Accelerating Network Traffic Analytics Using Query-Driven Visualization

E. Wes Bethel*
Computational Research Division

Scott Campbell
National Energy Research Scientific Computing Center Division

Eli Dart
Energy Sciences Network

Kurt Stockinger
Computational Research Division

Kesheng Wu
Computational Research Division

Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

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. This paper focuses on an interdisciplinary combination of scientific data management and visualization/analysis technologies targeted at reducing the time required for data filtering, querying, hypothesis testing and knowledge discovery in the domain of network connection data analysis. We show that use of compressed bitmap indexing can quickly answer queries in an interactive visual data analysis application, and compare its performance with two alternatives for serial and parallel filtering/querying on 2.5 billion records' worth of network connection data collected over a period of 42 weeks. Our approach to visual network connection data exploration centers on two primary factors: interactive ad-hoc and multiresolution query formulation and execution over n dimensions and visual display of the $n-$dimensional histogram results. This combination is applied in a case study to detect a distributed network scan and to then identify the set of remote hosts participating in the attack. 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.

CR Categories: 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]

Keywords: query-driven visualization, network security, data mining, visual analytics

1 Introduction

Visual Analytics is defined in [40] 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 to [40], 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.

The challenges in network traffic analysis are motivated by a combination of rapid growth in the internet combined with the time-critical nature of responding to problems that arise. According to Burrescia [4], 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. Presently, a typical day's worth of traffic at an "average" government research laboratory comprises tens of millions of connections resulting in multiple gigabytes' worth of connection records. A year's worth of such data currently requires on the order of tens of terabytes or storage.

A broad view of the network traffic analysis problem includes data collection, storage and management, feature detection, event characterization, analytical discourse to understand features and discover their relationships, and timely response to a particular incident. The work we present here explores a subset of the complete network traffic analysis problem. Namely, we focus on a multidisciplinary approach to feature mining and hypothesis testing by using high performance index/query technology to accelerate discovery of "interesting data." Nearly all previous works in filtering/searching network traffic analysis have complexity that is linear with respect to the size of the dataset. In contrast, the work we present here has sublinear performance complexity. This performance characteristic makes it possible to perform interactive analysis of very large datasets.

The complete "analytics duty cycle" includes time from several different components that comprise a cyclic process: data access, data mining, visualization, analyst interpretation, theory refinement and formulation. The objective of our work is to quantifiably reduce the time required for a subset of the complete analytics duty cycle, namely the time for data access, data mining and visualization. The other elements of the duty cycle – interpretation, theory refinement and formulation – are subjective and beyond the scope of this paper.

The main contributions of this paper are: (1) objective measurement of how using state-of-the-art index/query technology can reduce the filtering or data mining portion of the analytics duty cycle; (2) an approach to knowledge discovery that relies on multiresolution queries and statistics from user-defined multidimensional range queries presented as histograms to perform interactive analysis; (3) application of this combination to a network data analysis problem of a realistic size.

The rest of this paper is structured as follows. Section 2 presents a survey of previous work in fields related to the topic of this paper. Section 3 presents an experiment profiling the performance of different technologies for performing queries on a realistic-sized collection of network traffic data. The main point of this experiment is to compare the vastly different performance capabilities of different technologies. Section 4 contains a network traffic analysis case study where the confluence of scientific data management and visualization comprise a visual analytics implementation. Our case study focuses on first identifying the presence of a network scan through multiresolution exploration of a large network connection dataset, then identifies the remote hosts participating in the distributed scan.

*e-mail: ewbethel@lbl.gov, scampbell@lbl.gov, dart@es.net, kstockinger@lbl.gov, kwu@lbl.gov

2 Background and Related Work

As our presentation here represents an interdisciplinary approach to large-scale network connection data analysis, we give an overview of previous work in several different areas: network traffic analysis, network traffic visualization, query-driven visualization and analysis, and indexing and querying.

2.1 Network Connection Record Analysis

A network connection is defined as a set of packets passing between two hosts within a given time interval that have common characteristics. An example is a single communication session or an interaction between two hosts on the Internet. Several standard tools exist for capturing network connection data. For larger environments, routers and switches can provide connection data in specialized formats such as NetFlow [39] or SFlow [31]. In this study, we use Bro IDS [30] connection records rather than traditional flow records. Bro's connection records contain the same information as traditional flow records: source and destination IP address, source and destination port, byte and packet count by source and destination, connection start and end time, TCP state, and so forth. Another factor influencing their use here is the fact we have access to multiple years' worth of such data.

For network connection data analysis, the special purpose systems and software provided by network equipment vendors for analyzing NetFlow and SFlow records are typically optimized for aggregate network usage and trend analysis rather than forensic analysis. Freely available tools are typically architected as a collection of command-line utilities for collecting, concatenating, filtering, and summarizing network connection data. Fullmer's OSU Flow-Tools [9] is a good example of such a collection that is best suited for small-scale analysis. Plonka's FlowScan suite [32] consists of a collection of perl scripts and modules that provide stateful flow inspection and charting/graphing capabilities. Both these store data in flat files in uncompressed (Flow-Tools) or compressed (FlowScan) binary formats and use a sequential scan through data for filtering operations. Navarro [25] uses a relational database for storing and filtering collections of network connection records via a SQL query interface. Gates' SiLK Suite [10] is a collection of software utilities for flow data collection and analysis that uses a reduced-size flow record in conjunction with data compression and file/directory hierarchies to maximize filtering performance and minimize data footprint size on disk. SiLK uses bitmaps to accelerate filtering operations based upon IP address, but does not use bitmap indexing or any other form of indexing structure, other than directory and file hierarchy, to accelerate query operations.

