundant Arrays of Inexpensive Disks.

the hardware, or jumps from routine to routine so frequently that the cache is negated, a programmer can easily suffer performance degradation of several factors compared to a modified program where these errors have been corrected.

As the HEP community is getting ready to embrace object-oriented programming in a more substantial way, a similar Damocles sword could be put above our heads. OOP will allow for wonderful abstraction (and hopefully increased programmer productivity) but most books on the topic warn of reduced efficiency and unexpected performance problems if this issue is not properly tackled at the time of initial design and prototyping.

Another problem appears because of the layered approach all modern programming effort has taken. A modern HEP program can today rely on hundreds of CERNLIB library routines which offer the advantage that a large portion of the programming effort can be reduced to calls of a huge set of pre-coded subroutines from within an application-specific environment. This is typically the way HEP simulation and analysis jobs are produced, with the risk that performance issues may crop up because of the large number of routines (and the frequency of calls) in rather unexpected ways.

Even if a given performance problem is detected straight away (or after some time of running production), we are now confronted with the issue of distributed ownership. An individual physics programmer may find that the library support staff do not agree with a proposed change to a routine. Maybe neither side has the time and effort to produce a change and test it across all the CERNLIB-supported platforms, or maybe the ownership is so diluted that the routine is considered frozen with a 'take-it-or-leave-it' sticker attached.

There is also the problem of algorithmic aging. What was perhaps initially a clever and well-implemented algorithm will often end up being diluted by successive sets of changes, often implemented by (if 'this_special_case' then 'do_my_little_thing') to an extent where the famous expression, 'spaghetti code,' is the only way to describe a large and unmanageable set of routines that work, but that nobody dares touch to address performance issues, for fear of destroying program correctness.

Fortunately there are positive exceptions to these observations, and in the case of GEANT, S.Giam et al. recently produced a new release (3.21) that offered substantial gains in certain cases by adding a new layer of virtual divisions for speeding up the tracking inside a complex detector with a large number of volumes. As a matter of fact, this release was preceded by two other efforts to improve the detector tracking that had become an increasing worry for the next generation of experiments. As a preliminary result, the Atlas simulation job, *dice*, now runs 20% faster than with GEANT 3.15[1].

5. Frequent HEP Code Observations

When using a classical performance analysis tool like *prof* (for profiling the time spent in each routine), the most important observation for HEP codes is the flatness of the distribution. One is as far from the preferred case (more than 90% of the time spent in less than 10% of the code) as one could be. This is true for a simplistic simulation benchmark program such as GEXAMI, but also for a complex implementation of GEANT, such as *dice*. The top routine consumes less than 10% of the cycles, and 50% of the cycles are spread over approximately 20 routines. By looking at the Aleph simulation, reconstruction and analysis programs a similar flatness although slightly less pronounced, is also observed. One immediately understands that an attack on a single subroutine is of limited value.

In this context it is worthwhile to mention the feature of inlining. Given that HEP has decided to write programs where the major portion of the code is composed of precompiled library routines, we have abandoned the possibility of giving the compiler a global optimisation capability through inlining. This leads to a reduced efficiency, often estimated to be almost a 20–30% elongation of the execution time. Nobody puts in doubt, however, that well-structnred HEP libraries bring other advantages, such as human productivity gains.

Another observation is that the intensive use of Zebra memory structures forces the program into a pervasive indexing mode where a lot of time is

spent calculating indexes in the various structures. Since this is done throughout a typical HEP program, one needs to try to tackle the issue from another angle. At the heart of the problem is obviously the fact that FORTRAN-77 does not contain adequate data structures with pointers for rapid searches through layers and chains. Another way of expressing the problem is to say that the compiler has no way of being smart about such structures, and a study by one computer vendor found the calculation of indexes (through repeated sequences of multiplication, addition and loads) to be so bothersome that they looked at inventing such an 'indexing' instruction. Unfortunately for HEP, but quite expectedly, such an added instruction would have no positive impact on the official SPEC benchmarks, so the idea was abandoned.

6. What is the best way forward?

The best way forward is definitely to design and maintain applications with performance in mind right from the start. This does not mean that the design should be platform-specific, but rather that there is a good understanding between cause and effect when algorithms and data structures are programmed.

Ideally HEP should also try to stay as close as possible to an overall program-design that has three distinct phases (or set of phases): first, prepare to compute, then compute (as fast as possible), and third, perform postprocessing and cleanup.

The advantage of such a design is the fact that in the middle part, which now becomes the central kernel, one has isolated the compute intensive portion, so that performance optimisation or even porting effort to different architectures can essentially be concentrated here. It is claimed that such a design would always make tuning easier, and efforts to port to an existing parallel, vector-designs or even hypothetical VLIW[6] designs in the future, would have a higher chance of success.

The next thing to master is compiler behaviour and ways to influence it. Modern compilers come with a large set of options, and this is unfortunately made more complicated by a varying set of options for each vendor's compiler. For a full understanding of the compiler options, hands-on experience and an extensive study of the documentation are necessary.

What is also unavoidable (unfortunately) is a relatively good understanding of the hardware features, often hidden from the everyday programmer, that can govern performance. Unfortunately these hardware and architectural differences are often not made easily available by the manufacturers, and it may be necessary to consult specialised literature to get an impression of the differences between the vanous architectures [2]. A different number of entries in a cache or a translation look-aside buffer can have a significant impact on the execution speed.

For instance, a (very artificial) test program written for an HP-9000/735 slows down by a factor 3 when manipulating data in more than 120 pages in quick succession (D-TLB break-down) and a factor 10 when manipulating more than twice the number of cache-lines in quick succession (D-cache break-down).

It goes without saying that we need to understand the hardware implications of the features that can be manipulated directly such as the vanous extensions to the instruction sets, etc.

Armed with a good understanding of hardware functionality and compiler options, the next issue that needs to be addressed is the question of tools. Most UNIX systems come with a standard set of performance tools[3]. There are *prof* and *gprof* execution profiling and calling-tree reporting. DEC and SGI offer *pixie* for statement basic-block tracing and instruction counts, IBM offers *tprof* whereas SUN and I-IP offer only packages inside special products aimed at program development and analysis.

All vendors allow assembly-level code to be generated by the compilers for inspection, normally via a -S option, but such code can only be 'trusted' in the case of IBM and DEC. For the three other manufacturers there is a supplementary stage of optimisation of the machine code that takes place after the assembly code has been generated. In these cases, we are obliged to use the dissembler, *dis*, if there is a need to understand program behaviour at the lowest level. The debugger, normally *dbx* but *xdb* on HP, can also be

[6]Very Long Instruction Word.

used for code inspection. Finally it should be noted that the assembler, *as*, is present on all systems and can be used for testing modified assembler code, even if it is only as an effort to test modified behaviour and not an effort to rewrite code. The HEP community may, however, be guilty of having abandoned assembly-level optimisation too easily on RISC-platforms,[7] for fear of falling into unmanageable maintenance problems. It seems to the author that it would be possible to insert assembly-level code for kernel functions (typically routines from KERNLIB) with the knowledge that a high-level C or FORTRAN routine exists and is available as a fall-back solution. This tactic was adopted by L3 (for several of the bitmanipulating routines) during the recent benchmarking that preceded the acquisition of their SGI/Challenge. There seems to be no reason why other heavily used routines should not undergo the same treatment.

With all this hardware and software knowledge and tools there seems to be only one thing missing that would change the profession of a tuning expert from 'black magic' to a straightforward engineering job. This missing part is the capability within the processor to report (via hardware counters) 'hidden' events that potentially cause performance loss, such as inactive execution units, stalls because of serialised instructions or branching, TLB-faults, or cache misses, etc. Whereas at least DEC and IBM have the hardware capability to do this, there are no publicly available tools for getting at the information. It is nevertheless worth mentioning that IBM recently gave the author a demonstration of an internal tool, *rs2pm*, that shows dials on the computer screen during execution with needles indicating quite nicely the internal behaviour of the processor hardware.

7. Conclusions

High Energy Physics programs have profited greatly from the revolution in price/performance of modern microprocessors compared to mainframes of the eighties. This has allowed the LEP experiments to equip themselves with very comfortably sized computing clusters that exceed the prediction made ten years ago by at least two orders of magnitudes.

However, for a number of reasons this should not leave us complacent about performance optimisation, but encourage us to target our efforts in areas that will show 'return on investment.'

Representatives of the manufacturers are often amongst the most eager to look at performance issues, especially when a new processor has just been announced and needs positioning in the HEP arena, or when a bid for a specified performance yield needs to be won.

If and when HEP decides to invest heavily in OOP, with its promise of higher programming productivity, it is important to monitor the resulting performance right from the start to avoid that the savings in programming effort is outweighed by losses of performance.

Predictions for LHC show that up to 10^7 MIPS may be needed for the various computing phases. Given that we may need an insurance policy against the manufacturers' incapability to deliver continued speed increases along the same aggressive curve as in the past, we must make sure that our programs are not only well-tuned, but also designed to be tuneable (repeatedly) and adaptable to continued changes in computer architecture.

To achieve this, HEP should put pressure on the manufacturers to design the microprocessors with performance monitoring in mind and to provide all serious users with adequate tools for understanding program behaviour and the resulting performance levels.

References

[1] S. Ravndal: Minutes of GEANT monthly meeting, 8 March 1994.

[2] L. Gwennap: Year in Review: Alpha maintains performance lead in 1993, Microprocessor Report, Vol.7, No. 17 (December 1993).

[3] K. Dowd: High Performance Computing, O'Reilly & Associates.

[7] There are almost 100 assembly-level routines in use on VAX/VMS (mainly inside KERNLIB).

Object Structural Modeling in the DELPHI Online Event Display

Serge Dû, LAL - Orsay, France
Jean-Philippe Laugier, DAPNIA - Saclay, France

Abstract

The Online Event Display for Delphi has been developed for 3 years using Object Oriented methods and tools. The first experience was presented at CHEP'92 [1]. After several cycles in the software life, we are now in position to evaluate the benefits of such an approach.

INTRODUCTION

The DELPHI Online Event Display (OED) was aimed at visualizing all detector and trigger raw data in general views and other customizable forms as requested by the detector experts. The nature of the application, in particular its easy modelisation and its anticipated regular evolution depending on the physicists wishes clearly suggested to consider Object Oriented methodologies.

The "Class-Relation" method developed by Softeam [2] describes both the Object Structural and Object Behavioral models [3]. A powerful tool called *Objecteering* provides services like a graphical editor and a C++ code generator. Up to now, it is effective only for the Object Structural model, and our experience is restricted to that model.

PROJECT OVERVIEW

At the project start time in 1991, it was decided to evaluate *Objecteering* at least for the specification phase and early development steps. After 3 months, a first release, with a very limited number of detectors and functionalities, demonstrated that the prototype could be reused and extended to satisfy the overall specifications. An important point was the ability to incorporate existing and tested software from detector experts into a coherent and extensible framework.

The software then grew up to incorporate more detectors and to fullfill the requirements. In the two subsequent years, major releases were provided to satisfy new specifications that had not been foreseen at the early design stages.

APPLICATION ARCHITECTURE

Building Elements

Three types of components can be identified in the software :

- classes building the general framework (58 classes)
- classes describing real detectors

and calling existing routines (69 classes).

- routines provided by detector experts (mainly Fortran code)

The framework

The general framework is made out of two kinds of classes:

o a set of *concrete classes* that will effectively be instanciated in the application:

- general tools to manage the display (editors for Devices, Views, Display Lists,...)
- views definitions (View, Board, ViewManager, DisplayList,...)
- trigger response description (TriggerView, TriggerLevel,...)
- geometrical components (Shape, Cylinder, Cone, Pad,...)
- user interface components (built upon Xt via *OnX* [4])
- application shells (oed_shell, detectors_shell,...)

o a set of *abstract classes* are used to model the hierarchy of detectors and their representations: Device, Volume, DataBlock, RootDevice, EventHeader

The detectors

The sub-detectors of DELPHI are described by classes which all inherit from the previous abstract classes. The introduction of a new sub-detector can be considered itself as a sub-project with dedicated analysis and design steps.

First, there is a new requirements specification phase for each sub-detector. Detector experts define *what* has to be displayed: kind of data and relevant representation (general display, specific projection, combined display with another sub-detector). That step may imply some modifications at the framework level. Now that many detectors have already been described, the basic model appears to be stable.

To describe the new sub-detectors, new classes are created but the experts are responsible to provide detector specific code to acces the data. It has been decided to use existing code and not to convert it in some OO language. The idea is to rely on well tested materials, well understood and "guaranteed" by the experts.

APPLICATION LIFE CYCLE

Using the terminology proposed by the OMG [5] we can observe the following dominant *iteration strategies* during the development of the Online Event Display:

- 1991 : rapid prototyping
- 1992-1993: evolutionnary iterations (refinement of the model)
- 1994 : incremental iterations. (introduction of new sub-detectors).

The Object Model facilitates the incremental development. The sub-detectors can be described separately; the *Objecteering* translator and the C++ compiler will check that they are consistant with their parents. Then, they will act as expected in the application model.

The evolutionnary iterations are easy to manage because all the application structural model is fully described in the general framework, and only there. If any change occurs in the application requirements, the framework may evolve but the CASE tools will check the consistancy of the detectors descriptions with the new model. Very often, the only action to take at the detectors class level will be a simple recompilation.

The graphical editor is an essential tool to manage the evolution of the model. It provides general and detailed views which are always in strict conformity with the application structure.

TECHNICALITIES

Objecteering is available only on Unix systems (soon on PC). At Orsay, we used it on a DecStation/Ultrix and on HP/UX. It is really effective for the Object Structural Model.

The application was targeted to VAX/VMS system and the major technical problem was the portability of the C++ class libraries used by *Objecteering*.

We are looking at another CASE tool, *Paradigm Plus* [6], which can generate either C, C++ or Eiffel code. It is based on the OMT methodology (Object Modeling Technic) [7], which uses a similar Object Structural Model.

CONCLUSIONS

The use of Object Modeling allows a very flexible development strategy. Incremental and evolutionnary iterations are choosen at the very start, but that choice can be easily reviewed and adapted to different unexpected conditions.

A CASE tool is essential. Several effective tools are available now.

ACKNOWLEDGEMENTS

The authors thank Philippe Charpentier and Philippe Gavillet from the DELPHI-DAS group at CERN for their support and help with this project.

REFERENCES

1. Serge Dû, Christian Arnault, "Experience with Object Oriented methodologies in the new Delphi Online Event Display", CHEP'92 Conference Proceedings, pp. 873-876.
2. P. Desfray, "Ingenierie des objets. Approche Classe-Relation. Application à C++", Masson, 1992.
3. J. Martin, J .Odell, "Object Oriented Analysis and Design", Prentice Hall, 1992.
4. Guy Barrand, "OnX", CHEP'92 Conference Proceedings, pp. 439-442.
5. "Reference Model for Object Analysis and Design", Object Management Group, Document 92-10-1, October 1992
6. "Paradigm Plus User's Guide", January 1993, Technical Publications, Cadre Technologies Inc.
7. J. Rumbaugh, "Object Oriented Modeling and Design", Prentice Hall, 1991.

TOWARDS A 64-BIT VERSION OF SCIENTIFIC LIBRARIES

Perfecto Dipotet and Miguel Marquina
Computing and Networks Division, CERN
CH-1211 Geneva 23, Switzerland

Abstract

While most of the current scientific computers are still 32-bit machines, research and computation requirements, specially in the field of High Energy Physics, already require greater precision. We frame the problem in the context of FORTRAN source compatibility versus object-code performance, and evaluate the automatic promotion/ padding features of the FORTRAN compilers on the different computer platforms available at CERN. We make a comparative study taking into account different factors, including implementation of extended INTEGER and floating-point arithmetic. We propose one of the studied vendors' implementations as the right intermediate choice while waiting for true 64-bit hardware platforms.

INTRODUCTION

The precision requirements of the forthcoming HEP experiments and the problems involved in the production and maintenance of the CERNlib software motivated this study, while one of the authors was in charge of the CERN Program Library Office.

We have analyzed the portability and migration issues of a large FORTRAN library. We have intentionally excluded the problem of data transmission in interlanguage calls (notably between FORTRAN and C), due to the fact that there is no standard mechanism.

CERNlib software has been increasing its use of 64-bit arithmetic in different routines and packages, and notably in the Physics Simulation Montecarlo programs, for several years now. While the CRAY version was supported, we were obliged to maintain dual-path coding (via the PATCHY source code manager [1]) in order to produce arithmetically compatible results regardless of the computer where the Physics code was run. That meant essentially REAL(2) data declarations on 32-bit machines and REAL on the CRAY; 128-bit arithmetic has not been really used in these computations (theoretical physicists and magnet-design engineers have however made extensive use of such level of precision offered by the CRAY hardware).

With the re-introduction of 64-bit native hardware in the computer market by DEC, the software support for all the different architectures becomes the

implementor's nightmare. Not only the two major variants (32-bit vs 64-bit) must be supported, but also the floating-point representation adopted by each vendor is different (IEEE, IBM System/370, CRAY, little-endian vs big-endian byte ordering). Also, IEEE floating-point representation is a 32-bit standard [2], while its 64-bit extension [3] is not yet consistenly implemented.

For this reason, one big step towards the simplification of the maintenance operation would be to support only one variant, namely the 64-bit source for obvious reasons. In that case, 32-bit compatibility must be obtained by finding alternative software options to double the nominal precision.

We will not consider in this paper another problem as relevant, binary compatibility, for which there is yet no universal recipee. There are software packages like ZEBRA [4] which address data exchange across machines. Whatever solution is taken, it must necessarily provide backward-compatibility to preserve the investment made so far in data storage, and this will force a very complex data migration pattern. Even more, it is possible that 64-bit is used only during the computations but not when storing the results (achieving both permanent 32-bit compatibility and data reduction).

CONCEPTS

Most of the compiler developers have foreseen in their design the option to alter the amount of desired precision through software switches, and that in both directions: reducing (16-bit integer arithmetic) and expanding. However this offering is rather anarchic due to the lack of a standard. We are not interested now in reduced precision. Let us introduce two definitions to be used in the next section.

A variable is promoted at compilation time when its storage is doubled and the corresponding arithmetic operations extended.

A variable is padded when its storage is doubled, but the second least-significant part is filled with zeroes. The precision of arithmetic operations remains therefore unchanged.

Now, let us examine the behaviour of the different compilers. Additional definitions not required for the internal consistency of this paper are presented in [5].

ANALYSIS

We have summarized in Table 1 the compiler handling of REAL, REAL(2) and INTEGER variables by a three-character string, where p indicates variable promotion, and 0 padding.

The combination ppp is just the software emulation of all 64-bit operations (both floating-point and integer arithmetic). This mode slows down the computation by typically a factor 2 in numerically-intensive applications.

The next acceptable compromise is pp0, the IBM one; only the floating-point operations are handled with 64-bit precision while the INTEGER arithmetic is still 32-bit and adequate zero padding is made for variable alignment. In this case timing tests show [6] penalties ranging 10-50% compared to the performance of the original 32-bit code.

The minimal requirement in order not to violate the FORTRAN standard is the case 000, were storage consistency is granted (i.e., REAL(2) data takes twice the storage of REAL data) although no improvement in precision is provided.

Brand	Operating System	Fortran Compiler	switches	RDI action
CONVEX	Convex OS 10.x	fc 8.x	-p8	ppp
DEC	AXP/VMS 1.5	DEC-FORTRAN 6.1	/REAL=64/INT=64	p-p
DEC	AXP/OSF 1.3	DEC-FORTRAN 3.3	-r8 -i8	p-p
HP	HP/UX 9.0	f77 9.0	-r8	p--
HP	HP/UX ?	f77 ?	-Wc,autodblpad	pp0
IBM	VM/CMS	FORTRAN 2.x	AUTODBL(DBLPAD)	pp0
IBM	AIX 3.2.x	xlf 2.x	-qautodbl=dblpad	pp0
SGI	IRIX 4.x	f77 3.x	-r8	p--
SGI	IRIX 5.x	f77 4.x	-r8 -i8	p-p
SUN	SunOS 4.x, 5.x	f77 1.4.x, 2.x	-r8	ppp

Table 1. Compiler treatment of REAL, REAL(2) and INTEGER data

All other implementations (p-*) fail to at least handle properly REAL(2) variables, which should be either promoted to REAL(4) (128 bits) or padded.

CONCLUSIONS

The only 32-bit compilers which currently support adequately extended precision are those which perform true 64-bit emulation (CONVEX, SUN) at the cost of reducing the performance by a factor 2, and the IBM compilers where sacrifing INTEGER arithmetic seems to be an acceptable trade-off.

While true 64-bit hardware is not widely available, the best compromise precision/performance is the one currently offered by IBM. Other vendors (HP and DEC) have accepted CERN's feedback in order to implement IBM's implementation in coming versions of their FORTRAN compilers.

ACKNOWLEDGEMENTS

The authors would like to thank for the useful discussions and the comments to the draft of this report, to: R.Brun, F.Carminati, F.James, G.Folger, I.Mclaren and J.Shiers.

REFERENCES

1. J.Zoll, "PATCHY Reference Manual", CERN Program Library 1988.
2. IEEE Standard for Binary Floating-Point Arithmetic (Std 754-1985) SIGPLAN Not. Vol 22 N2 February 1987
3. W. Cody, "A proposed Radix- and Word-length-independent Standard for Floating-point Arithmetic". IEEE-Micro, August 1984.
4. R.Brun, M.Goossens and J.Zoll, "Q100 ZEBRA - Data Structure Management System", CERN Program Library 1991.
5. P.Aubry, C.Guerin, M.Roethlisberger and M.Marquina, "Studies of a 64-bit version of CERNlib", CERN Comp. NewsLetter 207, May-June 1992.
6. P.Aubry, C.Guerin, M.Roethlisberger "Porting of CERNlib/AUTODBL mode on IBM hardware", CERN Program Library internal report, May 1992.

Maintaining a software library at CLEO
— Moving from CMZ to CVS —

Nobu Katayama
Wilson Laboratory, Cornell University
Ithaca, N.Y., 14853, U.S.A.

Abstract

We discuss the maintenance of the CLEO software library. Presently, we maintain the library using CMZ. We are switching to use CVS and other UNIX tools as we become accustomed to UNIX platforms.

INTRODUCTION

The CLEO collaboration consists of 200 physicists from 22 institutions. It has made major contributions to many aspects of b, c, τ and $\gamma\gamma$ physics with precision unmatched anywhere in the world. Because it is an active collaboration, the development and maintenance of the software for the experiment is important. At any given time, we have many developers and users of the software. The event reconstruction processes and physics analyses, as well as detector hardware upgrades, require that the software be organized, modular, reliable, stable and flexible. In the following sections we describe the experience gained in managing the CLEO software libraries and the plan for switching from CMZ to CVS.

LIBRARY MAINTENANCE WITH CMZ

In 1989, the collaboration chose CMZ[1] as a tool to maintain its source code. By 1992 we had developed a system where (1) a librarian maintains all official CMZ files in a central place and accepts submmisions of the changes in a form of CAR files containing new DECKs, (2) the librarian merges the changes into the official CMZ files, (3) the librarian generates the latest object libraries and executables from the CMZ files on several platforms.

The system has worked reasonably well so far. Aside from minor inconveniences, CMZ has been a great tool for maintaining the CLEO library system. The major weak points of our system are: (1) Due to the rush submissions of changes before a data reconstruction pass begins, the executables used for the pass have never been synchronized with ones in the official library. (2) Occasionally more than one person submit changes to the same CMZ file so that the librarian has to manually sort out the changes. (3) The official CMZ files grow fast because CMZ stores a whole copy of each revision of the DECKs. (4) Branching is not possible with CMZ, so we end up having two CMZ files with different names that contain mostly the same source code.

We did not like the following features of CMZ (and/or PATCHY). (a) The syn-

tax of the SELF statement is not logical. (b) Although we use KUIP macros rather than cradle files or pilot PATCHes for an installation, it is hard for some people to understand the mechanism of the installation. (c) The directory structure inside CMZ is limited to the depth of two.

More importantly, in our use of CMZ face the following problems: (i) CMZ does not handle dependencies of "include" files well. The date attribute of the files created by CMZ for compilation is not the date of the last modification. This is because the included files (by SEQ and CDE statements) are expanded. It is thus hard to integrate with "make." (ii) We have too many CMZ files that are dependent on each other through SEQ/KEEP and subroutine calls. Since there are more than one hundred CMZ files in the CLEO, tracking the dependencies is a very time consuming job for the librarian. (iii) CMZ cannot set a version or tag in many CMZ files at once. This is crucial when tagging a release for the reconstruction pass.

LIBRARY MAINTENANCE WITH CVS

As we become accustomed to the UNIX platforms, we would like to develop code in a native shell environment rather than in CMZ. We have searched for a code management system that is freely available and used inside and outside the HEP community. CVS was the clear choice. Major advantages are:

1. It keeps all source files in separate ASCII files, each with its modification history.

2. The date attribute of the source files reflects the date of the actual modification so that the UNIX make utility functions as designed.

3. It can tag the current version of the entire library with a symbolic name so that at any later date, one can reproduce the named version such as the version used for a particular event reconstruction pass.

We also decided to follow the standards of GNU software from the Free Software Foundation and use autoconf and gmake as our package-building standard.

The purpose of using these tools is to minimize the revision and dependency problems in the multiple platform, multiple developer environment. These tools help us in the following ways:

1. We will be able to build the most up-to-date libraries and executables using the latest source code from the developers in the shortest time by simply issuing the gmake command.

2. At the same time, we can reproduce previous results (of a reconstruction pass/analysis) by regenerating the source code at the time of the pass.

3. Multiple developers can work almost independently even when they are working on the same source code.

4. In general, the developers and librarians do not have to worry about dependencies of the updated include files and object libraries. There are some exceptional cases where they have to manually take care of the dependency.

TOOLS

CVS is a version control system. Using it, one can record the history of one's source files. It keeps a single copy of each source file in a "repository." The copy contains all the information to permit extracting previous versions at any time, based on either a symbolic revision tag, or a specific date.

CVS is a front end to RCS. CVS extends the notion of revision control to a hierarchical collection of directories. Managing million lines of code in hundreds of directories such as the entire CLEO library in a single repository is not a problem with CVS. CVS provides the functions necessary to manage software releases and to control the concurrent editing of source files among multiple software developers.

CVS has been used widely both inside and outside HEP community.

GNU make is compatible with the UNIX utility make but has convenient extensions that lead to much simpler Makefiles. The make utility automatically determines which pieces of a large program need to be recompiled, and issues the command to recompile them in order to get the updated version of the program.

cpp is a macro processor that is invoked automatically by C/FORTRAN compiler to transform the source code before actual compilation. We use cpp directives #include, #define, #if, #else and #endif to handle conditional compilation and the inclusion of the files which define common blocks.

autoconf generates automatic configuration scripts called configure. The configure script then generates Makefiles. The configure script, together with gmake let us compile the software package in a different directory from the one that contains the source code. This in turn allows us to compile the package for several architectures simultaneously from a single copy of the source code and keep multiple sets of the object files.

makedepend reads each source file in sequence and parses it like a C preprocessor, processing all #include, #define,,, directives. The result is a description of the dependencies in the Makefiles. On platforms where the compilers support "−M" option, we use the compilers rather than "makedepend" for improved consistency in the dependencies.

CVS, GNU make and autoconf come from FSF, cpp from vendors. makedepend comes with X.

IMPLEMENTATION DETAILS

From CMZ to CVS

To help quickly convert from the current library system to the new system, a program **car2cvs** was written. This program reads a CAR file, sets up the directory structure, creates all Makefiles and a few other administrative scripts, and "imports" the files into the CVS repository. One needs to run this program once to convert.

car2cvs maps the PATCHes to subdirectories and the DECKs to individual source files. +SELF statements are converted to #if/#endif directives and +SEQ, +CDE to #include directives. Some of KUIP macros are translated into Makefiles. Most CMZ files in the CLEO library have a simple and standard structure. i.e. all DECKs in all PATCHes are compiled and put in an object library (archive) file. For these standard CMZ fles, the Makefiles are created automati-

cally.

- To maintain the parallel structure of the new and old system for a while, a program **cvs2car** was written to convert back a directory stracture to a CMZ file.

- car2cvs[2] was written in C++ because of a convenient String class. It could have been written in any other language, such as awk and perl. It is rather a long program because it contains templates for several makefiles and other scripts.

Code development procedure under the old system

A person who wishes to modify an existing library copies the CMZ file from the standard location where the latest version is kept. (If changes by other people are pending one might start from a soon-to-be-obsolete CMZ file.) Using CMZ, he modifies DECKs, compiles, links an executable program and tests it. Once he is satisfied, he then issues a "version" CMZ command, extracts the modified DECKs using a "ctoy" CMZ command into CAR format file. (Some people forget to send all the changes.) He then sends an E-mail to the librarian notifying where the CAR file is.

On Tuesdays, the librarian reads his mails and merges all changes he has received that week into temporary CMZ files. If more than one person has sent changes to the same CMZ file, the librarian needs to sort out the changes. (A librarian who is not familiar with the source code might have to merge the changes.) He then recompiles all CMZ files that are modified and that would be affected by the changes and rebuilds executables that would be affected by the changes. (It is difficult to determine exactly which CMZ files he has to recompile or which executables he needs to rebuild because the dependencies are not described anywhere.) If the standard tests successfully finish he copies the temporary CMZ files into the official area. On the second Tuesday of a month, he copies the entire library into a new area (clyymm where yy is the year, mm is the month), and change the softlinks "clib" (pro) and "dlib" (new) before updating the official CMZ files. Notice that there will be more than one CMZ file of the same source code in the clyymm areas.

Code development procedure under the new system

A person who wishes to modify an existing library issues a "cvs checkout" command to make a copy of the latest source code in his working area. Then using shell commands, he modifies source files. He compiles, links an executable program by simply issuing "gmake" command and tests it. Once he is satisfied, he issues a "cvs commit" command.

On Tuesdays, the librarian issues a "cvs update" command in /cleo/clyymm /cvssrc area. (This could be done automatically with cron.) He then recompiles all necessary portions of the source files and rebuilds all necessary executables by simply issuing a "gmake" command. (This could be done automatically with cron as well.) If the standard tests finish successfully, he then types "gmake install" and updates the official library area from the "build" area. On the second Tuesday of each month, he copies the entire library into a new area, and change

softlinks "clib" (pro) and "dlib" (new) before updating the official area. Note that he does NOT copy the repository.

The new system is much simpler and more reliable.

- It is simpler and more automatic both for the developers and librarian.
- We do not have to worry about dependency of the source code. "makedepend" will take care of it.
- Updates from the multiple developers are taken care of by cvs. The conflicts are solved by the developers not by the librarian.
- When building an executable for a reconstruction pass, one would "cvs checkout" the entire library, then issue "cvs tag PASS_VERSION." This way at any later time, anyone can recreate the same executable as far as the CLEO source code is concerned.

CONCLUSIONS

- CVS, gmake, cpp and autoconf together make a nice code development and/or library maintenance environment for a large collaboration.
- The new system is simpler and more reliable than the CMZ based system.
- We are in transit now. We expect to complete the switch by this fall.
- car2cvs provides a means to convert from the old to the new system easily.
- For those who like to continue using CMZ, car2cvs will also allow them to submit library changes in CAR format files.
- A few (non-UNIX) platforms do not support CVS, gmake or autoconf. Fortunately features of gmake and autoconf permit us to do cross compilation. We will be using cross compilation for the VAX, where we still maintain the CLEO libraries.

ACKNOWLEDGEMENTS

The author thanks Lawrence Gibbons of Univ. of Rochester and Simon Patton of Univ. of Minnesota for their collaboration with the implementation of the new library system.

REFERENCES

1. CMZ User's Guide & Reference Manual, CodeME S.A.R.L.
2. You will find information on car2cvs from http: //w4.lns.cornell.edu/public/car2cvs/car2cvs.html. Car2cvs is available through anonymous ftp from ftp@freehep.scri.fsu.edu. The .tar file contains complete documentation that describes the details, including the directory structure of the CLEO libraries and the contents of Makefiles and configure.in files.

Use of object-oriented techniques in a beam-line control system

D.R. Myers and W. von Rüden,
ECP Division, 1211 Geneva 23, Switzerland.

H. Butler,
Los Alamos, U.S.A.

J. Yang,
University of Science and Technology, Hefei, China.

Abstract

We describe the use of object-oriented programming in the control and data-acquisition system for the upgraded CERN neutrino beam-line. C++ in conjunction with Posix threads running under Lynx-OS have been used in several front-end PCs. These communicate using Remote Procedure Calls over ethernet with a workstation running the commercial supervisory package, FactoryLink.

1. The WANF Controls Project

Two experiments at CERN, CHORUS and NOMAD, are using the rejuvenated West Area Neutrino Facility (WANF) and the original beam-line control system, based on now obsolete machines, has had to be replaced [1]. In order to achieve this an industrial control package, FactoryLink [2], has been installed on a powerful HP workstation and interfaced to the existing CAMAC hardware via two front-end machines, one for beam control and one for flux monitoring. The front-ends are industrial PCs running the LynxOS real-time UNIX system [3]. On both PCs an identical software *daemon* is installed which receives commands from the workstation and controls the CAMAC via a VIC bus. The daemon is written in an object-oriented fashion and is matched to the hardware configuration at run-time by instantiating the appropriate set of objects which have been defined in a Filemaker-Pro database [4].

FactoryLink consists of a real-time kernel to which is interfaced a wide range of off-the-shelf tasks for alarm handling, data logging, trending, timing and so forth. The kernel incorporates a database consisting of named *tags* which store primitive data types. Tasks may declare an interest in any set of tags and are informed if the value of a tag is changed.

A powerful graphic user interface, including the possibility for picture animation based on the value of a tag, may be constructed using an off-line drawing tool which interfaces to the screen via the X-Window protocol. Apart from an interpreter available for mathematical and logical algorithms, FactoryLink tasks are configured in a completely data-driven fashion. Although software exists to connect Programmed Logic Controllers, we were obliged to write tasks which interface to the front-end computers using our own protocol based on Remote Procedure Calls (RPCs).

The architecture of the system was fixed in the Autumn of 1992 and the implementation started in October of that year. A version of the complete system, which has been well received by physicists from the experiments, was available twelve months later. The initial manpower required was the equivalent of three man-years, with perhaps another eight man-months added for improvements. We believe that this figure is extremely low compared with the effort which would have been required using tradi-

tional methods to produce a system of comparable sophistication.

2. Class Design

Within each front-end PC there is a process running which is the recipient of RPC calls from the FactoryLink interface task. The calls contain *messages* for particular target objects specifying what *action* should be performed by the object and containing any necessary data. Thus, as an example, a message might be sent to a particular magnet power supply object requesting it to set a current of a certain value.

The system has classes which can be categorized into three different groups:

C-1 Specialized classes which correspond to actual WANF beam-line components.

C-2 Classes which model CAMAC branches, crates and modules, such as input registers or ADCs. These are classes dealing with the interface between the equipment and the computer.

C-3 Finally there are classes which also deal with the computer interface but which model the functionality at an abstract level. Classes in this group include binary input and output bits, analogue sensors, and so forth.

For the first two classes the strategy adopted has been to map software objects to real-world objects as closely as possible. It is interesting to note that failure to do this in a few cases led to problems which later had to be corrected. The utility of the last category is a technical one and these classes turned out to be more useful than might at first have been imagined. This is best illustrated by an example.

In order to switch the polarity of the beam-line it is necessary momentarily to close a switch. Thus, a Polarity Controller object has a pointer to an `OutBit` object which controls the switch, but it does not need to know if this is implemented in CAMAC, VME or anything else; the important thing is that an `OutBit` abstracts all the properties of a generic switch. In fact an `OutBit` object contains a pointer to a hardware module and an index number in case the module contains several bits. For C++, the implementation language, it is possible to define the type of a pointer so that if it is supposed to point to an `OutBitModule`, a module containing output bits, one could never load it with a pointer to a module containing ADCs, for example. Thus, the system is type safe and also run-time configurable, because the pointer may be loaded with the address of any class of object which inherits from `OutBitModule`.

It is in the class implementing the hardware module where the real work takes place. However, it is interesting to note that the actual code executed will be selected only at run time. In object-oriented terminology this feature is called *polymorphism*. For example, all classes inheriting from `OutBitModule` must provide a function called `writeBit(index)`. The actual version executed is selected depending on the class of the particular object pointed to. If a hardware module is purchased from another manufacturer then a new class would be written to support it, but no other changes are needed anywhere in the system.

3. Implementation Issues

3.1 Threads

As a consequence of using objects which, at the current state-of-the-art, cannot easily be distributed over several processes, the problem arose as to how to handle asynchronous events. This has been solved by using Posix Threads [5] which are available under LynxOS. Threads are essentially light-weight processes running within the same address space. As an example, if an object decides it must wait for an interrupt it spawns a new thread to which control is transferred whilst the interrupt is pending. Meanwhile, the code in the initial thread can continue with something else.

Unfortunately Posix threads know nothing about objects but, by a subterfuge, it is still possible to transfer control to an object member function. Of course, if code in several threads needs to access the same variable then this must be protected by using mutual exclusion semaphores (Mutexes). With these two provisos, threads were found to be an elegant and powerful facility.

3.2 Object-to-Tag Mapping

No equally elegant way was found to deal with the problem of mapping PC objects with complex behaviour onto FactoryLink tags. Tags support only the functions read and write, whereas objects have functions (often called *methods*) which can respond to all sorts of commands. To illustrate this, consider a magnet which can be switched on or off or be set to a particular field. This could be modelled by a class with methods `Switch(onOff)` and `SetField(value)`. However, as FactoryLink does not support complex data structures, the magnet has to be represented by two independent tags (one boolean and one real).

The solution adopted was to impose a convention on all tag names such that characters after a "$" indicate a command. Thus, in our example we would have the tags:

```
bendingMagnet$onOff, and
bendingMagnet$setField
```

The first part of the name is transmitted to the daemon where it is looked up in a hash table in order to find the identifier (address) of the corresponding object. At this point, instead of being able directly to call the correct function, it is necessary to pass via an intermediate stage which first has to find the correct destination based on the command field.

3.3 The DataBase

Whilst the front-end daemons have the capability to instantiate objects of any class, a mechanism was required actually to specify these objects. Although an object-oriented database would undoubtedly be a better long-term solution, we used Filemaker-Pro on a Macintosh which was fast and effective with an excellent user interface.

The database has an item corresponding to each object specifying its class, name and parameter set. From this it is straight forward to produce configuration files for both PCs as well as wiring lists for the technicians. Files are also produced for FactoryLink listing all the tags with which the PCs must communicate.

3.4 The Survey Class

One of the jobs of a control system is to monitor various parameters to ensure they stay within pre-defined limits. FactoryLink provides a task to do this, but the question arose as to how the information could be collected. The solution adopted was to provide a `Survey` class, instances of which maintain lists of objects to monitor.

When an asynchronous survey object is started it spawns a new thread and transfers control to a member function which alternately sleeps and then sends a message to all the objects on its list. This asks *Have-You-Changed?* Each object must know how to compare its current value (or values) with its previous state and then reports the changes.

3.5 Object Independence

A control system may have many objects which supply values read from different hardware sources. Let us say that these are read out by one of the `Survey` objects. A design feature of the model adopted was that each channel appears as an independent object which is addressed by name. The user need not even know how it is implemented and this makes changes to the system very easy; all one needs to do is to modify the database so that the abstract object becomes associated with a different module object of the same base class.

Now, what happens when several objects are surveyed which are situated in the same physical module? To discover if their values have changed separate read requests will be sent to the module. This might be merely a slight inefficiency if all the values could be obtained by a single hardware access. However, if the action of reading a value clears all the hardware registers, then only the first value obtained will be meaningful.

To solve this the survey process was split into two parts. When any object adds itself to a `Survey` the module in which it is situated is added to a unique list. When a survey is started each object in the list of modules is asked to refresh all its values and store them locally. When the objects to be surveyed later obtain their current values these are then available without a new hardware access. This scheme has a number of useful features in that it works with hardware which does a clear on read, it is efficient and, finally, it guarantees that all values from a single module are obtained simultaneously. The last feature may be essential for data which are produced by a unique event.

3.6 Multiple Inheritance

Multiple inheritance was found to be essential for several of our classes. For example, an `ADCModule` is both a CAMAC module as well as a set of ADC registers. Multiple inheritance provides type safety in this case and also forces concrete subclasses of `ADCModule` to implement the required virtual functions.

4. Conclusions

Even though the C++ compiler available under LynxOS was old and rather unreliable, the use of object-oriented techniques for this application has been very effective. About 50 classes, corresponding to more than 20k lines of code, have been written, debugged and documented in twelve months. Of these classes 20 are specific to the WANF beam-line, but the rest could be re-used in other applications.

Object-oriented code is indeed modular and it has been straight forward to add classes for new types of hardware. Adding objects for which classes already exist requires no coding changes whatsoever. It is even possible to add new objects without interrupting the daemon.

In summary, we have used commercial tools wherever possible and object-oriented programming for that part of the project for which a commercial solution was not available. This strategy has enabled us to build the control system on time and with a quality we could not otherwise have hoped to achieve with the time and man-power available.

5. References

1 H. Butler, D.R. Myers, W. von Rüden and J. Yang, *Beam-line Operation using an Industrial Control System and Distributed Object-Oriented Hardware Access*, Real Time'93 Conference, TRIUMF TRI-93-1, Vancouver, Canada, June 8-11, 1993, pp 180-183.

2 FactoryLink IV Overview, United States Data Corporation, 1551 Glenville Drive, Richardson, Texas 75801, U.S.A.

3 LynxOS User's Manual, Lynx Real-Time Systems, Inc., 16780 Lark Avenue, Los Gatos, California 95030, U.S.A.

4 FileMaker-Pro User's Guide, Claris Corporation, 5201 Patrick Henry Drive, Santa Clara, California 95052, U.S.A.

5 IEEE Draft Standard For Information Technology -POSIX- Part 1: System Application Program Interface - Amendment 2: Threads Extension, P1003.4a/d8, October 1993.

REMOS: A Portable Object Oriented Environment for Multiprocessor Real Time Applications

Giuseppe Carnevale
Università degli Studi di Napoli "Federico II" - CERN
Jaap Panman - CERN
Fabio Riccardi
Università degli Studi di Napoli "Federico II" - INFN Napoli - CERN

Abstract

REMOS is an object oriented multiprocessor environment that extends the inter process communication facilities of a single processor multi tasking kernel. A micro kernel approach has been followed to achieve modularity and real time performance. Object oriented design has been extensively used across the whole system, obtaining a high degree of integration with the native operating system.

Introduction

REMOS is an object-oriented environment aimed to build distributed real time data acquisition (DAQ) systems. Different paradigms have been proposed for distributed object-oriented programming, each with advantages and disadvantages, according to the kind of application they were used for. We chose a distributed memory model, as we wanted to achieve the maximum possible performance with the hardware that is currently available for DAQ, typically VME and VSB bus based systems connected in a network of VIC (VME Inter Crate) buses.

This system was used at CERN (European Organization for Nuclear Research, Geneva, Switzerland) to build the DAQ system of the CHORUS experiment [1; 2], based on a network of 50 processors. Remarkable results have been obtained in terms of system performance, stability, scalability and maintainability.

Even though many of our design choices have been strongly influenced by this hardware architecture, we believe that our work has a degree of generality that makes it applicable to much larger contexts.

The distributed OS paradigm

Our main design goal was to write high performance multiprocessor parallel applications, extending the inter process communication facilities offered by single processor multitasking real time operating systems. For this purpose we designed a run time support system, which, running on a single CPU, disciplinates the access to the system's shared resources, and offering a semaphore-like set of distributed synchronization primitives.

We introduce a distributed operating system paradigm, enforced in our system by an object oriented approach, encapsulating low level details of interprocess communication into specific

classes, and building other classes that inherit their basic behavior.

Micro kernel approach

The new wave in the design of modern operating system (OS) is definitively the so called *micro kernel approach* [3; 4; 5]. This approach at building OSs consists in making a very small nucleus that only handles the most basic OS functions: memory management, task management and inter process communication (IPC).

Given these basic functionality a full featured OS is built by adding sever processes that export specific functionality (e.g.. file system support) that are invoked by means of an efficient IPC facility.

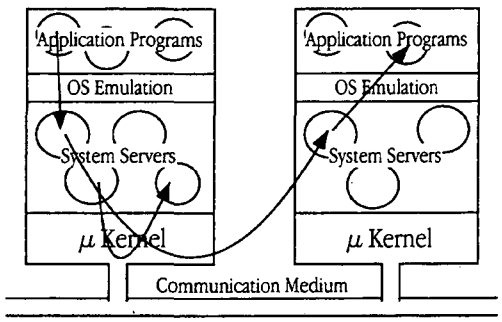


Fig.1 : The μkernel approach.

The major benefits of this approach are:

- simplicity: the code that resides in the kernel is usually difficult to write and debug, debugging user programs is instead easy and a variety of tools are available for it;

- configurability: server processes can be modified and located everywhere, in this way is extremely easy to incrementally add services to a system, and as their invocation is done via IPC, servers can be located on different machines, gaining at once the possibility of having a distributed OS;

- real time performance: the fact that all the I/O and other heavy processing is done out of the kernel, makes them preemptible at any time greatly increasing system responsiveness and eventually gaining real time performance.

Another important feature of μkernel systems is the possibility of handling interrupts from user programs (you don't need a kernel based driver to handle them).

Given these basic guidelines we used the OS9 real time kernel that runs on most VME controllers as a μkernel, we extended it by means of server tasks.

As a basis of all our system we designed a special OS9 driver that allows user processes to install interrupt service routines (ISR). They can synchronize with interrupt requests and the data that the ISR collects at interrupt time is automatically buffered and delivered to the user process at synchronization time.

A simple port based synchronous IPC mechanism has been added, to route messages across different CPUs. It has been implemented on a variety buses, by simply adding a proper ISR to those installed into a couple of interrupt driven router processes, called bridges (send and receive).

The REMOS server

The heart of our system is the REMOS server. This tasks provides a semaphore/queue like synchronization facility and the allocation of distributed memory resources (e.g.. VIC bus memory), user processes communicate with the server by means of RPC calls, completely unaware of its actual location.

Different instances of the REMOS server can administer different resources, therefore allowing to easily partition large systems to achieve scalability and good performance (see fig. 2).

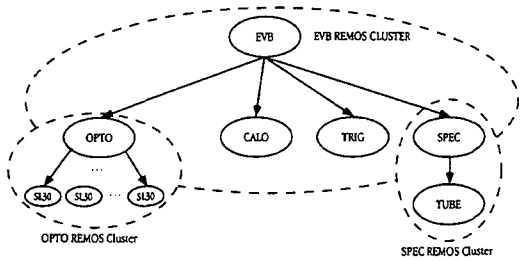


Fig. 2: Example of clustering in the CHORUS experiment DAQ.

Object oriented design

We designed abstract classes defining basic interfaces for specific system functionality, and daughter classes that inherit the interface, regardless of their specific implementation.

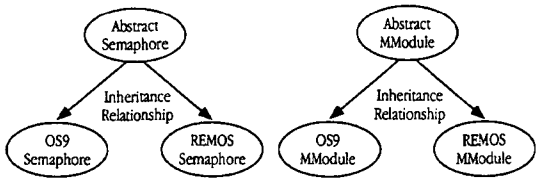


Fig. 3: The REMOS object oriented interfaces.

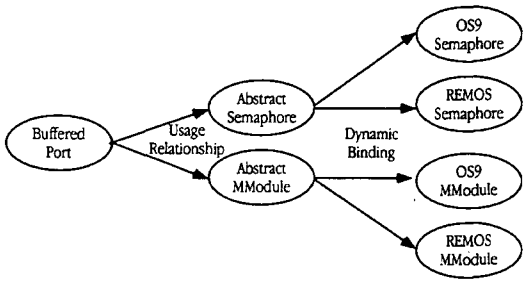


Fig. 4: Building generic high level objects.

This approach allows writing generic code that only relies on the abstract interfaces, that can be used in different contexts without modification, just changing the actual instances of the low level objects, thus ensuring a high degree of portability of the user code, that becomes totally independent from the operating system (see fig. 3 and 4).

All the RPC calls to the REMOS server are hidden into wrapper objects, that give to the calling process the illusion of direcly invoking a system call, these objects make a local image of the temote service, and are often called proxies in the literature [6].

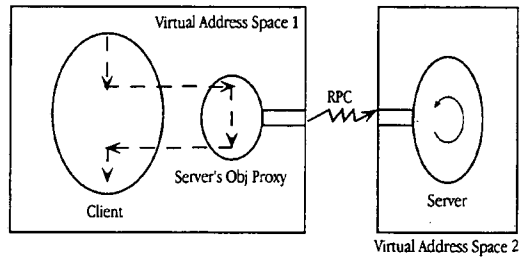


Fig. 5: Invoking REMOS services through a *proxy* interface object.

In this way we built a class library that can handle semaphores and shared memory, using the native operating system capabilities for objects shared among programs running on the same CPU, and using those offered by our run time system for programs running on different CPUs. With these basic tools we built a hierarchy of communication classes implementing more sophisticated programming paradigms (such as monitors and message passing), once more benefiting from the abstract interface approach of the underlying objects, as it is not necessary to produce different versions of them for single or multi processor environments.

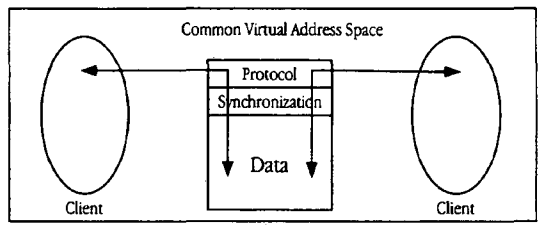


Fig. 6: An implementation of *monitor* objects in REMOS.

As processes refer to REMOS objects by name, and the REMOS server itself is modeled as an object, different instances of it can coexist, therefore permitting the clustering scheme sketched above.

Another advantage of the object oriented approach to these communication facilities is given by the possibility of adding more control to their usage, permitting a higher level of reliability of the whole system.

Performance

We have started to write a set of benchmark programs to measure our system's performance, some preliminary results show a very good performance (see tab. 1).

local OS9 Signal	17 µs
local OS9 Wait	18 µs
local REMOS RPC	190 µs
remote RPC	700 µs
remote Signal	200 µs
remote Wait	700 µs

Tab. 1: REMOS benchmark times, preliminary results (25 MHz CES FIC8234 + CES VIC 8251).

Conclusions

With the REMOS system we showed that modern OS technologies can be successfully used to implement complex real time DAQ systems.

This mixed object oriented and µkernel modular approach allowed us to develop the whole system in a very short time (less than one man-year), achieving at the same time a very high level of robustness.

The µkernel approach also had the benefit of preserving the original real time responsiveness characteristics of the underlying OS.

We believe that these encouraging results should lead towards the adoption of more modern kernels (such as Real Time Mach or CHORUS), that already provide these characteristics at their basis.

Furthermore the scalability that can be achieved in this way allows to use the same basic kernel on embedded VME systems and in large mainframes, leading towards an approach *at the large*, on the whole DAQ system, (file servers, graphic interfaces, ...) and not limiting it just to the on-line front end.

References

1 N. Armenise et al., CHORUS Collaboration, CERN - SPSC/90-42; M. de Jong et al., CHORUS Collaboration, CERN - PPE/93-134

2 G. Carnevale et al, "The CHORUS Data Acquisition System", to be published in the proceedings of the CHEP'94 conference, San Francisco, USA, 1994.

3 D.R. Cheriton, "The V Kernel: A Software Base for Distributed Systems", IEEE Software, 1(2):19-42, April 1984.

4 M. Rozier et al., "The CHORUS Distributed Operating System", Computing Systems, 1(4), 1988.

5 M.J. Accetta et al., "Mach: A New Kernel Foundation for UNIX Development", Proceeding of the Summer 1986 USENIX Conference, pages 93-113, July 1986.

6 R. Lea et al., "COOL: System Support for Distributed Programming", Communications of the ACM, pages 37-46, Vol. 36, N.9, September 1993.

USE OF OBJECT ORIENTED SOFTWARE METHODOLOGY FOR DESIGN OF EXPERT/CONTROL SYSTEMS FOR HEP: THE ZEUS EXPERT SYSTEM EXPERIENCE

Mariusz Flasiński*
Ulf Behrens, Lars Hagge, and Kars Ohrenberg
Deutsches Elektronen-Synchrotron
Notkestrasse 85, 22603 Hamburg, Germany

Abstract

ZEX is an expert system, which is designed to support operation of the ZEUS experiment. Experience gained in the OO analysis and design of the system is summarized in the paper.

INTRODUCTION

The Expert System of the ZEUS experiment, ZEX, is a distributed real-time diagnostic/control system, which is designed to help on-line operators to keep the experiment in optimum running conditions [1]. ZEX was developed using OO Analysis, Design and Programming [2]. This approach proved to be a powerful design/software-building paradigm. However, it had to be enhanced by advanced techniques of designing expert systems. Sharing an experience gained and giving methodological guidlines is the aim of the paper. We hope that it can be helpful in constructing expert systems being developed for HEP experiments (reported recently in the literature) as well as complex control systems for HEP.

OO AND AI PRINCIPLES IN ZEX DESIGN

Major principles of the Booch Object Model [3] were used for decomposing of ZEX. The abstraction principle allowed us to modularize the system according to various aspects of running the experiment (aspects of: Run Control (RC), Slow Control (SC), Data Acquisition (DAQ), etc.), and according to various layers of data processing (decision-theoretic pattern recognition of monitoring data [4], syntactic pattern recognition in time series, and high-level reasoning in symbolic space). This decomposition influenced a choice of an advanced AI method of expert systems design called the Blackboard Architecture [5]. In this approach, a system is partitioned into: hierarchically-organized encapsulated Knowledge Sources (KSs) running under the Scheduler, and a global hierarchical data structure, called the Blackboard. This approach is very efficient for applications, in which the problem space is large and factorable along some number of dimensions. Our experience is that it can be successfully used for design of real-time expert systems in HEP.

STRUCTURAL DATA VIEW

Expert system knowledge base is a combination of data structures, i.e. entity- level knowledge, (in the ZEX case, kept in the Blackboard), and interpretation procedures, i.e. problem-solving knowledge, (in the ZEX case, Knowl-

*on leave from Department of Computer Science, Jagiellonian University, Cracow

edge Sources). Both data structures and Knowledge Sources were defined as a hierarchy of classes with Class Diagrams. Multiple redefining of Class Diagrams during iterative development of ZEX showed that an extensive use of a full OOAD potential leads to a well-partitioned maintainable system. Three kinds of relationships were used: subclass/generalization,"using"/association and instantiation. System (ZEUS), Subsystems (e.g. SC), and Subsystem Components (e.g. CALo- rimeter SC) are subclasses of the Blackboard Object class. Relations among them were described with "using" relationships (objects of higher-level classes, e.g. SC, "use" results given by lower-level ones, e.g. CAL SC). This multidimensional frame structure allows one to treat each detector component according to various aspects of its functionality (as a SC component, DAQ component, etc). Class instantiation relationship was used for typing classes (e.g. Monitoring Data Ring keeping data as a function of time is typed as Ring Buffer). Knowledge Sources of two general classes were used : Production System Class [6], and Pattern Recognizer Class. (The latter class was used because of the good experience with combined syntactic [7] and semantic [8] schemes for recognizing distributed system status and its resource capabilities [9].)

BEHAVIOURAL VIEW

State Transition Diagrams, STD, were applied for modelling the behaviour of ZEX. "The state explosion" is the main disadvantage of this technique in case of using one STD for modelling the whole system behaviour. Therefore we limited the STD's use to single classes. For pattern recognition-based Knowledge Sources, STDs model both the general behaviour of the class (to identify its services) and behaviour of particular objects of the class. The ZEX Blackboard and highest-level interpretation procedures (KSs) were coded with the RTworks [10] real-time expert system shell supporting OOP. RTworks Knowledge Sources are sets of rules of the form: "<activation context>: if <conditions> then < *actions* >". Modelling of ZEX Behavioural View showed that State Transition Diagrams are adequate tools for designing rule sets, if states are defined as <activation contexts>, events activating transitions as <conditions>, and actions associated with transitions as rule <actions>.

ARCHITECTURAL VIEW

The Architectural View of our Blackboard - based expert system was described with Object Diagrams. The higher - level objects ("the knowledge sources", "the blackboard", "a data monitor" objects, and "a user interface" objects) communicate with the messages. The Architecture View modelled with the OO approach was then mapped to the architecture of the RTworks shell supporting the message passing system for communication among its processes: real-time data acquisition processes (RTdaq), inference engines (RTie), and point-and-click color human-computer interface processes (RThci).

CONCLUSIONS

The OO approach to the system analysis and design prooved to be a powerful software building paradigm. It al-

lowed the small group to design and implement the complex diagnostic/control expert system working in the context of hard real-time constraints. The strict encapsulation is the only OO principle, which is not observed at all levels of our hierarchical system. However, this is a common experience of the real-time expert system designers community that strict encapsulation makes large expert systems execute too slowly [11].

REFERENCES

1. U. Behrens, M. Flasiński, L. Hagge, K. Ohrenberg, "ZEX - An Expert System for ZEUS", Proc. 8th Conf. Real-Time Comp. Appl. Nucl. Particle Plasma Phys., June, 1993, Vancouver, Canada, pp. 168-173.
2. M. Flasiński, "Further Development of the ZEUS EXPERT SYSTEM: Computer Science Foundations of Design", DESY Reports, in print.
3. G. Booch, "Object Oriented Design with Applications", The Benjamin/Cummings Publ. Comp., 1991.
4. U. Behrens, M. Flasiński, L. Hagge, "ZEXP - Expert System for ZEUS: Problem Analysis, Theoretical Background and First Results", DESY Reports, DESY 92-141, October 1992.
5. L.D. Erman, F. Hayes-Roth, V.R. Lesser, D.R. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Solve Uncertainty", Computing Surveys, vol. 12, 1980, pp. 213-253.
6. P. Jackson, "Introduction to Expert Systems", Addison-Wesley, Reading, MA, 1990.
7. M. Flasiński, "Parsing of edNLC - Graph Grammars for Scene Analysis", Pattern Recognition, vol. 21, 1988, pp. 623-629.
8. M. Flasiński, G. Lewicki, "The Convergent Method of Constructing Polynomial Discriminant Functions for Pattern Recognition", Pattern Recognition, vol. 24, 1991, pp. 1009-1015.
9. M. Flasiński, L. Kotulski, "On the Use of Graph Grammars for the Control of a Distributed Software Allocation", Computer Journ., vol. 35, 1992, pp. A165-175.
10. Talarian Corporation, Mountain View, USA. "RTworks v. 2.1 User Manual", December 1992.
11. P. Harmon, "G2: Gensym's Real-Time Expert System", Intelligent Software Strategies, vol. 9, 1993, pp. 1-14.

DBS - An rlogin Multiplexor and Output Logger for DA Systems*

Gene Oleynik, Laura Appleton, Lourdu Udumula, Margaret Votava

Online Systems Department
Fermi National Accelerator Laboratory
P.O. Box 500

* This work is sponsered by DOE contract No. DE-AC02-76CH03000

Abstract

DART Bootstrap Services (**dbs**) is the first component of run-control for the DART Data Acquisition system - the DA for the 96' round of experiments at Fermilab - though it has potential usefulness as a powerful tool in other distributed applications.

dbs is an rlogin session multiplexer. It allows a user, running a single program, to start up any number of remote login sessions, feed shell commands to them, and collect their output into a single (or multiple) logfiles (a server keeps the sessions open and collects their output). From this program, any session can be attached to interactively so it appears just like an rlogin session - **dbs** becomes transparent. When finished with this interactive mode, the user can escape back to **dbs** and attach to a different session if so desired. Among many other useful features, **dbs** supplies a mechanism for cleanup (deletion) of all processes created under a session, allowing a fresh start.

1 Introduction

dbs is a Unix based tool designed to meet certain needs for working with a fully distributed system. Though it was developed for DART data acquisition systems, its domain of applicability goes beyond this specific application. We have taken care that the software product is generic enough to leverage its use across a variety of potential applications.

2 Motivation and Requirements

DART data acquisition systems consist of a number of intelligent computers, running a number of operating systems and a variety of applications, distributed across a LAN [1,2]. Many of these applications are replicated across nodes. Taken together, these cooperating applications make up a DA system. The specific applications that make up the DA and the architecture of the system, vary from experiment to experiment. **dbs** addresses the problem of how to start-up applications for a given experiment in a uniform manner such that the system comes up correctly as a whole, or is "bootstrapped".

There are several pertinent factors for the solution of this problem that we folded into **dbs** requirements [3]:

- It must be able to start-up all applications for the system from a source file on a single node with minimal user interaction.
- It must be easily tailorable with different sets of applications for different systems.
- It must be able to start-up any application. There should not have to be anything special added to an application.
- It must handle replication of applications efficiently.
- It must provide some mechanism to synchronize the start of client-server applications.
- It must be able to start-up applications on multiple operating systems, including Unix, VMS, and VxWorks (the real-time operating system we use on embedded processors), and be easily portable to others.

- It must be able to start applications in parallel across nodes so that the system can be bootstrapped quickly.
- It must provide a mechanism to remove all applications so that a clean start can be made.
- It must record information so that start-up failures are diagnosable, and collect the standard output and standard error from applications.
- It must use as little system resources as possible, and where possible, make use of existing system based software

3 dbs

We found the most effective solution to the bootstrap problem that meets the above requirements to be an application built on top of the ubiquitous, commercial network package, rlogin, which allows authorized users to login to a system without a password. rlogin software is bundled with the system on all Unix platforms and is available under VMS (e.g. via MULTINET) and VxWorks, as well as many other systems. In essence, no remote software needed to be written for **dbs**. **dbs** consists mainly of "host" software that runs on a single node to "launch" applications through rlogin connections to other nodes From here on, "host" will refer to the computer on which **dbs** runs to launch applications, and "remote" will refer to nodes on which the applications are started, which can include the host node.

The host **dbs** software consists of an rlogin multiplexing server and a user interface application, referred to as the "client", which drives this server [4]. Through the client, the user can create named rlogin "sessions" to remote nodes, non-interactively send commands to them to be executed, or connect "transparently" to any one of the sessions. In the latter case, the **dbs** client is transparent to the user, that is, it appears as a normal rlogin to the remote host, except that an "escape" breaks the user back to the **dbs** client. The client use the TCL command line interpreter (free software) [5] and can be driven from a command file or interactively. The **dbs** client program can be exited at any time, and restarted later - the server keeps the sessions that are open and available.

The rlogin multiplexor server manages the bookkeeping of the sessions and routes session input and output between the remote sessions and the client application, or, as will be seen later, it can route session output to files on the host.

4 Session Management and Commands

A **dbs** user creates an rlogin session to a remote node with a command that associates a "session-name" with the session. The session-name is used to identify the session in all further transactions. As will be seen, his simple session-name abstraction has powerful consequences.

Commands are sent to a session with the "s_command" client command. This command is asynchronous in the sense that it is sent to the sessions standard input without waiting for an acknowledgment, permitting commands to be executed on different sessions in parallel. A command can be broadcast to all sessions by leaving the session-name field blank; we have plans for more powerful wildcarding.

5 Session Output Management & Logging

The only feedback from remote sessions is their output through standard out and standard error. One implication of the requirement that **dbs** be able to start "any old application" is that **dbs** has no control over its output, which may contain important information. Therefore, **dbs** was designed to optionally capture session output and write it to a file.

dbs buffers the output from its sessions in the server. The buffer size can be specified per session when the session is created (a default is provided).

If no output file is specified for the session, with the s_fileout command, then if the session's buffer fills, the output will back up to the remote session and eventually block the application producing it, until the session is

attached to interactively through **dbs**, or an output file is specified for the session.

If an output file is specified for a session, then the sessions output buffer is flushed to the file periodically at a user specified interval. The output is also automatically flushed to the output file if the session's buffer fills.

Any number of sessions' output can be directed to a single file; the s_fileout command takes the "all sessions" wildcarding. A time and session-name stamp precedes each flush so one can make sense of the mixed session output.

6 Remote Session Software Library - Application Cleanup and Synchronization

We have strived to keep the remote software to a minimum. **dbs** can be run with no special software on the remote side. However, in order to be able to use the functionality of cleaning up (i.e. killing or removing) applications for a clean start, or synchronizing the start-up of a server and its clients, some remote software is required.

We provide a process "cleanup" registration routine that applications call if they need to be killed via the "s_cleanup" command. Registration and cleanup is done on a per session basis, but like other **dbs** commands, s_cleanup accepts wildcarding. Registration is done through foolproof mechanisms, such as "fuser" on Unix and a task variable mechanism developed for VxWorks.

Some applications need to reach a certain point before others can be started. An example is a network client-sever application, for which the server needs to begin listening for client connections before the clients can be started. **dbs** provides the "s_waitcommand" command for this purpose. s_waitcommand sends a command to the remote session, and waits with a time-out for a special character sequence to be sent back over the standard out of the remote session. A routine is provided for remote applications to write this sequence to their standard output. In the client-server example, the server would call this routine immediately before listening for client connections, and the clients would be started immediately after the server.

7 Security

Since **dbs** is rlogin based, it inherits rlogin security of account based .rhosts, in which trusted machine/account pairs are specified in this file. Account level security is integrated into **dbs**.

8 Future Plans

We currently plan to add two new features: routing session output to slave Xterm windows in addition to files and supporting full regular expression wildcarding for sessions. The former will add more visual monitoring of session output, and the latter the command multicasting capability, e.g. sending a command to all nodes of a given operating system type (by embedding the OS type in the session-name), as well as directing output from sessions matching the wildcarding to a single file.

We wish to integrate **dbs** with our DA configuration management "database", so that the systems application start-up is specified along with other DA parameters through the database. Since **dbs** can take command input through a pipe, we imagine this will be accomplished by piping into **dbs** the output of an application which extracts the start-up information from the database.

We also plan to upgrade the command parsing of the client TCL commands to an improved parser we have developed.

9 Conclusions and Observations

One experience we would like to pass on is the fact that we believe we gained in productivity and maintainability by writing the mutliplexing server in C++, and using a commercial class library, tools.h++, from Rogue Wave Inc. We were able to use their hash dictionary class for session-name mapping, and its regular expression capabilities will make it easy to extend the session-name wildcarding capabilities.

We have built a networking tool, **dbs**, to aid in the bootstrapping of systems built from distributed, cooperating applications. While we built this tool specifically for the needs of

HEP data acquisition, we feel that we were careful enough to make the tool general enough for use in other distributed applications.

References

[1] "Fermilab's DART DA System, Pordes et al, these proceedings.

[2] "DART - Data Acquisition for the Next Generation of Fermilab Fixed Target Experiments", Gene Oleynik et al, IEEE Transactions on Nuclear Science, Vol 41, No 1.

[3] ,"DART Run Control Requirements", Appleton et al, Fermilab Computing Division (CD) document, PN471 Fermilab CD library.

[4] "DART Bootstrap Services (**dbs**)", Appleton et al. Fermilab CD document, PN483.

[5] "Tcl: An Embeddable Command Language", Ousterhout, 1990 Winter USENIX Conference Proceedings.

Session 6

User Interfaces and Visualization

OnX

Guy Barrand
Laboratoire de l'Accelerateur Lineaire
Centre d'Orsay
Batiment 200
91405 Orsay
France

Abstract

A tool to build interactive graphical applications. Recent evolutions.

INTRODUCTION

OnX is a package to build portable interactive graphical applications. Its architecture, based on general programming standards (ANSI C, Xwindow, MOTIF) makes it a universal framework for portable applications.

OnX is already integrated into several interactive environments of HEP experiments (H1, DELPHI, NOMAD) and is planned for others (VIRGO).

Therefore, a significant evolution of the internal architecture and the widget library, has been achieved since its introduction at CHEP 92.

After a brief description of OnX's basic features, we present latest evolutions, and developments on visualization of GEANT objects.

BASIC FEATURES

They are:

- the integrated interface manager that allows to build and manipulate the components of the GUI (widgets) while the application is running.
- the use of interpreters to execute widget callbacks (ie. interactive fonctinalities).
- a library of widgets.

THE WIDGET MANAGER

The widget manager (or interface builder) is now exploiting directly the Xt hierarchy. Note that a user can declare to the OnX widget manager his own Xt widget set.

THE OnX WIDGET SET

An important effort has been done to implement OnX widgets according to the Xt specifications. These widgets are put in a separate library. They are:

- XoDraw: an XmForm drawing area.
- XoCamera: to do 3D graphic.
- XoPlot: to visualize plot, functions.
- XoCmap: a color chooser.
- XoGraph: a container visualizing children links.
- XzHplot: an HPLOT widget.

- public domain XoRheo and Eyes widgets.

With the Oz tool it is possible to build an interface able to handle ZEBRA files. This feature is examplified by the "po" resource file to do some interactive analysis.

THE INTERPRETERS

The C interpreter Ci is now quite close to a full ANSI C syntax. From a UNIX machine, it is now possible to write interpreted callbacks that send a command to an other host (accross the network).

OTHER FEATURES

Non rectangular widget window are available.

A user can define a "balloon" help system similar to the Macintosh one.

It is possible to give buttons colored icons at the "xpm" format.

OnX doc is now available with WWW at
http://lalinfo.in2p3.fr/OnX/OnXDoc.html.

Tar files could be retreived with ftp anonymous at: lalftp.in2p3.fr.

DEVELOPEMENTS

For XoCamera, we are implementing a z-buffer package to solve the hidden surface removal problem.

We want to address the non trivial problem of handling a scene with faceted volumes (detectors) intermixed with wire frame modeled objects (tracks, hits) by the means of z-buffering.

This will allow to provide depth-cueing, shading, texture mapping.

An effort has been made to be able to do some graphic with the GEANT package. Visualization of GEANT tracks, hits, volumes in a wire frame or a faceted frame mode, is now possible. Only the GEANT library is used: no link to graflib, grafX11 is necessary.

CONCLUSIONS

The general architecture of the OnX package is now quite stable and robust. Experience showed us that OnX is mature to cover most of the needs for HEP on-line, off-line interactive applications based on Xwindow. To our knowledge OnX, in its category, is presently the only package that offers such a level of functionalities at zero cost.

LookOnX : A New Technological Approach for the H1 Off-line Event Display

Antoine Perus,
Christian Arnault
Laboratoire de l'Accelerateur Lineaire
CNRS - IN2P3
91405 Orsay Cedex, FRANCE

Abstract

We present the re-engineering work achieved on a traditional-fashioned GKS graphical package with new "event driven" concepts required by using workstation interactivity.

INTRODUCTION

The Off-line Event Display for the H1 experiment H1look [2] was originally written using a basic kernel package - Look [1]- that provides a uniform control engine for graphical modules based on conventional GKS graphics.

The use of powerful UNIX workstations is growing daily and their new graphical functionalities and CPU power are well suited to interactive graphical data analysis.

Therefore re-engineering work was needed to put in some new features such as :

- interactive manipulations of graphical objects;
- object oriented definitions of the graphical representations and connectivity between graphical objects and physical objects;
- multi-windowing facility.

THE RE-ENGINEERING

This re-engineering was achieved with two underlying requirements :

- to keep available every existing feature and functionality of the original kernel as well as to reuse in a transparent way existing user modules and applications;
- to preserve the code structure in order to keep the possibility to rebuild either the previous GKS application or this Motif version; in addition this work had to be done in the frame of the code management system used by the H1 collaboration, CMZ.

The implementation of Motif version of the Look system was made in the following way :

- re-design of the global architecture around the event driven concept imposed by the required interactivity;
- emulation of the needed GKS entry points and mapping onto X-Window primitives;
- re-design of the user interface layer through the use of the interface builder OnX, including steering of the application and graphical manipulations.

Event Driven Architecture

The idea was to organize interactivity of the application around the architecture originating from interactivity concepts of Motif and X-Window, or the so-called "event driven" architecture.
The main input loop of the Look kernel was split into several basic pieces :

- to extract the different functionalities from it;
- to define the main action routine, called either from the Motif main input loop or directly from Look in GKS environment;
- to introduce the call to Look loop in Motif environment, to pass text arguments between Motif world and Look one.

GKS Emulation

Graphical functions were handled by a software emulation of the GKS entry points, using the OnX graphic functions based on X-Window primitives.
This allowed to keep strictly unchanged the complete set of already existing user defined software modules, and to enable the development of new ones with known GKS entry points.

CONNECTION TO MOTIF

The connection to Motif was achieved through the use of the external package OnX [3] developed at LAL Orsay .

The important point in the choice of this package was its strictly independence of any other package or library, apart Motif itself. Developed by LAL people for HEP needs, it is freely available. Written in pure Ansi C, it is highly portable and currently runs on a large number of different platforms : any UNIX machine (DEC/OSF1, DEC/Ultrix, Sun, HP/Risc, Next, Mac/AUX, IBM/AIX), DEC/VMS. It only needs X11R4 or X11R5 and Motif 1.1 or Motif 1.2 for correct behavior.

Two basic features of OnX were heavily used :

- the support of the Motif widget set for handling user interface with dynamic and interactive control of its behavior through the use of a C interpreter;
- the support of structured 3D graphics within dedicated Motif widgets, the "cameras", with a comprehensive set of tools to manipulate graphical objects.

Dynamic Control

The central mechanism used in OnX for connecting the user interface to the internal behavior of the application (and its functions) is based on an online C interpreter, available at any time for executing activation functions, "callbacks" in Motif terminology, on widgets.

Each Motif widget owns a set of resources, such as background color, label strings, text values, border width, etc... OnX gives access to any possible resource defined by Motif on widgets, either interactively (through the embedded interface builder) or by program (by providing convenient shortcuts to the Motif programming interface).

Among these resources, there are different callbacks available to a given widget. Each class of widget (PushButtons, ScrollBars, Text, etc...) owns its specific set of callbacks. For instance, a PushButton (frequently used as buttons or in menus) has an "activateCallback" that is called whenever one depresses it.

The callback resource takes the form of a text, understood by OnX as a C source code, and is therefore interpreted when needed. The C interpreter knows most of the OnX internal entry points, some of the standard C run-time library entry points and may be extended by the declaration, from the application program, of any number of private entry points, with the argument syntax.

The resulting is a normal C code, that connects to the internal functionalities of the application. The syntax of these scripts corresponds strictly to legal C syntax, so that they may be embedded later on in the application, and compiled.

As an example, in our Look implementation, the Look internal command interpreter is declared to the OnX interpreter under the name "LookInput", allowing the installation of any Look command or set of commands into a widget callback.

Example : Assume you want to have a "Next Event" button, you may install as its activateCallback the following C script :

```
{
  LookInput ("read");
  LookInput ("sc ctkr; nohead");
}
```

This code will execute, in sequence, first the reading of an event from the event file (the read command) and then the side view display of it with tracks in central tracker (sc ctkr) and without header text (nohead).

OnX stores the complete definition of the user interface into a textual "Resource file", the syntax of which is described in the OnX documentation, allowing to customize at the user level, different aspects or behaviors of the Look based applications.

3D Graphic

A direct connection between graphical constructs generated by the physics modules and OnX graphical objects is now possible. These 3D objects may then be manipulated interactively from the user interface, with the help of the mouse (rotations, displacements, zooming, selections, information requests, etc ...).

A naming convention on graphical objects may be defined (through the use of

4 new entry points), so that, for instance, interactive selection may be linked directly to internal Look based bank selection.

A front end manipulation scheme is also provided for multi-windowing use : OnX supports 3D graphics in a specialized graphics widget, named a "Camera". Such cameras may be resized, iconized at will, refresh operations are handled locally at the interface level, without any call to Look.

The notion of "target" camera provided by OnX is used to act at any place, with the mouse or in callback scripts, as the destination window where all possible graphical actions will be sent.

CONCLUSIONS

This work has showed the ability to plug in an old-fashioned but well-conceived software to a new one, which features interactive object oriented tools without upsetting all the existing code : in addition to the splitting of the Look main loop in seven subroutines, only about twenty existing subroutines were lightly touched.

Besides the independence with GKS, this re-engineering provides many new functionalities like permanently multi-viewing, editable panels of commands, connectivity between graphical objects and physical objects, or in short, more interactivity.

ACKNOWLEDGEMENTS

We thank Ursula Berthon of LPNHE, Volker Blobel of Hamburg University, and Serguei Levonian of Lebedev Physical Institute for their help in understanding Look internal structure; and Guy Barrand of LAL for his constant support and help with OnX and Motif.

REFERENCES

1. V. Blobel, "LOOK - a system for data analysis", The Official Manual for the LOOK Kernel.
2. U. Berthon, S. Levonian, Th. Naumann, "H1LOOK - General Purpose H1 Event Display", The Official Manual.
3. Guy Barrand, "OnX, A Tool to Build Interactive Graphical Applications", The OnX Reference Manual. LAL - Orsay. 1994

A Tool for Geometrical and Graphical Objects

Minato KAWAGUTI and Satoshi TANAKA
Fukui University
9-1, Bunkyo-3, Fukui, 910 Japan
Internet: kawaguti@i1mps1.fuis.fukui-u.ac.jp

Abstract

A multi-purpose software tool (1) for generating, modifying and/or viewing 3D geometrical objects, (2) for displaying experimental data, and (3) for generating publication-quality figures, was developed. Required executing environment is minimal: Unix and X-window.

Motivation for the Development

A simple graphics module was developed for use at various phases of research activities, such as (1) conceptual design of detector hardwares, (2) generation of figures for publication, (3) two-dimensional (2D) and three-dimensional (3D) display in physics data analysis, (4) direct measurement of geometrical parameters in the 3D view screen, and (5) correspondence across the network between collaborators concerning 3D objects.

Its main objective is two-fold: (1) to make all the desk-top machines eligible for handling graphical data, and at the same time (2) to devise a simple scheme for easy manipulation of conceptually straightforward geometrical objects without getting involved too much in the complexity of the general-purpose protocols used in most graphics software packages.

Implementation

The program consists of mutually independent two parts. One is dedicated to figure drawing, from quick display at the X window screen of raw data to high quality figures for publication. The other deals with 3D objects. These two functionally separate parts will henceforth be referred to as *2D module* and *3D module*, respectively.

The program is written using GNU g++, ANSI C and GNU bison. It assumes only the ordinary (that is, non-PEX) X-window environment for maximal portability. As a self-contained graphics package, it does not rely on any existing (but not necessarily omnipresent) graphics module, such as PHIGS packages. Furthermore, it does not request any hardware or software of specific make, such as graphic workstations, or optional graphic engines.

