Interactive Analysis of Large Network Data Collections Using Query-Driven Visualization

E. Wes Bethel, Scott Campbell, Eli Dart, Jason Lee, Steven A. Smith, Kurt Stockinger, Brian Tierney, Kesheng Wu

Abstract—Realizing operational analytics solutions where large and complex data must be analyzed in a time-critical fashion entails integrating many different types of technology. Considering the extreme scale of contemporary datasets, one significant challenge is to reduce the duty cycle in the analytics discourse process. This paper focuses on an interdisciplinary combination of scientific data management and visualization/analysis technologies targeted at reducing the duty cycle in hypothesis testing and knowledge discovery. We present an application of such a combination in the problem domain of network traffic data analysis. Our performance experiment results, including both serial and parallel scalability tests, show that the combination can dramatically decrease the analytics duty cycle for this particular application. The combination is effectively applied to the analysis of network traffic data to detect slow and distributed scans, which is a difficult-to-detect form of cyberattack. Our approach is sufficiently general to be applied to a diverse set of data understanding problems as well as used in conjunction with a diverse set of analysis and visualization tools.

Index Terms— (H.2.8.h) Interactive data exploration and discovery, (I.6.9.d) Multivariate visualization, (K.6.M.b) Security, (J.8.o) Traffic Analysis.

1 Introduction

Visual Analytics is defined in [Thomas2005] as "the science of analytical reasoning facilitated by active visual interfaces." It is motivated by the need to gain understanding of features, trends and anomalies present in large and complex data collections. While a thorough discussion of the immense scope of all possible technical challenge areas and motivations is well beyond the scope of this paper, interested readers are directed towards [Thomas2005], which is a broad survey of the current state of research and development challenges in the field. From that broad set of challenges, one in particular is the focus of this paper: how to quickly find "interesting" data in large, multidimensional collections of information. We explore this topic within the context of a cybersecurity application, namely network traffic analysis.

Network traffic datasets consist of records containing a number of variables that summarize a particular network connection, or "conversation" between two hosts on a network. These data – known as "connection records" – are generated by border routers, traffic analyzers or security systems, and contain such information as source and destination IP address of the conversing hosts, the source and destination ports, duration of connection, number of bytes exchanged and so forth. There is no information about the actual traffic content, only information about the two hosts participating in the transmission, duration of the connection and related data. With the explosive growth of the Internet, there is a corresponding rise in the amount of information collected about network connections as well as an increase in the number of anomalous events. Such events may be indicative of a misconfigured host or network, an inappropriate use of resources, an attack on a computer system or network, a compromised host, or any one of a number of other items of interest. Collecting, managing and understanding the growing amount of network connection data in a timely fashion all present substantial challenges for network operations personnel.

A broad view of the network traffic analysis problem would necessarily include data collection, data storage and management, automatic feature detection, event characterization, analytical discourse to understand features and discover their relationships along with timely response to a particular incident. The work we present here explores a of subset the complete network traffic analysis problem. Namely, we focus on a multidisciplinary approach to feature mining and hypothesis testing by combining scientific data management tools for indexing and querying with simple visualization tools. The overall objective of our work is to reduce the duty cycle in hypothesis testing and feature mining. This paper describes how we achieve that objective within the context of a network traffic analysis case study. The results are particularly relevant given the explosive growth in network traffic and network traffic data. To be effective, a complete visual analytics solution will need to address problems of scale, to make effective use of high performance computing resources, and to quickly provide answers to analysts. Arguably, effective data management is the cornerstone for all such applications.

The rest of this paper is structured as follows. Section

- *Bethel, Campbell, Lee, Stockinger, Tierney and Wu are Computer Scientists at Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley CA 94720. E-mail: [ewbethel | scampbell | jrlee | kstockinger | bltierney | kwu]@lbl.gov.*
- *Dart is a Network Engineer with the Energy Sciences Network (ESnet) at Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley CA 94720 USA. Email: dart@es.net.*
- *Smith is a Computer Scientist at Los Alamos National Laboratory, P.O. Box 1663, Los Alamos NM, 87545 USA. E-mail:sas@lanl.gov.*

Two presents a survey of previous work in fields related to the topic of this paper. Since our work realizes new capability by integrating and applying ideas from several different fields in a multi- and inter-disciplinary fashion, we discuss prior work from several different but related fields. Next, we present a series of performance studies that focus on highlighting the relevance of our approach to realistic-sized collections of network connection data. Specifically, we conduct a serial performance test that shows an order-of-magnitude decrease in the time required to answer a typical query on a realistic-sized network connection dataset. We follow with a series of tests that show the scalability characteristics of our approach. Next, we present a network data analysis case study conducted by network analysts using the technology we present in this paper. Finally, we conclude with general comments and potential future research directions.

2 Background and Related Work

In recent years, there has been a tremendous increase in research and development projects in the areas of network traffic analysis and visualization. Related to the material we present here is previous work in the fields of scientific data management, particularly techniques for efficient indexing and querying, as well as visualization systems that focus on limiting visual processing only to data deemed to be "of interest" to the viewer.

2.1 Network Traffic Analysis

There exists an increasing need for visual analytics tools in the fields of network analysis and forensics. Raw traffic logs have long been too large and complex for a human to digest and understand. In particular, there is a need for tools that provide insight into patterns in data sets that are large in volume, time or both. Intrusion Detection Systems (IDS) are typically good at analyzing events that are closely correlated in time, and where analysis can rapidly yield actionable results. However, over large time scales and/or very large data sets, the design decisions that make IDS software appropriate for rapid response limit applicability to problems of scale. Typical failure modes include memory consumption that grows without bound, or computational overload that inhibits a rapid response. Moreover, connection analysis lends itself to visual analytics, since features and patterns that are easily visible to a skilled analyst are often difficult to quantify precisely or to detect programmatically. Therefore, visual analytics tools are best thought of as a complement to IDS software, and part of a broad technology portfolio in the network analyst's toolbox.