Our work differs from these previous network data analysis efforts in the following ways. First, our approach uses compressed bitmap indexing to accelerate filtering operations. While Navarro [25] uses a relational database, such an approach is known to be significantly slower for use in large-scale data management and analysis that one using compressed bitmap indices [43]. Second, one of the main messages of this paper is on the value of using efficient indexing technology to accelerate queries. In contrast, many previous works are describing collections of software that comprise a network connection analysis solution suite. The benefits of the approach we describe here would be applicable to most of these previous works as well as future projects whose concern is large-scale network data management and analysis.

2.2 Network Traffic Visualization

Previous work in this space spans several different dimensions, including visualization of IDS alerts, traffic volume and statistics, traffic patterns, network topology and network connections.

In terms of visualizing network traffic levels and statistics, one of the most widely deployed tools is MRTG [26], which relies on RRDtool [27] for charting and graphing. These tools are in use at nearly all sites that run production networks for the purpose of showing traffic levels at multiple temporal resolutions. One shortcoming of RRDtool is that detail in data temporally distant from the present is lost due to its being summarized and averaged in the round-robin database.

More recently, [22, 24, 21] describe applications that map network connection variables to axes, then present activity, or lack thereof, at the appropriate grid location (a form of histogram). The basic idea is to facilitate rapid visual discovery of features, patterns or activity in network traffic or network connection data. Other applications use a linked node concept to convey the presence of traffic or traffic levels between addresses mapped to a grid [24, 44, 11], to geographic locations or some other representation of network topology [7, 18], or to a task-oriented metaphorical representation of internal and external networks [8, 23, 11]. Several other applications focus on visualization of IDS alerts [17, 23]. Within this constellation of previous visualization research, the work we describe here focuses on the interactive drill-down to first compute then display network connection statistics using histograms that show levels of activity satisfying user-defined query conditions to implement knowledge discovery and hypothesis testing on large collections of network connection data.

2.3 Query-Driven Visualization and Analysis

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." The basic premise is to restrict computational and cognitive load either limiting or prioritizing processing and interpretation to features of interest. This approach is consistent with the needs of many scientific users who need capabilities to help them find and focus on features hidden in large, multidimensional data [12].

Within the context of network traffic visualization and analysis, many previous works include some notion of "data filtering" as part of their knowledge discovery process. The OSU Flow-Tools [9], FlowScan [32] and SiLK [10] all have command-line interfaces to filtering utilities that find records matching a set of criteria. Filtering selection criteria include connection record variable values like source or destination IP or port number, IP protocol, interface number, and autonomous system number. For most of these systems, filtering is performed using a sequential scan through flat files, and the computational complexity of such an approach is $O(n)$ where n is the number of connection records.

Swift-3D [18], an integrated data visualization and exploration system, provides for query expressions using a C-style expression that are translated into C code, compiled into shared objects and dynamically linked into the application. This approach offers flexibility similar to that of *awk* and *sed* but with the speed of compiled code. This particular work achieves high performance by using Direct-IO to bypass the kernel, but does not appear to use any form of indexing to accelerate the search process. This approach is also $O(n)$ in computational complexity with respect to data size.

Other network analysis and visualization applications use the notion of filtering to implement a form of query-driven visualization and analysis. [24] uses interactive filter manipulation to reduce the amount of TCP activity on a per-port basis that is processed and displayed. VisFlowConnect [44] provides inclusive or exclusive port filtering, filtering based upon IP protocol, transfer rate, or packet size. [17] applies data range filtering to IDS alert output within the context of visual analysis. [23] employs what can be thought of as visual filtering by using LOD-based techniques to simplify the visual display of less relevant IDS alert information. The NVisionIP system [20] allows an analyst to use an iterative process of first visually exploring data (activity), transforming the visual pattern into a symbolic rule, then searching the data sources for that pattern using the symbolic rule. In effect, this is the same type of approach we use here.

The idea of query-driven visualization has had an impact in many other areas outside of network traffic analysis. The VisDB system combines a guided query-formulation facility with relevance-based

visualization [15]. Data items are ranked in terms of relevance to a query, and the top quartile of most relevant results are then input to a visualization and rendering pipeline. This approach results in $O(n)$ complexity. The TimeFinder system [13] 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 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 and its filtering algorithms appear to be sequential in nature – also $O(n)$ complexity.

Our work differs from previous efforts in the following ways. First, our application supports the formulation and execution of queries over any number of connection record variables: an multidimensional query produces a multidimensional histogram. Second, our application uses histogram display for query output and interaction for query formulation. Complex compound queries may be built up using a combination of selection boxes on histograms (similar to TimeFinder), or via a query string for less regular query expressions. Third, queries are multiresolution in the sense that the user specifies the granularity of the resulting histogram. This approach provides the ability to quickly generate query and visual results at a large temporal granularity (context), then target subsequent queries and display at finer temporal or other resolution (focus). Fourth, previous works in network traffic analysis and visualization are either silent on the size of dataset they study or indicate a data set size that is "relatively small." The term "relatively small" means traffic or summary information ranging from a few minutes to a few weeks in time. Our work shows performance and analysis results on 42 weeks' worth of network connection data containing 2.5 billion connection records.

The idea of applying compressed bitmap indexing accelerate the visualization process was previously described in [37]. The performance of compressed bitmap indexing was compared to that of search-accelerating algorithms in isocontouring. We are extending the work of [37] in this paper by: applying indexing and querying techniques to network connection data analytics; providing a performance comparison between several different filtering, or data mining technologies; capitalizing upon statistical information in the data management infrastructure to create easy-to-comprehend visual displays; combining all technologies into an integrated framework for rapid knowledge discovery; demonstrating the combination's applicability on a realistic-sized network connection data analysis problem.