The program relies solely on *Xlib* for X-window related services, except for the menu selection component which uses X/Motif. Tcl/Tk based menu, to be added in the near future, will make the software entirely free from any commercial product including X/Motif.

Drawing Figures

2D module, intended for drawing figures, serves for dual purposes: (1) convenient physics data display in a form of figures, and (2) generation of publishing quality figures, ready to be inserted in documents, say, typeset by TEX.

Figures can be edited at the X window screen in both types of applications. The primitives prepared for figure drawing include, among others, abscissa and ordinate axes of various scales and ticks (or mesh), line styles and marking symbols of function curves, raw data points, 1D and 2D error bars, figure title, and annotation texts.

Editing operations are performed to a significant extent through menu selection, for which standard menu policy of X/Motif was adopted unaltered. Currently 99 editing functions are implemented in the menu. Extension of the menu items is quite easy.

As a unique feature of *2D module*, it accepts any of TEX-typeset entities, such as mathematical formulae and/or tables, as a graphic object, and embeds it into the figure.

The final 3D drawings the *3D module* generates as its output is also considered as the graphic objects by *2D module*, ready to be included into the figures.

Achievement of high quality output figures, acceptable by demanding commercial publishing houses, with minimum labor for the authors is the primary concern of this module.

3D Geometry

To create a 3D object, *3D module* uses a modeler, which compiles the information defined by the user in a data file (*object description file*). Complex 3D objects can generally be represented in two complementary ways: (1) boundary representation (B-rep), and (2) constructive solid geometry (CSG) representation.

The modeler interprets the model data in the *object description file* and describes the solid objects based on the B-rep scheme. The modeler composes the objects with a set of polygonal surfaces. The hidden line elimination algorithm determines invisible portion of each object through geometrical calculation, first by dissolving these polygons into smaller triangles, and then by checking their mutual relationship in space.[1]

Solid Boolean operations (for 3D closures), used in CSG scheme, are not implemented in the current modeler.

Dimensional Accuracy

Geometrical parameters can be measured interactively at the X window screen. Physical quantity with regards to the object shape or mutual orientation, as far as it is calculable from the definition of the objects, may be extracted.

The hidden line (surface) elimination through geometrical calculation assures the mathematical rigor, and hence the resolution independence of the final processed output data from the display devices.

Object Description

The modeler recognizes the following primitives: (1) polyline, (2) circular or elliptical arc, (3) polygon, (4) column with arbitrary cross section, (5) object with the axial symmetry, (6) surface generated by sweeping an arbitrary generator curve partially around an axis, (7) quadrilateral mesh, and (8) triangle strip.

The object description language the modeler interprets has C-like language constructs, such as *for* loop, and *if-then-else* control constructs. Math expressions

with predefined 41 math functions can be used. Identifiers are declared as object names, variables, or constants.

Emulated Camera

The *3D module* emulates the process of shooting 3D objects with a distortion-free camera. The default setting assigns the camera with a standard 35 mm film. The idea behind it is to let user manipulate the solid objects with reasonable realism. Experiences in photography are expected to help grasp the degree of perspective foreshortening effect caused by a selected focal length of the imaging lens.

Multiple Views

Geometrical objects can be edited at the X-window terminal screen, while viewing arbitrary number of views simultaneously. Independently controllable cameras are allocated to each view. Combination of four views — plan, front, side, and bird's eye views, respectively, might be a typical example.

Depending on the purpose of 3D visualization, any of three drawing schemes can be selected to each object: (1) wire frame, (2) back- or front-facing facet culling, or (3) hidden-line elimination.

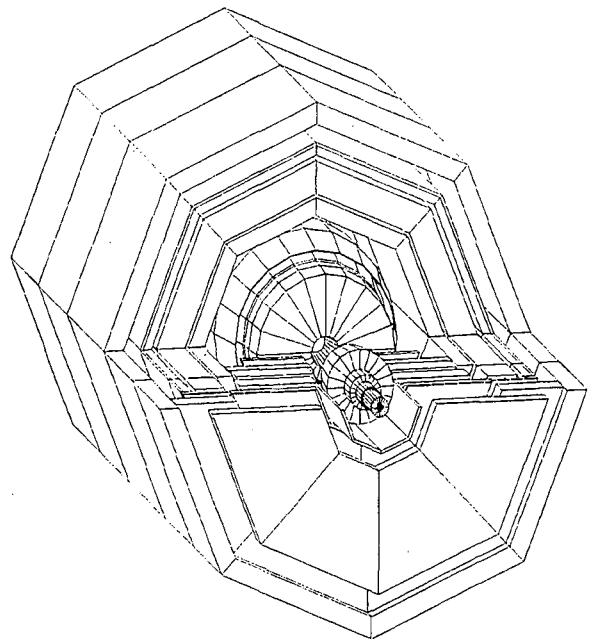
Pick and Drag

Solid objects can be picked by the mouse at the X window screen, and moved to a new location by dragging the mouse. To resolve the ambiguity stemming from the extra degree of freedom, a constraint condition should be imposed by the user prior to any dragging operation. Each view accepts entirely different constraint condition simultaneously.

To add 3D sensation at the screen, auxiliary "eye-guiding line" may be superposed to each of objects on display and to the *3D cursor*. It visualizes on the screen each of the three components of the coordinates system, be it Cartesian, cylindrical or polar.

Sample Drawing

Shown below is a part of the SDC detector[2] for the SSC as a typical example drawn by the *3D module*.



Acknowledgments

The authors thank our students, Hiroshi Hasui, Hirokazu Takashima, Takayuki Kato, Takeshi Funakoshi, Sumiyo Tanaka, and Jun-ichi Shiono, for their contributions and assistance.

References

[1] Ammeraal, Leendert, *Programming Principles in Computer Graphics*, John Wiley & Sons, Chichester, 1986.

[2] Solenoidal Detector Collaboration Technical Design Report (SSCL-SR-1215), 1992.

SCANNING HUGE NUMBERS OF EVENTS *

Giuseppe Zito
Sezione INFN Bari(Italy)

Abstract

I report about the development of new tools enabling the user to do visual scanning of huge numbers of events. By huge I mean more than 10,000. These tools can be used to do visual analysis of any kind of experimental data represented in graphical form. To test these tools I use a data base of around 50,000 dynamical systems.

INTRODUCTION

High energy physics has started by taking photographs of particle collisions and examining them visually. Now almost all data is read with electronic detectors and analyzed with statistical analysis programs. These programs are able to sift interesting events out of millions of them, so why develop these new tools? We have event display programs, that allow us to look at the single event from any point of view. Event display programs are very different from the statistical analysis programs,since they can be used by anyone.Anyone can "discover" by using them new phenomena like V0, muons,etc. Statistical analysis programs have a model behind and can be used only by experts. This means that you can find something only if it is in your model. It is like searching something in the dark:you find it only if you know what you are looking for. Unexpected phenomena are difficult to discover with this method. My goal is to allow the user to see all the events, without a model behind, like you now see a single event in the hope that this will make easier the discovery of new phenomena.

SCIENTIST AS MOVIE DIRECTOR

How can you see thousands of events like you see a single one [1, 2]? A guiding metaphor will help us to approach the solution of this problem: the metaphor of the experimental scientist as a movie director that collects sequences of frames without a script having at the end the formidable task of ordering in a coherent way the collected material. The tools developed should make his task easier. Following the metaphor I use only visual methods to process and order the experimental data. The ordering of the material in a coherent way means in the case of

*Complete document with color pictures available online with URL http : //alephinfo.cern.ch/al1$ user0/zito/html/chep94sl/chep94sl.html

the movie and also for the experimental data, that a huge amount of data can be analyzed in short time. In fact a simple computation shows that the information contained in a two hours movie is comparable to the information collected by a large hep experiment in one year. And as anyone knows a movie can be "analyzed" in two hours.

EXPERIMENTAL OBJECTS

Since I want to develop tools that can be used for any kind of data, I'll speak of "objects" instead of events. These objects come from experiments or simulations,are very complex and can be represented graphically in many ways. My approach is to represent them graphically in some way,order them using this graphical representation and then allow the user to scan them quickly by using animation. But which representation do we use? The chosen representation for each object should be a "global" representation:this is a 2D or 3D color symbolic representation of the object that uniquely identifies it in a unique way, like a portrait in a Identity Card identifies the person holding it. Like the portrait this global representation should also be meaningful to the user of the tools and scale gracefully with size.

THE VISUAL INDEX

We compute this global representation for each object and then to allow a fast visual scan of all objects, we use two methods: The first method consists in the ordering of objects (represented by these global representations) based on the pattern of pixels and on the color. The list of ordered objects is written on disc together with a file containing black and white icons, one for each object. These icons form a visual index that can be quickly scanned (around 1000 icons per second) to know what we have in the objects data base. Clicking on the icon will get the full object displayed, with its different representations.

FAST SCAN WITH ANIMATION

The second method exploits animation to quickly scan objects. Also here the ordering of objects on the basis of their visual features is essential. Implementation of animation depends heavily on the available hardware and software. These tools were developed by using the X11/Motif interface,on a Digital Alpha Openvms workstation. For this platform I have used until now this simple scheme. During a initial phase, the global representation of each object is computed once for all and stored in compressed format (JPEG) on disc. When later the user selects a range of (ordered) objects for animation, these graphical representations are loaded in memory and an animation is started by displaying them one after the other. Limiting the size of the global representation to 100x100x256 allows for a scanning at the rate of 50 objects per second, giving a smooth animation. With this size also the overhead in terms of disc space is not so big (30000 objects occupy around 100 Mbytes of disc space).

CONCLUSION

By using the global representation,I transform the data in a kind of alien landscape explored having a visual index as a map and animation as a way to move quickly through it. This is just a starting

point, equivalent to building the first road in a unexplored territory. From this first road now you can start the real exploration by selecting single interesting objects or classifying some of them in subsets of similar objects. You may order them and play animations by using representations different from the "global" one. Visual techniques more specific to the test data base are:

- Parameter animation
- Building of visual maps of sets of images depending from two parameters a la Mandelbrot.
- Breeding of formulas with a genetic algorithm

But now you don't need to be an expert to explore the data base putting in this way your visual system to work. Our visual system has "discovered" the Newton laws many years before Newton: will this new way to explore data allow us to discover new laws of nature more quickly? Let us see.

REFERENCES

1. E.R. Tufte , The Visual Display of Quantitative Information , Graphics Press, Cheshire, Connecticut(1982)
2. E.R. Tufte , Envisioning information, Graphics Press, Cheshire, Connecticut(1990)

HEP Visualization and Video Technology

Paul Lebrun and Dave Swoboda
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Ill 60510 USA

Abstract

The use of scientific visualization for HEP analysis is briefly reviewed. These aplications are highly interactive and very dynamical in nature. At Fermilab, E687, in collaboration with Visual Media Services, has produced a 1/2 hour video tape demonstrating the capability of SGI-EXPLORER applied to a Dalitz Analysis of Charm decay. This short contribution describes our experience with visualization and video technologies.

Character based terminals have virtually disappeared and have been supplanted by X-terminals or small workstations. Highly interactive systems capable of 3D color displays have come into use. Pictures that could be "picked" (selected) and directly manipulated on the screen were found useful. These visualization engines became available at least a decade ago, and are finally available on many physicist's desks.

Commercial systems for 3D visualization such as AVS were available years ago, and have been proposed on numerous occasions, with limited success, due to the relatively high startup cost (hardware and software). These costs are coming down (not necessarily as fast as HEP budgets, unfortunately), and one can anticipate that conventional graphics systems (such as DI3000, Phigs or GKS) will become "legacy" systems very soon. Currently, at Fermilab, the Accelerator Division has a few AVS licenses while the Computing Division strongly supports the SGI-Explorer system.

A few physicists are no longer considering conventional scientific visualization as "something to try", but are using them daily to understand the underlying physics they are interested in. In particular, they appreciate the capability of adding new elements to the picture quickly and easily, including a built-in Graphical User Interface. They are no longer limited to a specific set of rigid views carefully crafted through 3D to 2D transformations; the Render module takes care of this without a single line of user code. Once the 3D geometry is declared, the user is done!

SGI-Explorer has been used for magnetic field analysis in a Fixed Target experiment, to graphically represent tracks and vertices, to study Charm decay at E687 and B physics at CDF, and to understand complicated Dalitz plots and fit-

ting procedures. D0 is currently studying detailed "Lego" plots (electromagnetic vs hardonic response of calorimeter cells).

However, it should be noted that not everyone shares this optimism. The Explorer Render module is based on OpenGL, which is not available on most X-terminals and many workstations. Visualization is also CPU intensive, and older workstations (particularly VAXes) are simply out of the picture. Hardware availability is not the only root cause of the limited usage of conventional scientific visualization systems - a lot of good UNIX workstations are currently used as X-terminals. There may be a perception problem with these "new systems" : potential users simply don't like the idea of throwing away good, old, working graphics code that took years to build, and they don't really believe they will *not have* to rewrite lots of graphics code. They also think graphics glitz is distracting (correct!), but wrongly deduce that sober scientific visualization is a luxury HEP cannot afford.

In addition to commercial, "standard", visualization packages, our community has a definite need for complementary HEP-specific tools such as good statistical analysis packages: PAW, HistoScope and other similar Histogram or N-tuple browsers. Obviously, they are highly interactive.

Multi-dimensional visualization is by essence dynamical. Whether we are looking at a spinning event display or studying the effect of a particular selection on an Ntuple, the information content becomes explicit when things move on the screen and graphics is no longer static. Despite perspective rendering, hidden line removal or fancy lighting, a 3D event display can often be misinterpreted if seen only under a few rigid views. Likewise, using PAW or HistoScope, N-tuple selections followed by projections onto histograms are most readily appreciated when they are done quickly, so that relevant changes can be easily detected. The message is not carried by a rigid set of pictures, it is instead shown by actions such as the motion of the camera in the Render module, or the evolution of a curve as a slider moves. In essence, we are looking at movie clips. We have to make these movies easily and cheaply, and video technology provides a good way of doing so.

Obviously, other sources of images can also be made available on the workstation. Tele-video conferencing comes to mind, but will not be discussed in this note.

Video is obviously not the preferred information media in HEP. But, it might be worth trying. So, last summer, in collaboration with Visual Media Services (VMS) at Fermilab, we produced a video tape describing our experience using SGI-Explorer V1.0 while analysing the Dalitz plot of a typical Charm decay : $D^+ \rightarrow K^-\pi^+\pi^+$ observed in the E687 Charm photoproduction experiment at Fermilab.

The video tape was produced using the following setup. The computer images were generated on a IRIS Indigo workstation, R3000 based, equiped with a video board. This board accepts standard Composite Video as well as S-video inputs. Thus, pictures of the real detector can be imported from a VHS tape to the workstation, manipulated, compared to the simulation images and included into Postcript documents. In addition to Composite and S-video signals, the board

generated RGB output for high quality display devices. Computer images were recorded on 3/4 " tape using a Sony U-matic recording deck. VMS was responsible for taking narrative shots and for the final editing. VHS copies of this video can be made available to those who are interested in this subject.

There was little or no high-tech computing involved in this SGI-Explorer or video project. A lot more time was spent trying various fitting strategies, background subtraction methods and theoretical models rather than getting Explorer to work the way we wanted. Undoubtedly, the most difficult part was - and still is: simply doing the physics. (To this date, we still do not have a completly satisfactory phenomenological descrition of the $D^+ \rightarrow K^-\pi^+\pi^+$ Dalitz plot).

Despite the ease of use of visualization & video equipment on the workstation, it is worthwhile mentioning some improvements made since this video was produced, with respect to HEP applications :

- A complete list of Explorer V2.0 enhancements is available from SGI. Let us mention that the "User Pick" is now fully functional, an absolutely essential condition for any interactive event display. The new module firing algorithm allows us to build appropriate maps for animation sequences and a new set of interface modules to various video devices makes the whole system literally a "turn key" operation.

- On the Indy systems, video I/O capability is integrated into the workstation. There is no need for additional boards.

But one can easily ask for more integration: the video recording media - sequential magnetic tape - are not the one we use for storing data. Even 8mm Exabyte technology is not strictly compatible with Hi8 format. Digital video is around, but the digitization process is not integrated into the computing enviroment on the workstation. Thus, a special recorder needs to be purchased. A VHS device is obviously very affordable, but the resolution and overall picture quality is not great. If substantial editing needs to be done, software must be purchased.

Further integration will come from recent advances in Networking technology. Based on ATM devices (or equivalent), one could conceive of sending video clips to a remote workstation equipped with a good recorder almost as easily as we send Postscript files across the net to a remote printer. Needless to say, saving lots of pictures on disk (particularly Postcript) can be costly. Someday, a better compression algorithm may be implemented. (The disk-drive industry clearly does not like that idea. On the other hand, we should thank the entertainment community for helping make available cheap disk with which to store more serious HEP data...)

The authors gratefully acknowledge the great video media expertise of Fred Ullrich and Jim Shultz from Visual Media Services, Fermilab. Without their help, we probably would not have considered this technology as practical. We also wish to thank other members of the Physics Analysis Tools project, in particular Mark Edel, for their illuminating discussions on this subject and Robert Miller for helping review this document.

A SECOND LIFE OF THE CERN POISSON PROGRAM PACKAGE

V. I. Klyukhin and B. I. Klochkov
Institute for High Energy Physics
Protvino, Moscow region, RU-142284, Russia
Telex: 412 657 IPHE SU, Telefax: 007-(095)-230 23 37
E-mail: KLYUKHIN@mx.ihep.su, KLOCHKOV@mx.ihep.su

Abstract

The modifications of the CERN POISSON program package are presented. The applications of all set of programs to the designing of the collider setup magnetic systems are given as illustrations of new possibilities of the package.

INTRODUCTION

The CERN POISSON package [1] consists of a set of programs designed by R. F. Holsinger and C. Iselin for the solution of Poisson's or Laplace's equation in two dimensional regions. The package is the most popular to solve magnetostatic problems for the magnetic systems with the geometry reducing to two dimensional case. There are four constituents of the package: the mesh generator LATTCR, the equation solver POISCR, the plot program TRIPCR, and the force computing program FORCCR. To discretize the problem a topologically regular triangle mesh is used over all the programs and each node of this mesh is identified by two logical coordinates in the arrays of data and two physical coordinates in the xy- or rz-plane in which the geometry of magnetic system is described.

THE PACKAGE MODIFICATIONS

Our first experience in the use of the POISSON package has come from calculations of the superconducting solenoid magnetic field for a colliding beam setup planed to construct at UNK [2]. During this period we modified the program TRIPCR to plot not only magnetic flux

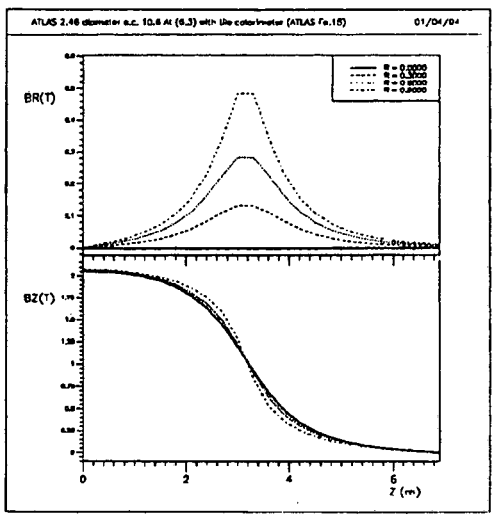


Figure 1. Solenoid magnetic flux density.

equipotential lines, as was provided by program, but also the dependence of flux density from one of the physical coordinates for different constant values of another one [3] as shown in Figs. 1 and 2.

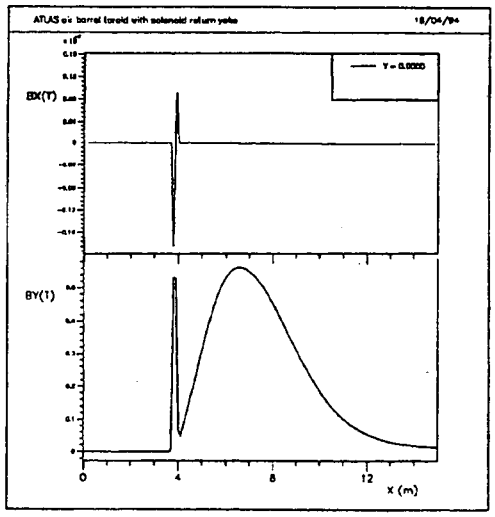


Figure 2. Toroid magnetic flux density.

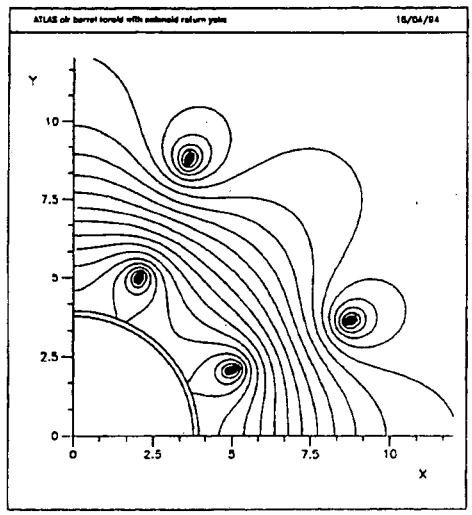


Figure 4. Toroid magnetic system.

New impulse for our interest to the package modification we got doing the calculations for the ATLAS magnetic system [4, 5]. This job causes a creation of two interface programs: POISGT [6],

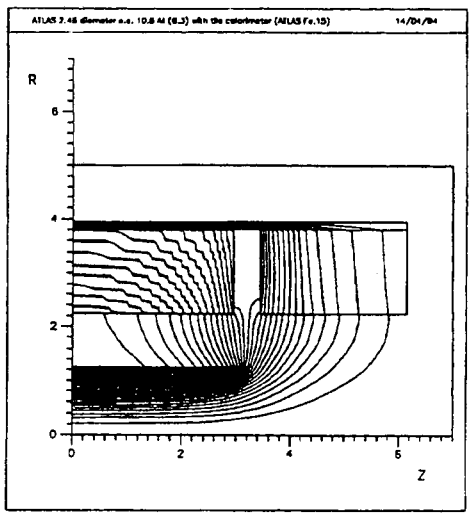


Figure 3. Solenoid magnetic system.

which extracts a magnetic field map into ZEBRA structure and permits to use it for the GEANT simulations, and LATTCP [7], which is an auto-mesh generator using the physical coordinates only. The last program enables to use such the graphic primitives as 'LINE', 'CIRC' and 'RECT' to prepare the input data for LATTCR in the most simple way and to describe such the magnetic system configurations as shown in Figs. 3 and 4. The program POISGT gives a possibility to calculate the magnetic field integrals [8] which determine the particle momentum resolution in the inner tracker placed into nonuniform solenoidal field.

The most essential modifications of the POISSON package are connected with the calculations of solenoidal magnetic field inside the ATLAS hadronic tile calorimeter [5], which consists of the interchanged 5 mm layers of iron and 4 mm scintillator tiles. Fig. 5 presents a detail view of such a calorimeter structure, and in Fig. 3 a general behaviour of the field lines is shown (the graphic images of the scintillator tiles are removed).

To resolve the problem we had to increase the number of the mesh nodes supported by the package from 40 000 to 245 000, and also to increase the number of regions characterized by the different

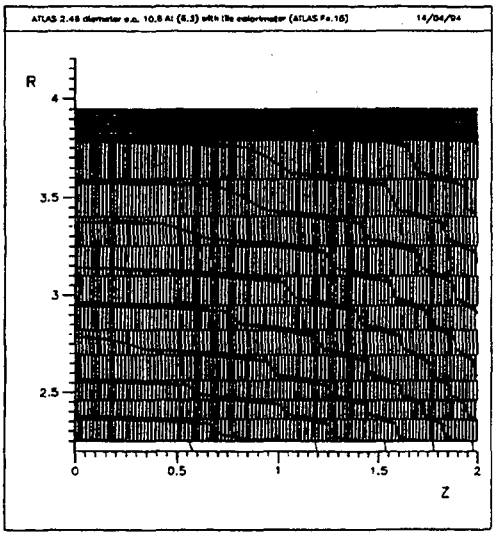


Figure 5. Tile calorimeter structure.

magnetic properties from 60 to 3 500. A calculation of the magnetic system shown in Figs. 4 and 5 takes about 3 hours of CPU time at the DEC ALPHA machine.

CONCLUSIONS

The modified POISSON package and new designed interface programs enable to prepare the geometry descriptions of the magnetic systems in a simple way, to increase essentially the number of elements in the magnetic system, to visualize graphically the magnetic field values for different cross-sections of the system, to extract magnetic field map and to use it in the GEANT simulations, to calculate magnetic field integrals for the tracking volume placed into ununiform solenoidal field.

All these new possibilities make the package well suitable to resolve a lot of magnetostatic problems for various types of the collider setup magnetic systems.

ACKNOWLEDGEMENTS

The authors thank the CHEP '94 Organizing Committee for an opportunity to attend this conference.

REFERENCES

1. R.F. Holsinger and C. Iselin, "The CERN-POISSON Program Package (POISCR) User Guide", Geneva, August, 1984.
2. I.A. Vishnyakov et al. Zh. Tech. Fiz. 62 (1992) 146-156; Sov. Phys. Tech. Phys. 37 (1992) 195-201.
3. B.I. Klochkov, E.A. Kozlovsky, IHEP Preprint 92-54, Protvino, 1992.
4. V.I. Klioukhine, EAGLE Internal Note TECH-No-002, May, 1992;
5. V.I. Klyukhin, B.I. Klochkov, ATLAS Internal Note TILE-No-007, January, 1994.
6. V.I. Klyukhin, CHEP '92 Conference Proceedings, CERN Report 92-07, Geneva, 1992, pp. 844-846.
7. E.M. Boldyrev, B.I. Klochkov, Yu.I. Portugalov, IHEP Preprint 93-84, Protvino, 1993.
8. V.I.Klyukhin, A.Poppleton, J.Schmitz, IHEP Preprint 93-38, Protvino 1993.

THE LSND INTERACTIVE ANALYSIS SHELL

Ion Stancu
Department of Physics
University of California at Riverside
Riverside, CA 92521

Abstract

The analysis shell for the LSND experiment (Liquid Scintillator Neutrino Detector) is presented. The user interface is entirely based on KUIP, while the HIGZ and ZEBRA packages handle the 2- and 3-D event displays and the event data structures, respectively. The complete PAW package has been also incorporated as a submenu, which makes the shell a very powerful tool for on-line data monitoring, as well as for off-line data handling and analysis. Aside from the numerous predefined menus for event displays, event selection, reconstruction, calibration, etc. the users can write additional menus pointing to their own software, which thus enhances the flexibility of the shell.

INTRODUCTION

The LSND experiment at LAMPF has been designed to study a variety of neutrino physics with a high sensitivity, using the neutrinos produced by the LINAC's 800 MeV protons onto the A6 beam stop.

The detector is a tank filled with 200 tons of mineral oil and dilute scintillator (0.03 g/l), viewed by 1220 photomultiplier tubes (PMTs). It is surrounded by an active veto shield - with 292 PMTs - and 2,000 g/cm^2 of steel overburden provide additional shielding against the cosmic-ray background.

The unique active medium proves to be an ideal combination: the mineral oil provides excellent imaging (via the Čerenkov radiation) and the dilute scintillator enhances the vertex resolution and provides calorimetry information.

The detector took data for the first time during the summer and fall of 1993 which yielded the first accurate measurement of the $\nu_\mu C \rightarrow \mu^- X$ inclusive reaction near threshold, as a well as cross-sections for the $\nu_e C \rightarrow e^- X$ inclusive reaction and $\nu\ e$ elastic scattering in agreement with theoretical predictions. Oscillation searches in the $\nu_\mu \rightarrow \nu_e$ and $\overline{\nu}_\mu \rightarrow \overline{\nu}_e$ are still underway, and preliminary analyses show that LSND will be able to measure the $\nu\ p$ elastic scattering cross-section to about 10%.

This paper briefly describes the LSND's operation mode and motivates the choice of the CERN packages in Section 2, while Section 3 gives a tour of the analysis shell.

CERN TO GO

LSND's running philosophy is very simple: all detector "activities" - above some tank threshold (set at 18 PMTs) - are continuously written into circular memory buffers. If something interesting appears to be happening (i.e. the primary threshold of 100 PMTs is asserted), then the event is read out, together with all "activities" in the previous $51.2\mu s$. In some cases, (i.e. when more than 300 tank PMTs fire), all "activities" in the following $1ms$ are recorded as well, in order to catch the 2.2 MeV γ's associated with a possible neutron production and its subsequent capture on hydrogen (mean capture time = 186 μs).

In this operation mode, during approximately 1.5 months of effective runtime, LSND recorded and processed online slightly over 100,000,000 events. Of these, at most 1,000 are relevant for the combined neutrino physics program, which means a total signal-to-noise ratio of $1 : 10^5$, in some cases being as low as $1 : 10^7$.

Given the simplicity of the detector and the high amount of background, the LSND analysis package had to be very powerful, flexible and user-friendly, rather than flashy and sophisticated. And since the detector was already a "fait accompli" well before the work on the shell was started, it had to be a very "detector oriented code": a tool for ultimately doing physics, rather than ultimately a purpose for doing physics.

The CERN libraries provide exactly the necessary tools to put together such a package, so there was no need to reinvent the wheel. In addition, many other arguments speak in the favour of the CERN software packages: (a) they have been around for a while, are regularly updated, and most likely will be around for some time as well ... (b) are widely available and machine independent, (c) are a "comfortable" standard for most physicists and, at last but not least, (d) PAW has become the de facto standard for most high energy physics analyses.

Thus, using KUIP as a command interpreter, HIGZ for the graphics routines, ZEBRA for the data structures and memory management and PAW for the data analysis appeared to be a very natural choice for the LSND Analysis Shell, the LAS.

THE LAS MENUS

When starting up the shell, the LAS-user is presented with the following 5 top-level menus:

- **KUIP**
- **MACRO**
- **PAW**
- **LAS**
- **LAS-USER**

The first two are the familiar menus from any interactive PAW session (or interactive GEANT, etc.); the 3rd one is actually the full-blown PAW package, while the remaining ones are all LSND specific menus and commands. It is the contents of "LAS" that is of main concern in this section, and its sub-menus are structured as follows:

LAS/UTILITIES/: commands to retrieve the used CPU-time, file-status, individual hit data, perform direct data dumps and display the data banks (via the DZDDIV and DZDISP routines of the DZ-package). Fig. 1 illustrates such a display, together with the graphical KUIP menu.

LAS/FILES/: commands to load, unload, reduce (according to the selected set of selection criteria), fill standard histograms (ntuples), recalibrate, copy, merge, etc. FZ-event files.

LAS/EVENT/: commands to load and reconstruct events, menus for the event display and event selection.

LAS/RECO-PAR/: displays, resets and modifies the reconstruction parameters.

LAS/CALIBRATION/: calibration software, surveys of the calibration constants, etc.

Event Displays

Events can be visualised in the LAS in a variety of displays. The standard one used in similar detectors - the unfolded tank - is illustrated in Fig. 2. Other displays look along the reconstructed direction of the particle to better illustrate the Čerenkov ring, or provide a 3-D view along the main cylinder axis, removing one of the endcaps. 2-D and 3-D views of the veto shield are available as well. The hit size is proportional to the charge and the colour is coded according to the corrected time information. Additionally, a tank-display animation displays the raw hits in their order of arrival, with an user-adjustable scanning speed.

THE LAS UTILITIES

Message Handling

All messages are handled through a dedicated package, which redirects them to the appropriate units (e.g. terminal, log-file, ...) and keeps statistical records. Each message has an "error-level" associated with it, as follows: I - informational, D - debug, W - warning, E - error and F - fatal error, which causes the LAS execution to be aborted. Furthermore, any of the above levels can be enabled or disabled by the user, except for the fatal error messages.

File Handling

All files are opened/closed through a dedicated package, which keeps track of all accessed files, access times, their form and status. It also automatically closes all files inadvertently left open upon the LAS exit.

Zebra Access

All ZEBRA access is normally performed through read/write from/to common blocks by apropriate subroutine calls, eliminating user calculation of ZEBRA banks pointers.

USER CONTROL

Users have control over many parameters of the shell. At the beginning of the session, the LAS looks for a control file - which sets a minimal number of parameters - and, if not found, a standard set of parameters is taken. All graphics controls can be modified at any time, such as colour-coding scheme, corrected-time cuts, charge-cuts, etc. All reconstruction parameters can also be modified at any time and the same event(s) re-reconstructed.

More important though, users can access their own custom analysis software through the predefined commands in **LAS-USER**: USER-INI, USER-ANA and USER-END. Finally, the entire predefined user menu can be directly changed by simply modifying the stan-

dard KUIP command definition file and linking to the appropriate custom analysis packages.

CONCLUSIONS

The main CERN software packages used - KUIP, HIGZ, ZEBRA and PAW - have proven to be very suitable in the development and the operation of the LSND Analysis Shell.

The net result is a well-documented (on-line), fool-proof, powerful and user-friendly analysis tool, which now runs on a variety of platforms (SGI, DEC, SUN, ...) and has provided most of the results of the LSND data analysis.

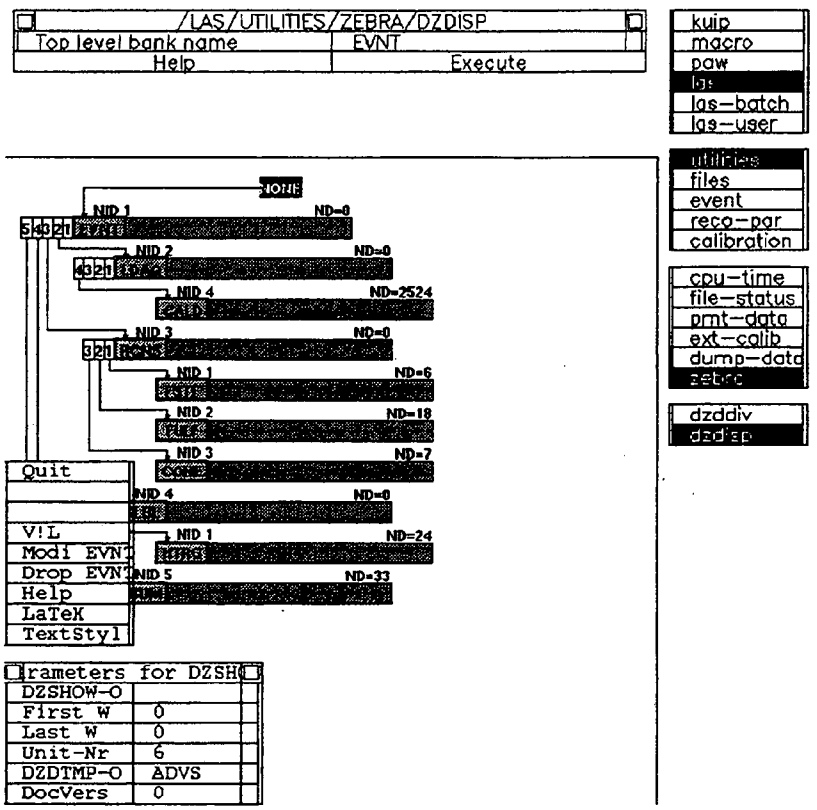


Fig. 1. Graphical display of the LSND data-structure and the LAS menu.

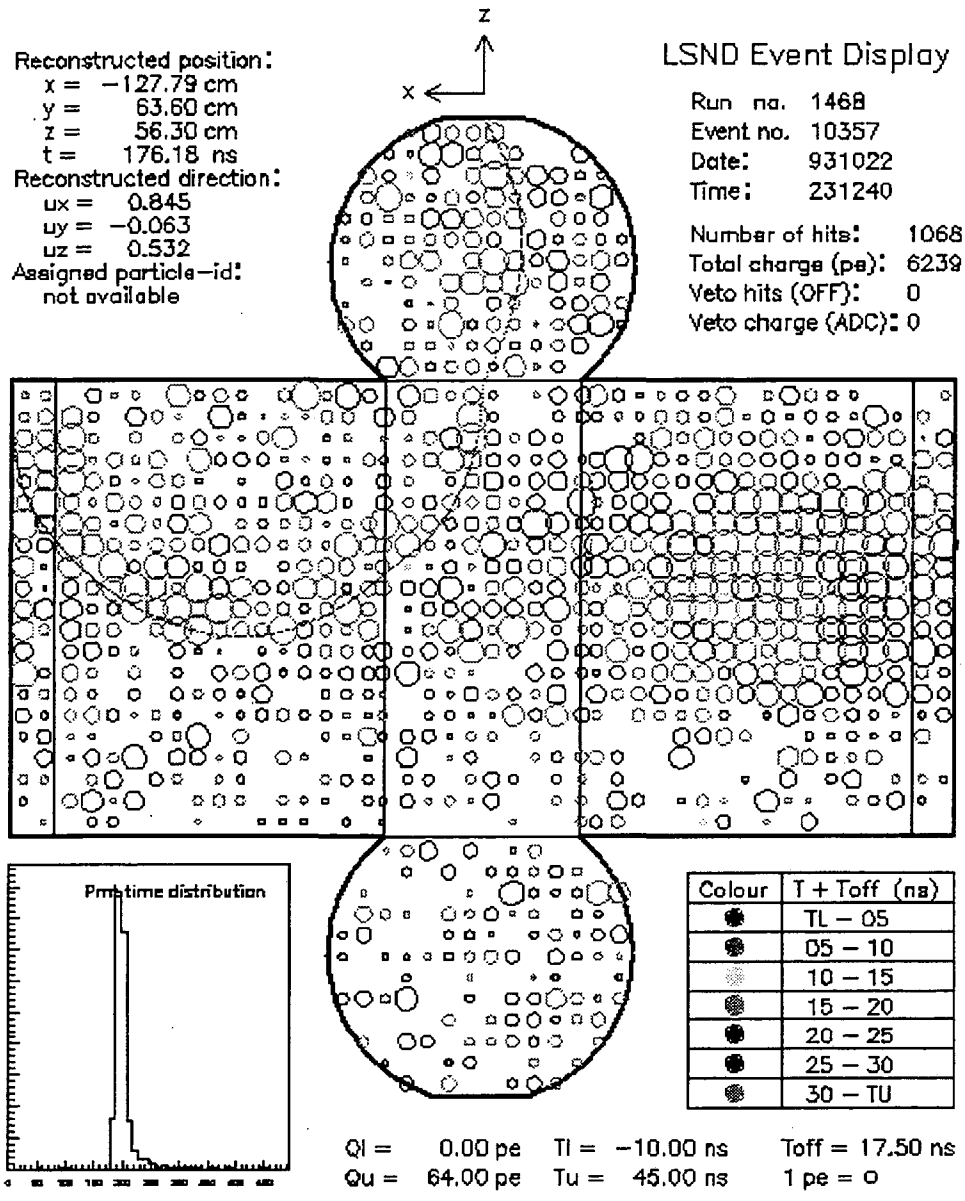


Fig. 2. LSND event display - unfolded tank.

Status of "Nirvana": High Quality GUI Based Software for HEP

Mark Edel, Joy Kryiakopulos, Paul Lebrun, Jeff Kallenbach, Suresh Ravoor
Fermi National Accelerator Laboratory, P.O. Box 500 Batavia, IL 60510, USA
Konstantine Iourcha
Petersburg Nuclear Physics Institute, 188350, Gatchina, Leningrad district, Russia

The Fermilab "Nirvana" project has produced a number of tools for data analysis in High Energy Physics. Our goal has been to produce software which takes maximum advantage of the workstation graphical user interface. Histo-Scope and NPlot enable users to browse data from running programs and HBOOK, Histo-Scope, and columnar text files. They also provide highly interactive two and three dimensional plots which can be rotated, scaled and adjusted directly with the workstation mouse. Our newest program, NFit, a GUI version of the MINUIT fitting program, will be ready in a few months. In addition, NEdit, our programmer's text editor, has recently become very popular outside of the HEP community since it was released as public domain in December. We will discuss the basics of these software products, as well as improvements in the newest versions of Histo-Scope, NPlot, and NEdit.

The fundamental paradigm for how people interact with computers now changed almost completely from text-based to the graphics-based interfaces. For most computer users, this is unequivocally good. For HEP, the benefits are less clear. We obviously benefit from the fast, inexpensive, cpu power and higher display bandwidth of new computing hardware, but the real revolution has been in software, mostly in the area of ease of use which is not a high priority in research. In fact it has been argued that X-Windows has actually made HEP software harder to use. For many physicists, X-windows is synonymous with crashiness, portability problems, interface slowness, and unnecessary complexity.

GUI Software

A basic reason for this frustration is that GUIs don't naturally make programs easier to use, they are mostly a way to trade developer time for user time. A goal of good GUIs is to create an illusion of simplicity. The high quality and quantity of features in Macintosh and Windows applications come mostly from software producers' abilities to distribute development cost over thousands of users.

Why are GUIs so unequivocally good for everyone else? Trading developer time for user time is simply the right thing to do given the number of people now using computers. GUIs exploit visual skills which are inherently parallel and therefore faster, compared with character based interfaces which use serial verbal skills. Direct manipulation, simulating a real-world actions to accomplish a computing task, is a powerful metaphor that gives us, drawing programs, CAD tools, and the Macintosh file interface. Graphical menus and dialogs make programs much easier to use by making their capabilities explicit and allowing users to refer to commands rather than remember them. This has great impact even on expert users, since no one exercises every feature of a software package. Properly constructed graphical menus also provide a natural transition between novice use and expert use by providing and displaying shortcut (accelerator) keys which experts can substitute for pulling down menus and pressing dialog buttons. A well written GUI is also safe to explore and experiment with, without worrying about losing data.

X and Motif

Having mostly standardized on X/Motif workstations, we are committed to a certain set of constraints and opportunities. X-Windows programming is actually a bit easier than Macintosh and Windows programming, but each is difficult for it's own reasons. The biggest problems with X and Motif are generally related to quality, including intrinsic bugs, poor design, poor documentation, lack

of testing, lack of consistency between applications, and speed. However, newer releases and faster machines have now made it possible to deliver very high quality software. From a user's perspective the biggest problems with X and Motif remain the quality, quantity, and cost of commercial applications available.

The "Nirvana" Software

HEP can benefit from GUI technology, but it is most appropriate for applications with larger numbers of users and/or problems which are inherently graphical. We looked for applications that were both widely applicable to as many HEP users as possible, and ones which gain unique advantage from the high-bandwidth graphics and the mouse. The tools we present here provide both new ways to visualize HEP data, and the true ease of use of a high quality GUI. By high quality, we mean:

- Modeless, one level interface with clear choices, careful use of screen real-estate and good graphical design
- Consistently safe, undoable and cancelable operations
- Positive visual feedback for all actions
- Fast response time
- Shortcuts for expert users
- Consistency with Motif/Windows/ Macintosh environment
- Non-crashing
- No special dependencies or knowledge of X-Windows required, easy to install
- Portable
- Interoperable with other HEP and general purpose X-Windows applications

Histo-Scope & NPlot

Histo-Scope is a tool to select and display histograms, n-tuples, and scalar values from a program as data is being created or analyzed. Using Histo-Scope, physicists can interactively "browse" through the large quantities of statistical data that their analysis and data acquisition programs gather as they run. It is intended to complement existing physics applications, providing immediate access to data while a program is running, as well as new interactive methods for viewing data.

Histo-Scope has two parts. The first is a small library of routines which can be inserted in physics analysis or data acquisition code without significantly changing its behavior. The other part is the "scope" process. Invoked upon user demand, the

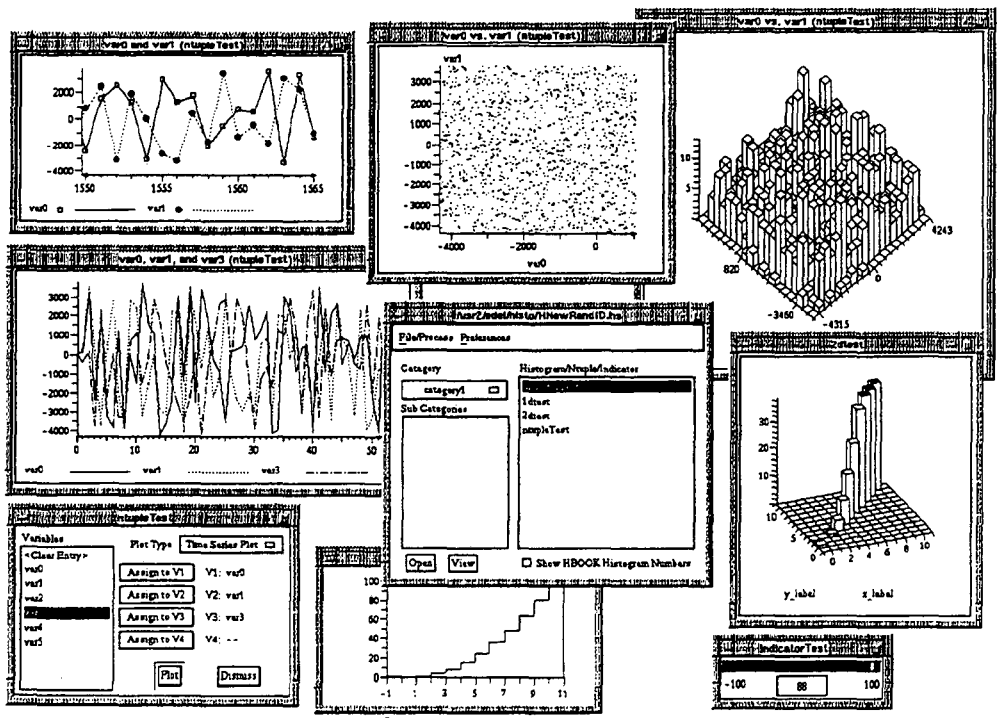


Figure 1. Viewing Data with Histo-Scope

scope requests and displays data continuously from the analysis process. The scope concentrates on interacting effectively with users. It responds to mouse and keyboard input and provides the interactive graphing and plotting that enable users to view their data quickly and effectively. The scope program, on its own, can read HBOOK and Histo-Scope format files.

NPlot is a tool for quickly plotting columnar data from text files. It is a simple re-packaging of the Histo-Scope n-tuple interface with a file reader.

Both Histo-Scope and NPlot produce highly interactive graphs and plots. These include: multi-variable graphs, two and three dimensional scatter plots, and one and two dimensional histograms. Users can re-scale, zoom, and pan these plots by dragging on axis scales and other sensitive areas. The three dimensional widgets, 2D histogram and 3D scatter plot, can be rotated accurately using the mouse as a hand on a "virtual trackball" sphere surrounding the plot. The plots can also be combined with animation sliders to reflect additional variables or to re-bin histograms.

Version 2.0 of Histo-Scope, adds new plot types including 3-D scatter plots, and 1 and 2-D adaptive histograms, as well as error bars, and VMS support. It also incorporates a method for setting variables and activating routines within the connected executable.

We are currently working on a version 3 release, including an enhanced application programming interface for working with n-tuple and histogram data independent of HBOOK.

NEdit

NEdit is a GUI style plain text editor. It was originally a simple test program for our menu and file access routines. With a little extra work, it became the text editor of choice within our group, then spread inside of Fermilab, and now on the Internet, to the point where it is now used by thousands of Unix and VMS programmers. Its appeal is its fully graphical orientation and its simple but very effective command set based on successful Macintosh and MS Windows editors. Though simple to learn and use, NEdit is very complete, and enables its users to work faster than they can with text-based editors like emacs, edt, and vi.

With the latest release, NEdit is available on at least twelve different platforms. Executables and source code can be fetched by anonymous ftp from fnpspb.fnal.gov.

NFit

NFit is a program based on MINUIT, for fitting curves to (mostly histogram) data. It works with typed-in expressions, or with compiled FORTRAN or C modules. The visual display and interactiveness provide users with a better understanding of the theoretical model on which the fit is based. We expect NFit to be completed within the next 3 to 4 months.

GUI Library Software

Also within the next 6 months, we intend to release a documented source and object code kit for our specialized plotting widgets. These can be used easily from any Motif or X-Toolkit based application to provide high-quality plots with the interactive rotation and scaling capabilities of Histo-Scope and NPlot.

Future Goals

Our long term goal is to create a complete programmable interactive data analysis package. The package will integrate both an interactive GUI part and an interpreted language. Whether we go ahead with this plan depends largely on how users react to our current offerings. So far the reactions from people who have used the software are very positive, but we would like to see a higher penetration of these tools into everyday use in HEP before we feel completely comfortable embarking on the larger project.

Conclusion

We hope these short descriptions have whet your appetite to try this software, and since the software is very easy to install and use, we guaranty that doing so will involve very little pain and personal suffering. These products also have the full support of the Fermilab Computing Division, so you can feel relatively safe that they won't disappear as soon as you get to rely upon them.

The Computing Environment for Physics Analysis for ZEUS Experiment at HERA

Olaf Manczak
DESY Hamburg, Germany
olavi@zow.desy.de

Abstract

We present here the computing environment, designed to meet computing and data storage needs of physics analysis at the ZEUS Experiment at HERA

1. Introduction

The ZEUS detector, operating in the HERA electron-proton collider at DESY, Hamburg, recorded 5 million particle collisions during its first data-taking period in 1993, and is expected to record two or three times that number in 1994. Typical event sizes for raw and reconstructed events are 150 KB and there are about 250 physicists at the ZEUS Collaboration working on data analysis at DESY. Therefore ZEUS requires for purposes of physics analysis, powerful and user friendly computing facilities with efficient access to a mass storage system containing several TBytes of data.

2. ZEUS computing environment

Over the last 2 years the ZEUS Experiment has moved most of its physics analysis environment (i.e. software development and computing intensive analysis) from an IBM mainframe and VAX cluster to a distributed, Unix based, client-server environment.

The present configuration is illustrated in figure 1. This new environment consists of personal desktop computing facilities (workstations and X Terminals) and powerful, central computing servers which provide direct access to TBytes of hierarchical mass storage. We found that central, multiprocessor machines with a simple batch system running on top of the Unix can preserve most of the mainframe functionality but deliver significantly more computing power at much lower price, while workstations and X Terminals are more comfortable, more human friendly, and more suitable for efficient data analysis and software development than traditional mainframe terminals. However, in many aspects this new distributed, computing model still lacks mainframe quality.

In fact, this new model of computing environment was a natural solution imposed by the following constraints:

a) ZEUS physicists will use several TBytes of reconstructed data; fast access to a huge part of that data is critically important for the efficient physics analysis.

b) Required CPU power is defined by an enormous amount of data and relatively large number of about 250 users (physicists working on physics analysis).

c) Data analysis, as it is done by most of physicists in HEP experiments, can be considered as an infinite loop of interactive and non-interactive steps. Interactive part includes transformation of "ideas" into a

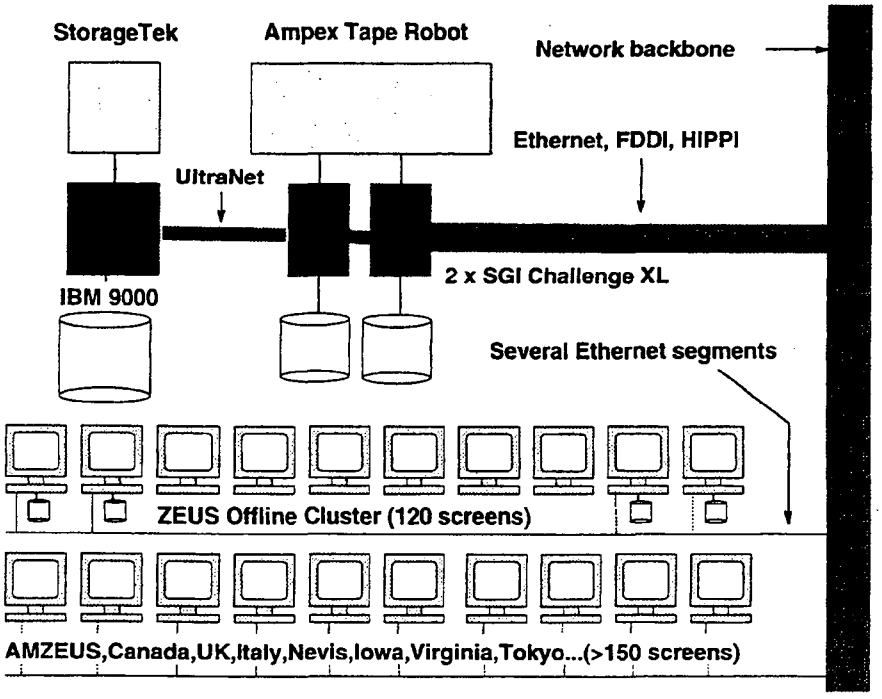


Figure 1. ZEUS computing environment

physics analysis program (software development) that runs in batch (non-interactive) mode for some time (typically five minutes to one week) to produce, so called, ntuples or histograms used later for interactive visual data analysis.

CPU requirements of the central batch facility (ZARAH) are met by using two SGI Challenge machines each with 18 R4400 150 MHz processors and 1 GB RAM providing about 3200 SPECint92 benchmark units. Both machines have access to an AMPEX tape robot and to STK silos connected to an IBM mainframe. SGI machines are connected to the DESY network backbone via Ethernet, FDDI and HIPPI interfaces. One machine is used strictly for reconstruction of physics events and other Challenge machine is used strictly as a batch platform for some 250 physicists. The interactive facility consists of several "clusters" of about 140 workstations (mainly DECstation 5000) and 130 X Terminals connected via several segments of Ethernet to the central network backbone. Idle CPU of these interactive workstations is efficiently used by FUNNEL - a distributed system for Monte Carlo generations.

A typical ZEUS physics analysis is performed using a standard program with user hooks. The user submits from his workstation to central batch facility a CPU intensive job which runs through a large sample of data producing relatively small output which is normally in the form of PAW ntuples or histograms which is subsequently analyzed in an interactive graphics session (locally on user's workstation). By operating in this fashion, we can effectively make use of both central and distributed CPU power and avoid network bottlenecks in accessing large amount of data.

Our batch system is based on the Network Queuing System (NQS) available in the pub-

lic domain. A customized client-server front end has been developed which allows the user to submit jobs remotely from their workstations and then retrieve results at job completion. A typical job consists of a controlling shell script and all source code required to compile the user code and link to the standard ZEUS libraries. The user may remotely query the system about job status, and at any time may transfer resulting output back to his workstation for subsequent analysis.

Fast access to data is provided by having a large (about 100 GB) permanent disk store containing a complete MINI-DST data sample of interest to all physics groups and a smaller subset of the most accessed Monte Carlo datasets. MINI-DSTs are a stripped down version of the full data sample, created by choosing a subset of detector information and physical quantities that is the minimum required for a typical physics analysis.

The rest of the data is available from tape storage. Raw, reconstructed and Monte Carlo datasets all reside on STK silos connected to the IBM mainframe. Files may be transferred to the disks at a rate of 1-2 MB/s via the fast UltraNet network. The total storage capacity of the STK silos is 30 TBytes, which is shared by all DESY experiments.

In addition to the STK robot there is a 6 TB storage capacity AMPEX tape robot available exclusively to the ZEUS experiment. The AMPEX drive is used primarily by the data reconstruction group and for storage of reconstructed and raw data sets and is not intended as a user file storage medium.

3. Conclusions

ZEUS has been using the "offline" computing environment described above since June 1993. Most analysis done by the ZEUS collaboration is now performed using the ZARAH central batch facility and the presence of a complete data set on disk has been found to be extremely valuable in terms of ability to analyze the data quickly. On any given day above 500 jobs are submitted by 60 different users to the batch queues. Currently about 100 GBytes of data per day is transferred from tape to the batch machine.

Future endeavors include obtaining additional 500 GBytes disk space to continue to provide complete disk data sets for the users and also the development of a event compression scheme which will effectively double the amount of available disk space without adding much CPU overhead to analysis jobs. The design of a more efficient tape staging system is also becoming more urgent as the demand for Monte Carlo tape data sets increases.

In summary, the move from a mainframe oriented computing environment to a distributed client-server based system has been highly successful. ZEUS is going to abandon use of the IBM mainframe in January 1995.

4. Acknowledgments

I would like to thank the following people for their contributions to ZARAH project: Lothar Bauerdick and Matthias Kasemann for the original idea and guidance, Dave Gilkinson and Oleg Derugin as ZARAH coworkers.

Session 7
Computation

The LSNDMC Monte Carlo

K. McIlhany
University of California, Riverside

D. Smith
Embry-Riddle Aeronautical University

A.M. Eisner, Y-X. Wang
University of California IIRPA

D. Whitehouse
Los Alamos National Laboratory*

Abstract

A Monte Carlo program describing the response of the Liquid Scintillation Neutrino Detector (LSND) at the Los Alamos Meson Physics Facility (LAMPF) was written using the GEANT geometry and simulation package. Neutrino interactions was simulated in the detector through the production of Cerenkov and scintillation light in the range of 2-3 eV. Since GEANT does not normally track photons to electron-volt energies, the tracking program (TRAK) was modified to produce both Cerenkov and scintillator light, the latter being simulated using the Birks equation. The LSND Monte Carlo program was used to predict the quantity of scintillator (b-PBD) used in the mineral oil to provide a ratio of roughly 4:1 light output resulting from scintillation and Cerenkov light respectively.

Introduction

The Large Scintillator Neutrino Detector (LSND) at the Los Alamos Meson Physics Facility (LAMPF) is an above-ground neutrino detector designed to search for neutrino oscillations. The detector is located 27 meters behind the LAMPF beam dump to insure that no charged particles coincident with the beam line enter the detector. The *appearance* of both ν_e and $\bar{\nu}_e$ in the LSND detector will give the best measurement of neutrino oscillations to date. The observation of electrons(positrons) in the appropriate energy range, as well as the subsequent capture of neutrons, can produce a signal unique to neutrino oscillations.

The Detector

The active volume of the LSND detector is cylindrically shaped, 6 meters in diameter and 9 meters in length. The longitudinal axis of symmetry is parallel to the LAMPF proton beam. The cylindrical volume is filled with mineral oil (180 tons of CH_2) doped with a scintillator (butyl-PBD). The inside surface of the detector is lined with 1220 8-inch Hamamatsu photomultiplier tubes covering approximately 25% of the inner wall with a sensitive photocathode surface.

The LSND detector was designed to produce Cerenkov and scintillation light in the visible region (2-3 eV). The scintillation light provides the vertex location while Cerenkov light is used to determine the momentum of the ejected charged particle, assuming $\beta > 1/n$, (n = the index of refraction of the mineral oil = 1.47). The combination of Cerenkov and scintillator detectors, merged into one large active volume, provides a unique opportunity to improve the sensitivity for neutrino oscillation searches.

A veto counter surrounds the active volume to reduce the triggers due to cosmic rays while an overburden above the veto is used to reduce the cosmic ray flux. Neutrino data were collected for two months this last summer using the LSNDMC detector.

LSND Monte Carlo -- LSNDMC

Geometry

Monte Carlo programs are usually "up and running" well before the detectors in most

experiments, and the LSND Monte Carlo (LSNDMC) was no exception. During the proposal phase of the LSND experiment, it was used to estimate the detector sensitivity and test reconstruction algorithms. During the building phase of the detector, the LSNDMC was used to "shake down" the data acquisition. Once the detector started taking data, the motivation for the monte carlo program shifted to other issues such as acceptance calculations, background calculations and comparison of monte carlo results to data. The LSNDMC program was constructed using GEANT because it provided a flexible geometry to "build" the detector in software and also met most of the needs listed above. However, a number of modifications needed to be made so that GEANT could simulate the production and tracking of visible light in the detector. These modifications resulted in the LSND monte carlo program (LSNDMC).

Once the design of the detector had been determined, the active volume, the veto, and the over-burden were implemented using the GEANT geometry package. The detector's simple design was an easy task for the GEANT geometry package and no modifications were required in this area.

Event Input
The LSNDMC program is set up to read two kinds of event generators. The first event generator is the internal generator already packaged in GEANT. However, in order to simulate the numerous sources of particles in the LSND detector, a feature was added to read in generated events from multiple external files (i.e., multiple input streams). Each file would contain a separate class of events, such as cosmic rays, background neutrons, muon decays, etc. Each file was also assigned a *rate* in which events were injected into the LSNDMC. This feature permits the user to study the backgrounds of rare events. Both of these event generators, as well as full schematic of the LSNDMC, are shown in figure 1.

Light Production
The production of visible light in the detector was also simulated using the GEANT program. Three processes produce visible light in the detector: scintillation, Cerenkov, and fluorescence. When reconstructing the light signals from the detector, one tries to differentiate the isotropic light from the directional light (i.e., the Cerenkov cone). This has the effect of separating position information from direction information.

The isotropic component is made up of direct scintillation light along with short wavelength Cerenkov photons re-radiating through fluorescence. The direct scintillation light (i.e., from charged particle) is proportional to the deposited energy in a step times a correction for saturation. The light output, L, is calculated using Birks' formula[1,2].

$$\frac{dL}{dx} = \frac{A\frac{dE}{dx}}{1 + kB\frac{dE}{dx}} \qquad (1)$$

The amount of directional light (i.e., Cerenkov radiation) is calculated over a range of 300nm - 700nm using the equation

$$N = N_o \sin^2 \theta_c \qquad (2)$$

where N_o is 873 photons/cm.

Tracking
Once the light is generated, it is tracked through the detector using the GEANT tracking program. The low-energy photons were internally assigned to geantinos where their interactions were controlled by the programmer. While in the tracking program, the photons could be attenuated in the active volume, reflected, absorbed (e.g., at a wall), or hit a phototube. If the photon hit a phototube, the probability of producing a photoelectron was calculated using the quantum efficiency as a function of wavelength.

The final feature added to the tracking was a method to "feed back" delayed events such as neutron captures and muon decays. This process is shown in figure 1. These events were given a time-stamp consistent with their respective process and re-introduced into the "Event Timestream."

LSNDMC Output
Finally, the PMT hits are converted into charge-time information and smeared according their respective resolutions. A typical Michel electron event is shown in figure 2. The two large circles highlight the Cerenkov cone produced by the LSNDMC generated electron (45 MeV) as well as the

subsequent fit to the Cerenkov cone. The smaller circles are proportional to the number of photoelectrons. The spurious hits outside the Cerenkov cone are produced primarily by scintillation light.

Conclusions

The LSNDMC program has already made a significant contribution to the LSND neutrino experiment by determining the optimum amount of scintillator to use in the detector (0.0315 grams/liter). This concentration was shown to give the best results when combining both the isotropic and directional components of visible light.

The LSNDMC program has also reproduced the endpoints and moments of various distributions, such as number of "hit" tubes and charge distributions. It also reproduces the detector resolutions.

The present operation of the LSND detector and its ability to reconstruct neutrino events is well understood when compared to the LSNDMC program. The monte carlo program will continue to play an important role when comparing the detector's response to rare neutrino events along with neutrino oscillation candidates. The calculation of background events will be crucial to validating any oscillation signal or determining a limit.

Acknowledgments

The authors would like to thank Steve Yellin for his contributions to tuning the LSNDMC program. This work was also supported by NSF contract PHY-9203087.

[1] J. Birks, Proc. Roy. Soc. Edinburgh A170 (1971/72) 22.

[2] R.A. Reeder, et al., Nuc. Inst. and Methods in Physics Research A 334 (1993) 353-366.

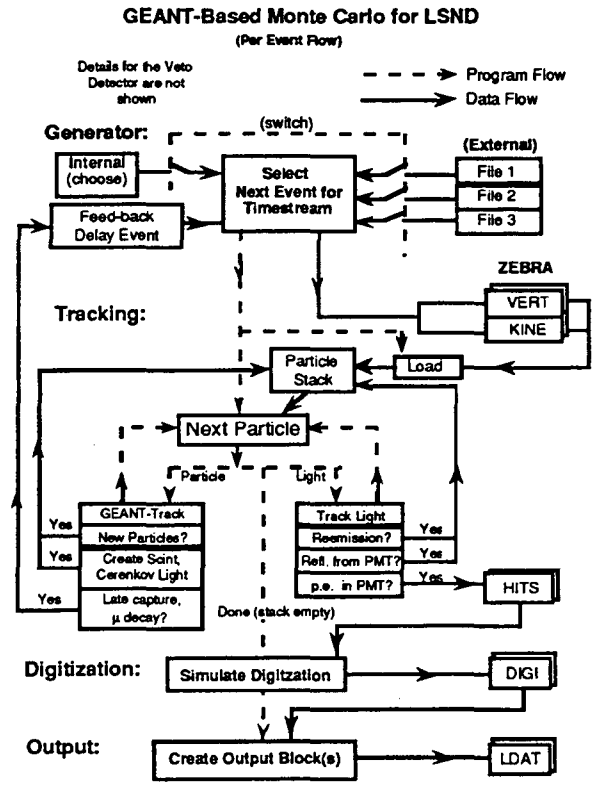


Fig. 1 The LSNDMC flowchart describing the LSND modifications GEANT.

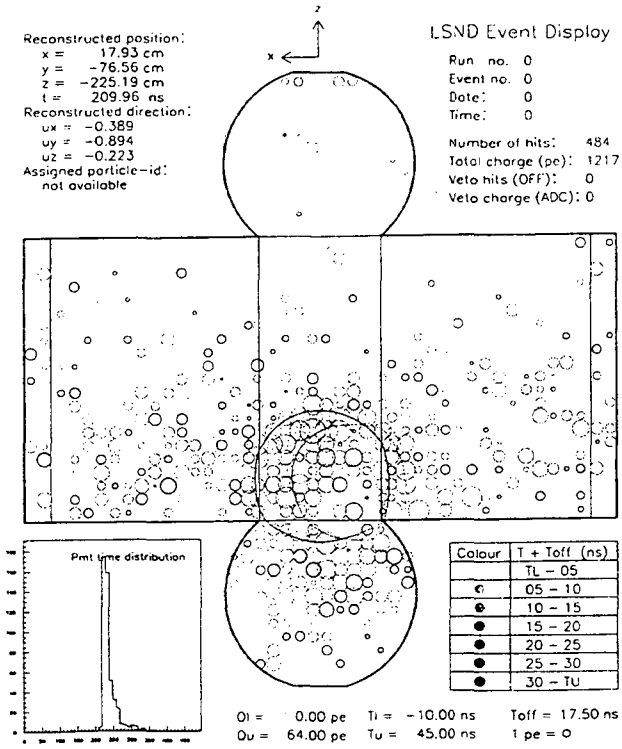


Fig. 2 A typical Michel electron event generated by LSNDMC. The unfolded detector show the hit phototubes.

ROBUST ESTIMATION METHODS APPLIED TO VERTEX RECONSTRUCTION AND TRACK ASSOCIATION IN A COLLIDING BEAM EXPERIMENT

Winfried A. Mitaroff
Institut für Hochenergiephysik
Austrian Academy of Sciences
A-1050 Vienna, Austria

Abstract

This note presents a novel technique for finding and estimating secondary vertices in particle physics collisions. It is based on a robust estimation method (M-estimator) which, contrary to estimators used so far (least squares resp. Kalman filter), has the advantage of downweighting the influence of outlier tracks on a vertex fit. The different methods are systematically compared in a Monte Carlo study of charm and bottom decays, with the M-estimator demonstrating its superiority. The new method has also been applied to the analysis of real data in the DELPHI experiment at LEP.

INTRODUCTION

A new generation of complex superdetectors is providing high-precision track measurements by micro-vertex detectors close to the beam tube, and efficient particle identification by RICH and muon detectors covering most of the solid angle. It is therefore desirable to complement these informations by an optimal primary and secondary vertex evaluation.

The straightforward least-squares technique, well-known since the old bubble chamber experiments, becomes prohibitive at collider energies with high multiplicties n: the computing required for one fit increases proportional to n^3, and has to be repeated many times if ambiguities exist in the association of tracks to vertices (combinatorial overhead).

A solution to this problem can be found in system theory by the Kalman filter and smoother method. It is equivalent to least-squares, but the computing increases only linearly with n, except for the full covariance matrix requiring computations proportional to n^2 [1]. In addition, it provides a test criterion for track association, based on the χ^2 statistics.

But the efficiency of these tests is limited by distortions caused by outlier tracks (e.g. secondary vertex tracks in the primary vertex fit), which is due to the L_2-norm objective function used. This can be overcome by a robust (necessarily non-linear) estimator, downweighting the influence of outliers. We have chosen an M-estimator, which can be implemented as an iterated Kalman filter formalism with modified weights, for testing track association to the primary vertex.

VERTEX ESTIMATION

Given are n reconstructed particle tracks in a magnetic field, defined at some reference surface by their 5 parameters $\boldsymbol{p}_k$ and corresponding covariance matrices $\boldsymbol{V}_k = \boldsymbol{G}_k{}^{-1}$, which are regarded as "virtual measurements" for the vertex fit. The task is to estimate the position $\boldsymbol{x}$ of and the momenta $\boldsymbol{q}_k$ at the common vertex, and also to provide test criteria for the association of each track to this vertex (pattern recognition).

The Kalman filter

For a thorough introduction see [2]. The "state vector" $\boldsymbol{Q} = (\boldsymbol{x}, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)$ to be estimated is updated for each added track, starting with the beam profile $\boldsymbol{x}_0$ and its covariance matrix $\boldsymbol{C}_0$ (if not available, set $\boldsymbol{C}_0^{-1} = 0$). The track model determines the measurement equation,

$$\boldsymbol{p}_k = \boldsymbol{h}_k(\boldsymbol{x}, \boldsymbol{q}_k) + \epsilon_k, \quad \mathrm{cov}(\epsilon_k) = \boldsymbol{V}_k$$

which is to be linearized by a first order Taylor expansion at some point $(\boldsymbol{x}_e, \boldsymbol{q}_{k,e})$:

$$\boldsymbol{h}_k(\boldsymbol{x}, \boldsymbol{q}_k) \approx \boldsymbol{A}_k\boldsymbol{x} + \boldsymbol{B}_k\boldsymbol{q}_k + \boldsymbol{c}_{k,e}$$

where $\boldsymbol{A}_k = [\partial \boldsymbol{h}_k/\partial \boldsymbol{x}]_e$, $\boldsymbol{B}_k = [\partial \boldsymbol{h}_k/\partial \boldsymbol{q}_k]_e$ are the (5×3) Jacobians at $(\boldsymbol{x}_e, \boldsymbol{q}_{k,e})$.

The "Kalman filter" performs the addition of a new track k to a vertex already fitted with $k-1$ tracks:

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{C}_k[\boldsymbol{C}_{k-1}^{-1}\tilde{\boldsymbol{x}}_{k-1} + \boldsymbol{A}_k{}^T\boldsymbol{G}_k{}^B(\boldsymbol{p}_k - \boldsymbol{c}_{k,e})]$$
$$\tilde{\boldsymbol{q}}_k = \boldsymbol{W}_k\boldsymbol{B}_k{}^T\boldsymbol{G}_k(\boldsymbol{p}_k - \boldsymbol{A}_k\tilde{\boldsymbol{x}}_k - \boldsymbol{c}_{k,e})$$
$$\mathrm{cov}(\tilde{\boldsymbol{x}}_k) = \boldsymbol{C}_k = (\boldsymbol{C}_{k-1}^{-1} + \boldsymbol{A}_k{}^T\boldsymbol{G}_k{}^B\boldsymbol{A}_k)^{-1}$$
$$\mathrm{cov}(\tilde{\boldsymbol{q}}_k) = \boldsymbol{D}_k = \boldsymbol{W}_k + \boldsymbol{E}_k{}^T\boldsymbol{C}_k^{-1}\boldsymbol{E}_k$$
$$\mathrm{cov}(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{q}}_k) = \boldsymbol{E}_k = -\boldsymbol{C}_k\boldsymbol{A}_k{}^T\boldsymbol{G}_k\boldsymbol{B}_k\boldsymbol{W}_k$$

with

$$\boldsymbol{W}_k = (\boldsymbol{B}_k{}^T\boldsymbol{G}_k\boldsymbol{B}_k)^{-1}$$
$$\boldsymbol{G}_k{}^B = \boldsymbol{G}_k - \boldsymbol{G}_k\boldsymbol{B}_k\boldsymbol{W}_k\boldsymbol{B}_k{}^T\boldsymbol{G}_k$$

Each filter step has a chi-square of

$$\chi^2_{k,F} = (\tilde{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_{k-1})^T\boldsymbol{C}_{k-1}^{-1}(\tilde{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_{k-1}) + \boldsymbol{r}_k{}^T\boldsymbol{G}_k\boldsymbol{r}_k$$
$$\boldsymbol{r}_k = \boldsymbol{p}_k - \boldsymbol{A}_k\tilde{\boldsymbol{x}}_k - \boldsymbol{B}_k\tilde{\boldsymbol{q}}_k - \boldsymbol{c}_{k,e}$$

with two degrees of freedom.

After all n tracks have been added, the "smoother" recomputes momenta and covariances with the final vertex position:

$$\tilde{\boldsymbol{q}}_k^n = \boldsymbol{W}_k\boldsymbol{B}_k{}^T\boldsymbol{G}_k(\boldsymbol{p}_k - \boldsymbol{A}_k\tilde{\boldsymbol{x}}_n - \boldsymbol{c}_{k,e})$$
$$\mathrm{cov}(\tilde{\boldsymbol{x}}_n, \tilde{\boldsymbol{q}}_k^n) = \boldsymbol{E}_k^n = -\boldsymbol{C}_n\boldsymbol{A}_k{}^T\boldsymbol{G}_k\boldsymbol{B}_k\boldsymbol{W}_k$$
$$\mathrm{cov}(\tilde{\boldsymbol{q}}_k^n) = \boldsymbol{D}_k^n = \boldsymbol{W}_k + \boldsymbol{E}_k^{nT}\boldsymbol{C}_n^{-1}\boldsymbol{E}_k^n$$

If required, the full covariance matrix and the total chi-square of the fit are

$$\mathrm{cov}(\tilde{\boldsymbol{q}}_k^n, \tilde{\boldsymbol{q}}_j^n) = \boldsymbol{E}_k^{nT}\boldsymbol{C}_n^{-1}\boldsymbol{E}_j^n, \quad k \neq j$$
$$\chi^2_n = (\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_n)^T\boldsymbol{C}_0^{-1}(\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_n) + \textstyle\sum_{k=1}^n \boldsymbol{r}_k^{nT}\boldsymbol{G}_k\boldsymbol{r}_k^n = \sum_{k=1}^n \chi^2_{k,F}$$
$$\boldsymbol{r}_k^n = \boldsymbol{p}_k - \boldsymbol{A}_k\tilde{\boldsymbol{x}}_n - \boldsymbol{B}_k\tilde{\boldsymbol{q}}_k^n - \boldsymbol{c}_{k,e}$$

which has $2k$ (resp. $2k-3$) degrees of freedom. Track association may be tested in a symmetric way by removing track k from the vertex fit ("inverse filter"):

$$\tilde{\boldsymbol{x}}_k^{n*} = \boldsymbol{C}_k^{n*}[\boldsymbol{C}_n^{-1}\tilde{\boldsymbol{x}}_n - \boldsymbol{A}_k{}^T\boldsymbol{G}_k{}^B(\boldsymbol{p}_k - \boldsymbol{c}_{k,e})]$$
$$\mathrm{cov}(\tilde{\boldsymbol{x}}_k^{n*}) = \boldsymbol{C}_k^{n*} = (\boldsymbol{C}_n^{-1} - \boldsymbol{A}_k{}^T\boldsymbol{G}_k{}^B\boldsymbol{A}_k)^{-1}$$
$$\chi^2_{k,S} = (\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_k^{n*})^T(\boldsymbol{C}_k^{n*})^{-1}(\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_k^{n*}) + \boldsymbol{r}_k^{nT}\boldsymbol{G}_k\boldsymbol{r}_k^n$$

This smoothed chi-square $\chi^2_{k,S}$ is the test criterion (a). But if there are several outlier tracks, $\tilde{\boldsymbol{x}}_k^{n*}$ is distorted by the remaining ones, and the test becomes less powerful. This will be shown.

The M-estimator

Vertex estimation can be made less sensitive to outliers by using a "robustified" objective function, downweighting measurements with large residuals. First, all weight matrices are to be diagonalized by n orthogonal transformations,

$$\boldsymbol{G}_k \rightarrow \boldsymbol{U}_k\boldsymbol{G}_k\boldsymbol{U}_k{}^T, \quad \boldsymbol{p}_k \rightarrow \boldsymbol{U}_k\boldsymbol{p}_k$$

yielding a global measurement vector

$$\boldsymbol{P} = (\boldsymbol{x}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$$

with its diagonal covariance matrix

$$\boldsymbol{V} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_{5n+3}^2)$$

With $a_{ij} = [\partial P_i/\partial Q_j]_e$ and the residuals

$$r_i = P_i - \textstyle\sum_{j=1}^{3n+3} a_{ij}Q_j - c_{i,e}$$

a generalized objective function would be

$$\mathcal{M}(\boldsymbol{Q}) = \textstyle\sum_{i=1}^{5n+3} \Psi(r_i/\sigma_i)$$

Defining $\psi(t) = d\Psi/dt$, the L_2-norm (Kalman filter) is chosen by $\psi(t) = t$, and the L_1-norm by $\psi(t) = \mathrm{sgn}(t)$. Following [3], we choose a combination of both,

$$\psi(t) = \begin{cases} t, & |t| \le R \\ R \cdot \mathrm{sgn}(t), & |t| > R. \end{cases}$$

(R = constant of robustness, usually between 1 and 2). Minimizing $\mathcal{M}(\boldsymbol{Q})$ yields

$$\partial\mathcal{M}/\partial Q_j = \sum_{i=1}^{5n+3} (r_i/\sigma_i) \cdot (a_{ij} w_i/\sigma_i) = 0$$

with the extra weight factors

$$w_i = \frac{\psi(r_i/\sigma_i)}{r_i/\sigma_i} = \begin{cases} 1, & |r_i| \le R\sigma_i \\ R\sigma_i/|r_i|, & |r_i| > R\sigma_i \end{cases}$$

Thus, the M-estimator can formally be obtained by the Kalman filter method, but with a modified global weight matrix

$\boldsymbol{G}' = \mathrm{diag}(w_1/\sigma_1^2, \ldots, w_{5n+3}/\sigma_{5n+3}^2)$

downweigthing the influence of outlier measurements. The w_i depend on the r_i still unknown, so they must be computed iteratively, starting with all $w_i = 1$.