Traditional network traffic analysis usually begins when the IDS provides an alert, such as one or more IP addresses associated with a possible attack. If the analyst will be examining short-term connection data – less than 24 hours' worth of data – then the connection logs are dumped straight to local disk. These logs can usually be obtained in 10 seconds or less. For analysis that spans a longer range of time, a dedicated system is available that can process connection logs at the rate of 5-7 minutes per month for a single IP address. Multiple IP addresses can be processed simultaneously. In all cases, the tool used for the search step is *grep*. Once the subset of interesting connection data have been extracted from the larger set of logs, the analyst performs more specialized processing pursuant to the particular line of inquiry. Initial queries – search terms – based on anything other than a single column are rarely performed. Post-processing of search results is performed using *perl*, shell scripts and *gnuplot.* The time required for these steps is proportional to the amount of script development and analysis required for the particular incident.

Historically, the problem with large-scale analysis tools for network traffic has been that the duty cycle for testing a given hypothesis is measured in hours for non-trivial data sets. Therefore, a reduction in turnaround time from hours to seconds represents an unprecedented new capability. The new capability migrates large-scale network connection analysis from the realm of overnight batch jobs to that of interactive tools.

Visualization techniques permit an analyst to look at thousands or hundreds of thousands of connections simultaneously. Simple data filtering is both valuable and necessary to deal with the huge volume of data and the large dynamic range exhibited by many of the attributes. Applying multi-dimensional transformations, deriving statistical quantities and applying clustering techniques provides new and often more relevant quantities to visualize, as well as more relevant keys for indexing and querying.

A network connection can be thought of as a set of packets passing a point on a network within a given time interval that have common characteristics. An example of a network connection is a single communication session or an interaction between two hosts on the Internet. For this discussion we describe connection dynamics in terms of TCP session characteristics. Several standard tools exist for capturing network connection data. *Tcpdump* is one of the most commonly used; it is a pcap-based application that can run on most operating systems and logs network traffic based on a filtering expression ([JLM89] [MLJ94]). For larger environments, routers and switches provide NetFlow [NetFlow05], LFAP [LFAP97] or SFlow data [Sflow2001]. Such data contains either complete or sampled profiles of traffic observed by the network device. Special purpose systems and software have been implemented for various reasons, including efficient and flexible handling of the huge data rates and volumes seen on larger networks ([Uphoff04], [Smacq02], [Bro]). Individual network services like HTTP are application-level services built atop transport-level protocols (TCP, UDP). Some network connection data collection and reporting systems generate a separate record for each direction of bidirectional connection [Bro]. Other systems generate a single full-duplex connection record that also contains byte and packet counts for the reverse direction [Uphoff04]. Complicating the job of network analysts is a growing number of ad-hoc application level protocols not associated with normal services.

A network connection record generally contains at least the following information:

1. Source IP address,
2. Destination IP address,
3. Source Port,

4. Destination Port,
5. Byte and packet count sent by source,
6. Byte and packet count sent by destination,
7. Start and End time in milliseconds.

A typical day's worth of traffic at an "average" government research laboratory might involve tens of millions of such connections comprising multiple gigabytes worth of connection records. A year's worth of such data currently requires on the order of tens of terabytes or more of storage. According to [Burrescia05], traffic volume over ESnet, a production network servicing the U. S. Department of Energy's research laboratories, has increased by an order of magnitude every 46 months since 1990. This trend is expected to continue into the foreseeable future.

2.2 Network Traffic Visualization

There has been a good deal of work in recent years in the area of interactive network traffic visualization. A thorough survey of such work is outside the scope of this paper since we are focusing on coupling data management technology with network traffic visualization and analysis tools. See [Livnat2005] for a good survey of previous work in this area. Generally speaking, previous work has focused on filtering and visual presentation of different types of network traffic data, including connection data, routing information, IDS alerts and so forth.

Visualization applications aimed at providing overall situational awareness by visualization network connection data are described in ([Lau04], [McPherson04]). These applications map network connection variables to axes, then present activity, or lack thereof, at the appropriate grid location. The basic idea is to facilitate rapid visual discovery of incidents or other features. [McPherson04] offers the ability to reduce the amount of data displayed by allowing the user to interactively manipulate filters. In this way, a user can focus visual inspection only on data that matches the filtering criteria. In both these examples, the problem size consists of datasets comprised of a few thousand unique network connections.

Komlodi et al. [Komlodi05] describe an Intrusion Detection Toolkit that supports a number of different visualization techniques for viewing alert or packet data. This system implements a form of data reduction through filtering, either as part of the interactive visualization activity or as a separate process. This work is silent on the subject of filtering, or query methodology and performance, and shows examples of what appear to be small datasets.

The VisAlert system described in [Livnat2005] presents a visualization paradigm for correlating network alerts generated by multiple sensors deployed across a network. The paradigm is based on the observation that every network alert must possess three fundamental attributes – what, when and where – that in turn provide a consistent basis for correlation. VisAlert uses a unique and flexible two-dimensional display metaphor that is effective in helping users to switch between context/focus modes of inspection. For their results, they use a dataset consisting of 12-hours' worth of network traffic.

While these previous works in network traffic visualization have produced novel and useful presentation and interaction techniques, they were tested using only small amounts of network data. Our approach in this paper is complementary to these works: we extend the idea of filtering out of the visualization application and into the domain of high performance scientific data management systems. We believe such an approach is required to gain traction on analysis and visualization of realistic-sized volumes of network traffic and thus offers a completely new and complementary capability to the field of network traffic analysis and visualization.