2.4 Indexing and Querying

As has been discussed in previous related work, many network traffic analysis applications provide the ability to locate records that match a set of user-defined criteria: IP number of source or destination, packet arrival rate, etc. The sequential scan approach used in nearly all related previous works has computational complexity of $O(n)$, where n is the number of connection records or packets. Algorithms with such performance do not exhibit desirable scaling performance on realistic-sized problems where query response time is a concern. As with previous work, our approach is to implement compound ad-hoc range queries of the form "StartTime > 20050501 AND 10.102.0.0 <= SourceIP <= 10.105.255.255."

Accelerating searches through the use of an indexing structure has been well-studied in the field of computer science [16]. One tradeoff in this area is balancing update and query performance: when a record is updated, the indexing structure must also be updated. In our application, existing data records are never modified: ours is an append-only operation since we never change existing network connection records. Therefore, we look to solutions that optimize query response to reduce search time.

Roughly speaking, algorithms for index/query can be decomposed into two broad classes: tree-based and non-tree-based. Most tree-based indexing methods suffer from what is known as "the curse of dimensionality" in which adding more dimensions (or, more generally, variables or attributes that are indexed and searchable) results in an exponential growth in storage and processing requirements [1]. Tree-based multi-dimensional indexing structures, like the kd-tree [2], exhibit exponential growth in complexity and storage with increasing dimensionality. Such characteristics are not desirable given our objective of achieving sublinear performance for queries on large, multidimensional datasets.

Alternatives that don't suffer from exponential complexity with increasing dimensionality include bitmap indexing (BI) and the projection index. BI, which is a part of many commercial database systems, is efficient for low cardinality attributes [28] and high cardinality attributes through Word Aligned Hybrid coding (WAH) [43]. In a 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. Floating-point data may be indexed and searched in BI via one of several different types of binning strategies that place an artificial upper bound on attribute cardinality. There are a number of strategies for reducing the size of a bitmap index, including binning [35, 36], multi-component encoding [5], and compression [14, 42]. 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; its worst-case computational search complexity is proportional to the number of items matching the search criteria, the theoretical optimum [43].

The projection index [29] extracts the attribute values and stores them separately so that only those attributes relevant to a query are loaded into memory. The results of queries over individual attributes are intersected to produce a final result that matches all attribute conditions. The projection index forms the basis of ROOT [3], which is a software system for the management and analysis of large collections of high energy physics data. ROOT is in use today at several large facilities that routinely collect and analyze terabytes of detector event data, and is widely accepted as the "gold standard" for data management and analysis in the high energy physics community.

Another indexing approach, called the Vector Approximation file (VA-File) [41], is designed for similarity searches in high-dimensional vector spaces. Each indexed attribute is approximated by a set of feature vectors represented as a bit string of a predefined length. Since the VA-File does not partition the vector space with a tree-like data structure, it does not suffer from the curse of dimensionality. The VA-File is particularly efficient for processing multi-dimensional queries that cover all indexed attributes. However, for multi-dimensional queries that cover only a subset of the indexed attributes (partial similarity queries), there is additional overhead involved since the entire VA-File must be read into memory. The partial VA-File [19] overcomes this overhead by storing each set of vectors per indexed attribute in a separate file as opposed to storing all sets of feature vectors in a single file. Thus, the partial VA-File performs better than the VA-File for partial similarity queries. However, for multi-dimensional queries that cover all indexed attributes, the VA-File outperforms the partial VA-File. In essence, the (partial) VA-File is very similar to a certain type of bitmap index. However, a direct comparison of the partial VA-file with bitmap indices is still an open research problem.

For our append-only data collection, BI's performance characteristics for update and query response make it a good choice for our application. Later in Section 3, we compare the performance of

BI with that of ROOT's projection index implementation for locating interesting network connection records based upon user-defined ad-hoc queries.

3 Query-Driven Network Traffic Analysis Performance Study

In this section, we measure the performance of three different technologies for discovering "interesting data" in a large network connection dataset. We begin with three background sections. The first describes the network traffic data we use in the performance measurements as well as in Section 4's case study. The second identifies the experiment's computing environment. The third focuses on each of the three competing implementations. The performance study consists of two sets of tests – serial and parallel – for each of the three competing technologies.

3.1 Network Traffic Data

The dataset we use in our performance experiments and case study consists of network connection data from over a 42-week period consisting of 2.5 billion records. The data source is a Bro system running at the National Energy Research Scientific Computing Center (NERSC). While Bro is best known for its role as a security device, it can also produce network connection records. Each record contains the "standard" set of network connection variables: IP addresses, ports, duration, rate, TCP state, and so forth. We save all variables of each record in flat files using an uncompressed binary format for a total size of about 281GB. To increase query efficiency, we have split IP addresses into four octets A, B, C and D. For instance, IPS_A refers to the class A octet of the source IP address. The bitmap indices themselves require a total of about 78.6 GB of space. We created a standalone utility that generates the indices: it uses FastBit [34] to generate compressed bitmap indices from the connection data. While these indices are about one-third the size of the original data, tree-based alternatives (e.g., B-trees) typically require about three times the space of the original data.

3.2 Computing Environment

We use a single computing platform for both the performance experiments and the network traffic analysis case study. The platform is an SGI Altix comprised of 32 IA64 1.4Ghz processors, 192GB of RAM and 40TB of fiberchannel RAID capable of delivering 500GB/s in sustained I/O performance. We chose this platform due to its combination of vast amounts of memory and its high-performance I/O to secondary storage. Such platforms are well-suited to data intensive analysis and visualization tasks.

3.3 Query Implementation

For the purposes of measuring and comparing query response time, we use three different approaches. The first is representative of the type of technology typically used in production networks for traffic and security analysis. The second is projection indices from the ROOT implementation, which is in widespread use for data management and analysis of large-scale high-energy physics data. The third is the FastBit implementation of compressed bitmap indexing.