Test criteria for track association are:
(b) $\chi^2_{k,S}$ with original weight matrix $\boldsymbol{G}_k$;
(c) $\chi^2_{k,S}{}'$ with the final downweighted $\boldsymbol{G}_k{}'$;
(d) $\pi_k = \prod_{i=5k-1}^{5k+3} w_i$ (final extra weights).

The smoothed "chi-squares" $\chi^2_{k,S}$ and $\chi^2_{k,S}{}'$ are no longer χ^2-distributed. After a possible removal of outlier tracks, the final vertex fit is re-done by the Kalman filter and smoother, yielding correct covariance matrix and total chi-square.

MONTE CARLO STUDY

We present the results of a simulation of charm and bottom decays in the DELPHI detector [4], comparing the efficiencies of different methods (Kalman filter, M-estimator and impact parameters) for a separation of primary and secondary vertex tracks.

A detailed description of event generation, detector simulation and track reconstruction is given in [5]. The vertex reconstruction package developed by the author performs the following steps:

1. First approximate primary vertex fit, either by the Kalman filter, or by the M-estimator ($R = 1.5$).

2. Tests of track association, using either one of the test criteria (a–d) defined above, or the impact parameter in space resp. projected.

The efficiency of each test criterion is measured by (i) plotting its distribution separately for primary vertex (PV) and secondary vertex (SV) tracks, the latter for 5 intervals of the distance δ between PV and SV; (ii) determining the cut value for a fixed quantile $\alpha = 10$ % loss of good PV tracks (type I errors); (iii) evaluating the corresponding percentages β of contamination with bad SV tracks (type II errors), which should be low. The graphs $\beta(\delta_{centre})$ for 6 test criteria are shown in Figs. 1a and 1b for the charm and bottom samples, respectively.

Results

The Kalman filter and M-estimator tests (a–d) are clearly superior to the usual impact parameter tests, which use only part of the information available.

The M-estimator tests (b–d) do not significantly differ from the Kalman filter test (a) in the case of charm, but are clearly superior in the case of bottom. This is due to the kinematical differences between the two samples:

Charm events have a characteristic topology of two opposite secondary vertices with jet-like 2- or 3-particle de-

cays. Therefore, a first approximate primary vertex fit by least-squares is only slightly distorted by SV tracks, and the M-estimator cannot give substantial improvement.

Bottom mesons, because of their higher masses, have secondary vertices with lower momenta and wider decay angles than charm mesons, and show also cascade decays. Such events get significantly distorted by outlier tracks in a first approximate primary vertex fit by least-squares. The M-estimator is now able to exhibit its superiority.

An overall comparison between the two samples shows all six test criteria being less efficient for charm events than for bottom events for $\delta > 5$ mm, as expected from the kinematical differences of the two samples.

REAL SHORT-LIVED DECAYS

The power of the M-estimator method is demonstrated by an analysis of real DELPHI data of 1992, using selected hadronic events with at least 2 identified muons, and looking for short-lived decays into J/ψ and ψ_{2S}.

Vertex reconstruction is now performed by the following steps:

1. First approximate primary vertex fit by the M-estimator ($R = 1.5$).
2. Tests of track association, using test criterion (c) of above.
3. Secondary vertex search by combinatorial bundling, optimized either for 2-prongs with $\mu^+\mu^-$, or for 4-prongs with $\mu^+\mu^-\pi^+\pi^-$.
4. Final primary and secondary vertex fits by the Kalman filter method.

Preliminary results

Decays into $J/\psi \to \mu^+\mu^-$ are seen in the effective mass $(\mu^+\mu^-)$ of reconstructed 2-prong and 4-prong secondary vertices (Figs. 2a and 2b).

Decays into $\psi_{2S} \to J/\psi\pi^+\pi^-$ show up in the effective mass $(\mu^+\mu^-\pi^+\pi^-)$ of reconstructed 4-prong secondary vertices (Fig. 2c). Applying an additional cut on $3.0 \leq m_{eff}(\mu^+\mu^-) \leq 3.2$ MeV, 8 unambiguous event candidates are found with a background of ca. 3 (Fig. 2d).

These events can be attributed to bottom decays $B \to J/\psi X$ resp. $B \to \psi_{2S} X$ and are therefore of particular physical interest.

A comparison with simulated J/ψ events gives agreement without any background (which might be caused by reconstruction inefficiencies) for 2-prong and 4-prong secondary vertices.

ACKNOWLEDGEMENTS

The author wishes to thank R. Frühwirth and M. Regler for their substantial contributions to the methods described, and P. Kubinec and D. Liko for their cooperation in the analysis of real data.

REFERENCES

1. P. Billoir, R. Frühwirth, M. Regler: Nucl.Instr.Meth. A241 (1985) 115.
2. R. Frühwirth: Nucl.Instr.Meth. A262 (1987) 444.
3. R. Frühwirth: PhD Thesis, University of Technology, Vienna 1988.
4. P. Aarnio et al. (DELPHI Collab.): Nucl.Instr.Meth. A303 (1991) 233.
5. R. Frühwirth, P. Kubinec, W. Mitaroff, M. Regler: Comp.Phys.Comm. (in preparation).

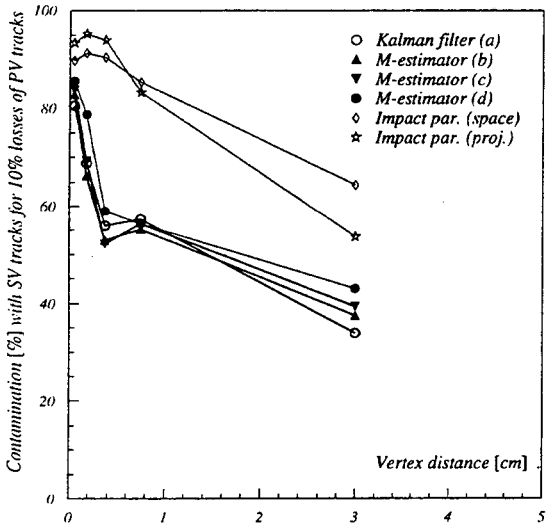


Fig. 1a: Comparison of test criteria (charm)

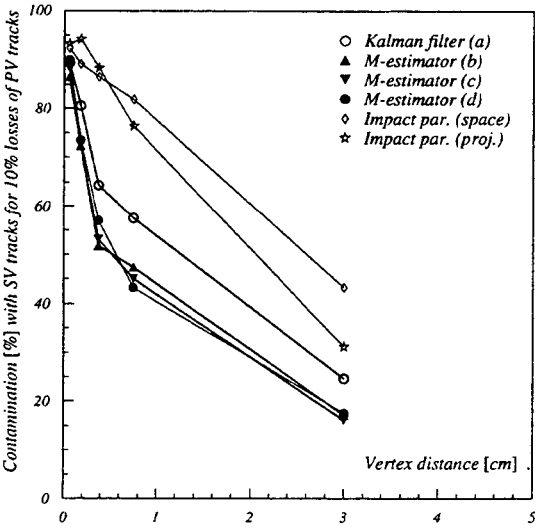


Fig. 1b: Comparison of test criteria (bottom)

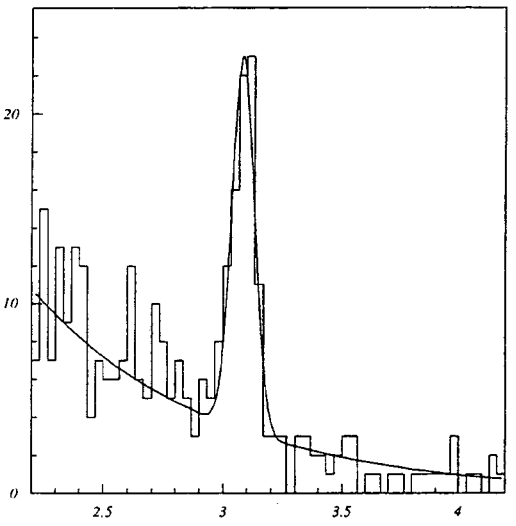


Fig. 2a: $m_{eff}(\mu^+\mu^-)$ in sec. 2-prong vertices

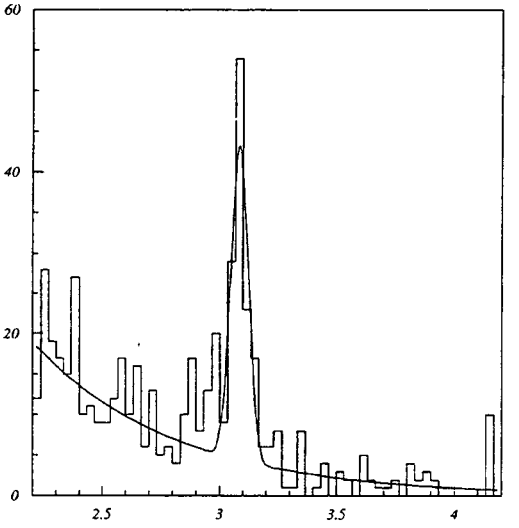


Fig. 2b: $m_{eff}(\mu^+\mu^-)$ in sec. 4-prong vertices

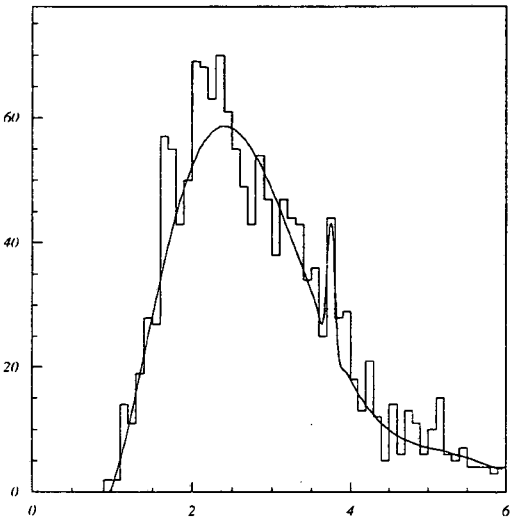


Fig. 2c: $m_{eff}(\mu^+\mu^-\pi^+\pi^-)$ in sec. 4-prong vertices

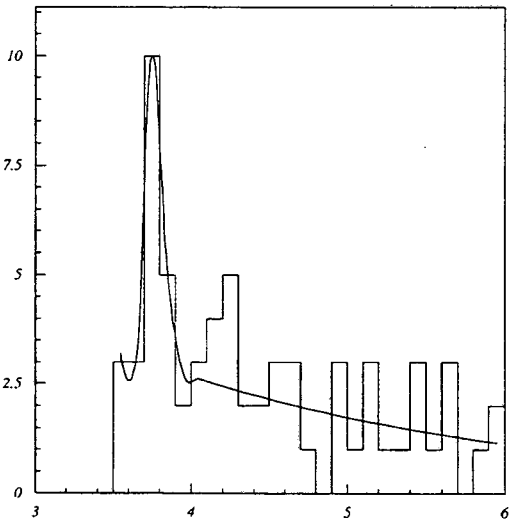


Fig. 2d: $m_{eff}(\mu^+\mu^-\pi^+\pi^-)$ with cut on $m_{eff}(\mu^+\mu^-)$

THE INVESTIGATIONS OF THE INFLUENCE OF THE SDC MBT MAGNET FIELD NONUNIFORMITY ON MUON MOMENTUM RESOLUTION

J.A.Budagov, S.B.Vorozhtsov,
A.M.Khasanov and N.O.Poroshin
JINR Dubna Russia

Abstract

The first results of the simulation work on understanding, how magnet field nonuniformity influences on muon momentum resolution, are presented. The investigations were performed on the base of SDC Muon Barrel Toroid (MBT), but we hope, the methods and some our conclusions may be used in other experiments, in LHC, for example.

INTRODUCTION

Sometimes, by the different calculations the possible magnet field nonuniformity is not taken into account. The definition of the muon momentum resolution as function of pseudorapidity η for SDC MBT [1] was made in [2]. The magnet field is considered there as uniform. However, in reality the permeability for different blocks of steel may be dispersed. As it was announced in "Minutes of SDC MBT Meeting – Apr.7, 1993" the permeability measurements for the 24 samples of steel from ATOMMASH heat 43 were in range between 152 and 163.

The aim of our simulation work is to understand how magnet field nonuniformity influences on muon momentum resolution.

MODEL OF THE MAGNET FIELD

More detailed description of the magnet and magnet field distribution may be found in [3]. Briefly, the magnet has a toroidal – octagonal shape with length along Z axis 28.032m, the steel thickness 1.5m and consists from 192 plates with thickness 0.14m along Z.

Two ideal magnet field models were considered.

Model 1: The field is equal to zero outside the steel and is equal to 1.8 Tesla for all plates, the field vector in steel is tangent to the circles with center in IP (Interaction Point), in XOY plane, the field integral from IP through the midpoint of any face of the magnet is 1.5m*1.8T.

Model 2: As in Model 1, but the permeability and consequently magnet field are different for each plate and uniformly distributed with mean 1.8T and half of width ΔB=0. (case of Model 1), 0.1, 0.2,

0.3 and 0.5 Tesla.

CALCULATION TECHNIQUES

The calculations were performed by GEANT3.14 [4], implemented for PC [5], and GEANT3.15 on SUN Station.

The first variants of our calculations were faulted because of not enough machine-precision of original GEANT code for muon energy more than 100 GeV. For the following simulations we have introduced double precision in some GEANT routines. This allowed us to obtain the suitable results subsequently.

Here we would like to study the influence of the magnet field nonuniformity only. We don't take into account the space resolution of detectors and reconstruction algorithms on this first step. It's supposed to do in our following investigations.

To get muon momentum we used the bending angle (Θ) between the incoming into and outcoming from MBT muons. When Θ is small, P is proportional to Θ. By GEANT simulation the following main physical processes were included: Bremsstrahlung, Landau fluctuations and Gaussian scattering.

For obtaining momentum resolution we used the following algorithm:

- R.M.S. for all events were obtained,
- new mean and new R.M.S. were recalculated for events with bending angle inside $\pm$3RMS range from mean,
- $\sigma\Theta/\Theta$ is considered to be equal to $\sigma p_t/p_t$.

RESULTS

Model 1: On the Fig.1 we show for comparing the results of calculations from [2]. Our results for muon momentum resolution as a function of η for p_t=0.1, 0.3, 1.0, 3.0 TeV are presented on the Fig.2.

As seen, momentum resolution is independent of muon p_t, because Θ and Gaussian scattering angles are proportional to 1/P.

Comparing the pictures on Fig.1 and 2 one can see, that space precision of detectors has more strong influence on momentum resolution than physical processes inside MBT.

Model 2: We have performed series of the calculations for ΔB=0.1, 0.2, 0.3, 0.5 Tesla and for p_t=0.1, 0.3, 1.0, 3.0 TeV. On Fig.3 we present muon momentum resolution versus as p_t and η, as magnet field nonuniformity.

From comparison Fig.3 with Fig.1 we can conclude – the distribution of steel permeability weakly influences on momentum resolution and becomes comparable, for example, in case p_t=1.0, η=0. at ΔB=0.2T with Fig.1.

CONCLUSIONS

The simple short model of MBT was made by GEANT system. Two ideal models of MBT magnet field were considered. The simulation allowed to conclude, that:

- muon momentum resolution is independent of muon p_t
- space precision of detectors have more strong influence on momentum resolution than physical processes inside MBT
- the distribution of steel permeability weakly influences on momentum resolution at $\Delta B < 10$

Fig.1. Muon momentum resolution for P_t=0.1,0.3,1.0,3.0 TeV/c (from /1/).

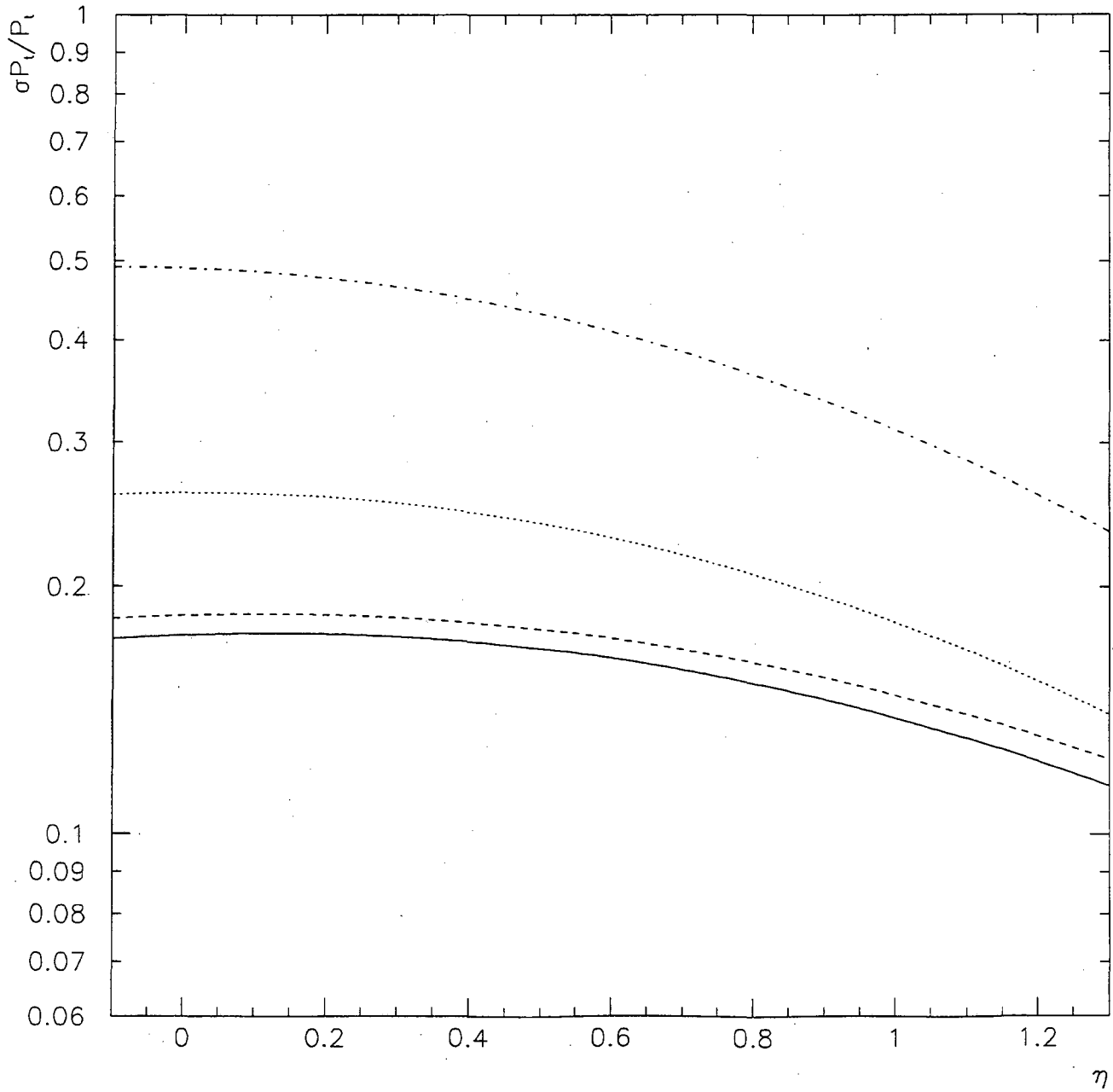


Fig.2. Muon momentum resolution for P_t=0.1,0.3,1.0,3.0 TeV/c and ΔB=0.T

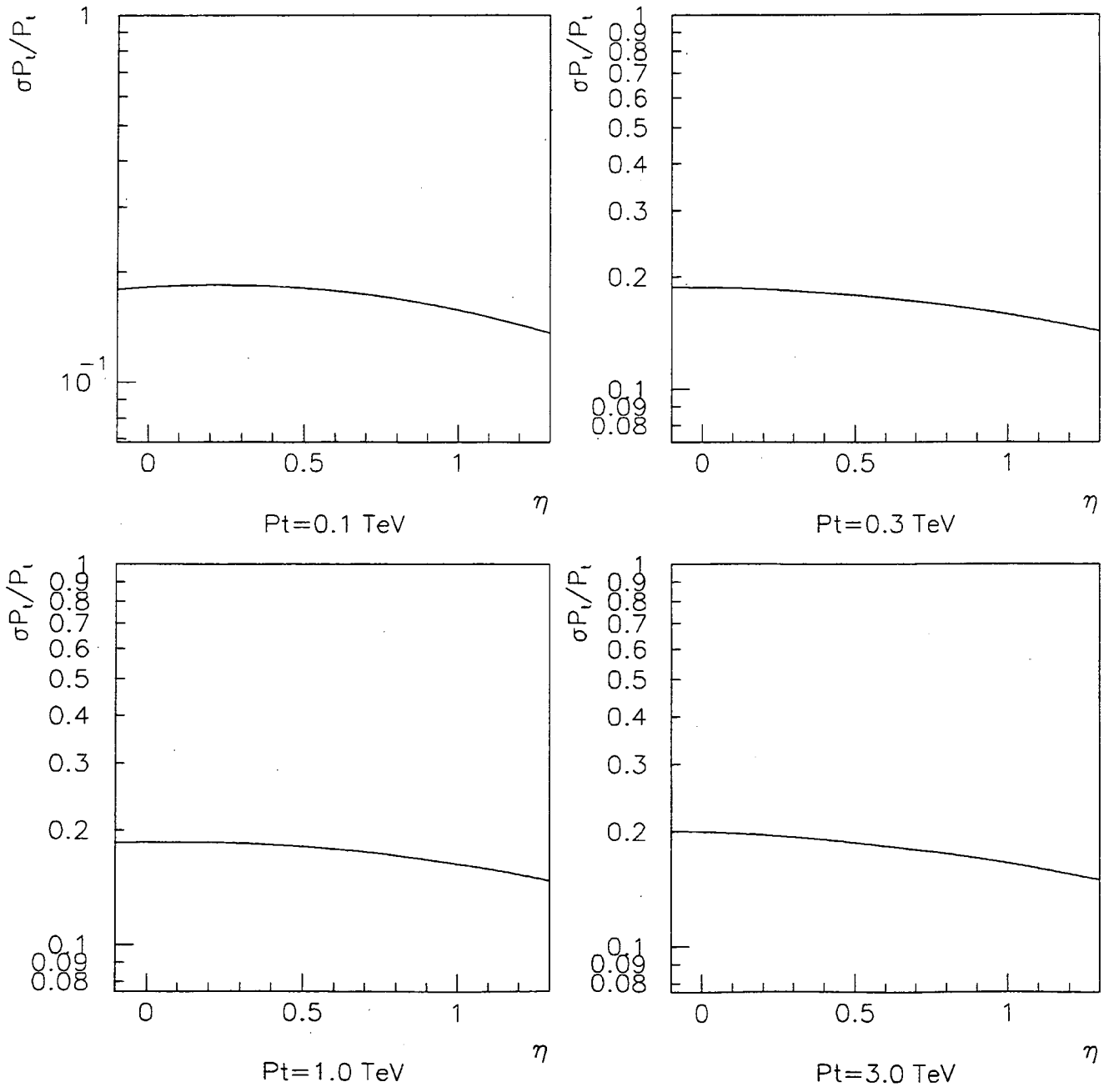
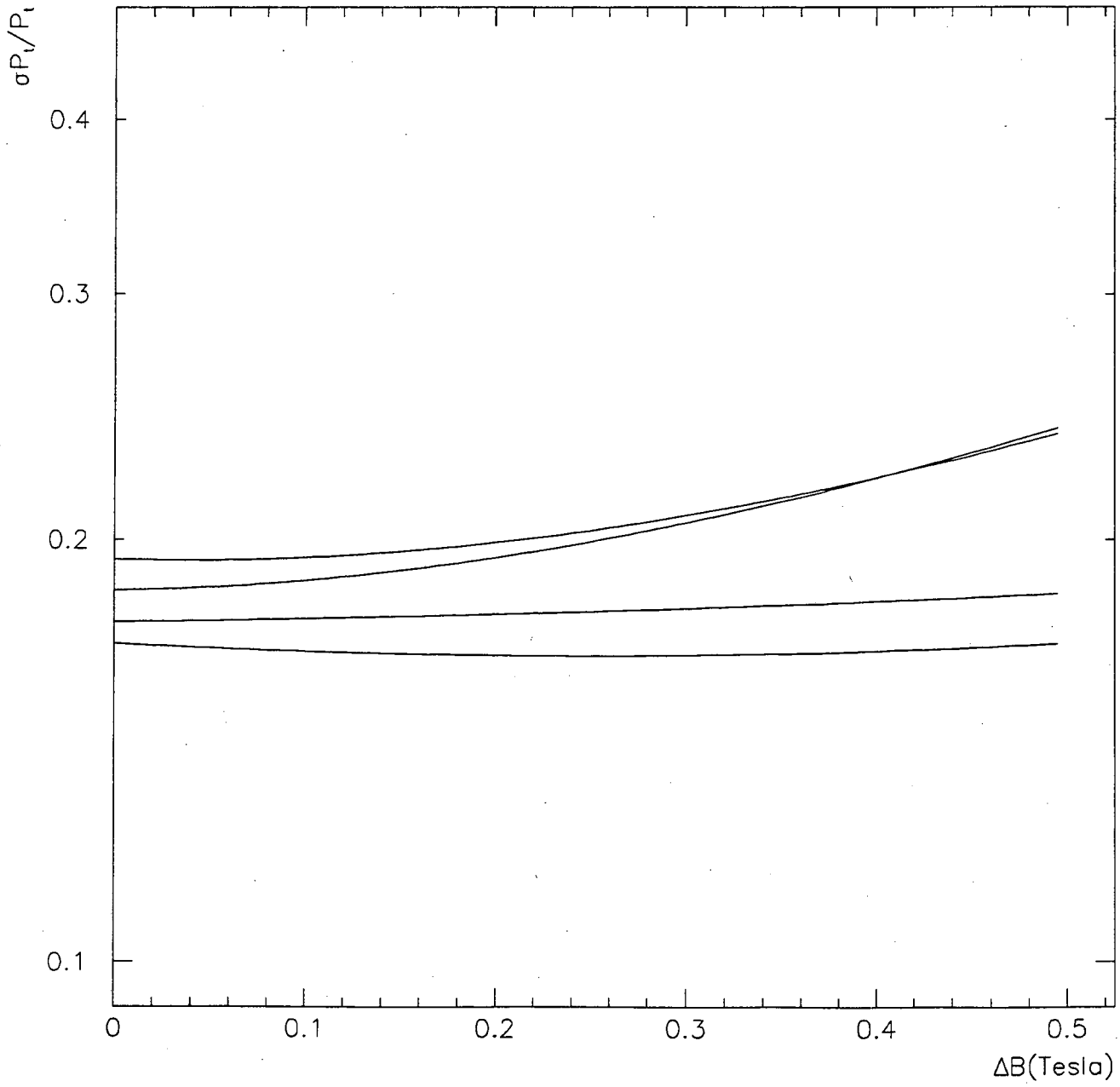


Fig.3. Muon momentum resolution vs magnet field nonuniformity for P_t=0.1 and 1.0 TeV/c and η=0. (highest) and η=1.0 (lowest two curves)



Distributed event simulation with PVM

Shingo Ichii, Takashi Sasaki and Yoshinobu Takaiwa
KEK, National Laboratory for High Energy Physics

Abstract

We implemented the distributed event simulation program based on GEANT with PVM. The measured performance shows the PVM can be used in HEP practically , if the number of processors are not so large. Still we need further study to use it under the environment which have huge number of processors.

INTRODUCTION

Future experiments, such as KEK B factory, are expected to have much higher event rates, *i.e.* 10^9 events/year, and require huge number of simulated events for the analysis. To process them, some sort of mechanism for parallelization of event simulator is necessary. We decided to try event parallelization scheme for our event simulator because of easiness of implementation. PVM, Parallel Virtual Machine [1], is a software system that provides a parallelization scheme for workstations and other computers connected by the network or high speed buses , which is available free of charge. It consists of process control part and libraries for C and FORTRAN, and it is easily applicable for existing programs. PVM also has its own machine independent I/O mechanism using a portable networking library to support multi-vendor configurations. This software is used very widely in various field including High Energy Physics already. We decided to use this for the basis of our implementation.

We have previously studied the parallelization of the AMY detector simulator on a Sparc1 based multiprocessor computer AP1000[2]. From this experience, it was proven that I/O is a bottle neck of HEP application due to heavy I/O rate. Some of other parallelization schemes use NFS to write the result on disks for communication, but the performance of NFS is poor even if the faster network such as FDDI are used due to the way NFS is implemented to it. Since extremely high I/O band width is required for HEP simulation, PVM may provide the solution for this problem, too. Utilizing this software system, we parallelize the event simulator for the B factory.

PVM

The PVM software package is developed by Jack Dongarra et. al. at Oak Rigde National Laboratory and distributed freely. PVM provides the user process control and message passing li-

brary for communication among tasks running even on different machines. PVM supports C and even FORTRAN binding on variety of machines. Some other parallelizing scheme on distributed computing only support homogenous environment, but PVM support heterogeneous environment which have variety of machines, like workstations and also parallel computers. Communication and process control between computers are provided by the pvmd3 daemon. User can control all of the process giving commands to the pvmd3 daemon from the pvm console command to start or halt the PVM slave process. And PVM can be installed and used without requiring super user privilege.

The messaging passing mechanism have both machine independent and dependent mode, and user can choose. For machine independent mode, XDR is used inside. This guarantee the portability of the PVM software package.

When the parent process is invoked, it can spawn slave process and can be communicate each other as showing Figure 1. To make message passing, data must be packed into buffer according to data type. Then, master send data to slave processes. PVM has both broadcast and point to point message mechanisms. Slave process receive data and also can send data to master.

IMPLEMENTATION

Our event simulator is based on GEANT 3.15 [3], but we are using TBS, TRISTAN Bank System [4], for event I/O. This software have I/O routines TBREAD and TBWRIT to read and write data respectively. Within GEANT framework, the subroutine GUKINE reads event records and the other subroutine GUOUT writes the resulting event. Inside those routine, we use the TBREAD subroutine to read records and the TBWRIT subroutine to write the result. To parallelize our simulator, we implemented the subroutines TBRECV and TBSEND, and TBREAD and TBWRIT in GUKINE and GUOUT are replaced by them respectively. TBRECV and TBSEND are implemented with PVM subroutines. The parallelized version of our simulator has divided into two part, master and slave. Random number seeds are passed from the parent process to slaves at the initialization step. And inside master program, the TBREAD subroutine reads date from disks or tapes and send it to slave process using TBSEND. Slave process get a event record using TBRECV in GUKINE, then do a simulation of a event and return the result using TBSEND in GUOUT, and wait for next event data. This implementation is shown in Figure 3. The implementation itself was very easy and took only a week to do that.

PERFORMANCE

The performance is measured on the system of 4 Sparc10/41 stations connected with each other via FDDI running SUN OS 4.1.3. We generated 1000 $B \longrightarrow J/\psi\ K_s$ samples and simulated using both parallelized and not parallelized version of our detector simulator. Typical resulted event size is about 200KB. It took about 50 seconds/event for usual not parallelized version and about 52 seconds/event for new parallelized version. The overhead of PVM is estimated about 4 % under this condition. This value is small enough for our purpose.

Conclusion

We have implemented a parallelized version of our detector simulator successfully using PVM. The overhead of PVM is enough small for small number of processors. We can say PVM is enough usuful for a condition which have up to 10 or more processors in CHEP. However, we still need to further study for estimating latency of I/O when we use huge number of processors to use future realistic conditon. We are estimating that we need about 30000 MIPS computing power for event simulation at the KEK B factory experiment, and the computer system will consists of huge number of processors. If many processes try to send or receive data at the same time, processes must wait finishing other process's I/O and overhead may become very large. Because PVM does not have asynchronous I/O mechanism in the current version. PVM is still under-developing and we wish it will have asynchronous I/O mechanism in the future. Besides it, we will investigate other parallelization packages available which have asynchronous I/O mechanism.

REFERENCES

1. Al Geist et al., PVM3 user's guide and reference manual, ORNL/TM-12187
2. Y. Takaiwa, talk given in MC93 International Workshop
3. CERN CN, GEANT 315 refference manual
4. S. Kawaba et. al., TRISTAN BANK SYSTEM User Manual, KEK internal

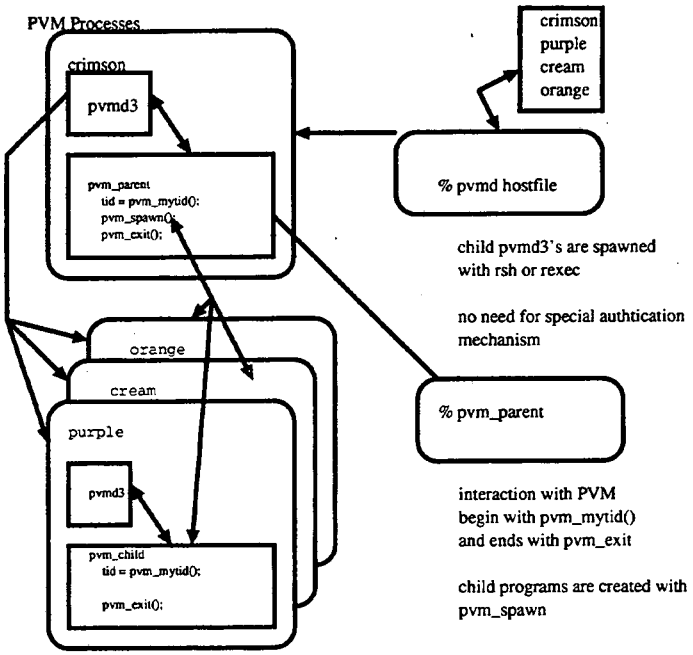
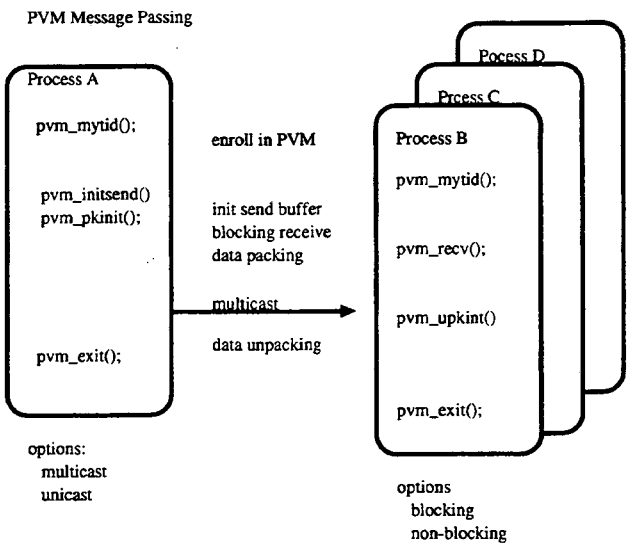


Figure 1



Fiugure 2

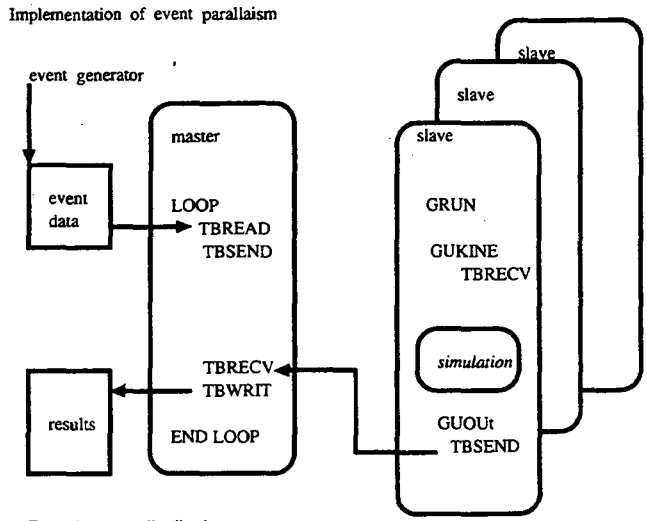

Implementation of event parallaism
event generator
event
data
master
LOOP
TBREAD
TBSEND
TBRECV
TBWRIT
END LOOP
results
slave
slave
slave
GRUN
GUKINE
TBRECV
simulation
GUOUt
TBSEND
Event-by-event distribution
Replace TBREAD/TBWRIT with TBRECV/TWSEND
in GUKINE and GUOUT routines
Event generator will be integrated with master program

Figure 3

FEASIBILITY STUDIES FOR A HIGH ENERGY PHYSICS MC PROGRAM ON MASSIVE PARALLEL PLATFORMS

L.M. Bertolotto, K.J. Peach
Department of Physics and Astronomy, University of Edinburgh, UK

J. Apostolakis, C.E. Bruschini, P. Calafiura, F. Gagliardi,
M. Metcalf, A. Norton, B. Panzer-Steindel
CERN, Geneva, Switzerland

Abstract

The parallelisation of a Monte Carlo program for the NA48 experiment is presented. As a first step, a *task farming* structure was realized. Based on this, a further step, making use of a distributed database for showers in the electro-magnetic calorimeter, was implemented. Further possibilities for using parallel processing for a quasi-real time calibration of the calorimeter are described.

INTRODUCTION

Although the use of parallel processing is well established in theoretical particle physics, there has so far been rather little use of such machines by the experimental particle physics community. Since the new generations of experiments will exceed today's computing potential by orders of magnitude, the investigation of high performance computing potentials becomes an issue of primary importance. In the following sections, the use of parallel processing techniques in the context of a particle physics experiment is described.

CP VIOLATION AND NA48

The observation of CP violating effects is still entirely confined to the neutral kaon system: the decay of a long-lived neutral kaon into two either charged or neutral pions with a branching ratio of about 10^{-3}, and a charge asymmetry in semi-leptonic decays. Such an effect is mostly due to the mixing of the two CP eigenstates K_1 ($CP = +1$) and K_2 ($CP = -1$) into the physical eigenstates K_L and K_S. The complex parameter ϵ describes this mixing. In addition, CP violating decays may be induced through virtual transitions involving heavy quarks. The relative magnitude of such transitions involving CP violation in $K^0 \to 2\pi^0$ decays is given by the complex parameter ϵ'. The aim of the NA48 experiment at the CERN-SPS [1] is to measure $Re(\frac{\epsilon'}{\epsilon})$ with an accuracy of 2×10^{-4}.

NA48 Experimental Method

The measurement of $Re(\frac{\epsilon'}{\epsilon})$ is obtained by the determination of the double branching ratio

$$R = \frac{K_L \to 2\pi^0}{K_S \to 2\pi^0} : \frac{K_L \to \pi^+\pi^-}{K_S \to \pi^+\pi^-} = 1 - 6Re(\frac{\epsilon'}{\epsilon})$$

with simultaneous observation of all the four decay modes in the same experimental setup. Cancellation of detection ef-

ficiencies as well as of effects introduced by accidental activity in the detector elements occur if the K_L and K_S beams coincide, and if the decays are evaluated independently for different kaon momenta and decay positions in the detector.

NA48 Experimental Setup

The basic feature of the NA48 scheme consists of two collinear beams of K_L and K_S respectively, which are produced concurrently and allow the record of charged and neutral decay modes at the same time and from the same fiducial length. A magnetic spectrometer with four drift chambers will reduce background in the charged pion mode, and a fast liquid krypton calorimeter is planned to reduce background in the neutral mode. Very good energy, space and time resolution are required for this detector. The properties of liquid krypton, in particular the short radiation length and high density, and the design of the calorimeter, with high granularity and a quasi-homogeneous structure, combine to satisfy the requirements.

Error Sources

The most important experimental problem is to collect sufficient statistics, especially for the $K_L \to 2\pi^0$ decay channel, which occurs with a branching ratio of only 9.09×10^{-4}. To achieve the desired accuracy on $Re(\frac{\epsilon'}{\epsilon})$, about 10^7 events for each decay channel are needed.
The dominant systematic error sources are possible differences in the reconstructed energy scales of $K \to 2\pi^0$ and $K \to \pi^+\pi^-$ decays, and the residual three-body decay backgrounds given by $K_L \to 3\pi^0$, $K_L \to \pi e \nu$ and $K_L \to \pi^+\pi^-\pi^0$, whose branching ratios are higher by more than two orders of magnitude than those of the CP violating decays.
In order to study these systematic effects and determine the acceptance for the experiment with sufficient accuracy, typically 10^8 Monte Carlo events per decay mode have to be generated in a single run.

Why Massively Parallel Processing ?

For experimental reasons, a fast feedback from the Monte Carlo is desirable. The initial target is to generate the statistics for a first evaluation of acceptances and tail effects (about 10^7 MC events) overnight. With a more massive system it should then be reduced to a few hours. On a single workstation, the necessary time would be much longer: for example, the average time needed to generate one event on an HP735 is about 70 msec (CPU time about 20 msec, and I/O to disk about 50 msec), so that it would take more than 8 days to generate 10^7 events. For this reason, the use of parallel processing has been considered in order to improve the performance of the simulation program.

The Monte Carlo Programs

The experiment uses two different and complementary MC programs. The first consists of a full and detailed GEANT simulation, called NASIM [2,3], which is considerably time consuming (for example, it needs about 10 sec to generate an average electro-magnetic shower in the calorimeter). For this reason, a second, dedicated and fast detector simulation code, called NMC [4], was developed by the NA48 collaboration - see fig. 1.

It does not perform a full event-by-event simulation of shower depositions in the liquid krypton calorimeter, making use instead of a so-called *shower library* generated by GEANT. Each entry is identified by the particle type, its energy, and the coordinates of its impact point on the surface of the electro-magnetic calorimeter. It contains the energy deposited by the particle in the cells of a predefined area centered on its impact point.
The size of a single shower depends on the particle type. In the final version of the library it will be about 1 kbyte for photons, electrons and positrons, and about 5 kbytes for charged pions. Typical events, as seen by the calorimeter, are shown in fig.2. The number of entries being 10^6 or more, the ultimate size of the shower library will be at least of the order of 5 Gbytes. Since the access time to this database has to be small, a large distributed memory is needed.

Parallelisation of the MC Program

First Step: Task Farming

The simplest approach is to consider *task farming*. In such an implementation the same program runs on different processing elements, called *workers*, generating different events at the same time. A *source* processor initializes the working environment of the message passing interface as well as of the main code itself. It then splits the total amount of events to be generated into "work packets" of predefined size, which are sent to the workers upon reception of a work request message. Each of the worker processors generates events running a slightly modified version of NMC. The results are then passed on to and collected by a so-called *sink* processor.
The differents tasks were interfaced using the message passing language CHIMP [5], developed at the Edinburgh Parallel Computer Centre.
Scalability for up to 14 processors has been observed for message passing overhead less than 2% [6].

Second Step: Task Farming with Shower Library

Encouraged by the results obtained with the simple task farm structure, we proceeded towards a further step in the parallelisation of NMC. The essential features of this approach are shown in fig. 3. An additional category of slave programs, the *shower workers*, was added to the scheme described above. Each of them is used to access a different part of the shower library. Whenever during the simulation of an event, information from the shower library is required, the worker examines an internal database to figure out which part of the library has to be addressed, and activates the corresponding shower worker. The simulation of the event continues after the requested information has been received.
Since the final version of the shower library is currently being generated, at the moment each shower worker reads a different copy of the same temporary version of the shower library, whose size is 16 Mbytes.

Some advantages of Task Farming

A special effort was made in order to keep the structure of the parallel code as flexible as possible. The Monte Carlo

program NMC is in an evolution phase, and new versions are released whenever needed. In order to be easy to update, the parallelisation has to consist of only minor modifications to the serial code, which should be "transparent" to the user. The message passing language we used, CHIMP, is particularly suited for this purpose.
A fundamental role in any simulation program is played by random numbers. In the implementation described above, the handling of random numbers (generated by the CERN library routine RANMAR) is reliable: it is assured that different processes do generate different events. This is one clear advantage with respect to running independent jobs on different workstations.

A PLAN FOR THE FUTURE

We are studying the possibility of making use of parallel processing techniques for the calibration of the liquid krypton calorimeter. Since the core of a typical electromagnetic shower covers several cells, the calibration of each of the 13000 cells is coupled to that of all its near neighbours. Thus, a 13000 x 13000 matrix must be "inverted" in principle to deconvolute the response. The aim of our work is to study whether a beam calibration could be reconstructed and analyzed in quasi-real time using the power of the MPP machines to both reconstruct the data and to accumulate the cell calibrations.
For this purpose, a large processing power and a large memory are needed. Since each of the processors will see directly only a part of the calorimeter, also a high bandwidth network fabric is required.

REFERENCES

1. G.D. Barr *et al.*: "Proposal for a Precision Measurement of (ϵ'/ϵ) in CP violating $K^0 \rightarrow 2\pi$ Decays", CERN/SPSC/90-22, SPSC/P253, July 1990.
2. R. Brun *et al.*: "GEANT User's Guide", CERN Program Library W5103.
3. M. de Beer *et al.*: "NASIM User's Guide", NA48 Internal Note.
4. F. Leber *et al.*: "NMC User's Guide", NA48 Internal Note.
5. J.G. Mills *et al.*: "CHIMP Concepts", EPCC Technical Report EPCC-KTP-CHIMP-CONC, 1991.
   R. Alasdair *et al.*: "CHIMP Version 2.0 User Guide", EPCC Techical Report EPCC-KTP-CHIMP-V2-USER, 1993.
6. B. Panzer-Steindel: "Test of Scalability for Task-Farming with CHIMP", CERN/CN-GPM Internal Note, August 1993.

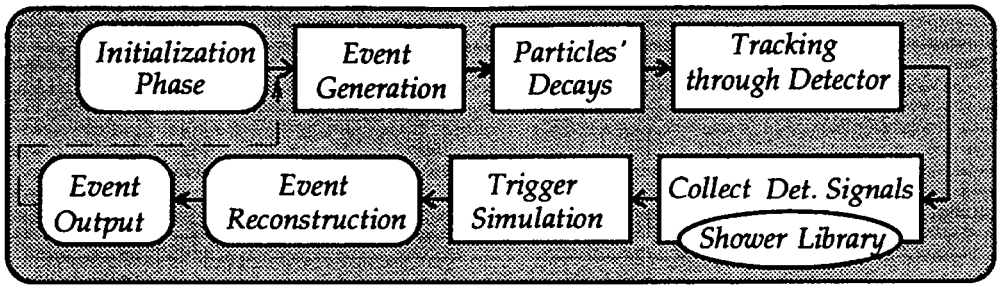


Figure 1: The flow chart of the NMC Monte Carlo.

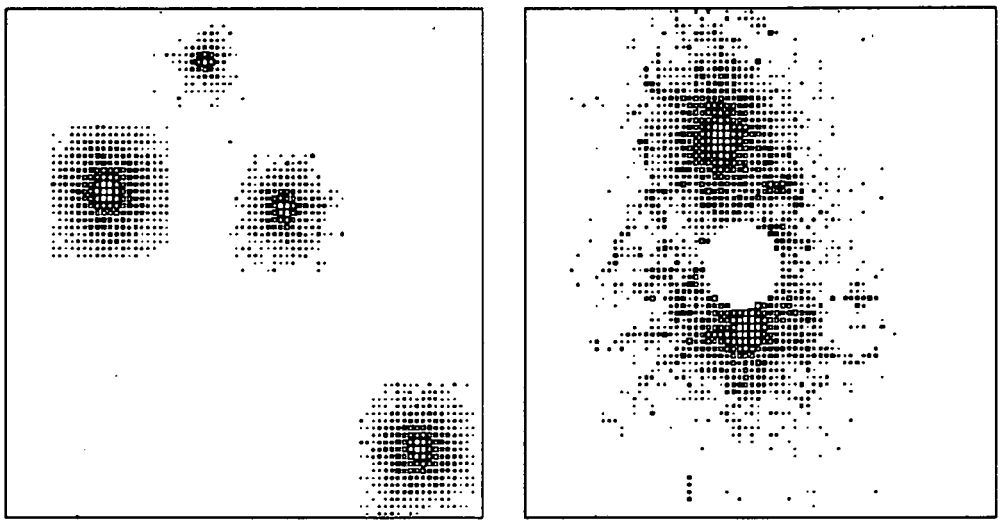


Figure 2: The appearance of typical events in the liquid krypton calorimeter. On the left is a $2\pi^0$ event, on the right a $\pi^+\pi^-$ event.

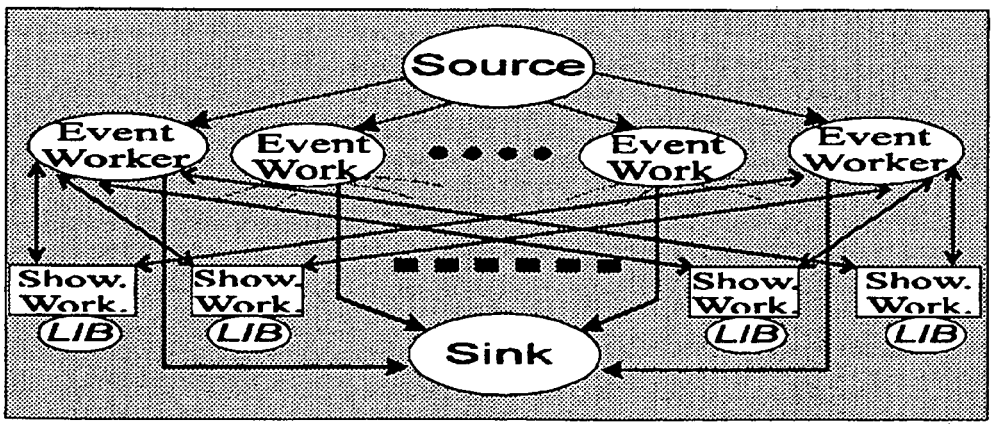


Figure 3: Parallel NMC with shower library.

Gaussian and Feed-Forward Neural Network Classifiers for Shower Recognition, Generalisation and Parallel Implementation.

W. Lourens, A.W. Lodder, H.M.A. Andree and A. Taal,
Department of Physics and Astronomy, Utrecht University,
P.O. Box 80000, 3508 TA Utrecht, The Netherlands.

Abstract

The performance of Gaussian and feed-forward neural network classifiers, is compared with respect to the recognition of energy deposition patterns in a calorimeter. Implementation aspects of these classifiers for a multi-processor architecture are discussed.

INTRODUCTION

Feed-forward neural networks can be successfully trained to recognise showers in a calorimeter [1]. If the distributions of the patterns from the different categories are assumed to be multivariate Gaussians, then the Bayesian classifier applies quadratic discriminant functions. If all covariance matrices of the Gaussian distributions are the same, both Gaussian and feed-forward neural network classifiers are able to perform the optimal classification. Implementation aspects for a multi-processor architecture are discussed.

GAUSSIAN CLASSIFIERS

Consider an indexed set of categories C_i, with $i \in I_0$ and $I_0 = \{1, 2, \ldots, M\}$. Let the class conditional density $p(\xi|C_i)$ of the patterns $\xi \in \mathcal{R}^d$ of a category C_i be a multivariate Gaussian

$$\frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\xi-\mu_i)^T\Sigma_i^{-1}(\xi-\mu_i)} \qquad (1)$$

with μ_i the mean and Σ_i the covariance matrix. Both μ_i and Σ_i can be estimated using a data base of patterns. For the data base presented in [1] the generalisation capability of both classifiers is shown in figure 1. In the case of a symmetric loss function the optimal Gaussian classifier [2], a Bayesian classifier, applies the discriminant functions:

$$g_i(\xi) = \log p(\xi|C_i) + \log p(C_i) \qquad (2)$$

with $p(C_i)$ the priori probability of category C_i. For an optimal classification a pattern ξ is assigned to category C_i if

$$g_i(\xi) = \max_{j \in I_0}\{g_j(\xi)\} \qquad (3)$$

Expression (2) reduces to the general quadratic form

$$\begin{aligned} g_i(\xi) &= \xi^T A_i \xi + \xi^T \mathrm{b}_i + c_i \qquad (4) \\ A_i &= \Sigma_i^{-1} \quad , \quad \mathrm{b}_i = -2\Sigma_i^{-1}\mu_i \\ c_i &= \mu_i^T \Sigma_i^{-1} \mu_i \\ &+ \log p(C_i) - \frac{1}{2}\log|\Sigma_i| - \frac{d}{2}\log 2\pi \end{aligned}$$

A set of indexed functions $G_{I_1} = \{g_{ij}\}$, with $i, j \in I_0$ and $i < j$, is defined on the domain $D = \mathcal{R}^d$, with

$$g_{ij}(\xi) = g_i(\xi) - g_j(\xi) \qquad (5)$$

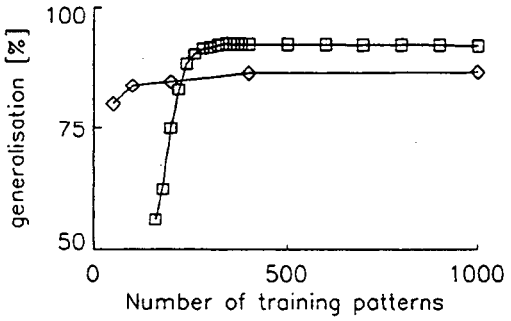


Figure 1. Mean generalisation, averaged over 20 sets, of a Gaussian (□) versus a feed-forward neural network (◇) classifier as a function of the trainingset size.

The Voronoi diagram, $VoD(G_{I_1})$ partitions the input space by a paraboloid arrangement G_{I_1}, with $g_{ij} = 0$ the decision surfaces. Define the set $h_{I_0} = \{h_i\}$ of affine functions with

$$\begin{aligned} h_i(\xi) &= \xi^T \mathbf{w}_i + w_{0i}, \\ \mathbf{w}_i &= b_i \quad , \quad w_{0i} = c_i. \end{aligned} \tag{6}$$

The set $H_{I_1} = \{h_{ij}\}$ is defined in analogy with the set G_{I_1}. In case all covariance matrices Σ of the functions $g_{I_0} = \{g_i\}$ (eq. 4) equal, $VoD(G_{I_1}) = VoD(H_{I_1})$, i.e. both Voronoi diagrams incorporate the same cell structure.

Consider a feed-forward neural network, build of linear threshold units, that is trained on patterns with probability densities given by eq. 1. The optimal set of paraboloids G_{I_1} is approximated by the set H_{I_2} of hyperplanes represented by the partition layer. If for both sets only non redundant decision surfaces are considered, then in general $|H_{I_2}| > |G_{I_1}|$.

IMPLEMENTATION

It is assumed that the first-level trigger provides RoI's (Regions of Interest) in a global memory. Each RoI is a block of zero suppressed data of at most 32×32 (d_{cal}) data words. Each word consists of a calorimeter address with the corresponding deposited energy. To each RoI a processor node of a mesh of second-level trigger processors is assigned.

Only the calculation of the term $\xi^T A_i \xi$ is considered. The time required by a single processor node to label a shower is

$$\begin{aligned} T^{(1)} &= T_{g\to l} + T_{Pr} + T_{l\to g} \\ T_{g\to l} &\leq d_{cal} T_1 + K_1 \\ T_{Pr} &= T_{Pr,1} + T_{Pr,2} \\ T_{Pr,1} &\leq dT_2 + d_{cal}(T_3 + T_4 + T_5 + T_2) \\ T_{Pr,2} &= M\Big(d(d(2T_2 + T_3) + (T_2 + T_3)) + T_5\Big) \\ T_{l\to g} &= T_1 + K_1 \end{aligned} \tag{7}$$

$T_{g\to l}$ time to copy the RoI data from global memory to local memory,

T_1 bus cycle time; if there are R RoI's in global memory and a synchronous bus, then the worst case is $R \cdot T_1$,

K_1 fixed overhead cost due to address calculation and bus access,

T_2 time of a Store or Load operation,

T_3 time of a Multiply-Accumulate cycle,

T_4 time of an Add operation,

T_5 time of a Compare operation,

$T_{Pr,1}$ time to map the RoI on a vector ξ, such that the centre element of ξ coincides with the absolute maximum. Initialise vector ξ (dT_2), for each energy value in a RoI calculate its address ($T_3 + T_4$), compare energy value with the largest value so far (T_5), and store energy value (T_2).

$T_{Pr,2}$ time to produce the value $v_i = \xi^T A_i \xi$, following a compare with the largest value so far (eq. 3),

$T_{l\to g}$ time to copy the shower label ($i \in I_0$) from local to global memory.

Consider a processor node with $N+1$ processors connected to a single input bus and a single output bus, e.g.[3]. If

$M = kN + r$ with $0 \leq r < N$, each of N processors receives k complete matrices A_i. The remaining r matrices are divided over N processors such that, if $rd = k' \cdot N + r'$ with $0 \leq r' < N$ then r' processors receive $\lceil rd/N \rceil$ contiguous rows, and the remaining processors each receives $\lfloor rd/N \rfloor$ contiguous rows. The total time $T^{(N+1)}$ to label a shower is

$$\begin{aligned} T^{(N+1)} &= T_{g\to l} + T'_{Pr} + T_{l\to g} \qquad (8) \\ T'_{Pr} &= T_{Pr,1} + T'_{Pr,2} + T_{Pr,3} \\ T'_{Pr,2} &= k\Big(d(d(2T_2 + T_3) + (T_2 + T_3)) + T_5\Big) \\ &+ \Big\lceil \frac{rd}{N} \Big\rceil (d(2T_2 + T_3) + (T_2 + T_3)) \\ T_{Pr,3} &= N(6T_1 + K_1) + 2NT_4 + (N + r)T_5 \end{aligned}$$

$T_{Pr,3}$ time spend by the $(N+1)$th processor to collect and process the data from the N processors. Each processor sends at most 6 words: the largest value v_i with its corresponding label i, and at most two partial values with labels. The $(N+1)$th processor composes r-values ($2NT_4$), and determine the shower label ($(N + r)T_5$).

If K_1 and all T_i are taken equal, the speedup becomes (see also figure 2)

$$\begin{aligned} SP &= \frac{T^{(1)}}{T^{(N+1)}} \qquad (9) \\ T^{(1)} &= M(3d^2 + 2d + 1) + d + 5123 \\ T^{(N+1)} &= \lfloor \frac{M}{N} \rfloor (3d^2 + 2d + 1) \\ &+ \Big\lceil \Big(\frac{M}{N} - \lfloor \frac{M}{N} \rfloor\Big) d \Big\rceil (3d + 2) \\ &+ M - \lfloor \frac{M}{N} \rfloor + 10N + d + 5123 \end{aligned}$$

In the case of a feed-forward neural network the complexity of the calculation to be performed and the possible speedup for multi-processor architectures satisfy the above calculations. As each neuron in the partition layer can be represented by a d-dimensional row in a matrix (for a partition layer of n neurons this yields an $n \times d$ matrix).

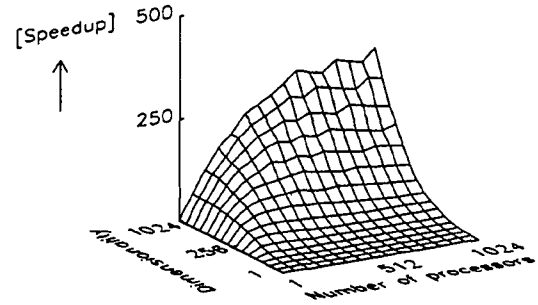


Figure 2. The speedup as a function of the number of processors and the dimensionality d of the pattern vector ξ (M=3).

For the minimum number of categories ($M = 2$), according to eq. 5 only one matrix-vector calculation has to be performed (this corresponds to $M = 1$ in the speedup calculation)

CONCLUSION

Interesting special purpose hardware is available, that can be used efficiently to parallelise Gaussian and feed-forward neural network classifiers. Considerable speedup can be obtained provided a complex classification problem is dealt with.

REFERENCES

1. H.M.A. Andree, W. Lourens, A.Taal and J.C. Vermeulen, Proc. 3rd Int. Workshop on Soft. Eng, AI and Expert Syst. for High Energy and Nuc. Phys., Oberammergau, Germany, Oct. 4-8, 1993, pp. 315–320.
2. N.J. Nilson, *The mathematical foundation of learning machines.* Morgan Kaufmann Publishers, 1990.
3. Adaptive Solutions: D. Hammerstrom, Proc. of 1990 IJCNN, II–537.

GEANT STEPS INTO THE FUTURE

R.Brun, O.Couet, A.Dellacqua, N.Cremel, S.Giani,
G.Kozlovsky, A.Nathaniel, F.Rademakers, S.Ravndal
CERN
Geneva, CH

Abstract

The algorithms and software developed to produce a computer generated movie on HEP simulations are explained. The GEANT program is used to design and visualize the detectors and to simulate LHC events. The PAW program is used to store such events and to produce a sequence of frames for animation. The geometrical modeling and the new tracking system of GEANT 3.21 will be explained in more detail, together with the ray-tracing technique developed for visualization purposes. The PAW ntuples then allow the temporal sorting of the particles positions and the storing of the GEANT 3.21 images and kinematics. Finally, how to produce a time sequence showing the real development of a simulated event will be analysed.

Introduction

The purpose of this paper is to describe the software technology and the algorithms developed at CERN to produce the film "GEANT steps into the future". Essential improvements of two CERNLIB major packages, GEANT and PAW, are presented and explained.

The GEANT Tracking

The tracking of particles through the geometrical data structure is the key functionality of GEANT. At every particle's step, the program must find the volume where the particle is located and the next boundary it will cross. This can take about 60% of the total simulation time (even for detectors described in an optimized way); therefore, a new logic has been introduced to minimize the time spent for the search in the geometrical tree. Instead of a linear or binary search (time spent proportional or logarithmic with the number of volumes, respectively), a 'direct access' technique has been developed to make the time basically independent of the number of volumes. Every volume containing other volumes is 'virtually' divided into equal slices at initialization time (the best direction is computed automatically); for each slice, a data structure is filled with a list of the volumes IDs intersecting such slice; slices with identical lists are pointing to the same contents and are merged. At tracking time, it is straightforward to find in which slice the particle is and only

its contents have to be checked. The same is true about finding the next boundary to be crossed: only if the intersection point with a content lies outside the current collection of slices, the next one will be checked. The new algorithm gives in average about a factor 2 in speed for the overall simulation in the LHC and LEP detectors. It also allows a fast tracking even in geometrical structures received from CAD systems. The initialization time for a detector like ATLAS (11 millions of volumes) is less than 1 sec on HP735 and the specialized data structure for the tracking is about 50000 words. As a result we have a very powerful tool to navigate in extremely complex geometrical structures (corresponding to huge geometrical data bases).

Detector Visualization

Such a tool has been found to be essential in the visualization of the experimental set-ups: based on this new GEANT tracking, a set of routines doing light processing has been written explicitly to visualize the detectors (useful also to visualize the results of boolean operations). Visible light particles are tracked throughout the detector until they hit the surface of a volume declared not transparent; then, the intersection point is transformed to the screen coordinates and the corresponding pixel is drawn with a computed hue and luminosity. Automatically, the color is assigned according to the tracking medium of each volume and the volumes with a density lower/equal than the air are considered transparent; alternatively, the user can set color and visibility for the desired volumes. Parallel view and perspective view are possible and the detector can be cut by three planes orthogonal to the x,y,z axis or/and by a cylindrical surface. Different light processing can be performed for different materials (from matt plastic to metal to glass). Parallel light or point-like-source can be selected and an extra light source can be positioned in space with the desired intensity. Realistic renderings have been produced for detectors made of a few thousands objects as well as for detectors made of several millions of objects. Each frame produced for the film (half million of pixels) takes in average less than 2 minutes on HP735. A parallel version of the program, giving an almost linear speedup with the number of processors, is also available.

Visualization of Events

A new way of visualizing tracks and events has also been developed. Instead of drawing the particles trajectories one after another, all the particles (represented by points or circles, the size depending on the energy) are plotted at the same time in the space position they occupy at a given instant. So the visualization is driven by time steps of 20 pico-seconds. This technique enhances the understanding of the physical processes and also helps to visualize the behaviour in time of hadronics and electromagnetic particles. This was made possible by saving in a RWN the position and time of flight of all the particles at every GEANT step during the simulation. After that, they must be sorted in time. This happens in 4 phases: an array is filled with the numbers of particles existing at each time sequence of 20 pico-seconds; memory-resident RWNs are created with a buffer size proportional to the numbers of particles for each time se-

quence; they are filled via HFN, which implicitly sorts them in time; finally, a CWN is created and one row is filled for each time sequence (each row contains a variable number of particles).

Interactive Visualization and Analysis

A large portion of the film has been dedicated to the GUI interface of Geant++ and Paw++. An extensive set of interactive facilities has been demonstrated. One of the new features is an interactive tool for fitting histograms. Fitting histograms is a difficult computation problem. Often, the underlying minimization problem has numerous local minima and, therefore, a minimization program can miss the global minimum. In many cases a user who has understanding of the problem can help Minuit to find the right solution. We developed an interactive user interface to Minuit which allows the user to change the fit function parameters using Motif scale widgets and to call Minuit minimization and error estimation routines. The changes in the shape of the fit function are immediately reflected in the graphics window of PAW++.

The Video Production

The Integrated Video system (by Integrated Research) is used for the sequence generation and display. These sequences can be stored on disk in various formats. For this film we have used "movie" files, which allow long sequences (45 s), at the expense of the quality of the pictures. This format has been used for the showers development sequences. For high quality pictures (short sequences of 10s maximum), "rgb" files are used instead. This format has been used for ray-traced detector sequences (a total of about 800 frames have been produced). Besides the sequence generation, the program "ivideo" has been used for the mixing of pictures (detectors in the background and the shower on top), and for the generation of smooth transitions between sequences. All the titles have been produced with the system "Ntitle" (from XAOS Tools). The hardware used to produce the film is a SGI Indigo-2 machine with a 24 bit graphics display, 160Mb of memory and 2Gb of disk space. A "Galileo Box" has been used to generate a video signal (PAL, NTSC) from the full screen or from a portion of the screen (PAL or NTSC format). The "Galileo box" was driven with the SGI software "videoout".

Conclusions

The GEANT 3.21 algorithms allows extremely fast tracking in very complex geometrical structures: a new tool is provided to visualize detectors made of millions of objects. A new way of showing simulated events is presented: time-steps are driving the visualization allowing a better understanding of the physical processes. New interactive facilities for data visualization and analysis are provided.

REFERENCES

1. GEANT User Guide, CN/ASD, CERN, 1994.
2. PAW User Guide, CN/ASD, CERN, 1994.

Session 8

Information Systems and Multimedia

DESIGN AND OPERATION OF THE HIGH ENERGY PHYSICS INFORMATION SERVER[1]

Jeff J. Dingbaum, David E. Martin
Fermi National Accelerator Laboratory
Batavia, IL 60510

Abstract

HEPIC an information "center of centers" for the HEP community, is a 24 hour on-line location where a HEP researcher can start her/his search for information. Operated by the HEP Network Resource Center, HEPIC is accessible via WWW, gopher, anonymous FTP, DECnet, and AFS. This paper describes HEPIC's design and future plans, and the HEPNRC's efforts to collect information and link high energy physics researchers world-wide.

1. This work is supported by the U.S. Department of Energy under contract No. DE-AC02-76CHO3000.

INTRODUCTION

The High Energy Physics Network Resource Center (HEPNRC), formerly National HEPnet Management, has provided higher layer network services to the physics community for about three years. Anonymous ftp and DECnet were offered first, followed by AFS, gopher and world wide web (WWW). These services are offered by an information server known as HEPIC, the high energy physics information center that is available via the Internet and the HEP-NSI DECnet.

By supplying timely and useful information to high energy physics researchers, HEPIC has become a first place to look for public, on-line HEP information, with over 285 MB of files stored directly on HEPIC and links to over 40 other HEP information servers.

HEPIC provides several different levels of access. The first level, file access, allows a remote user can browse the HEPIC file system and retrieve files. File access is provided via anonymous FTP, DECnet DIR and COPY commands, and AFS. Descriptive file names and readme files in every directory help the user look for information. Although HEP researchers use a wide variety of computers in their work, almost all machines have the necessary software already available to access HEPIC.

The next level, gopher access, makes use of the gopher protocol (developed at the University of Minnesota) to supply the remote user with a series of menus to navigate information available on HEPIC. In addition, further capabilities of the gopher protocol are used to provide links to other HEP gopher and FTP information servers located throughout the global Internet.

The highest level, hypertext access, uses the world-wide web (WWW) protocol (developed at CERN) to supply a series of hypertext "pages" that contains links to information stored on HEPIC and other HEP information anywhere on the Internet. In addition, these pages incorporate in-line

graphics, as well as links to disparate file types such as audio files and images.

Both gopher and WWW hypertext access require clients be installed on the users desktop machine. Fortunately, such clients are free and readily available for all major platforms with both text and graphical user interfaces.

Some samples of information stored on HEPIC are newsletter archives (CERN Computing Newsletter in 8.5" x 11" format, HEPnet News, FermiNews, What's New, FYI, Physics News Updates.), all sci.physics.accelerators and sci.physics.particles traffic, MBONE (packet videoconferencing) software, HEPNRC reports, room-based videoconferencing documents and archives of numerous HEP mailing lists. Much information is stored in flat ASCII and Postscript, but hypertext markup language usage is increasing due to its ability to tie-in related documents or sections.

New information is added daily and HEPNRC personnel are always on the look-out for new sources of information. In fact, some sessions from CHEP '94 will be recorded, digitized and stored on HEPIC.

DESIGN OVERVIEW

HEPNRC started offering anonymous ftp and DECnet services on a DEC MicroVax II that ran VMS and Multinet. This model was used for about a year and a half before it became apparent that more services would be added and a better platform would be required. A spare Sun Microsystems ELC was pressed into service to run anonymous ftp and the wide area filesystem AFS. The ftp area was setup so that the files actually reside in the AFS filespace, thus incorporating the two and eliminating the need to duplicate files.

The anonymous ftp files were NFS exported to the MicroVax to maintain DECnet access. DIR, COPY, and SET DEF commands work as though the files were actually served by a vax. The only problem with accessing commands via DECnet is when there are both upper and lowercase letters in the filenames. Multinet does character substitution in this case, leaving somewhat cryptic filenames. With this in mind, HEPNRC avoids the use of mixed case filenames wherever possible.

After ftp and AFS were successfully merged, higher level services were incorporated. Gopher and World Wide Web were added onto the machine, although initially, all data files pointed to actually resided in the anonymous ftp area. The WWW home page pointed to the gopher area, whose menus pointed to ftp data. Although the configuration files and executables for gopher and WWW resided in the SunOS filesystem, all other files sat in the AFS filespace.

Integration Problems

The integration of services with AFS was not without minor setbacks. One problem arose because we chose to build anonymous ftp in the AFS filespace. Since we are using the ftp daemon from Washington University, St. Louis, ftp users can use automatic compress/uncompress and tar on the fly. That is, a user can do a "get" on a compressed file, but if he does not have a program to uncompress it on his machine he can specify a a filename without the .Z extension. The ftp daemon will decompress it as it is being transferred. If the file is not compressed the user can specify a .Z extension and the daemon will compress it before it is sent out. Tar works similarly in allowing you to tar files together before transfer (filename.tar) with the option to tar and compress with one command (filename.tar.Z).

When attempting to use the compression utility, however, it was discovered that a special file called /dev/zero was required to zero out memory. The anonymous ftp account does a chroot to change root directory file access for all commands for an ftp session. Thus the file /dev/zero was not there relative to ~ftp root.

For example, the ftp directory on HEPIC is really located at /afs/hepafs1.hep.net/public/anon_ftp. When an anonymous ftp user logs in he or she gets an ftp root directory that is located there. Thus, when the file /dev/zero is needed in the ftp session it is expected to be found at /afs/hepafs1.hep.net/public/anon_ftp/dev/zero, or rather, ~ftp/dev/zero.

A command called mknod is normally available when making files such as ~ftp/dev/zero. Since this file was to exist in AFS filespace it would require a mknod command as part of the AFS filesystem distribution. Unfortunately, that AFS command does not exist, since it would not make sense to support special files in a global filespace.

A workaround to this problem is to statically link the libraries when compiling the compression utility, as opposed to the default dynamic linking. Thus, file ~ftp/dev/zero is not required.

SERVER OF SERVERS

Data Types

At present, there are two types of documents on the server, dead end and hypertext. Dead end documents can be thought of as a dead end street - there's a house there to visit, but you can't go anywhere else. Documents can be thought of in much the same way; once you get to your document you may as well read it because you can't go anywhere else (except back the way you came of course). Ftp, gopher and AFS files resemble this model, you can view a document, but it takes you nowhere else (electronically).

On the other hand, a WWW hypertext document is exactly as the name implies - part of a web. A document you are reading on HEPIC contains a link to a document on a server at CERN, which in turn contains a link to a document at SLAC, which has a link that points back to the original document you started reading on HEPIC.

This method of reading documents, although convenient for the reader, requires more work for the person maintaining it on the server. If a paper is a dead end document it can be placed into the filesystem and left there. Before a hypertext document can be placed into the filesystem it must be converted into html format. Depending on length, this could take anywhere from five minutes to several hours. In short, marking a paper for WWW involves putting in tags for formatting such as headers, paragraphs, lists, character bolding and italics, etc., as well as inserting links to other documents.

Online Data

HEPNRC maintains many different types of information on its server, as indicated earlier. In addition to that, CERN has given HEPNRC permission to mirror the CERNlib writeups in html form. Because these documents previously resided only at CERN, response times were often slow because of high traffic on the transatlantic link. Users in the U.S. looking for CERNlib writeups should see an improvement in speed by going to HEPIC.

Link Maintenance

The HEPIC information server maintains links to ftp, gopher and WWW servers

world-wide. The vast amount of information is growing, as new WWW servers are being brought up not just at the company level, but at the division and even workgroup level. People responsible for writing and updating papers can in this way keep them current and have them readily accessible. HEPNRC searches out links to new sources of HEP information to incorporate into HEPIC. In that way a HEP user will be able to go to HEPIC to find the data needed, whether it is online there or a pointer to another WWW site.

FUTURE CONSIDERATIONS

Before the introduction of gopher and WWW on HEPIC the Sun ELC was adequate to handle the load placed on it. With the addition of these services, however, more and more users are now placing greater demands on it. Because of these demands, a new server is being purchased to replace the existing ELC.

New Services

The new server will be installed as the primary fileserver for all network services now offered, as well as for new ones to be added. Search capabilities will be added to allow searching of both information stored on HEPIC and that stored on other servers. Search protocols such as the wide-area information service (WAIS) will be fully supported. Other methods such as archie, veronica and new WWW mappers will be looked at also.

Redundant Server?

As mentioned earlier, all data files for the information server are not tied to a machine *per se,* but actually reside in AFS. When the new fileserver is installed it will handle all of the ftp, gopher, WWW, and decnet through nfs traffic, as well as that from AFS. The old system will be used in an experiment as an AFS file and database server and backup server for the other services. Because data replication is possible in AFS, if a fileserver is down the data is still available via AFS. This does not help someone using one of the other services such as WWW because the program on that machine that serves the data would not be running.

An experiment that HEPNRC is working on will allow AFS machines to run other file services such as WWW, gopher and ftp from two different machines. While this is not new, both machines would be pointing to the same data in AFS space and both would be registered in the local Domain Name Server (DNS) database as the same machine name, but have different IP numbers.

If the client requesting a DNS lookup does so via a gethostbyname DNS will return a list of IP numbers in succession. If the first address fails it will supply a second number. In this way you could have two machines running as www.hep.net, both running WWW, gopher, and ftp clients. If one machine were down for any reason the client would go to the second one for the data.

CONCLUSIONS

HEPNRC has seen an exponential increase in the number of documents accessed from HEPIC. From several hundred per month just a few years ago to over 15,000 per month now. Clearly the users have spoken and the message is that they use gopher and world wide web.

HEPIC is available via anonymous FTP (ftp.hep.net), WWW (URL: http://www.hep.net), gopher(gopher.hep.net), DECnet (HEPNET::), and afs (/afs/hepafs1.hep.net/public/anon_ftp).

Videoconferencing using workstations in the ATLAS collaboration

C. Onions (CERN/ECP, Geneva, Switzerland), R Blokzijl, K. Bos (NIKHEF, Amsterdam)

The ATLAS collaboration consists of about 1000 physicists from close to 100 institutes around the world. This number is expected to grow over the coming years. We realised that we needed to do something to allow people to participate in meetings held at CERN without having to travel and hence we started a pilot project in July, 1993 to look into this. Our colleagues from Nikhef already had experience of international network meetings (e.g. RIPE) using standard UNIX workstations and public domain software tools using the MBONE, hence we investigated this as a first priority.

1. Introduction

The initial goals of the pilot project were to broadcast a weekly software meeting to outside institutes. As the internal CERN network did not support multicasting, the first trials were done with a point-to-point connection between CERN and Nikhef in Amsterdam, with video transmission in one direction only, primarily to show slides projected on an overhead projector. This was a low priority project

2. Conference room equipment

Hardware:

- SUN Sparc 10 Model 30LC
- Videopix card
- External amplifier and 4 speakers
- Microphones (6)
- Sony CCD G100 colour video camera
- Ethernet connection

Software:

- sd [1] for session control (FIGURE 1.)
- mmcc [3] as an alternative to sd
- vat [1] for the audio (FIGURE 2.)
- nv [2] for the video
- wb [1] for the whiteboard

This software is available in the public domain. The network bandwidths used for the different transmissions are:

- vat 16 kbps to 64 kbps
- nv 120 kbps for 5 frames/sec.
- wb anything goes

3. First tests

3.1. Audio

We started in August 1993, concentrating initially on audio, since without good audio you can not have a conference. Unacceptable delays of 3 to 4 seconds in the sound transmission were found. On investigation, it was found that playout times from Nikhef to CERN were of the order of 180 ms.,compared to around 3 seconds in the other direction. This was mainly due to packet reordering caused by routers in between the conference room and Nikhef, forcing *vat* to increase its internal buffer sizes in an attempt to save as many packets as possible. The effect was long playout times. In addition, PCs on the network were having problems coping with the traffic. As a result, a special ethernet segment was put in to the conference room with a direct connection to the FDDI backbone and from there to the outside world. The playout times were then the same in both directions, of the order of a few hundred milliseconds, and excellent sound quality was obtained. Connecting up the external amplifier and speakers proved to

be a problem - the signal to noise ratio was effectively zero. We were unable to find technical specifications for the SUN external speaker box and eventually the engineers in Nikhef had to take a box into the lab to work out what filters were needed between it and the amplifier. They set up a complete system in the lab and then brought it to CERN and installed it. One hour after installation, someone broke into the conference room and stole the speakers! This delayed us by one month.

