An Interactive Visualization Model for Analyzing Data Storage System Workloads

Steven C. Pungdumri, Chris Lupo, Zoë Wood, and John Oliver

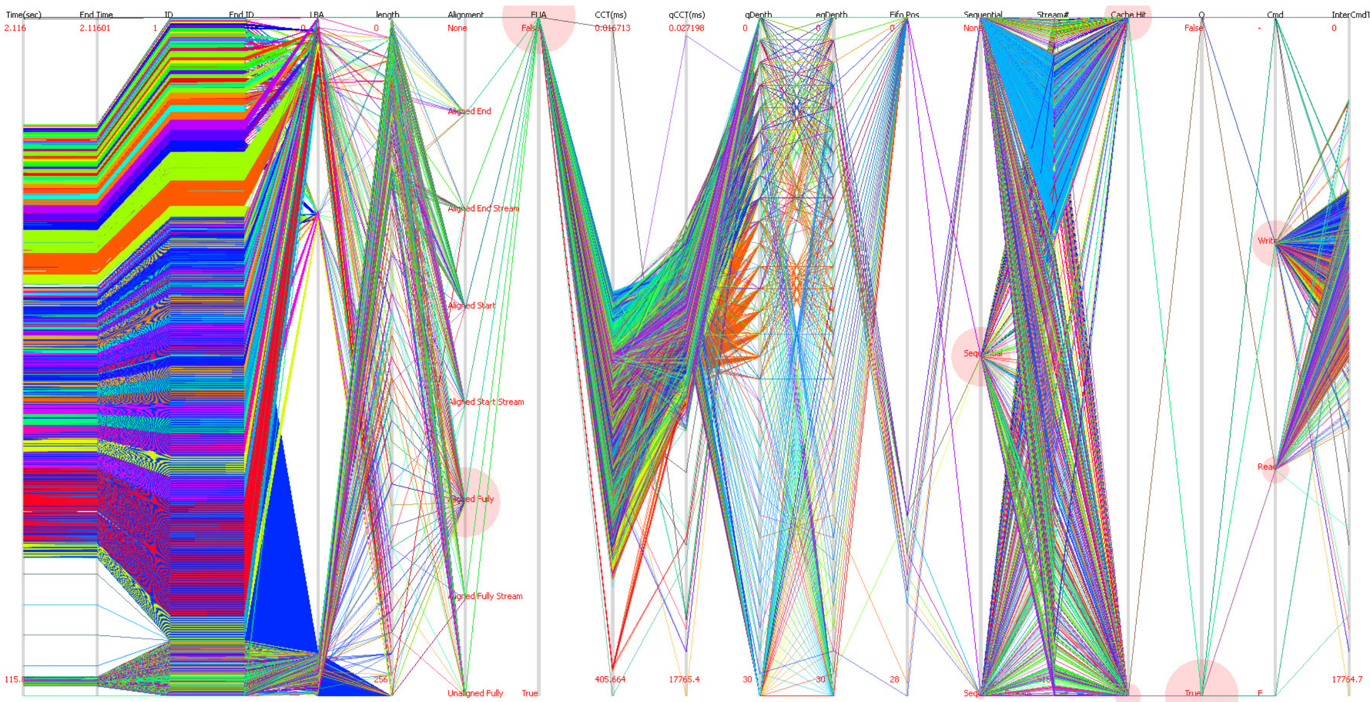

Fig. 1. Overview mode of a sample storage system workload dataset with all dimensions displayed.

Abstract—
The performance of hard drives has become increasingly important as the volume of data storage increases. At the bottom level of large-scale storage networks is the hard drive. Despite the importance of hard drives in a storage network, it is often difficult to analyze the performance of hard drives due to the sheer size of the datasets managed by hard drives. Additionally, hard drive workloads can have several multi-dimensional characteristics, such as access time, queue depth and block-address space. The result is that hard drive workloads are extremely diverse and large, making extracting meaningful information from hard drive workloads very difficult. This is one reason why there are several inefficiencies in storage networks.

In this paper, we develop a tool that assists in communicating valuable insights into these datasets, resulting in an approach that utilizes parallel coordinates to model data storage workloads captured with bus analyzers. Users are presented with an effective visualization of workload captures, along with methods to interact with and manipulate the model in order to more clearly analyze the lowest level of their storage systems.