2.3 Query-Driven Visualization

The term "Query-Driven Visualization" refers to the process of limiting visualization processing and subsequent visual interpretation to data that is deemed to be "interesting." This idea has been implemented in a diverse range of research projects and applications. Several factors contribute to the overall motivation for the query-driven visualization approach. As data grows larger and more complex, simply building larger, more scalable visualization systems produces a greater amount of output, which in turn increases the cognitive load on the viewer. In some cases, increasing the amount of visible output may actually hinder understanding as depth complexity increases, important features are "hidden," and so forth. Similarly, with increasing data size and complexity, finding and displaying relevant data becomes increasingly important to foster scientific understanding and insight. The query-driven visualization approach also offers a new foothold for gaining traction on the challenges of temporal and multidimensional visualization and analysis as processing can be focused on "sets" of data that satisfy the conditions of a given line of scientific inquiry. An example here might be "what is the average electrical charge of particles in an accelerator model that that spin away from the main beamline and strike the accelerator wall over the course of the entire simulation?"

The VisDB system combines a guided query-formulation facility with relevance-based visualization [Keim94B]. Each data item in a dataset is ranked in terms of its relevance to a query, and the top quartile of results is then input to a visualization and rendering pipeline. Data is presented in a way to cluster more relevant items closer together, and less relevant items further apart. It is especially well-suited to display the results of "fuzzy queries" in that inexact matches are ranked and visually displayed in a way to convey relevance.

The TimeFinder system described in [Hoscheiser01] supports interactive exploration of time-varying data sets by providing the ability to quickly construct queries, modify parameters, and visualize query results. A query is formed by manually "drawing" a rectangular box on a 2D plot where the x-axis represents time and the y-axis represents the data range. Each such rectangular box is called a "timebox" and each comprises a range query. A user forms a multidimensional range query through the union of several timeboxes. The query results are presented in a fashion that implements a form of data mining – more detailed information about the items satisfying the query are presented in a separate window. TimeFinder reads all data into memory and is therefore able to operate on only modest-sized datasets.

The Scout software system provides the ability to perform expression-based queries using a simple programming language along with visualization, where both queries and visualization are executed entirely on a GPU [McCormick04]. A Scout program, which is realized as fragment assembler, operates on source data that is loaded as an OpenGL texture on the GPU. The program is executed during rendering: two-dimensional data is rendered as a single quadrilateral and three-dimensional data is rendered as view-aligned slices in back-to-front order as direct volume rendering. As described, Scout is limited in data size by the amount of GPU memory and is subject to the floating-point accuracy of the GPU.

Recently, the idea of coupling a visualization pipeline with a high performance index and query system was described in [Stockinger05]. That work shows that the computational complexity of visualization processing can be constrained to the number of items returned by a query. As such, that approach is the most suitable for use in query-driven visualization and analysis of very large multidimensional datasets.

We are extending the work of [Stockinger05] in this paper by applying indexing and querying techniques to network connection data, by providing a preliminary study of parallel performance and scalability, and by demonstrating their applicability within the context of a "hero-sized" network connection data analysis problem.

2.4 Indexing and Querying

One approach for examining a large amount of network connection data is to focus attention on a relatively small number of "interesting" connections. The naïve approach of checking every connection to see if it satisfies the definition of "interesting" may take too long or be impractical. The basic strategy to accelerate the selection process is to use an indexing structure [Knuth98]. Most well known indexing structures are designed for data that are frequently updated, like bank transactions. In banking applications, when some arbitrary record is modified, the index structure must be similarly updated. In this type of application, the two main functions that an indexing structure provides – querying and updating – both need to be very efficient. In fact, for many tree-based indexing structures, the time required for updating a record is nearly the same as the time required for locating a record. Network connection data is different from these types of transactional databases in that the existing records are never modified. The only change to the data is the addition of new records. For this type of data it is possible to sacrifice some update efficiency in favor of a significant gain in query efficiency. Bitmap indexing technology is an excellent example of this tradeoff.

Bitmap indexing has been implemented in commercial database systems and it is well accepted that it is efficient for low cardinality attributes where there are few distinct values [O'Neil87]. Recently, it was shown that such efficiency can be extended to high cardinality attributes with Word Aligned Hybrid coding (WAH) [Wu04]. FastBit [FastBit] is a research code that implements a number of different forms of bitmap index compression, including WAH.

In a basic bitmap index, one bitmap is allocated for each distinct value of the indexed attribute, where each bitmap has as many bits as the number of records in the indexed dataset. The size of the index grows linearly with the attribute cardinality and is small only for low cardinality attributes. A number of strategies have been proposed to reduce the size of a bitmap index, including binning ([Shoshani99], [Stockinger00]), multi-component encoding [Chan98], and compression ([Johnson99], [Wu01]). In particular, WAH compression was proven to keep the index sizes compact as well as to significantly reduce the query processing time compared to other indexing schemes [Wu04].

In this paper, we compare the performance of a WAH compressed bitmap index with one called a *projection index* [ONeil97]. The projection index extracts the attribute values and stores them separately so that when answering a query, only the attributes involved in the query are read into memory. This approach is known to work well for range queries, which are commonly used for analysis of large datasets.

While exploring network connection data, an analyst might use a query of the form "StartTime > 20050501 and 10.102.0.0 <= SourceIP <= 10.105.255.255." In this example, each term like "StartTime > 20050501" is called a *range condition*. In a typical exploration, the analyst may specify a number of different range conditions on different attributes. Such queries are typically referred to as *ad hoc* since they do not follow a predefined pattern. With ad hoc queries, the bitmap index has a unique advantage over tree-based indexing structures because the answers to individual range condition can be efficiently combined.

Most tree-based indexing methods suffer from the "curse of dimensionality." As the number of attributes in a dataset increases, tree-based indices become progressively less competitive against the projection index. Typical multi-dimensional indexing tree-based – kd-trees, for example [Bentley75] – usually index all variables or dimensions of a dataset. When answering an ad hoc range query involving only a few variables, or dimensions, tree-based multi-dimensional indices are much less efficient than the projection and bitmap indices, which do not suffer from the "curse of dimensionality."