3.2. Video

Video tests were rather disappointing. When viewing the full image projected by the overhead projector even reasonably good overheads were unreadable. Only on zooming to about a third of the overhead were things acceptable. This meant that the first idea to have a fixed video camera needed to be revised - we would at least need a remote control camera with pan/tilt and someone to operate it if we followed that line. It was noticed that the network traffic was non-negligible even if the image was fixed - at first we thought it might be the autofocus continually adjusting, but in fact it was found to be movement of the suspended screen due to air currents. The solution was to use a rigidly fixed screen.

3.3. White board

The whiteboard was used for communication mainly during the time we were setting up the audio.

4. Current status

We have just installed a Parallax [4] card instead of the Videopix one and get very good video resolution. This card is 24 bit (as opposed to 8 bit) video but is much more expensive (a factor 5) and requires the use of a commercial software package Uniflix [5]. Using this card it was possible to read projected transparencies without zooming the camera. A conference was set up between CERN and Fermi lab to discuss the R&D proposal P55 "Object Oriented Approach for Software Development for LHC experiments". *mmcc* was used to set up a point-to-point connection. As Uniflix was not installed at that time in Fermilab, *nv* was used to receive and send video An additional TV monitor was used to display the video from Fermilab. The session lasted two hours. The sound quality was good and the video image was reasonable. Machine prepared transparencies with large font sizes and dark colours were fine, some hand written ones were not legible.

A software meeting was also held in which a talk was given by someone sitting in Nikhef. The transparencies were prepared in advance and made available on the World Wide Web. They were printed off just before the meeting and put onto the overhead projector when requested by someone in the room.

5. Next steps

We have since started trials using *sd* and IP multicasting with people in RAL, England and Nikhef. We have not yet managed to achieve good audio performance, probably due to our lack of experience with the tools. We wish to try out conferencing with Uniflix software installed at the receiving end to benefit from the 24 bit video. In addition, we want to try out sessions across different platforms. We aim to have regular meetings with our American and Japanese colleagues on the P55 proposal. Having realised that it is important to see the people in the room as well as the projected slides, we will add a second camera.

Tests will also be done with transparencies prepared in advance in machine readable form and displayed using an LCD overhead projector.

We hope to be able to set up a production system in the next few months when we have optimised the setup.

6. Acknowledgments

Many thanks are due to the authors of the various public domain packages listed below. Without their excellent work none of this would have been possible. In addition, we would like to thank Ad Berkien for the selection, testing and installation of the audio equipment in the conference room; Bernard Duthion for the other installation work; and Christian Isnard and Francois Fluckiger, who did much of the work on the networking side and also installed the Parallax card and Uniflix software.

7. References

[1] *vat* (Visual Audio tool), *wb* (Whiteboard), and *sd* (Session Directory) were written by Steve McCanne and Van Jacobson at LBL and are available via ftp at ftp.ee.lbl.gov in directories conferencing/vat, /wb and /sd

[2] *nv* (Network Video) by Ron Frederick at Xerox Palo Alto Research Center is available via ftp at parcftp.xerox.com in directory pub/net-research

[3] *mmcc* is an alternate session control written by Eve M. Schooler of USC/ Information Sciences Institute, Marina del Rey, CA and available via ftp at ftp.isi.edu, directory confctrl/mmcc

[4] from Parallax Inc., USA

[5] from Paradise Inc., USA

FIGURE 1. sd main window

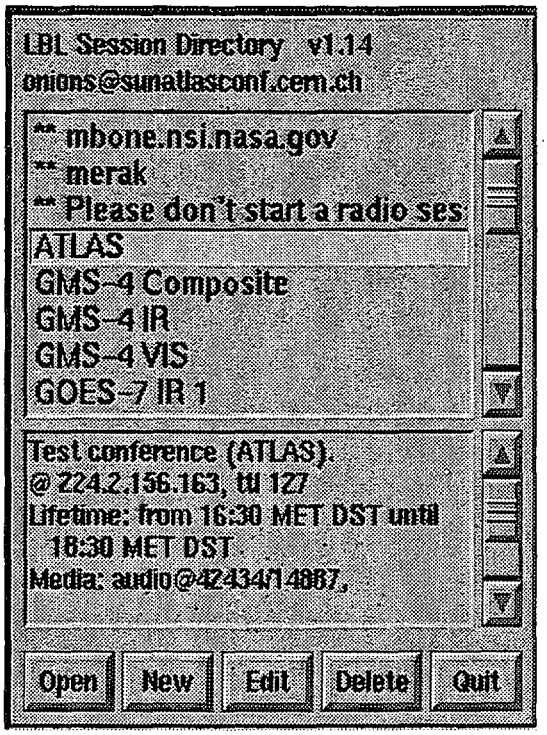
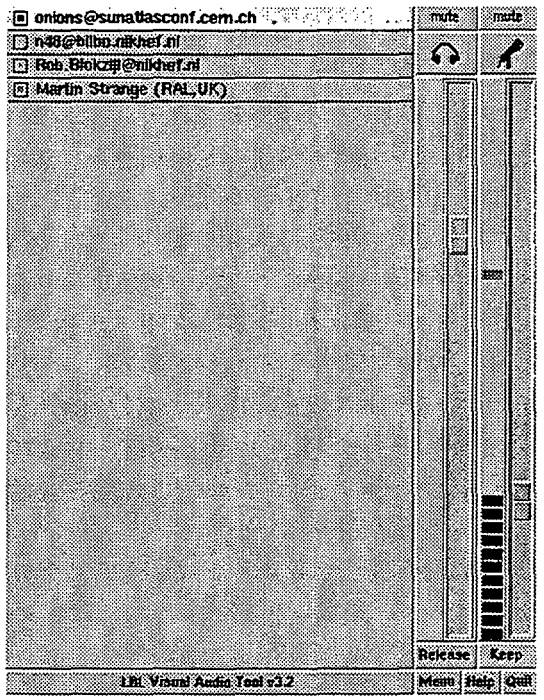


FIGURE 2. vat main window



The use of the World-Wide Web in HEP

R. Cailliau, M. Dönszelmann, S. Fernández Vega, A. Luotonen *

Abstract

All the major HEP sites and experiments have adopted WWW as either the backbone information system or as a supplement to their existing systems. The state of usage of WWW in the HEP community in Europe and the U.S. is reviewed and the state of the software, the system and the standards are presented. Plans for the future and possible new applications are outlined.

Brief History of WWW

In 1990 Tim Berners-Lee wrote the NeXTStep browser-editor and the first http server. This combination opened the road to true distributed hypertext systems on the Internet network.
In 1991 followed the Line-Mode Browser (LMB), a viewer which uses character-grid presentation of HTML documents. This browser was written in basic C (not ANSI C), compiled everywhere and made the WWW really popular.

Format negotiation, defined in the initial design, was implemented in 1992, making it possible to transmit other than HTML documents, provided the client site has appropriate viewers. This ensured that WWW would not remain in the closet as a system with a proprietary format.

NCSA introduced the famous X-Mosaic browser, which came as a single executable file and the promise to have implementations on the Macintosh and PC. Mosaic soon dominated over other X viewers. Its impact on the spread of WWW is even superior to that of the LMB, because its capability to display in-line colour images (gif, xbm) gave WWW the attractive look it needed to convince the uninitiated. That it was more successful than purely HEP developments probably derives from the better visibility of NCSA inside the computing community.

Both the Internet itself and WWW in particular have grown spectacularly during the last half of 1993 and no sign of slow-down is visible so far in 1994.

The expansion can truly be compared to that of a shock-wave. We can only speculate as to what 1994 will bring.

The number of specific clients on different or difficult platforms, such as X-Window, Mac and PC has exploded. Many have been built on top of existing toolboxes, although the most successful ones have been those which were written in native code, close to the operating systems. In parallel, NCSA has implemented a coherent set of clients for X, Mac and PC under the name "Mosaic", which are now almost a de-facto standard, to the point that WWW and Mosaic are being confused.

With minimal manpower, CERN has implemented a number of services in the

* Robert Cailliau is head of CSE (Communication Systems for Experiments) section in ECP Division's Programming Techniques Group, cailliau@www.cern.ch; Mark Dönszelmann is a Fellow in ECP DS group, working with the Delphi LEP experiment duns@vxdeop.cern.ch; Susana Fernández Vega is a Fellow in CSE section, susana@ptsun00.cern.ch; Ari Luotonen is a Technical Student in CSE section, luotonen@ptsun00.cern.ch.

http server that were needed by the physics community:

- access authorisation,
- automatic searches and indexing,
- execution of scripts,
- caching and proxy server.

These services have restored the interest of part of the HEP community, who were wary of internal use of a system which was completely open.

Brief review of WWW properties

WWW is based on HTTP: a light-weight stateless protocol with fleeting connections. This is the basis for the efficiency of WWW.

HTML is an SGML markup which all viewers are required to understand. HTML is ***not*** a format, but a language for expressing document structure. This is the basis for the universality of access: the client controls the presentation according to its capabilities.

Navigation is limited to hypertext links & index searches. This is the basis of its extreme ease of use.

URLs form a uniform name space for objects on Internet. This is the basis for WWW's capability to incorporate any existing and future information access protocol on the Internet. Thus they give access to other systems: local files, ftp, telnet, news, gopher, WAIS, Archie.

WWW scales, because URLs use the Internet domain name server schemes. This is the reason for the performance despite of the growth rate and distribution.

Servers interpret URLs. The server decides what it will do, the client is unable to distinguish URLs pointing to files from those requiring data base access or synthesis. This is the basis for the ease of publishing of existing material: a URL subspace can be constructed which will map onto a collection of information independent of its structure.

Format negotiation is open: closed groups of users may decide to exchange information in a format proper to them, without having to worry about the users at large. This is the basis of specialist and in-house usage.

Inside HEP

The use of WWW has become routine in the HEP community: most institutes have servers providing information on their organisation and experiments.

All four LEP experiments have adopted W3 as the way into the future, although time will be required for the switch-over. The LHC proto-experiments, Atlas and CMS, are committed to W3 and have provided the main push for access authorisation and other server improvements. Internal CERN usage includes both physics and administrative information about the lab, as well as a preprint server and the Library.

In Europe, DESY and NIKHEF have been the first and most enthusiastic users, providing many good ideas and suggestions, with smaller European institutes following, sometimes in difficult circumstances (we think of the eastern European countries). In the US, SLAC, FNAL and Los Alamos have been the driving forces. Very little effort has often been available for results which must be termed spectacular. The FNAL activity has lead to the Astrophysicists moving to the Web and the Los Alamos preprint server brought the theoretical physicists into the fold. SLAC cooperated with CERN from the very early days in all respects.

We conducted a survey of the use of WWW by HEP institutes. The responses were very inhomogeneous, but not entirely without interest. We give here an overview of results and try to put forward some comments.

Implementation history of WWW servers

Figure 1 shows an incomplete and historically perhaps not quite accurate picture of the spread of WWW servers inside HEP institutes. Gathering this data is very difficult, because the exact dates have been forgotten or lost, the exact number of servers is not quite known either. It is not well-defined what constitutes a service and/or a server. The graph gives some idea though:

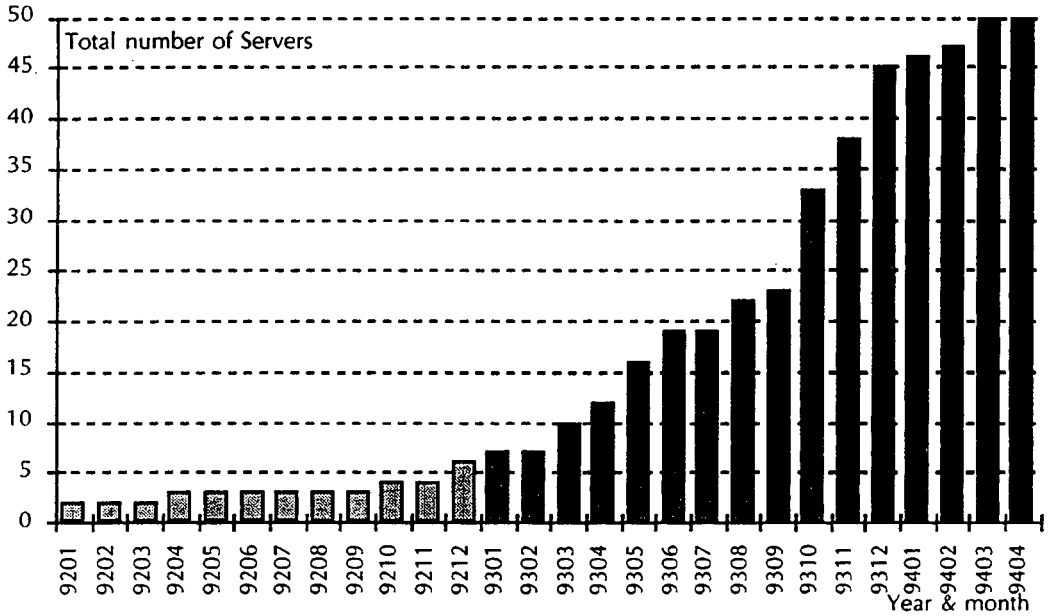


Figure 1: Installation history of servers

The shape looks roughly like an S-curve, seemingly indicating that a saturation effect is occurring, but we think this is mainly due to the fact that the so-called "main" servers have now almost all been installed. The effort goes more into specialised local ones, which are more difficult to know about and are often not reported.

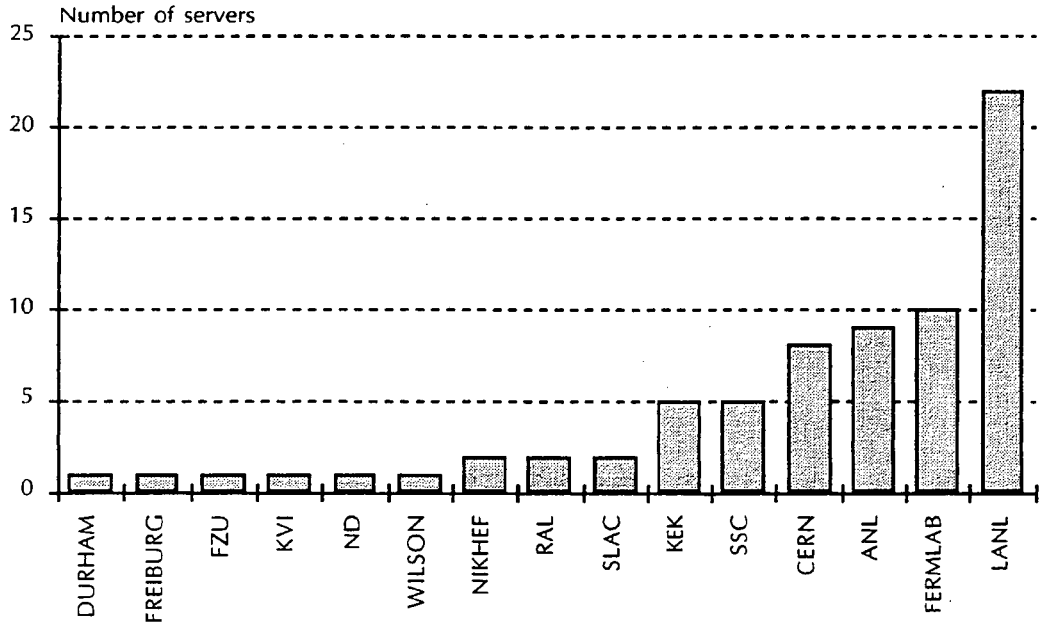


Figure 2: Number of servers per institute

Number of servers

If we now try to look at the number of servers per institute (figure 2), we also get an incomplete picture. CERN looks like it has fewer servers than Fermilab, but the 18 servers of the Delphi on-line system have not been counted. The servers of the small experiments at CERN who actually use the WWW system for diffusion are all lumped together as one single server. LANL has many servers, but they are not all purely HEP. CERN has at least one server that is partially administrative. It also holds the info.cern.ch server, which is entirely outside HEP.

Number of documents

The next item in the survey was even more inconclusive: number of documents distributed per institute (figure 3). This has a very different meaning to different people.

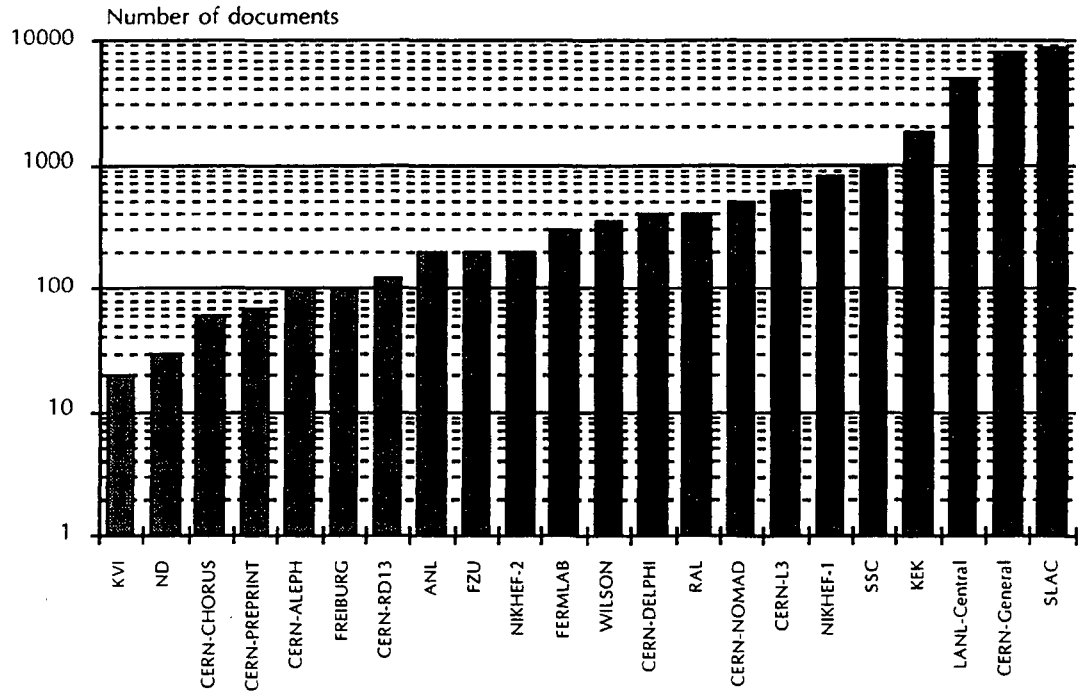


Figure 3: Number of documents per institute

The richness of a server at a particular site is difficult to estimate: shall we count only HTML files, or will we take into account every document that could conceivably be generated in HTML on the fly? Does a news repository count? Does an imbedded image count as a separate item? What if it is reused in another document? Yet we all do have a feeling for what server richness means. The measure of it needs to be refined and could be the subject of a paper in itself, though it belongs in the WWW domain, and not in computing for HEP.

The figure needs a log scale: certain sites have become extremely rich, others have remained modest. Note carefully that richness is not at all an indication of usefulness: baroque sites will come out well in this graph, but modest and useful ones may well be much more appreciated!

Activity

A good measure, which should be easy to obtain is the number of requests a server gets per day (figure 4).

However, a glance at the figure suffices to make one doubt that Fermilab really gets only 30 requests, being outperformed by a newly webbed CERN experiment like Chorus. Here too, we had to go by very incomplete information, often guessed by the administrators who for valid reasons did not run logging or did not keep log files. Gopher and ftp accesses also do not show up.

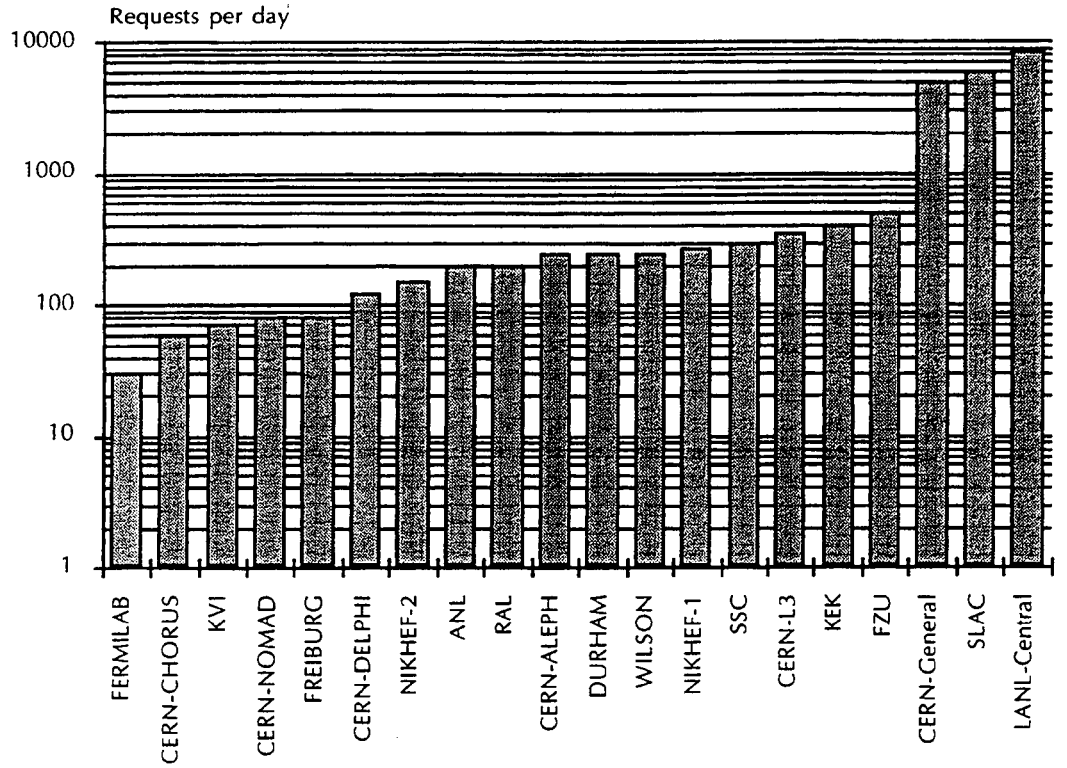


Figure 4: Requests per day

In addition, if the information is not well-structured or too fragmented, users may go through a lot of documents before reaching the one they want, and they may do so every time, either because they do not use a hot-list or because their browser does not have one.

What is wanted here is the number of documents actually read, i.e. the target document, not the documents along the path. We call this number the "target access frequency", to distinguish it from the "indicated access frequency". It is a number that only the users can give us, estimating it needs a much larger and semi-psychological survey.

Furthermore, we have no idea of how many accesses are still just for the purposes of demonstration.

In the absence of serious editors, a large number of accesses is also made simply by the authors while they construct their documents and by peers who review them. A quick glance at a particular subset of documentation on www.cern.ch showed that a quarter of the HTML files had been accessed only once ever. That access obviously was by the author(s), testing their document (although even there, some have been known to publish on the Web without ever looking at the document themselves, which explains some of the error messages.)

The target access frequency, or actual use by the real consumer of the information, is unknown. Ideally, it should be an order of magnitude higher than that made by the authors and the demonstrators. We think that the target access frequency is a sizeable, hopefully dominating fraction of indicated use, but nevertheless we suspect that the desired order of magnitude difference does not exist yet.

Policies

Finally, we asked administrators whether there was any laboratory policy about the contents published on the Web. The results were:

Fermilab has a policy,

SLAC is working on one,

CERN has an unofficial one,

LANL follows the same policies as for printed material,

RAL obeys the Data Protection Act.

Two WWW administrators said they did not know what the question might mean.

Some took the question to mean: "are there documents with restricted access?". The answer to this question was generally yes, but of course this does not mean that their management is aware of how their laboratory is seen outside HEP, or even that there is freely accessible WWW information about it at all!

The others had used no policy. We did not ask whether the WWW administrators had raised the policy question with their management.

Future use of WWW in HEP

In the past year, several technological changes have happened to W3, including more sophisticated browsers (clients) and servers, as well as better specifications of the HTML markup language and the HTTP exchange protocol. Inside the HEP community we foresee a shift from development effort to support and sophisticated usage. While pure Web development will increasingly be done outside HEP, there will be HEP-specific developments, such as detector surveillance, data analysis, collaborative work, data storage, experiment planning and scheduling, publication of results and so on.

Showcase: Delphi on-line

One of us (M. Dönszelmann [1]) has successfully experimented with bringing on-line information into the web. The 18 existing non-WWW servers in the on-line system were extended to incorporate the WWW library so they also became WWW servers. This makes the detector and machine information available to all collaborators.

In addition, the Delphi servers will use the forms and image-map features for the following:

- control of the system via gateways to the existing control system,
- Event Display,
- Submission of jobs on remote nodes for data analysis,
- software maintenance (with CVS for instance).

The experience here shows the same problems of WWW that have also been reported from many other sources:

- people get "lost" in the web: organisation of the information is important,
- lots of stuff needs to be converted, which creates a maintenance problem,
- there are no group annotations.

These problems are being addressed in the future developments.

An interesting note is that this work also resulted in a reliable server for VMS systems, now distributed to over a hundred HEP nodes.

Future developments of WWW

HyperText Markup Language (HTML) is the name for the SGML document type definition that is used between clients and servers. The Web sends only the document content and structure, not the layout, and so users retain the freedom of choosing the

platform on which they work and the presentation they like best.

This property is to be preserved: it makes the bulk of the transfers efficient and places the penalty of sophisticated display at the user's side, where it belongs.

Yet, there is a large community of authors who think that the exact presentation the reader receives should be under their control. They favour form over content. This is not always an unreasonable request, and SGML systems have indeed a way to overcome this: the style sheet. Therefore, we will see systems which will transfer not only the content, but also a parallel set of indications as to how to present the content. Other systems, which combine the two in an inextricable manner are perhaps not so good.

Inevitably the early HTML definition was incomplete, for example tables and forms were absent, but this has been largely corrected in HTML+. The single major deficiency for use in HEP remains the lack of mathematical formulae.

The original design of the Web was extensible, for example allowing for format negotiation, thus the new version has been able to accommodate many new features. Any document format can be sent, although the most popular ones are those which most clients are capable of presenting, such as HTML, plain text, gif images, PostScript, JPEG photo format, MPEG movie format and a sound format.

On the protocol side, we will soon see the possibilities for clients to return documents through the PUT and POST methods. This is the mechanism required to implement a true browser/editor. A good SGML based browser/editor will give HEP the collaborative work tool that it needs.

Desires

Quality

The software available today is of reasonable quality, but not yet produced in a commercial way. During discussions we found a fairly even split between people who want public domain software (for reasons of budget and distribution) and those who want industrial software (for reasons of stability and guaranteed support). We think 1994 will probably give us at least some of the tools from industry.

Editors

There still are no editors worth the name. A successful editor must:

- fulfill the navigation functions of the current browsers,
- allow the user to modify the documents which he/she possesses,
- be easier to use than today's word processors on Macintoshes/PCs, so that technical and secretarial staff can maintain information routinely.

Security

There is still no real security. Passwords fly over the Internet in readable form, so does the information. It will be some time before we have real security on the Internet, maybe never. There is ***no*** purely technical solution to security: important factors of society, psychology and civilisation (culture) matter just as much. Therefore this problem will not be solved inside the WWW development effort.

Searches

Many of us want searches everywhere on the Web. Work is in progress to make indexes and search engines that will scale. In the meantime, there are some good indexes and meta-indexes. Inside HEP, the task is less daunting, since we know our community quite well and could even envisage setting up a HEP index.

Other networks

Out there are non-Internet networks using other protocols than TCP/IP. There will probably be a serious effort to either transport WWW to these protocols or to construct a similar system on them. The most impressive contribution from the commercial sector will probably be security, in which they are far ahead.

Other interaction models

In many cases, especially on-line controls and data acquisition, the stateless http protocol with its fleeting connection is inefficient. Connections for synchronous operation with state will be required for spreading the Web to those applications.

Presentation

"Presentation must be as ***I*** want it" (this is a wish of both server and client!). This is reasonable from both sides, and is solved by the inclusion of style sheets on both sides. There is however not yet a style sheet formalism.

The other solution to the problem, putting formatting markup into HTML, ***must*** be resisted. There is a tendency to want HTML tags for presentation like "italics". Italics may be used for presenting the name of the publication in a references list, but it may also be used for the list of authors. If both are marked up as "italics", then the presentation is right on those systems that can display italics, but not on others. The meaning of the strings is also lost. If on the other hand the strings are marked up with "author" and "journal" respectively, then we know much more, and can still do the presentation as we want. Formatting markup creates lazy authoring and makes some forms of lookup impossible.

Conclusion

The survey taught us that it is impossible to give a reasonable picture of the state of WWW in HEP (or, for that matter, anywhere else) after a simple enquiry. The number of parameters governing the state is large. Many aspects are subjective and open to various interpretations. An in-depth survey will take a large effort and will not converge because of the rapid evolution.

We need now to build up definitions and at the same time tools that filter out everything that does not correspond to them, so that an automatic survey can be made.

However, we reckon that in the present state of things, such a survey with tools is well beyond our resources. We can only hope that the WWW community will in time provide them.

We shall not spend effort on statistics just for the sake of history: that is the work of the historian. A large part of the interesting early history of WWW seems however already to be lost irretrievably. Maybe something can be salvaged if the pioneers who made WWW a success in the beginning do not try to recall the early days in numbers and figures.

Inside HEP, we may be proud of the WWW development. We should also look forward to the day that WWW is handled completely outside our community and we will have become users.

References

[1] A configurable Motif Interface for the DELPHI experiment at LEP, M. Dönszelmann; DELPHI 92-150 DAS 134

CooMan- a Global Collaborative Project Management System.

Jano M. de Souza, Sergio Palma
COPPE - Federal University of Rio de Janeiro
jano@CERNVM, palma@cos.ufrj.br

Abstract. Project Coordination and Management have long been recognized as an area with growing problems and unsatisfactory solutions. Conciliating flexibility with target achievements is historically the main problem to face. The difficulties have been growing at more than linear ratio with the size and complexity of the Projects being developed in the present days.
The HEP communities suffer additional challenges because of the distributed nature of the collaborations, the novelty of each project; and the less authoritarian form of leadership and management of team and individual. This prevents the adoption of more centralized focus of decision.
CooMan intends to be a Global Collaborative Project Management System. This paper discusses the basic aspects of the concepts involved, outlining how task coordination, acts of speech, and World-Wide hyper media can be used to support project management activities. A distributed tool to implement such proposition is described, and a first prototype is presented.

1 - Introduction. CooMan is a collaborative system, now under development, based on the perspective of helping work groups to more effectively conduct their communication, coordination, and information sharing activities. We are interested in the advantages gained with collaborative processes, speech acts and the use of the World-Wide Web, in helping to solve the problem of controlling a project with groups physically located in different parts of the world.

The next section illustrates some ideas of Project Management Techniques. The third section introduces a Conversation Model for Task Coordination based on the Language/Action perspective. The World-Wide Web is the subject of the next section. The last section outlines the implementation of the CooMan.

2 - Conventional Project Management Technique. The project planning and managing effort involves two major concepts: research and estimation. The definition of the problem scope imposes research while the solution achievement deals with some degree of uncertainty of the estimation. Project Management can be achieved through some techniques as the PERT network, the Gantt Chart, the Critical Path Method among others.

3 - Collaborative Processes. Collaborative Processes can be defined as sequences of action performed by individuals in a way that the action of one member affects the space of actions of the other members of a group. Much of this theory is involved while project management is under concern.

3.1 - Speech Act Theory. Much work in Computer-Supported Cooperative Work (CSCW) has been influenced by Speech Act Theory. Our theoretical orientation is taken from the Language/Action perspective of Winograd and Flores [1]. Their work is based on a hermeneutic view of human interaction which holds that all human actions can be seen as being performed through language.

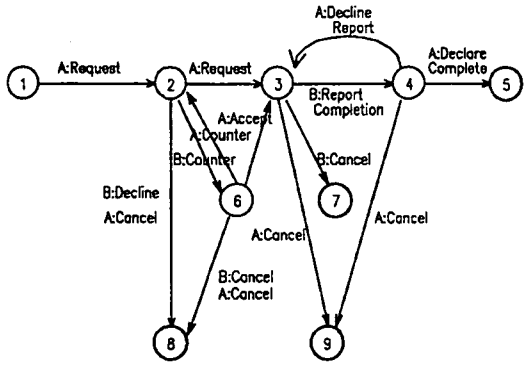


Fig. 1 - States of Conversation for Action[2].

3.2 - Conversations for Action. Speech Acts participate in larger conversation structures and are not stand alone events. An important example is the case of a simple "conversation for action," in which one party (A) makes a request to another (B). The request is interpreted by each party as having certain conditions of satisfaction, which characterize a future course of actions by B. After the initial utterance (the request), B can accept (and thereby commit to satisfy the conditions); decline (and thereby end the conversation) ; or counter-offer with alternative conditions. Each of these can have continuations. The overall structure is diagrammed in Figure 1.

4 - The World-Wide Web. The World-Wide Web is a Distributed Hypertext System created by CERN, the European Laboratory for particle physics in Switzerland. The WWW was designed to help physicists access any information available on the internet, no matter where it physically lies.

5 - CooMan: Joining New and Traditional Techniques. The basic aspects of the conceptual model of the CooMan system can be divided in two main subjects that are the **Structure model** and the **Cooperative network**. Each of these use the Language/Action perspective for their implementation which will be in each case explained in the following section.

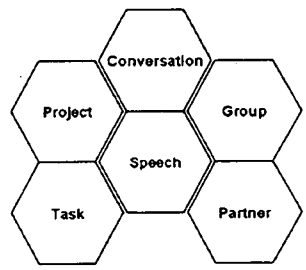


Fig. 2. The Conceptual Model

5.1.1 - The Structure Model. The objects which are the basis for the system are the Projects, Tasks, Partners, Resources, Groups, Conversations, and Speeches. These objects

are assembled to form a consistent set of types to represent the actual scenario of where actions take place and are presented in Fig. 2.

In order to identify clearly the attributes of the Project and its Tasks CooMan proposes an outline that may be used as a guide. In this initial stage the scope of the Project which has its objective, major functions and other characteristics should be informed . These attributes should include the Project's Name, Coordinator, Dates (Creation, Initial and Final), Objective, and Description.

- **Conversation for Action**

Request - Cooperation -Meeting - Information

Offer - Cooperation

- **Conversation for Possibilities**

New

Responsibilities - Coordinator -Manager

Framework - Project -Task -Group -Partner

- **Notification &** - **Reply**
- **Housekeeping &** - **Query**

Speech - I didn't read -My Partner didn't read - I didn't Reply

Fig. 3 - Conversation types of CooMan.

Human Resource coordination is the most important goal of CooMan and in its favor all the work was done. In order to satisfy the Language/Action perspective we decided to name it Partner.

.5.1.2 - The Conversation Model. CooMan supports several types of Conversation for Action and Conversation for Possibilities, following the Language/Action perspective. These are all listed in Figure 3.

5.2 - System architecture. The architecture of the system can be divided into three main parts: the data module, which derives information automatically by the conversations it handles, the interactive module,composed by the cooperation handler and the user interface, and the graphical module.CooMan system will be developed with the use of the resources available in the WWW, as lists, buttons, text editors, and the hypertext facility.

5.3 - Conclusions. In this paper, we have described a system called CooMan that intends to join traditional project Management with Speech Act Theory, allowing a distributed access through a global Hypermedia system. We believe that with this system it can be relatively easy to keep track of Projects even in a distributed environment.

REFERENCE

[1] Flores, F., Hartfield, B., Winograd, T., *Computer Systems and the Design of Organizational Interaction*, ACM Transactions on Office Information Systems, 1988.

[2] Malone, W, Crowston, K, *What is Coordination an How Can It Help Design Cooperative Work Systems*, Conference on Computer-Supported Cooperative Work, 1990.

[3] Souza, J, Palma, S, *CooMan - A Global Collaborative Project Management System,* Tech. Report 296/94, COPPE/UFRJ 1994.

DYNAMIC PERCEPTRON: SOME THEOREMS ABOUT THE POSSIBILITY OF PARALLEL PATTERN RECOGNITION WITH AN APPLICATION TO HIGH ENERGY PHYSICS.

Antonio L. PERRONE[◇○†], Gianfranco BASTI [†○]
[◇]Dept. of Physics, II Univ. of Rome "Tor Vergata",
Via della Ricerca Scientifica 1, I-00133 Rome (Italy).
[○] Istituto Nazionale di Fisica Nucleare-INFN Section of Rome "Tor Vergata"
[†] Pontifical Gregorian University, P.zza della Pilotta 4, I-00184 Rome (Italy).

Abstract

In the context of M.Minsky's and S.Papert's theorems on the impossibility of evaluating simple linear predicates by parallel architectures we want to show how these limitations can be avoided by introducing a generalized input-dependent preprocessing technique that does not suppose any a priori knowledge of input like in classical input filtering procedures. This technique can be formalized in a very general way and can be also deduced by meta-mathematical arguments. A further development of the same technique can be applied at level of learning procedure to introduce in such a way the complete notion of "dynamic perceptron". From the experimental standpoint, we show two applications of the "dynamic perceptron" in particle track recognition in high-energy accelerators. Firstly, we show the amazing improvement of performances that can be obtained in a perceptron architecture with classical learning by adding our "dynamic" pre-processing technique, already introduced last year in another paper presented at this Conference. Secondly, we show the first results of this technique extended also at the level of learning procedure always applied to the problem of particle track recognition. This work is a part of "Fenice" international collaboration supported by INFN (National Institute for Nuclear Physics) devoted to the study of the time-like electromagnetic form factor of neutrons obtained by electron-positron high energy collisions in ADONE (Frascati, Rome) storage ring.

1 The problem: parallel computation in the "geometrical" perceptron.

Let us consider a classical perceptron computing on a lattice R a linear function $\Psi(X)$ of a generic *input* $X \subset R$. The problem we want to discuss is whether there exists a predicate Ψ able to assume the input characteristic function, given a fixed topology according to which the perceptron "reads" the input. In other terms, given some partition $\{A_i\}$ of the retina R, our aim is to ascertain whether there exists a predicate Ψ able to decide whether a generic input X has at least one point in each cell A_i. Formally,

Definition 1 *Let the retina R be an arbitrary set of points (it can be also finite).*

Definition 2 *We define as the input X a generic subset of R: $X \subset R$.*

We indicate by bold letters the *collections* of subsets of R

Definition 3 *Let $\varphi_{\mathbf{F}}(X)$ be a predicate or a mask:*

$$\varphi_{\mathbf{F}}(X) = \begin{cases} TRUE & iff\ X \subset \mathbf{F} \\ FALSE & iff\ X \not\subset \mathbf{F} \end{cases}$$

Effectively, a predicate is a variable statement whose truth ($\varphi = 1$) or falsity ($\varphi = 0$) depends on the choice of the geometric figure X drawn on the retina R.

Definition 4 *Let $S(\varphi)$ be the support of a predicate φ:*
$S(\varphi) \mid \forall X \subset R \ \ \varphi(X) = \varphi(X \bigcap S)$

Where $X \bigcap S$ is the intersection of X and S, that is, the set of points belonging both to X and S.
This definition helps us to make explicit the idea that the function φ "depends" only on a certain subset of R. So, $S(\varphi)$ denotes the subset of R upon which φ *really depends.*

Definition 5 *Let Ψ be a linear threshold function with respect to the collection Φ of predicates φ in R:*

$$\exists\theta \in \Re, \exists\{\alpha(\varphi) \in \Re\}\forall\varphi \mid$$
$$\Psi(X) = \lceil \sum_{\varphi \varepsilon \Phi} \alpha(\varphi)\,\varphi(X) > \theta \rceil = L(\Phi) \quad (1)$$
$$\text{with } \lceil x \geq 0 \rceil = +1, \quad \lceil x < 0 \rceil = 0$$

Obviously, the predicates of which support is very small are too *local* to be interesting. The primary interest is in fact for those predicates of which supports are the whole retina R, but that can be defined as linear threshold combinations of predicates with much smaller supports. To quantify such an exigency, let us introduce the notion of *order* k of a predicate. This notion indicates that no one of the predicates composing the linear threshold function has a support containing more than k points. Formally:

Definition 6 *We define as order of a predicate the smallest integer number k for which there exists a set Φ of predicates φ satisfying Ψ :*

$$\begin{cases} \mid S(\varphi) \mid \leq k, \ \forall\varphi \in \Phi \\ \Psi(X) = L(\Phi) \end{cases} \quad (2)$$

where $\mid S(\varphi) \mid$ is the number of points in $S(\varphi)$.

Definition 7 *We define two predicates φ and φ' as equivalent with respect to the group G, $\varphi \overset{G}{=} \varphi'$, if there exists a member g of G such that $\varphi(gX)$ and $\varphi'(X)$ are the same for each X.*

Definition 8 *We define as equivalence class of predicates the set Φ of predicates φ equivalent with respect to the group G.*

We can thus formulate without ambiguity the following theorem " $\Psi_{\text{One-in-a-box}}$" [1]

Theorem 1 *Let $A_1, ..., A_m$ be disjoint subsets of R and define the predicate:*

$$\Psi(X) = \left\lceil \left| X \bigcap A_i \right| > 0, \forall A_i \right\rceil \quad (3)$$

that is, there is at least one point of X in each A_i. If for all i, $|A_i| = 4m^2$, then the order of Ψ is $\geq m$.

Corollary 1 *If $R = A_1 \bigcup A_2 \bigcup ... \bigcup A_m$ the order of Ψ is at least $\left(\frac{1}{4}|R|\right)^{1/3}$.*

2 A possible solution: the "dynamic" perceptron

2.1 An intuitive presentation of the proposed solution

Our suggestion is that it is possible to maintain the polynomial associated with $N_j(X)$ within the limit m imposed by the topology of the single partition A_i, on condition that the topology given by the set $\mathbf{F}$ of the different masks φ (see definition 3) is allowed to modify itself with respect to their input A_i. Let us consider the linear expression defining in a general form whichever predicate Ψ:

$$\Psi(X) = \left\lceil \sum_{\varphi\in\Phi} \alpha(\varphi)\varphi(X) \right\rceil \quad (4)$$

the main problem is the *a priori* evaluation of the summation with respect to the input and/or with respect to the predicate we are actually considering. Group theory notions are very appropriate in this context because they make invariant the masks φ under the actions of group elements. Following this idea, we can simplify the precedent summation by grouping the masks according to *equivalence class* Φ_j. I.e., the class composed by all the predicates invariant with respect to a given group G. In this way, the summation is simplified because we can demonstrate [1] that masks belonging to the same equivalence class have the same weights. So, we can rewrite the predicate Ψ in the simpler form:

$$\Psi(X) = \left\lceil \sum_j \alpha_j \sum_{\varphi\in\Phi_j} \varphi(X) \right\rceil \quad (5)$$

At this point let us evaluate the number $N_j(X)$ of the masks (characteristic functions) $\varphi(X)$ satisfied by the generic input X for each equivalence class.
This term grows up combinatorially with the number of the input points $\mid X \mid$ and with the number of the mask points $\mid S(\varphi) \mid$, $\varphi \in \Phi_j$:

$$\binom{|X|}{|S(\varphi)|} = \frac{|X| \cdot (|X|-1) \ldots \cdot (|X| - \mid S(\varphi) \mid +1)}{\mid S(\varphi) \mid !} \quad (6)$$

This growth represents an insuperable difficulty because it requires a mask with a support as large as the retina R. Of course, this destroys in principle the perceptron capacity of parallel computation because a calculation unit able "to see" the *whole* input becomes necessary.
This limitation occurs every time we want to work mathematically with sets in which all the elements are completely specified. According to us, a possible solution is to substitute such a definition of a set via the previous complete specification of all its elements (in this case, all the masks $\varphi \in \Phi_j$) with a dynamic procedure of *mutual element-specification/set-definition*. In this way, starting from the initial specification of one element, we can construct progressively the definition of the whole set. Or, better, we can always reach a set definition as general as we *effectively* need to perform the calculation.
In the next paragraphs we offer some formal demonstrations of this procedure in some cases that result to be irresolvable for the classical "geometric" perceptron.

2.2 The Dynamic Perceptron and the solution of "Ψ-one-in-a-box" problem

Let us consider the combinatory term providing us with the number of masks satisfied by the generic input X:

$$N_j(X) = |\{\varphi \mid \varphi \in \Phi_j \text{ AND } S(\varphi) \subset X\}| \quad (7)$$

Minsky and Papert themselves stress that the evaluation of this term would become significantly easier if an appropriate pre- processing was available. Nevertheless, the authors abandon the study of such a solution because in their cognitive scheme this would imply an a- priori knowledge of the whole input.
On the contrary, we can obtain the same result without any a priori knowledge. Indeed, if we consider only the density and the distribution of the input points, it is sufficient to vary the support S of each mask φ according to the density variation of the input X "read" by the mask itself in two successive instants with two different topologies. Formally [1]:

$$\mid \Delta S_X(n) \mid \propto \varphi_{S_X(n)} - \varphi_{S_X(n-1)} \quad (8)$$

This process stops if we have the *right* topology in S for the input X; in this case the equation for ΔS reads:

$$\varphi_{S_X(n)}(X) = \varphi_{S_X(n-1)}(X) \Leftrightarrow \mid \Delta S_X(n) \mid = 0 \quad (9)$$

We give now a list of formal definitions to make rigorous such a procedure. We premise, however, that from now on we consider only *square geometries*. In fact, it is easy to obtain extensions for every type of geometry by the simple substitution of the *linear* unit of Δl with a *curvilinear* unit.
From now on we consider a *bidimensional* retina $R = \{x, y\}$. More exactly,

Definition 9 *R is a bidimensional lattice $\{x, y\}$ of step p.*

Definition 10 $X \subset R$

Let us consider a generic element i of the input set X, corresponding to the pair $\{x_i, y_i\} \in R$: $i \in X$. For each point i of the set X let us construct the sets $S^i_X \subset R$ defined as follows:

Definition 11

$$\begin{aligned} S^i_X(0) &= \{0\} \\ S^i_X(1) &= \text{squareofside2}p\text{centeredat}i \\ S^i_X(n+1) &= S^i_X(n) + \Delta S^i_X(n),\ n > 1 \end{aligned} \quad (10)$$

[1] For the rigorous definition of $\Delta S_X(n)$ see below the Definition 13.

Definition 12 $\varphi_{S^i_X(n)}(X)$ = *number of X points in $S^i_X(n)$.*

Definition 13

$$\Delta S^i_X(n) = \begin{cases} \{0\} \text{ if } \varphi_{S^i_X(n)}(X) - \varphi_{S^i_X(n-1)}(X) = 0 \\ \text{the set constructed by subtracting} \\ \text{to the square centered at } i \\ \text{and of side } 2np \\ \text{the square centered at } i \\ \text{of side } 2(n-1)p \\ \text{if } \varphi_{S^i_X(n)}(X) - \varphi_{S^i_X(n-1)}(X) > 0 \end{cases} \quad (11)$$

Definition 14

$$n_{max} = n : \Delta S^i_X(n) = 0\ \forall i \quad (12)$$

We have thus all the necessary definitions to enunciate the following.

Theorem 2 *Let $A_1, A_2, ..., A_m$ be disjoint square subsets of R and let us define the predicate:*

$$\Psi^D(X) = \left[\left|X \bigcap A_i\right| > 0 \text{ for any } A_i\right] \quad (13)$$

that is, there is at least a point of X in each A_i. We indicated by $\Psi^D = \left[\sum_{\varphi_{S^j_{A_i}(n)}} \alpha(\varphi_{S^j_{A_i}(n)})\varphi_{S^j_{A_i}(n)}(X)\right]$ the "dynamic" perceptron where the masks $\varphi_{S^j_{A_i}(n)}(X)$ are determined following the definitions (11-13) . Hence, if for all $i \mid A_i \mid = 4m^2$, then the order of Ψ^D is $4m^2$.

Proof. Let $A_1, \ldots, A_m$ be a set of square disjoint sets $A_i \bigcap A_j = \emptyset\ \forall i \neq j\ i = 1, \ldots, m$
If we consider as input set the sets A_i, the "dynamic" equations (11-13) are to be rewritten as follows:

$$\begin{aligned} S^k_{A_i}(0) &= \emptyset \\ S^k_{A_i}(1) &= \text{squareofside2}p\text{centeredat}i \\ S^k_{A_i}(n+1) &= S^k_{A_i}(n) + \Delta S^k_{A_i}(n),\ n > 1 \end{aligned} \quad (14)$$

$$\varphi_{S^k_{A_i}(n)}(A_i) = \text{number of } A_i \text{ points in } S^k_{A_i}(n) \quad (15)$$

$$\Delta S^k_{A_i}(n) = \begin{cases} \emptyset \text{ if } \varphi_{S^k_{A_i}(n)}(A_i) - \varphi_{S^k_{A_i}(n-1)}(A_i) = 0 \\ \text{the set constructed by subtracting} \\ \text{to the square centered at } i \\ \text{and of side } 2np \\ \text{the square centered at } i \\ \text{of side } 2(n-1)p \\ \text{if } \varphi_{S^k_{A_i}(n)}(A_i) - \varphi_{S^k_{A_i}(n-1)}(A_i) > 0 \end{cases} \quad (16)$$

Hence, the following holds:

$$\min_{k \in A_i} \left| S^k_{A_i}(n_{max}) \right| = \mid A_i \mid = (2m)^2 = L \qquad (17)$$

because $\mid A_i \mid = 4m^2 \; \forall i$.

$$\max_{k \in A_i} \left| S^k_{A_i}(n_{max}) \right| = 4 \cdot \mid A_i \mid = (4m)^2 = M \qquad (18)$$

The proof of the precedent statements is immediate.
Moreover, the following holds: $\forall A_i \; \exists 1! \; S^k_{A_i}(n_{max}) = S_i(max) : |S_i(n_{max})| = L$.
Let us define $p_s(X) = \lceil \mid X \bigcap S \mid > 0 \rceil$ and $p_i(X) = p_{S_i(n_{max})}(X)$.
Let P be the equivalence class defined as follows:

$$P = \{p_i, i = 1, \ldots, m\} \qquad (19)$$

Let us explicate the predicate Ψ^D in its linear form (see the definition 5) as follows:

$$\Psi^D = \left\lceil \sum_{p \in P} \alpha(p) \cdot p(X) \right\rceil = \left\lceil \sum_{i=1}^{m} \alpha(p_i) \cdot p_i(X) \right\rceil \qquad (20)$$

The order (see the definition 6) of the predicate Ψ^D is thus:

$$\text{ord}\left(\Psi^D\right) = \max_i \left(|S_i(n_{max})|\right) = L = (2m)^2 \qquad (21)$$

Q.E.D.

2.3 A comparison with multi-layer architectures

The results till now obtained lighten also the classical problem of the comparison between single-layer and multi-layer architectures. Indeed it is well-known in literature [6] that the problems of *unbounded order*, e.g., $\Psi_{\text{One-in-a-box}}$, can be easily solved by multi-layer architectures in which the predicate φ is on its turn a linear threshold function:

$$\varphi_i = \left\lceil \sum_j \beta_{ij} x_j > \theta_i \right\rceil \qquad (22)$$

so that, the function Ψ becomes:

$$\Psi = \left\lceil \sum_i \alpha_i \left\lceil \sum_j \beta_{ij} x_j > \theta_i \right\rceil > \theta \right\rceil \qquad (23)$$

For instance, let us re-consider the just discussed problem of Ψ_{Parity}. It is possible to demonstrate that it can be computed by a simple muti-layer architecture. The trick is simply of determining a mask $\varphi_{(n)}$ that counts the number of points n of its support S and that gives as output 0 or 1, respectively if this number is or is not exceeded by the number of points $\mid X \mid$ of its input X. We are thus defining a predicate practically for each possible number n of points of the input. Obviously, if we can define an indefinite number of these predicates, any input X is allowed to have a number $\mid X \mid$ of points $x_i \in \{0, 1\}$ that is recognized by some of these masks. Formally:

$$\varphi_{(n)}(X) = \lceil \mid X \mid > n \rceil = \left\lceil \sum x_i > n \right\rceil \qquad (24)$$

Moreover, let us define (Ψ is defined below):

$$\begin{aligned} \alpha_0 &= \Psi(0) \\ \alpha_1 &= \Psi(1) - \alpha_0 \\ &\vdots \\ \alpha_{n+1} &= \Psi(n+1) - \sum_{i=0}^{n} \alpha_i \end{aligned}$$

Then, Ψ_{Parity} can be re-written in the form:

$$\Psi(X) = \left\lceil \sum_{i=1}^{|R|} \alpha_i \cdot \varphi_{(i)} > 0 \right\rceil \qquad (25)$$

where all the masks $\varphi_{(i)}$ have an order $\mid S(\varphi_{(i)}) \mid \leq n$ bounded by n. It seems then that the multi- layer architectures can solve any problem posed by the single-layer. The problem is that the requested number of predicates $\varphi_{(i)}$ grows up with the dimension of R. Indeed, in the summation of the equation (25) we count $\mid R \mid$ predicates. Practically, by a multi-layer architecture we simply displaced the unbounded growth from the *order* of the single predicate in the single-layer architecture to the *number* of these predicates in the multi-layer architecture.
In the multi-layer architectures it would be possible to recognize any pattern F simply by defining the single predicates φ as follows:

$$\varphi_F = \left\lceil \sum_{x_i \in F} x_i - \sum_{x_i \notin F} x_i \geq \mid F \mid \right\rceil \qquad (26)$$

In this way a *generic* class $\mathbf{F}$ of figures F could be recognized by:

$$\Psi_{\mathbf{F}} = \left\lceil \sum_{F \in \mathbf{F}} \varphi_F > 0 \right\rceil \qquad (27)$$

but this formula is trivial because it means simply that the class $\mathbf{F}$ has a characteristic function $\Psi_{\mathbf{F}}$ that can be expressed if we define *as many predicates* φ_F as the figures F of the class $\mathbf{F}$ are. Obviously, such a number grows too fast for any realistic implementation.

In this context, the philosophy of our *single-layer but dynamic* architecture (see the definitions 8-10) begins to reveal itself as a winning strategy. Indeed, also our architecture defines a new set of predicates $\varphi_{S_X(n_{max})}(X)$ for each input X. However, these predicates are *dynamic*. That is, because they depend on a dynamics that is a function of the input X, they *do not need* the simultaneous definition of all the predicates as, on the contrary, it is requested in a multi-layer architecture. In other terms, instead of defining all the characteristic functions $\varphi_{(i)}$ as it is the case of the multi-layer architecture, it seems more advantageous to use the result[2] of the dynamics of mutual redefinition between the characteristic functions $\varphi_{S_X(n)}(X)$ and their supports $S_X(n)$, driven by the individual input X according to the prescriptions of the definitions 8-10. In this case, indeed, we do not consider any longer all together the elements of the generic class $\mathbf{F}$, but only those *actually present* as inputs of the net. Notwithstanding this drastic limitation of predicates, the occurrence of a new figure $F \in \mathbf{F}$ is equally recognized by the net within the class $\mathbf{F}$ because the modalities of the dynamic redefinition of the predicates $\varphi_{S_F(n)}(F)$ and of their supports $S_F(n)$ with respect to the new figure F result to be the same of the other figures considered till now. This holds, as long as all these figures, e.g. F_i and F_j, belonging to the same class $\mathbf{F}$ result to be linked by a generic reversible transformation $T : F_j \underset{T^{-1}}{\overset{T}{\rightleftharpoons}} F_i$ defined on the retina R^3.

Effectively, in this case we can suppose that the dynamic rule for the mutual redefinition between supports S_X and predicates φ_{S_X} (see Eq. (8) and definition 13), is of the form:

$$| \Delta S_X(n) | \propto f(\Delta\varphi(n)) = f(\varphi_{S_X(n)} - \varphi_{S_X(n-1)}) \quad (28)$$

where $f : \Re^+ \longrightarrow \Re^+$ is a generic function defined on the positive reals and expressing the correspondence between the figures F_i and F_j at the level of the supports and of the predicates describing them. The limit of the convergence of the dynamics, however, does not change under the action of f since this obvious relation holds:

$$\varphi_{S_X(n)} - \varphi_{S_X(n-1)} > 0 \Longleftrightarrow f\left(\varphi_{S_X(n)} - \varphi_{S_X(n-1)}\right) > 0 \quad (29)$$

For this reason, the form of the definition 13 does not change under the action of f. This means that the supports $S_X(n_{max})$ of the functions $\varphi_{S_X(n_{max})}$ at the end of the dynamics for the two inputs F_i ed F_j of the same class $\mathbf{F}$ are comparable, so that their classification by the weights α becomes possible. This result is made more evident by the application discussed in the next paragraph. Another complementary strategy to solve the problem of the unbounded order of single-layer architectures and/or of the unbounded number of predicates of multi-layer architectures is pursued by the "new" *connectionist* architectures and particularly by the so-called *back-propagation* algorithm. Effectively, this algorithm can give good results on *real problems* because generally they can be simpler than the ideal case discussed in the perceptron theorems. Nevertheless, it is well-known that the convergence time of learning dynamics of the back-propagation algorithm can be often excessively long. It seems thus that the use of non- linear transfer functions in the learning dynamics, if sometimes can result useful in practice, generally tends to transfer the computational problem from the "space" domain of the unbounded number of the predicates, to the "temporal" domain of the necessary convergence time. This depends on the fact that the use of a non-linear term in the learning dynamics does not solve formally the computational root of the problems here discussed.

However, a formal demonstration of this hypothesis would require a more deep and attentive discussion because it ultimately involves the problem of the formal demonstration of the so-called *NP completeness* theorem in computability theory. Nevertheless we outline in the final section of this work the main lines that we intend to follow in the next future for dealing with this essential question from the standpoint of our "dynamic" approach to computability theory. For the moment it is sufficient to demonstrate experimentally in the next section that our algorithm, in dealing with a recognition task characterized by a high complexity of the input patterns (simultaneous presence of many scales in the pattern space), does not suffer the temporal limitations of the non-linear architectures just as it does not suffer the formal limitations of the linear architectures before discussed.

3 An application to high-energy particle recognition

3.1 Introduction

We applied the precedent theorems to solve a problem common to the great majority of the experiments in high-energy physics: the automatic discrimination/recognition of the produced events. Several works have proposed to use also neural net architectures for this aim [3]. Generally, the best results do not exceed the 70%-85% of successful discrimination/recognition on Monte-Carlo simulations of real data.

A fundamental research field in high-energy physics is the understanding of the nucleon structure. This aim can

[2] For instance, the support $S_X(n_{max})$ of the dynamic predicate $\varphi_{S_X(n_{max})}(X)$.

[3] For instance, the two figures are coincident if rotated, translated, etc.

be achieved through the study of nucleon electromagnetic form factors. In the time like region ($q^2 > 0$) the nucleon (N) form factors can be measured through the "branching ratio" of the annihilation process:

$$e^+e^- \longrightarrow N\bar{N} \tag{30}$$

The differential cross section of this reaction in the "one photon exchange" approximation, is given by:

$$\left(\frac{d\sigma}{d\Omega}\right)_{c.m.} = \frac{\alpha^2\beta}{4s}\left[|G_M|^2(1+cos^2\theta) + \frac{4M^2}{s}|G_B|^2 sin^2\theta\right] \tag{31}$$

where: $s = q^2$ is the square of the energy in the center of mass, θ is the angle between the electron beam and the nucleon direction, β and M are respectively the velocity and the mass of the nucleon and α is the fine structure constant.

The actual experimental knowledge of the nucleon electromagnetic form factors is not very satisfactory. The proton form factors have been extensively investigated by high statistics experiments in the space-like and, recently, time-like region[4]. On the contrary neutron data are still poor in the space-like region and before the measurements recently performed at ADONE e^+e^- storage ring (Frascati, Rome), there was no experimental information about the neutron electromagnetic structure in the time-like region. Perturbative QCD (PQCD), Extended Vector Meson Dominance Models (EVDM) and other theoretical models give very different predictions on the neutron form factors in the time like region. A suitable quantity to compare different models is the following ratio of the cross-sections σ:

$$r = \frac{\sigma(e^+e^- \longrightarrow n\bar{n})}{\sigma(e^+e^- \longrightarrow p\bar{p})} \tag{32}$$

According to QCD sum rules, a value of $r = 0.25$ is expected at high q^2. EVDM-inspired theories predict $r \simeq 2 \div 100$ because of the not well established location of the vector meson recurrences.

Clearly, both the theoretical and the experimental state of art claim for data about $e^+e^- \longrightarrow n\bar{n}$ in view of a better understanding of the nucleon structure. The *FENICE*[5] experiment, at the upgraded ADONE e^+e^- storage ring, has been especially designed for the measurement of the cross section of annihilation processes (30) and hence for the determination of the neutron form factors in the time-like region. The *FENICE* experimental apparatus is described in details elsewhere [4]. The experiment is producing every month from 1 to 10 millions of events and globally it will produce 300 millions. At first level triggering, only 30 millions of them will survive. Approximately, a sample of $\sim 10^5$ must be analyzed by humans. Events such as neutron-antineutron ($e^+e^- \longrightarrow n\bar{n}$), proton-antiproton ($e^+e^- \longrightarrow p\bar{p}$) and multi-hadron productions belong to the *non-collinear* events category. Events such as Bhabha scattering ($e^+e^- \longrightarrow e^+e^-$), gamma-gamma and muons anti-muons $e^+e^- \longrightarrow \mu^+\mu^-$ productions belong to the *collinear* events category.

The so-called "machine-noise" events constitute a numerous family. These events are the result of the following interactions: $e^+e^- \longrightarrow p\bar{p}\eta$, $e^+e^- \longrightarrow p\bar{p}\pi^0$, $e^+e^- \longrightarrow \pi\pi$, KK, $\pi\pi\pi^0$.

In our specific application, we began by distinguishing between muons and Bhabha scattering. This task is difficult because of the topology similarities in the two classes of events. The only difference is the presence of a limited swarming at the extremities of Bhabha tracks in opposition with the almost perfect collinear character of muon tracks. On the other hand, the elimination of muon tracks is really essential because, despite of the presence of veto-revelators, they effectively constitute till the 70% of the produced events. Indeed the μ particles, often originating from cosmic rays, are able to go through the revelation chamber practically with any possible bearing. In this way, they switch an absolutely casual number of revelators and/or interact with little swarming effects. We give in figure 1 some examples of such an occurrence.

Our task was thus double. Firstly, it was necessary to discriminate the Bhabha events from the muon events, because only the former are relevant for the process physics. Secondly, it was necessary to recognize and to filter cosmic events with strange topologies. The discrimination difficulty consists in the impossibility of using as parameters both the point number and the linear regressions on a straight line, because the point number is varying for each track so that it is not class discriminating.

3.2 The network structure.

Following the main ideas presented in the last subsection and discussed elsewhere [5], we designed a discrete state $\{0,1\}$ net of which input is the acquisition chamber bidimensional XY image. This image consists of 300 x 300 points, each with two possible states: $\{0,1\}$. The only available information is thus the *topological* information. No further physical information is used such as released energy, fly time, ADC, etc. The net after having received its input, evolves its topology according to the "dynamic" perceptron scheme of definitions 8-10. When it reaches a fixed topology, the simple summation of the "geometric" perceptron is calculated according to the following:

$$\sum_k \alpha_k \left(\varphi^k_{S_X(n_{max})}\right) \cdot \varphi^k_{S_X(n_{max})}(X) > \theta \tag{33}$$

[4]Experiment PS-170 at LEAR and experiment E760 at Fermilab.

[5]Fattori Elettromagnetici del Neutrone In Collisioni Elettrone-positrone

where the $\varphi^k_{S_X(n_{max})}$ are determined by the "dynamic" perceptron and the weights $\alpha_k\left(\varphi^k_{S_X(n_{max})}\right)$ are defined according to the following:

$$\alpha_k = \begin{cases} |S^k_X(n_{max})| \text{ if } |S^k_X(n_{max})| > \langle|S_X(n_{max})|\rangle \\ \alpha_{k_n}|S^k_X(n_{max})| \text{ otherwise} \end{cases} \tag{34}$$

The term α_{k_n} is determined according to the following succession:

$$\begin{aligned} \alpha_{k_1} &= 1 \\ \alpha_{k_2} &= \alpha_{k_1} \cdot |S^{k_2}_X(n_{max})| \\ &\vdots \\ \alpha_{k_n} &= \alpha_{k_{n-1}} \cdot |S^{k_n}_X(n_{max})| \end{aligned} \tag{35}$$

where the double index k_i runs along all the *dynamic* masks, redefined with respect to input X and not with respect to the partitions A_i. The weight renormalization by the factorial term $(\mathrm{MAX}_k|S^k_X(n_{max})|)!$ resulted to be useful in simulations. The average term $\langle|S^k_X(n_{max})|\rangle$ is given by:

$$\langle|S^k_X(n_{max})|\rangle = \frac{1}{N_X}\sum_k |S^k_X(n_{max})| \tag{36}$$

with N_X as a suitable renormalization constant. This particular strategy of weight definition allows the net to enhance input zones characterized by higher densities (scattering) typical of Bhabha events so that the net is successful in discriminating these tracks from μ collinear tracks. It is important to notice that two generic events X_1 and X_2 belonging to the same class, e.g., μ, are identified *by two different sets of weights* α_{X_1} e α_{X_2}. This derives directly by the proposed dynamics (35) for the computation of the weights α_k. Because they depend on the individual input X[6], all this suggests to us a procedure of *dynamic* definition also for the weights α that is similar to the procedure described in the definitions 8- 10.
This point requires a further inquiry and it is not developed here.
Notwithstanding algorithm simplicity, we reached the encouraging results shown in Table I on a sample of 3860 *real* particle tracks (2784 Bhabha and 1076 μ). The plotting of the net non renormalized output (i.e., $\sum_k \alpha_k\left(\varphi^k_{S_X(n_{max})}\right)\cdot\varphi^k_{S_X(n_{max})}(X) > \theta$ without the Heaviside function $\lceil x\rceil$) is given in figure 2. In figure 3 we show the plotting of the number of classified particles as function of the discrimination threshold θ.

Obviously we chose for θ the value of 1230 because it results to be the best compromise for the discrimination of the two particle classes. The net algorithm is written in F77 FORTRAN language. With the weight dynamics included, it requires approximately 5 minutes of CPU time on a 4 MIPS (*Peak* value) machine to recognize the above described set of 3860 real (not Monte- Carlo) particle tracks. This means that the algorithm is able to recognize the events *in real time by software*, because the event acquisition rate of FENICE apparatus is approximately 10 Hz.

[6]Notice that such a relationship between the net weights and the predicates is classical. Indeed, it corresponds to the original learning rule of the classical perceptron: see [1] ch.11.

4 Conclusions

Starting from M.Minsky's and S.Papert's classical work on linear perceptron, we demonstrated that it is possible to overcome under given conditions some fundamental limitations of parallel computation. According to the theorem (1), "$\Psi_{\text{One-in-a-box}}$", the order of linear function $\Psi(X)$ is $\geq m$, where m is the dimension of a retina partition. This means that the problem is intractable in parallel computation. We stressed that the main obstacle consists in the fixed topology by which the "geometrical" perceptron "reads" its input. We demonstrated that it is possible to overcome this limitation by introducing the concept of variable topology in a "dynamic" perceptron scheme able to accommodate itself on the individual input topology. In this case $\Psi(X)$ becomes $\leq 4m^2$. We applied the same principle also to the theorem (2), "Ψ_{Parity}", according to which the "geometrical" perceptron is unable to calculate in parallel the odd or even parity of its input. Also in this case the general principle of the dynamic redefinition of the topology on the input, allows us to overcome this limitation. It seems thus that the "dynamic perceptron" is another way to avoid the intrinsic limitations of the "geometric" perceptron. This approach seems indeed to be computationally more robust and time-saving than the multilayer architectures, back-propagation included, that are proposed till now to solve the same problems.
Finally, we applied the cognitive scheme of "dynamic" perceptron on a real case. That is, we applied this idea in particle tracks recognition in high energy physics. These results seem very promising, as we showed in Tab.I: the success percentage is high with a drastic reduction of calculation time.
Another component of the problem, that requires a further development, is the possibility of applying a recursive scheme of the same type of the scheme presented in the definitions 8-10 also to a learning dynamics. By such a development, it will be possible to outline a definitive theoretical justification of the computational higher robustness and velocity of the dynamic architecture here discussed with respect to multilayer architectures using a non-liner transfer function such as back-propagation algorithm.

References

[1] M.Minsky and S.Papert, *Perceptrons. An Introduction to Computational Geometry. Expanded Edition* (MIT Press, Cambridge Mass., 1988), p.59; p.48.

[2] H.D.Ebbinghaus, H.Hermes, F.Hirrzebruch, M.Koecher, K.Meinzer, J.Neukirch, A.Prestel and R.Remmert, *Numbers* (Springer, Berlin, 1990), p.101.

[3] B.Denby, Computer Phys. Comm. **49** (1988), 429.
C.Peterson, Nucl. Inst. & Meth. **A279** (1989), 537
M.Gyulassy and M.Harlander, Computer Phys. Comm. **66** (1991), 31.
B.Denby, Fermi National Accelerator Laboratory preprint, FERMILAB-Conf-92/121-E.

[4] A.Antonelli *et al.*, LNF **87-18(R)** (1987).
A.Antonelli *et al.*, Nucl. Inst. & Meth. (to be published).

[5] A.Perrone, G.Basti e A.Chiavoni, in Applications of Artificial Neural Networks III, Orlando FL, April 21-24, 1992, ed. S.K.Rogers (SPIE, The International Society for Optical Engineering, Washington, 1992), SPIE Proceedings Series 1711, p.470.
G.Basti and A.Perrone, in *Int. Symp. on Information Physics, as a part of Int. Symp. on Information Sciences (ISKIT '92), Iizuka, Fukuoka, July 12-15, 1992* (Kyushu Institute of Technology Press, Iizuka, 1992) p. 122.

[6] A. Gamba, L. Gamberini, G. Palmieri, R. Sanna, Nuovo Cimento Suppl. **20** (1961) 221.
See also [1], §13.1.

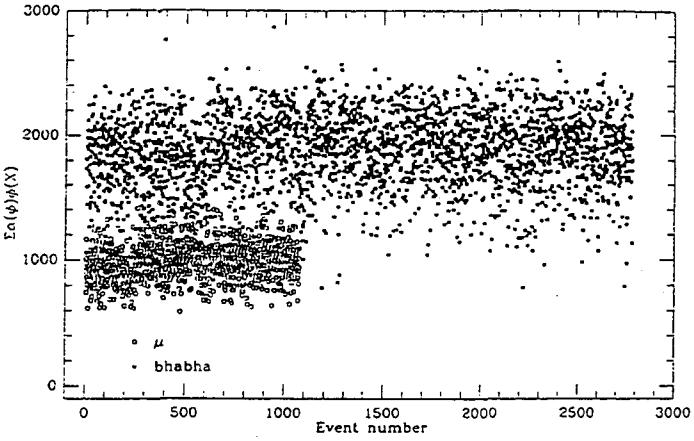


Fig. 2. Plotting of the net non-renormalized output (see text) for μ and bhabha tracks discrimination.

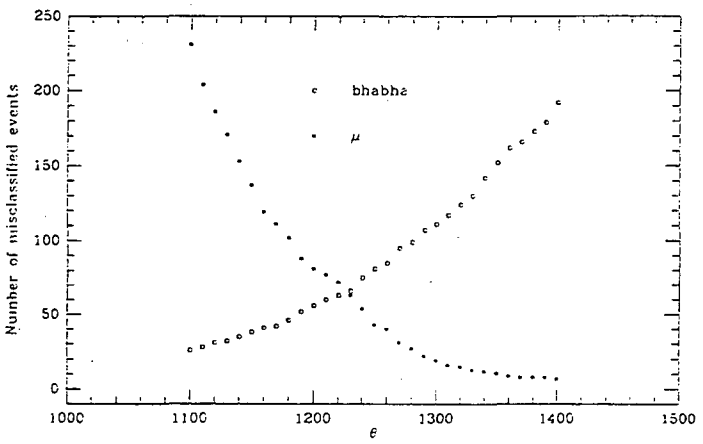


Fig. 3. Plotting of misclassified bhabha and μ events versus θ threshold.

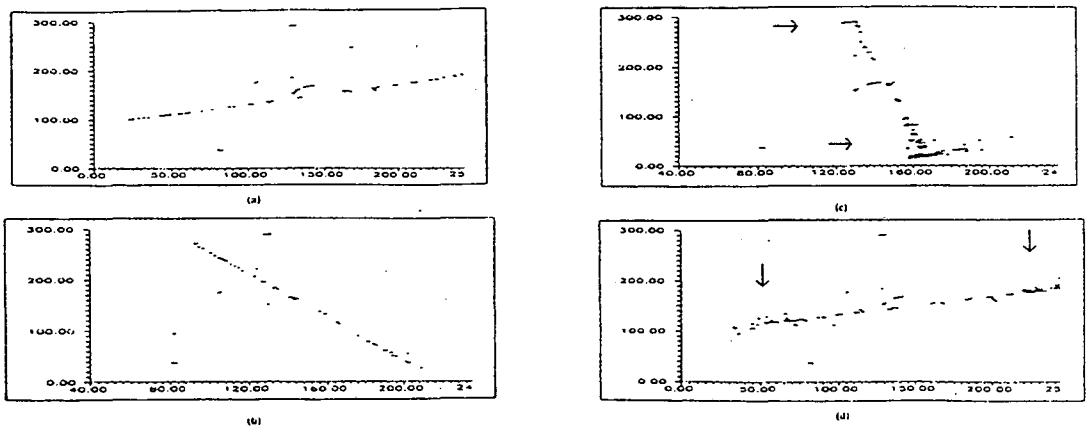


Fig. 1. Typical μ and bhabha (e^+e^-) tracks going through FENICE revelation chamber. Bhabha events can be recognized for their characteristic swarming effect at the extremities (see arrows in the figure).

D0 TRIGGER MONITORING *

N. Denisenko, S.Fuess, M.Narain, L.Paterno
(Fermi National Laboratory, Batavia, IL)
D.Cutts, J.Hoftun, T.Fahland
(Brown University, Providence, RI)
D.Edmunds, P.Laurens
(Michigan State University, East Lansing, MI)
N.Amos
(University of Michigan, Ann Arbor, MI)

Abstract

We describe the general architecture of the D0 trigger system with particular attention to the methods used to monitor the performance of the system components.

INTRODUCTION

The D0 Experiment [1,2] at the Fermilab Tevatron uses three levels of triggers to select several events per second from the over 10^5 interactions per second occurring within the detector.

Run conditions are established in each trigger component by a series of configuration scripts downloaded from a task on the central online host computer. Once a data taking run begins each element of the trigger system operates independently.

The first two levels of trigger decisions are made in hardware. Signals from the detector are fed to the trigger framework, which is controlled and monitored by tasks on a Trigger Control Computer.

A successful trigger at the hardware level initiates the readout of the entire event information into one node of a farm of VAXstation 4000/60 and VAXstation 4000/90 processors. At this point a software filter reduces the approximately 160 Hz trigger rate to 2-3 Hz to be transferred to a host system for logging. Monitoring processes running in the farm nodes accumulate information about each node's performance, timing and event counts.

To unify monitoring of the diverse components of the trigger system, we have also developed a Global Monitor program which executes on the online host system. A central server collects information from all of the component tasks and then distributes the information to multiple clients. These tasks interpret the trigger definition scripts in use during a run and allow graphical displays of the component elements.

Below we will describe the methods used to monitor each trigger level and the information provided by the monitoring processes about the data flow through the D0 Data Acquisition system.

*This work is supported in part by the US Department of Energy under Contract No. DE-AC02-76CH03000

HARDWARE TRIGGER MONITORING

The hardware trigger system [3] is programmed through the Trigger Control Computer (TCC), a VAXstation 4000/60 running the VAXELN real-time operating system. The trigger framework synchronizes the acquisition system and tags each event with a 32 bit mask indicating which trigger has fired. The trigger framework controls the acquisition system as 32 independent geographic sectors and can use any of the 32 triggers to initiate the digitization of any combination of geographic sectors. The Trigger Monitor Program (TRGMON) is available on the host system to monitor the programming and operation of the trigger framework and the performance of the acquisition system, including rates and deadtimes. TRGMON formats and presents information periodically collected from a monitoring pool server job running on the TCC. TRGMON organizes the information being monitored in a list of separate displays arranged by topic. Unexpected interruption of data flow or hardware problems are also detected by the trigger framework and sent to TRGMON for notification and alarm.

The monitoring pool server on the TCC is also used to provide information about the hardware trigger system to the Global Monitor server for its data flow displays.

SOFTWARE FILTER MONITORING

The software filtering farm [4,5,6] consists of 32 VAXstation 4000/60 and 16 VAXstation 4000/90 nodes. In addition, several special purpose nodes take part in the operation and monitoring of the farm. Two nodes, called the Supervisor and the Surveyor do most of the monitoring work for the farm. The farm nodes run under the VAXELN real-time operating system. The Supervisor selects one node for an event which fired the hardware trigger and allows the readout of the event into the selected node. Each filtering node runs FORTRAN filter code which consists of independent "tools". There are up to 128 possible filters defined by a "script" which sequentially invokes one or more of these "tools". For each fired hardware trigger a specified sequence of "scripts" run. The primary result of the filtering is a 128-bit filter mask, which is used to steer events to data streams and online monitoring tasks. If the event has any set bits in the filter mask it will be retained and passed on to the host computer for recording. Each node keeps counters and collects statistics on processing time and pass rates. This information is available on request through either the Surveyor node or directly to a display task running on a VMS host machine. The Supervisor accumulates information about the overall flow of events through the system and periodically sends this information to the monitoring process running in the Surveyor node. All this data is then available to remote displays and to the Global Monitor server running on a host computer. The remote displays show statistics related to the filtering system.

DATA FLOW MONITORING

Although each of the above mentioned trigger system components has a method to monitor its part of the data flow path, it is not sufficient. In order to obtain the overall picture of the D0 trigger performance, the Global

Monitor program was created. The program consists of two parts which run independently: GM_SERVER and GM_CLIENT. GM_SERVER reads the trigger and filter configuration files, created by the D0 Run Control program (COOR) and connects to the many permanently running data acquisition (DAQ) subsystems to get updated information about data flow status. GM_SERVER connects to the TCC monitoring tasks, several monitoring processes running on the SURVEYOR, to the luminosity server (D0_LUMIN, which records accelerator information) and to the alarm server (which distributes significant event information, including run activities). GM_SERVER periodically sends requests to DAQ processes, with a period unique to each process. Fig.1 shows the sources of GM_SERVER information together with the corresponding time between requests.