The first approach is a series of scripts and shell-based tools (awk, grep, etc.) that parse and search through an ASCII version of the connection records. At first glance to one outside the network security business, this approach may seem to be naïve. In fact, this approach is widely used in network traffic analysis to overcome limitations caused by proprietary data formats and the frequent need to perform different types of ad-hoc analysis. Shell scripts and tools are easy to create and change, readily shareable, and transportable across platforms. They are relatively slow compared to compiled and optimized applications. This approach has has $O(n)$ complexity – all data records must be examined in the search for those that meet a given set of criteria. Historically, network analysts typically work with relatively small collections of data – hours' or days' worth of traffic. For those small data sizes, script-based tools typically execute with a duty cycle on the order of 10s or 100s of seconds. Since we are tackling a much larger problem in this paper – 42 weeks' worth of network traffic – there is value in comparing performance with commonly used methods.

Both the second and third query types are accelerated with indexing. The second type uses ROOT's implementation of projection indices. For the third type, we extended ROOT so that it uses FastBit's compressed bitmap indexing. To support this type of dual-mode use, we created two separate versions of the network traffic data – one organized specifically into ROOT's projection index format, and one that uses FastBit's native data storage format.

3.3.1 Serial Range Query Performance

To establish a performance baseline, we measure the time required to answer the following three-dimensional network traffic analysis query (i.e., a query comprised of three variables): "select IPS_B, IPS_C, IPS_D where $IBS_B < 100$ AND $IPS_C < 100$ AND $IPS_D = 128$." This query locates those network connections originating from a given range of IP addresses, and is not geared towards any particular network analysis scenario. We perform this query on 42 weeks' worth of network data.

The time required to answer this query using "typical network traffic analysis software" is approximately 51,000 seconds. [1] With ROOT's projection indices, the time required to answer the query is 1269 seconds. With ROOT's FastBit implementation, the time required to answer the query is 419 seconds. This factor of three performance improvement is consistent with previously published results comparing multidimensional query performance on large collections of high energy physics data [38].

3.3.2 Parallel Exact Match Query Performance

Like many other endeavors in high performance computing, data intensive operations stand to benefit from parallelism. We implemented a battery of tests to measure the performance of the three competing technologies when run in a parallel configuration. We ran tests using 1, 2, 6, 13 and 21 processors – these levels of parallelism were chosen so that each processor is responsible for an equal number of weeks' worth of network traffic data. Since there is variation in the amount of connection data from week to week, a by-week domain decomposition does not ensure even load balance in terms of computation or I/O. For these tests, we used a variant of a query that appears in Section 4: find all records where (DP==5554) and (STATE==1) and (IPS_A==220) and (IPS_B==184) and (IPS_C==26). This particular query is one of several that comprise the analysis Case Study in Section 4.

The results are summarized in Table 1. The first column shows the number of processing elements (PEs). The second column shows the query response time for evaluating the 5-dimensional query with the shell-based approach. Columns three and four show the performance results of ROOT/Projection Index and ROOT/FastBit. The uniprocessor ROOT/Projection Index time is comparable here to that of Section 3.3.1 – this approach has $O(n)$ complexity. The ROOT/FastBit approach uses much less time to answer this query than the one in Section 3.3.1 for two reasons: (1) because the new query selects many fewer records, and (2) the bitmap indices we use in these tests are better suited for equality than range queries. The shell-based performance decreases compared to that shown in Section 3.3.1 due to increased *awk* processing logic. None of these three approaches shows optimal scalability due to by-week load imbalance.

In earlier work, Gates [**?**] reports that precompiled binaries operating on binary data exhibit about a three-fold speed increase compared to shell-based scripts that rely on UNIX utilities for manipulating data. We would therefore expect about the same level of improvement to our shell-based script performance data here if we

[1] Our "shell scripts" consist of a mixture of *awk* and *grep* run in a Bourne shell script. Whether processing time would be substantially reduced by using "perl" is an open question, but outside the scope of this paper.

were to use on of FlowScan or Flow-Tools. Even with that level of improvement, these tools, which have no indexing capabilities, would still be one to two orders of magnitude slower that ROOT's projection index implementation.

PEs	Shell-based	ROOT/Projection Index	ROOT/FastBit
1	156381.14	1357.07	5.36
2	71835.32	600.05	3.72
6	21952.12	214.14	2.66
13	9389.96	113.88	2.58
21	2237.53	98.95	2.05

Table 1: Parallel performance evaluation of a 5-dimensional query over 42 weeks' worth of network connection data: Find all records where (DP==5554) and (STATE==1) and (IPS_A==220) and (IPS_B==184) and (IPS_C==26). The time is reported in seconds. These results highlight the performance gains that result when reducing computational complexity from linear to sublinear with respect to the data size. Timings for shell-based tools are extrapolated from a battery of parallel runs on 30-day collections to the full 42 weeks. These results indicate that use of highly efficient index and query technology can dramatically reduce the search/filtering phase of the analytics duty cycle.

4 Network Traffic Analysis Case Study

In this section, we present a case study illustrating how the combination of fast search/query operations combined with interactive analytics and visualization gives rise to a highly practical and efficient methodology for network traffic analysis. For this case study, we use the same 42 weeks' worth of network data and computing platform as in Section 3. The case study shows how an iterative process of visual inspection and data mining leads to the discovery of a distributed scan.

A distributed scan is a specialized form of a "port scanning" attack in which multiple distributed hosts systematically probe for vulnerabilities on a set of hosts. A specialized form of distributed scanning involves attacking hosts that have themselves been compromised and conscripted for use in a distributed scan attack by a third party. The third party acts as a "central authority" for managing the attack. This form of distributed scan is known as a "bot-net" attack.