To answer multidimensional range queries, we first use the bitmap indices to resolve each individual range condition and then combine the partial solutions with bitwise logical operations. The time required to resolve each range condition is proportional to the size of the bitmaps involved. Moreover, the overall query processing time grows linearly with the number of range conditions specified. The time required by the projection indices also scales linearly with the number of range conditions, however, the time required to resolve each individual range condition using a projection index is typically much longer than that of a bitmap index as we will empirically show later in Section 3.3, *Performance Study*. Additionally, the query response time of both the bitmap index and the projection index scales linearly as the number of records increase [Wu04]. These scaling properties indicate that both projection and bitmap indexing are well suited for large datasets like those encountered in network connection analyses.

3 Query-Driven Network Traffic Analysis Performance Study

In this section we analyze the performance of our prototype query-driven network traffic analysis application. We begin with a brief description of the network traffic data we use in our experiments. Next, we present details about the query and display implementation. Finally, the performance studies indicate excellent serial performance of FastBit compared to a projection index system, as well as favorable scalability characteristics. These results show that by combining visualization with high performance scientific data management technology, we are able to perform interactive queries – one of the fundamental elements of visual analytics – on realistic-sized network connection datasets.

3.1 The Network Traffic Data

For our performance study, we run queries on two different datasets. One consists of 24 weeks' worth of network connection data collected from Bro at LBNL containing about 1.1 billion records. The other consists of 42 weeks' worth of network connection data, also collected from Bro at LBNL, containing about 2.5 billion records. Each of the two data sets contains 25 attributes, including the source IP address, the destination IP address, source port, destination port, start time, duration, number of bytes sent along with additional network connection information. The total sizes of each are 146.5 GB for the 24-week dataset and 281.7 GB for the 42-week dataset. Each week's worth of network traffic data is stored as a separate file. Note that this distribution of data is not necessarily the most efficient for parallel computing but is convenient for the analysts who manage the data and interpret the query results.

Given the raw data, we construct bitmap indices for each attribute. The total index size for each of the two datasets is 44.4 GB and 78.6 GB, respectively. Note that in both cases the size of the bitmap indices is about a third of the size of the raw data. This is fairly small compared to B-trees, which are the most commonly used index for transactional database systems. B-tree index structures are often three times larger than the raw data.

3.2 Query and Visual Display Implementation

The prototype implementation of our query and display system is based on ROOT [Brun97], which is an object-oriented data analysis system originally developed for scientific analysis and data management of large volumes of high-energy physics data. The ROOT system has a comprehensive set of analysis capabilities and basic visualization features. ROOT is straightforward to extend through loadable modules. We extended ROOT so that it can answer multidimensional range queries using FastBit.

In the database community, most data are viewed as tables where each row represents one record or a data object. Most existing data management systems physically cluster attributes of a record both on disk and in memory. This storage organization is called "horizontal data organization". In addition, ROOT also supports "vertical data organization," which is commonly known as a "projection index". Since the projection index is efficient for the type of multidimensional range queries that an analyst would request when studying network connection data, we have organized all of our data into projection indices to ensure the best possible query processing performance when using the ROOT-only query engine in our performance study.

The analyst's multidimensional range queries on network connection data select a subset of records and return the values of a small number of attributes for further study. In the database community, this type of operation is known as "selection and projection." The selections and projections can be parallelized in a straightforward manner by distributing records amongst a number of processors. Because the number of traffic sessions can vary significantly from week to week, our per-week data decomposition is not necessarily balanced. We are currently preparing a detailed study of load and data balance as well as other issues related to scalability. In this paper, we will focus on single processor performance and provide an early glimpse of FastBit's scalability characteristics.

3.3 Performance Study

In this subsection, we present three separate performance measurements to show the efficiency of our ROOT-FastBit implementation. The first test compares the time required by projection and bitmap indices to answer a query on 24 weeks' worth of network connection data. The second test is a scalability study reporting ROOT-FastBit's parallel execution performance on 24 weeks' worth of network connection data. The third is a more extensive ROOT-FastBit scalability study using 42 weeks' worth of network connection data.

For the first two performance tests, we used Platform "J," which is an SGI Onyx3700 comprised of twelve 600Mhz R14000 MIPS processors, 24GB of RAM, and a 5TB fiber-channel RAID capable of delivering about 600MB/s in effective sustained I/O bandwidth. For the third test, we used Platform "D," which is an SGI Altix comprised of 32 1.4 Ghz IA64 processors, 192GB of RAM, 23TB of fiber-channel RAID capable of delivering about 500MB/s in effective sustained I/O bandwidth.

3.3.1 Serial Performance Comparison

For the first test, we compare the time required to answer a query for projection and bitmap index implementations on Platform "J." We use a typical query over three variables –"select IPS_B, IPS_C, IPS_D where IBS_B < 100 and IPS_C < 100 and IPS_D = 128." For the projection index test, we use ROOT with only its projection indices, i.e., without FastBit. The time required to answer the query above is 2467 seconds. By using FastBit's bitmap indices in our ROOT application, the same query was answered in 309 seconds. This represents an order of magnitude in performance gain simply by migrating from projection to bitmap indices. While these results were run using network connection data as the source, we have observed a similar performance gain in similar performance experiments using high-energy physics data in which the average performance gain of FastBit for processing multi-dimensional queries is about a factor of 10 [Stockinger2005b].

3.3.2 First Scalability Test

Next we measured the parallel query performance of ROOT-FastBit on the 12-processor Platform "J". As we mentioned previously, the 24-week data set that we use for our performance evaluation is divided into weekly chunks. Thus, each processor is responsible for executing queries on two weeks' worth of data. Using 12 processors, the ROOT-FastBit system answers query shown in Section 3.3.1 in 22.8 seconds – as opposed to 309 seconds when run on a single processor.