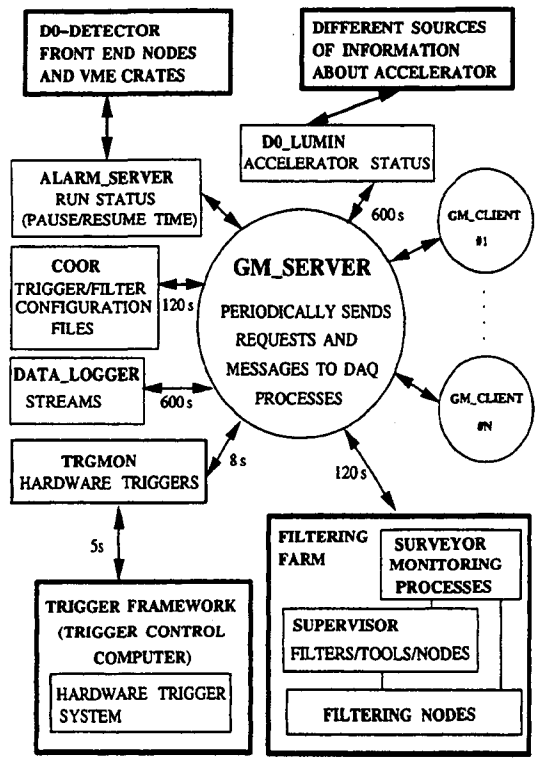


Fig.1

GM_SERVER runs as a detached process on a host computer and accepts connections from multiple interactive GM_CLIENTs. The main goal of the client program is to present information collected by the server. Each client can request specific sets of information from the server. Graphical displays are created using Motif and DI3000.

REFERENCES

1. S.Abachi et al., "The D0 Detector", Nucl.Inst. and Meth. A338(1994) pp.185-253
2. J.F. Bartlett et al., "Control and Monitoring Of The D0 Detector", Submitted to Nucl.Instr.and Meth.
3. M. Abolins, D. Edmunds, P. Laurens, J. Linnermann, B.Pi, "A High Luminosity Trigger Design For The Tevatron Collider Experiment in D0", IEEE Transactions on Nuclear Science, Vol.37, No.1, 1989, pp.384-389.
4. D. Cutts et al., "Data Acquisition at D0", Comput. Phys. Commun., 57 (1989) pp. 339-342.
5. D. Cutts et al., "Operation Of The D0 Data Acquisition System", Proceedings Of The International Conference On Computing in High Energy Physics '92 ", pp.262-264.
6. J. Linnemann, "Triggering The D0 Experiment", Proceedings of the American Physical Society Division of Particles and Fields Conference, Fermilab, Nov, 1992, pp. 1641-1643.

DATA ACQUISITION AND CONTROL SYSTEM OF THE COSMIC RAY EXPERIMENT KASCADE

The KASCADE Collaboration
presented by
H.J. Gils
Kernforschungszentrum Karlsruhe, Institut für Kernphysik
P.O. Box 3640, 76021 Karlsruhe, Germany

Abstract

The data acquisition and control system of the cosmic ray experiment KASCADE is based on a network of local front-end systems and a central multi-processor array, both using transputer processors. The hardware architecture, communication- and application-software is described.

INTRODUCTION

KASCADE (**KA**rlsruhe Shower Core and **A**rray **DE**etecor) is a large detector system which is being built on the site of the Karlsruhe Nuclear Research Centre to study extensive air showers initiated by high energy cosmic ray particles in the atmosphere. The main aim of the experiment is to obtain information about the chemical composition of primary cosmic rays in the energy range 300 TeV to some 10 PeV where little if anything is known at present. In addition the arrangement will be able to identify point sources in the quoted energy range.

The experiment consists of three main components, an array of about 250 detector stations distributed on a quadratic grid of 200m edge length, a compact central detector of 320 m^2 size, and a 50 m long and 5 m wide underground detector tunnel located close to the central detector (see Fig. 1).

The detector array registers the numbers of both electrons and muons. With the calorimetric central detector the number and energy of hadrons and the number of high energy muons in the shower core are determined. In the well shielded underground tunnel tracking detectors are installed for identifying muons in the vicinity of the shower core where the other muon detectors are blind owing to punch through of high energy electrons.

DETECTOR ARRAY

The detector array is composed of 16 independently working "clusters" consisting of 16 detector stations each and a control station placed in the centre of each of them. The streamertube tracking detectors in the muon tunnel are treated like a 17^{th} and 18^{th} cluster. Each cluster is controlled by a twin VME system containing the front-end electronics, trigger unit, clock and other devices as shown in Fig. 2. The VME crates are each controlled by a specially developed VME-controller with T800 transputer processor. An extensive air shower hitting a clu-

ster is recognized by a three-fold signal height and multiplicity trigger logic.

The trigger information is transmitted to the central processor system (see below) by software, which then starts readout of all other clusters.

CENTRAL DETECTOR

The central detector consists of a hadron sampling calorimeter with 8 active layers and two layers of MWPC for muon counting. The signals from 40960 identical analog channels (liquid ionisation chambers) of the hadron calorimeter are processed and finally collected in 5 parallel front-end processors as scetched in Fig. 3. A separate trigger plane of about 450 scintillation detectors provides trigger signals for hadrons and muons as well, where the latter is also used to read out the MWPC muon chambers, when no hadrons are detected in the calorimeter.

CENTRAL MULTI-PROCESSOR SYSTEM

The central multi-processor system is also built up from specially developed VME-modules (LIM = Link multiplexer) housing up to four T800 processors for application and one T425 processor for setting up the link configuration and booting purposes. The processors of these modules are interconnected to each other and to the front-end systems as drawn in Fig. 4. Fibre optics transmission is used for the long distances to the array control stations.

Communication on the multi-processor system uses the INMOS virtual configurer, whereas the front-end processors communicate with the central network by a user-specific protocol. Each instruction of this protocol is characterised by a specific tag number. This concept enables to reset, switch off, and reboot individual front-end systems e.g. for maintenance purposes without stopping the whole acquisition system.

In the local systems as well as in the central processor array different layers of software and process abstraction are implemented partly according to the csp-model. The on-line data preanalysis for determination of the size, core position, and direction of incidence of the detected extensive air showers uses neural network and fuzzy logic algorithms.

ACKNOWLEDGEMENTS

Following collaborators contributed to this work:

K. Bekk, D. Bormann, M. Brendle[1], J. Engler, H. Keim, H.O. Klages, H.W. Klein, J. Knapp[2], M. Kretschmer, H. Leich[3], D. Manger, H.J. Mathes, U. Mayer[3], W. Miller[4], D. Pröhl[5], U. Raidt[1], K.D. Rusch, H. Schieler[2], G. Schmalz, A. Schulte, U. Schwendicke[3], J. Wentz, P. Wegner[3], D. Wochele, J. Wochele, A. Wolf[5], J. Zabierowski[4], and S. Zagromski

[1]Universität Tübingen, Germany
[2]Universität Karlsruhe, Germany
[3]DESY-IfH Zeuthen, Germany
[4]INS Lodz, Poland
[5]Forschungszentrum Rossendorf, Germany

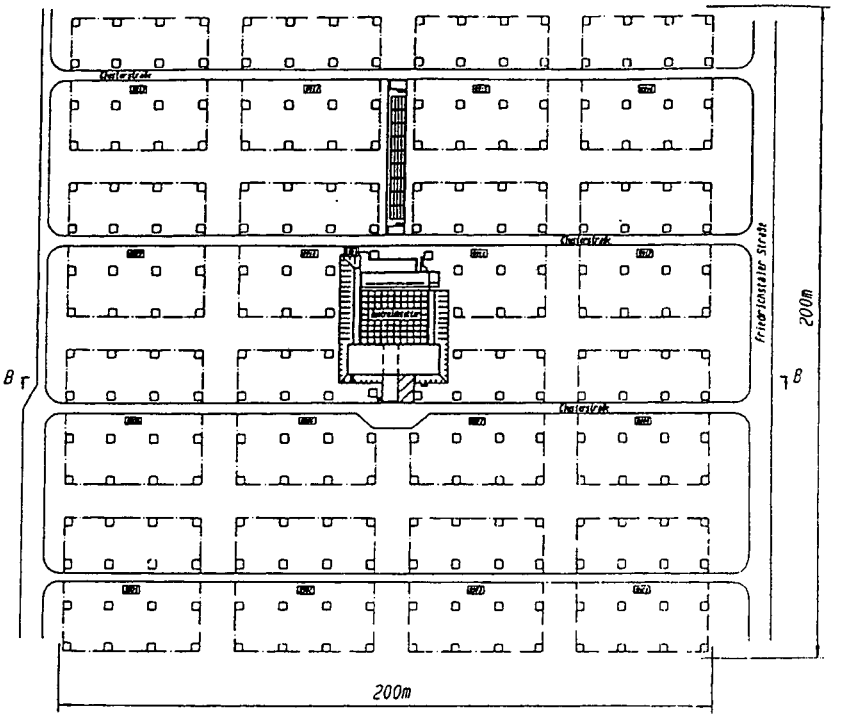


Fig. 1:
Layout of the KASCADE experiment: small squares represent detector stations for e^-/γ- and μ-components, small rectangles represent electronic containers supplying clusters of 16 stations each. Central detector and muon tunnel in the upper part are obvious.

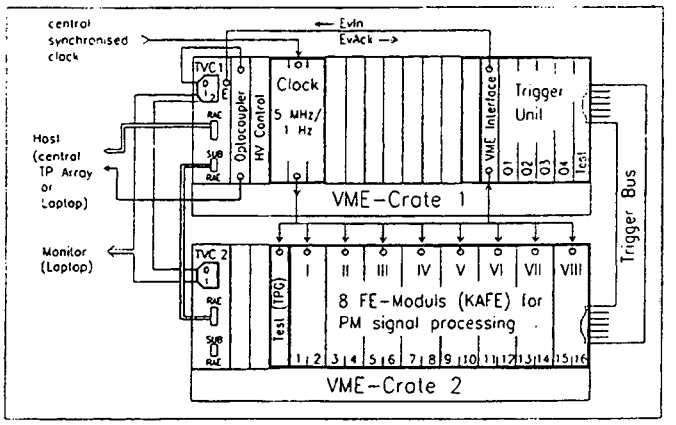


Fig. 2:
Array front-end electronics with TVC (Transputer-based VME controller).

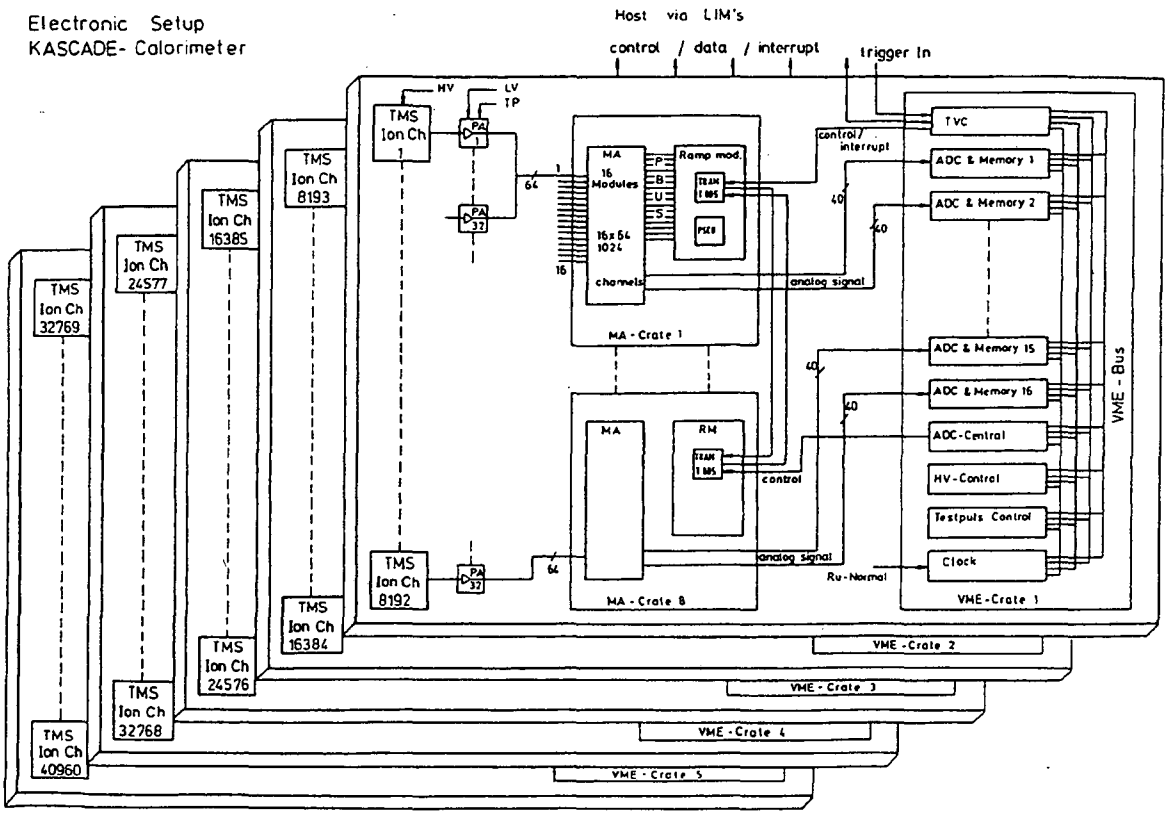


Fig. 3:
Signal processing in the calorimeter.

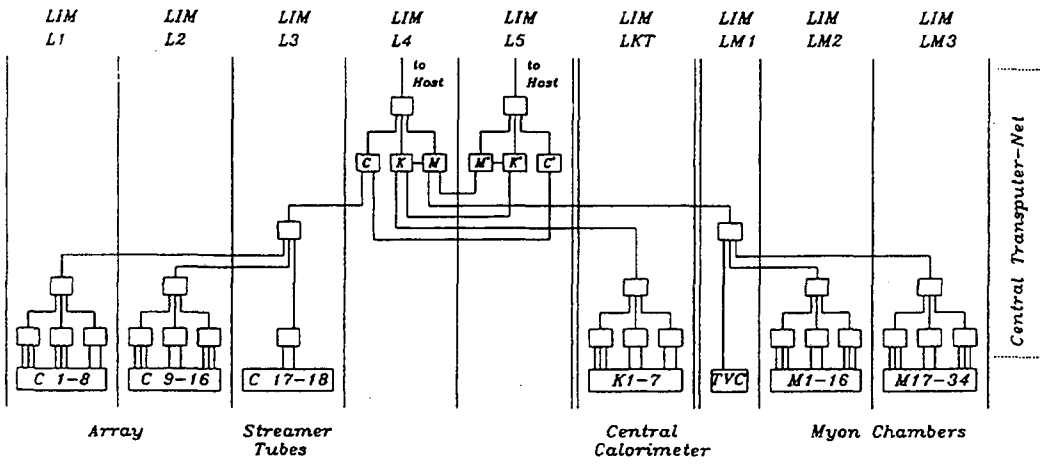


Fig. 4:
Central multi-processor system for the event-builder.

The RPC trigger system for the L3 Forward Backward Muon Detector

A.Aloisio, M.G.Alviggi, E.Brambilla, G.Carlino, N.Cavallo, R.De Asmundis, V.Innocente[1], S.Lanzano, L.Lista, P.Paolucci, P.Parascandolo, G.Passeggio, D.Piccolo, S.Patricelli, C.Sciacca, V.Soulimov[2], N.Zaitsev[2]

INFN Sezione di Napoli and Università degli Studi di Napoli "Federico II", Dipartimento di Scienze Fisiche

[1] Now at CERN

[2] On leave of absence from Nuclear Physics Institute, Gatchina, Russia

Abstract

We describe the trigger system for the L3 Forward Backward (F/B) Muon Detector which make use of Resistive Plate Counters. Using 192 double gas gap chambers covering an area of more than 300 m^2, this system is the largest built up to now and its operation will constitute an important test of industrially produced detectors of this kind.

Introduction

In view of the approved LEP program for the energy upgrade of the machine to $E_{c.m.s.} \approx 190$ GeV, which will give the physics opportunity to test the Standard Model by W^+W^- production as well as to extend the search for a Higgs boson up to a mass of 90 GeV, the L3 experiment has extended the polar angle coverage for muon detection lowering the minimum detection angle from 43° to 22° both in the forward and backward regions.

For each of this regions, muon measurement is achieved by the addition of 3 planes of precision drift chambers (FI,FM,FO) and by the toroidal magnetization of the doors of the existing solenoidal magnet (Fig.1).

In the region $43°>\theta>36°$ muons are measured by the MI and MM chambers of the barrel system and the FI chamber located inside the magnet door ; the trigger is the one used for the barrel system. In the region $36°>\theta>22°$ muons are momentum analysed by the toroidal spectrometer formed by the FI,FM and FO chambers and muon trigger in this region is provided by separate fast detectors, also essential for cosmic rays rejection since no scintillation counters are present in F/B regions.

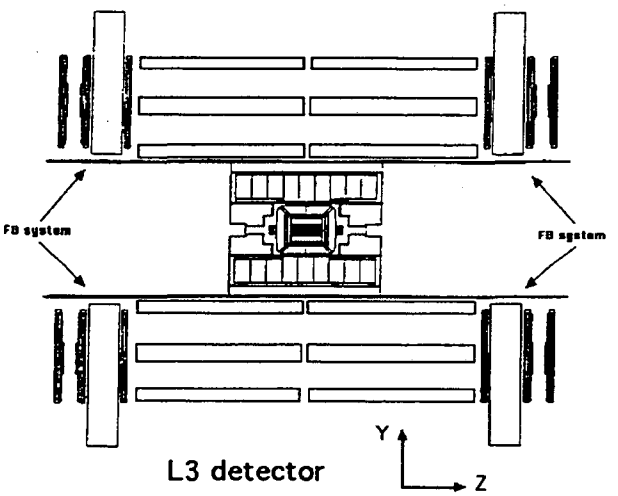


Fig. 1 - The L3 detector

Resistive Plate Counters (RPC)

Resistive Plate Counters[2] have been chosen for their fast time response, relative low cost and possibility of large scale industrial production. The sensitive volume of these detectors is a 2 mm thick gas layer operated under an electric field of ≈ 4kV/mm. Free electrons generated by ionizing particles induce an avalanche -> streamer process and resistive high-voltage electrodes, transparent to the pulses originated in the gas volume, allow a capacitive readout through external pickup electrodes. We use two counters mounted in the same mechanical structure with independent HV but read by a single pickup plane of aluminum strips 29 mm wide separated by 2 mm. The induced charge is ≈ 100pC and the pulse has a rise time of ≈ 2 ns and a FWHM of ≈ 10 ns.

In the L3 F/B muon system, each chamber of the FM and FO planes is equipped with one layer of double gas gap RPC. The two drift chambers are joined by a truss structure and the RPCs are located on the inner surfaces of the chambers. Readout strips are perpendicular to the octant center line; there are 192 RPCs for a total of 6144 readout strips. The resulting total area of bigap RPCs is > 300 m^2.

The trigger system

RPCs will provide a Level-1 muon trigger in the F/B region by searching for particles traversing the 1 m thick magnetized door and pointing to the interaction vertex.

A muon generated at the interaction point and which traverses the magnet door will move on a straight line in the space between the two RPC planes. The slope of its trajectory depends on the polar angle θ at the origin, charge sign and muon momentum. We can therefore associate to each readout strip in the inner plane (FM) a limited number of strips in the outer plane (FO). In a 96 * 96 coincidence matrix (Trigger matrix) , muons coming from the interaction point will populate a *road* whose *width* is dependent on their momentum.

Readout electronics

Induced pulses on readout strips are discriminated and amplified by fast transistors whose output drives directly TTL electronics. The front-end boards are mounted on the detectors.

The signal from each of the 96 strips in a FM (FO) layer is then ORed with the signal from the same strip of the contiguous layer of the same octant : each octant provides 96 signals from the strips in the FM plane and 96 from the FO plane. These signals are sent to a Zero suppressor and Encoder Module[3] strobed by the beam pickup. The output of this module consists of n 8 bits words , one for each hit strip, containing encoded information of its number and plane and sent to the following module via a dedicated bus.

Zero suppression of the input data and transmission of output data to the subsequent module are done in parallel and the total processing time is less than 900 ns. There are 32 of such modules, two for each of the sixteen octants.

Track finder and trigger generator

The two Zero suppressor and Encoder Modules for FM and FO planes of the same octant transmit their output to a Track Finder and Trigger Generator Module (Fig.2). This module is responsible for detecting a coincidence between two strips in the two planes which is included in regions (e.g. *roads*) of the Trigger Matrix which are user programmable. Sixteen of such modules are connected to a VME backplane and process data in parallel for all octants.

Each cell of the Trigger Matrix is mapped into one of 96 * 96 locations of a 2 bits RAM. Roads for muons coming from the interaction vertex can be programmed by filling the appropriate locations of the RAM and the use of two bits allows the definition of two different *roads* in the same matrix, corresponding to different momentum cuts.

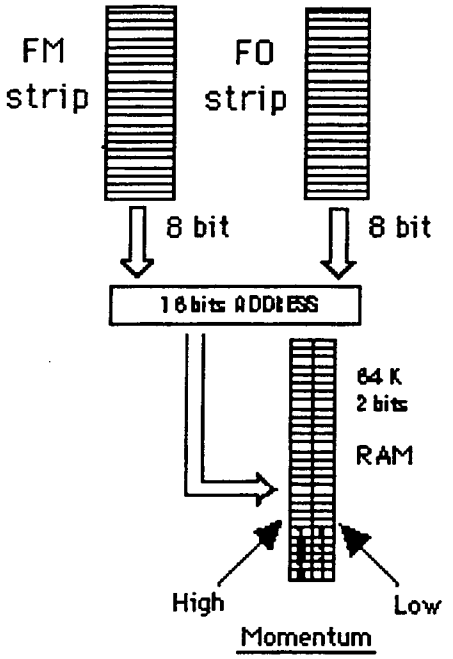


Fig. 2 - Track finder scheme

Track finding starts when all data have been received from the Zero Suppressor and Encoders : if n and m are the numbers of hit strips respectively on the FM and FO planes, the system will scan through the n x m possible search addresses which can be formed and a *track found* signal will be generated when coincidence is found in the pre-defined range. Time required to scan all search addresses (n x m x 100 ns) depends on the number of hit strips and if this is greater than the maximum allowed for the Level-1 trigger generation, a timeout condition is set and the corresponding octant is set to *track found*. One output of this module is used in coincidence with small angle trigger.

High and low momentum triggers for all octants are ORed to generate the Level-1 trigger of the FB muon detector.

Information with the encoded numbers of hit strips together with two bits trigger words will be transferred to the Level-3 trigger processor via the Fastbus data readout system when an event is accepted by the complete L3 Level-1 trigger. This informations will also be written in the event raw data on tape when the Level-3 trigger will finally accept the event.

References

[1] L3 Forward-Backward Muon Detector. Dec. 3 1991 - L3 Internal Note.

[2] R. Santonico and R. Cardarelli - NIM 187 (1981) 377.
R. Cardarelli et al. : NIM A263 (1988) 20.

[3] A. Aloisio et al. : RT93 (1993) 71

Design and Implementation of a VME Based Event Building Protocol for the New ALEPH Data Acquisition System

Josep A. Perlas
CERN, CH-1211 Geneva 23, Switzerland

Abstract

This paper describes and discusses the design and implementation of the protocol for event building in the new ALEPH VME readout system. Special emphasis is given to the consequences of dealing with a large and complex multi-crate system, a feature that is not incorporated in the specification of the VME standard.

Introduction

The new ALEPH readout system contains 20 VME crates. The original system was implemented using Fastbus [1] and has recently been replaced by commercial VME hardware, while keeping Fastbus in the front-end digitizers [2]. The readout protocol had to be revised and a new implementation made.

Data generated in each detector component has to be collected and assembled correctly. The readout protocol guarantees that this is done properly. The protocol is implemented as a software library, the *readout library*, that ensures that it is strictly obeyed by detecting protocol violations during event assembly.

To implement a multi-crate VME based protocol to perform event building for ALEPH, different possibilities have been considered. This includes protocols based on pure VMEbus and VICbus resources and also protocols based on processor-specific resources. The advantages and inconveniences of the considered cases will be discussed and the final choice justified and described.

Abstract protocol overview

In the old Fastbus readout system, the *abstract* event building protocol was as follows. The readout function was separated into two independent and asynchronous tasks: the sender and the receiver, which ran in different processors in the different stages of the event building "tree". The sender was responsible for detecting the presence of a new event and asserting a request indicating data available. The receiver read the different pieces of data according to the requests received from its sources until the whole event had been assembled.

The complete description of the abstract data transfer protocol, together with the Finite State Machine model used to handle the different phases of it, can be found elsewhere [3,4].

A protocol for a VME multi-crate system

After having decided to move to a new readout hardware, a revision of the old readout protocol had to be made. We started by looking at the function-

alities that the new hardware could provide, classifying the considered possibilities into two main groups: protocols based on resources of pure external busses (VMEbus and VICbus) and protocols based on processor-specific resources[1].

In the first group we can shortly mention the following:

- *Using VME interrupts.* The sources in a VME crate generate a VME interrupt which is then translated to a VIC interrupt and finally to a new VME interrupt in the destination crate. This needs a VME interrupt handler to identify the source. In addition, the identification information of the different sources is not easy to handle.

- *Using VIC mailboxes.* A source in a VME crate writes in one of the mailboxes of the VIC module its identification information, generating a VIC interrupt. This generates a VME interrupt in the destination crate. It has the limitation of eight sources per crate and needs also an interrupt handler.

In the second one we can emphasize:

- *Using FIC mailboxes.* A source writes directly its identification information into the receiver mailbox RAM, generating a CPU interrupt. This allows for a big number of requests but it doesn't provide enough freedom for the source identification information data structure.

- *Using FIC fifos.* The sources also write directly into the receiver using the VIC module in *transparent access* but in this case they write into the fifo port of the FIC. This generates a CPU interrupt in the receiver who will then read the source identification information in a special memory accessible through VME.

Clearly, a protocol based on the first group offers the advantage of being processor-independent, which means that it can be replaced by any other VME processor without changing anything in the readout configuration. Nevertheless, the poor resources offered by VME (e.g. the difficulties in handling the identification of the different sources) means the protocol would not be robust, which was considered essential in the design.

On the other hand, a protocol based on the second group, although it is processor-dependent, offers more robustness since it is 'simpler' and doesn't rely on additional software to support, for instance, VME interrupt handling.

At this phase of the design, we didn't consider it as absolutely necessary to follow the specifications given by [4]. Nevertheless, since we had a positive experience running with this approach, we wanted the new design not to be very different from the old one, assuming that a correct implementation could be found. For instance, although a *push* protocol[2] is also a valid one, it introduces completely new schemes of work with the corresponding new problems and our lack of experience. Consequently, rather than investigating further the *push* protocols, we decided to

[1] See [5,6] for an explanation of the technical concepts used in the following discussion.

[2] In a *push* protocol it is the source who writes its data into the receiver instead of the receiver reading the source.

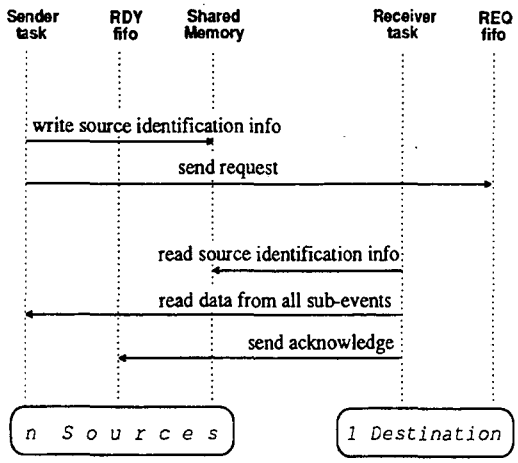


Figure1: 'Event trace' diagram for the ALEPH readout protocol.

chose the old and safe *pull* protocol, as described in the previous section.

Taking into account its anticipated robustness and simplicity, and given the fact that we had decided to chose a FIC8234 as processor after an evaluation performed on several processor boards [2], we finally decided to use the *"FIC fifos"* option to implement our readout protocol. Furthermore, the *'enable request'* phase as specified in [4] was considered not necessary and has been removed from the protocol.

Description of the implementation

In the design of an event building protocol, two important aspects have to be considered: how the sources notify the receiver of available data and how the receiver identifies the different sources.

For the first function, an interprocessor interrupt facility has been used. This is based on the FIC8234 *fifos* [5], a powerful resource of this processor. In this approach, the source has simply to write a longword (with a pre-determined format) into the fifo port of the receiver. This is done by writing to the local VME bus at a certain address such that if the VIC8251F module [6] is used in *transparent access*, this generates a remote VME cycle through the VIC bus.

To implement this facility, an OS9 device driver and library have been written. Its purpose is to buffer the fifo messages and dispatch them to the requesting processes. Furthermore, since our data acquisition system supports the concept of *partitioning* [1], the driver uses different *channels* to dispatch the fifo messages. A channel is a number used by the driver to create a link between two communicating tasks. In this way, the library implements multi-user access to any fifo.

A limitation in the functionality of the fifo port has influenced the protocol implementation: it is not possible to perform DMA transfers into it. Consequently, our original idea of writing the source identification information (consisting on 4 longwords: ID number, event buffer offset, event length and trigger number) into the fifo port was discarded. Instead, we use a special memory (separated from the OS9 system memory and having a colour number) called *shared memory*, accessible through VME.

To implement the protocol, our first approach was to allocate three fifos in each processor of the readout tree. One to signal *Data Available* (called REQ fifo), another for *Data Ready* (RDY fifo) and a third one used to store the indices to the shared memory of the receiver (IDX fifo); this last one is not attached to the driver and it is merely used as a FIFO memory.

When a source has data available, it takes an index from the IDX fifo and

uses it as a pointer to the shared memory where it writes its identification information. Then it writes the index into the REQ fifo of its receiver to interrupt it and be read-out. Once all the sources are read, the receiver writes into the RDY fifo of its slaves who then free their event buffer and prepare for the next event.

After having exercised this implementation in the real experiment (see below), we decided to locate the shared memory in the sources instead of in the receiver and not to use the IDX fifo. The aim was to minimize the number of 'dead locks' in the VIC bus. See Fig. 1 for the corresponding 'event trace' diagram.

Tests and performance

Extensive tests in the laboratory have been performed to check the viability, reliability and performance of the protocol described in the previous section. In a first test a single crate was used, containing 1 receiver reading 3 sources. The sources generated small events (64 bytes) at a rate of about 240 Hz (these conditions were chosen to generate very frequent VME arbitrations). The test showed almost no error after more than 14 million events.

In a second test, two VME crates interconnected with a VIC bus were used: one housing the receiver and the other the 3 sources. This time the aim was to gain experience with the potential *dead lock* situations, expected as 'normal' errors when more than one bus is used. As expected, after having enabled the 'VME retry' mechanism we only found *Bus Errors* in some DMA transfers (when the receiver was reading a source) which were cured by a single software retry at the level of the readout library[3].

Although the tests done in the laboratory gave us some confidence that the protocol was working very nicely, the real experiment with the real readout configuration was the ultimate check. Preliminary tests performed with a small partition of the ALEPH detector containing 1 receiver and 6 sources showed immediately hardware imperfections in the VME Bus Requester module of the VME Master Interface of the FIC and also in its VME Slave Interface and in the Slot 1 Function provided by the CES VMDIS 8004 display module. All these problems were temporarily overcome by adding checks in all the VME actions performed by the readout library and retrying if a Bus Error was found.

After having solved the hardware problems, made more robust readout software and tuned the different parameters in the VME system (arbitration scheme, requester mode and bus timeout values), the event building was running well. The cosmic runs made during the commissioning period add another prove to this.

Conclusions

The upgrade of the ALEPH readout system from Fastbus to commercial VME hardware motivated a revision of the old event building protocol and led to a new implementation of it. Amongst several considered possibilities we have chosen one that adapts well to our needs while being robust and acceptably simple for a multi-crate VME system. The mechanisms used to achieve this have also been described. The extensive tests performed

[3]The BMA hardware provided with the FIC8234 to perform DMAs does *not* have the 'retry feature'.

in the laboratory and in the real experiment during the commissioning phase show that this design adequately fulfills the original requirements.

ACKNOWLEDGEMENTS

I would like to thank B. Jost, of the ALEPH online group, for many useful discussions.

REFERENCES

1. W. von Rüden, "The ALEPH data acquisition system", IEEE Trans. on Nucl. Science **Vol. 36, no. 5**, 1444-1448 (1989).
2. D. Botterill et al., "Report of the Study Group for a New ALEPH Readout Processor", ALEPH internal communication (1991).
3. J.A. Perlas et al., "The new implementation of the event building protocol for the ALEPH data acquisition system", IEEE Trans. on Nucl. Science (1993).
4. ALEPH Dataflow Group, "ALEPH Data Acquisition System Hardware Functional Specifications", ALEPH DATACQ note 85-21, CERN (1985).
5. Creative Electronic Systems S.A. (Geneva), "FIC8234 Dual 68040 Fast Intelligent Controller, User's Manual" (1993).
6. Creative Electronic Systems S.A. (Geneva), "VIC8251F VIC to VME Interface with Mirrored Memory, User's Manual" (1993).

Custom Solution for a Data Readout Architecture: a System Level Simulation.

A. Aloisio [1,2], F. Cevenini[1,2], D. della Volpe[1,2], L. Merola[1,2], P. Parascandolo[2]

[1] Università degli Studi di Napoli "Federico II", Dipartimento di Scienze Fisiche.
[2] INFN Sezione di Napoli.

Abstract

Behavioral simulations of data readout architecture based on VME and custom high speed buses show that it is suitable as data acquisition and event building system for high energy physics experiments.
This paper describes a reliable but simple auxiliary bus designed to afford asynchronous transactions up to 10 Mtransfers/s, sparse data scan operations and crate-level event building. An intercrate connection is also presented to accomplish system level event building and data concentration by means of synchronous transactions of rates up to 10 Mtranfers/s.
This architecture has been simulated using Verilog HDL. Preliminary performance estimates are presented and briefly discussed in view of system application in KLOE experiment at the DAΦNE Φ-factory in Frascati (Italy).

Introduction

In H.E.P. experiments, Data Acquisition systems use high performance parallel buses to gather data from the front-end electronics to event builder engine and on-line processors.

Software protocols and CPU driven operations may slow down the whole system even if state-of-the-art devices and communication channels are used.

We describe in this paper an architecture which relies on custom buses and hardware controllers in order to achieve high bandwidth without software penalties and CPU overheads.

System architecture

The proposed architecture is shown in Fig.1

Each chain contains a number of VME crates, which hosts a CPU board, acquisition modules and a Readout Controller (ROC).

Through VME backplane, the CPU can access both the Front-End modules and the ROC. This controller is the only master on a custom auxiliary bus (AUXbus) which connects all the Front-End modules.

The structure and the protocol of the AUXbus are optimized to meet the requirements of high speed data transfer. In our scheme the VME CPU initializes the whole system, runs diagnostic tests and, if necessary, samples events from time to time for performance monitoring of the system itself.

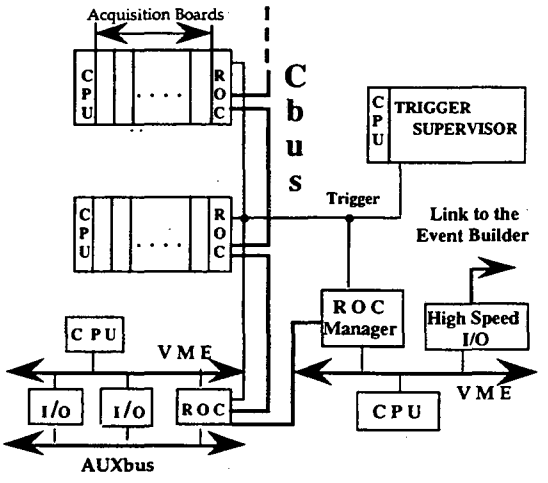


Fig. 1 - Read out Architecture

All these tasks may be afforded by a low-medium performance CPU in a standard software environment.

All the ROCs of a chain are connected through a single master cable-bus, called Chain-bus (Cbus), which is driven by a ROC Manager.

This last module is housed in a VME crate together with a high performance CPU and a high speed I/O board used to transfer data to the event builder.

Protocols overview

Each acquisition board is designed to respond to a trigger number request sending all the data associated to it onto the AUXbus. Acquisition results are automatically associated to the channel number inside the module.

The ROC drives the readout cycle on the AUXbus in two steps: broadcast and transfer.

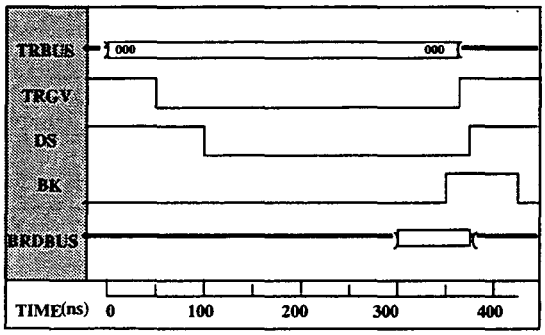


Fig. 2 - Broadcast cycle

During the broadcast cycle (Fig. 2), the ROC puts on the TRbus lines a trigger number forwarded asyncronously to the whole system by the trigger supervisor of the experiment. The boards with data associated to that trigger number assert a line on the BRDbus. The slowest board then validate this bus with the BK line.

As a result, the ROC records the BRDbus pattern by means of a single transaction synchronous to the BK assertion.

The broadcast cycle then proceeds with the identification of the modules to be read and the preparation of the table containing the addresses (sparse data scan).

In the following transfer cycle (Fig.3) the ROC addresses successively the modules to be read. Data transfer comes in blocks for each module using an asynchronous VME-like protocol.

An EOB line is used to flag in hardware the end of a block data transfer. It allows handling indefinite long block operations, overcoming the VME limitations. Data cycle pipelining is also provided to improve data transfer rates.

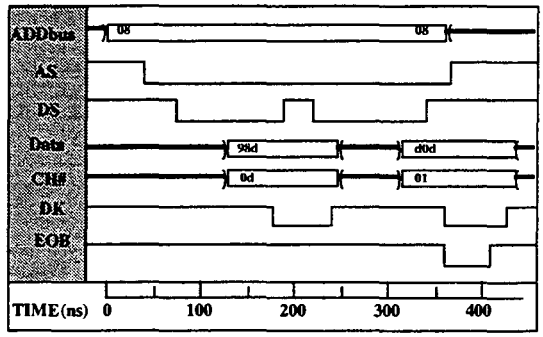


Fig 3 - Transfer cycle

Into the ROCs, data are organized in frames starting with a trigger number. Data words collected through the AUXbus from the acquisition modules follow. The frame is then terminated with a control word (End-Of-Frame). The frames are queued into a dual port FIFO buffer, accessible from the intercrate link.

The ROC Manager controls a chain of ROCs through the Cbus in a token-ring fashion.

Its readout cycle starts by presenting and validating a trigger number on the Cbus lines. Then a token is sent onto the bus.

The ROC nearest to the master catches the token and then start the synchronous transmission of the data associated to the given trigger number. The end of the data is flagged with the EOF control word. The token is then passed to the next ROC in the daisy-chain.

This procedure is repeated until all the ROCs of the chain have sent either the data related to the processed trigger number or an empty frame - if no data are available.

Traffic on AUXbuses and Cbus are fully decoupled by means of FIFO buffers to afford possible large change of trigger rate.

The data frame inside the ROC Manager is similar to that inside the ROC but also contains the crate address.

Simulations

Architecture and protocols presented above have been modeled and simulated at behavioral level using Verilog-HDL [1].

The timing of the transactions on both AUXbus and Cbus comes from the hypothesis of using off-the-shelf TTL components and from conservative criteria.

In our simulations, we considered the data acquisition layout proposed for the KLOE experiment [2] at the DAΦNE Φ-Factory in Frascati (Italy). The KLOE DAQ system [3] will rely on chains of 8 crates. One crate houses 16 acquisition boards with 32 channels each.

The simulation of the activities on AUXbus was done for an increasing number of words randomly spread into the crate.

Time the ROC needs to complete a readout cycle increases linearly with the amount of data to be transferred (Fig. 4).

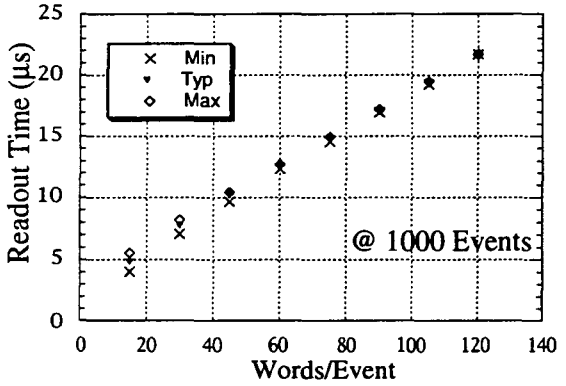


Fig. 4 - AUXbus simulated performances

The analysis of the whole chain (8 crates) has been carried out from 120 up to 480 words/event randomly spread into the system. For each occupancy 1000 events have been simulated.

The typical acquisition time at 250 words/event is about 70 µs as shown in Fig. 5.

Conclusions

The introduction of auxiliary buses in a DAQ scheme relaxes the requirements on the CPU and makes it possible to use VME as an additional channel for monitoring, diagnostics and on-line event sampling.

Data transmission in our proposal is driven by hardware processors through very reliable protocols, developed *ad hoc* to meet the requirements of a typical H.E.P. experiment.

A hardware implementation of the described architecture is under development in Naples in view of application in the KLOE DAQ system. The AUXbus controller is based on a XC3164 XILINX Gate Array [4] while the Cbus interface fits in a MACH210 AMD device [5].

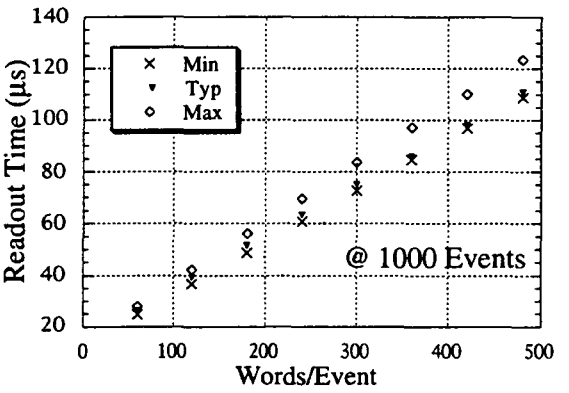


Fig. 5 - Cbus simulated performances

The AUXbus (64 lines) shares the VME/J2 backplane, the Cbus uses a 50 twisted pairs cable. Prototypes of both VME/AUXbus custom backplanes and Cbus cable segments have been already designed and successfully tested.

In KLOE experiment, at 10 KHz expected trigger rate the DAQ system will be able to afford a 15 MByte/s data traffic per chain - assuming 32-bit wide words and a maximum occupancy of 400 words/event.

References

[1] Cadence Design System Inc., VERILOG-XL Reference Manual, Version 1.6, 1991.

[2] The KLOE Collaboration, " KLOE A general purpose detector for DAΦNE", LNF - 92/019 (IR).

[3] A. Aloisio *et al.* , "Fast readout system for KLOE", KLOE Note # 88, Jan. 94.

[4] XILINX, The Programmable Gate Array Data Book, 1993.

[5] Advanced Micro Devices, MACH 1 and 2 Family Data Book, 1993.

Beam Steering: A Test Bench for Generic Algorithms in Accelerator Controls

B. Autin, M. Arruat, F. di Maio, M. Martini
PS Division
CERN 1211 Geneva 23 Switzerland

Abstract

The operation of a complex accelerator system such as LEP or the future LHC at CERN demands automatic and standard controls to make it easy to use and reliable in all circumstances. A class of beam manipulations is the steering in the various machines and transfer channels. An algorithm has been devised to satisfy the required condition of genericity. It is based on a least squares method and yields a correction system which is not necessarily well conditioned in a usual operational environment where correctors and monitors may be either missing or redundant.

The algorithm has been coded in *Mathematica* and implemented in the CERN PS control system. It is called through the *MathLink* protocol by an application program linked to standard beam position measurement software. With this technique, the development of the algorithm and its linkage to the control system are fully de-coupled. The application will become generic as soon as the magnetic and optical parameters will be loaded in data bases.

1. INTRODUCTION

The main two parameters of a collider are its energy and its luminosity. If it is usually straightforward to reach the design energy, it is much more delicate to obtain the highest luminosity. This requires permanent development work to correct the various imperfections, a deep understanding of the beam behaviour and even the design of new schemes which overcome basic limitations. Besides the machine development, it is crucial that the operation of the machine have a high level of reliability and that all the setting-up be as fast as possible so as to maximise the time for particle physics. In both development and operation activities, the man machine interface through controls has to take advantage of all the progress in software technology to be as efficient as possible and three criteria can be outlined: universality, simplicity and connectivity.

(i) The code developed in the design and development stages should address a large class of problems and be implemented in the controls without having to be re-written for the operation.

(ii) The programming language should be close to the physics concepts. A code is not appraised by its number of lines but rather by its compactness and its clarity. From this standpoint, advanced symbolic languages using functional programming, pattern recognition and list processing are far superior to traditional numeric codes. Text processing and graphics facilities are indispensable tools in the elaboration and test of the code.

(iii) The algorithms are just one aspect of a control system. The other elements are on one hand a relational data base which contains all the relevant information about the components of the machines and on the other hand the communications with the instruments through equipment modules and the user through graphics interfaces. The programming language must thus have a linkage protocol which connects it to the rest of the controls world which has its own standards (UNIX, C, etc.).

An approach to controls fulfilling those requirements has been tested for the steering of a beam in a transfer line (TT2) situated between the PS and SPS machines at CERN. It will be analysed in each of its aspects: machine development with diagnosis of an alignment error, correction algorithm written with *Mathematica* [1] which has list processing, functional programming, graphics facilities and a linkage protocol available and integration of the algorithm into the PS control system. A generalisation of the method to beam steering in all CERN machines is under development and its general features are presented.

2. BEAM STEERING

2.1. Operational environment

The method devised for beam steering in transfer channels and closed orbit correction in synchrotrons [2] is applicable to arbitrary distributions of beam position monitors and corrector magnets. The devices used for beam position measurements in the CERN PS complex are mainly secondary emission monitors (SEM-grids) and electrostatic pick-up stations. The correction algorithm solves a linear system iteratively using a least squares method. Its purpose is indeed not to find an exact solution which is both irrelevant and dangerous when the various experimental errors are taken into account but to minimise a residual vector; in brief, the algorithm is a minimise and not a solver. The method has been widely applied because it is robust, flexible, fast and has the very appreciable practical advantage to converge with a small number of correctors.

The dipole corrector fields are provided either by steering dipole magnets or by moving quadrupole magnets transversally, yielding the angular kick

$$\delta\varphi_i = \frac{l\Delta B}{B\rho} \qquad (1)$$

$$\delta\varphi_i = Klu \qquad (2)$$

where l is the corrector length, u the transverse displacement of the quadrupole magnet, K the quadrupole strength, $B\rho$ the beam rigidity, and ΔB the dipole field. The corrections calculated in milliradian are converted either in Ampere for dipole power supply settings or millimetre for quadrupoles displacements.

2.2. *Correction algorithm*

The matrix A of the linear system is made of as many columns as possible correctors. Each component of a column is the position of the particle trajectory when the corrector has a unit excitation, all the other correctors being set to 0. The errors to be corrected are collected in a vector U. The unknown corrections are the components of a vector $\delta\varphi$ and calculated in such a way that the norm of the residual vector $A\delta\varphi + U$ is minimised in an iterative manner. That correction magnet which yields the lowest residual r.m.s. distortion at monitor positions is first selected. Then, the residual distortion is re-analysed and the next best magnet selected. Keeping all correctors from the previous iterations but recalculating their strengths, the method proceeds until the residual r.m.s. distortion is compatible with measurement errors.

The trajectory deviations at the i-th monitor due to a unit kick at the j-th thin lens corrector for a transfer channel and a ring are

$$a_{ij} = \sqrt{\beta_i \beta_j}\,\sin(\mu_i - \mu_j) \qquad (3)$$

$$a_{ij} = \frac{\sqrt{\beta_i \beta_j}}{2\sin\pi Q}\cos(|\mu_i - \mu_j| - \pi Q) \qquad (4)$$

respectively where β and μ are related to the amplitude and phase of the betatron oscillation and Q is the machine tune [3]. In matrix notation $A = (a_{ij})_{j=1..m}^{i=1..n}$ denotes the full correction matrix.

The various steps of the calculation at the k-th iteration are:

(i) Compute the corrector strengths from the measured trajectory distortion as

$$\Delta\varphi^{(k)} = -(A_i^{(k)T} A_i^{(k)})^{-1} A_i^{(k)T} U \qquad (5)$$

for $i = 1...m$, with $i \neq i_1, ... i \neq i_{k-1}$, where the matrix $A_i^{(k)}$ is formed by concatenation of $A_{i_{k-1}}^{(k-1)}$ with the i-th column of A ($A_i^{(1)}$ being the i-th column of A), and $\Delta\varphi^{(k)} = (\delta\varphi_i)_{i\in\{i_1..i_k\}}$ is the vector kick. The letter T denotes the transpose of a matrix.

(ii) Compute the norm of the residual vector $R_i^{(k)}$ as

$$|R_i^{(k)}|^2 = U^T U + \Delta\varphi_i^{(k)T} A_i^{(k)T} U \qquad (6)$$

(iii) The k-th best corrector is the one which minimises the norm of the m-k+1 residual vectors

$$|R_{i_k}^{(k)}| = \min_i(|R_i^{(k)}|) \qquad (7)$$

The re-distribution of the columns of the correction matrix is well suited to list processing. In addition, the generalised inverse $(A^T A)^{-1} A$ is a built-in function of *Mathematica*. As a result, the total code does not exceed one hundred lines and has been written and tested in a few days. It is called through the *MathLink* protocol by a control application written in C and linked to the standard emittance measurement software installed in the PS control system [5]. In the case of very large machines, the computing time can be substantially reduced by resorting to Householder's transformations [4] which have not yet been implemented.

3. PROTOTYPE FOR A BEAM CORRECTION SYSTEM

3.1. *PS to SPS transport line trajectory correction*

With the new and more stringent requirements of beam emittance for LHC, the beam profile has to be measured with an enhanced precision. This is done in the transfer line TT2 using high resolution SEM-grid monitors (0.35 mm wire step size, appropriate for minimum transverse beam size of, say 3 mm, for the LHC). The measurement necessitates a good centring of the beams which is carried out first by means of low resolution SEM-grid devices (2.5 mm ribbon step size), then by a finer beam centring using the high resolution SEM-grid devices. The correction algorithm did not converge at the beginning of the tests in the vertical plane and an absence of convergence is the symptom of anomalous measurement errors. The alignment of the SEM-grids on the axis of the adjacent quadrupoles has thus been checked and it turned out that one grid had indeed an excessive offset which was eventually taken into account so that a complete correction could be done.

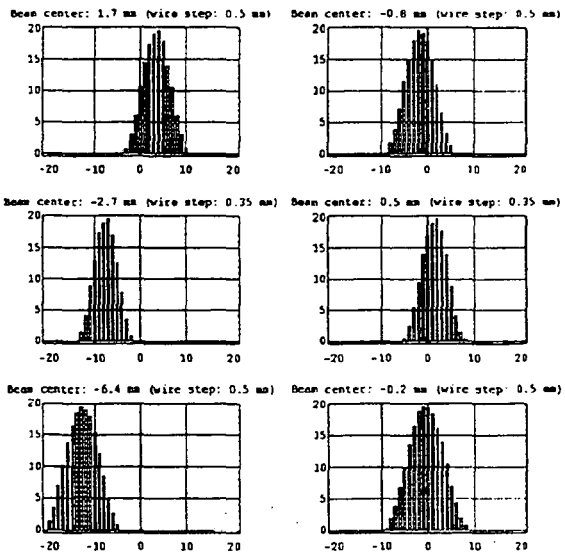


Fig. 1: SEM-grid signals before (left) and after (right) centring.

The beam is centred on the monitors using two steering dipoles selected out of a set of three; the r.m.s. trajectory

distortion and the peak-to-peak value decreased from 2.4 mm to 0.3 mm and from 8.1 mm to 1.3 mm in the horizontal (Fig. 1) and vertical plane respectively.

3.2. AA and AC closed orbit correction

Corrections of closed orbits are important to increase the machine aperture. The vertical closed orbits have been corrected in the CERN Antiproton Collector (AC) and Antiproton Accumulator (AA) rings which are equipped with the same number of correctors and monitors: 28 in AC and 12 in AA. That machines have no on line control of the closed orbit and the corrections performed off-line took less than one hour.

In both rings the correction has been accomplished using 3 correctors. In AA, the r.m.s. distortion and the peak-to-peak value decreased from 0.7 mm to 0.2 mm and from 6.6 mm to 2.2 mm respectively and the vertical machine acceptance increased after correction from 15 π mm.mrad to 24 π mm.mrad. In AC, the r.m.s. distortion and the peak-to-peak value were reduced by a factor 2 thus enlarging the transverse acceptance from 183 π mm.mrad to 196 π mm.mrad (Fig. 2).

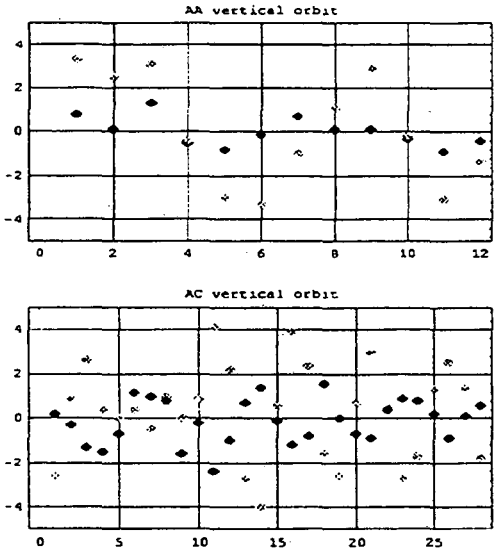


Fig. 2: Closed orbits before (grey dots) and after (black dots) correction.

3.3 On-line operation of the trajectory correction prototype

The trajectory correction in TT2 has been implemented in a prototype application program, using the CERN PS control system. This program has been extensively used on-line, during machine development sessions dedicated to LHC type beam (Fig. 3).

The correction algorithm implemented with *Mathematica*, which was developed and tested off-line, has been directly integrated into the program with only minimal interface adaptations. The beam trajectory is acquired through SEM-grids instruments and the corrections are applied through the power-supplies of the horizontal and vertical dipoles.

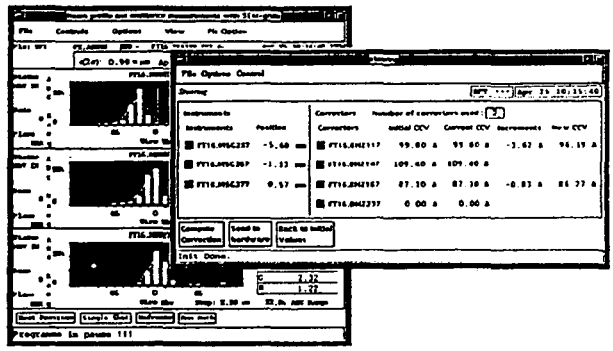


Fig. 3: Display of the interactive windows used for the beam steering programme.

In the control system, the correction function requires three distinct modules which are inter-connected (Fig. 4). An instrumentation application produces the trajectory, the computation module calculates the corrections to be applied and the steering application creates the communications between the instrumentation application, the computation module, the correction equipment and the operator.

Beam trajectories are acquired with SEM-grid equipment. Such an equipment requires a dedicated program which implements the control of the devices as well as the treatment and presentation of the acquisitions. The control of the devices includes commands like: "Put SEM-grid in the beam", "Select gain", etc. The treatment of the acquisitions consists of filtering, interpolation and r.m.s. calculation.

The SEM-grid application is a *X-Window/Motif* program. Through the *X-Window* protocol, it exports beam trajectories to the steering application synchronised with machine events or on operator's request.

The computation module is a *Mathematica* session executing the trajectory correction algorithm. The *Mathematica* session is activated by means of the *MathLink* protocol during the software initialisation. The correction functions are loaded from a file which can be updated without having any program to be re-compiled nor re-linked. Should the syntax of the functions remain unchanged, corrections or extensions of the algorithm can therefore be very quickly implemented.

The correction functions are executed through the *MathLink* protocol. On operation workstation (DEC-MIPS), this does not introduce any user-perceptible delay (i.e. still less than 2 seconds between clicking on the button and having results displayed).

The steering application is a *X-Window/Motif* program which integrates the different software modules, implements the user interface with the correction algorithm and controls the correction equipment. It activates the measurement application and the *Mathematica* session, receives data from the instrumentation application and executes the correction

calculation through the computation module. The various options of the correction algorithm, like the selection of a sub-set of monitors or correctors, are presented inside a dialogue window. Computed corrections are displayed and can be applied under the operator control.

Each instance of this program is defined in an *Oracle* table. The definition includes the instrumentation application, the set of correction equipment and the *Mathematica* file.

4. ARCHITECTURE OF A GENERAL CORRECTION SYSTEM

In the present status of the beam steering procedure, the optical parameters are included in the code. In fact, the code should contain the correction algorithm only to be applicable to any ring or transfer line and the optical parameters stored in a data base. The de-coupling between algorithm and data is the major improvement to be implemented in the new steering control. As a consequence, the presence of a data base forces to revise the flow of information between data, algorithm and controls themselves.

4.1 Data base

A relational data base for accelerator physics has already been elaborated for the LEP collider [6] and will eventually be extended to LHC. It will thus be adapted to a general beam steering control. The data concern the characteristics of the magnets and of the beam monitors. A first set contains all information about the type of the elements, their position, length, magnetic field and conversion factors. Those data serve as input to a beam optics program which returns the parameters of the particle beam. That quantities have an intrinsic interest since they reflect the status of the channel and can be used not only for beam steering but also for any other beam manipulation such as the adjustment of the beam spot size at an intersection point or at a target.

4.2 On-line operation of a general correction system

The CERN-PS control system already includes computers which are dedicated to on-line *Oracle* applications (like reference values and sequencer description). However, the two required data-bases have to be set-up with their configuration procedures and tools.

The set of instrumentation equipment integrated into this architecture can be easily extended to the pick-up equipment of the transfer lines which are all controlled through a single instrumentation application in the new CERN-PS control system. Reference trajectories must also be included into the instrumentation applications for generalising the correction (conversion from magnetic values to electrical settings). This is illustrated in a simplified version of the data-flow diagram (Fig. 4).

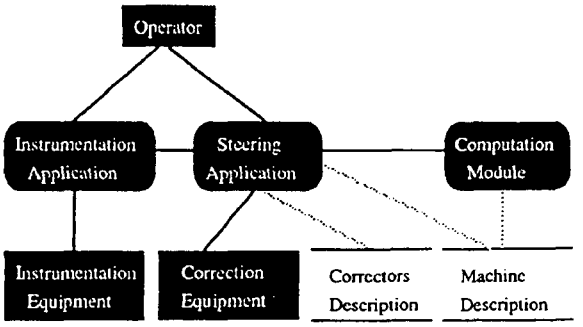


Fig. 4. Full system software architecture with its present status (black background) and planned extensions (white background).

5. CONCLUSION

An automatic beam steering procedure has been successfully implemented in a transfer line of the CERN PS complex and all the stages of development from machine experiments to integration into a control system have been described. As a net result, a manipulation which required operator's expertise and a substantial setting-up time in the past is now routinely used and performed in a few seconds, the incompressible time for data acquisition and trajectory correction. The next step is to make the present application generic so that it can be used on any machine and even for other classes of problems such as chromatic or geometric aberrations whose solution relies on the iterative solution of a linear system. In that context, a new architecture containing relational data bases is being designed.

6. REFERENCES

[1] S. Wolfram, *Mathematica*, a system for doing mathematics by computer, Addison Wesley (1992).

[2] B. Autin, Y. Marti, Closed orbit correction of A.G. machines using a small number of magnets, CERN ISR-MA/73-17 (1973).

[3] E. Courant and H.S. Snyder, Theory of the Alternating Gradient Synchrotron, Ann. Phys. 3, 1-48 (1958)

[4] G.H. Golub, Numerical methods for solving linear least squares problems, Numer. Math. 7, 206-216 (1965)

[5] M. Arruat, M. Martini, The new standard method to measure emittances in the PS transfer lines, CERN/PS 92-59 (PA).

[6] R.Bailey, A.Belk, P.Collier, M.Lamont, G.de Rijk, M.Tarrant, Development of the LEP high level control software using *Oracle* as an on-line database, Proc. ICALEPCS 1993 (to be published).

VMEbus based computer and Real-time UNIX as infrastructure of DAQ

Y.Yasu, H.Fujii, M.Nomachi, H.Kodama, E.Inoue
KEK, Oho 1-1, Tsukuba 305, Japan

Y.Tajima, Y.Takeuchi
Tokyo Institute of Technology, Meguro, Tokyo 152, Japan

Y.Shimizu
Mitsubishi Space Software Inc., Kamakura, Kanagawa 247, Japan

ABSTRACT

This paper describes what we have constructed as the infrastructure of Data acquisition system (DAQ). The paper reports recent developments concerned with HP VME board computer with LynxOS (HP742rt/HP-RT) and Alpha/OSF1 with VMEbus adapter. The paper also reports current status of developing a Benchmark Suite for Data Acquisition (DAQBENCH) for measuring not only the performance of VME/CAMAC access but also that of the context switching, the inter-process communications and so on, for various computers including Workstation-based systems and VME board computers.

1. INTRODUCTION

DAQ is extremely influenced by the evolution of computer architecture and operating system(OS). DAQ should be independent on them as possible. We adopt the VMEbus based computer system as hardware platform and UNIX including real-time UNIX as software platform for DAQ. The platforms we chose are industrial standard. According to this strategy, we had made VME/CAMAC device drivers[1,2,3] and then a data acquisition system, UNIDAQ[4,5] had adopted those drivers. The drivers have common user interface to VME/CAMAC access.

We have developed the VME/CAMAC device drivers for HP742rt/ HP-RT[6], ported the UNIDAQ and used for a test experiment at KEK[7]. The system has worked well and we got the high performance of the data acquisition.

The CAMAC device driver for ALPHA/OSF1 with VMEbus adapter has been also developed[8] and then the performance has been measured.

DAQBENCH[9] which is the tool for the performance measurement, have been developed for evaluating the DAQ performance.

2. COMMON USER INTERFACE TO VME/CAMAC

We had developed the CAMAC device driver & library on DECStation 5000[3] and SunSPARC[2]. We also had developed the VME device driver & library on DECStation 5000. Those functions has common user interface to the variation of the hardware and the software. Namely, the programs using those functions can be used without the modification of the code among our systems.

We have ported the CAMAC device driver & library on SPARC/SunOS to that on SPARC/Solaris.

The available systems of VME/CAMAC access are shown in Table 1.

3. DAQ for HP742rt/HP-RT

The UNIDAQ has been basically ported to the HP742rt and worked on the system for a test experiment at KEK[7]. Some of GNU tools have been ported. CERNLIB for the HP742rt is being ported. The development of the VME/CAMAC device driver for the full functionality is still in progress. At the test experiment, the collector gathers data from CAMAC, the recorder writes the data to the disk storage and the data are also transferred to a HP-UX system for the analysis and the display of the data on the system. The event size is 64 bytes and approximately 2 kHz of the event rate could be handled. The test experiment proceeded successfully.

Table 1. VME&CAMAC SUPPORT

	HP742rt HP-RT	Alpha OSF1	DECStation ULTRIX	Sparc SUNOS	Sparc Solaris
CAMAC					
Single	O	O	O	O	O
Block	*	O	O	O	*
Interrupt	O	O	O	O	O
KEK list	*	O	O	O	O
Kinetic list	*	O	O	O	O
VME					
Map I/O	O	X	O	--	--
Interrupt	O	*	O	--	--
Direct I/O	*	*	*	--	--

HP : HP742rt(VME board computer)
DEC : DECStation5000 with DEC's VMEbus Adaptor
DEC3000/400(Alpha AXP) with DEC's VMEbus Adaptor
SUN : Sparc IPC, Sparc IPX,Sparc 2, Sparc10, Sparc classic with SFVME(VMEbus Adaptor)
Sparc2E(VME board computer)
VME-CAMAC interface : Kinetic 2917
"O" means "supported"; "*" means "to be supported";
"X" means "no plan"; "--" means "not scheduled";

4. CAMAC DEVICE DRIVER FOR ALPHA/OSF1

All of the CAMAC functions have been supported. The evaluation of the system has been made. Table 2 shows the performance of the basic CAMAC accesses. The VME device driver is under development[8].

5. DAQBENCH

We already have many experiences that the standard benchmark suites like SPECmark is not suitable to the index to evaluate the DAQ performance. The recent real-time benchmark suites do not give complete set of indexes because they do not include DAQ own parameters like VME/ CAMAC access. We think that the exact definition and measurement of DAQ performance are very difficult because of the difference of the operating system, the hardware architecture and so on. Nevertheless, we need such a benchmark suite for the DAQ performance. Then, we have made the DAQBENCH that requires the VMEbus as hardware and the UNIX interface as software.

We have defined parameters of the DAQ performance such as VMEbus/ CAMAC access speed, the interrupt task response time, the context switching time, speed of the inter-process communication and so on, and measured the parameters on various computers. Table 2 and 3 show the part of the results. The DAQBENCH will be available from KEK[9].

6. SUMMARY

The DAQ system with the HP742rt had best performance among the systems in our measurements. The new VME board computer announced from HP will support VME block transfer as VMEbus master. We expect it has the better performance of VME access.

From the result of DAQBENCH, ALPHA/OSF1 has good DAQ performance. DEC announced new single VME board computer compliant with VME64 [10]. We expect the device driver will work on it.

The DAQBENCH has shown that DAQ performances on various computers depends on not only the computer architecture but also the OS. Those results show the DAQBENCH will become an index to analyze and evaluate the DAQ, and to select the computer system for DAQ.

High performance VMEbus standard, VME64 becomes ANSI standard recently. VME board computers with the high

performance CPU become cheaper these day. Real-time functions become industrial standard as POSIX. We expect that DAQ system based on VME64 board computer with the real-time UNIX will have best performance.

Table 2. Performance of CAMAC access

		HP742rt	Alpha	DECS	Sun-1	Sun-2
Single Action(μ sec)		20	90	160	130	120
Block Action	overhead(μ sec)	*	360	1000	720	*
	speed(KB/sec)	*	1000	980	920	*
Interrupt Handling(μ sec)		65	200	570	*	980

Table 3. Performance of IPC

IPC functions		HP742rt	i486	HP735	Alpha	DECS	Sun-1	Sun-2
context switch(μ sec)		12	23	22	30	44	79	117
Semaphore(μ sec)		12	16	11	13	67	114	79
Message Queue	overhead(μ sec)	41	44	37	41	126	148	213
	speed(MB/sec)	9.8	8.0	31	36	6.5	6.0	6.1

HP742rt : HP742rt/HP-RT V1.1
Alpha : DEC3000/400 /OSF1 V1.3, DEC VMEbus adaptor
DECS : DECStation 5000/125 ULTRIX V4.2A, DEC VMEbus adaptor
i486 : DECpc 66MHz, LynxOS V2.2 ; Sun-1 : Sparc2/SunOS4.1.2 ; Sun-2 : Sparc2/Solaris2.3
"*" means "not measured"

REFERENCE

[1] M.Nomachi, et al., "A high-speed data acquisition system using VME and RISC/UNIX workstation", Proc. Int. Conf. Computing in High Energy Physics, Tsukuba, 1991, ed. Y.Watase and F. Abe (University Academy Press, 1991) p.681

[2] Y.Takeuchi, T.Tanimori and Y.Yasu, Development of Data Acquisition System using RISC/UNIX(TM) Workstation,Nucl. Instrum. Meth. A328(1993)526.; SDC-92-397

[3] Y.Yasu et al.: POSIX Real-time Extension and RISC/UNIX Data Acquisition,Proc. Int. Conf. Computing in High Energy Physics, Annecy, France, 1992, (CERN92 07,1992)p.627.

[4] M.Nomachi et al, UNIDAQ, these proceed

[5] UNIDAQ collaboration, UNIDAQ Document Set,SDC-93-478, Superconducting Super Collider Laboratory, 1993.

[6] Y.Yasu and Y.Tajima, DAQ Performance of Real-time UNIX on New HP VME Board,1993, KEK Internal 93-11, SDC-93-600, AIDA-101

[7] Y.Tajima and Y.Yasu, Development of the data acquisition system with VME board computer and real-time OS for KEK-PS E248, Presented at the 49th annual meeting of Physical Society of Japan in Fukuoka, March 1994

[8] Y.Yasu and Y.Shimizu, Development of VME & CAMAC Device Driver for ALPHA/OSF1, to be published.

[9] Y.Yasu and Y.Tajima, Development of a Benchmark Suite for Data Acquisition (DAQBENCH), to be published.

[10] P.S.Yoo, RISC and Realtime: Alpha AXP Board Computers for VMEbus and Futurebus+, Open Bus Systems '93, Munich, Germany, 1993

A KEYBOARD CONTROL METHOD FOR LOOP MEASUREMENT *

Z. W. Gao
Dipartimento di Fisica
Universitá Degli Studi di Roma"La Sapienza"
and INFN Sezione Roma
Piazzale A. More 2, I-00185 Roma, Italy

Abstract

This paper will describe a keyboard control mode based on the DEC VAX computer. The VAX Keyboard code can be found under runing of a program was developed. During the loop measurement or multitask operation, it able to be distinguished from a keyboard code to stop current operation or transfer to another operation while previous information can be held. The combining of this mode, we successfully use one key control loop measurement for test Dual Input Memory module which is used in a rearrange Energy Trigger system for LEP 8 Bunch operation.

INTRODUCTION

With development of experimental high energy physics, the reliability of electronics modules become more important for data acquisition. It is indispensable to the step using the computer to test the performance for long periods, the reliability for those modules should be installed before the experimental system. The DEC VAX computer widely used in the last ten years in high energy physics laboratories instead of PDP/11 computer. We have developed a keyboard control method based on VAX computer in order to control loop or long periods measurement. The aim of such method is to flexible application in computer automic control and measurement. The paper will describe the following problem: How can a VAX keyboard code be found ? An application example in the loop measurement which is test Dual Input Memory (DIM) module where anyone keyboard code can be selected as the identificate code to control loop measurement.

HOW CAN A VAX KEYBOARD CODE BE FOUND ?

In order to find anyone keyboard code as the identificate code, we can seek a VAX keyboard code through call VAX Screen Managenment SMG Run-Time Library routines [1]. Fig. 1 is a flowchart that shows how a VAX keyboard code be found. A corresponding program which is overlap well with all of VAX versions from 4.4 to current 5.5 version was developed.

The decimal ASCII number of a VAX

*This work is supported by the International Center for Theoretical Physics (ICTP) and Istituto Nazionale di Fisica Nucleare (INFN).

keyboard can be found under runing such a program. For example, twentysix alphabet keys a,b, ... x,y,z, their corresponding number for 97,98, ... 120,121,122; but capital letter key A,B, ... X,Y,Z, their corresponding number for 65,66, ... 88,89,90. Other symbolic key whose decimal ASCII number also can be known via the same way.

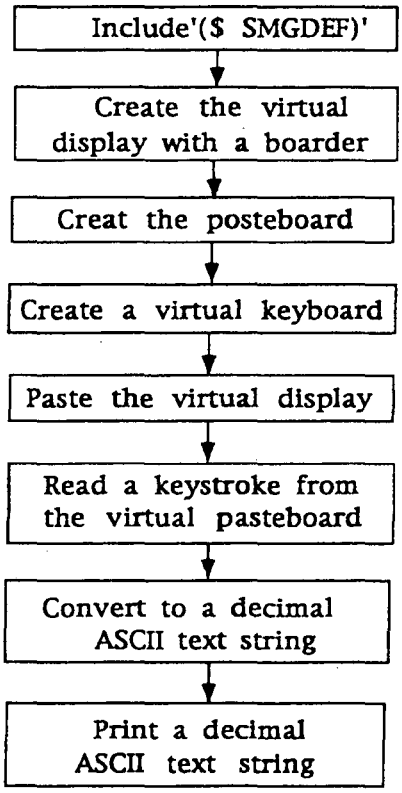


Fig. 1. A flowchart for recogntze keyboard code

AN APPLICATION EXAMPLE IN THE LOOP MEASUREMENT

Based on the above method, we could edit a special subroutine to read a single keyboard code if it was selected as an identificate code for stop loop measurement. Under loop measurement condition, the program will call this subroutione to determine if loop measurement will continue or stop. This method is useful to control long periods of testing the performance of electronic modules.

Test Dual Input Memory Module

Our combining of this method is to test new DIM module for a rearrange Energy Trigger system [2] for the LEP machine will operate with 8 bunches of electrons and 8 bunches of positrons. In the 8 bunch mode, the time interval between two bunch crossings is 11.1 us (in previous LEP 4×4 bunches it was 22.2 us).

DIM is a fast single unit CAMAC module which can receive two 16 bit words via two front panel connectors, and are memorized in a sequential mode into a 1024*16 bit memory with $\sim 20ns$ delay time from input to output[3]. DIM with Fast Encoding and Readout ADC (FERA) modules LRS 4300, FERA Driver, Memory-Look-Up (MLU) modules LRS 2372, Bus Switch (BSW), consist of a conversion group [4][5]. There are nine conversion groups in new trigger processor to reduce the decision time in order to search for "hits" for L3 experiment under LEP 8 bunch operation. Simply, there are many cycles of Write and Read Memory operations for DIM under such circumstances. Because of the module use fast ECL memory with some ECLPS IC may create some noise, glitch and error for Write and Read operations. Therefore, it is necessary to test for long periods the reliability for multi-cycles Write and Read Memory before the module install of the experimental system.

Fig. 2 shows a basic test flowchart for multi-cycles Write and Read or loop measurement for DIM. Fig. 3 shows a test block diagram for the measurement

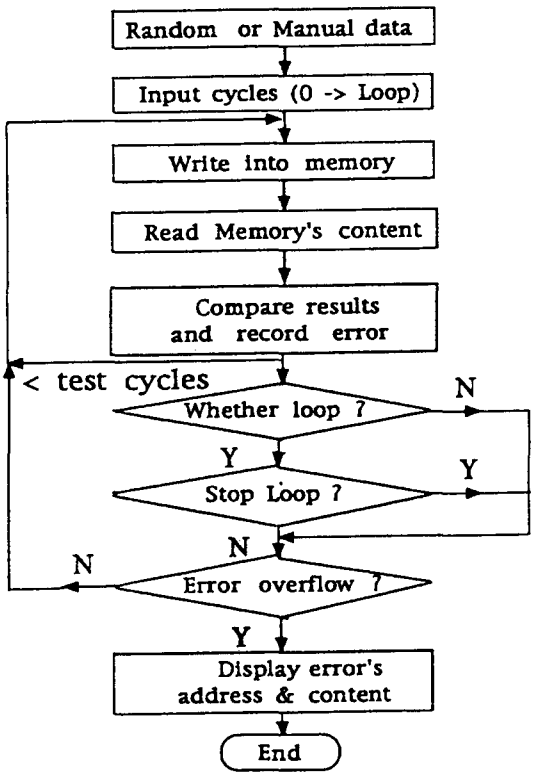


Fig. 2. A basic test flowchart for measure DIM module

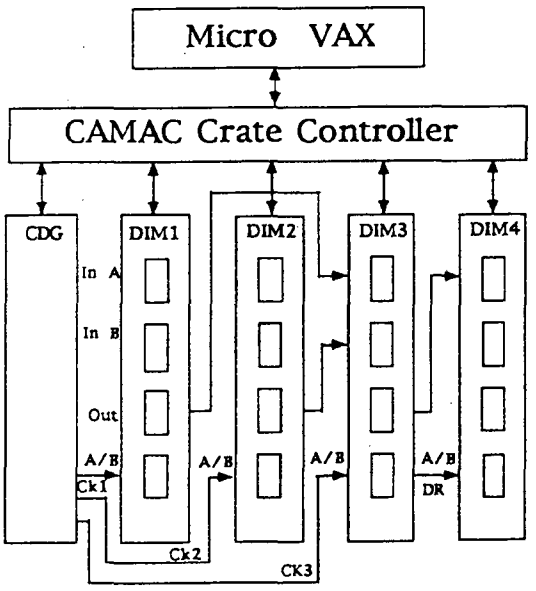


Fig. 3. Test block diagram for DIM R/W operation

of four DIM modules in order to verify Read and Write operation via both dataway and front pannel. The test program set Clock Delay Generator (CDG) to generate clock signals for test operation. During CK1 and CK2, the data via CAMAC CC will be wrote to DIM1 and DIM2 where the readout information is sequential via front pannel connectors sent to DIM3 under function of CK3 signal. When Data Ready (DR) signal of DIM3 is true and DIM3's readout information will also be wrote to DIM4 where the last readout information will be sent to dataway. All of Write and Read information will be compared and error information with relevant address of Memory's cell will also be displayed and recorded.

Select an identificate code for control loop measurement

We can define a key at terminal dialogue to decide if a loop measurement is desired or not. After initial test we may choose the loop measurement when we may leave for a long time (e.g. after evening). The terminal will display following content:

Now, loop test begin, you can leave drink coffee or go home. Stop loop, only keystroke p key after the ≫ prompt

We select p key (decimal ASCII number 112) as an identificate code for stop loop measurement. When we keystroke p key after the ≫ prompt, the loop measurement will be immediately stoped and all error information will be displayed and recorded to the disk. We should evaluate a value of number as an error overflow threshold. When the error number is over the threshold, the loop measurement will also be stoped.

CONCLUSION

According to such a control method, we may successfully measure more than 30 modules of DIM via both CAMAC Dataway and front panel by using both randam and manual data. This is helpful to the stable operation for the modules in the experimental system.

Certainly, call $SMGDEF may occupy some cpu time and will be weighed by user and depend on the different environment. However, this keyboard control method could also be used on the other computer automatic control system, as our described in the example based on the DEC VAX system. Generally, people should know how to edit a subroutione for reading certain keyboard codes in their own computer system: for instance, above described SMG Run-Time Library routines for VAX computer while in PC computer circumstance there could also be found a similar content in Bus I/O System (BIOS). Combining the subroutione to main program to control loop measurement or other application such as on-line monitor etc. will be done.

