Distributed Control of Protein Crystallography Beamline 5.0 Using CORBA*

C. Timossi, C. Cork, Advanced Light Source, Berkeley Lab, USA

Abstract

The Protein Crystallography Beamline at Berkeley Lab's Advanced Light Source is a facility that is being used to solve the structure of proteins. The software that is being used to control this beamline uses Java for user interface applications that communicate via CORBA with workstations that control the beamline hardware. We describe the software architecture for the beamline and our experiences after two years of operation.

1 MACROMOLECULAR CRYSTALLOGRAPHY FACILITY

1.1 Background

The Macromolecular Crystallography Facility (MCF) at Berkeley Lab's Advanced Light Source was established in 1995 in order to develop a full-service facility for protein and large molecule crystallography. Its mission, over an extended lifecycle of 10-15 years, is to develop and maintain a number of x-ray beamlines and experimental stations. The first MCF beamline, BL05.0.2, was put into production for experimenters in September 1997. This beamline collects multiwavelength datasets for as many as 30 samples in one week, producing approximately 100 GByte of image data. Two additional branchlines (BL05.0.1 and BL05.0.3) are currently under construction and more are in the design phase. These stations are expected to be in constant use and will require very high reliability and minimum time to repair. In particular, software upgrades must be incremental and provide for rapid installation and testing.

1.2 Beamline BL05.0.2 Instrumentation

The x-rays from the accelerator are initially collimated by a vertically-focusing pre-mirror which has both tilt and focus controls. The proper energy is then selected by a double crystal monochrometer that adjusts the Bragg angle and crystal spacing. Tilt and roll of the second crystal are also adjustable. The next optical element is a re-focusing mirror, which is supported by a six strut parallel actuator (hexapod) giving complete control of tilt, roll and yaw as well as 3 degrees of translation. The final main component is the sample goniometer with 3 degrees of rotation. In addition to these main components are control and monitoring of diagnostics such as beam position monitors, multiplexors and TV screens.

2 CONTROL SYSTEM

2.1 Performance

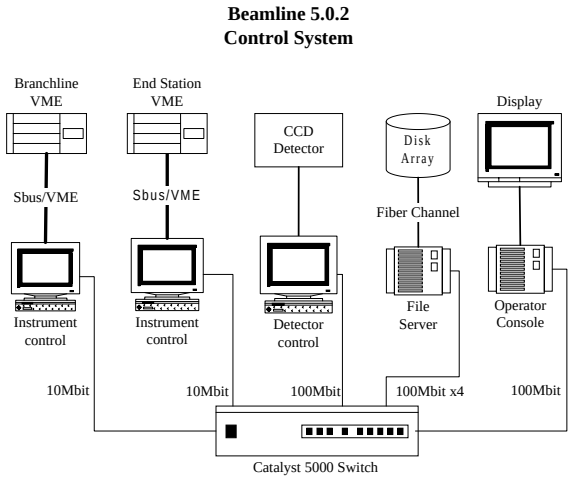
The two main categories of computer control at the MCF are motion control (mirrors, monochrometer, and sample goniostat) and detector control and data acquisition. The performance requirements for the control system are not severe. Performance is limited by the times for the mechanical motion of the mirrors and monochrometer and by the rate of data collection, which is dominated, by the readout rate of the CCD detector. The data rate between the control computers and the operator console is easily handled by the ~1ms/call measured for our current Common Object Request Broker (CORBA[1]) implementation from Sun Microsystems, NEO[2]. More severe were the requirements for high reliability, little availability of the production system for testing, and minimal programming and engineering personnel.

2.2 Architecture

The control system architecture is illustrated in figure 1. This architecture reflects the distributed design that was necessary to accommodate both the extended nature of the facility and the parallel processing requirements for optics control and data acquisition. Since there are relatively few people to work on the control system (2), predictably we were drawn towards off-the-shelf hardware, such as workstations for both control and development, and off-the-shelf software, such as Sun's Solaris and NEO and high level object-oriented languages like C++ and Java. We decided to distribute the control across standard Ethernet (switched 10baseT) to small instrumentation servers (Sun SPARCstation-5 computers) that were bus-coupled to VME crates that house the motion control and beam monitoring modules. The detector control and data acquisition is distributed across fast Ethernet (switched 100BaseT and future 1000BaseT) using high performance servers (Sun Ultra-2 and Enterprise-3000) and workstations (Sun Ultra-10). This decision gave us a good

* This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Material Sciences Division, U. S. Department of Energy, under Contract No. DE-AC03-76SF00098

distributed control environment with soft real-time capabilities. Fast feedback and coordinated sequencing are performed either in hardware or at the local control level. Closed-loop motion control is performed within the motion control modules.



The software follows a client/server model with server modules, written in C++, running on the instrumentation workstations handling the individual instruments. The services provided by these modules are available to clients, such as operator display screens, which are primarily written in Java. Services are located using NEO name services. NEO also handles the tricky problems of object life cycle and initialization, server object locking when multiple clients are requesting services, object error logging and general distributed server and object management. We have found these tools to be extremely reliable and robust.

2.3.2 Instrument Server Software

As mentioned previously, server modules were developed to run on the instrument server machines to control the various instruments. The servant skeleton C++ code is auto-generated by the NEO IDL compiler. This skeleton basically handles the communication with the clients and calls the custom modules that we develop to talk to the hardware. Last year we started using the ACE[3] development tools to develop these custom modules with the objective of making the handler code platform independent.

2.3.3 Data Acquisition Software

The data acquisition software presented a special challenge since it is a large vendor supplied application based on socket level communication to coordinate distributed processes handling the Detector interface, DAQ sequencer, goniostat & monochrometer control, data reduction, and data monitoring. For the application to communicate with the goniostat and monochrometer, we installed hooks enabling CORBA communication to these services.

2.3.4 Operator Interface Software

Java Applets were developed to allow operator control of individual instruments and to scan the monochrometer to display energy versus intensity plots. Skeleton Java code is produced by a Java IDL compiler for communication with the instrument servers while the GUI portion was developed using the Symantec Visual Café Java development software. Although the development software runs on Microsoft Windows, the resultant classes run on the operator console's Solaris platform without modification.

The data collection application is a third party product that was modified to operate with our detector.

3 SUMMARY

After two years of operational experience with this control system design, we are pleased with the robustness and ease of development. The hardware, the Solaris OS, the use of workstations to run and develop the instrumentation code and the Sun's implementation of CORBA (NEO) have all proven to be good choices in these regards.

The main limitation that we've seen using workstations bus-coupled to VME is the slow response for handling board level interrupts. For harder real-time applications this issue would need to be addressed.

Unfortunately, Sun no longer supports its NEO product so we are now investigating other CORBA implementations looking for similar functionality and performance.

REFERENCES

[1] Object Management Group, "The Common Object Request Broker Architecture and Specification", 2.2 ed., Feb. 1998.

[2] SunSoft, "NEO Programming Guide", 1995, Sun Microsystems .

[3] D. C. Schmidt, "ACE: An Object-Oriented Framework for Developing Distributed Applications", Proceedings of the Sixth USENIX C++ Technical Conference, USENIX Association, April 1994.