In order to better understand the ROOT-FastBit scalability characteristics, we performed the following experiment. First, we executed the query from Section 3.3.1 on twelve weeks' worth of data and varied the number of processors between one and six. Next, we ran the same query on 24 weeks' worth of data. Again, we measured the performance with one to six processors. Figure 1 shows an effective parallel speedup of about 80% when using two processors, about 70% when using four processors and about 60% when using six processors. We believe that the decrease in parallel efficiency with increasing processors is due primarily to load imbalance. Data is distributed such that each processor receives the same number of weeks' worth of data. A close examination of the data reveals that some weeks contain up to 10 times more data than others. Therefore, assigning an identical number of weeks' worth of data to each processor will result in processing and I/O load imbalance.

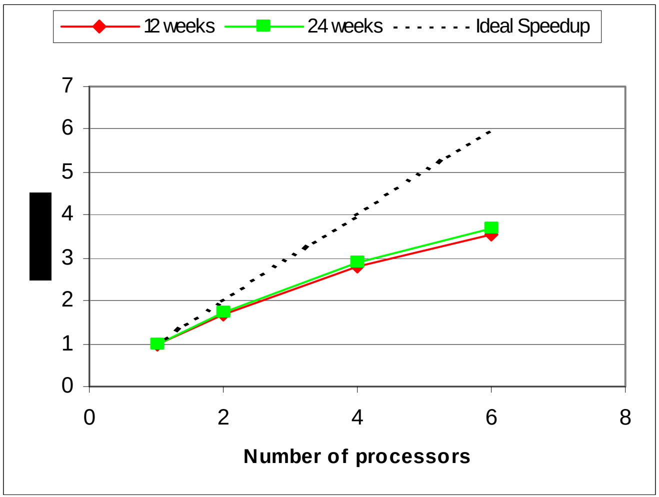

Figure 1. Parallel speedup factor when processing three-dimensional queries on Platform "J". For these tests, we measured the time required to answer the query specific in Section 3.3.1 for both 12 and 24 weeks' worth of data using varying numbers of processors. One six processors, we are realizing an effective parallel speedup of about 60%. The decrease in speedup results from processing and I/O imbalance caused by the fact that some weeks have up to 10 times more traffic than others.

3.3.3 Second Scalability Test

Over the course of this project – after conducting the preliminary serial and parallel tests – we wanted to expand our scope of testing to perform queries on ever-larger collections of network data to test our approach with "hero-sized" collections of network connection data. To that end, we collected and prepared indices for 42 weeks of network connection data.

In the second scalability test, we run queries of several different levels of complexity and a dataset twice as large and over a larger number of processors than in the first test. We ran tests that measure query response time over one, two, three and four variables when executed on up to twenty-one CPUs of Platform "D." As with the first set of parallelism tests, we used a weekly decomposition of data, and in each of the second set of scalability tests, each processor was assigned the same number of weeks. Also as with the first set of tests, this approach to data distribution does not ensure even balance of computation or I/O load since some network traffic is not evenly distributed from week to week. In these tests, we generated a set of random one-, two-, three- and four-dimensional range queries. There were a total of 320 unique one-dimensional, 271 two-dimensional, 223 three-dimensional and 223 four-dimensional queries.

Figure 2 shows the average speedup factor for these random one-, two-, three- and four-dimensional queries as well as the ideal speedup. We can see that with up to seven processors, the parallel query performance is close to ideal. The speedup factor decreases for a larger number of processors due to the load imbalance resulting from variance in the amount of traffic in each chunk of weekly connection data. Despite the processing and I/O load imbalance, the speedup factor is still significant. The average elapsed time to answer a large random collection of three-dimensional queries on 2.5 billions records using 21 processors is 19.34 seconds compared to 224.11 seconds using a single processor.

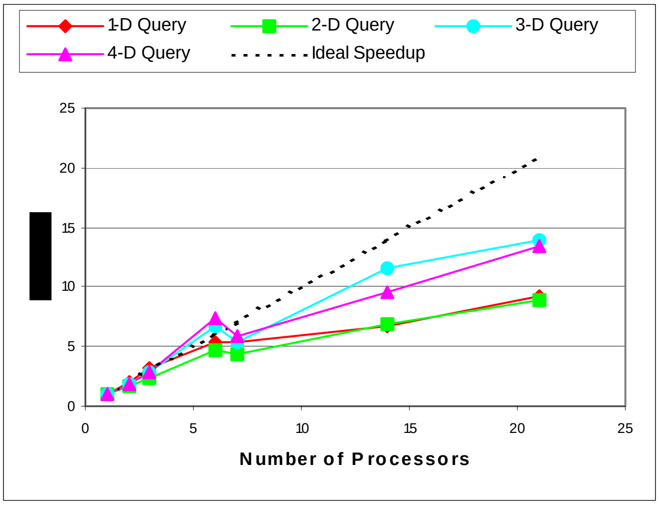

Figure 2. Speedup factor for multi-dimensional queries over 2.5 billion records on Platform "D". "Dim 1" means the query is over one variable, "Dim 2" means the query is over two variables, and so forth.

In Figure 2 we also see that the speedup for high dimensional queries is more significant than for low-dimensional ones. The reason is that the measurements include the time for processing the bitmap index (Step 1 of responding to the query) as well the as the time for retrieving the results (Step 2). We know that for high dimensional queries the result size accumulated over all dimensions is higher than for low dimensional queries. In other words, for high dimensional queries, there are more data elements to be read in Step 2. Since our experiments are executed on a parallel file system, Step 2 is automatically parallelized.

We can draw the following conclusions from our experiments: (1) FastBit performs better than projection indices by an order of magnitude; (2) the two scalability studies show that the queries parallelize well even for large load imbalances, (3) using parallelization allows us to perform queries on a hero-sized data set in 19.34 seconds on average for a three-dimensional query using 21 processors.

4 Network Traffic Analysis Case Study

Within visual analytics, a growing problem space is the field of network security. The analytics challenge presented by this field are significant – large sites will often have between 50 million connection attempts per each day, and some types of analysis require examining weeks' or months' worth of log data. Interactive exploration of this data can help identify compromised hosts and can also aid in the understanding the nature of attacks.