ACKNOWLEDGEMENTS

I am grateful to Mr. G. A. HU for the helpful discussion on section II and to Professor R. Bizzarri, B. Borgia and F. Cesaroni for their concerning work. M. Corsi is acknowledged for her helpful english.

REFERENCES

1. DEC Co., "MicroVMS Programmer's Manual" AI-Z212B-TE, April,1986.
2. R.Bizzarri et al. "Modification of L3 Trigger and DAQ for LEP 8 Bunch Operation" L3 internal report, # 900, 1991, CERN , Geneva, Switzerland.
3. F.Cesaroni, S.Gentile, G.Pascale and Z.Gao "A fast Ecl-bus memory: dual input memory" Nucl. Inst. and Meth A323(1992) 549.
4. "The Construction of the L3 Experiment" p 200, L3 internal report,#678, October 11, 1989, CERN, Geneva, Switzerland.
5. P.Bagnaia et al. "The L3 energy trigger for the LEP 8 bunch operation" Nucl. Inst. and Meth. A323(1992) 528.

AN INTEGRATED APPROACH TO CONTROL, MONITORING, TEST, AND CALIBRATION OF SILICON MICRO–STRIP DETECTOR MODULES AND SYSTEMS.

Trygve Ødegaard, Einar Nygård, Oddvar Søråsen
Department of Informatics, University of Oslo
P.O.Box 1080 Blindern
N–0316 Oslo, Norway

Abstract

This paper presents a concept integrating traditional slow–control functionality into a *Control & Monitoring* unit. Functional flexibility and electrical isolation are keywords behind the development.

INTRODUCTION

For the time being, much resources are spent on developing sophisticated Front-End circuits being able to handle the high data rates at future LHC [1] experiments. Most experimental collaborations seem to aim at data readout through multiplexing over optical fibres [2]. One of the advantages of such a scheme is the electrical isolation between the detector modules and the DAQ/trigger systems.

However, less attention has been paid to the fact that, in order to fully take advantage of such optical decoupling, the numerous electrical cables traditionally used for biasing and control of detector modules must be removed. Consequently, the Control & Monitoring unit (C&M) [3,4] described herein is being developed as an alternative to the traditional means of biasing and control. The unit is not just a substitution but also a provider of many new advantages and features.

The C&M has a flexible architecture which allows implementation of important functions including the ability to control bias currents and voltages used by the Front-End (FE) chips, to calibrate these chips, and to adjust the phase of BCO-clk and 1'st level trigger signals. Also, desirable monitoring functions used to verify outputs from control circuitry, to measure temperature, absorbed dose, and power-supply voltages etc. can be implemented. This applies also to various testing functions.

SYSTEM OVERVIEW

Figure 1 shows how the C&M unit can be placed in a possible detector module for a future LHC experiment. In the figure the unit is shown to share an optical link with the event data readout. This communication configuration is not necessitated by the C&M specification, but should rather be viewed as an attractive solution since this would normally add only an insignificant overhead to the event data stream.

The figure indicates a C&M unit where most of the foreseen functions have been implemented. The hierarcial con-

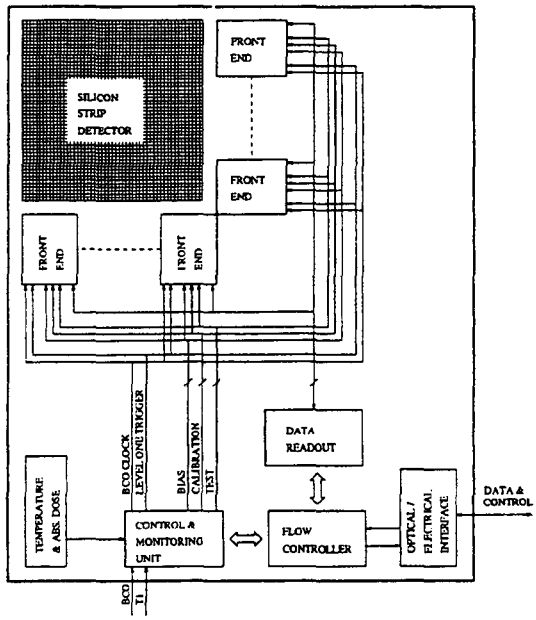


Figure 1. Control & Monitoring unit shown in possible future detector module.

cept makes it easy to design prototypes with less functionality, or to add new features if that should be desirable.

For the time being, there are different opinions with respect to whether the Front-End chips can share for example common bias-lines, or if some critical biases have to be generated individually for each FE chip. The figure shows shared lines, but the flexible architecture of the C&M unit makes an extension to individual control easy.

THE C&M UNIT

The system is built up around a Bus Handler (also called master), the C&M bus, and a number of slave modules. A sketch of the system is shown in figure 2. The Bus Handler has two tasks, controlling the bus activity, and interfacing to the external message exchange. The slave modules constitute the various functions mentioned in the introduction.

The C&M bus signals can be seen in figure 3. The bus is kept as simple as possible, featuring mainly a 8-bit parallel bidirectional data bus and a 4-bit parallel address bus. With one address reserved for broadcast mode, the bus allows for a maximum of 15 slave modules.

As can be seen in figure 3, each slave have a dedicated interrupt-line to the Bus Handler (BH). Inside the BH these lines are divided into two different priority groups. If the C&M unit share an external communication link with the event data readout, situations might occur where slave module messages should take priority. For example, this might be the situation if the power consumption or temperature move outside limits that have been preset. An example of how this could work is described below.

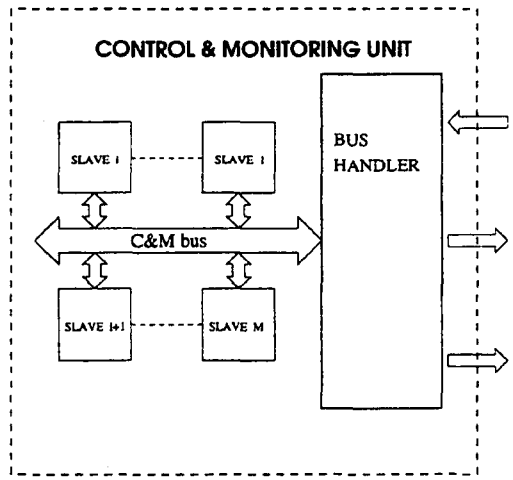


Figure 2. The Control & Monitoring unit.

SLAVE MODULE EXAMPLES

Figure 4 shows the foreseen monitoring function. Slave i will continuously cycle through the analog inputs and convert these in a slow speed ADC. The results will be stored in the output buffer, from where data typically may be read out upon requests from outside the detec-

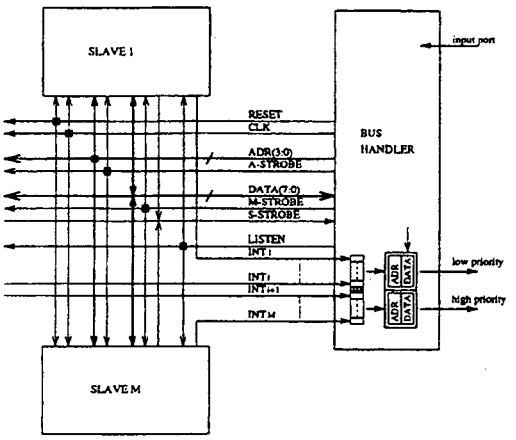


Figure 3. C&M bus signals.

tor module. In parallel to this, the data are also loaded into the data buffer in slave *i+1*. Here they will be compared to the minimum and maximum values that have already been preset. If the data turn out to be outside these limits, the slave will immediately initiate a readout cycle by asserting its interrupt line. This interrupt line corresponds to the high priority output port, which means that external circuitry should recognize the message as a message containing urgent data. In this case no external request for data readout is required.

Another slave is intended to enclose the function of the BCO–clk and 1'st level trigger phase adjust. The main elements here will be two delay lines and two data registers. The delay lines can be implemented with inverter chains, and the output positions in these chains are determined from the content of the data registers. In case a scheme where the 1'st level trigger is modulated on the BCO–clk signal [5], a stage demodulating this information could be added in front of the phase adjusting stage.

Design of a slave module serving the common bias configuration has been started. The main features are a *coarse DAC*, and a number of *fine DAC's* corresponding to the number of bias lines. The coarse DAC is used to generate a current proportional to the number of FE chips in the module. This current is then mirrored onto the bias lines. Each current–mirror have an individual scaling factor, giving the minimum value for the corresponding bias with the given number of FE's. The fine DAC's will then be used to adjust the individual biases by adding current with small increments. The mid–range value in the fine DAC's corresponds to nominal current values on the output bias lines.

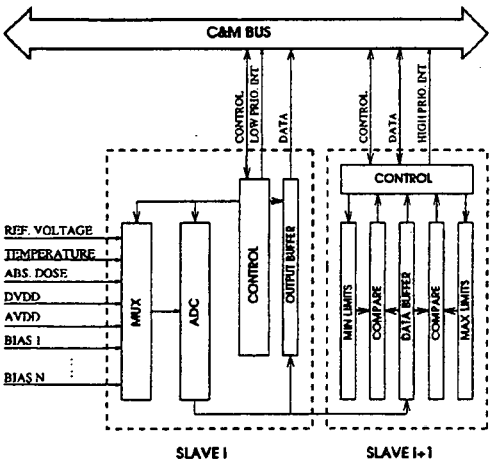


Figure 4. Monitor slaves.

CONCLUSION

This paper has presented a flexible concept intended to be used for control and monitoring of possible detector modules in a future tracking detector for LHC. The authors believe that gathering all necessary control and monitoring functions into a coherent system as proposed herein, is necessary to fully exploit the potential of the foreseen complex detector modules. The hardware development has been started.

REFERENCES

- CERN, "LHC – Large Hadron Collider", CERN Publications, European Laboratory for Particle Physics, Switzerland, June 1990.
- The ATLAS collaboration, "ATLAS, Letter of Intent", Technical Report CERN/LHCC 92–4, CERN, May 1992.
- O. Søråsen et. al. "A Switched Interconnection Network for Large Scale Instrumentation", ATLAS Internal DAQ Note 06, October 1992. RD–20 Note 10, October 1992.
- O. Søråsen et. al. "A high performance data driven packet–switching network for detector data readout and event–building in an LHC inner detector experiment", IEEE Eighth Conference on Real–Time Computer Applications in Nuclear, Particle and Plasma Physics. June 8–11, 1993.
- B. G. Taylor. "Timing, Trigger and Control Distribution for LHC Detectors", CERN ECP Division, Rev 1.2, 22 November 1993

Data Compression And Readout of Multiplexed Drift Chamber Data in the CLAS Detector at CEBAF

David C. Doughty Jr., D. P. Heddle, David Allgood
CHRISTOPHER NEWPORT UNIVERSITY
50 Shoe Lane, Newport News, VA, 23606, USA

Abstract

A unique system for multiplexing drift chamber wires is used in the CLAS detector at CEBAF. Discriminator outputs from two non-adjacent wires have different pulse widths and are XOR'd before being sent into the TDC. Additional circuitry integrates the charge on the wires, then discharges and uses the same TDC channel as a Wilkinson rundown ADC. This multiplexing, while reducing the cost of the electronics, causes the information read out of each TDC to be scrambled, and roughly three times larger than required. The event rate of 1-2 KHz and raw event size of 20-40 KBytes make data extraction and compression mandatory. This will be done in the Fastbus readout controllers, in special VME compression/transmission crates, and in the on-line farm. We briefly detail the multiplexing scheme, describe the extraction/compression algorithm, and present results of running this code on several Fastbus readout controllers, VME boards, and DSP's.

I. INTRODUCTION

The CEBAF Large Acceptance Spectrometer (CLAS) is designed for kinematic analysis of several final state particles. It uses a toroidal magnetic field generated by six coils for momentum analysis; for this reason the detector package has been partitioned into six wedges or sectors. Each sector fits between two adjacent coils and consists of four types of detectors: six superlayers of hexagonal cell drift chambers for charged particle tracking, Cerenkov detectors for electron-pion separation, scintillation counters for particle identification by time-of-flight, and an electromagnetic calorimeter for energy measurements of electrons, neutral pions, and photons. Details of the detector design are given in reference [1].

The CLAS detector is designed to run at luminosities exceeding 10^{34} $cm^{-2}s^{-1}$ producing a hadronic interaction rate of several Megahertz. The data acquisition system is being designed to handle event sizes of 20-40 Kbytes and an expected event rate of up to 2 kHz. To acquire desired events with high efficiency while minimizing the deadtime, a two-level hierarchical trigger has been designed. The LEVEL 1 trigger is deadtimeless, processing all prompt signals through a three-stage pipelined memory lookup within 90 ns. Details of the LEVEL 1 trigger are given in reference [2]. The resulting signal provides a common start to the photomultiplier tube ADCs and TDCs; delayed versions (to allow for the drift time) of this trigger control the operation of the drift chamber TDC's as discussed below. After the LEVEL 1 trigger accepts an event the detector is dead for 2 μs. During this time the LEVEL 2 trigger uses hits in the drift chambers to find tracks and match them with the trigger requirements [3].

Once an event is accepted at LEVEL 2, conversion of the front-end data is initiated and the detector will not go live until all ADCs and TDCs have digitized and locally buffered their data, a process which typically takes 20 μs. After conversion the data in each crate is scanned by a readout controller (ROC). Data is sent from several ROC's to a VME crate for compression, then transmitted over ATM or FDDI data links through an event building switch into an on-line processor farm for final reformatting and triggering before the event is written to tape.

The drift chambers in the CLAS detector have a total of about 34,000 wires. Timing

resolution of less than 1 ns over a 1.5 μs drift interval is required to support the tracking requirements. The drift chambers will also be used for particle identification by dE/dx, so the energy loss must be measured with a precision of 9 bits. Therefore the drift chamber electronics need to provide an ADC function as well as a TDC function for all wires. To support the triggering and data acquisition requirements, they must provide hit bits for the LEVEL 2 trigger, they must be capable of digitizing within 20 μs, and must handle a readout rate of up to 2 kHz.

II. MULTI-HIT PIPELINED TDC's

The use of common stop, multi-hit pipelined TDC's with multi-event readout buffers allows all of these goals for the drift chamber electronics to be met. These are capable of meeting the TDC resolution requirements, but have three additional features which make their use attractive. Since they are common stop they eliminate the need for delay cables for each of the 34,000 wires. Also, the TDC can be used to perform an ADC function by integrating the charge and using the TDC to time the discharge, as in a Wilkinson rundown converter. Finally, if the TDC's are capable of recording both rising and falling edges of the discriminator pulses, pulse width encoding can be used to multiplex two wires onto one channel, greatly reducing the electronics cost.

Preamplified signals from the CLAS drift chamber wires are sent into a custom 'Front End Board' (FEB). A block diagram of one channel on the FEB is shown in Figure 1 [4]. Each channel is split and sent into a comparator and (after passing through a short delay line) a charge to time converter (QTC). The comparator output triggers two monostables; one is used to enable the QTC's integrating function, while the other produces the logic output pulse used for timing. A hit bit which is used by the LEVEl 2 trigger logic is also set. If no delayed trigger is received by the end of the integrating pulse, the QTC and the hit bit are automatically reset and the channel is ready for another event.

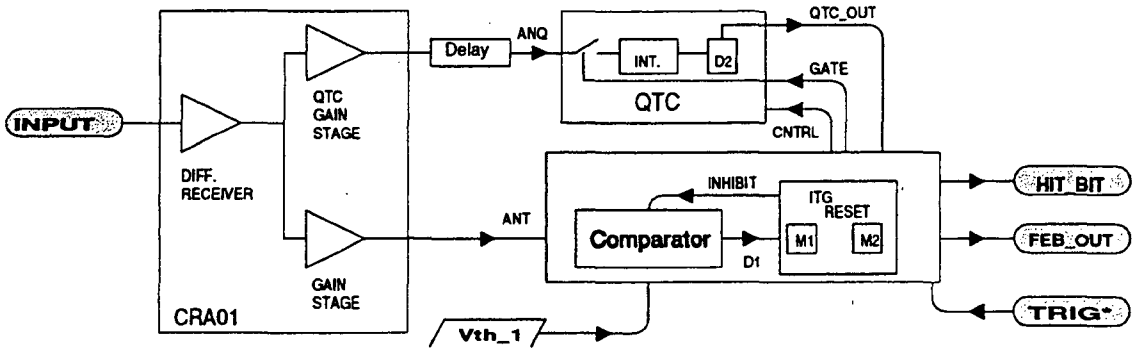


Fig. 1. A block diagram of one channel of the front end board. The hit bit goes to the LEVEL 2 trigger, the FEB_out goes to the TDC, and the Trig (input) is a delayed version of the LEVEL 1 trigger

Receipt of the delayed trigger causes the integrator to hold and after a short delay begin discharging at a constant rate. After the QTC has discharged to a preset threshold, the output logic monostable is triggered again, producing two more hits in the TDC. The time interval between the trigger and the leading edge is therefore proportional to the integrated charge. The final stop for the TDC's is produced by a further delayed version of the LEVEL 1 trigger. Each wire chamber hit thus produces four pieces of data in the TDC, two edges (one rising and one falling) for the time signal, with the leading edge corresponding to the time of the signal, and two other edges (with the same pulse width between them) from which the analog charge may be determined.

Since only the leading edges are significant in the two pulses recorded above, why are the falling edges recorded? The answer is that knowledge of the pulse width allows more than one wire to be multiplexed onto one TDC channel. If the outputs of two

channels are XOR'd together the resulting pulse train can always be decoded, no matter what the relative timing of the hits on the two wires, if the two different pulse widths are carefully chosen. If only one wire has data the output looks like the input, while if they both have data the output will be a combination of the two inputs, as shown in Figure 2.

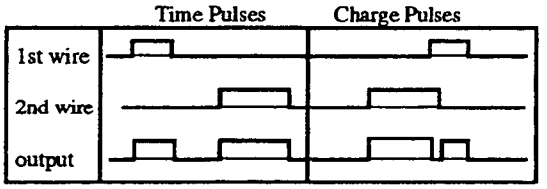


Fig 2. The output from each wire contains a time pulse and a charge pulse. Two wires 4 cells apart are XOR'd to produce an output which is sent into the TDC.

Because the pulse widths are unique, the original data can be extracted, even if an edge is lost due to the double pulse resolution of the TDC. More computational power is required to decode the case in which both wires have data; for this reason wires four cells apart are multiplexed together, to minimize the probability of correlated hits in the wires. This multiplexing cut the number of channels in half, reducing the electronics cost by nearly one million dollars.

III RECOVERING THE DATA

This multiplexing technique, while reducing the number of electronics channels and saving money, has its penalties. Firstly, the pulse widths must be examined to determine which of the two wires multiplexed into this channel produced the hits. If both channels were hit the data may be confused and have to be disentangled before times and charges can be assigned to the two channels. Secondly, the amount of data corresponding to a hit on a wire is increased. If only the leading edges of the timing and charge pulses were recorded each hit would produce two 32 bit words of data. By recording both edges, each hit produces four 32 bit words of data, or twice as much. In fact, these 128 bits are nearly three times as much as required, since the address, time and charge of a hit could each be stored in a 16 bit word.

To deal with these complications some form of data extraction and compression must occur before the events are written to tape. Because the excess amount of data is large it needs to be reduced as early in the readout chain as possible, in order to minimize the bandwidth required in the links between the FASTBUS and VME crates, and between the VME crates and the event building switch. This implies compression of as large a fraction of events as possible by the FASTBUS readout controller within each crate, and the completion of compression by the VME compression/transmission crates.

To study the rate of extraction and compression obtainable with various readout controllers and processors, a C program called CRUNCH (*CLAS Reduction of UN*compressed *CH*amber data) was developed. This program has two phases, a generation phase and an analysis phase. In the generation phase one crate's worth of data (about 3 Kbytes) is generated for typically 100 to 200 events. In the analysis phase these events are compressed and the required CPU time is measured.

The algorithm used for analysis has been carefully optimized. At the low to moderate noise levels expected for CLAS, the most likely occurrence is that only one of the two wires has data. When both wires are hit, there are 13 different cases for the relative timing of the two wires multiplexed onto one channel. The relative probability of each case depends on the noise, the occupancy, the double pulse resolution and the two unique pulse widths. Space limitations prevent us from including figures for all 13 cases, but they can easily be deduced: the first wire's falling edge comes well before (i.e., compared to the double pulse resolution) the second wires rising edge, the second wire's pulse comes well before the first wire's pulse, the permutations where edges come within the double pulse resolution and some are lost, etc. An example is shown in Figure 3. The analysis first searches for the most frequent combinations to minimize the aggregate time. Thus, the case of a single wire is tested first. If that fails, the 13 possible cases where multiplexing has occurred are tested in order of decreasing likelihood.

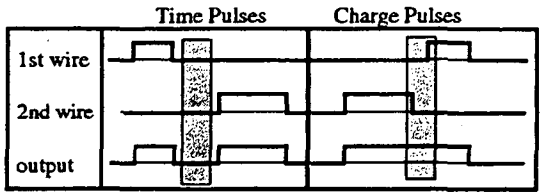


Figure 3. An example of multiplexing. The shaded band represents the double pulse resolution. For the time data, the multiplexing is clean. For the charge data, a transition is lost due to the insufficient separation of two edges. CRUNCH can recover both the original time and charge data.

IV. COMPARATIVE MEASUREMENTS

Data files were prepared in both big-endian and little-endian formats to allow the analysis program to be tested on a variety of machines. Because the compression may be carried out in the Fastbus crates, the VME compression/transmission crates, or the on-line processor farm, the code was tested on a variety of machines, ranging from Hewlett Packard PA architecture and DEC MIPS based RISC machines, through FASTBUS readout controllers, to 68040 and 320C30 (DSP) based VME boards. Representative results are given in Table 1.

The rates for the conventional CISC and RISC processors are indicative of the integer operation of these machines, and in fact scale reasonably well with the MIPS rating. The slowness of the FRC compared with the Decstation 5000 is thought to be due to a slower overall memory design with the only cache being that built into the LR33000 embedded processor chip. One surprising result is the relative slowness of the 320C30 DSP chip in performing this algorithm. DSP chips have a reputation for very fast performance on arithmetic sum-of-products calculations in which both calculations (the product and the running sum) can be done at once. They appear to have poorer performance (relative to the clock rate) in this kind of integer and logical bit manipulation algorithm.

Table 1. Results of running the compression algorithm (CRUNCH) on several processors.

Computer	Processor	Clock-MHz	MIPS	Location	Events/sec
HP-735	PA 7100	99	124	CEBAF	6450
DEC 5000	R3000	40	40	CEBAF	2350
MOT 167	68040	25	20	CEBAF	1870
Fermi FRC	LR33000	40	40	FERMI	1800
Pentek DSP	320C30	32	16	CNU	505

V. CONCLUSIONS

The Fermilab-developed FRC has good potential for compression. While some fraction of its time will be used in managing the FASTBUS backplane and scanning the data, its design should allow roughly 50% of its time to be used for processing. This would allow it to compress data on the fly at a rate of 800-1000 Hz. A fast RISC processor of 60-75 MIPS in the VME compression/transmission crates should be able to handle the remaining uncompressed events from two to three crates.

VI. ACKNOWLEDGMENTS

The authors wish to acknowledge Mac Mestayer and Fernando Barbosa at CEBAF for useful discussions. Also the FERMILAB staff for facilitating the use of a prototype FRC. This work was supported in part by Department of Energy Contract DE-AC05-84ER40150.

VII. REFERENCES

[1] Conceptual Design Report, CEBAF Basic Experimental Equipment, CEBAF, April 13, 1990.

[2] D. Doughty et al "A VXIbus Based Trigger for the CLAS detector at CEBAF." IEEE Transactions on Nuclear Science, NS 39, 1992 p 241-247.

[3] D. Doughty et al "A Massively Parallel Track Finding System for the Level 2 Trigger in the CLAS detector at CEBAF" IEEE Transactions on Nuclear Science, NS 41, 1993 p 267-273.

[4] Fernando Barbosa, private communication.

Data Flow Manager for DART[1]

David Berg, Dennis Black, Dave Slimmer
On-Line Systems Group/Computing Division

Jürgen Engelfried
E781
Physics Department
Fermilab Research Division

Vivian O'Dell
Special Assignments/Computing Division

1. This work is sponsored by DOE contract No. DE-AC02-76CH03000

Abstract

The DART Data Flow Manager (**dfm**) integrates a buffer manager with a requester/provider model for scheduling work on buffers. Buffer lists, representing built events or other data, are queued by service requesters to service providers.

Buffers may be either internal (reside on the local node), or external (located elsewhere, e.g., dual ported memory). Internal buffers are managed locally. Wherever possible, **dfm** moves only addresses of buffers rather than buffers themselves.

1 Introduction

DART[1], a collaboration of fixed target experiments, developed a buffer management system[2] that will satisfy their varied requirements[3] with a single coherent facility.

dfm requester tasks assemble lists of pointers to buffers, representing, for example, various fragments of an event, and schedule them to **dfm** provider tasks, which perform services such as logging events to tape. Each experiment designs its own requesters and providers, building on a framework of **dfm** functions.

A tool, **snapshot**, was also developed for debugging and performance measurement. This tool runs as a separate process that spies on **dfm**'s control structures.

2 Requirements

The basic requirements for **dfm** were that:

- The data flow manager should be almost invisible to the experimenter, making the use and integration relatively easy.
- There must be well defined interfaces between the data flow manager and other data acquisition components, e.g., run control, logger, Hoist[4].
- The data flow manager must either avoid, or be able to detect and gracefully recover from, deadlocks, insufficient resources, etc.
- The data flow manager should impose no strict real time requirements on Unix workstations used as filter farms. The occasional slow analysis of an event must not halt the overall flow. Even if a processor completely drops out, data acquisition must not halt (though it may slow proportionately).

- Similarly, flow control problems with back end workstations must not interfere with data acquisition.
- There should be no restrictions on the content of buffers.
- Data buffers should be externally tagged with characteristics that can be used to control flow, buffer selection and distribution among sinks, etc.
- A data buffer may be represented to the buffer manager directly, as a collection of data to be moved, or indirectly, as the address of the data.
- Beyond those necessary for fault tolerance, there are no requirements for dynamic re-configuration during a run.
- In addition to the buffer manager routines being used by programs written in C (or C++), there must also be FORTRAN bindings.

3 Buffers

Local buffers are managed as a pool that is pre-allocated using a distribution of discrete sizes specific to each experiment. A request for a buffer is made by size, and satisfied by the smallest available buffer meeting the requirement.

There are three kinds of buffers. There may be any number of buffer classes defined for each kind, up to an aggregate of 255 classes. Classes are defined by the experiment as needed. The three kinds are:

- *external* - buffers not managed by **dfm**.
- *internal* - local buffers, including buffer lists, managed by **dfm**.
- *sub-buffer* - a local buffer may be divided arbitrarily into sub-buffers; each such parent buffer defines a separate class.

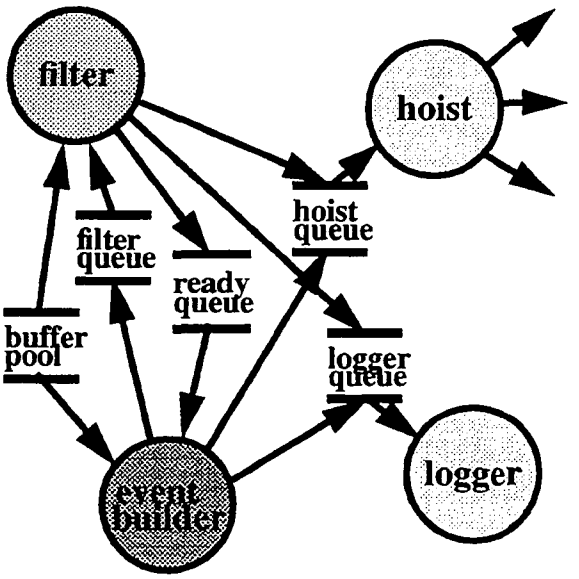
Associated with each internal buffer is a buffer descriptor consisting of the buffer class and buffer use count. The buffer descriptor physically precedes the buffer in memory.

4 Buffer List

A buffer list is a data structure containing a set of buffer locators and control information associated with a single event or other unit of data. The pointer to this control structure is the item that is passed onto message queues between requesters and providers. Because the buffer list is itself an internal buffer, it has an associated buffer descriptor. Furthermore, buffer lists may themselves be included as buffers in other buffer lists.

5 Service Requesters and Providers

Service requesters reserve buffers and build or modify buffer lists, which they schedule to service providers. One list may be scheduled concurrently to many providers. Service providers access the lists and buffers; when a service completes, it attempts to return the buffer list to the free pool. A single process may be a requester, a provider, or a combination of both.



Requesters schedule buffer lists by specifying a service group. All scheduling is limited by a time out period, so if any queue is full, the requester will block until there is room, or until the time out.

- *broadcast* - attempt to post the list to every provider in the service group.
- *round robin* - take providers in the service group sequentially, and attempt to post the list to the next one. If the queue is full, skip to the next one.
- *next available* - maintain a ready queue of providers, and attempt to post the list to the first provider on the queue. If the ready queue is empty, or the service

queue is full, block until a provider becomes ready, or until the time out.

Providers are responsible for acquiring buffer lists from their respective queues, and blocking if their queues are empty, or until a time out. Providers may specify several optional posting requirements, which they may also change at any time. If requirements are specified, lists are posted only if they meet all requirements at the time of the request. These requirements include: *type mask, size limit,* and *frequency modulus*

Buffer use counts are maintained by the **dfm** calls in requesters and providers. They are used to track when the buffer is still in use or when it can safely be returned to the pool. The use count is incremented as a result of reserving, inserting and scheduling; it is decremented when a requester or provider has finished with the buffer. When the use count reaches zero, the buffer will be returned to the free buffer pool.

6 Semaphores

On Unix, **dfm** uses System V IPC semaphores for both mutual exclusion and synchronization. Message queues are implemented within **dfm.**

Mutual exclusion semaphores are used to insure the integrity of the information stored in the control structures, message queues, and buffer descriptors, all which are located in shared memory segments. Currently, the semaphore granularity is at the service group and buffer class level.

Synchronization semaphores are used to signal a blocked or waiting process that the status of the queue has changed. Each process using **dfm** in the system requires its own synchronization semaphore.

7 Memory

dfm uses two shared memory segments: control and data. At initialization these segments are reserved in shared memory or kernel memory, depending on the configuration of the user's system.

The control segment holds all data structures and message queues. All internal buffers, including buffer lists, are located in the data segment.

8 Message Queues

Queues and linked lists are used throughout **dfm**, in several different forms:

- Each service provider is associated with one or more message queues to which buffer lists are posted. A limited number of lists, and at most one provider, may be queued or blocked on each.
- Each service group has a message queue to which ready providers are posted, which is used for the next available distribution mode of scheduling buffer lists. A limited number of providers, and any number of requesters may be queued or blocked on each.
- Each internal buffer class has a corresponding free pool or queue and any number of requesters may be blocked on each.

9 Summary

The initial implementation of **dfm** was for VxWorks, a real-time multi-tasking kernel for embedded processors; this is currently in use for data acquisition by at least one Fermilab experiment. An implementation for Unix has been developed and is actively being integrated into several others.

References

[1] Pordes et al., "Fermilab's DART DA System," these proceedings.

[2] D. Berg et al., "DART Data Flow Manager Design," Fermilab CD DS-225.

[3] D. Berg et al., "DART Data Flow Requirements," Fermilab CD DS-226.

[4] Oleynik, Gene, "Integrating UNIX Workstations into Existing Online Data Acquisition Systems for Fermilab Experiments," Proceedings of Computing in High Energy Physics, 1991.

Global Decisions with a Neural Second-Level Trigger System

J.M. Seixas , L.P. Caloba , A.L. Braga and **E.M.S. Moura**
COPPE/EE/UFRJ, C.P. 68504, Rio de Janeiro 21945, Brazil

Abstract

A second-level trigger system for LHC is developed using neural networks. The system is based on a global decision unit that combines information coming from different detectors. Detector data is obtained from Monte Carlo simulations and all events passed the conditions of a simulated first-level trigger. Performance reached 98.8% electron efficiency with 2.9% of jets misidentified.

Introduction

For LHC, detailed physics simulations indicate that a second-level trigger system is desired to achieve the background rejection needed. The operation of such system is being conceived in two phases [1]. Firstly, the main features of each detector are extracted. This is performed over regions of interest (ROIs) identified by the first-level trigger (L1). Next, these features are combined by means of a global decision unit.

For calorimeters, neural networks have been proved to perform better than classical methods as feature extractors [2, 3]. In this paper, a neural global decision unit is proposed, trying to explore the ability of neural networks in identifying correlations in a multidimensional space.

Four subdetectors were simulated [4]: a fine-grained calorimeter, a segmented preshower, a TRD and a muon detector. Physics variables were smeared according to the typical resolution of the subdetector. All events survived a L1 filtering algorithm. Note that the simulation data used is provisional and do not represent in full details the detector behaviour. Therefore, the results we present can only be taken as estimates of the actual performance of the proposed triggering system. More realistic data is being prepared.

The trigger decision analyzed with the techniques described here involves particle identification only. The identification of physics channels by combining the information contained in the ROIs is under development. Results obtained using another approach can be found in [5]. Network simulations were performed by JETNET 2.0 package [6] and backpropagation was the training algorithm used.

A Single Network Approach

Figure 1 shows a set of powerful decision variables that was fed into the network. The process $H \rightarrow ZZ \rightarrow 2e2\mu$ was

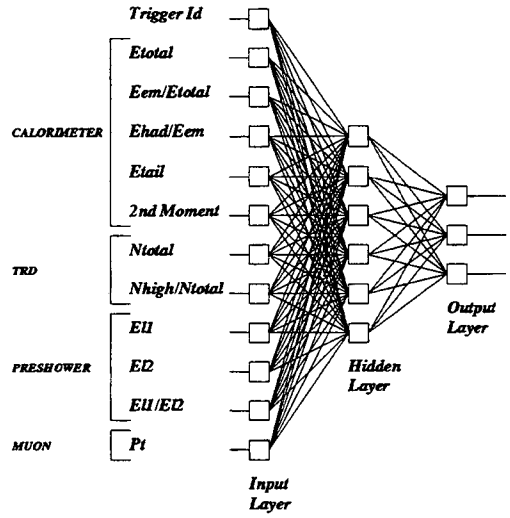


Figure 1. A single neural network for implementing the global decision unit.

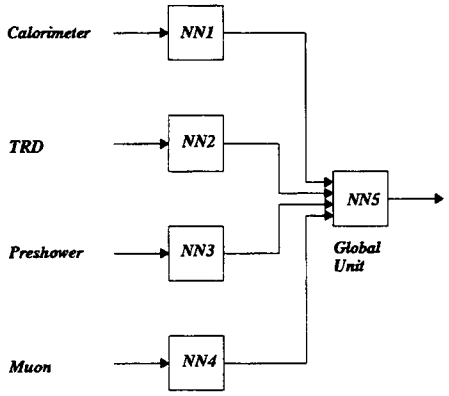


Figure 2. Expert Neural Network Topology.

used to train and test the network. The training file was built by using half of the total number of events and it comprised 1000 electrons, 500 jets and 1000 muons. Testing was performed on events not used during the training phase. For interpreting the output vector of the network, the maximum probability criterion was used. The achieved performance is shown in Table 1.

Particle	Identified as		
	electron	jet	muon
electron	99%	1%	0%
jet	3%	97%	0%
muon	0.1%	0%	99.9%

Table 1. Results for a single network topology

Expert Networks

An alternative design for the global unit is to break the required multidimensional analysis into pieces and make a set of networks to perform the resulting subtasks. In order to do that, we can train one network for each detector and use a global one to combine outputs of these expert networks. Figure 2 shows this scheme.

In order to identify the most important variables for the triggering task, the relevance criterion [7] was used:

$$R(x_{i,j}) = \frac{\sum_{i=1}^{N_{patterns}} (out(x_i) - out(x_i|x_{i,j} =< x_j >))^2}{N_{patterns}} \quad (1)$$

The first term in the summation is the network output for each input pattern and the second one is the output when the component j in the pattern vector is replaced by its mean value. The summation is carried over all patterns forming the training set. Figure 3 shows the relevance for calorimeter features.

For the expert network design, each specialized network was trained and tested using the respective detector data for the process $H \rightarrow ZZ \rightarrow 2e2j$. Only the most relevant detector features were considered. Calorimeter and TRD contributed with three features each, preshower with two, and the muon detector was not taken into account. The output of each of these networks was fed into a 3-3-1 global unit. The overall system was able to identify 98.8% of electrons

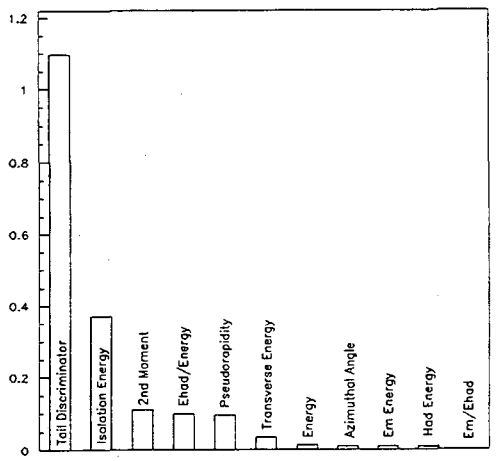


Figure 3. Relevance for calorimetry.

and 97.1% of jets correctly. It should be mentioned that the discrimination power of each detector was not deteriorated by using the reduced set of features selected by the relevance criterion, instead of all available.

Conclusions

A global decision unit design based on neural networks was proposed. Using a single network fed with 12 features from different detectors, 99% of electrons, 97% of jets and 99.9% of muons were identified.

An expert neural network approach was also developed for designing a more compact global unit. It is based on training one network for each detector and feeding network outputs into a global neural network. Applying the relevance criterion for reducing the dimension of the feature space and using expert networks, 98.8% of electrons and 97.1% of jets were correctly classified.

Acknowledgements

We are thankful for the support that has been provided to this work by CNPq (Brazil) and CERN. We would like to thank the RD11 Collaboration for providing the simulated data sets. In particular, we express our best thanks to R.K. Bock, F. Block, W. Krischer, I.C. Legrand and J. Carter, all from CERN, for fruitful discussions concerning this work, and E. Ribeiro for helping in part of the analysis.

REFERENCES

1. J. Badier et al. IEEE Trans. on Nuclear Science **40** (1993) 45.
2. J.M. Seixas et al. A Second-Level Trigger System Based on Calorimeters and Using Neural Networks for Feature Extraction and Electron/Jets Discrimination. IV Int. Conf. on Calorimeters for High En. Phys. Elba, Italy, 1993.
3. J.M. Seixas et al. Neural Networks Applied to a Second-Level Trigger System. III Int. Workshop in Softw. Eng., Artificial Intell. and Expert Sys. for High-En. and Nucl. Phys. Oberammergau, Germany, 1993.
4. R.K. Bock et al. Test Data for the Global Second-Level Trigger. CERN/EAST 93-01, 1993.
5. R.K. Bock et al. Real Time, Data Driven Architectures Simulated in Concurrent C++ for the LHC Second-Level Trigger. Oberammergau, Germany, 1993.
6. L. Lönnblad et al. Comput. Phys. Commun. **70** (1992) 167.
7. A. Gruber and J. Moeck. Performance of Backpropagation Networks in the Second-Level Trigger of the H1-Experiment. Oberammergau, Germany, 1993.

Modelling of Local/Global Architectures for the Second Level Trigger at the LHC Experiment *

Zbigniew Hajduk, Wieslaw Iwański, Krzysztof Korcyl
Institute of Nuclear Physics, High Energy Physics Department
30-055 Krakow, ul. Kawiory 26 A, Poland
John Strong
Royal Holloway and Bedford New College (University of London)
Egham Hill, Egham, Surrey TW20 0EX UK

Abstract

Different architectures of the second level triggering system for experiments on LHC have been simulated. The basic scheme was local/global system with distributed computing power. As a tool we have used the object-oriented MODSIM II language. [4]

INTRODUCTION

At maximum LHC luminosity events arrive with a frequency of 10^9/sec. However, interesting physics is rare and data reduction of 10 orders of magnitude is necessary. We have assumed three levels of triggering and worked on possible solutions for the second level.

NEW CONCEPTS FOR SECOND LEVEL TRIGGER

The general structure of the architectures studied is based on the 'local/global' [1] idea. There are two new concepts in triggering at level two aimed at reducing the load, bandwidth and computing power needed :

- Regions of interest;
- Feature extraction.

A Region of interest (RoI) is an area of the detector where trigger level one (T1) has given a positive result. We assume that level two (T2) makes use of this geographical information and concentrates its acivity only in RoIs.

The feature extraction concept is to trigger on well defined, relatively simple physical features of interesting events such as : high energy lepton(s), high P_t muons, high energy jets , etc.

FUNCTIONAL PARTS OF T2

The level two can be split functionaly into three areas :

- Feature extraction; within RoI (local detector activity)
- RoI object building; features from different detectors are correlated within one RoI (local RoI activity)
- Trigger event building; correlation of all RoI objects gives a trigger event with decision. (global activity)

We have split our studies into two steps:

- detailed model of the local processing board, embedded into a simple global system;

*This work has been supported by KBN grants *115/E-343 SPUB-20693* and *2 P302 047 06*

- general architectures with parameters of the local board established by former studies.

LOCAL SIMULATIONS

In general the RoI area does not match the area covered by a module. We have to provide a means of horizontal communication between processors to ensure the data exchange for full RoI coverage.

This basic building block of the system [2] contains one or more TMS320C40 DSP processors serving as a FeatureExtractor and a BoardManager. The DSP has been chosen because of its computing power and 6 communication links.

Former studies have shown [3] that the data from events accepted by T1 should be stored in the T2-buffer immediately after every positive T1 decision. Our board contained a T2-buffer and could communicate and exchange the data with other boards via links provided by DSPs. The full system forms a 'mesh' covering the surface of detector.

We have investigated the following issues:

- average and peak latency of the event i.e. time from positive T1 decision until readiness for global processing;
- average and peak processing time for variable number of processors on the board;

The following conclusions can be drawn from our studies (Fig. 1 and 2):

- processing units should be built as farms of several processors to decrease significantly local queues;

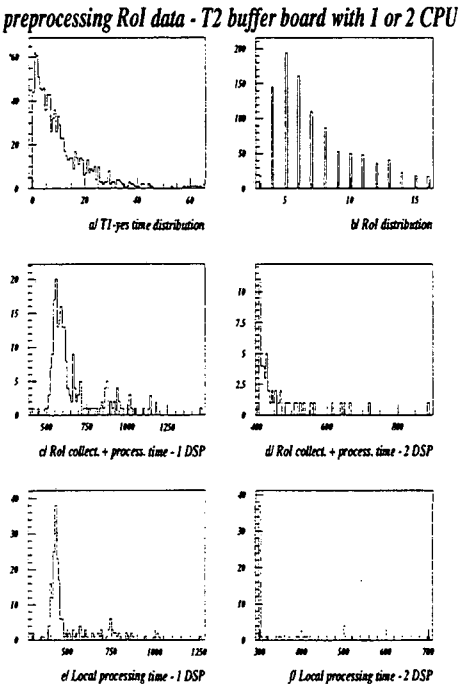


Fig. 1: Input and results for board studies

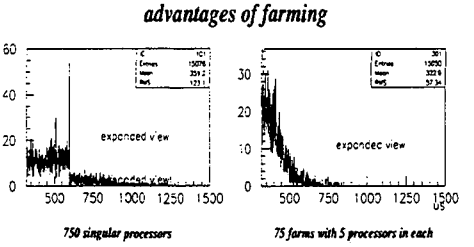


Fig. 2: Local processing time distribution

- efficient management of T2-buffer is very important.

GLOBAL SIMULATIONS

In our studies of the overall architecture a system composed of the following components has been simulated (Fig. 3):

- *T2 buffer*, where in some cases feature extraction might be executed;
- *Local Switch*, where data was distributed to a local farm;
- *Local Farm*, where both feature-extraction and object building might be performed;

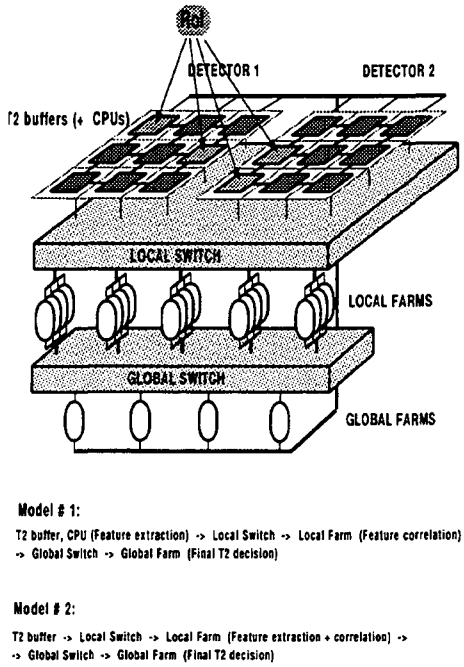


Fig. 3: Model of the LHC asynchronous T2 architecture

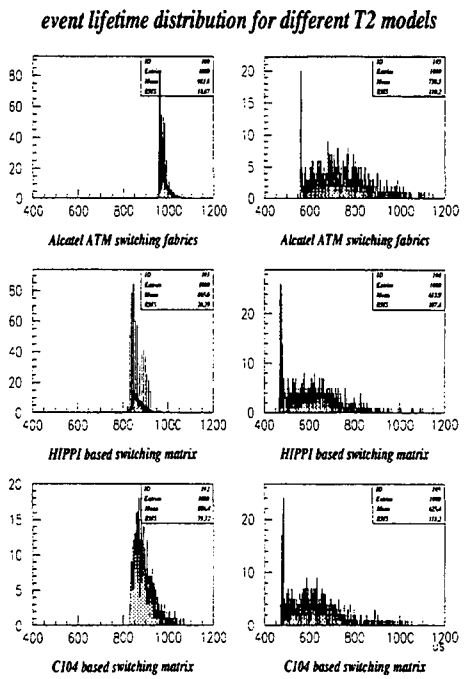


Fig. 4: *left:* raw RoI (0.5Kb)-model 2 *right:* preprocessed RoI(32/48b)-model 1

- *Global Switch*, distributing the data to a global farm;
- *Global Farm*, where final trigger algorithm is run.

For the switches we have modelled switching networks built from C104 transputer links, commercial HIPPI and ATM-Alcatel [1] switches .

CONCLUSIONS

We draw the following conclusions from our studies (Fig. 4):

- none of the simulated switches prove any special superiority.
- a flexible and intelligent event identification management is important;
- the best performance is exhibited by the system where 'feature-extraction' is performed in the farm AFTER the Local Switch .

MODSIM II has proven to be a very useful tool for behavioural simulations.

[1]The ATM-Alcatel has been treated as a 'black box' and the model was simplified, no traffic shaping was implemented

REFERENCES

1. D.Crosetto et al.: A local/global architecture for level 2 calorimeter triggers, LHC Workshop, Aachen, 1990
2. P.Clark, Z.Hajduk, K.Korcyl, W.Iwański et al: The FEAST project - draft - EAST note # 93-11 CERN 13 August 1993
3. G.Appelquist, W.Iwański: Simulation of a calorimeter read-out system using VHDL, EAST note # 92-27, FERMI note # 13, CERN November 1992
4. CACI Products Company - MODSIM II Reference Manual

FAST SECOND LEVEL TRIGGER USING A NEURAL NETWORK ARCHITECTURE FOR THE H1 EXPERIMENT AT HERA

P.Ribarics, J.Fent, W.Fröchtenicht, A.Gruber, C.Kiesling, T.Kobler, J.Möck
Max-Planck-Institut für Physik, Föhringer Ring 6, 80805 München , Germany
D.Goldner, H.Kolanoski, T.Krämerkämper
Institut für Physik, Universität Dortmund, 44221 Dortmund, Germany

Abstract

At the HERA e-p collider the expected machine background rates are 10^5 times higher than the rates from physics. The greatest challenge in the trigger is finding methods which suppress the machine background without using lengthy pattern recognition algorithms. Our present investigations show that feed-forward networks, trained on the topological energy sums from the H1 calorimeter and on first level tracking trigger information, provide an additional background suppression factor.

OVERVIEW OF THE H1 TRIGGER

Background Suppression

The H1 multi-level trigger scheme will face the task of reducing an expected interaction rate at HERA of about 0.5 MHz down to a data logging rate of a few Hz. This huge rate is dominated entirely by the machine background coming from the strong interaction of the proton beam. The two main background components are the interaction of protons with the residual gas in the beam-pipe and the loss of the proton beam due to scattering on different elements of the beam-line.
The rate coming from electro-weak physics is much lower. For neutral current events beyond $Q^2 > 10\ GeV^2$,a rate of about 3 Hz is expected. If the machine background events are not filtered out efficiently at the lower trigger levels, they saturate the trigger system and give rise to large dead-time losses.

Multi-level Trigger System

The dramatic reduction factor of about 10^5 will be achieved in a total of four levels: Two levels of hardware (L1 and L2) which are intended to bring the rate down to 100 Hz, and two levels of software (L3 and L4), which will have to further reduce the rate to the foreseen data logging rate of 5 Hz. Thus by far the greatest reduction factor has to be achieved in hardware, with an additional requirement of minimising of deadtime.
L1 is dead-time free and based on discriminator output and hard wired logic only. The output rate must be less than 5 kHz so that the dead time introduced by this level is less than 10 % for a level 2 timing of 20 μsec. L1 receives information from the calorimeter, the central

tracking chambers and from the time-of-flight counters. The L1 event selection is based on successive discrimination (3 thresholds) using the topological energy sums from the calorimeter which are subsequently combined with the L1 tracking quantities to make the so-called L1 trigger cocktail.
This complex trigger cocktail gives a rough selection in the multidimensional space of the trigger quantities. There is a general belief in the collaboration that it is tuned to its limits, harder cuts would already deteriorate physics. To go beyond the scope of L1 we have to exploit the correlations between the various trigger quantities.
Due to the tight time requirement for the L2, only massively parallel algorithms can be employed to analyse the abundant trigger information. Such algorithms seem to be most efficiently realized with neural networks which map the trigger patterns on their inputs to an output representing the trigger decision. The internal parameters of a neural net are fitted on a sample of events for which this assignment is known (training).

L2 PERFORMANCE

The neural networks were trained with the Aspirin/MIGRAINES software package using the available trigger informations. The tracking system delivers histograms about the intercept of tracks with the beam axis (z-vertex histograms) and track multiplicities. The calorimeters give 8 bit energy information for 252 towers pointing to the interaction zone. We also obtain the tranverse energy and missing transverse energy, the number of electron candidates in the main calorimeter and information from the backward single electron trigger. From the muon system the trigger will get hit multiplicities. TOF measurements from two scintillator walls in front of the detector allow to discriminate between particles coming from the interaction point or from beam-wall interactions. The electron tagger system detects electron energies from γ–proton events scattered at very small angles.

Training with Real Data

At the last level (L4) of the H1 trigger system it is decided to keep or reject an event. The L4-rejected events can be used as a background sample to train networks. The training was done with two sets of input variables, a small one with 58 quantities, using the liquid argon calorimeter, track triggers, backward electromagnetic calorimeter, muon wire chambers and the time-of-flight veto system. The large set with 70 variables used information from the electron and photon tagger and some more quantities in addition which are not available at the second level so far.
The results of the best 3-layer feedforward networks trained on selected physics events against the L4-reject sample are shown in Tab. 1. The numbers describe the efficiencies on the physics files in percent and show the percentage of rejected background events. A realistic L2-rejection factor is calculated from the L4-reject sample.
In the case of neutral current events with low Q^2 we reach a high efficiency for physics and rejection rates of the desired order of magnitude. The increased performance for the γ–proton network with 70 variables can be traced back to the utilisation of the electron tagger which

Tab. 1. Network performance on real data

	58 quantities			70 quantities		
trained on	background	physics	rejection	background	physics	rejection
NC (low Q^2)	99.2	99.4	7.5	99.2	99.8	11.2
NC (high Q^2)	94.0	92.1	5.8	95.3	93.2	6.9
Charged Current	89.7	89.7	4.6	86.5	90.1	4.5
γ-proton	91.2	89.6	6.2	99.7	99.4	40

was crucial for the selection of these events. The network for charged current physics has low efficiencies and rejection factors. Reasons for this are the soft cuts for the sample selection leading to a charged current sample with a lot of background events.

HARDWARE REALISATION

Since the H1 trigger information is entirely digital it seems most natural to consider a digital implementation. The digital technique is preferred on grounds of stability, controllability and reproduceability. A digital VLSI neural network chip has become available recently (fast enough to meet the 20 μs time constraint for level 2): the CNAPS chip from Adaptive Solutions (USA). Adaptive Solutions have in their program a VME board built around the chip, together with the necessary software. The limitation of 8 bits for the input into the network in the CNAPS chip is not a problem since most of the trigger information is in 8 bit accuracy. Other information not available in 8 bit form could be transformed appropriately in a "data box". A sufficient number of chips is arranged on a VME board ("NN-board") which can model a neural net with a maximum of 64 inputs, 64 hidden nodes, and 1 output node. A VME crate houses the NN-boards, one board for each physics reaction or for a background region. Each NN-board has a companion data server board ("data box") which supplies exactly the foreseen trigger information for the neural net modelled onto the board. Both the data boxes and the NN-boards are loaded from a host computer via microprocessor controller boards. A 20 MHz data bus, divided in 8 parallel groups, each 16 bits wide, provides the trigger information from the various subdetectors to L2. The transfer is divided chronologically into two parts: During the first 3.2 μs all data with fixed format (trackers, muon system etc.) are transferred, corresponding to a maximum of 32 words (16-bit) for each group. Then the zero-suppressed data (e. g. energies from the calorimeter) are sent. After 5 μs all the data is available. During the time of data transfer the data boxes select precisely the information from the bus which is needed for their companion NN-board.
Another 5 μs later the data is loaded completely onto the NN-boards and the data boxes give start signals to the NN-boards which present their outputs back to the data boxes in 10 μs. These in turn supply their signal to a Final Decider which allows rate limiting measures such as prescaling or bypassing of trigger decisions.

The Brookhaven National Laboratory Experiment 871 Data Acquisition System *

Robert D. Martin† and A. Dayle Hancock
Department of Physics, College of William and Mary, Williamsburg VA 23187 USA
David F. Connor
Department of Physics, University of California at Irvine, Irvine CA 92717 USA
Philip D. Rubin
Department of Physics, University of Richmond, Richmond VA 23173 USA

Abstract

A data acquisition system, capable of handling high data rates, has been developed for Brookhaven AGS Experiment 871. Carefully selected data from front end crates are loaded into dual port memory modules connected to 8 SGI V35 processors, where a software "event quality" trigger further reduces the data. The data are uploaded to a host IBM RS/6000 53H through two RPC client/server pairs.

INTRODUCTION

The Brookhaven AGS Experiment 871 is a search for very rare decays of the long lived neutral kaon. It is expected to reach a single event sensitivity of 10^{-12} in the searches for the decays $K_L \to \mu e$ and $K_L \to ee$ and in the measurement of the branching ratio for the decay $K_L \to \mu\mu$. Extraordinary discrimination is required to minimize the number of events being written to tape. This is accomplished through a multi-level trigger and custom built readout scheme, both of which have been upgraded from a previous experiment (BNL-E791)[1].

*This work is supported by the National Science Foundation, the US Department of Energy and the University of Richmond Faculty Research Committee.

†Representing the BNL AGS E871 collaboration.

READOUT SYSTEM COMPONENTS

The data acquisition system consists of two hardware triggers; 40 custom built "Fasterbus" crates with data sparsifying controllers; about 12,000 channels of custom built, event buffering front end electronics (6 bit, 2.5 ns least count TDCs; 8 bit, 220 ps TDCs; 8 bit charge integrating ADCs; and 96 bit coincidence Latches); a custom built readout supervisor; 8 VME crates, each with a SGI V35 processor and 5 Dual Port Memory (DPM) boards; and an IBM RS/6000 53H host computer. The readout supervisor, 10 CAMAC modules, controls interactions between the hardware triggers, front end electronics, the DPM boards and the SGI processors. The processors apply a software trigger to data in the DPMs and build events that pass the trigger. The events are uploaded over a dedicated Ethernet subnet to the IBM host computer, where they are buffered on disk and even-

tually written to one of two Exabyte 4200 DAT tape drives. The front end electronics and the readout supervisor are nearly the same as used in BNL AGS E791, [2, and references therein]. The DPM are new for E871 [3], as are all the computers.

THE DATA UPLOAD PROCESS

Overview

The data upload process involves two client/server pairs, DataReady and DataUp. The client/server communication utilizes Sun Remote Procedure Call library routines. The communication between the DataReady and DataUp routines employs UNIX SVR4 InterProcess Communication (IPC) objects.

When the DPMs are full, the controllers in the front end crates signal the readout supervisor, which in turn sends a START signal to an available processor, which then applies the software trigger. The DataReady client maps events that pass the trigger to a segment of SVR4 Shared Memory. A RPC call is made to the DataReady server (known as the "dispatcher") on the IBM. This call informs the dispatcher that the processor has a buffer full of data that is ready to be uploaded. The dispatcher also serves as the DataUp client. In this role, it makes a RPC call to the DataUp server on a SGI. This server copies the data out of a shared memory segment into a "result" structure. This structure is then sent over the Ethernet to the IBM and written to a run file. Multiple run files are buffered on disk until TAPESIZE is exceeded. The files are then written to DAT tape.

The DataReady Client

The upload clients on the SGIs, DataReady, read the DPMs, build events and trigger in addition to notifying the IBM that data are ready to be uploaded. On startup they open and clear their DPMs, open the interrupt handler for the START and attach to the various IPC objects before starting a `setjump/longjump` loop. Inside the loop, the routine polls for the kernel flag set by the START interrupt. The `longjump` is invoked only at the end of acquisition.

Once the START is received, the software trigger is applied to the data. Events that pass this trigger are fully extracted from the DPM and placed into an event buffer. The event buffer is then attached to a shared memory segment. The size of the event buffer is placed into a SVR4 message queue where it will be extracted by the DataUp server. The RPC call is then made to the dispatcher. Once the reply from the dispatcher is received, the DataReady client returns to polling for the START kernel flag.

The Dispatcher

The dispatcher on the host computer is both the DataReady server and the DataUp client. In addition, it communicates with the readout supervisor, the online slow controls and with the user interface.

When an upload request arrives from one of the processors, a DataUp RPC call is made to the server on that processor, which returns the data and the length of the data. The data are written to the run file and the length of the data is added to the running count of the run size. A dummy reply is then sent to the client. The readout supervisor is notified that the processor is ready to ac-

cept more data. Next, a CAMAC read of an input register, where bits are set for the end-of-spill (EOS) signal and high voltage trips, is done. If the EOS bit is set, the spill scalars in a CAMAC crate are uploaded and written to the run file. If one of the high voltage trip bits are set, the run is paused (the readout supervisor will not accept any triggers) and the user is notified. After the slow control processing is finished, the dispatcher returns to polling for upload requests. If the run size is greater than 200MB, the old run file is closed and a new run is begun. The user is notified of this change.

The communication with the user interface is done through UNIX signals, a message queue and a pair of FIFOs. When the user changes the run state, a run word is placed in a message queue and SIGUSR1 is sent to the dispatcher. The dispatcher then reads the word and changes the state of the readout supervisor and run file accordingly. The dispatcher writes the run size to a FIFO on every upload, where a X11/Motif based GUI (dubbed Xacq) reads the FIFO and writes the size on the interface. When one of the high voltage trip bits is set, the dispatcher writes a word describing the trip to another FIFO where Xacq reads it and informs the user. When the run size exceeds 200MB, the dispatcher sends SIGUSR2 to Xacq.

DataUp Server

The DataUp servers on the SGI processors poll for the DataUp request from the dispatcher. When a request is received, the data member of the RPC result structure is attached to the shared memory segment. The size of the data buffer is also placed in this structure. The result structure is then sent across the Ethernet to the dispatcher.

MONITOR AND SLOW CONTROLS

Several control and monitoring task are also performed by the host computer. Over 1100 channels of high voltage are controlled and periodically readout by the system through CAMAC interfaces. High voltage trips on wire chambers are automatically reset during running. Scanning ADCs monitor several hundred experimental parameters. These data are calibrated and displayed to the experimenter in real time as well as inserted into the data stream to tape.

REFERENCES

1. Arisaka, *et al.* "Improved Sensitivity in the Search for the Rare Decay $K_L^0 \rightarrow e^+e^-$" *Phys. Rev. Lett.* **71**,3910 (1993).
2. K. Biery, "The Data Acquisition System for BNL AGS Experiment 791", *Proc. Conf. on Computing in High Energy Physics*, Sante Fe, NM, April 1990.
3. D. Connor, R. Atmur and W. Molzon, to be published in *Nucl. Inst. and Meth.*

b-TAGGING USING SHAPE VARIABLES IN THE HADRONIC DECAYS OF THE Z^0 *

G.Cosmo, A.De Angelis,
CERN, Geneva
L.Lyons,
University of Oxford, England
A.Saccavini, F.Stringhetta,
University of Udine, Italy

Abstract

The performance of b-taggers, based on Multidimensional Analysis applied to variables related to the shape of hadronic events, is analyzed on a sample of about 166,000 hadronic decays of the Z^0 collected by the DELPHI detector at LEP during 1991. The classification efficiency is computed from the data, by using tagging on hemispheres.

INTRODUCTION

We discuss the possibility to extract a discriminant variable from variables related to the shape of a multihadronic event, and to evaluate its performance in an almost simulation independent way, using a feed-forward Neural Network as a multidimensional classifier (for a review of results recently obtained see Ref. [3] [4] [5]).

The strategy [2] is based on the independent tagging of suitably defined hemispheres in which events are divided. Thus, an almost Monte Carlo independent tagging can be obtained, paying the price of a worsening of the performance.

We report here on preliminary results obtained in this direction.

*This work is supported by the DELPHI Collaboration at CERN, Geneva, and by the ANNETTHE Collaboration.

EVENT SELECTION

The sample of events used in this analysis was collected during 1991 by the DELPHI detector at the LEP e^+e^- collider.

A description of the apparatus can be found in Ref. [6]. Features of the apparatus relevant for the analysis of multihadronic final states (with emphasis on the detection of charged particles) are outlined in Ref. [7].

The influence of the detector on the analysis was studied with the simulation program DELSIM [8]. Events were generated with the JETSET 7.3 Parton Shower Monte Carlo program [9] (JETSET PS in the following) with parameters tuned as in [10].

THE DOUBLE TAGGING METHOD

The major drawback of multidimensional methods is that the dependence on

the simulation introduces on the physical observable a systematic error which is harder to estimate compared to the single variable case.

In order to overcome this problem, a double tagging method was used in the present analysis.

Hadronic events were split into two hemispheres according to the plane perpendicular to the thrust axis, then only hemisphere-defined variables were considered. Let ϵ_b be the probability of tagging a hemisphere in a b event, and ϵ_l the same probability in u, d, s, c events. In the hypothesis in which the two hemispheres are statistically uncorrelated the following equations hold:

$$f_1 = \epsilon_b R_b + \epsilon_l (1 - R_b) \quad (1)$$
$$f_2 = \epsilon_b^2 R_b + \epsilon_l^2 (1 - R_b) \quad (2)$$

where f_1 is the fraction of tagged hemispheres and f_2 is the fraction of events in which both hemispheres are tagged. The quantities ϵ_b and ϵ_l can be determined from the equations above, by assuming $R_b = 0.217$ as predicted by the Standard Model.

It would be possibile to estimate R_b using a third equation coming from a lepton tagged sample but this possibility will not be considered here.

The correlations between hemispheres, and possible differences in efficiency between the u, d, c, s quarks, modify equation (2) in such a way that

$$f_2 = \epsilon_b^2(1+c_b)R_b + \epsilon_l^2(1+c_l)(1-R_b) \quad (2')$$

The coefficients c_b and c_l have to be determined by simulation:

$$c_j = \frac{f_{j,2}}{f_{j,1}^2} - 1 \quad (3)$$

where $j = b$ or l.

The purity π of a selected sample of b-tagged events (i.e., of events in which both hemispheres are tagged as b) can be obtained from ϵ_b and ϵ_l through

$$\pi = \frac{\epsilon_b^2(1+c_b)R_b}{\epsilon_b^2(1+c_b)R_b + \epsilon_l^2(1+c_l)(1-R_b)} . \quad (4)$$

USED VARIABLES

Among many variables which have been considered for feeding the network, we have chosen ten variables on the basis of their discriminating power: sphericity and aplanarity, sum of momenta and sum of products of momentum components, sum of the p_T^2, longitudinal momentum, invariant mass, directed sphericities. Impact parameters have been excluded because our intention is to combine our results with the one in Ref [1] using impact parameter variables.

THE NEURAL NETWORK

We adopted a feed-forward Neural Network architecture trained with Back-propagation. Both the architecture and the learning procedure have become by now quite well known, so we refer to the literature for details (see [11]), just giving here a brief sketch of them and a list of the choices made for the parameters.

The Neural Network we used has 10 input variables, strictly coming from the shape of the event. There were 2 hidden layers of 10 and 8 nodes and one output node. The weights were updated every 10 events. The learning and momentum parameters were geometrically decreased and increased respectively during the training. The system was trained on a set of 20,000 simulated events equally divided between the two classes. Each event is described using variables related

to a single hemisphere. By monitoring the behaviour of the generalization error, the training was stopped after 400,000 updates. The performance can be assessed in terms of signal efficiency and purity. The test sample consisted of about 50,000 simulated events, and 50,000 real data taken during 1991 using the DELPHI detector.

Each event is classified by testing the Network separately on both the hemispheres and considering an event as $b\bar{b}$ only if it is classified as b on both the hemispheres. The curve purity versus efficiency for the selection of b quark pairs is given in Figure 1. The black markers refer to real data, while the solid line is for the simulation. Purity and Efficiency are computed using the solutions of the equations (1) and (2') described in the previous Section.

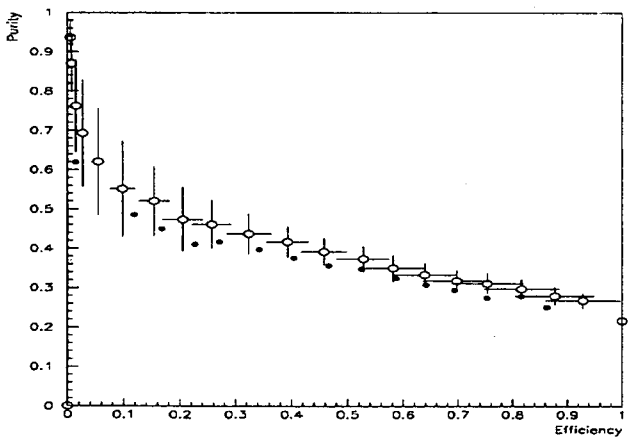


Figure 1. Purity vs efficiency from the NN using real data and the double-hemisphere method.

SUMMARY AND CONCLUSIONS

Using the independent information coming from suitably defined hemispheres in jet events proves to be a viable strategy to obtain a multidimensional tagging method based on Neural Networks, which is less sensitive to Monte Carlo assumptions than previously achievable.

Further analysis is required to better quantify the influence of the correlation coefficients on the classification in terms of efficiency and purity; this work is presently in progress.

REFERENCES

1. G.V. Borisov, DELPHI 94-27, March 1994; V.Castillo et al., DELPHI 93-75 (1993), submitted to the Marseille Conference.
2. ALEPH Collaboration, CERN-PPE/93-113, July 1993.
3. *New computing techniques in physics research II*, D. Perret-Gallix ed., World Scientific 1992.
4. *Neural networks: from biology to high energy physics*, World Scientific 1993.
5. *Computing in high energy physics*, CERN 92-07, Geneva 1992.
6. P. Aarnio et al. (DELPHI), *Nucl. Instr. and Meth.* **A303** (1991) 233.
7. P. Aarnio et al. (DELPHI), *Phys. Lett.* **B240** (1990) 271.
8. *DELSIM Reference Manual*, DELPHI 89-68 PROG 143, CERN, September 1989.
9. T. Sjöstrand, Comp. Phys. Comm. **27** (1982) 243, ibid. **28** (1983) 229; T. Sjöstrand, *PYTHIA 5.6 and JETSET 7.3*, CERN-TH.6488/92, September 1992.
10. W. de Boer et al., IEKP-KA/91-07, Karlsruhe, June 1991.
11. J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley 1991.

Local processing for a farm-based second level trigger at LHC

John Strong
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

Abstract

A design for part of the second level of a three level trigger scheme for ATLAS is described. The units needed to provide the system are discussed and plans for demonstrations summarised.

1. Introduction

The trigger systems for experiments at the Large Hadron Collider (LHC) must meet the challenge of reducing an initial interaction rate of up to 10^9 Hz to a rate which can be written to storage media for off-line analysis. For the ATLAS experiment [1] it is proposed to use a three stage trigger system. Simulation work indicates that the first level trigger should provide a rate reduction to about 40 kHz. The design goal for the second level is to accept an input rate of at least 10^5 Hz and reduce this to 10^3 or less.

2. The second level trigger

The second level trigger must retain all 'interesting' physics events and those required for calibration, while rejecting the rump of the high-rate processes which pass the first level. Events passed to the second level will closely resemble good triggers and can only be rejected by performing analyses not possible at level 1.

Second level trigger processing may be divided into several stages. Data for analysis must be moved to a processor and then analysed to produce features (e.g. tracks parameters). Related features from different sub-detectors (e.g. a track and a calorimeter cluster) can then be combined and finally information from different areas of the detector evaluated to provide an overall decision. Because it operates on a limited volume of the data, feature extraction is a local operation and global data access is not required.

Considerable benefits in data transfer and processing rates arise if information from the first trigger level can be used to guide level two. Only data from regions of interest (RoIs) in sub-detectors need be moved and processed. The first level trigger can provide these pointers by indicating the position of all significant energy deposits in the calorimeter and high energy tracks in the muon chambers.

In ATLAS, data will be moved to buffers off the detector after the first level trigger. The latency of the second level trigger is determined by the capacity of these buffers which must hold the data during the level two decision time. It is estimated that a total buffer capacity of 1 GB will be needed for each millisecond of latency at a Level 1 rate of 100 kHz. Latency of a few milliseconds is, therefore, not a problem.

3. Local and global processing

A latency of several milliseconds means that programmable devices can be considered as the processing units in a second level trigger. The trigger discussed here is based on a continuation of the work on a local/global architecture described at the Aachen workshop on LHC [2]. Features extraction is performed in the local part of the architecture and feature combining and topological assessment is performed in the global area.

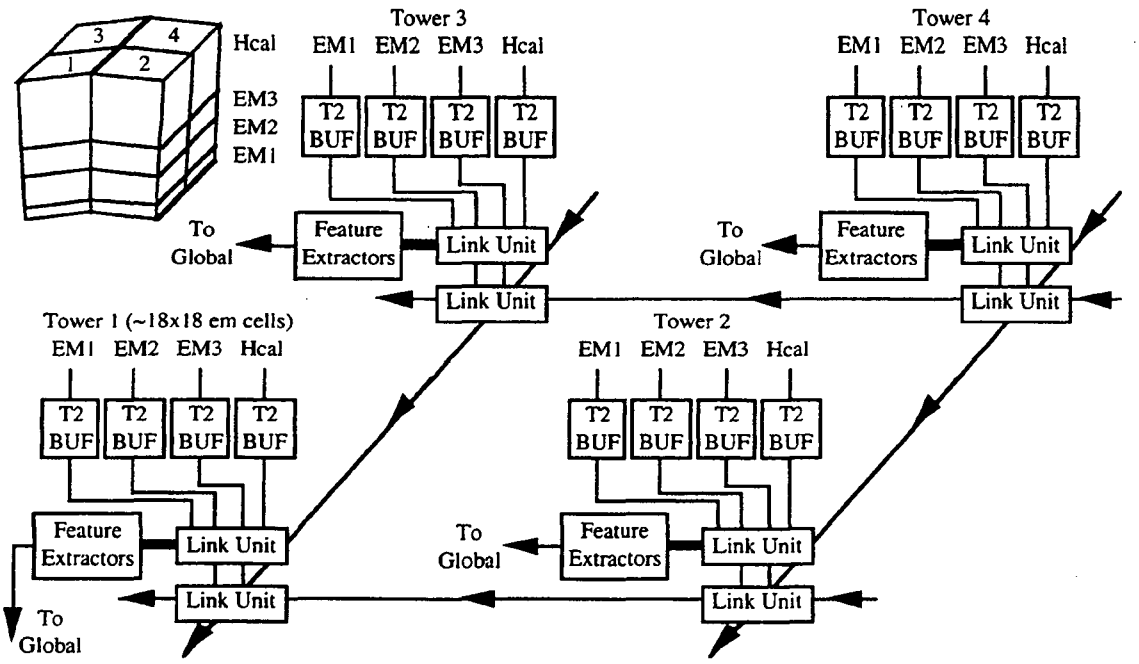


Fig. 1. Local processor network for a calorimeter

It is assumed that digitised data from the front-end electronics arrives at the second level buffers (T2 buffers) on links operating at about 1 Gbit/s to limit the number of buffers. For a buffer with 1 MB of memory about one thousand events can be stored which allows a second level latency of up to 10 ms.

Table 1. Basic calorimeter parameters

calorimeter	252x252 cells/layer
e.m.cell size	0.0249x0.0238
e.m. layers	3
hadronic layers	4
T1 cell size	4x4 e.m. cells
trigger cells	3969
buffer coverage	324 (18x18) cells
buffers	784 (4x196)
crates (T2bc)	49 (784/16)

The information from one sub-detector is spread across many buffers but the data corresponding to one RoI is limited to only a few. For a calorimeter with three electromagnetic layers of granularity 0.025 by 0.025 ($\Delta\eta$, $\Delta\phi$) and four hadronic layers of 0.05 by 0.05, with data from 324 (18x18) cells connected to one buffer, the data for a RoI would spread over four buffers in each layer at most (see Fig. 1)

Table 2. Basic trigger parameters

trigger rate	100 kHz
RoIs/event	5
RoI size	12x12 e.m. cells
RoI data size	1.15 kB
with identifiers	1.38 kB
Total RoI data rate	688 MB/s

For feature extraction the RoI data from all layers must be collected into one processor and the network shown in Fig. 1 is sufficient for this task. One link unit concentrates the data from one tower and another provides the mesh connection which routes the data to the appropriate feature extractor (FeX).

Table 1 shows the basic parameters for a calorimeter, table 2 gives the trigger rates and table 3 the rates at various points in the system for the architecture shown in Fig. 1.

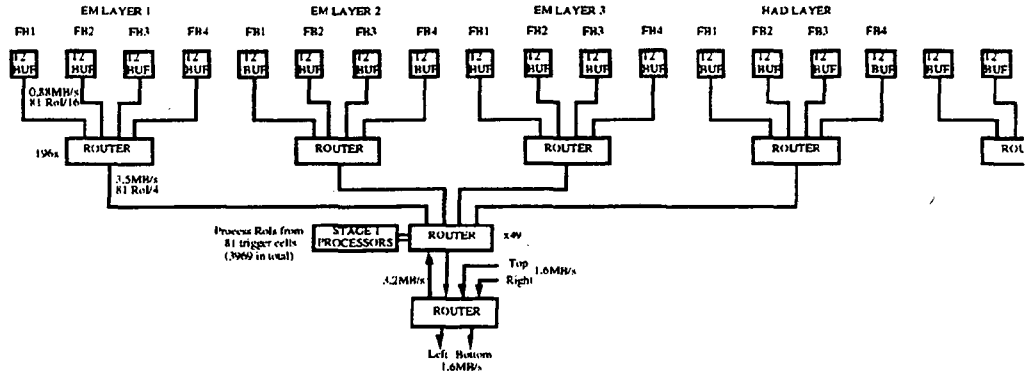


Fig. 2. Local processor network with added stage of data concentration

An estimate, using a simplified monte carlo, of the number of RoIs from the Level 1 trigger gives a mean value of about two [3]. However, five RoIs is used to provide a reasonable safety margin.

Table 3. Rates for Fig 1.

from T2 buffer	~1 MB/s
into FeX	~3.5 MB/s
FeX event rate	2.6 kHz
Link units	392
Link boards	98 (49 in T2bc)
FeX (farms)	196
FeX boards	98
Crates	8

To cope with queuing problems and 'hot spots' in the detector, the feature extractors can be expanded to small farms.

Table 4. Rates for Fig. 2

between routers	3.5 MB/s/link
into FeX	14 MB/s
FeX event rate	10.2 kHz
Link units	294
Link boards	74 (49 in T2bc)
FeX (farms)	49
FeX boards	49
Crates	4

Fig. 2 shows an architecture where an additional stage of data concentration has been added. Table 4 gives the data rates for this arrangement where it is assumed that each FeX is a small processor farm.

4. Demonstrations

Based on a commercial board using the TMS320C40 DSP, a T2 buffer with 100 MB/s data input acceptance has been built and tested. It will be used this summer in a test beam at CERN as part of a minimal test of the architecture described above. The link unit and feature extractors will be similar DSPs. The global network wil consist of a small SCI ring and Alpha single board computers. The results obtained will allow comparison with modelling studies [4] and the development of modelling techniques.

5. References

[1]. ATLAS Letter of Intent.
CERN/LHCC 92-4
[2]. D. Crosetto et al.: A local/global architecture for level 2 calorimeter triggers. LHC Workshop, Aachen, 1990
[3]. D. Johnson and J. Strong.: A study of a calorimeter based trigger system using single electron and two jet events. ATLAS CAL-NO-030 October 1993
[4]. Z. Hajduk et al. : Modelling of local / global architectures for the second level trigger at the LHC experiment. Proc. of this conference.

Paradigms and Building Tools for Real-Time Expert Systems

Ulf Behrens, Mariusz Flasiński, Lars Hagge, Kars Ohrenberg
Deutsches Elektronen-Synchrotron (DESY)
22603 Hamburg, Germany

Abstract

An expert system is a software which can simulate the problem solving behaviour of a human expert. The rule-based paradigm is chosen to describe the different aspects involved in expert system development. Differences between expert systems and common procedural or object-oriented programs are investigated. Expert system shells are introduced as a building tool for expert systems, together with some guidelines on the evaluation of such shells. A discussion of special needs for real-time expert system development concludes the paper.