4.1 Visual Analytics Application Overview

To achieve the results we present here, we constructed a custom interactive visual analytics and data mining application and used it to discover a distributed scan. The application makes calls to FastBit [34] to perform network traffic data I/O and queries. FastBit provides the ability to return the number of items that would satisfy a query, and our application leverages this capability to rapidly construct and display histograms. These histograms, combined with visualization and interaction, provide the basis of our visual analytics application. The visualization and rendering portion of our application uses OpenRM Scene Graph [6] and the GUI is built using FLTK [33].

The application's use pattern is as follows. First, the user loads a FastBit metafile containing information about the number of variables and indices like data type, min/max values, number of bins per variable, and so forth. Next, the user selects any of the variables for display. The application then asks FastBit for a histogram indicating the number of connection records in the bitmap index bins. The application provides the means for forming and posing several different types of queries. The results of a query, which consist of another histogram, then appear as a new variable in the application's list of variables and can then be visualized using one of several different dimension-appropriate visualization techniques. A query is formulated by the "cross-product" of range selections that may span an arbitrary number of connection record variables. This multivariate cross-product query may be further qualified with an arbitrary conditional string.

The application provides three different mechanisms for specifying query range selections. The first is an interactive selection widget for specifying a contiguous range of histogram bins with a selection box that is similar to Hochheiser's "timeboxes" [13]. This mechanism may be applied to any of the network connection variables to produce an $n-$dimensional query. The second mechanism consists of a data starting value, an ending value and a step size. The application will automatically generate histogram bins that are evenly spaced over the user's range selection specified by the typein. The third is a typein for specifying a set of individual histogram bins. This set is simply a list of integer values – bins may be specified in any order, including disjoint, descending, random, etc. The application's query engine combines parameters from each of the three different query specification mechanisms to form an ordered $n-$dimensional query. The field resulting from such a query is an $n-$dimensional dataset of V vector elements. The n axes correspond to range selection from each of the n source variables. The V vector elements are formed by V user-specified histogram bins. Such a formulation of spatial axes and vector components was convenient for this particular case study.

4.2 Network Traffic Analysis

The starting point for our case study presumes an IDS alert has indicated a large number of scanning attempts on TCP port 5554 (Sasser type worm). That alert may have come from an actual IDS system, or a phone call or email from a colleague. Given the alert, our objective is to examine a 42-week collection of data to determine whether this activity has been occurring and if so, to better understand its characteristics. Since it is not practical to visually analyze 42 weeks' worth of data at one-second resolution, we use an interactive, multiresolution approach to support context and focus changes. This iterative approach, which consists of an analyst posing different filtering criteria to examine data at different temporal scales and resolutions, relies on very fast filtering and query machinery to achieve interactivity for such a large dataset.

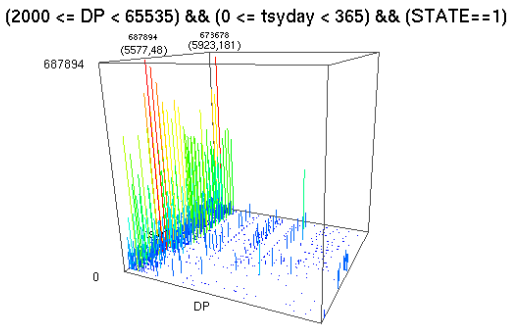

Figure 1: This histogram shows the number of unsuccessful connection attempts over a 42-week period on ports 2000 through 65535, with radiation excluded. One axis is destination port, the other is time at daily resolution. These results confirm a high level of activity on port 5554 on day 48 (the Seventh Week).

First, we will search the 42 weeks' worth of data to identify those ports on which there is a large number of unsuccessful connection attempts. Figure 1 shows a histogram where one axis is destination port number and the other is time. Each "bin" in the time axis represents one week's worth of activity. The height of each bar represents the number of unsuccessful connection attempts during a particular week on a particular destination port across all addresses within the destination network. We conclude from this stage of the analysis that there is a high degree of suspicious activity in Week Seven on destination port (DP) number 5554. The ability to quickly look at a weekly summary allows us to focus subsequent investigation within a narrower temporal range.

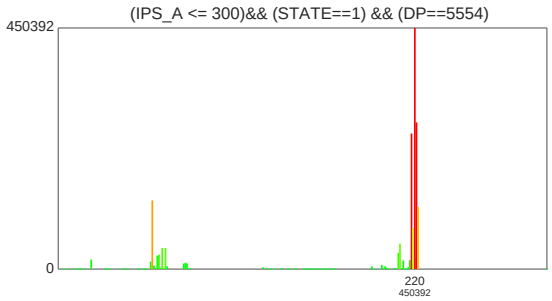

Figure 2: This image shows a histogram of unsuccessful connection attempts during Week Seven to port 5554. The horizontal axis corresponds to the A octet of the source address. The bin with the largest number of counts is indicated as "220," which means the source of the suspicious activity is from a host or hosts having an IP address where "220" is the first address octet. The bars are colorized such that red bars lie three standard deviations or greater from the mean, which in this case is a value close to zero.

The next step is to determine the addresses of the hosts from which the unsuccessful connection attempts originate. We will use an iterative approach where we first identify the A octet addresses, followed by the B, C and D octet addresses. We pose a query asking for the number of unsuccessful connection attempts to port 5554 during Week Seven over each address within the Class A octet. The resulting histogram is shown in Figure 2. Each bin in the horizontal axis is one address from the source Class A octet. The height of each bar indicates the number of unsuccessful connection attempts from that Class A address. The largest such spike occurs at 220, which means that most of the suspicious activity is originating at a host or hosts within a range of IP addresses of 220 in the A octet. In this study, the analyst chooses to focus subsequent queries on hosts having an A address octet of 220.

To aid in quickly and positively identifying anomalies, our application provides a toggle on each variable's display panel to highlight the "top N" items in a histogram. In addition, it provides a statistically-based transfer function so that histogram bars are color-coded by the relationship between a bar's variance and the standard deviation of all histogram counts. Bars that are colored red lie about three standard deviations above the mean histogram count.