4.1 Analysis Of Scan Data

For an example of the interactive analysis, we look at the details surrounding a suspected distributed scan. Any open site is constantly being probed for open, unpatched services that might be exploited. Most of these probes come from relatively unsophisticated attackers who use simple tools that naïvely locate and target unpatched systems or services. Somewhat more sophisticated attackers may use "botnets", which are collections of hosts that search using more subtle means – the task of scanning can be distributed across the entire botnet collective. These types of scans are described as distributed since each component host takes a small fraction of the destination address space and identification of the scanning activity is therefore more difficult.

In this example, we are interested in determining if a set of hosts from a remote subnet is taking part in such a distributed scan. We have reason to think that this may be the case since all the identified hostile addresses are within the same class-C subnet and are scanning for the same service. Note that we have not specified how we arrived at the conclusion that a scan is underway – such is typically the job of IDS facilities. Instead, we are focusing on the forensics part of the analysis in these examples.

To see if there are other, unidentified hosts from the same subnet, we look at the third and fourth octet of the source IP address for all hosts coming in from the suspect address range and attempting to connect on a particular destination port. If the source subnet is 10.95.C.D, we create a scatterplot such that two of the axes correspond to the C and D address octets, and the third axis corresponds to time. This plot, which is shown in Figure 3, clearly indicates that there are many unique hosts that seem to be involved with this incident. To create this plot, we pose a four-dimensional query of the form "IPS_A = 10 AND IPS_B = 95 AND T1 <= Time <= T2 AND DestPort = P" and then display the results.

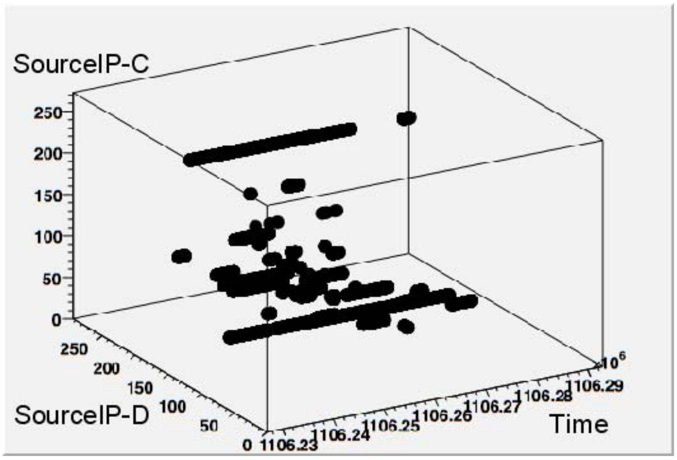

Figure 3. Shows the results of a three dimensional query where we are focusing on activity from a specific /16 address group within a given time range where connection attempts are made on a specified port.

While Figure 3 displays what amounts to only about one week's worth of data, we are extracting that week's worth of data from a total of 24 weeks worth of source data. To extract the temporal subset of data used as the basis for the query and plot shown in Figure 3, we used the ROOT-FastBit implementation to create a smaller subset from the 24-week dataset. While it is certainly possible to broaden the query to display connections from the suspect /16 address range attempting to connect to the particular port, doing so would not be beneficial in this case since we are interested in activity within a particular range of time. From the smaller subset, which contains about one days' worth of connection data, the time required to perform the query and generate Figures 3, 4 and 5 was about two seconds on a single processor of Platform "J."

For a larger view of how the local network block is being scanned, we transform the view so that all connection attempts from the (hostile) 10.95.0.0/16 subnet are plotted in terms of the destination octets into the local 10.1.0.0/16 net block. If a connection goes to 10.1.2.3, we plot (2,3) in the one plane with time along the third axis. As seen in Figure 4, we make several observations. Line-like features indicate linear trolling along one specific subnet (such as 10.1.2.0/24), while the larger planar or box structures may be wholesale scanning of the entire subnet. The "dust" surrounding these general forms could be noise, or part of a structure that falls outside of the selection criteria.

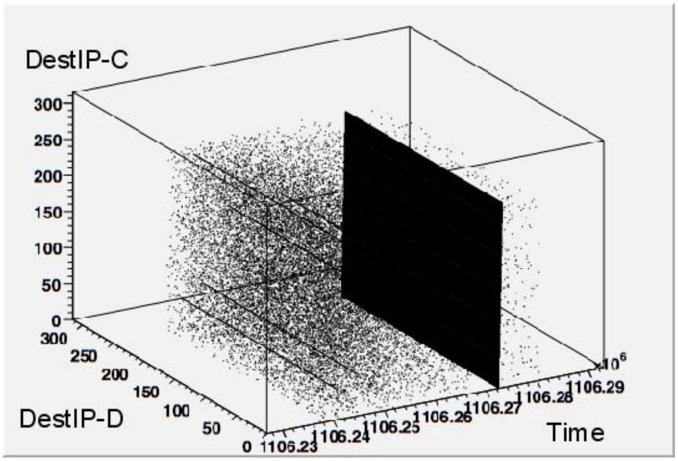

Figure 4. This scatterplot shows connection attempts on a specified port where two of the axes are the destination addresses and the third axis is time. Linear structures indicate scans through an address range on the local (target) network, while the plane-like structure indicates wholesale scanning of the entire subnet.

Now that there is a general understanding as to what the connections look like in terms of time, we can link together source and destination for all the connections to the target port to see what relations can be found. To do so, we keep the X- and Y- axes exactly the same as before, but spread out the range of source addresses in the following manner. If a connection has a source address of 10.95.A.B, the last two octets can be expressed on a single axis by multiplying the third octet by a constant, and adding the fourth. In this case we have Z=(255*A)+B. The results of this can be seen in Figure 5.