INTRODUCTION

Modern High-Energy Physics experiments usually involve large international collaborations which operate complex detector equipment to perform an extensive experimental program. In terms of expert knowledge this has three consequences:

- expert knowledge is widely distributed among lots of experimentalists, institutes or countries.
- different areas of knowledge have to be combined (eg detector hard- or software, experiment operation, research program)
- individual experimentalists have only incomplete expert knowledge available

On the other hand, an experiment can only be successful if any expert knowledge is available whenever and whereever it is needed. A way to achieve this for experiment operation is the use of an expert system to control and analyze the experiment.

COMPONENTS OF AN EXPERT SYSTEM

An expert system is a software which can simulate the problem solving behaviour of a human expert. According to the tasks of human experts, expert systems can be split into several components:

- The *knowledge base* often contains a common data base which is holding all the (factorial) data occuring in the problem domain, but it also stores the (executable) problem solving knowledge required by the inference engines.

- An *inference engine* interprets and applies the problem solving knowledge.

- The *explanation component* is closely linked to the inference engine. Its task is to explain the design and the different steps of the reasoning process.

- *User interface* and *knowledge acquisition component* should be based on commonly available toolkits, in order not to distract the development team from the focus of the expert system.

RULE-BASED SYSTEMS

Rule based systems store knowldege in terms of rules, where rules are expressed as if-clauses, ie *if-<condition>-then-<action>*. Depending on the implementation language, condition and action can contain tests and modifications of facts, calls to pre-defined or user-provided functions for statistical evaluations etc.

Fig. 1 illustrates the operation of a rule based system. First, the antecedents of all rules are *matched* to the facts. This results in a *conflict set* of applicable rules. According to a conflict resolution strategy (eg breadth/depth-first searches, priorities, context switching) one rule is *selected* from the conflict set for *execution.* This may result in modifications of the data base. The process is iterated.

It is also possible to match the consequents instead of the antecedents to find out the condition under which a given fact may have occured. This inference strategy is called backward chaining (cp forward chaining).

There are striking differences between rule-based and common procedural programming languages:

- Procedural languages are based on sequence, branching and iteration. This enables for every program the definition of a data flow and a control flow. Rule-based languages operate on a set of equally-righted rules. Rules are evaluated and queued when the database has changed. Thus, a "new" program is created for every situation.

- In common programming languages, program sequences can be grouped into procedures as a structuring tool. Rule-based languages may support the concept of a context to switch rules on or off for evaluation.

- Performance of procedural languages can be increased by exploiting concurrency or real-time parallelism. Those techniques can be applied to rule-based systems as well. However, performance of rule-based systems is strongly dominated by conflict resolution and search strategies.

It can be observed that rule based knowledge bases are the method naturally chosen by humans to express their knowledge (eg manuals, briefings) and are thus the easiest to maintain in an evironment of non-professionals in computing.

EXPERT SYSTEM BUILDING TOOLS (ESBTs)

Expert system shells have become popular for expert system development. An expert system shell can be regarded as an empty expert system, which provides

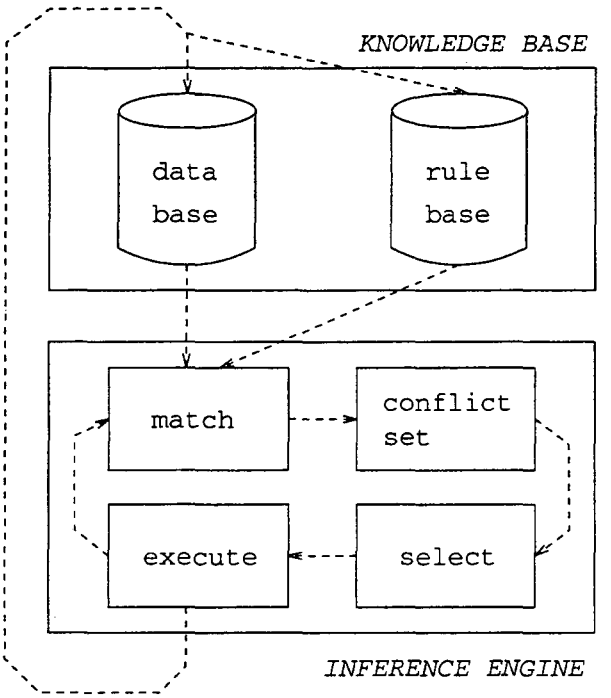


Figure 1. Rule-based system

the developer with knowledge acquisition and maintenance tools, inference engines and a UIF toolkit. Powerful expert system shells are hybrid, ie support several knowledge representation paradigms.

The data flow diagram fig. 2 shows a possible architecture of an expert system. A good shell would require the developer to only fill the knowledge base and the rule base to obtain a running expert system.

EVALUATING ESBTs

Expert system shells show big differences in their features, performance and cost. However, it is difficult to make benchmarks or general evaluations of ESBTs as they strongly depend on the application of the tool. Instead, this section lists some important aspects which need to be considered when evaluating ESBTs:

- *knowledge base / facts:* availability of frames (object oriented representation), ringbuffers (history over time) and dynamic objects
- *knowledge base / rules:* symmetry of left hand and right hand side, availability of wildcards (general rules) and generic rules (rules to create rules)
- *inference engine / selection:* support of contexts and/or priorities, multiple bindings, multiple queuing, possibility to choose between or mix different search strategis
- *inference engine / execution:* availability and performance of tests for sets of facts/unknown facts, temporal reasoning, power of callable function library
- *tools:* mouse and click interfaces, availability of an automatic explanation component, graphical UIF building tool

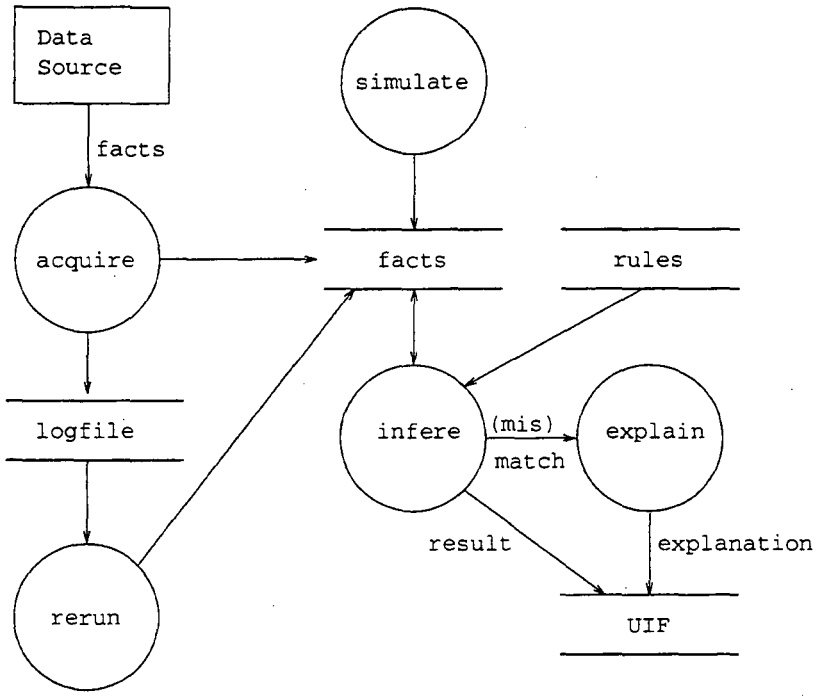


Figure 2. Possible architecture of an expert system

- *testbed:* availability of logfiles, simulation and replay facilities, debugging support
- *architecture:* open, distributed, multiple incarnation of components, power of interfaces to existing software
- *licensing:* availabilty for platforms, portability, cost

REAL-TIME EXPERT SYSTEMS

Real-time expert systems require the reasoning to deliver a result within a determined time. This can be achieved by introducing a scheduler which interrupts endless inference procedures, thus lowering the requirement of receiving "the optimum solution" to obtaining "the best solution in time". Alternatively, if the inference is to be continued until the best solution is found, a "delay" of the system to its expected response time can be measured. This strategy is usually pursued by real-time expert system shells. To keep the delay small, a careful design of the rulebase with the goal of obtaining short searches is mandatory.

REFERENCES

1. P. Jackson, *"Introduction to Expert Systems"*, Addison-Wesley, Reading, MA, 1990.
2. L. Brownston et al., *"Programming Expert Systems in OPS5. An Introduction to Rule-Based Programming"*, Addison-Wesley, Reading, MA, 1986.

Switched Interconnection of Parallel Processes (SWIPP) used for detector data readout and partial event-building for LHC

Morten Lømo, Oddvar Søråsen
University of Oslo, dept. of informatics, P.O.Box 1080, N-0316 Oslo, Norway

Abstract

This paper presents a high-performance data acquisition architecture based on SWIPP. SWIPP (Switched Interconnection of Parallel Processors) is a high speed packet switching network being developed at the University of Oslo. An introduction to the main characteristics of SWIPP is given. Then follows a description on how SWIPP can be used for data readout and partial event building for the planned microstrip inner detector in the LHC (Large Hadron Collider) ATLAS experiment at CERN. Finally an implementation of a partial event buffer which meets the requirements of the above application, is described in some more details.

1. Introduction

The total LHC data rate from the detectors is expected to be several Tbytes/s [7]. Only a small fraction of this will reveal interesting physics results and should be stored on tape. The rest will be discarded in a step by step manner by the use of trigger levels. In parallel to this, event building takes place; i.e. data which belong to the same event will be assembled. Traditionally this has been done by using a time-division muliplexed (TDM) approach of bus based acquisitions systems (e.g. Fastbus, VMEbus). However, due to the high bandwidths of the LHC experiments, this approach will no longer be possible. SWIPP offers an alternative approach. It is scalable, data driven and based on a high speed parallel switching fabric.

A general description on how SWIPP can be used for data readout and event-building in an LHC inner detector experiment, has been presented in [3]. This paper carries this idea further by examining in more details some architectural and implementational aspects related to the partial event buffer for the second level trigger.

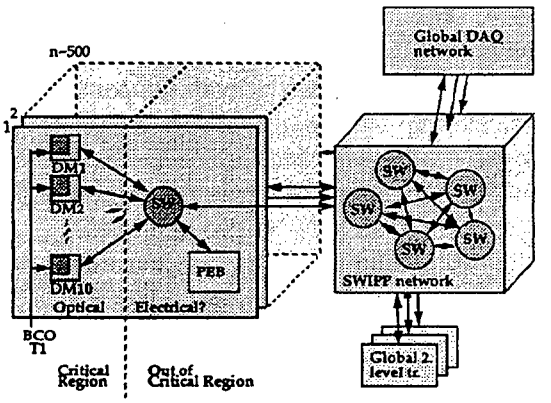


Figure 1. Overview of the SWIPP readout network

Figure 1 shows a SWIPP readout network. The particles are detected by the silicon strip detector modules (DMs) [6]. Data which pass the 1. level trigger and the zero level suppression on the DMs, are pushed out on the SWIPP network, one link for each DM. A switch (SW) collects data from several DMs into one Partial Event Buffer (PEB). Based upon Region of Interest (RoI) and 2 level trigger information, data are moved out from PEB into higher levels in the data acquisition system.

2. Overview of SWIPP

SWIPP is a packet switching network [3]. Each link is bidirectional and can operate at up to 1 Gbit/s in each direction. Data are packed into packets, each having a header field which contains routing information for the switches and a data field of variable length.

SWIPP is an interconnection for heterogeneous nodes, e.g. processor nodes, memory nodes, transducers etc. In the SWIPP concept these nodes are called Compute Engines (CE). They interface to the SWIPP network by a Protocol Engine (PE). The PE assembles/disassembles the packets and relieves CE from communication tasks.

Inside the SWIPP network there may be any number of switches (SWes). The SWes are non-blocking 16 x 16 xbar switches. No address lookup table is needed since the address is found directly from the packet. This reduces the complexity of the switches and speeds up the switching since it can be done 'on-the-fly'. An input buffering approach has been selected in order to provide queuing necessary to handle bursty traffic. A mathematical analysis of this approach can be found in [5].

Flow control is implemented in the SWIPP network so that the buffering can be distributed in the network and loss of data can be avoided.

3. Partial Event Buffer (PEB)

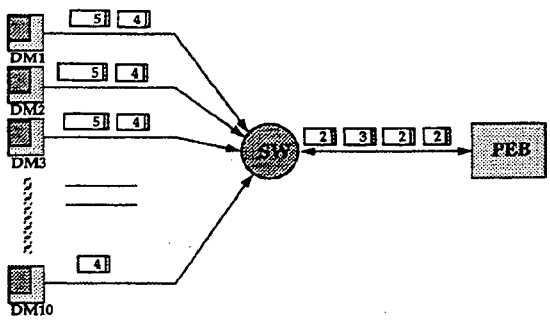


Figure 2. Data from DMs into PEB

In figure 2 we take a closer look at how SWIPP packets are pushed out from the DMs and are multiplexed by a switch (SW) on a single link to PEB. According to simulations in [1] and [2] the average amount of data we can expect from one DM related to one single event is 46 bytes. This relative small amount of data advocates for packing the whole data event from one DM into one single SWIPP packet. Each packet is identified by its event number which is placed in the header of the data field. Figure 4 shows that DM1 has sent out one packet with data from event number 4 followed by one packet with data from event number 5. Similarly the other DMs are sending out packets. Due to the different sizes of the packets and possible timing skews, the packets may arrive out of sequence with respect to event number to PEB. PEB must cope with this so that packets associated with the same event number are assembled in memory.

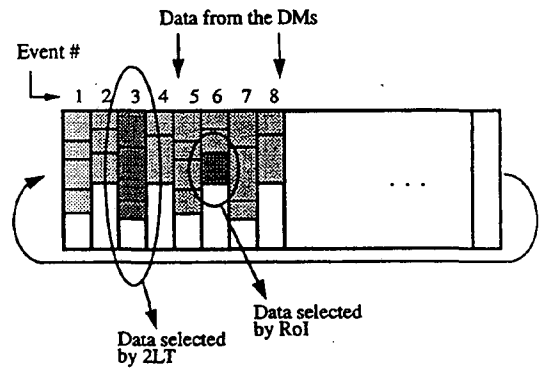


Figure 3. The main memory accesses of PEB

The memory of PEB is divided into buffers of equal sizes, each buffer being large enough to contain all packets with the same event number; i.e. one partial event. PEB must store the partial events until the 2. Level Trigger (2LT) decisions arrive. Partial events which are not selected by 2LT will be overwritten by new events. As a result PEB should be seen as a ring buffer. There must be more than 100 buffers. This comes from the fact that the average 1. Level Trigger (1LT) decision rate is 10^5 Hz and the average 2LT decision rate is 10^3 Hz;

i.e. there is on average one 2LT-decision for every 100 1LT-descisions.

Partial events which are selected by the 2LT must either be kept in memory or be moved to higher levels of the data acquisition system (figure 3). Also, certain packets, which are identified as Region of Interests (RoI), shall be moved to the local 2LT.

4. Implementation of PEB

The PEB must be able to receive and transmit packets to the SWIPP network at a high rate. Data should be copied as few times as possible in order to minimize the overhead. The PEB must be able to assemble packets which contain data from the same events. The partial events and the individual packets of the partial events must be stored in a systematic way so that they can quickly be accessed.

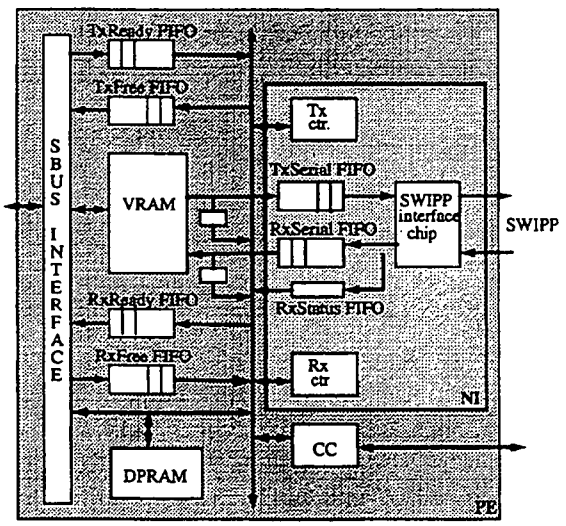


Figure 4. Implementation of PEB

Figure 4 shows how PEB can be realized in hardware. The memory of PEB is built up of a triple port Video RAM (VRAM) much like the Afterburner in [4]. The VRAM is selected because it provides one standard random access port and two high speed serial ports; i.e. it offers multiple high bandwidth ports. The memory is split into buffers of equal sizes. Two FIFOs, RxReady and RxFree, are used as buffer descriptors for input data and two buffers, TxReady and TxFree, are used as buffer descriptors for output data. A buffer descriptor contains the start address and the size of the data block which is stored in the buffer. An ASIC called the SWIPP interface chip, contains the high speed part of PEB. It handles the low-level SWIPP protocol. The remaining part of PEB is implemented in programmable logic devices and standard components. Packets are received from the SWIPP fabric by the SWIPP interface chip and written to two FIFOs, RxSerial and RxStatus. A RxController controls the addressing of the VRAM in such a way that partial events are built up in memory. When a partial event has been completely built up, a reference to this buffer, i.e a buffer descriptor, is pushed onto RxReady FIFO. This can now be read by an Sbus master. Data transmission is analogous. The TxController reads buffer descriptors from TxReady, and uses these to address the VRAM. Data are read from the VRAM via TxSerial FIFO and the SWIPP interface chip and output on the SWIPP link.

5. Summary

This report has described how SWIPP can be used to read out data from the microstrip inner detector of ATLAS and perform partial event building. An implementation of a partial event buffer has been presented. SWIPP could also be used in other parts of an LHC experiment.

[1] Gaarder, P. E., "A description of a simulation program for the front-end electronics of the SITV", Internal report, Dept. of Physics, Univ. of Oslo, 1992

[2] Nilsen, F. B. "Application of a switched interconnection concept for instrumentation at a high-energy physics experiment", Cand. Scient. theses, Dept. of informatics, University of Oslo May 1993.

[3] Søråsen O., et al, 'A High Performance Data Driven Packet-Switching Network for Detector Data Readout and Event-Building

in a LHC Detector Experiment', Eigth Conference on real-time computer applications in nuclear, particle and plasma physics, Vancouver, 8 - 11 June 1993, p 20-25

[4] Dalton, C. et al, "Afterburner", IEEE Network, July 1993

[5] Hluchyj, M.G. and Karol, M.J.,"Queuing in High-Performance Packet Switching", IEEE J. Sel. Areas in Comm, vol. 6, no. 9, Dec 1988.

[6] Hall, G. et al., "RD20 Status report - Development of High Resolution Si Strip Detectors for Experiments at High Luminosity at the LHC", CERN/DRDC 92-28, CERN, May 1992.

[7] The ATLAS collaboration. "ATLAS, Letter of Intent", Technical Report CERN/LHCC 92-4, CERN, May 1992.

DATA LOGGING AND ONLINE RECONSTRUCTION IN H1

Patrick Fuhrmann, Ralf Gerhards, Uwe Krüner-Marquis,
Jan-Erik Olsson, and Zbigniew Szkutnik
Deutsches Elektronen Synchrotron, Notkestraße 85, D-22603 Hamburg

Abstract

In spring 1992, the H1 detector at the HERA electron proton collider at DESY came into operation. The high bunch crossing rate and, correspondingly, the large data volumes are placing demanding requirements on the data logging and event reconstruction. Both tasks are performed on an SGI Challenge series computer. This note reviews the development and the experience with the data logging and online reconstruction in H1.

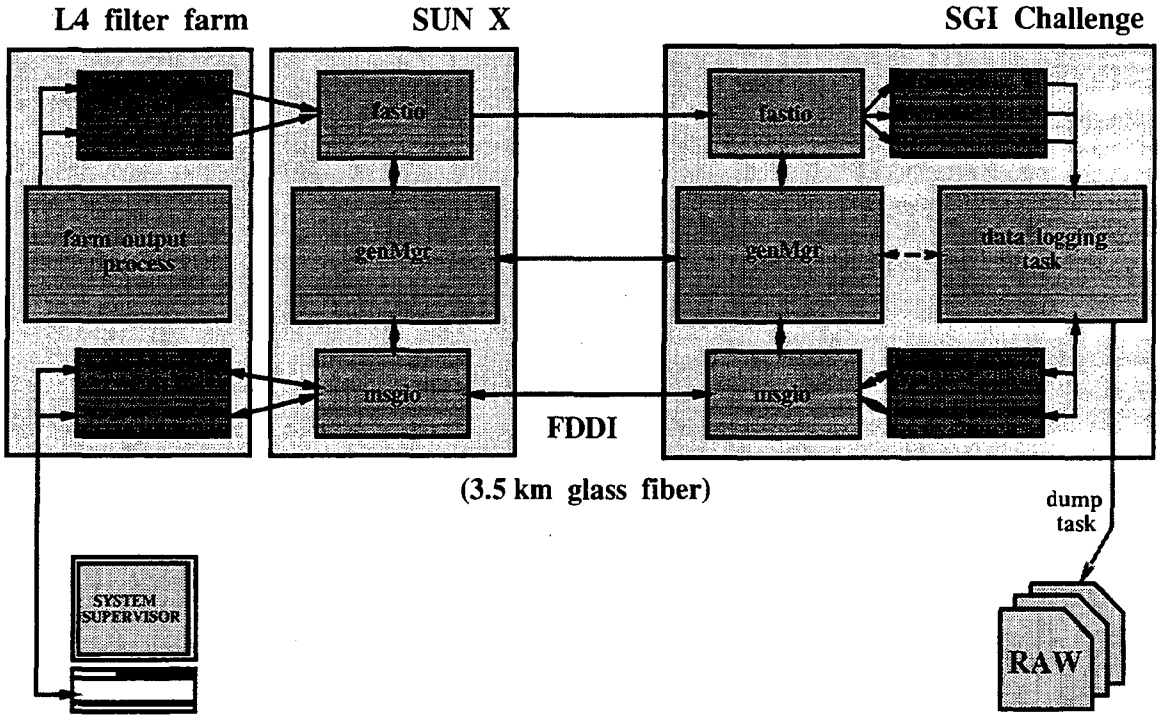
Introduction

About two years ago, the electron proton collider HERA at the DESY laboratory in Hamburg, Germany, delivered first luminosity to both the experiments ZEUS and H1. 26.7 GeV electrons are collided with 820 GeV protons at a bunch crossing rate of 10.41 MHz. Thus, HERA experiments are not only entering a completely new physics regime, but also facing new challenges in the area of triggering and data acquisition, data recording and data processing.

The H1 detector and its data acquisition system are described elsewhere [1, 2]. Various levels of hardware triggering, software filtering, and digital suppression are used to reduce about 3 MB of raw digitized information to acceptable recording rates. A parallel array ("filter farm") of MIPS RISC processors [3] is serving as a final software trigger to reject background events. During normal HERA operation, the data taking proceeds with an average rate of 10 Hz or about 500 KB/s. Assuming about 10^7 seconds of operation a year, this results in a yearly data volume of about 5 TB. This data is reconstructed almost parallel to the data taking with a delay of a few hours only, to be available for physics analysis as early as possible. The computing power needed for this *quasi-online reconstruction* is exceeding by far the capabilities of the DESY IBM mainframe computer. Hence, it has been performed from the beginning of the *ep* data taking on an SGI multiprocessor workstation based on RISC technology [4].

Recently, the data logging which was previously performed on the IBM mainframe computer was ported to the SGI Challenge series computer as well. An FDDI link using the standard TCP/IP protocol is used for data transfers.

The software to operate both the data



Schematic view of the data flow in the H1 data logging system

logging and the online reconstruction has been developed within the H1 collaboration together with the groups of the DESY computer center.

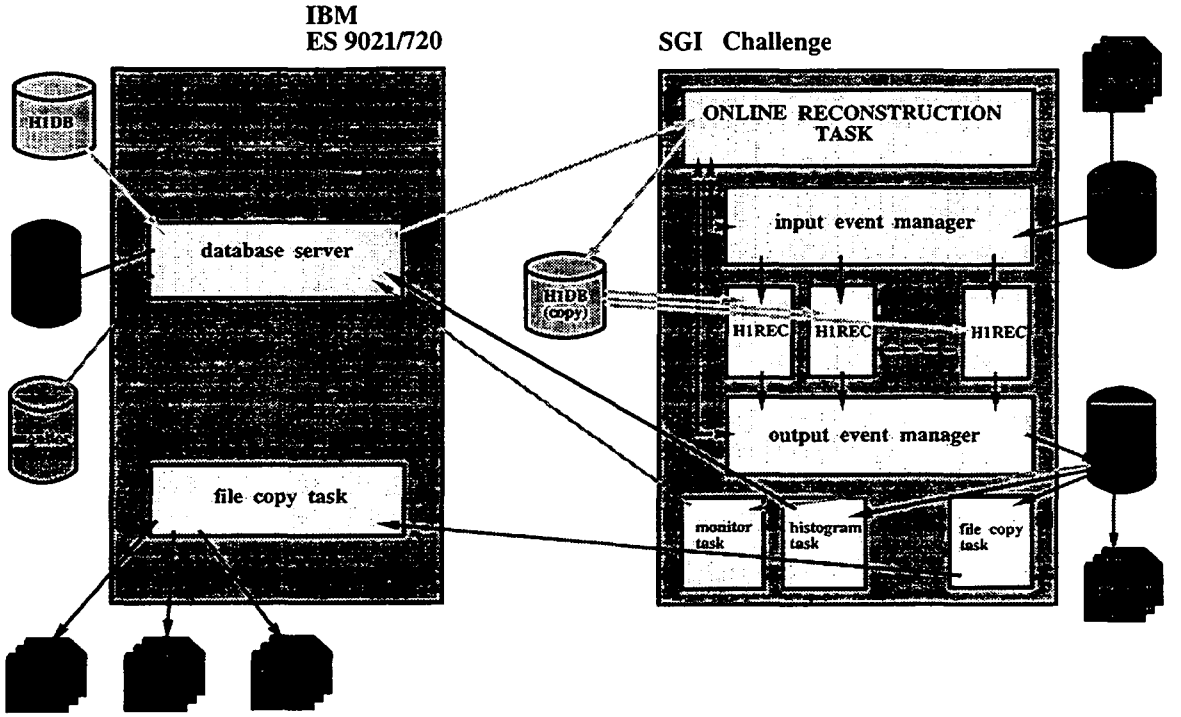
Setup

Figure 1 shows a schematic view of the data flow in the data logging system which can be divided into two independent tasks, the *data exchange* between the data acquisition system and the SGI and the *data logging task*.

Events accepted by the filter algorithm on the *filter farm* are collected by the *farm output processor* in two event buffers of 2 MB size. When such a buffer is filled, it is made available via VME to an I/O process running permanently on a dedicated *SUN SparcStation*. Data is sent via a fast *FDDI* link, using the standard TCP/IP protocol, to an *SGI Challenge* series computer with 10 MIPS R4400 RISC processors, and received by a corresponding I/O task. Three event buffers of 2 MB size are filled alternately from where the data is available for further processing. An *online event consumer interface* is used to communicate with the user application, i.e. the *data logging task*, using *shared memory areas*.

A parallel, independent path is used to exchange messages and requests between the data acquisition supervisor MacIntosh and the data logging task, e.g. status information which is displayed on the operator console.

The data is checked for internal consistency, converted to the standard H1 offline data format [5], and written to two disks, alternately. Whenever about 500 MB of data have been collected, a dump process is started to write data onto *raw data tapes*. An *Ampex DST 800* mass storage system with 3 DST 600 helical scan D2 recorders has been connected to



Overview of the data flow for online reconstruction in H1

the SGI to store the raw data [6]. This allows for more convenient and much faster access to this data.

Selected events for monitoring and calibration are copied out of the data stream. Important calibration information is determined already on the filter farm and stored by the logging task into the H1 database which is residing on the DESY IBM.

Figure 2 shows an overview of the data flow for online reconstruction. The *raw data tapes* provide the input to the *online reconstruction,* which is held until the best possible calibration constants for particular data are available. Therefore, the processing is typically delayed for a few hours.

The online reconstruction is performed on the same SGI Challenge computer as the data logging. When started, the *online reconstruction task* forks two processes for data input and output as well as a number of reconstruction processes, independently processing different events. A set of *shared memory areas* and *semaphores* is used to share data among processes. Each process is accessing a local copy of the H1 database which is updated regularly by the *online reconstruction task* from the master database on the DESY IBM, using a database server.

Accepted events are written alternately to two disks. Whenever about 260 MB of data have been collected, a process is forked that sorts the data by run and event numbers and copies about 200 MB to *production output tapes* (POT)[1] and the rest to an intermediate file which will be included in the next sorting process. This ensures that data is sorted globally throughout tape files. All files are stored in the Ampex mass storage system. As in the data logging, monitor-

[1] 200 MB is a convenient unit for distribution of data within the H1 Collaboration.

ing and calibration information produced in the reconstruction process is stored in the H1 database.

The complete system is controlled and supervised from the SGI. The status information is available via NFS mounted directories to other UNIX workstations within the cluster.

Performance

The data logging to the SGI was first applied in early spring 1994 over a period of several weeks, including cosmic muon runs. The main goal to record all data via the SGI was completely reached. Some performance figures are worthwile to discuss. The *transfer rate* using FDDI between the SUN X and the SGI Challenge, including TCP/IP protocol evaluation, is about 8 MB/s. Access to the VME memory on the filter farm side is supposed to reduce the rate to about 5 MB/s which would be the maximum possible data logging rate. Presently, about 1.8 MB/s was reached including the data logging task, i.e. data formatting, disk access etc.

Tapes in the Ampex storage system can be written with more than 10 MB/s from memory and about 3 MB/s including disk access. The capacity per tape amounts to 25 GB, the total robot capacity is 6.2 TB. During five days of data taking, about 200 GB of data were recorded at an average rate of 1 MB/s.

The performance of the online reconstruction has been measured during 1992 and 1993 data taking (using a 6 processor SGI 4D/460) and during a reprocessing from raw data early this year (using the present setup of a 10 processor SGI Challenge). The average reconstruction time was about 1.5 seconds per event on a 150 MHz R4400 processor, making use of optimization of the reconstruction program code. This resulted in a maximum processing rate of 6 Hz by making full use of 10 such processors. The event size is roughly doubled in the reconstruction. The event reconstruction program has proven to be quite stable. To catch occasional failures (occuring once in about 10^5 events) a signal-handler was implemented.

To keep the amount of reconstructed data reasonably small, a filter step has been included in the reconstruction program. The resulting data volume was of about the same size as the raw data sample. Because of relatively long breaks between luminosity periods it was up to now no problem to catch up with the data taking and all data acquired by the H1 detector has been made available for physics analysis within a few hours. At future data taking at higher luminosity, however, a more restrictive filtering will be needed and is foreseen.

The reliability and the performance of the various hardware components is impressing. Occasional problems with the Ampex mass storage device could be handled by making additional copies of all raw and reconstructed data files to IBM 3490 cartridges in a *StorageTek Automatic Cartridge System*. These copies were anyhow needed because the access to the Ampex tape robot is not made publically available to all H1 users.

Conclusion

During the first operation in spring 1994, the *data logging* to the SGI was reliably working. The potential increase in performance with respect to data transfer, data storage, and data processing has been demonstrated. The *online recon-*

struction was able to cope with the rate of H1 data taking in 1992 and 1993. Keeping in mind the expected increase in luminosity delivered by HERA, further improvements are needed to speed up the data processing and to reduce the data volume. It has been proven that both tasks can cope with the design parameters, i.e. a sustained data rate of 500 KB per second. The challenging requirements of data processing at HERA can be fulfilled by the present scheme.

Acknowledgements

The authors gratefully achnowledge the help of many colleagues from the DESY computer center in operating all the hardware involved and in developing the necessary software. The authors also express their gratitude to their colleagues in the H1 collaboration for many fruitful discussions which helped developing and improving this challenging computing project.

REFERENCES

1. I. Abt *et al.*, "The H1 Detector at HERA," DESY preprint 93-103 (1993), to be published in Nucl. Instr. Meth.
2. W.J. Haynes, "Experience at HERA with the H1 data acquisition system," Proc. Int. Conf. Comp. in High Energy Physics, Annecy, France (1992)
3. A. Campbell, "A RISC multiprocessor event trigger for the data acquisition system of the H1 experiment at HERA," Proc. Int. Conf. Real Time 1991, Jülich, Germany (1991)
4. R. Gerhards, Z. Szkutnik, "First experience with Online Reconstruction in H1," Proc. Int. Conf. Comp. in High Energy Physics, Annecy, France (1992)
5. V. Blobel, "The F-package for Input/Output", Proc. Int. Conf. Comp. in High Energy Physics, Annecy, France (1992)
6. M. Gasthuber, "Distributed Mass Storage and Management Systems at DESY", these procedings

FNAL E665 - A Case Study in Portable Computing *

Heidi Schellman
Northwestern University
Dept. of Physics and Astronomy
2145 Sheridan Rd
Evanston, IL 60208 USA
schellman@fnal.gov

Abstract

The Fermilab Muon Scattering Experiment, E665, has used highly portable code to maximize CPU resources by moving to faster systems as soon as they become available. E665 has been able to take advantage of the FNAL UNIX Farms and CLUBS systems to reduce and analyze 1 TB of data without custom designed hardware.

INTRODUCTION

Fermilab Experiment E665 is a muon scattering experiment which had three runs between 1987 and 1991. The combined data sample of $\approx$ 1 Terabyte had to be reconstructed, filtered and distributed to approximately 100 physicists at 18 institutions, each with unique local computing facilities, for analysis. When the experiment began, the complete offline reconstruction, simulation and analysis package was specified to run on Vaxes, the Fermilab Cybers and an IBM 4381 at MPI Munich. While writing portable code represented a sizeable initial effort, E665 can now move quickly to new operating systems when the opportunity arises. In 1990, the increased statistics expected in the 1990-91 run necessitated a move to the Fermilab Unix-based Farm systems for reconstruction.

*This work is supported by the A.P. Sloan Foundation and the U.S. Department of Energy under Contract No. DE-FG02-91ER40684.

The original conversion to UNIX occupied three experts for approximately 2 months, individual users could then make the transition in 1-2 days.

The E665 Experiment

E665 consisted of an open geometry spectrometer with muon identification and electromagnetic calorimetry[1]. Charged tracks with $x_F > 0$ were detected in 50-100 planes of drift chambers and PWCs, many of which were located in highly inhomogenous magnetic fields. The experiment has two major physics objectives, measurements of the inclusive scattering cross section (Structure Functions) and studies of the final state distributions in the inclusive event sample (Jets and Fragmentation). Both physics topics involve the total inelastic cross section, thus once the data sample has been reduced to of order 1-5 M events, many of the major analysis projects require the full sample. As a result substantial effort

has been put into defining micro-dst's and ntuples for analysis.

Data Sample Parameters

The final 1991 run of E665 logged 70,000,000 triggers to tape over a period of six months. The physics sample consists of 43M events on 843 single density 8mm tapes.

Table 1. 1991 Data Sample

	Events	Event Size	tapes
Raw Events	43M	18KB	843
Processed	15M	30KB	972
DST	12M	8KB	60
Sub-DST	6M	8KB	26
Beams	2M	4KB	17
Ntuple	15M	280B	4GB

Data Processing

Reconstruction

The events were reconstructed on the Fermilab UNIX Farms[2]. Each event took approximately 5 CPU seconds to reconstruct, with most of the CPU time spent in magnetic ray tracing. Between 24 and 48 nodes were used and production took a total of 6 months.

At each stage in the reconstruction, trigger failures were identified and rejected in order to save CPU time. On 1/10 of the tapes, the filtering was turned off; these tapes are used to monitor the filter performance. To ensure correct luminosity measurements in the event of data loss, special normalization records and prescaled unscattered beam events were retained with the physics triggers. In addition, a 70 word ntuple was written for each output event. This ntuple served both as a monitor of the reconstruction process and as the primary analysis sample for the structure function analysis.

Data Summary and Splitting

After reconstruction, the data were converted to DST format on the Fermilab CLUBS Unix Batch system[3]. This system consists of a user accessible front-end, FNALU, with 10 50-100 MIP batch nodes accessible through LoadLeveler. Data are staged from 8mm tapes into a 3480 cartridge silo and then to disk for processing. The system became available in early 1993 and E665 was the first major user.

The 60 2GB DST tapes were further split into eight physics subsamples ranging in size from a 2GB exclusive vector meson sample to the full 50 GB deep-inelastic scattering sample. The 50 GB DIS sample can be processed in less than 2 days of real time once the tapes have been staged to disk.

Once the data had been reduced to samples of less than 50 8mm tapes, these tapes were copied and distributed to several collaborating institutions where data analysis is performed on stand-alone workstations.

Code and Data Standards

The standard E665 environment[4] requires: FORTRAN 77-Standard, the current version of CERNLIB[5], ZEBRA[6] and the ability to read and write 8mm tapes. The full E665 code suite has run successfully on the following operating systems: CDC Cyber-NOS, DEC-VMS, AMDAHL/IBM - VM/SP, SGI-IRIX, IBM RS6000-AIX, HP-UX, DEC-OSF1 and DEC-ULTRIX. For amusement, a collaborator at Wuppertal has

implemented the E665 code on an 486 based IBM clone running UNIX, no 8mm tape I/O has been attempted.

To ensure portability, E665 data must be in one of two formats, ASCII for constants files and exchange format ZEBRA for data files. The only exception is analysis ntuples which can be transported through ZFTP.

Code management is done with the CERN PATCHY package. The E665 code is divided into 20 separate PAM files, with a total of approximately 500,000 lines. While PATCHY is emminently portable and supported on almost all platforms, it does introduce substantial overhead in code management. In E665 each PAM is the responsibility of a PAM Manager who ensures that changes are made properly and tested.

Platform independent I/O

ZEBRA and I/O devices are insulated from the user by general interface routines. When the code is migrated to a new operating system, only these interface routines and two short data files which link logical devices with disk and tape files need to be changed. As a result the same E665 user job can run on VM/VMS/AIX/IRIX with a single patchy switch change.

Conclusions

When CPU prices can drop by a factor of 2 every year; quick migration to new (and usually empty) machines, is a very cost effective way for a small group to gain access to computing resources that would otherwise be unavailable. By insisting upon portability as the software philosophy for the collaboration, E665 has been able to adapt quickly to changes in computing technology while collaborations with more sophisticated, system dependent, code and data management systems could not. The software methods used are simple, even crude, but effective.

ACKNOWLEDGEMENTS

The E665 collaboration would like to thank Brigitte Leupold and John Schell, who did much of the data processing, the Feynmann Computer Center Operations staff, the Farms group, the Large Systems Group (CLUBS) and the Unix System Support Group. Thanks also to the authors of CERNLIB and ZEBRA and the Fermilab CERNLIB support group.

REFERENCES

1. M. Adams et al, Nucl. Inst. and Meth., **A291**, 533 (1990).
2. "Clubs User's Guide", Fermilab Computing Division Note GU0007.
3. M. Fausey, *et al.*, "CPS User's Guide, CPS Version 2.9", Fermilab Computing Division Library GA009, June 24, 1993.
4. H. Melanson, *et al.*, "E665 Offline Software Users Guide, Version 2.0", April 15, 1986, E665 Internal note SW007.
5. "CERNLIB" , CERN Program Library Short Writeup.
6. "The ZEBRA System", CERN Program Library Long Writeup, Q100/101.

A NOVEL USE OF FORTRAN

Michael Metcalf
CERN, Geneva, Switzerland
Roland Windmolders
University of Mons, Mons, Belgium

Abstract

The introduction of the new Fortran 90 standard makes possible the maintenance of data structures without having to resort to auxiliary packages written especially for the purpose. As an experiment, a data structuring module, `eagle`, has been written in Fortran 90, making extensive use of pointers, dynamic storage, data hiding, explicit interfaces and recursion. It has been tested by using it to maintain the alignment files of the SMC experiment.

Outline of `eagle`

Earlier versions of Fortran have frequently been criticized for the absence of language features that enable programmers conveniently to establish and manipulate data structures. The new standard, Fortran 90, contains all the required features, such as pointers, recursion and dynamic memory allocation, that are required to program data structures directly. Its data hiding capability provides further safety. These features are fully described in [1], which contains also an extended example using them as an appendix.

As an experiment, a module containing many of the facilities required to support and manipulate tree structures has been written. It is called `eagle` and consists of less than 800 lines of Fortran 90. This demonstrates one of the remarkable features of the language—the succinct way in which it can be used to express algorithms. As an example, the complete code to traverse a tree deallocating all its associated storage consists of just five lines of executable statements.

For the purposes of this paper, a tree data structure consists of a set of connected nodes, arranged in levels. The top node points to some nonzero number of nodes at the second level. All other nodes point at zero or more nodes at the next lower level, and point back to exactly one node at the next higher level. The set of nodes pointed at by a single higher node is referred to as a layer. This is thus a standard tree of mother nodes each connected to a set of daughter nodes.

The module supports an arbitrary number of independent trees. They may be manipulated simultaneously.

At each node are stored, as far as the user is concerned: an index number supplied by the user (does not have to be sequential), the index of the node

that the node points back to (its mother node), an optional fixed character component (could be made variable), an optional integer array, an optional real array, and the optional links (pointers) to nodes with specified index numbers (the daughter nodes). The node contains also a running index number maintained by the module.

In a development version, an optional array of a fixed derived-data type may also be stored at each node, as well as a single reference link to another node (even in a different tree). A reference link is a pointer between any two nodes and is not part of the tree structure as such.

The user interfaces are:

start must be called to initialise a tree immediately before the first call to new_node for that tree (i.e. no reference to another tree may be made between these two calls).

new_node stores the data provided at the node whose index number is supplied as a second argument, and sets up pointers to all the specified daughter nodes that will be stored in subsequent calls. It makes some consistency checks.

retrieve retrieves a specified node; the data arrays (if present) are accessed via pointers, as are the links (pointers) to any daughter nodes.

next like retrieve, for the node with the next following running index number (not user index number).

next_in_layer like next, for the next node in the current layer. (In a sequence of calls, the next layer down will be automatically taken when the current one is exhausted, and this can be detected by a change in the value of the back pointer. If such a call follows one to next or retrieve the next layer is not taken.)

previous retrieves the data in the mother node of the last node accessed.

dump_tree write a complete tree to a specified unit.

restore_tree read a complete tree from a specified unit (initialisation is not required, even for a new tree).

set_reference establishes a reference link between a node of one tree and a node of the same or another tree.

get_reference like retrieve, but for the data at the node that is the target of the reference link at the specified node.

finish deallocate all the storage occupied by a complete tree.

These interfaces are simple. To add a new node to a tree, with a name, some integer data, and links to three daughter nodes, we write, for example,

```
call new_node(tree_name, index,  &
 node_name,                      &
 integer_data =                  &
          (/ (i, i = 1, 10) /), &
 links = (/ 2, 3, 5/))
```

and to retrieve data we write

```
call retrieve(tree_name, index,  &
          back, name, j, y, link)
```

where back is the index of the mother of the node index, and j, y,and link are pointers to any optional integer data, real data or links stored at that node. These names are chosen by the user, and retrieved data can be referred to directly, say as j(16).

Internally, the module allocates a node, and allocates data at that node if the corresponding argument is present in the call to new_node. Details of the node data type are in a fuller paper [2]; its internal structure is inaccessible outside the module.

Future plans are to add at least the following features: replace a node; add a node; extend a component; allow an error exit if insufficient storage is available for allocation for dynamic memory; and further validation and navigation facilities.

eagle in an application

This section shows how the description of the SMC (NA47) [3] detector given on an alignment file can be structured with the help of the new features provided in Fortran 90.

As an example, we defined two trees managed by eagle, one for proportional and drift chambers the other for hodoscopes, and use derived data types for several other parts of the data. They are described in the fuller paper [2].

Derived data types

Various derived data types have been defined for the description of the corresponding piece of equipment: target (polarised target), magnet (forward spectrometer magnet), calor (H2 calorimeter), absor (hadron absorber), and mgpos (last bending magnet in beam-line). An example is

```
type target
   real, array :: center(3),     &
                  par(4), dir(2)
   integer     :: year
end type target
```

Program use

For every block on the input file, the program prints a message telling whether it has been included in the structure or not. In general, blocks may be read in any order compatible with the implied logic: the list of detectors must appear before any particular detector information but data related to various chambers and hodoscopes may be interleaved.

Two special blocks related to the ST67 chambers have been included in the detector structure: the X positions (STWX) are added to the corresponding plane information (one additional number in the real data array), and the list of variable pitches (STPI) is added as an additional bank after the last plane.

Most of the blocks not treated so far can easily be connected to the existing trees: trigger hodoscopes, time calibrations and non-linearity parameters for drift chambers.

The main program contains several examples of how data can be found in the structures, either by using the general subroutine retrieve to search for a bank with a given index value, or by using a subroutine adapted for SMC structures only and searching for a block with a given name (subroutine srname).

Beam track fitting with eagle

As an application using a structure managed by eagle, we have rewritten

the procedure fitting the direction of the incident muons in the SMC experiment. Normally this procedure is included in the pattern recognition program (Phenix) and uses hits in two sets of hodoscopes separated by a distance of 10m, and in a small proportional chamber (P0B) located behind the last hodoscope. As a first step, the beam track parameters are determined by a least-squares fit using associated hits in the two groups of hodoscopes. These lines are extrapolated to P0B where corresponding hits are searched for within a given roadwidth and finally a new least-squares fit, using the three detectors, is performed.

In the working version of the SMC software, the data structure containing the hit coordinates in each plane is managed by ZEBRA [4] while all parameters related to the detectors, such as X positions, wire spacing and rms, are stored under various names in a large number of different common blocks.

The present application is set up as a simple test independent of the SMC software. Hit coordinates are read in from an input file and the detector information is fetched from the two trees managed by `eagle`. The trees are created in the initialisation phase by reading the standard SMC alignment file. The structures a well as the labelling and numbering conventions are described in Sections 2.1 and 2.2 of [2].

The general beam fitting processor (subroutine p0bhod) and the least-squares fitting routine (subroutine stfit1) have been rewritten in Fortran 90. In both cases the number of executable statements is reduced by about 25% due to the use of the new array features, which also makes the code more readable. Some results are given in the fuller paper [2].

Conclusion

We have successfully written a data structuring module using new Fortran 90 features, and used it to store, retrieve and manipulate data from the SMC detector. The code of both the module and the user program is simpler and clearer than the equivalent FORTRAN 77 code and, in particular, the programmer is relieved of the burden of managing data in memory, and is able to use mnemonic names of his or her choice. The fact that the interfaces to subroutines in a module are explicit means that errors in calling sequences are detected at compile time rather than giving rise to obscure run-time bugs.

The code has incidentally been used to test five existing Fortran 90 compilers: NAG, DEC, IBM, EPC, and Cray.

Future work will depend on other developments in the use of Fortran 90 for high-energy physics.

REFERENCES

1. Fortran 90 Explained, M. Metcalf and J. Reid, (Oxford University Press, Oxford and New York, 1993 printing).
2. A novel use of Fortran, M. Metcalf and R. Windmolders, CERN, CN/94/6.
3. SMC, B. Adeva et al., Phys. Lett. B302 (1993) 533.
4. R. Brun and J. Zoll, ZEBRA User's Guide, CERN Program Library, Q100, 1987.

SOFTWARE TOOLS FOR GAS-FILLED CHAMBER DESIGN

Pavel Pogidin*
Joint Institute for Nuclear Research
Dubna 141980 Russia

Abstract

Software for the computer aided design of gas-filled chambers has been developed. It can be extremely helpful in the development of new microstrip gas detectors for High Energy Physics.

INTRODUCTION

During the last several years new gas position sensitive detectors have been applied for fast and precise tracking on the new high luminosity colliders. The development of such detectors is a difficult task because many factors, such as construction parameters and applied materials properties, can affect their operation. To achieve outstanding time and spatial characteristics one should apply high technology and modern materials, as well as optimized detector design. For these reasons we suggest a program for computer-aided design of gas-filled chambers. The program for electric-field calculations in the microstrip gas counters is described in ref. [1]. The present version uses another field calculation algorithm and offers Monte-Carlo simulation of the detector operation.

*Present address: Department of Physics and Astronomy, The University of Iowa, Iowa City, IA 52242

THE SOFTWARE PACKAGE

The software package involves many features extremely helpful in detector design work.

The electric field

Using this package it is possible to compute the electrostatic field in the interior of the detector and plot corresponding field-line map. The program takes into account the influence of the surface charges appearing on the dielectic during detector operation.

The electrical properties of the detector

Capacitances of the electrodes and the signal induced by the avalanche on each of them may be calculated. The program may be used for study of leakage currents and interelectrode cross-talks.

Simulation of the characteristics

The time and amplitude characteris-

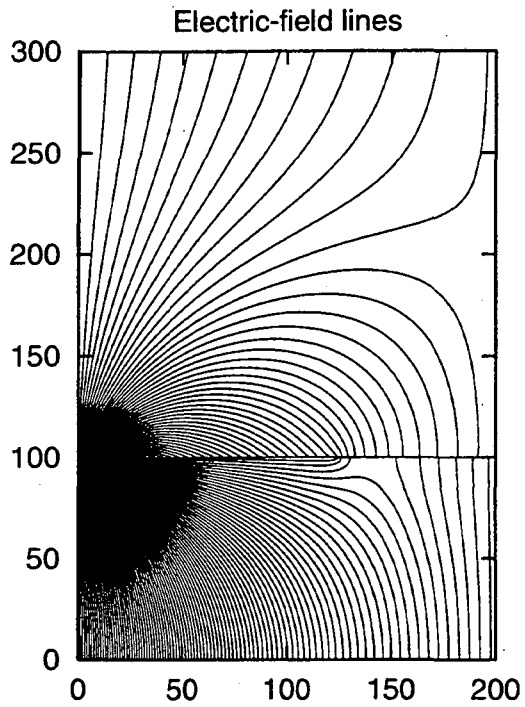


Fig. 1. Electric-field map in the microstrip gas chamber calculated by the program and created by GNUPLOT.

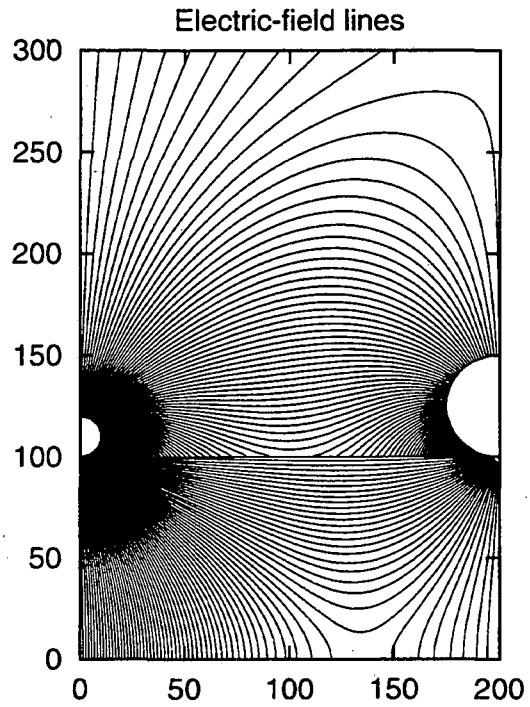


Fig. 2. Electric-field map in the multiwire foil gas chamber.

tics of the detector signal and the detection efficiency may be simulated by Monte-Carlo for various operational conditions.

The program allows a designer to analyze a wide variety of different detector geometries and dielectric substrate materials. The user is also able to:

i) specify the electrode voltages;

ii) choose parameters of the gas mixture;

iii) vary intensity of the particle beam.

THE USER INTERFACE

The user interface has been built using GNUPLOT software available on most computers including PC and SUN SPARC. The interface offers a user many convenient tools for displaying and printing graphical data. As an example fig. 1,2 show the electric-field map in the microstrip gas detector and multiwire foil chamber respectively, calculated by the program and created using GNUPLOT.

THE HARDWARE AND SOFTWARE REQUIREMENTS

The program can run on a wide variety of computers including PC 386/486, SUN SPARC and DEC ALPHA. For most applications 2MB of RAM is required. Software required for installation of the package includes the GNU C++ compiler and GNUPLOT.

THE SOFTWARE PERFORMANCE

The package contains a high performance program for the electric-field computations which has been written as an implementation of the boundary inte-

Table 1. Electric-field calculation accuracies and execution times demonstrated by the program using SUN SPARC Station IPX, PC 486DX2-66MHz and IBM RISC 6000.

Number of boundary elements	Amount of required memory	Calculation accuracy $\delta\epsilon/\epsilon$	Execution time		
			SUN SPARC IPX	PC 486DX2-66	IBM RS/6000
135	150 KB	0.01	8 sec	35 sec	3 sec
265	600 KB	0.004	44 sec	2 min 5 sec	15 sec
405	1.4 MB	0.0008	3 min 41 sec	11 min 21 sec	1 min 18 sec

gral equation technique. The performance of the field calculation algorithm was analyzed by taking into account execution time and computation accuracy achieved. For the test, multiwire proportional chamber geometry was used. The exact analytical solution of the field equations for this geometry was taken from the ref. [2]. Table 1 presents the data on software performance demonstrated using a SUN SPARC IPX workstation, PC 486DX2-66MHz and an IBM RISC 6000.

CONCLUSION

A software package intended for simulation of gas filled chambers widely used in High Energy Physics experiments has been developed. Described tools were used in the development of thin multiwire proportional chambers for the high particle flux experiment [3]. The software has demonstrated reliable work and good performance. The package can be applied for design of recently developed microstrip gas counters as well as conventional gas detectors like multiwire proportional and drift chambers.

ACKNOWLEDGEMENT

The author wishes to thank Pavel Akishin of the Laboratory of Computer Technique and Automation, and Daniel Vinyar of Frank Laboratory of Neutron Physics of the Joint Institute for Nuclear Research, for their support and help with this project.

REFERENCES

1. J.J.Florent et al. The electrostatic field in microstrip chambers and its influence on detector performance. Nucl. Instr. and Meth. A329 (1993) 125-132.
2. G.A.Erskine. Electrostatic problems in multiwire proportional chambers. Nucl. Instr. and Meth. 105(1972) 565-572.
3. P.I.Pogodin. Diploma thesises. Moscow, 1993.

A FUZZY RADON TRANSFORM FOR TRACK RECOGNITION

J. Blom, C. Th. A. M. de Laat, P. G. Kuijer*, and W. Lourens
Department of Computer Topics in Physics,
* Subatomic Physics Department,
Buys Ballot Laboratory, Postbus 80000,
508 TA Utrecht, The Netherlands

Abstract

In this contribution a fuzzy Radon transform is shown for applications such as in ALICE and ATLAS. (typical track density: 8000 tracks in 1 unit of rapidity). Resolution is introduced by the "broadening" of the matching tracks in the Radon transform, which is obtained by making a convolution of the matching tracks with a Gaussian kernel. In a good approximation, an analytical expression for the fuzzy Radon transform is given. An example of two track separation with noisy input is added.

INTRODUCTION

The use of the *Radon* transform or the *Hough* transform, in particle physics is not new [1, 2]. A Radon transform is defined by the family of line integrals over a class of trajectories where the integrand is mostly a (measured) density. It defines a transform from coordinate space to track parameter space.

Due to detector imperfections the original Radon transform intrinsicly is instable. Therefore the concept of resolution is introduced in the generalised *fuzzy* Radon transform by making a convolution of the original Radon transform with a Gaussian kernel.

RADON TRANSFORM

The Radon transform is defined as

$$R(\mathcal{P}) = \int \rho(\vec{r})\, \rho_T(\vec{r}, \mathcal{P})\, d\vec{r} \qquad (1)$$

where 1. $\rho(\vec{r})$ is the (measured) density function, and 2. the track density $\rho_T(\vec{r}, \mathcal{P})$ describes a parameterised track T with parameterisation parameter τ and the extra parameter set $\mathcal{P} = \{p_1, \ldots, p_m\}$; ρ_T is

$$\rho_T(\vec{r}, \mathcal{P}) \equiv \int_T \delta(\vec{r} - \vec{r}(\tau, \mathcal{P}))\, d\tau \qquad (2)$$

The Radon transform measures the exact match between T and ρ.

Introduction of Resolution

Gyulassy et al. [1] generalised (1) to the *fuzzy* Radon transform $R(\cdot; \sigma)$

$$R(\mathcal{P}; \sigma) \equiv \int \rho(\vec{r})\, \rho_T(\vec{r}, \mathcal{P}; \sigma)\, d\vec{r} \qquad (3)$$

The convolution of ρ_T (2) with a kernel G defines the fuzzy track density $\rho_T(\cdot; \sigma)$

$$\rho_T(\vec{r}, \mathcal{P}; \sigma) = \int_T G(\vec{r}(\tau, \mathcal{P}) - \vec{r}; \sigma)\, d\tau \qquad (4)$$

where σ is the resolution parameter of the blur kernel G. An n-dimensional Gaussian kernel is used for its advantages [3, 4]

$$\rho_T(\eta_g, z_g, \kappa, \gamma; \sigma) \simeq \frac{1}{2\pi\,\sigma^2} \cdot \frac{|\kappa|}{\sqrt{\eta_g + \kappa^2\,\gamma^2}}\, e^{-\frac{1}{2\sigma^2}\left[\frac{(\eta_g-1)^2}{\kappa^2} + \frac{\eta_g\, z_g{}^2}{\eta_g+\kappa^2\,\gamma^2}\right]}$$

with $\eta_g \equiv \sqrt{(\kappa\, x_i + \sin\phi)^2 + (\kappa\, y_i - \cos\phi)^2}$ and $z_g \equiv \gamma\,\tau_i - z_i$ and

$$\tau_i = \mathrm{sgn}\,(\kappa)\, \arctan\left(\frac{y_i \sin\phi + x_i \cos\phi}{\kappa^{-1} + x_i \sin\phi - y_i \cos\phi}\right) + n\pi; \quad n = 0, 1, \ldots$$

Table 1: The expression of the the track density, calculated with gauge coordinates.

$$G(\vec{r}; \sigma) \equiv \frac{1}{(2\pi\,\sigma^2)^{\frac{n}{2}}} e^{-\frac{\|\vec{r}\|^2}{2\sigma^2}} \qquad (5)$$

With (4) the fuzzy Radon transform finds a blurred match between T and ρ.

Fuzzy Radon Transform for a Spiral

The parameterisation of the charged spiral tracks in a nuclear physics detector with a magnetic field $\vec{B} = B\hat{z}$ is given by: the signed curvature κ, the emission angle in the xy-plane ϕ, and the z-velocity γ.

If the N detector channels measure hit positions, the measured density becomes

$$\rho\,(\vec{r}\,) = \sum_{i=1}^{N} \lambda_i\, \delta(\vec{r} - \vec{r}_i) \qquad (6)$$

Channel output λ_i is 1 (0) if channel i is active (non-active). The fuzzy Radon transform (3) becomes with (6)

$$R(\kappa, \phi, \gamma; \sigma) = \sum_{i=1}^{N} \lambda_i\, \rho_T(\vec{r}_i, \kappa, \phi, \gamma; \sigma) \quad (7)$$

Reduction with Local Gauge Coordinates

The integral in (4) for the spirals cannot be solved analytically. But for each measured position $\vec{r}_i$ a formal gauge coordinate transformation is constructed such that one of the axes of this gauge system coincides with the direction of the distance in the xy-plane between track and the i-th channel. Now (4) can be solved in a good approximation (table 1).

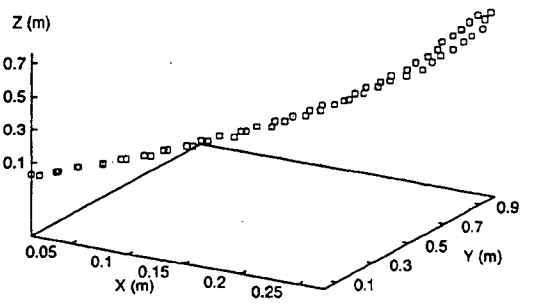


Figure 1: Two artificial tracks with noise.

EXAMPLE

In figure 1 an example is given of two artificial tracks which differ in curvature value. The parameter values are $r \equiv \kappa^{-1} = 2 \pm 0.03\,\mathrm{m}$, $\phi = 60$ deg and $\gamma = 1.5\,\mathrm{m}$. To the coordinates of these ideal tracks random drift values are added. The tracks are shown in figure 1.

In the images of figure 2 each row contains the fuzzy Radon transform with the following standard deviations from the top to the bottom row $\sigma = 0.5\,\mathrm{mm}$, $\sigma = 1.8\,\mathrm{mm}$ and $\sigma = 5\,\mathrm{mm}$. In each image row γ is varied, in the columns of each $2D$-subimage r is varied and in its rows ϕ is varied. The original track parameters are in the center image of the cen-

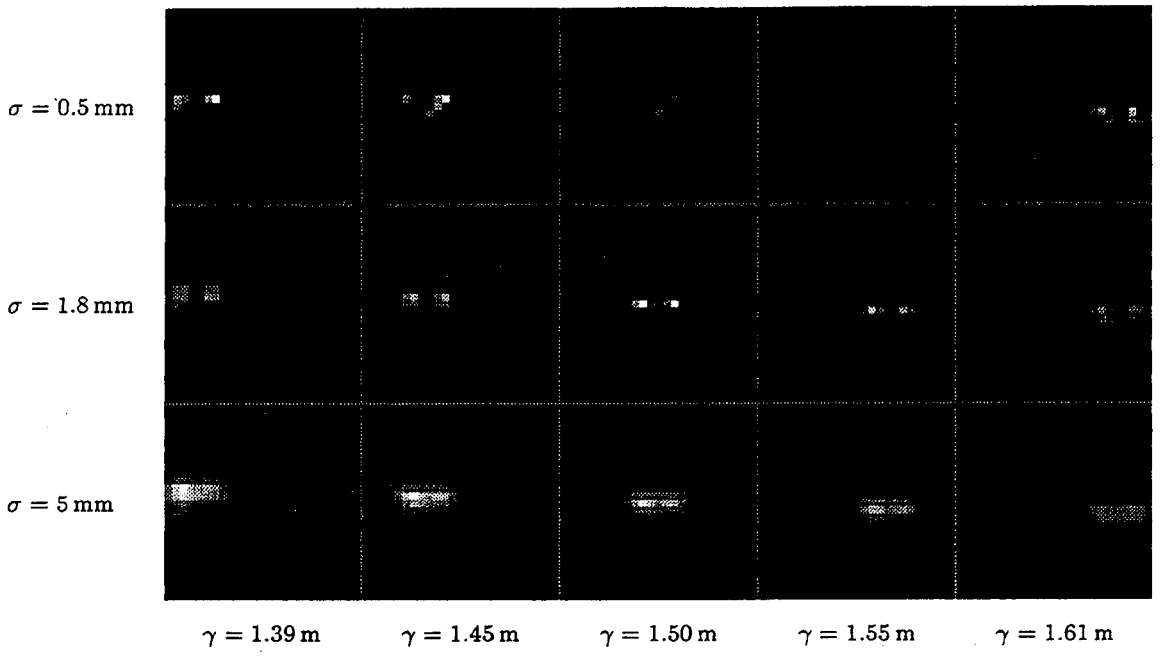


Figure 2: The fuzzy Radon transform for three values of the resolution parameter σ.

ter row. The images of each image row are separately scaled. The discretisation steps in all subimages are $\Delta r = 0.017\,\mathrm{m}$, $\Delta\phi = 0.5$ deg and $\Delta\gamma = 0.05\,\mathrm{m}$.

We see in figure 2 that there are two local maxima in the Radon transform which are found for the correct track parameters. The used value of σ has about the same size as the mean drift distance.

CONCLUSIONS

The fuzzy Radon proves to be a stable way to obtain the track parameters from noisy detector input as is shown in the example. The standard deviation of the used Gaussian kernel should be comparable with the size of the drift distance.

Commonly used algorithms discretise in a much earlier stage. These strategy had some disadvantages: 1. an implicit minimum resolution is introduced by the early discretisation, and 2. no possible analytical reduction.

REFERENCES

1. M. Gyulassy and M. Harlander, "Elastic Tracking and Neural Network Algorithms for Complex Pattern Recognition," Computer Physics Communications no. 66, 1991, pp. 31–46.
2. M. Ohlsson, C. Peterson, and A. L. Yuille, "Track Finding with Deformable Templates – The Elastic Arms Approach," Computer Physics Communications no. 71, 1992, pp. 77–98.
3. J. J. Koenderink, "The Structure of Images," Biol. Cybern. vol. 50, 1984, pp. 363–370.
4. J. Blom, B. M. ter Haar Romeny, A. Bel, and J. J. Koenderink, "Spatial Derivatives and the Propagation of Noise in Gaussian Scale-Space," J. of Vis. Comm. and Im. Repr. vol. 4, March 1993, pp. 1–13.

NEW SUPER-COMPUTING FACILITY IN RIKEN

Shigemi Ohta
The Institute of Physical and Chemical Research (RIKEN)
Wako-shi, Saitama 351-01, Japan

Abstract

A new supercomputer, Fujitsu VPP500/28, was installed in the Institute of Physical and Chemical Research (RIKEN) at the end of March, 1994. It consists of 28 processing elements (PE's) connected by a high-speed crossbar switch. The switch is a combination of GaAs and ECL circuitry with peak band width of 800 Mbyte per second. Each PE consists of a GaAs/ECL vector processor with 1.6 Gflops peak speed and 256 Mbyte SRAM local memory. In addition, there are 8 GByte DRAM space, two 100 Gbyte RAID disks and a 10 TByte archive based on SONY File Bank system. The author ran three major benchmarks on this machine: modified LINPACK, lattice QCD and FFT. In the modified LINPACK benchmark, a sustained speed of about 28 Gflops is achieved, by removing the restriction on the size of the matrices. In the lattice QCD benchmark, a sustained speed of about 30 Gflops is achieved for inverting staggered fermion propagation matrix on a 32^4 lattice. In the FFT benchmark, real data of 32, 128, 512, and 2048 MByte are Fourier-transformed. The sustained speed for each is respectively 21, 21, 20, and 19 Gflops. The numbers are obtained after only a few weeks of coding efforts and can be improved further.

INTRODUCTION

The Institute of Physical and Chemical Research (RIKEN)[1] is located in a northern suburb of Tokyo, within 30 minutes ride of subway from the city center. It also has a few branch laboratories in such cities like Sendai, Tsukuba, Nagoya, and Harima. It covers researches in physics, chemistry, biology, etc.

Its activities in physics include theoretical (lattice QCD, nuclear theory, etc.), astro (X-ray observations, etc.), nuclear (secondary beam of neutron-rich nuclei like ^{11}Li, etc.), and so on.

Demand
for super-computing is strong, especially from lattice QCD, astrophysics (large-scale FFT), nuclear physics (efficient data handling), environmental research (large-scale CFD), biophysics (protein folding,

[1] http://www.riken.go.jp/

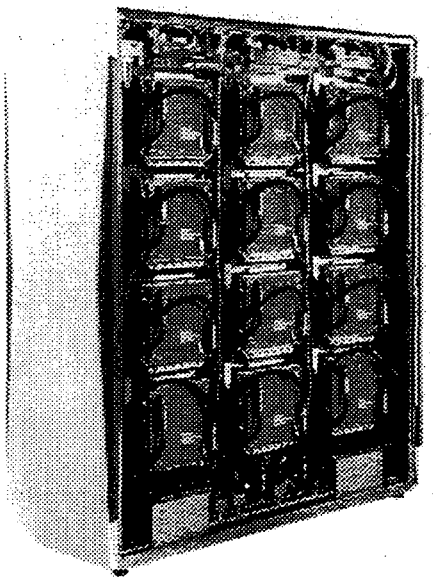


Figure 1. A 2m-tall cabinet of VPP500 fully loaded with 12 PE's.

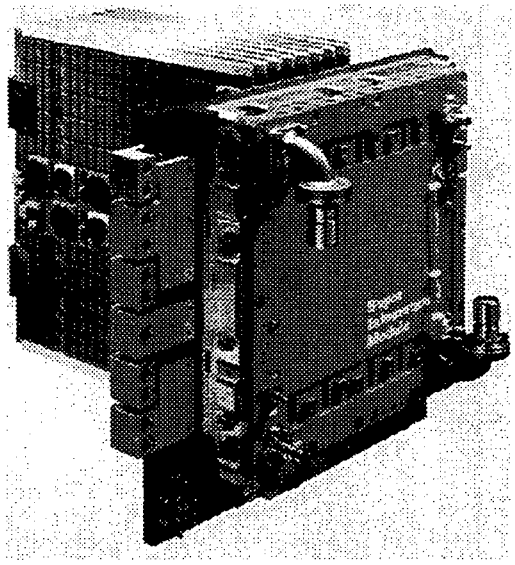


Figure 2. A processor element of VPP500. About $(40cm)^3$ with eight memory boards behind the processor board. The crossbar switch fits in the same unit volume.

genom data handling) and material science (large-scale FEM).

NEW HARDWARE

To satisfy the demand, RIKEN decided to buy a new super-computer in April, 1993. After several months of the selection process which included the three benchmark tests described below, Fujitsu VPP500/28 was installed at the end of March, 1994.

The new super-computer is a MIMD parallel computer consisting of a high-speed crossbar switch, 28 processor elements (PE's), a control processor, a front-end computer, and semiconductor and magnetic disks. There are additional magnetic disks and tape archive connected by another (slower) crossbar switch.

The high-speed crossbar switch is made of GaAs and ECL circuitry and allows 400 MB/s data transfer each-way between any pair of PE's. It allows arbitrary partitioning of the PE's. It also connects the control processor.

The processor elements are also made of GaAs and ECL circuitry. Each of them consists of a scalar cpu, a vector cpu and memory. The scalar cpu runs at 100 MHz. With its LIW architecture, this means 300 MOPS of logical operations and 200 MF of floating-point processing. There are 32 32-bit general-purpose registers and 32 64-bit floating point registers. The vector cpu has one each of mult, add/logic, div, mask, load and store pipe lines, 128 KB of vector and 2 KB of mask registers. Its peak speed is about 1.6 GF. There are 256 MB of memory made of 1-Mbit, 18-ns SRAM's.

The control processor has the same scalar cpu as the PE's, and 128 MB of SRAM memory. It handles various I/O processing with the front-end and the disks.

The front-end also has the same scalar cpu and 128 MB SRAM memory. In ad-

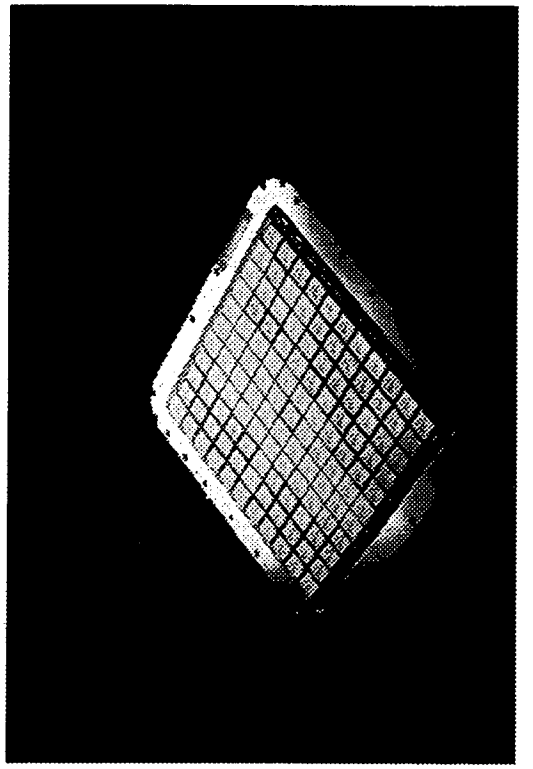


Figure 3. The processor board. About $(35cm)^2$. The switch board is of about the same size.

dition to communicating with the control processor, it handles the batch queues, compilations, and LAN communication through an external crossbar switch (Gigaswitch).

There are four kinds of disks: 8 GB semiconductor, 40 GB magnetic (18 MB/s peak communication rate), 60 GB magnetic RAID3 (50 MB/s sustained), and 100 GB magnetic RAID7 (connected through Gigaswitch, up to 10 MB/s). In addition, there is a 10 TB tape archive (SONY Filebank) also connected through Gigaswitch, with data transfer rate not exceeding a few MB/s.

SOFTWARE

UNIX SVR4 system runs on the front-end of the super-computer. A modified UNIX kernel runs on the control processor and processor elements. The modifications are mainly for handling of communication with the high-speed crossbar switch. These operating systems combined allows MIMD operation and arbitrary partitioning of the PE's.

All the PE's are reserved for batch jobs only. Batch queues are handled by NQS on the control processor. The front-end is used for compiling source codes, submitting batch-jobs, and communication with the LAN.

Currently, three different languages are available: fortran 77, C, and assembler. In fortran, vectorization and parallelizations are coded by compiler directives embedded in the codes as comments. In C, they are handled by message-passing libraries. A few different sets of such libraries are supported.

BENCHMARKS

Three kinds of benchmark tests were done: modified LINPACK, lattice QCD and Fast Fourier Transformation (FFT).

In the modified LINPACK test, with constraints on the array sizes removed, the machine ran well over 28 GF.

In the lattice QCD test, calculational speed in inverting the staggered fermion propagation matrix on 32^4 lattice was more than 30 GF.

In the FFT test, one-dimensional real arrays of the size 32 MB, 128MB, 512 MB, and 2048 MB are transformed. The measured speeds are 21 GF, 21 GF, 20 GF, and 19 GF respectively. In this test the data transfer rate from the RAID3 disk to the memory was also measured: a little more than 50 MB/s sustained.

APPLICATION

Two research projects have already started using the machine:

- Lattice QCD: pure-gauge hadron spectroscopy with staggered fermions.
- Astrophysics: large-scale FFT processing of data from satellite-bourn X-ray observatory.

Several more are being prepared:

- Lattice QCD: thermodynamics with effects of staggered fermions fully taken into account.
- Nuclear shell model calculation: large-scale ones for pf-shell and beyond, involving diagonalizations of 10^6-dimensional matrix or larger. Two different methods, auxiliary-field Monte Carlo (AFMC) and Lanczos will be tried.
- Global environment research: large-scale CFD calculations.
- Biophysics: reconstructing evolution tree from gene data.
- Material science: large-scale FEM.

CONTACT

If anyone is interested in using the machine, a trial account will be set up easily. Such service will start soon, hopefully within June, 1994. Please write to Shigemi Ohta, `ohta@theory.riken.go.jp`.

The application of artificial neural network techniques to the data analysis of SATURNE experiment LNS258 : The measurement of the branching ratio for the decay of the eta meson into two photons

Eta Collaboration, presented by Paul Fuchs
Laboratoire National SATURNE
91191 Gif-sur-Yvette Cedex
France

Abstract

We discuss the possible applications of neural networks for the data analysis of experiments performed at the η tagging facility of Saturne and focus on one specific application : the reconstruction of photon kinematics from calorimeter data.

INTRODUCTION

The branching ratios of the rare decays of the η meson are not well known. With the source of tagged η's installed at the Saturne synchrotron of Saclay [1], considerable improvement can be achieved on most of the decay modes. The η are produced with high intensity by using the $pd \rightarrow {}^3He\ \eta$ reaction near threshold. The ^3He are detected in the magnetic spectrometer SPES2 to tag the η meson.

With experiment LNS258 [2] we measured the branching ratio $BR(\eta \rightarrow \gamma\gamma) \equiv \Gamma(\eta \rightarrow \gamma\gamma)/\Gamma_{tot}$. The two photons were detected with two identical γ detectors configured at $\pm 65°$ with respect to the proton beam direction (corresponding to $\pm 90°$ in the η rest frame). Each arm did consist of an array of hexagonal BGO scintillation counters (5.5 cm inner diameter), which measured energy and direction of the photon. The overall acceptance was defined by one lead collimator in front of one of the BGO detectors.

In order to study decay modes where three or more particles are produced, we are building a large acceptance detector (3π) consisting essentially of a barrel of pure crystals of CsI, to detect photons and charged particles, and a barrel of thin plastic scintillators to identify charged particles [3]. By triggering only on ^3He, all decay modes will be recorded at the same time, the selection being made off line.

The signal for the decay $\eta \rightarrow \gamma\gamma$ is free of background and can be used to study the performance of algorithms to solve the problems of the reconstruction of photon kinematics, photon identification, two photon separation, electromagnetic split-offs and so on which are important for the data analysis of future experiments with the large acceptance CsI detector.

In this paper we apply a neural network algorithm to the reconstruction of

the photon kinematics from calorimeter data.

THE NEURAL NETWORK ALGORITHM

The calorimeter consists of hexagonal BGO crystals with a width of 5.5 cm and a length of 20 cm. Photons from the decay of the η meson originate at the target with a typical energy of 300 MeV. They get absorbed in the calorimeter. The problem is to determine energy and momentum of the photon from the pattern of deposited energy.

Due to the size of individual modules and the high mass of BGO, the electromagnetic shower generated by a photon of 300 MeV energy is completely contained within a cluster of seven elements - the module with the highest energy deposition and its six nearest neighbours.

One can immediately define the total energy and a first approximation of the position of impact.

$$E_{tot} = \sum_{i=0}^{6} E_i \qquad \vec{x}_{ews} = \sum_{i=0}^{6} \vec{x}_i \frac{E_i}{E_{tot}} \quad (1)$$

where E_0 is the maximal energy deposition with $E_1 \ldots E_6$ denoting the energy deposition in the neighbouring modules (counted anti-clockwise). The vector $\vec{x}_{ews}$ is the energy-weighted sum of the positions of the modules within the cluster evaluated at a depth within the crystal where the longitudinal shower profile peaks.

The fully reconstructed position $\vec{x}_s$ is the result of a general fit to the lateral shape of the electromagnetic shower and will be the standard with which we compare the neural network results. It only makes use of the three elements within the 7-cluster which have the highest energies. Without going into details, we simply write :

$$\vec{x}_s = \text{empirical fit to the data} \quad (2)$$

We have tried a variety of neural network structures and sizes (both in the number of hidden layers and the number of nodes within a hidden layer) as well as a number of possible input and output data vectors depending on how much a priori knowledge about the symmetry and orientation of the shower we want to offer to the neural network [4]. The best and at the same time simplest network architecture has been chosen for this study.

The Input Vector

The input set **I** is defined as the set of normalized energies $E_i, i = 0 \ldots 6$, E_{tot} and a bias unit which is always set to a value of -1.