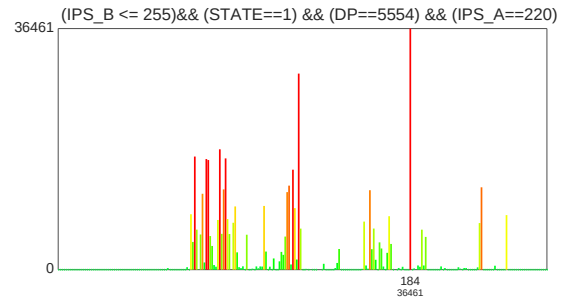

Figure 3: Here we see that from within the 220 Class A octet, the Class B address with the most suspicious traffic is 184. Several potential class B candidates emerge – colored red – and we choose 184 for further investigation. The horizontal axis is the 254 possible Class B addresses, and the vertical axis is the count of unsuccessful connection attempts.

We next refine our search to identify the B address octets where suspicious activity originates. To do so, we refine our previous query to include only those hosts having an A octet address of 220 along with the previous query conditions. The result is the histogram in Figure 3. The histogram spike showing the most activity occurs at $IPS_B = 184$. While there are several other interesting possibilities here, like the one at about $IPS_B = 128$, the analyst here chooses to focus subsequent inquiry on the 184 B address octet. Using the results from Figures 2 and 3, subsequent inquiries will focus on connection records from hosts within the 220.184 block of IP addresses.

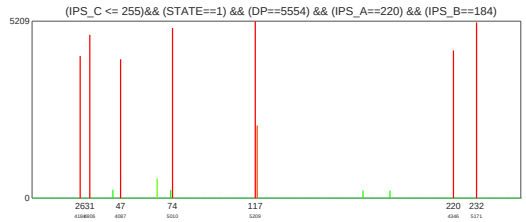

Figure 4: This histogram shows levels of suspicious activity on port 5554 during Week Seven across C address octets from the 220.184 IP address block. We will use top seven C addresses for subsequent inquiry – these show "similar" levels of activity.

The next step is to discover the source host address range within the Class C octet. Figure 4 shows there are unsuccessful connection attempts from hosts in several different addresses in the Class C octet having similar levels of traffic. We hypothesize that hosts on these seven Class C network segments are part of a distributed scan.

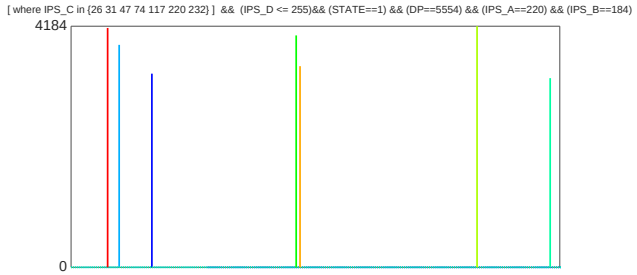

Figure 5: This histogram shows levels of suspicious activity across D address octets to port 5554 during Week Seven from a number of C addresses within the 220.184 IP address block. The spikes are color-coded by their associated Class C address. The absence of duplicate-colored spikes indicates only one D address, or host, per C octet address is participating in this particular attack. This color-coding scheme is also used in Figures 6 and 7.

We take this idea one step further to discover the source addresses within the D octet. Unlike previous queries, each of which is a straightforward compound range query, we must take a slightly different approach to find the D octet addresses. Our application lets us specify a discrete set of variable values to include as part of a query – in this case, these discrete values are the seven unique Class C addresses seen in Figure 4. The next query is comprised of seven different sub-queries – one for each of the seven unique Class C source addresses. The result is Figure 5, which indicates traffic from each of the Class D IP addresses corresponds to a single Class C address. At this point, we have positively identified the IP addresses of all the hosts from which the suspicious traffic originates.

The next step is to look at the problem from a different direction. Rather than focus on identifying the source host addresses – which we have now identified – we are interested in discovering their access patterns through destination IP addresses. Looking first at the destination Class C octet, we see in Figure 6 that each of the seven participating hosts is sending traffic to about 21 or 22 contiguous class C addresses.

Extending the idea of Figure 6, we we formulate a higher-dimensional query that produces the histogram shown in Figure 7. That visualization uses a 3D scatterplot to confirm the expectation

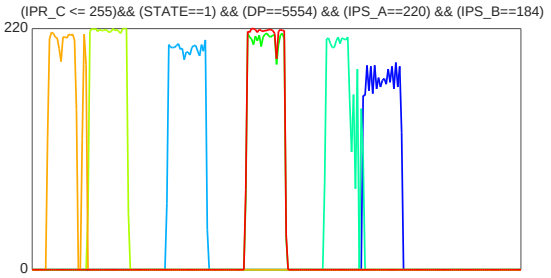

Figure 6: This histogram shows the set of Class C destination addresses being scanned by each of the source hosts. Note the overlap in one of the destination address ranges. There are seven different colored plots in this image; they colored by source host IP number. Two remote hosts are attempting to scan the same block of destination class C addresses as evidenced by the "overlay" of red and green plots. The horizontal axis is destination C octet addresses, and the vertical axis is the number of unsuccessful connection attempts.