Looking at Figure 5, we see that several naïve scans completely cover entire subnets or net blocks as shown in Figure 4. In addition there are at least four discrete addresses that seem to be associated with a distributed scan by the degree of overlap coupled with the relative small number of connection attempts per source IP. Plotting the density of connection attempts across destination addresses provides the final indication that this is in fact a distributed scan. If it were not, than the relative variation in density across the local address space would be significantly higher.

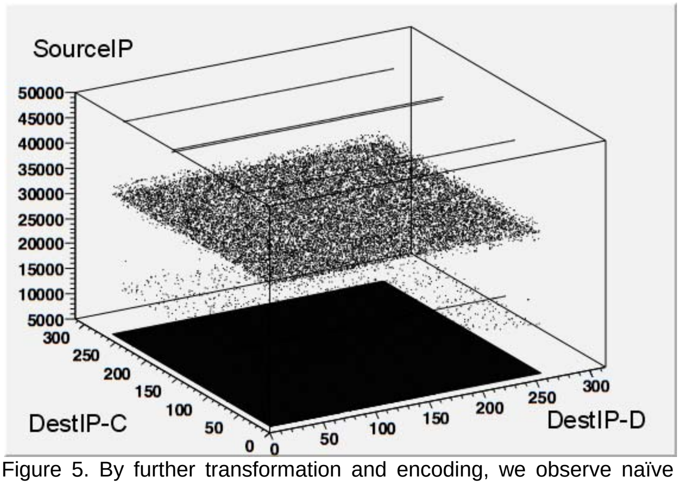

Figure 5. By further transformation and encoding, we observe naïve scans from a large number of hosts, and distributed scans associated with only a few hosts.

5 Conclusion and Future Work

The main point of this work is to demonstrate a new capability for network connection data analysis. This new capability results from combining technologies from different fields in an interdisciplinary fashion. Our work combines technology from scientific data management, data analysis and visualization. The results we present here show an order of magnitude in performance gains are possible in serial configurations through the new technology combination, and that an additional order of magnitude in performance gain is realized through parallelization. In terms of network security analysis, the significance of our work is that two network engineers on our team demonstrate interactive analysis of network connection data on realistic-sized datasets using the technology described in this paper.

The work we present here is best viewed as proof-of-concept. It does not address several important issues that would impact general usability. For instance, it offers no help in formulating the initial query. There is some previous work in this area, and extending those prior works to use the information easily obtainable from the bitmap indices would be a helpful part of an assisted query formulation facility. The work we present here uses rudimentary visualization techniques as part of the analysis. There are many useful and interesting network data visualization and analysis applications that would stand to benefit from advanced index and query capability – they would benefit from the ability to process realistic-sized datasets. Related, there are a number of different tools and techniques for performing automatic clustering, feature identification and tracking. Coupling those tools and techniques with advanced index and query capabilities is a promising area in terms of analyzing larger and more complex data.

The scalability studies we present here are best viewed as preliminary but hopeful glimpses of the potential of our approach. More detailed scalability and performance studies with different types of scientific data would provide a well-rounded characterization of its capabilities.

Another important future application of the work we present here is in the area of generating “anonymized” datasets for use in algorithm testing. A long-standing difficulty in the field of network analysis algorithm development is striking a balance between privacy concerns and the need for developers to have access to “live” data. Typically, a researcher is interested in performing analysis or algorithmic development on a subset of the entire connection collection since many results do not require accessing the entire set at the same time. Such subsets may be defined by port use, IP or time slice. For example, to characterize mail server traffic, one might only be interested in mail related ports in one-month blocks over some particular time period. Index and query engines – like FastBit – would be highly instrumental in quickly creating smaller, portable data sets that contain all relevant connection information. The portable data sets could be in the same format as the larger one so that tools that are designed to run against one would be able to run against the other without modification.

ACKNOWLEDGMENT

This work was supported by the Director, Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, the National Nuclear Security Administration, and the Department of Homeland Security National Visualization and Analytics Center. The authors would also like to thank the Computational Systems Group at the National Energy Research Scientific Computing Center, the University of New Mexico Center for High Performance Computing, and to Rene Brun of CERN and the ROOT project for assistance porting ROOT to one of the computer systems used in our performance study.

REFERENCES

[1] [Bentley75] J. L. Bentley. “Multidimensional Binary Search Trees Used for Associative Search,” *Comm. ACM*, 18(9):509–516, 1975.

[2] [Bro] V. Paxson. "Bro: A System for Detecting Network Intruders In Real-Time." *Proceedings of the 7th USENIX Security Symposium,* San Antonio TX, January 1998.

[3] [Brun97] R. Brun and F. Rademakers, “ROOT – An Object Oriented Data Analysis Framework,” in *Proceedings of the AIHENP’96 Workshop,* Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86, Lausanne Switzerland.

[4] [Burrescia05] J. Burrescia and W. Johnston, “ESnet Status Update,” Internet2 International Meeting, Philadelphia PA, September 2005.

[5] [FastBit] URL http://sdm.lbl.gov/fastbit, 2005.

[6] [Hinneburg99] A. Hinneburg, D. Keim, M. Wawryniuk. “HD-Eye: Visual Mining of High Dimensional Data.” IEEE Computer Graphics and Applications, 19(5), September/October 1999, pp. 22-31.

[7] [Hochheiser01] H. Hochheiser and B. Shneiderman. “Visual Specification of Queries for Finding Patterns in Time-Series Data,” *Proceedings of Discovery Science 2001*, pp441-446, Washington DC USA, 2001.

[8] [JLM89] V. Jacobsen, C. Leres, and S. McCanne, “tcpdump”’, URL: ftp://ftp.ee.lbl.gov, 1989.

[9] [Johnson99] Johnson, T. 1999. Performance measurements of compressed bitmap indices. In VLDB99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK.

[10] [Keim94] D. Keim. *Visual Support for Query Specification and Data Mining*, Dissertation, July 1994, Shaker Publishing Company, 1995, ISBN 3-8265-0594-8.