The Output Vector

We have trained the neural network to learn the position of impact relative to the central element of the cluster. Convergence to the optimum set of weights during the learning phase was only possible if this central position was evaluated at the depth in the crystal at which the longitudinal shower profile peaks.

$$\vec{x}_{nn} = \vec{x}_{real} - \vec{x}_0 \quad (3)$$

The Network Architecture

The structure of the network is a fully connected multi-layer-perceptron with two hidden layers. The transfer function chosen for the hidden layer is

given by $f(x) = tanh(x)$, where x is the sum of weighted inputs. The output layer is linear since we like to describe a linear function with in principal arbitrary range. The sizes of the hidden layers have been empirically fixed to 6 neurons in the first layer and 3 neurons in the second.

The Training and Testing

The simulation package GEANT (version 3.13) has been used to generate data samples for the training and testing of the neural network algorithm. During the simulation, energy and angles of the photon as well as the distance between the target and the simulated detector were varied within the ranges required by the experiment.

Training was performed on a data set of 500 events. The testing was performed on an enlarged data set of 2000 events.

The neural network was trained with the traditional backpropagation algorithm with a learning term and a momentum term. The weights were updated online, i.e. after the presentation of each single event. Furthermore, we updated the learning parameters after each epoch (i.e. after 500 events have been presented for training). If the network was able to improve its description of the desired output, the learning parameters were increased by 5 %, otherwise they were decreased by 25 %. This resulted in very stable learning behaviour and the optimal solution could be reached after typically 50 epochs.

RESULTS

The comparison between the standard and the neural network algorithms was based on the test set - 2000 fully simulated gamma events. The neural network algorithm provides an improvement of $\approx$ 20 % in the sigma value (both x and y) over the standard algorithm [4].

In order to validate the neural network algorithm for use in the data analysis, we have studied its response to additional noise in the detector and to various energy thresholds applied to the simulated data. The results are listed in the tables.

It can be seen that the algorithm is very robust in that the position resolution (1) is unaffected by the addition of noise and (2) degrades smoothly to the standard algorithm as information is thrown away due to an increase in the threshold.

Finally, the algorithm was tested in the field on the data from experiment LNS 258. The variable, which is most sensitive to the gamma kinematics, is the two photon opening angle correlation function, which is the difference between measured opening angle and expected opening angle coming from a knowledge of the ^3He kinematics. After inclusion of further measurement uncertainties, an improvement of 10 % in the FWHM could be gained.

CONCLUSIONS

We developed and tested a neural network algorithm to reconstruct the photon kinematics from calorimeter data. It was found that the algorithm was easy to implement with very short learning times, lead to an improvement in the position resolution over the standard algorithm and proved to be very robust against the addition of noise or a change in energy threshold.

The effect of Gaussian noise on the position resolution. The noise was added to each individual BGO element with a sigma of X % $\times \sqrt{E[\mathrm{GeV}]}$. The nominal value for the amount of noise necessary to the data analysis of experiment LNS258 is 1.75 %.

	standard		neural	
X in [%]	σ_x [cm]	σ_y [cm]	σ_x [cm]	σ_y [cm]
0.0	0.869	0.845	0.686	0.680
0.5	0.864	0.848	0.687	0.681
1.0	0.872	0.851	0.688	0.681
1.5	0.878	0.852	0.691	0.684
2.0	0.882	0.861	0.694	0.687
2.5	0.890	0.869	0.698	0.692
3.0	0.885	0.868	0.710	0.702
3.5	0.888	0.876	0.717	0.709

The effect of an increasing energy threshold on the position resolution. The nominal threshold for the data analysis is 0.001 GeV.

	standard		neural	
E_{thr} in [GeV]	σ_x [cm]	σ_y [cm]	σ_x [cm]	σ_y [cm]
0.000	0.869	0.845	0.686	0.680
0.005	0.856	0.828	0.700	0.696
0.010	0.826	0.813	0.739	0.752
0.015	0.843	0.826	0.791	0.803
0.020	0.883	0.871	0.846	0.877
0.025	0.931	0.904	0.896	0.909

The application of this algorithm in the data analysis of experiment LNS 258 has brought a small improvement in the gamma-gamma opening angle correlation function. This was not crucial in that experiment due to a small background under the $\eta \rightarrow \gamma\gamma$ signal. However, the technique has worked well for us and we plan to extend the area of application to the problems of particle identification, two gamma separation and electromagnetic split-offs as preparation for future experiments with a large acceptance CsI detector.

ACKNOWLEDGEMENTS

I am very greatful to my collaborators Alex Efendiev (JINR) and David White (UCLA) for valueable discussions and the development of standard position reconstruction algorithms. I am also greatful to the whole Eta collaboration for their continued support and encouragement.

REFERENCES

1. B. Mayer, in "Rare decays of light mesons", ed. B. Mayer (Frontières, Gif-sur-Yvette, 1990) p 161.
2. **LNS258** proposal, M. Garçon, M. Clajus, L. Lytkin et al., 1992
3. **LNS284** proposal, B. Mayer, D. Mzavia et al., 1993
4. P.Fuchs, in the proceedings (forthcoming) of the workshop on "Artificial Intelligence in High Energy and Nuclear Physics", Oberammergau, Germany, October 4-8, 1993.

A Bi Directional Euclid Geant Interface

V. Bonichi, A. Floquet, C. Girard, M. Maire

In2p3 - Lapp (Annecy)
BP110 F-74941 Annecy-le-vieux, France

We have developed an interface between the Cad system Euclid and Geant which allows to build geometries from the powerful graphical and interactive Euclid capabilities. Those geometries can be absorbed in Geant. Vice versa the geometries defined by Geant can be exported to Euclid.

Objective

The purpose of this tool is to help (force) an Euclid user to build his detector in such a way that the result can be understood by Geant3. This implies that we have to emulate the Geant3 functionalities inside Euclid in a compatible way with all the standard Euclid functions.

Emulate Geant Functionalities

Create volumes : Definition from the 12 basic Geant shapes.
Manage volumes with 'free' parameters and/or defined by gsposp.
Manage tracking medium numbers.
Position the copies : Check for complete inclusion within mother.
Manage translation vectors, rotation matrix, symmetry.
Check for overlap between sisters : only/many flag option.
Make divisions.

Create and position volumes

The Geant shapes are available from a menu. One can pass parameters either from the keyboard or from the screen. Alternatively one can recreate valide Geant shape from pre-existing Euclid elements.
After or during the creation of the volumes the user can position, rotate and duplicate the volumes by using the standard Euclid functions. The results (volumes, parameters, rotation matrix ...) are continuously updated and displayed. Copies can be changed individually : a call to gsposp is automatically generated.

The copy number and the rotation matrix number are managed semi - automatically. (check is done for the matrix already created). The symmetry is taken account. At this step the tracking medium is treated by name.

Validation

The inclusion function establishes a mother-daugther relationship. First, the tool checks that daugthers are completly included within their mother (dedicaced code for all shapes). Then it looks for the overlaping of all couple of daugthers (using standard Euclid functions) : the konly flag is updated. Finally the daugther is validate with his attributs : rotation matrix number, tracking medium number.
The exclusion function allows to remove a daugther from the tree.

Divisions

The Divisions are treated as a special case of the above scheme : the tool creates automatically the prototype of the division and the occurrence number one (it was found not necessary, even not wanted, to generate all the copies). However a given division can always be drawn. The mother-daugther relation is also automatically established.
The 4 kinds of Geant divisions are considered : an object can be totally or partly divided, symmetrically or not. The relevant parameters are updated and displayed.

Output files

1- *The geometry is stored within the standard Euclid data structures.*
Those data structures contain normal Euclid informations plus the Geant hierarchical history. Therefore the Euclid_Geant session can be restarted and continued.

2- Exchange file for Geant.
The tool scans the Euclid data structure (the starting point can be user defined), and write down the informations for Geant. Those informations are the content of the calling sequences of the 9 relevant Geant routines devoted to the geometry definition - material / tracking media - rotations matrix - volumes definition - divisions - positioning.
This file is an human-readable ascii file : keyword + free format. The file is read by a devoted routine linked to Geant, which creates the geometry immediatly. No compiling or linking is needed. It is a compact file, as it reflects directly the powerful of the Geant geometry concepts. Indeed, it is independent its Euclid origin. Follow is an example of the exchange file :

```
TMED numed 'named' numate 'namat'
ROTM irotm teta1 phi1 teta2 phi2 teta3 phi3
VOLU 'name' 'shape' numed npar par
DIVN 'name' 'mother' ndiv iax
DVN2 'name' 'mother' ndiv iax c0 numed
DIVT 'name' 'mother' step iax numed nmx
DVT2 'name' 'mother' step iax c0 numed
POSI 'name' nr 'mother' x y z irotm konly
POSP 'name' nr 'mother' x y z irotm konly
                                npar par
```

Re entrance : Go from Geant to Euclid

We want to force Euclid to create automatically a Geant oriented model. The Euclid_to_Geant exchange file contains obviously the full informations needed, in compact and readable format. The Euclid_to_Geant program is able to read and interpret his own output file. *Therefore one keeps the same file format for both Geant to and from Euclid direction.* A devoted routine in Geant scans the Zebra banks and writes down the exchange file in the format described above. This feature achieves the bi directional exchange.

Availability

1- The Euclid_Geant Exchange is an Euclid application, written in Fortran77 (Euclid 2.3). This tool is freely available, together with documentation, from IN2P3. Contact Boninchi@Lyoeu1.in2p3.fr

2- The Geant I/O are few routines already incorpored to the Geant code (Geant3.16). The doc is in the standard Geant writeup : chapters Geom and Xint.

A test system for the Local Trigger Supervisor of the DELPHI experiment at LEP

B.Schulze[1,2] A.Branco[1] J.Buytaert[3] S.Cairanti[3] F.Formenti[3] G.Valenti[4]

1.LAFEX / CBPF, R.Xavier Sigaud 150, 22290 RJ-BR; 2.INFN / Rome-II, Tor Vergata, I00173 Rome, IT; 3.CERN / ECP, 1211 Geneva 23, CH; 4.INFN / Bologna, I-40126 Bologna, IT

Abstract

The work describes a test system for the new Local Trigger Supervisor module (LTS) [1] under construction for the DELPHI experiment at LEP. The new LTS is a programmable fastbus module responsible for the generation of the timing signals of the detector with a stability better then 1ns. The test system implements an automatic test procedure and produces diagnostics on the timing and logic capabilities of the LTS. The measurement of the module under test is done by an autonomous acquisition system with the readout done by a local fastbus processor. It is envisaged the possibility to run the test either in the lab or at the experiment site. To fulfill the above requirements, the system has a Xbased user interface and graphics output capabilities. The test system runs in a heterogeneous distributed processing environment over different operating systems and integrates them through remote procedure calls, client-server processing and NFS [2,3].

1.0 Introduction

The quality and the depth of the test procedure the new LTS will be put through is defined by the requirements of the data taking in the experiment. Digitization in the DELPHI detectors requires synchronization with the beam circulating in LEP, of the order of 1ns or better.

In order to verify if the working conditions of the new LTS satisfies this requirements the present test setup has been designed. In addition to fail-pass test, we do logging of the timing measurements of each module, in order to generate calibration tables and trace any degradation.

At the experimental site tests are performed under real working conditions too difficult or expensive to completely simulate at the lab. So, after lab testing we expect to pass the new LTS from the lab environment to the experiment site in a rather smooth move, by checking the test procedures and results with the physics data taking.

2.0 Architecture overview

The test system has a distributed architecture, as represented in the figure.1 and 2.

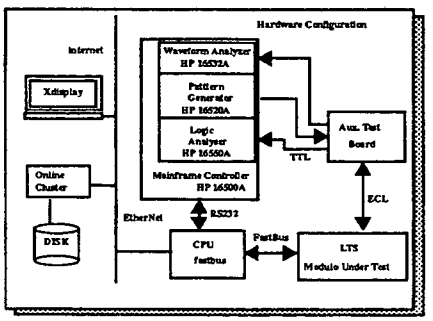


FIGURE 1. Hardware blocks.

The computing and networking facilities of the experiment are used to centralize the test data in the on_line cluster and allow a remote control and follow up

of the lab tests at any point over the network.

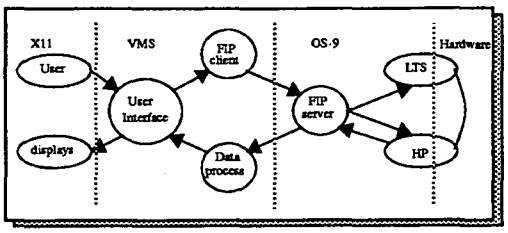


FIGURE 2. Test system software block diagram.

The control of the test is done by sending specific commands using TCP/IP[2] services.

The collected data are partially processed at acquisition time and partially stored for graphical visualization and logging of the individual modules conditions.

3.0 Test Description

The test has to tell if the basic input & output of the module is working properly, according to the different modes and trigger sequences and whether it generates correctly the programmed timing signals. The test steps are: a) fastbus interface; b) I/O, state sequences & modes; c) timing specifications (jitter, linearity, offset, crosstalk, etc...).

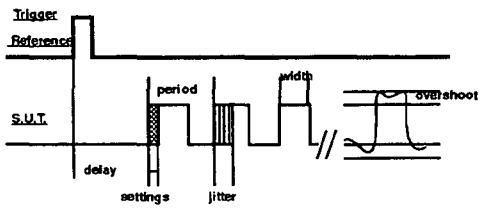


FIGURE 3. Timing signal parameters readout.

The module under test is tested with all the very inputs and outputs of the system monitored in a non invasive condition.

3.1 Data processing

We currently implement a fast and slow test mode:

In **fast mode** the test uses local processing capability of the HP signal analyser[4] with reduced amount of data being readout. Typically the outputs are Max., Minn. & Average values.

In **slow mode** the individual data are readout and post processed for statistics and histogramming in a remote workstation and display, as in figure.4.

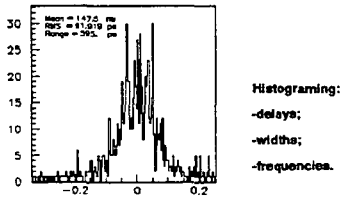


FIGURE 4. Histogramming readout data

For the logic signals the data are also readout and to identify any failure in the logic sequences we run a template recognition program on the acquired data. In case of long verification, the signal analyzer has to do a preselection of failure conditions which are then postprocessed by the template recognition program.

4.0 Current Implementation

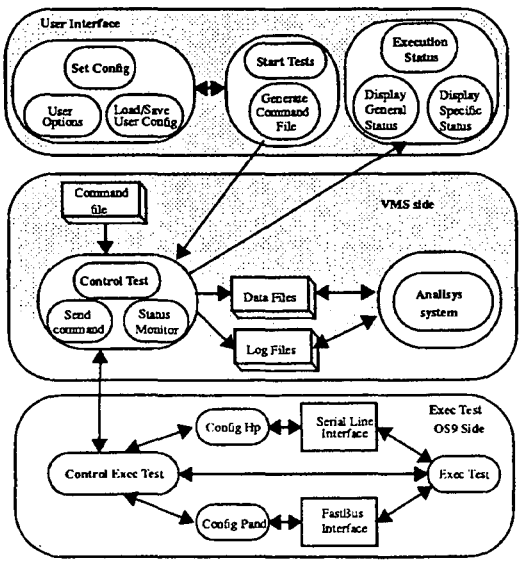


FIGURE 5. Software structure.

The figure.5 shows a simplified block diagram of the test system software structure.

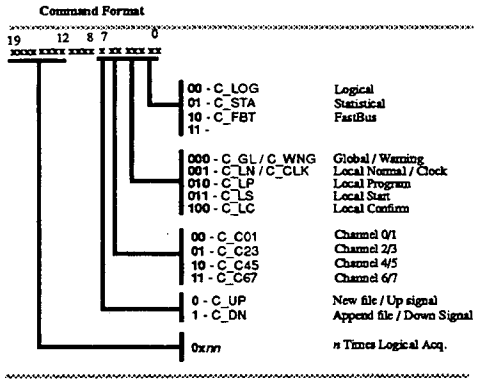


FIGURE 6. Format of commands

The test commands are written into a file from the user interface and read by the control process running in the VMS side. This process talks to the OS-9 side process which passes commands and messages. These are then executed locally or forwarded to the signal analyzer. In figure 6 is given a list of the implemented commands.

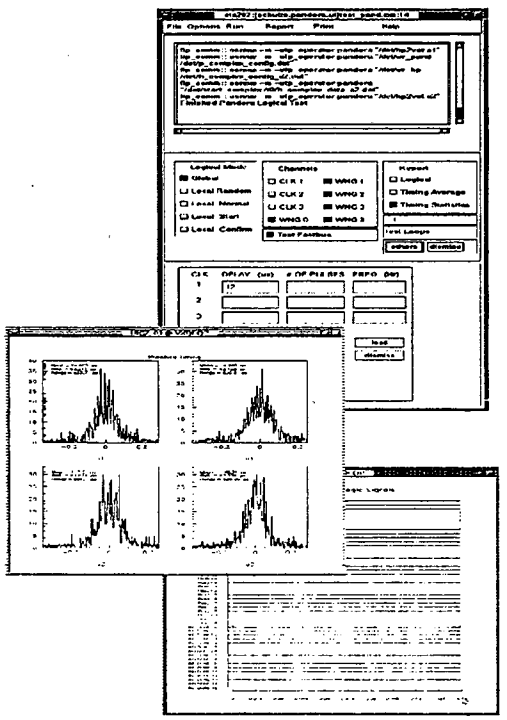


FIGURE 7. User interface.

The user interface of the test system is Xbased using Motif[5] templates, while the graphics capabilities are implemented using Cernlib/Higz[6]. A version of the user interface, an example of jitter histograms and a sample plot of the logic signals of a trigger sequence can be seen in figure.7.

5.0 Bibliography

[1] V.Bocci, A.Branco,J.Buytaert, S.Cairanti, F.Formenti, B.Schulze, G.Valenti, **Preliminary specifications of the new DELPHI LTS-CB (PANDORA)**, Rev.2.1, CERN.

[2] Multinet, on_line documentation;

[3] Charpentier Ph., Guglielmi L. - Using FIPs with the NFS system, Delphi Note, 4/93.

[4] HP16500 Logic Analysis System, **Programming Reference, Volume1 & 2**;

[5] Motif/OSF, VMS online manuals;

[6] HIGZ and HPLOT manuals, CERN;

Consumer Server: A UNIX Based Event Distributor in New CDF Data Acquisition System

F. Abe, Y. Morita, M. Nomachi, and Y. Watase
KEK, National Laboratory for High Energy Physics

K. Maeshima and J. Patrick
Fermi National Accelerator Laboratory

E. Hayashi
Institute of Physics, University of Tsukuba

Abstract

Consumer Server is a program to handle event data and consumer trigger requests I/Os among Level 3 farm and consumer processes in CDF new data acquisition system. This program uses standard UNIX libraries and commercial network technologies to obtain higher portability. We describe the concept and configuration of the Consumer Server and report its performance.

1. Introduction

To take advantages of recent progress of RISC/UNIX machines, configurations of recent data acquisition system become complex. Most of them are using distributed computing scheme which consists of RISC machines connected by network. How to distribute event data to Data Logger and monitoring processes is one of the problems in such a data acquisition system.

Figure 1 shows the downstream of the new CDF Data Acquisition(DAQ) System [1] schematically. The CDF DAQ System is upgraded to take data up to 10-20 Hz of data logging rate. To obtain high computing power, 8 Silicon Graphics servers each of which has 8 CPU's are used in Level 3 trigger system which makes final software trigger. They are connected by ULTRAnet as well as by FDDI/Ethernet. In a typical data taking, 3 VIP consumers (Data Loggers) and 4 Non-VIP consumers(monitoring programs for detector system) are running on a Silicon Graphics server for data logging and a VAX cluster for detector monitoring. Each consumer makes individual trigger request so that they can receive only a subset of events based on their request. The Consumer Server is a program to receive and forward event data as well as trigger requests between Level 3 trigger system and consumer processes. It is running on a Silicon Graphics server connected to the Level 3 trigger system via ULTRAnet and to consumers via FDDI/Ethernet. Using this scheme, the system become much simpler and easier to develop.

2. Configuration

The Consumer Server is developed using standard UNIX libraries to obtain high portability. It consists of a parent daemon process and a set of subprocesses. Communication between these processes is made via System V Message Queue's and Shared Memory. The communication between consumers is made via BSD socket. The communication between Level 3 is made via ULTRAnet socket and BSD socket. Figure 2

shows configuration of the Consumer Server software. Server Daemon is the parent process of the other processes. It has a global socket to receive registrations from Level 3, VIP Consumers, Non-VIP Consumers, and Monitors. When this process receives registration, it creates transmitter and receiver processes to communicate between corresponding Level 3, consumer, or MON.

Buffer Manager manages event buffers on a shared memory and I/Os between Level 3 and consumers. It is a subprocess of Server Daemon and created soon after startup of the Consumer Server. The algorithm of the Buffer Manager is based on the queuing and trigger matching. All the events which satisfy the request from a VIP consumer are sent to the corresponding consumer. The queuing for Non-VIP consumers have by-pass processing to skip the consumer. And events are sent when the corresponding Non-VIP consumer makes event request. Communication among subprocesses is made by message queues. Each subprocess has a unique queue id to receive messages from other processes.

Buffer Manager handles trigger requests from consumers too. The Buffer Manager stores trigger requests on its local memory to make trigger matching, and forwards them to all Level 3 servers so that they can make Level 3 trigger decisions. All other subprocesses are transmitter and receiver processes to handle socket I/Os.

MON is another kind of consumer process which receives status information of Consumer Server. This program is used to monitor connection, buffer management, and other status information like number of events received or sent.

3. Performance

The performance of the current Consumer Server was measured by a benchmark test using event data taken at the previous CDF run. Typical event size was 250 KB/event. Figure 4 shows the configuration of the benchmark test. The result for the dummy consumers without DISK output was 2.7 MB/s to VIP consumer and 0.15 MB/s to Non-VIP consumer. As we are using slow DISKs for staging of the data logging, the speed with DISK output degraded to 1.2 MB/s to VIP consumer. The I/O speed to Non-VIP consumers were not changed.

4. Summary

Consumer Server is developed to solve complexity in communication among Level 3 farm and consumer processes. It uses standard UNIX tools to get higher portability. The performance so far achieved is 2.7 MB/s to VIP consumer and 0.15 MB/s to Non-VIP consumer. It is going to be used in high-luminosity CDF run(Run 1b, 1993-94).

Acknowledgements

We would like to thank to R. Kennedy and people who are working on CDF Level 3 upgrade for their valuable discussion on the technical issues. We appreciate Professor K. Kondo and Professor M. Mishina for their advices and helps.

References

[1] J. Patrick, these proceedings

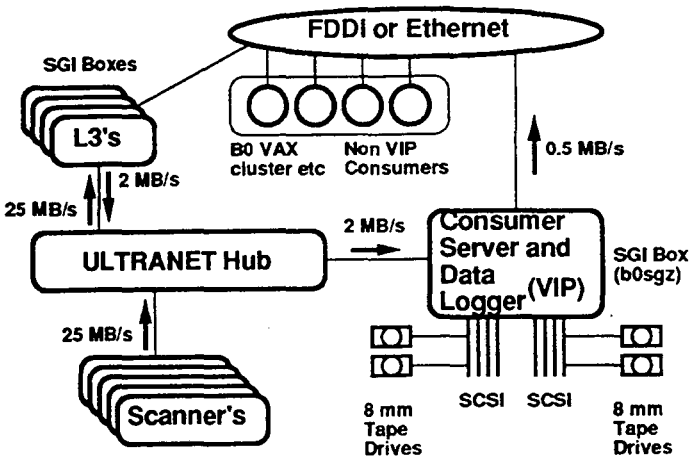


Fig. 1. Hardware Configuration

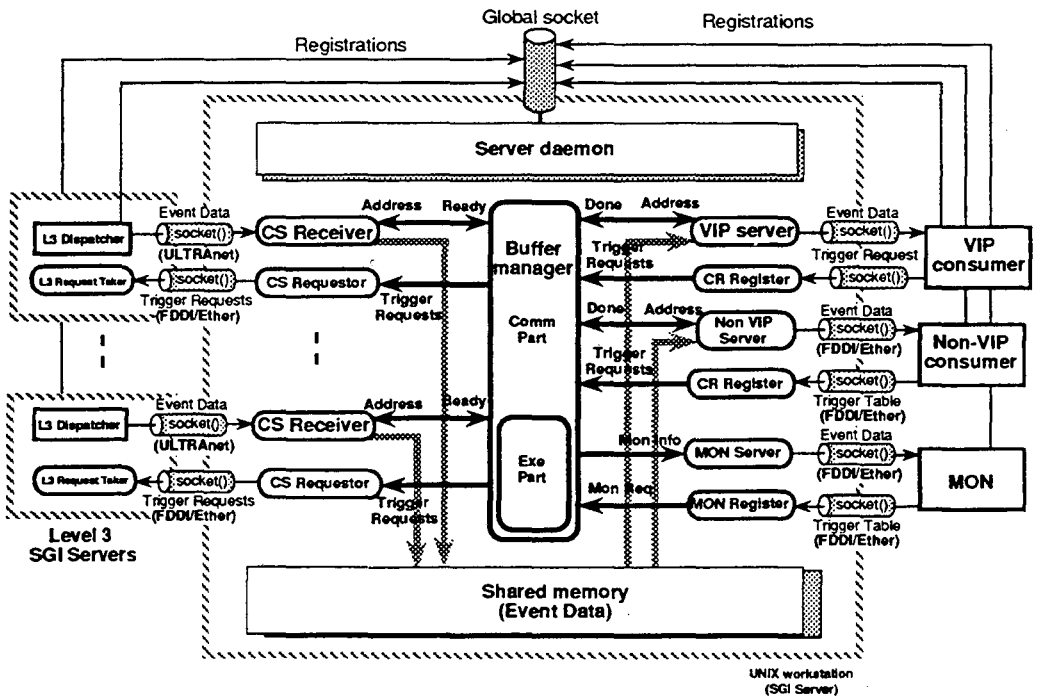


Fig. 2. Software Configuration

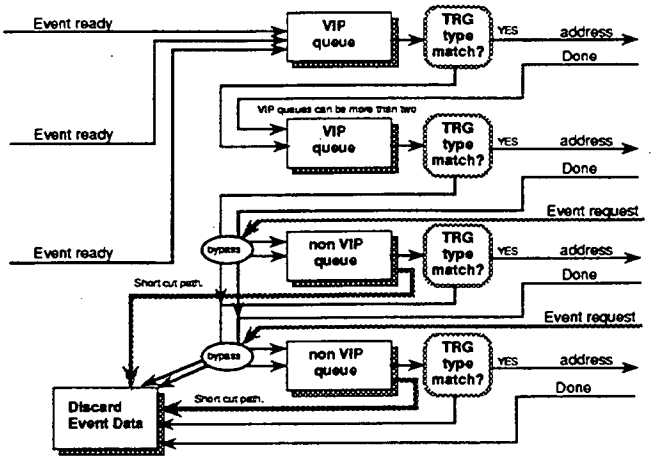


Fig. 3. Agorithm of Buffer Manager Execution Port

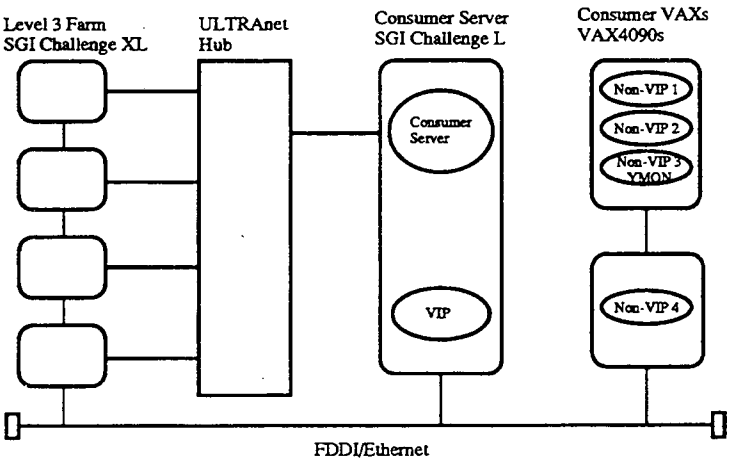


Fig. 4. Configuration of the Benchmark Test

Farhad A. Abar
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-4516*
(708) 840-8208
abar@fnal.gov

Fumio Abe
KEK,
National Laboratory. for High Energy Physics
Oho 1-1, Tsukuba
Ibaraki, Japan 305
81-298-64-5475
81-298-64-4402
fumio@kekvax.kek.jp

Gerry Abrams
Lawrence Berkeley Laboratory
1 Cyclotron Rd.
50A-160
Berkeley, CA 94720
(510) 486-7188
(510) 486-5101

Alberto Aloisio
INFN Napoli
Mostra d'Oltremare Pad. 20
Napoli, Italy 80125
39.81.7253229
39.81.2394508
aloisio@na.infn.it

Katsuya Amako
KEK,
National Laboratory for High Energy Physics
Oho 1-1, Tsukuba
Ibaraki, Japan 305
0298-64-5325
0298-64-2580
amako@kekvax.kek.jp

Giovanna Ambrosini
INFN
Via Bassi 6
Pavia, Italy 27100
39 382 392 578
39 382 526938
ambrosini@cernvm.cern.ch

E. Walter Anderson
Iowa State University
Ames, IA 50011-3160
(515) 294-2823
(515) 294-8712
anderson@iastate.edu

David Aston
Stanford Linear Accelerator Center
P. O. Box 4349, MS 62
Stanford, CA 94309
(415) 926-2457
(415) 926-3587
dyaeb@slac.stanford.edu

Masuo Atsuyosh
Digital Equipment Corp. Japan
Mitsuiseimei Tsukubakuen
1-10-10, Amakub, Tsukuba-shi
Ibaraki, Japan 305

Paul Avery
Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-3895
(708) 840-2783
avery@fnal.gov

Giuseppe Bagliesi
INFN, Pisa
via Livornese 582/a,
S. Piero a Grado
Pisa, Italy 56010
(50) 880111
(50) 880317
bagliesi@cernvm.cern.ch

Vladilen S. Barashenkov
Joint Inst. for Nuclear Research
Moscow Region
Dubna, Russia 141980
(7-095) 926-22-18
(7-096) 21 65145
barashenkov@main1.jinr.dubna.su

Guy G. Barrand
Universite de Paris-Sud
Lab. de l'Accelerateur Lineaire
Bat. 200, Orsay Cedex,
France 91405
64 46 84 17
69 07 94 04
barrand@lalcls.in2p3.fr

Gaspar Barreira
L.I.P., Portugal
Av. Elias Garcia 14-1.o
Lisbon, Portugal
1000 Lisboa
351-1-7973880
351-1-7934651
gaspar@vaxlip.lip.pt

Lothar A.T. Bauerdick
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 2459
49 40 8998 3092
bau erdick@desy.de

Frank Behner
ETH Zurich-Inst. for Particle Physics
Zurich, Switzerland CH-8093
41-22-716332031
47-22-7827558
behner@rs13eth2.cern.ch
fbehner@cernvm.bitnet

Ulf Behren
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 3092

S. Berezhnev
Moscow State University
Moscow, Russia 199899
(07) 95 939-18-18
(07) 95 939-03-97
sfb@npi.msu.su

Suman Bala Beri
Fermi National Accelerator Lab.
P. O. Box 500, MS #352
Batavia, IL 60510
(708) 840-3899
(708) 879-0048
sumanofnaldo.fnalv.gov

Eileen F. Berman
Fermi National Accelerator Lab.
P. O. Box 500, MS 234
Batavia, IL 60510
(708) 840-3941
(708) 840-2783
berman@fndaut.fnal.gov

* The first number is the telephone number; the second is the fax number.

Laura M. Bertolotto
University of Edinburg/CERN
Geneva 23, Switzerland CH-1211
41-22-767-8997
41-22-767-8950
laurab@cernvm.cern.ch

Matthew Bloomer
Lawrence Berkeley Laboratory
1 Cyclotron Road, MS 50D
Berkeley, CA 94720
(510) 486-5436
(510) 486-4818
(510) 486-5436

Joannes A. Bogaerts
CERN
Geneva 23, Switzerland CH-1211
(41) 22-767-7155

Rolf Bork
Cal Tech
L1G0
102-33 Bridge Lab.
Pasadena, CA 91125

Kors Bos
Natl. Inst. for High Energy
& Nuc. Phys
NIKHEF
P. O. Box 41882
Amsterdam,
The Netherlands NL-1009 DB
31-20-592.5944
31-20-592.5155
k.bos@nikhef.nl

Eduarda A. Brandao
LAFEX/CBPF/CNPq
Rua Dr. Xavier Sigaud
150 - 5 Andar
Rio de Janeiro, Brazil 22290-180
(00 55 21) 542-3837
(00 55 21) 541-2047
eduarda@lafex.cbpf.br

Alan M. Breakstone
University of Hawaii at Manoa
2505 Correa Road
Honolulu, HI 96822
(808) 956-7391
(808) 956-2930
alanb@slac.stanford.edu

Don Briggs
Stanford Linear Accelerator Center.
P. O. Box 4349, Bin 95
Stanford, CA 94309
(415) 926-2067
(415) 926-2957

Michael Brown
Silicon Graphics Inc.
2011 N. Shoreline Blvd.
Mountain View, CA 94043
(415) 390-3548
(415) 390-3548

Rene S. Brun
CERN
Geneva 23, Switzerland CH-1211
(41) 22-767-4124
(41) 22-767-7155
brun@cernvm.cern.ch

Elizabeth J. Buckley-Geer
Fermi National Accelerator
Laboratory
P. O. Box 500
Batavia, IL 60510
(708) 840-8650
(708) 840-2783
buckley@b0i911.fnal.gov

Toby Burnett
University of Washington
Seattle, WA 98195
(206) 543-8963
burnett@phys.washington.edu

Joel N. Butler
Fermilab
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-3140
(708) 840-2783
bitnet: butler@fnalv

Robert M. Cailliau
CERN
Geneva 23,
Switzerland CH-1211
41-22-767-5005
41-22-767-8730
cailliau@www.cern.ch

David J. Candlin
Edinburgh University
King's Buildings, Mayfield Road
Edinburgh, UK
EH9 3JZ
44-31-650-5279
44-31-662-4712
d.j.candlin@e.ac.uk

Thomas A. Carey
Los Alamos National Laboratory
P. O. Box 1663
Group P-2, MS-D456
Los Alamos, NM 87545
(505) 667-1239
(505) 665-4986
tcarey@lanl.gov

Tony Cass
CERN
Geneva 23, Switzerland CH-1211
41-22-767-8724
41-22-767-7155
tony.cass@cern.ch

Sandro Centro
Universita' di Padova/INFN
Via Marzolo 8
Padova, Italy I-35131
39(49) 844-281
39(49) 844-245
centro@mvxpd5.pd.infn.it

Francesco Cevenini
INFN Napoli
Mostra d'Oltremare Pad. 20
Napoli, Italy 80125
39.81.7253229
39.81.2394508
cevenini@na.infn.it

Philippe Charpentier
CERN
Geneva 23
Switzerland CH-1211

Cheng-Yi Chi
Columbia University, Nevis Lab.
136 South Broadway
Irvington, NY 10533
(914) 591-8100
(914) 591-8120
chi@nevis.nevis.columbia.edu

Valerie Chorowicz
Univ. Paris VI, VII
Paris, France 75230
44-27-42-79
44-27-42-79

Alessandra Ciocio
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50A-2160
Berkeley, CA 94720
(510) 486-4729
(510) 486-5101
ciocio@lbl.gov or csa::ciocio

Al Clark
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50B-5239
Berkeley, CA 94720
(510) 486-6923
(510) 486-6923

Jacques Cohen-Ganouna
IN2P3 Computing Center
27 bd du 11 Novembre 1918
Villerurbanne Cedex,
France F-69622
(33) 78 93 08 80
(33) 78 94 30 54
ganouna@frcpn11.in2p3.fr

Laird R. Cormell
Superconducting Super Collider Lab.
2550 Beckleymeade Ave.
Dallas, TX 75237-3997
(214) 708-2000
(214) 708-0006
cormell@vx1.ssc.gov

R. Les Cottrell
Stanford Linear Accelerator Ctr.
P. O. Box 4349, MS 97
Stanford, CA 94309
(415) 926-2523
(415) 926-3329
cottrell@slac.stanford.edu

Olivier Couet
CERN
Geneva 23, Switzerland CH-1211
(41) 22-767-4886
(41) 22-767-7155
couet@cernvm.cern.ch

Frans J.G.H. Crijns
University Nijmegen Toernooiveld
Nijmegen,
The Netherlands 6525 ED
31-80-652980/652099
31-80-553450
u632709@vm.uci.kun.nl

Dario Crosetto
Superconducting Super Collider Lab.
2550 Beckleymeade, M/S 2005
Dallas, TX 75237
(214) 708-6212
(214) 708-6088
crosetto@ssc.gov

Dave Cutts
Brown University
Providence, RI 02912
(401) 863-1463
cutts@brhep1.physics.brown.edu
cutts@fnalv.fnal.gov

Ferdinando D'Isep
INFN
1, Via Giuria
Torino, Italy 10125

Michael Dahlinge
GSI, Darmstadt
Planckstr. 1
Darmstadt, Germany D-64291
49-6151-859546
49-6151-859986
m.dahlinger@gsi.de

Trevor Daniels
SERC, Rutherford Appleton Lab.
Chilton Didcot,
Oxon, UK OX11 OQX
44-235-445755
44-235-446i626
t.daniels@rl.ac.uk

Sridhara R. Dasu
University of Wisconsin-Madison
1150 University Avenue
Madison, WI 53706
(608) 265-3043
(608) 263-0800
dasu@wishep.physics.wisc.edu

Edward W. Davis
North Carolina State University
P. O. Box 8206
Raleigh, NC 27695-8206
(919) 515-7045
(919) 515-7896
davis@csc.ncsu.edu

Christopher T. Day
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50B-3238
Berkeley, CA 94720
(510) 486-5736
(510) 486-4004
ctday@lbl.gov

Michel L. De Beer
Commissariat a l'Energie Atomique
Dapnia/spp, Cedex
Gif-sur-Yvette, France 91191
33-1-69082350
33-1-69086428
debeer@hep.saclay.cea.fr

Silvano M. De Gennaro
CERN
Geneva 23, Switzerland CH-1211
022-767-61111
022-767-7155
silvano@vxcern.cern.ch

Daniele De Pedis
Instituto Nazionale di Fisisa Nucleare
INFN
P.zza Aldo Moro 5
Rome, Italy 00100
(396) 49914414
depedis@cernvm.cern.ch

Albert A. De Roeck
DESY
Notkestrasse 85
Hamburg, Germany D-22603
(49) (40) 89 98-0
(49) (40) 8994-4385
hikadr@syibm.desy.de

Denice C. Deatrich
University of Victoria/CERN
c/o CERN
OPAL, Division PPE
Geneva 23, Switzerland CH-1211
41-22-767-3839
41-22-782-1974
deatrich@cernvm.cern.ch

Cornelius T.A.M. deLaat
University of Utrecht
Princetonplein 5
Utrecht,
The Netherlands NL3584CC
(31)30534585
(31)30537555
delaat@fys.ruu.nl

Manuel C. Delfino
Florida State University, B-186
Tallahassee, FL 32306-4052
904-644-3066
904-644-0098
delfino@scri.fsu.edu

Andrea Dell'Acqua
Univ. of Wisconsin-Madison/
CERN
Geneva 23, Switzerland CH-1211
41-22-7673428
41-22-7678350
dellacqu@cernvm

Kirill G. Denisenko
Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-2418
(708) 840-8208
denisenk@fnal.fnal.gov

Nina Denisenko
Fermi National Accelerator Lab.
P. O. Box 500, MS 357
Batavia, IL 60510
(708) 840-8141

Jeff J. Dingbaum
HEPNRC
P. O. Box 500
Batavia, IL 60510
(708) 840-8472
(708) 840-8463
dingbaum@hep.net

Hans Döbbeling
GSI, Darmstadt
Gesellschaft fuer
Schwerionenforschung
Postfact 110552
Darmstadt, Germany D-64220
49-6151-359-554
49-6151-359-986
dob@hp9a.gsi.de

Dave Doughty
CEBAF
12000 Jefferson Avenue
Newport News, VA 23606
(804) 249-7100
(804) 249-5800

Michael Draper
CERN
AS Division
Geneva 23, Switzerland CH-1211
41-22-767-3348
41-22-767-8142
mick@cernvm.cern.ch

Serge S. Dû
Universite de Paris-Sud
Lab. de l'Accelerateur Lineaire
Bat. 200
Orsay Cedex, France 91405
64 46 84 17
69 07 94 04
du@lalcls.in2p3.fr

Richard Dubois
Stanford Linear Accelerator Ctr.
P. O. Box 4349
Stanford, CA 94309
(415) 926-3824
(415) 926-2923
richard@slac.stanford.edu

Jean-Pierre Dufey
CERN
Geneva 23, Switzerland CH-1211
41-22-767-4408
jpd@vxomeg.cern.ch

Mark W. Edel
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-2496
(708) 840-2783
edel@fnal.gov

Stephan Egli
Physik-Institut der Universität
Zürich
Winterthurerstrasse 190
Zurich, Switzerland CH-8057
(01) 257 57 21/22
(01) 257 57 04

Michael Ernst
DESY
Notkestrasse 85
Hamburg, Germany 22603
49 40 8998 3618
49 40 8998 4429
ernst@desy.de

Francois Etienne
CPP Marseille
163 Avenue de Luminy,
Case 90
Marseille Cedex 20, France 13288
(33) 91827299

M. L. Ferrer
INFN
Via Enrico Fermi 40
Frascati, Italy 00040
396-9403415
396-9403427
ferrer@vaxlnf.lnf.infn.it

Thomas Finnern
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 2753
49 40 8998 3092
finnern@desy.de

Luigi Fiore
INFN
Via Amendola 173
Bari, Italy I-70126
39-80-242433
39-80-242470
flower@vaxba0.ba.infn.it

Mark S. Fischler
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-4339
(708) 840-8208
mf@fnal.gov

Stephen M. Fisher
Rutherford Appleton Laboratory
Chilton, Didcot
Oxon, UK OX11 0QX
44 (235) 44 5493
44 (235) 44 6733
s.m.fisher@rl.ac.uk

Mariusz Flasinski
Deutsches Elektronen-Synch.
(DESY)
Notkestrasse 85
Hamburg, Germany 22603
040-18998-2347
040-18998-3282
flasinski@vxdesy.desy.de

C. Foudas
University of Wisconsin
1150 University Avenue
Madison, WI 53706
(608) 263-0800

Boda J. Franek
Rutherford Appleton Laboratory
Chilton, Didcot
Oxon, UK
OX11 0QX
44-235-821900
44-235-446733
franek@v2.rl.ac.uk

Hans Frese
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 3092

Wolfgang Friebel
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-324
49 33762 77-216
friebel@ifh.de

Paul K. Fuchs
Laboratoire National Saturne
Cedex
Gif-sur-Yvette, France 91191
33-1-69.08.61.97
33-1-69.08.90.11
fuchs@lnsbm2.cern.ch
or 33608::fuchs

Philippe P. Gaillardon
IN2P3 Computing Center
27 bd du 11 Novembre 1918
Villeurbanne, France F-69622
(33) 78 93 08 80
(33) 78 94 30 54
phg@frcpn11.in2p3.fr

Irwin Gaines
Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-4022
(708) 840-2783
gaines@fnal.gov

Martin Gasthuber
DESY
Notkestrasse 85
Hamburg, Germany 22607
49-40-8998-2564
49-40-8998-4429
martin@viper.desy.de

Paul S. Gee
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50B-3238
Berkeley, CA 94720
(510) 486-4005
(510) 486-5977
psgee@lbl.gov

Ulrich Gensch
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-345
49 33762 77-330
gensch@ifh.de

Krzysztof L. Genser
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-8278
(708) 840-8481
genser@fnalv.fnal.gov

Ralf Gerhards
DESY
Notkestrasse 85
Hamburg, Germany D-22603
(49)(40) 8998 3858
(49)(40) 8998 3093
gerhards@desy.de

Simone Giani
CERN
Geneva 23
Switzerland CH-1211
(41) 22-767-3102
(41) 22-767-7155
sgiani@cernvm.cern.ch

Jsai Gianluca
INFN, Via Add Nejri 18
Cagliari, Italy 09127

Bruce G. Gibbard
Brookhaven Nat'l Lab., Bldg. 510C
Upton, NY 11973-5000
(516) 282-7969
(516) 282-3253
gibbard@bnl.gov

Hans J. Gils
Kernforschungszentrum Karlsruhe
P. O. Box 3640
Karlsruhe, Germany D-76021
49-7247-82-3383
49-7247-82-4075
gils@ik3dc1.kfk.de

Thomas Glanzman
Stanford Linear Accelerator Center
2575 Sand Hill Road, MS 95
Menlo Park, CA 94025
(415) 926-3160
(415) 926-2657
dragon@slac.stanford.edu

Jorge O. Gomes
L.I.P., Portugal
Av. Elias Garcia 14-1.o
Lisbon, Portugal, 1000 Lisboa
351-1-7973880
351-1-7934651
jorge@vaxlip.lip.pt

John C. Gordon
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire, UK 0X11 0QX
(44) 235 446574
(44) 235 44662i6
j.c.gordon@rutherford.ac.uk

Charles D. Granieri
Stanford Linear Accelerator Ctr.
2575 Sand Hill Road, MS 97
Menlo Park, CA 94025
(415) 926-2844
(415) 926-3329

William H. Greiman
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50B-3238
Berkeley, CA 94720
(415) 486-7458
(510) 486-4004
whgreiman@lbl.gov

Gilbert Grosdidier
Laboratoire de l'Accelerateur Lineaire
Faculte des Sciences, Bat. 200
Orsay, France 91405
33-1-6446-8905
33-1-6907-1526
grodid@frcpn11.in203.fr

Robert L. Grossman
University of Illinois at Chicago
851 S. Morgan (mc 249)
Chicago, IL 60607-7045
(312) 413-2164
(312) 996-1491
grossman@lac.math.uic.edu

Laurent J. Guglielmi
College de France
11 Pl. Marcelin Berthelot
Paris Cedex, France F-75231
1-44-27-15-39
1-43-54-69-89
guglielmi@cdfvax.in2p3.fr

Lars Hagge
DESY/ZEUS Collaboration
Notkestr. 85
Hamburg, Germany D-22309
49-40-8998-2716
49-40-8998-3092
hagge@x4u2.desy.de

John Renner Hansen
Niels Bohr Institute
Blegdamsvej 17
Copenhagen, Denmark 2100
45-35325327
renner@nbivax.nbi.dk

Frank Harris
University of Oxford
c/o CERN
ECP, 11-1-12
Geneva 23, Switzerland CH-1211
22-767-7196
harris@vxcern.cern.ch

John Harvey
CERN
ECP Division
Geneva 23, Switzerland CH-1211
41-22-7677358
41-22-7830672
harvey@alonl.cern

Tom Heiremans
Brussels Free University
Pleinlaan 2
Brussel, Belgium B-1050
32-2 641 32 03
32-2-641-38-16
theirema@vub.ac.be

Christian Ch. Helft
Universite de Paris-Sud
Lab. de l'Accelerateur Lineaire
Bat. 200
Orsay Cedex, France 91405
64 46 84 12
69 07 94 04
helft@lalcls.in2p3.fr

Otto Hell
DESY
Notkestr. 85
Hamburg, Germany D-22603
40-89981
40-8998 3282
rolhll@dsyibm.desy.de

Fabio Hernandez
IN2P3 Computing Center
27 bd du 11 Novembre 1918
Villerurbanne
Cedex, France F-69622
(33) 78 93 08 80
(33) 78 94 30 54
fabio@frcpn11.in2p3.fr

Graham Heyes
Southeastern Univ. Res. Assoc./ CEBAF
12000 Jefferson Avenue
Newport News, VA 23606
(804) 249-7030
(804) 249-5800
heyes@cebaf.gov

CW Hobbs
CERN
Digital Joint Project
Bldg. 513, Office 1-026
Geneva 23, Switzerland CH-1211

Harvard H. Holmes
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50F
Berkeley, CA 94720
(510) 486-5742
(510) 486-6363

Wade Hong
Carleton University
1125 Colonel By Drive
Ottawa, Ontario.
Canada K1S 5B6
(613) 788-2600 x7461
(613) 788-4061

Walter Hoogland
CERN
Geneva 23,
Switzerland CH-1211
(41) 22-767-3950
(41) 22-782-0168
walter_hoogland@dgmail.cern

Julius Hrivnác
Inst. of Physics,
Academy of Sciences
Na Slovance 2
Prague, Czech Rep. CZ-180 40
42-2-6605-2159
42-2-858-4569
hrivnac@cernvm.cern.ch

Pier G. Innocenti
CERN
LCP Division
Geneva 23, Switzerland CH-1211
41-22-767-4633
41-22-767-4100
innp@cernvm.cern.ch

Zbigniew Jakubowski
DESY
Notkestrasse 85
Hamburg, Germany D-22603
49-40-8998-2345
49-40-8994-4429
jakubowski@desy.de

Sverre Jarp
CERN
Geneva 23, Switzerland CH-1211
41-22-767-4944
41-22-767-7155
sverre@cernvm.cern.ch

Marvin E. Johnson
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3168
mjohnson@fnal.fnal.gov

Tony S. Johnson
Stanford Linear Accelerator Center
P. O. Box 4349
Stanford, CA 94309
(415) 926-2278
(415) 926-2923
tony_johnson@slac.stanford.edu

Chris M. Jones
CERN
Geneva 23, Switzerland CH-1211
41-22-767-4884
41-22-767-7155
chris@cernvm.cern.ch

Robert J. Jones
CERN
Geneva 23, Switzerland CH-1211
41-22-767-8046
41-22-767-7155
jones@vxcern.cern.ch

Lars Jonsson
University of Lund
Solvegatan 14
Lund, Sweden S-223 62
46 46 107695
46 46 104709
lars@quark.lu.se

Beat Jost
CERN
ECP Division
Geneva 23, Switzerland CH-1211
41-22-7677392
41-22-7830672
jost@alonl.cern

Junichi Kanzaki
KEK,

Natl. Lab. for High Energy Physics
Oho 1-1, Tsukuba
Ibaraki, Japan 305
81-298-64-5339
81-298-64-2580
kanzaki@kekvax.kek.jp

Yukio Karita
KEK
Natl. Lab. of High Energy Physics
Ohi 1-1, Tsukuba
Ibaraki, Japan 305
81-298-64-5483
81-298-64-3136
karita@kek.jp

Matthias Kasemann
DESY
Notkestr. 85
Hamburg, Germany D-22603
40-8998-2483
kasemann@desy.de

Nobu Katayama
Cornell University
Wilson Laboratory
Ithaca, NY 14853
(607) 255-8778
(607) 255-8062
nk@lns62.lns.cornell.edu

Hans-Hellmut Kaüfmann
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-246
49 33762 77-330
ho1kfn@dhhdesy3.bitnet

Minato Kawaguti
Fukui University
9-1, Bunkyo-3
Fukui, Japan 910
81-776-27-8575
81-776-27-8751
kawaguti@i1mps1.fuis.fukui-u.ac.jp

Gottfried Kellner
CERN
ECP Division
Geneva 23, Switzerland CH-1211
41-22-767-6935
41-22-767-4100
kellner@cernvm.cern.ch

David P. Kelsey
Rutherford Appleton Laboratory
Chilton, Didcot
Oxon, England
UK OX11 0QX
44-235-821900, x5746
44-235-44-6733
kelsey@v2.rl.ac.uk

Vyatcheslav I. Klyukhin
Institute for High Energy Physics
Protvino, Moscow Region
Moscow, Russia RU-142284
7-095-230-23-37
klyukhin@mx.ihep.su

Jurgen R. Knobloch
CERN
ECP
Geneva 23
Switzerland CH-1211
(22) 767-7354
(22) 783-0672
knobloch@cern.ch

Yutaka Kodama
Hitachi, Ltd.
6-1, Takezono 1-Chome
Tsukuba-shi
Ibaraki-ken, Japan 305
81-298-58-2011
81-298-55-2789
kodama@fhl.hitachi.co.jp

Axel Köhler
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-370
49 33762 77-216
koehler@ifh.de

Burkhard W. Kolb
GSI
Planckstr 1
Darmstadt, Germany D-64251
49-6151-3590
4122 7824897
kolb@vxwa80.cern.ch

Bohdan T. Kotlinski
Paul Scherrer Institute
Villigen, Switzerland CH-5232
(41) 56 993268
(41) 56 993294
kotlinski@cvax.psi.ch

Thomas Kozlowski
Los Alamos National Laboratory
Mail Stop D456
Los Alamos, NM 87545
(505) 667-7747
kozlowski_thomas@lanl.gov

Helmut W. Kramer
Universitaet Mainz
Becherweg 45
Mainz, Germany D-55099
49-6131-395855
49-6131-392964
kramer@kph.uni-mainz.de

Karsten Künne
DESY
Notkestr. 85
Hamburg, Germany D-22607
49-40-8998-3315
49-40-8998-4429
kuenne@desy.de

Shuichi Kunori
University of Maryland
c/o Fermilab
P. O. Box 500, MS 352
Batavia, IL 60510
(708) 840-3837
(708) 879-0719
kunori@fnalv.fnal.gov

Paul Kunz
Stanford Linear Accelerator Center
P. O. Box 4349
MS 62
Stanford, CA 94309
(415) 926-2884
(415) 926-3587
paul_kunz@slac.stanford.edu

Stephen Lammel
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3931
(708) 840-2783

Eric C. Lancon
Commissariat a l'Energie Atomique
Dapnia/spp, Cedex
Gif-sur-Yvette, France 91191
33-1-69082350
33-1-69086428
lancon@cernvm.cern.ch

Juha T. Lappalainen
Institute for Particle Physics Tech.
Rakentajanaukio 2 C
Espoo, Finland FIN-02150
358-0-451 3168
358-0-4513164
lappala@cernvm.cern.ch

Rochelle Lauer
Yale University
260 Whitney Avenue
New Haven, CT 06511
(203) 432-3366
(203) 432-6125
lauer@yalph2.physics.yale.edu

Jean-Philippe A. Laugier
Commissariat a l'Energie Atomique
Dapnia/spp
Cedex
Gif-sur-Yvette, France 91191
33-1-69082350
33-1-69086428
laugier@hep.saclay.cea.fr

Patrick JL Le Dû
Commissariat a l'Energie Atomique
Dapnia/spp
Cedex
Gif-sur-Yvette
France 91191
33-1-69082350
33-1-69086428
dphvx2::ledu
or 32788::LEDU
or VXCERN::LEDU

Joseph Le Foll
Commissariat a L'Energie Atomique
CE/Saclay - Dapnia/Sei
Cedex
Gif-Sur-Yvette
France 91191
33-1-69083298
33-1-69083147
lefoll@hep.saclay.cea.fr

Paul L.G. Lebrun
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3947
(708) 840-2783
lebrun@fnal.gov

Holger Leich
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-275
49 33762 77-330
leichh@ifh.de

Michael J. Leitch
Los Alamos National Laboratory
P. O. Box 1663
P-2, MS D456
Los Alamos, NM 87545
(505) 667-5481
(505) 665-4986
leitch@pzhp3.lanl.gov

Michael J. LeVine
Brookhaven National Laboratory
Bldg. 510D
Upton, NY 11973-5000
(516) 282-4575
(516) 341-1334
levine@bnl.gov

Lorne J. Levinson
Weizmann Institute of Science
Rehovot, Israel
(972) 8-342084
(972) 8-344106
fhlevins@weizmann.weizmann.ac.il

Bill P. Lidinsky
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-8067
(708) 840-8463
lidinsky@hep.net

James T. Linnemann
Michigan State University
E. Lansing, MI 48824
(517) 355-33328
(517) 355-6661
linnemann@msupa.pa.msu.edu

Stewart C. Loken
Lawrence Berkeley Laboratory
1 Cyclotron Road
50B-2239
Berkeley, CA 94720
(510) 486-7474
(510) 486-6363

Morten Lomo
University of Oslo
P. O. Box 1080
Oslo, Norway N-0316
47-22852452; 47-22852401
morten@ifi.uio.no

Willem Lourens
Utrecht University
Fac. Natuur-en Sterrenkunde, FI
BBL, Princetonplein 5
Utrecht,
The Netherlands 3584 CC
3130 533905
3130 53 7555
lourens@fys.ruu.nl

Gerry Lynch
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

Paul B. MacKenzie
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3347
mackenzie@fnal.gov

Carmen Maidantchik
CERN-Atlas Experiment
Geneva 23, Switzerland CH-1211
(41) (22) 767-3100
lodi@cernvm.cern.ch

Michel Maire
LAPP/IN2P3
BP 110, Cedex,
Annecy le Vx, France F-74941
(33) 50 09 17 77
(33) 50 27 94 95

David M. Malon
Argonne National Laboratory
9700 S. Cass Avenue, Bldg. 362
Argonne, IL 60439
(708) 252-5174
(708) 252-5076
malon@silence.ctd.anl.gov

Olaf Manczak
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 3092

Livio P. Mapelli
Lawrence Berkeley Lab./CERN
1 Cyclotron Road, 50A-2160
Berkeley, CA 94720
(510) 486-2160
(510) 486-5101
mapelli@lbl.gov
mapelli@cernvm.cern.ch

Miguel Marquina
CERN
CN Division
Geneva 23, Switzerland CH-1211
41-22-767-4912
41-22-767-7155
miguel.marquina@cern.ch

David E. Martin
Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-8275
(708) 840-8463
dem@hep.net

Robert D. Martin
College of William and Mary
Williamsburg, VA 23187
(804) 221-3571
(804) 221-3540
rmartin@bnl.gov

Agnese L. Martini
INFN, LNF
via E. Fermi 40
Roma, Italy 00044
39-6-9903-622
39-6-9404-427
martini@lnf.infn.it

Alberto Masoni
INFN
Via Ada Negri 18
Cagliari, Italy I-09127
39-70-670833
39-70-657823
masoni@vxcern.cern.ch

Pere Mato
CERN
Geneva 23
Switzerland CH-1211
41-22-7676111
41-22-7830672
mato@alonl.cern.ch

Edward N. May
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439
(708) 252-6222
(708) 252-5076
enm@hep.anl.gov

Kenneth W. McFarlane
Superconducting Super Collider Lab.
2550 Beckleymeade, M/S 1006
Dallas, TX 75237
(214) 708-6120
(214) 708-6088
mcfarlane@ssc.gov

Kevin L. McIlhany
Los Alamos National Laboratory
P. O. Box 1663
Los Alamos, NM 87545
mcilhany@nuview.lampf.lanl.gov

Charles P. McParland
Lawrence Berkeley Laboratory
1 Cyclotron Road
70A-3307
Berkeley, CA 94720
(510) 486-6956
(510) 486-7105
mcparland@lbl.gov

Klaus D. Merle
Universitaet Mainz
Becherweg 45
Mainz, Germany D-55099
49-6131-396301
49-6131-396407
merle@uni-mainz.de

Pierrick Micout
Commissariat a L'Energie Atomique
CE/Saclay - Dapnia/Sei
Cedex
Gif-Sur-Yvette
France 91191
33-1-69083623
33-1-69083147
micout@hep.saclay.cea.fr

Jaroslaw Milewski
Hamburg University
Luruper Chaussee 149
Hamburg, Germany 22761
(0049-40) 8998-2005
(0049-40) 8998-3092
milewski@vxdesy.desy.de

Mariano S. Miranda
Fermi National Accelerator Lab.
P. O. Box 500, MS 369
Batavia, IL 60510
(708) 840-3791
miranda@fnal.gov

Shigeki Misawa
University of California, Berkeley
Berkeley, CA 94720
(510) 642-5892
(510) 643-8497
misawa@physics.berkeley,edu

Winfried A. Mitaroff
Institute of High Energy Physics
Nikolsdorfer Gasse 18
Vienna, Austria A-1050
(43-1) 55 73 28 32
(43-1) 55 75 89
mitaroff@vxcern.cern.ch

Paulo C. Mora de Freitas
Ecole Polytechnique-P.N.H.E.
Route de Saclay
Palaiseau, France F-91128
(33)(1) 69.33.31.82
(33)(1) 69.33.30.02
mora@frcpn11.in2p3.fr

Youhei Morita
KEK,
Natl. Lab. for High Energy Physics
1-1 Oho, Tsukuba
Ibaraki, Japan 305
81-298-64-5473
81-298-64-4402
morita@kekvax.kek.jp

Richard P. Mount
California Institute of Technology
c/o CERN
CERN-PPE
Geneva 23, Switzerland CH-1211
41-22-767-6366
41-22-782-8923
mount@lepics.cern.ch

David R. Myers
CERN
Division ECP
Geneva, Switzerland CH-1200
41-22-767-46-46
41-22-767-87-50
myersdr@vxcern.cern.ch

E. Thomas Nash
Fermi National Accelerator Lab.
P. O. Box 5000
Batavia, IL 60510
(708) 840-3211
(708) 840-2939
nash@fnal.gov

Ping Ni
North Carolina State Univ.
/INFN Italy
Raleigh, NC
(919) 515-5016
niping@lea.csc.ncsu.edu

Judith E. Nicholls
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3989
(708) 840-7783
nicholls@fnal.gov

Bogdan B. Niczyporuk
Southeastern Univ. Res. Assoc./
CEBAF
12000 Jefferson Avenue
Newport News, VA 23606
(804) 249-7251
(804) 249-5800
bogdan@cebaf.gov

Bjorn S. Nilsson
Niels Bohr Institute
Blegdamsvej 17
Copenhagen, Denmark 2100
45-35325283
45-31421-16
bjorn.s.nilsson@nbi.dk

Klaus-Henning Noffz
University of Mannheim
Computer Science V, B6
Mannheim, Germany D-68131
(49) 621-292-5755
(49) 621-292-5756
noffz@mp-sun6.informatik
.uni-mannheim.de

Robert G. Nolty
Caltech, 256-48
Pasadena, CA 91125
(818) 395-6695
(818) 795-3951
nolty@cithe501.cithep.caltech.edu

Msaharu Nomachi
KEK,
Natl. Lab. for High Energy Physics
Oho 1-1, Tsukuba
Ibaraki, Japan 305
0298-64-5383
0298-64-2580
nomachi@kekvax.kek.jp

John S. O'Neall
IN2P3 Computing Center
27 bd du 11 Novembre 1918
Villeurbanne
Cedex, France F-69622
(33) 78 98 08 80
(33) 78 94 30 54
jon@frcpn11.in2p3.fr

Michael Ogg
Carleton University
Ottawa, Canada K1S 5B6
(613) 788-2600, x4367
(613) 788-4061
ogg@physics.carleton.ca
hepcu::ogg

Shigemi Ohta
RIKEN
(The Inst. Phys. Chem. Research)
Wako-shi, Saitama 351-01
Japan
81 48 462 1111, ext .3265
81 48 462 4694
ohta@theory.riken.go.jp

Gene A. Oleynik
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-39
(708) 840-2783
oleynik@fnal.gov

Douglas L. Olson
Lawrence Berkeley Laboratory
1 Cyclotron Road, MS 50D
Berkeley, CA 94720
(510) 486-4567
(510) 486-4818
dlolson@lbl.gov

Kazuo Omata
University of Tokyo
Midori-cho, Tanashi
Tokyo, Japan 188
424-69-9512
424-64-9480
omata@insie1.ins.u-tokyo.ac.jp

Christopher J. Onions
CERN
Geneva 23, Switzerland CH-1211
41-22-767-5039
41-22-767-8350
onions@cernvm.cern.ch

Anthony M. Osborne
CERN
CN Division
Geneva 23, Switzerland CH-1211
41-22-767-3567
41-22-767-7155
tonyo@cernvm.cern.ch

Joar M. Ostby
University of Oslo
P. O. Box 1080
Oslo, Norway
N-0316
47-22852452
47-22852401
joar@ifi.uio.no

Elisabetta Pace
INFN-LNF
Via E. Fermi 40
Frascati, Italy 00044

Bernd Pagels
Institute for Particle Physics Ethz
Würenlingen + Villigen
Villigen PSI,
Switzerland CH-5232
56 99 32 80
56 99 32 94

Kwang Paick
Prairie View A&M University
P. O. Bos 355
Praire View, TX 77446
kwang@hp73.pvamu.edu

Sergio Palma
Rederal Univ. of Rio de Janeiro
Rio de Janeiro, Brazil
55-021-226-7850

Andrea P.T. Palounek
Los Alamos National Laboratory
Mail Stop D456
Los Alamos, NM 87545
(505) 665-2574
(505) 665-4986
aptp@lanl.gov

Bernd Pamzer-Steindel
CERN
Geneva 23, Switzerland CH-1211
41 22 767 4981
41 22 767 8950
panzer@cernvm.cern.ch

James F. Patrick
Fermi National Accelerator Lab.
MS 318
Batavia, IL 60510
(708) 840-2626
(708) 840-2968
patrick@fnal.fnal.gov

Tomas J. Pavel
Stanford Linear Accelerator Center
P. O. Box 4349,
MS 62
Stanford, CA 94309
(415) 926-2543
(415) 926-3587
pavel@slac.stanford.edu

Ken Joseph Peach
University of Edinburgh
JCMB, Mayfield Road
Edinburgh, UK EH9 3JZ
44-31 650 5307
44-31 662 4712
peach@ed.ac.uk

Joseph A. Perlas
CERN
Division ECP
Geneva, Switzerland CH-1200
41-22-76i7-7174
41-22-767-0672
perlas@alonl.cern.ch

Antoine A. Pérus
Universite de Paris-Sud
Lab. de l'Accelerateur Lineaire
Bat. 200
Orsay Cedex, France 91405
64 46 84 43
69 07 94 04
perus@lalcls.in2p3.fr

Jan Hendrik Peters
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 3092

Martti Pimia
CERN
Division ECP
Geneva, Switzerland CH-1200
41-22-767-87-50

Pavel I. Pogodin
University of Iowa
Van Allen Hall
Iowa City, CA 52242
(319) 335-1839
(319) 335-1753
pogodin@iasunup
.physics.viowa.edu

Ruth Pordes
Fermi National Accelerator Lab.
P. O. Box 500
Batavia, IL 60510
(708) 840-3921
(708) 840-2783
ruth@fnal.gov

Till Poser
DESY
Notkestr. 85
Hamburg, Germany D-22549
49 40 8998 3092

David R. Quarrie
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50B-3238
Berkeley, CA 94720
(510) 486-4868
(510) 486-4004
drquarrie@lbl.gov

Fons Rademakers
CERN
European Org. for Nuclear Res.
Geneva 23, Switzerland CH-1211
41-22-7674886
41-22-7677155
rdm@cernvm.cern.ch

Myrna Ramos
3301 Royal Lane
Irving, TX 75063

Harry R. Renshall
CERN, CN Division
Route De Meyrin
Geneva 23, Switzerland CH-1211
(41) 22-767-2377
(41) 22-767-2377
hrrcr@cernvm

Pál Ribarics
Max-Planck-Institut für Physik
Föhringer Ring 6
München, Germany D-80805
(89) 32 30 82 19
(89) 32 26 704
par@dmumpiwh

Fabio Riccardi
CERN-Chorus Experiment
Geneva 23, Switzerland CH-1211
(41) (22) 767-3100
riccardi@cernvm.cern.ch

Lee Roberts
Suprconducting Super Collider Lab.
2550 Beckleymeade Ave., Ste. 260
Dallas, TX 75237
(214) 708-2000
(214) 708-0006
lroberts@sscvxl.ssc.gov

Clemens C.J. Roothaan
Hewlett Packard
1501 Page Mill Rd., MS 3U
Palo Alto, CA 94304
(415) 852-8284
(415) 857-5172

Bertrand Rousseau
CERN
Geneva 23, Switzerland CH-1211
(41) 22 767 82 63
rousseau@plsunoo.cern.ch.

Paul Routenburg
University of Alberta
Edmonton, Alb.
Canada T6G 2N5
(403) 492-9658
(403) 492-3637
paul@phys.ualberta.ca

Per Sommer Rydstrom
Niels Bohr Institute
Blegdamsvej 17
Kobenhagen,
Denmark DK-2100 O
35 32 52 99
31 421016 or 35 431087
psommer@nbivax.nbi.dk
psommer@hepax1.nbi.dk

Attilio Santocchia
INFN Perugia
Via Pascoli 1
Perugia, Italy I-06100
(39) 75-585-3053
(39) 75-585-3057
santocch@vaxpg.infn.it
santocch@cenvm.cern.ch

Alberto F.D.S. Santoro
LAFEX/CBPF/CNPq
Rua Dr. Xavier Sigaud
150 - 5 Andar
Rio de Janeiro, Brazil 22290-180
(00 55 21) 542-3827
(00 55 21) 541-2047
santoro@lafex.cbpf.br

Takashi Sasaki
KEK,
Natl. Lab. for High Energy Physics
1-1 Oho, Tsukuba
Ibaraki, Japan 305
81-298-64-5474
81-298-64-4402
sasaki@amyuxz.kek.jp

R. D. Schaffer
LAL Orsay
Bat. 200 Centre d'Orsay
Cedex
Orsay, France 91405
33-1-64468378
33-1-69071526
schaffer@cernvm.cern.ch

Terry L. Schalk
University of California, Santa Cruz
National Science II
Santa Cruz, CA 95064
(408) 459-2996
(408) 459-3043
tas@scipp.ucsc.edu

Joachim J. Schambach
University of Texas at Austin
RLM 5.208
Austin, TX 78712
(512) 471-1303
(512) 471-9637
jschamba@utpapa.ph.utexas.edu

Heidi M. Schellman
Northwestern University
2145 Sheridan
Evanston, IL 60208
(708) 491-8608
(708) 840-4343
schellman@fnal.gov

Ben M. Segal
CERN
CN Division
Geneva 23, Switzerland CH-1211
41-22-767-4941
41-22-767-7155
b.segal@cern.ch

Jorge A. Silva
Collége de France
11, pl Marcelin Berthelot
Paris, Cedex, France F-75231
44 27 1499
silva@cdfvax.in2p3.f3

Antonio Silvestri
INFN
Via Ada Negri, 18
Cagliari, Italy 09127
39-70-670833-4-5-6
39-70-657823
silvestri@cagliari.infn.it
silvestri@vaxca.ca.infn.it

Lucia Silvestris
INFN Bari
c/o Instituto di Fisica
via Amendola 173
Bari, Italy I-70126
39-80-242433
39-80-242470
silvestr@cernvm.cern.ch

Gian Piero Siroli
University of Bologna
Via Irnerio 46
Bologna, Italy I-40126
39-51-351048
41-22-782-1974
siroli@vxcern.cern.ch

Dane D. Skow
Fermilab
P. O. Box 500
MS 122
Batavia, IL 60510
(708) 840-4730
(708) 840-3687
dane@fnal.gov

Darrel W. Smith
ERAU
3200
Prescott, AZ 86301
(602) 771-6663
(602) 776-3827
dsmith@physics.pr.erau.edu

Wesley H. Smith
University of Wisconsin-Madison
1150 University Avenue
Madison, WI 53706
(608) 262-4690
(608) 263-0800
wsmith@wishep.physics.wisc.edu
wsmith@wishep.bitnet

Soren P. Sorensen
University of Tennessee
Knoxville, TN 37996-1200
(615) 576-8764
(615) 576-5780
sorensen@uther1.phy.ornl.gov

Ion Stancu
Univ. of California,
Riverside/LANL
P-11, MS H-846
Los Alamos, NM 87545
(505) 667-0559
stancu@nue.lampf.lanl.gov

John A. Strong
Royal Holloway &
Bedford New College
EGHAM Surrey,
UK TW20 0EX
44 784 443451
44 784 472794
strong@vl.ph.rhbnc.ac.uk
strong@vxcern.cern.ch

Roger Strub
IN2PE CRN Strasbourg
C.R.N 23 rue du Loess
B.P. 20
Strasbourg, Cedex,
France F-67037
33-88-106588
33-88-106234
strub@frcpn11.in2p3.fr

Zbigniew Szkutnik
DESY
Notkestr. 85
Hamburg, Germany D-22607
(4940) 89983858
(4940) 89983093
fhiszk@dsyibm.desy.de

Fritz Szoncsó
Institute for High Energy Physics
Nikolsdorfugasse 18
Vienna, Austria A-1050

Helio Takai
Brookhaven National Laboratory
Bldg. 510A
Upton, NY 11973-5000
(516) 282-2812
(516) 282-5568
takai@bnldag.ags.bnl.gov

Stephen A. Tether
Massachusetts Inst. of Technology
Fermilab-318
500 Wilson Road
Batavia, IL 60510
(708) 840-8412
(708) 840-2968
tether@cdfmt1.fnal.gov

Shaheen Tonse
Lawrence Livermore National Lab.
P. O. Box 808, L-397
Livermore, CA 94551
(510) 423-4223
(510) 423-8086

Francois Touchard
CPP Marseille/IN2P3
163 Avenue de Luminy, Case 90
Marseille Cedex 20, France 13288
(33) 91827253
(33) 91827299
touchard@marcpl2.insp3.fr

Craig E. Tull
Lawrence Berkeley Laboratory
1 Cyclotron Road, 50-245
Berkeley, CA 94720
(510) 486-7253
(510) 486-7379
cetull@lbl.gov

Marc Turcotte
Superconducting Super Collider
Lab.
2550 Beckleymeade, Ste. 260
Dallas, TX 75237
(214) 708-6152
(214) 708-6354
turcotte@sscvxi.ssc.gov

Michael C. Turnill
Oracle Corporation
The Ring
Bracknell
Berkshire, UK RG12 1BW
44 344 860066/828010
44 344 828067
mturnill@uk.oracle.com

Willem M. Van Leeuwen
NIKHEF
Kruislaan 409
Amsterdam,
The Netherlands NL 1098 SJ
31-20-592-2000
31-20-592-5155
w.vanleeuwen@nikhef.nl

Jean-Pierre Vialle
LAPP/IN2P3
BP 110
Cedex
Annecy le Vx, France 74940
(33) 50091600
(33) 50279495
vialle@cernvm.cern.ch

Harald Vogt
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-289
49 33762 77-330
hvogt@cernvm

Alan Watson
The University of Birmingham
School of Physics & Space Res.
Birmingham, UK B15 2TT
(44) 21-414-4619
(44) 21-414-6709
atw@i.ph.bham.ac.uk
watsona@cernvm.cern.ch

Chip Watson
Southeastern Univ. Res. Assoc./
CEBAF
12000 Jefferson Avenue
Newport News, VA 23606
(804) 249-7101
(804) 249-5800
watson@cebaf.gov

Peter Wegner
DESY - Zeuthen
Platanen allee 6
Zeuthen, Germany D-15738
49 33762 77-363
49 33762 77-216
wegnerp@ifh.de

Torre J. Wenaus
Lawrence Livermore National Lab.
7000 East Ave., L-397
Livermore, CA 94550
(510) 423-1813
(510) 423-7884
wenaus@llnl.gov

Bebo White
Stanford Linear Accelerator Ctr.
P. O. Box 4349, MS 97
Stanford, CA 94309
(415) 926-2907
(415) 926-3329
bebo@slac.stanford.edu

Vicky A. White
Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-3936
(708) 840-2783
white@fnal.gov

David A. Whitehouse
Brookhaven National Laboratory
Bldg. 510C
Upton, NY 11973-5000
(516) 282-2072
(516) 282-3253
davew@bnl.gov

David O. Williams
CERN
Geneva 23
Switzerland CH-1211
41 22 767 4282
41 22 767 7155
davidw@cernvm.cern.ch

Wojciech A. Wojcik
IN2P3 Computing Center
27 bd du 11 Novembre 1918
Villeurbanne Cedex,
France F-69622
(33) 78 93 08 80
(33) 78 94 30 54
wojcik@frcpn11.in2p3.fr

Fermi National Accelerator Lab.
P. O. Box 500, MS 120
Batavia, IL 60510
(708) 840-3950
(708) 840-2783

David S. Woodruff
Massachusetts Inst. of Technology
Lab for Nuclear Science
77 Massachusetts Ave., 26-402
Cambridge, MA 02139
(617) 253-6943
(617) 258-6923
dsw@mitlns.mit.edu

Douglas M. Wright
Lawrence Livermore National Lab.
P. O. Box 808, L-403
Livermore, CA 94550
(510) 423-2347
(510) 423-7884
wright20@llnl.gov

Rafi Yaari
Weizmann Institute of Science
Rehovot, Israel 76100
00-972-8-342669
00-972-8-344106
fhyaari2@weizmann.weizmann.ac.il

Yoshiji Yasu
KEK,
Natl. Lab. for High Energy Physics
Oho 1-1, Tsukuba
Ibaraki, Japan 305
0298-64-5383
0298-64-2580
yasu@kekvax.kek.jp

Glenn R. Young
Oak Ridge National Laboratory
P. O. Box 2008
Building 6003, MS 6372
Oak Ridge, TN 37831
(615) 576-2770
(615) 576-2822
young@orph01.phy.ornl.gov

Fukuko Yuasa
KEK,
Natl. Lab. for High Energy Physics
1-1, Oho Tsukuba City
Ibaraki, Japan 305
81-298-64-5481
81-298-64-3136
yuasa@kekvax.kek.jp

Lucia Zanello
University of Rome
P. le a.moro 2
Rome, Italy 2-00185
(00396) 4991-4-360
(00396) 4957697

Gerard Zehnacker
IN2P3/CNRS
Centre Recherches Nucleaires
23 Rue du Loess BP 20CR
Strasbourg, France F-67037
(33) 88 10 6584
(33) 88 10 6273
2ehnacke@crnsu1.in2p3.fr

Xing Zhao
University of Stockholm
Box 6730
Stockholm, Sweden S-113
46-8-164661
46-8-347817
xzhao@vana.physto.se

Giuseppe Zito
Sez. INFN Bari
c/o Instituto di Fisica
via amendola 173
Bari, Italy I-70126
39 80 242433
39 80 334938
zito@vxcern.cern.ch

Ralf Zoz
University of Mannheim
Computer Science V, B6
Mannheim, Germany D-68131
(49) 621-292-5755
(49) 621-292-5756
zoz@mp-sun6.informatik.uni-mannheim.de

Lawrence Berkeley Laboratory
University Of California
Berkeley, California 94720