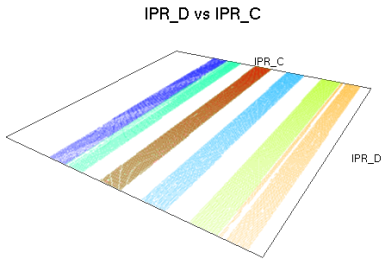

Figure 7: This image shows the coverage in destination IP space by each of the hosts participating in the distributed scan. The axes are the destination C and D address octets. We represent an unsuccessful connection attempt at each (C,D) destination address with a transparent point, and color-code the point based upon the source host address. Two of the remote hosts are scanning the identical block of (C,D) addresses. This feature appears as an overlap in the red-green group in the center of the image.

that attacking hosts are attempting to have uniform coverage in the destination address space. As with Figure 6, Figure 7 color-codes each point based upon the IP address of the attacking host. We use transparent point primitives in 3D for this visualization. Looking closely at the "red stripe," it is possible to see the overlay of red points over green points, confirming the behavior we saw previously in Figure 6. Figure 7 shows that each attacking host's scan pattern is (1) to "march through" all Class D addresses for each Class C address, and (2) that each has been assigned a contiguous group of Class C addresses to scan.

4.3 Discussion

Two of the authors on this paper are network security experts whose job duties include operation of production networking facilities. They both contributed to the design and engineering of this work, and both are in a good position to evaluate its effectiveness. Their comments, paraphrased below, are particularly insightful.

Because our visual analytics application processes and visualizes statistical information about network traffic data – rather than actual network traffic data – it affords a certain amount of insulation from sensitive information. This approach will allow more people access to network data than otherwise might be possible due to data sensitivity and data size.

The application's general design principles result in a system that is simple to use and easy to understand. The visualizations are very straightforward and require no complex mental mapping to understand. The simple yet effective user interface and interaction design means that this type of interactive analytics very accessible since the learning curve is very shallow and only a passing knowledge of network traffic data is needed to interpret the results. Their analysis decisions are, of course, based upon decades' worth of collective experience.

Both network experts felt this approach was a very useful method for determining the components of a distributed scan. Both were eager to apply the technology to other types of forensic network security projects. Both were excited by the extremely short duty cycle in the data mining process. Neither had ever had the opportunity to explore a single collection of network traffic data of such a large size.

During the course of this case study, we identified several features that would be nice to have in this particular application. One is the ability to more easily exclude specific items (bins) from a histogram cross-product. This feature would make it easier to eliminate "radiation noise" from the display (i.e., this term refers to known scanning traffic). Another is the ability to compute and display statistical information from the data distributions. A third is the ability to filter based upon temporally-based statistics.

5 Conclusion

In an information-dominated age, the ability to quickly and accurately understand data makes the difference between success or failure in science, business, medicine and education. The work we have presented in this paper takes direct aim at reducing the searching and filtering portion of the analytics duty cycle. Our approach blends technologies from data management, visualization, analysis and interactive discourse. Our network traffic analysis case study highlights how such a combination provides new capabilities enabling rapid detection and analysis of a distributed network scan.

To reduce the searching and filtering portion of the analytics duty cycle in network connection data analysis, we have leveraged several different concepts. First, we accelerate filtering by using compressed bitmap indices. Second, our visualization techniques are centered about the idea of generating and displaying statistical data (histograms) that is readily available from the compressed bitmap indexing infrastructure. Such an approach supports rapid context/focus analysis, which has proven crucial in our network connection data analysis case study. Third, complex multiresolution and multidimensional queries are automatically constructed through "histogram cross-products," which has proven to be a highly effective visual analytics technique for data mining. We demonstrated these concepts on a realistic-sized dataset consisting of about 2.5 billion records of network connection data collected over a 42-week period.

Acknowledgement

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors wish to acknowledge the contributions by Steven A. Smith, Los Alamos National Laboratory; Brian Tierney and Jason Lee, Lawrence Berkeley National Laboratory to a LBNL Technical Report LBNL-59166, which is a predecessor of the work reported in this paper. The authors wish to particularly thank Smith for a suggestion that evolved into the inspiration for using FastBit's histogram bins as the source of visual analytics in performing guided query formulation.

References

[1] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[2] John Bentley. Multidimensional Binary Search Trees Used for Associative Search. *Communications of the ACM*, 18(9):509–516, 1975.

[3] Rene Brun and Fons Rademarkers. ROOT – An Object Oriented Data Analysis Framework. In *Proceedings of the AIHENP 1996 Workshop*, pages 81–86, 1997.

[4] Joe Burrescia and William Johnston. ESnet Status Update. Internet2 International Meeting, 2005.

[5] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *Proceedings of the 1998 ACM SIGMOD: International Conference on Management of Data*, pages 355–366, New York, NY, USA, 1998. ACM press.

[6] R3vis Corporation. OpenRM Scene Graph. http://www.openrm.org, 1999-2006.

[7] Kenneth C. Cox, Stephen G. Eick, and Taosong He. 3D Geographic Network Displays. *SIGMOD Rec.*, 25(4):50–54, 1996.

[8] Mike Fisk, Steven A. Smith, Paul Weber, Satyam Kothapally, and Thomas Caudell. Immersive Network Monitoring. In *Proceedings of the 2003 Passive and Active Measurement Workshop*, April 2003.

[9] Mark Fullmer and Steve Romig. The OSU Flow-tools package and Cisco Netflow Logs. In *Proceedings of the 14th Systems Administrator Conference (LISA 2000)*, pages 291–303, 2000.

[10] Carrie Gates, Michael Collins, Michael Duggan, Andrew Kompanek, and Mark Thomas. More NetFlow Tools: For Performance and Security. In *Proceedings of the USENIX18th Systems Administration Conference (LISA 2004)*, pages 121–131, November 2004.

[11] John Goodall, Wayne Lutters, Penny Rheingans, and Anita Komlodi. Preserving the Big Picture: Visual Network Traffic Analysis with TNV. In *Proceedings of the 2005 Workshop on Visualization for Computer Security*, pages 47–54, October 2005.

[12] Bernd Hamann, E. Wes Bethel, Horst Simon, and Juan Meza. The NERSC Visualization Greenbook: Future Visualization Needs of the DOE Computational Science Community Hosted at NERSC. *The International Journal of High Performance Computing Applications*, 17(2):97–124, 2002.