[11] [Keim94B] D. Keim and H. P. Kriegel, “VisDB: Database Exploration Using Multidimensional Visualization,” *IEEE Computer Graphics and Applications*, 14(5):40-49. 1994.

[12] [Knuth98] D. E. Knuth, 1998. The Art of Computer Programming, 2nd ed. Vol. 3. Addison Wesley.

[13] [Komlodi05] A. Komlodi, P. Rheingans, U. Ayachit, J. Goodall, A. Joshi, “A User-centered Look at Glyph-based Security Visualization,” in *Proceedings of the 2005 Workshop on Visualization for Computer Security*, Minneapolis MN USA, pp21-28, 2005.

[14] [Lau04] S. Lau, “The Spinning Cube of Potential Doom,” Communications of the ACM, 47(6):25-26, 2004.

[15] [LFAP97] P. Amsden, J. Amweg, P. Calato, S. Bensley, G. Lyons, “Cabletron’s Leightweight Flow Admission Protocol Specification, Version 1.0,” IETF RFC#2124, http://www.ietf.org/rfc/rfc2124.txt, 1997.

[16] [Livnat2005] Y. Livnat, J. Agutter, S. Moon, R. Erbacher, S. Foresti. “A Visual Paradigm for Network Intrusion Detection.” *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point NY, 17-19 June 2005.

[17] [McCormick04] P. McCormick, J. Inma, J. Ahrens, C. Hansen and G. Roth, “Scout: A Hardware-Accelerated System for Quantitatively Driven Visualization and Analysis,” in *Proceedings of IEEE Visualization 2004*, pp171-178, Washington DC USA, 2004. IEEE Computer Society Press.

[18] [McPherson04] J. McPherson, K. L. Ma, P. Krystosk, T. Bartoletti, M. Christensen, “PortVis: A Tool for Port-Based Detection of Security Events,” in *Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security*, ACM Conference on Computer and Communications Security, October 2004.

[19] [MLJ94] S. McCanne, C. Leres and V. Jacobson, “libpcap”, URL: ftp://ftp.ee.lbl.gov, 1994.

[20] [NetFlow05] Cisco Systems, “Cisco Netflow Collection Engine,” http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/, 2005.

[21] [O’Niel87] P. O’Neil, 1987. Model 204 architecture and performance. *In 2nd International Workshop in High Performance Transaction Systems*, Asilomar, CA. Lecture Notes in Computer Science, vol. 359. Springer-Verlag, 4059.

[22] [O’Neil97] O’Neil, P. and Quass, D. 1997. Improved query performance with variant indices. In Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA, J. Peckham, Ed. ACM Press, 3849.

[23] [Sflow2001] SFlow] InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. Internet Engineering Task Force Request for Comment. URL: http://www.apps.ietf.org/rfc/rfc3176.html, 2001.

[24] [Shoshani99] Shoshani, A., Bernardo, L. M., Nordberg, H., Rotem, D., and Sim, A. 1999. Multidimensional indexing and query coordination for tertiary storage management. In 11th International Conference on Scientific and Statistical Database Management, Proceedings, Cleveland, Ohio, USA, 28-30 July, 1999. IEEE Computer Society, 214225

[25] [Smacq02] Fisk, M. and Varghese, G. 2002. *Agile and Scalable Analysis of Network Events*, In Proceedings ACM SIGCOMM Internet Measurement Workshop, Marseille, France, 2002.

[26] [Stockinger00] Stockinger, K., Duellmann, D., Hoschek, W., and Schikuta, E. 2000. Improving the performance of high-energy physics analysis through bitmap indices. In 11th International Conference on Database and Expert Systems Applications DEXA 2000, London, Greenwich, UK.

[27] [Stockinger05] K. Stockinger, J. Shalf, K. Wu, E. W. Bethel, “Query-Driven Visualization of Large Data Sets,” in *Proceedings of IEEE Visualization 2005*, pp 167-174, Minneapolis MN USA, October 2005.

[28] [Stockinger2005] K. Stockinger, J. Shalf, K. Wu, E. W. Bethel, “Query Driven Visualization of Large Data Sets,” *Proceedings of IEEE Visualization 2005*, pp 167-174, 2005. (Conference proceedings).

[29] [Stockinger2005b] K. Stockinger, K. Wu, R. Brun, P. Canal, “Bitmap Indices for Fast End-User Physics Analysis in ROOT,” to appear in Nuclear Instruments and Methods in Physics Research, Section A – Accelerators, Spectrometers, Detectors and Associated Equipment, Elsevier.

[30] [Thomas2005] J. J. Thomas and K. A. Cook, *eds., Illuminating the Path – The Rsearch and Development Agenda for Visual Analytics*. IEEE Computer Society Press, Los Alamitos, Califor.ia, 2005

[31] [Uphoff04] B. Uphoff and P. Criscuolo, “A Framework for Collection and Management of Intrusion Detection Data Sets,” in *Proceedings of the 16th Annual FIRST Conference on Computer Security and Incident Handling*, Budapest Hungary, 2004.

[32] [Wu01]Wu, K., Otoo, E. J., and Shoshani, A. 2001. A performance comparison of bitmap indexes. In Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001. ACM, 559561.

[33] [Wu04] Kesheng Wu, Ekow Otoo, and Arie Shoshani, 2004. On the Performance of Bitmap Indices for High Cardinality Attributes. *In Proc. VLDB* 2004, pages 24 – 35.

First A. Author Biographies should be limited to one paragraph consisting of the following: sequentially ordered list of degrees, including years achieved; sequentially ordered places of employ concluding with current employment; association with any official journals or conferences; major professional and/or academic achievements, i.e., best paper awards, research grants, etc.; any publication information (number of papers and titles of books published); current research interests; association with any professional associations.

Second B. Author Jr. biography appears here. Degrees achieved followed by current employment are listed, plus any major academic achievements.

Third C. Author is a member of the IEEE and the IEEE Computer Society.