[13] Harry Hochheiser and Ben Shneiderman. Visual specification of queries for finding patterns in time-series data. In *Proceedings of Discovery Science*, pages 441–446, 2001.

[14] Theodore Johnson. Performance measurements of compressed bitmap indices. In *Proceedings of the 25th International Conference on Very Large Data Bases*, September 1999.

[15] Daniel Keim and Hans-Peter Kriegel. Visdb: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14(4):40–49, 1994.

[16] Donald Knuth. *The Art of Computer Programming, 2nd Ed., Volume 3*. Addison-Wesley, 1998.

[17] Anita Komlodi, Penny Rheingans, Utkarsha Ayachit, John Goodall, and Amit Joshi. A user-centered look at glyph-based security visualization. In *Proceedings of the 2005 Workshop on Visualization for Computer Security*, pages 21–28, October 2005.

[18] Eleftherios E. Koutsofios, Stephen C. North, Russell Truscott, and Daniel A. Keim. Visualizing large-scale telecommunication networks and services (case study). In *VIS '99: Proceedings of the conference on Visualization '99*, pages 457–461, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[19] Hans-Peter Kriegel, Peer Kröger, Mattias Schubert, and Ziyue Zhu. Efficient Query Processing in Arbitrary Subspace Using Vector Approximations. In *Proceedings of the International Conference on Scientific and Statistical Database Management*, July 2006.

[20] Kiran Lakkaraju, Ratna Bearavolu, Adam Slagell, William Yurcik, and Stephen North. Closing-the-Loop in NVisionIP: Integrating Discovery and Search in Security Visualizations. In *Proceedings of the 2005 Workshop on Visualization for Computer Security*, October 2005.

[21] Kiran Lakkaraju, William Yurcik, and Adam Lee. NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness. In *Internet Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC-2004)*, October 2004.

[22] Stephen Lau. The Spinning Cube of Potential Doom. *Communications of the ACM*, 47(6):25–26, 2004.

[23] Yarden Livnat, Jim Agutter, Shaun Moon, Robert Erbacher, and Stefano Foresti. A visual paradigm for network intrusion detection. In *Proceedings of the 2005 IEEE Workshop on Information Assurance And Security*, June 17–19 2005.

[24] Jonhathan McPherson, Kwan-Liu Ma, Paul Krystosek, Tony Bartoletti, and Marvin 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 Communication Security*, October 2004.

[25] John-Paul Navarro, Bill Nickless, and Linda Winkler. Combining Cisco Netflow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. In *Proceedings of the 14th Systems Administrator Conference (LISA 2000)*, pages 285–290, 2000.

[26] Tobias Oetiker. Multi router traffic grapher. http://mrtg.hdl.com/.

[27] Tobias Oetiker. Round robin database tool. http://oss.oetiker.ch/rrdtool/.

[28] Patrick O'Neil. Model 204 architecture and performance. In *Second International Workshop in High Performance Transaction Systems*. Lecture Notes in Computer Science, vol. 359, Springer-Verlag 4059, 1987.

[29] Patrick O'Neil and D Quass. Improved query performance with variant indices. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. ACM Press, May 1997.

[30] Vern Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[31] Peter Phaal, Sonia Panchen, and Neil McKee. Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks. IETF RFC 3176, http://www.app.sietf.org/rfc/rfc3176.html, 2001.

[32] Dave Plonka. FlowScan: A Network Traffic Flow Reporting and Visualization Tool. In *Proceedings of the 14th Systems Administrator Conference (LISA 2000)*, pages 305–317, 2000.

[33] Easy Software Products. The fast light toolkit. http://www.fltk.org, 2006.

[34] Lawrence Berkeley National Laboratory Scientific Data Management Group. Fastbit. http://sdm.lbl.gov/fastbit, 2006.

[35] Arie Shoshani, Luis Bernardo, Henrik Nordberg, Doron Rotem, and Alex Sim. Multidimensional indexing and query coordination for tertiary storage management. In *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*. IEEE Computer Society 214225, July 1999.

[36] Kurt Stockinger, Dirk Duellmann, Wolfgang Hoschek, and Erich Schikuta. Improving the performance of high-energy physics analysis through bitmap indices. In *Proceedings of the 11th International Conference on Database and Expert Systems Applications*, 2000.

[37] Kurt Stockinger, John Shalf, Kesheng Wu, and E. Wes Bethel. Query-Driven Visualization of Large Data Sets. In *Proceedings of IEEE Visualization*, pages 167–174, October 2005.

[38] Kurt Stockinger, Kesheng Wu, Rene Brun, and P. Canal. Bitmap indices for fast end-user physics analysis in root. In *Nuclear Instruments and Methods in Physics Research, Section A – Accelerators, Spectrometers, Detectors and Associated Equipment*, volume 599, pages 99–102. Elsevier, 2006.

[39] Cisco Systems. Cisco netflow collection engine. http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/, 2005.

[40] James J. Thomas and Kristin A. Cook eds. *Illuminating the Path – The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[41] Roger Weber, Hans-J. Schek, and Stephen Blott. A Quantative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the International Conference on Very Large Data Bases*, September 1998.

[42] Kesheng Wu, Ekow Otoo, and Arie Shoshani. A performance comparison of bitmap indices. In *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*. ACM 559561, November 2001.

[43] Kesheng Wu, Ekow Otoo, and Arie Shoshani. On the performance of bitmap indices for high cardinality attributes. In *In Proceedings of the International Conference of Very Large Data Bases*, pages 24–35, 2004.

[44] Xiaoxin Yin, William Yurcik, Michael Treaster, Yifan Li, and Kiran Lakkaraju. VisFlowConnect: NetFlow Visualizations of Link Relations for Security Situational Awareness. In *Internet Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC-2004)*, October 2004.