Design decisions regarding the feature set of this tool are based on the analysis needs of domain experts and feedback from a conducted user study. Results from our user study evaluations demonstrate the efficacy of our tool to observe valuable insights, which can potentially assist in future storage system design and deployment decisions. Using this tool, domain experts were able to manipulate the visualization to make observations and discoveries, such as detecting logical block address banding and observe various dataset trends which were not readily noticeable using conventional analysis methods.

Index Terms— Bus trace analysis, information visualization, storage system, SAS SATA bus activity, data mining, workload, parallel coordinates, focus+context.

1 Introduction

Modeling complex data storage system workloads, specifically bus activity traces between host and storage for analysis, provides an opportunity for storage designers and administrators to interact with and extract information from these complex systems. Analysis from such a model can lead to insights in what is inefficient regarding patterns of drive access, of a particular system design or application [10]. This information assists in identifying possible areas of failure or underutilization in these complex storage systems, avoiding unnecessary maintenance costs.

• *Steven C. Pungdumri, Chris Lupo, Zoë Wood, and John Oliver are with Cal Poly San Luis Obispo, E-mail: spungdum@calpoly.edu, clupo@calpoly.edu, zwood@calpoly.edu, joliver@calpoly.edu.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

1.1 Current Methods

Currently, storage system performance is captured using bus analyzers, a tool designed to observe particular computer bus protocols for the intention of analyzing bus or device failures. Current bus analyzer software tools often focus on bus protocol rather than providing a broad overview of what is occurring between host and storage. Many of the more sophisticated offerings of bus trace analysis tools are developed in house by corporations interested in optimizing their products or systems, rather than the manufacturers of the bus analyzers. Current methods generate graphs which are static and do not allow manipulations to the visualization while viewing the data, lacking functions such as scaling particular areas of interest and highlighting interesting trends. At most, these tools offer various graphs comparing a few dimensions of data at a time, as well as general statistics of the input data [10]. Figure 2 illustrates a screenshot of a tool currently utilized by our collaborators in the storage system industry to analyze storage system bus captures, plotting sector and logical block address (LBA) values. While this plot and similar tools have proven effective at analyzing these datasets, they are limited in the amount of data they can display. These tools are simple and prove effective, however it is not possible to view more than a few dimensions of records at once, which is less likely to effectively convey a larger picture of the activity occurring between host and storage.

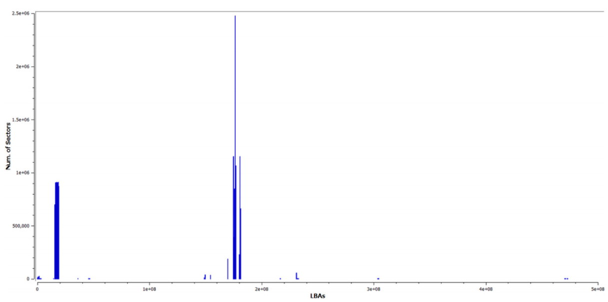

Fig. 2. Output of a tool currently used by our collaborators to analyze the number of sectors and LBA values from a data storage system bus capture [1].

1.2 Our Contribution

We present a tool that incorporates statistical and information visualization techniques, as both an alternative and an addition to current tools available. An initial statistical analysis of the dataset yields identified trends and perceived threads, conveyed to the user in the visualization. The user is presented with an interactive experience in visualizing the bus trace data with our implementation of parallel coordinates—a prominent and effective method of visualizing multi-dimensional datasets. This tool allows the user to manipulate the model through highlighting particular ranges of data, applying focus+context—an information visualization technique that allows users to more specifically analyze particular areas—on ranges of interest, and applying an overlay to display machine identified threads. This tool accommodates bus traces of varying sizes, whether the dataset is a single drive used in a personal computer configuration, or a large-scale data center. Working with real bus analyzer data, we have determined that in order to extensively model large, multi-dimensional datasets of more complex storage systems, parallel coordinates is an effective approach. We present our visualization tool and its contributions toward analyzing storage systems, outlining the various functions implemented in the visualization tool, including our results after conducting a user study and collaborating with data storage system designers in utilizing the tool to analyze a dataset from a corporate data center capture.

2 Related Works in Parallel Coordinates

Parallel coordinates is a method of visualizing multi-dimensional datasets. Each dimension of data is represented by a parallel axis in the model. Records in the dataset are illustrated as a point on each dimension, with the position on the axis corresponding to its value in the particular dimension. Further, the collection of points for each record are connected with line segments between each pair of adjacent axes, resulting in a polyline across the set of dimensions for each record in the dataset. This basic implementation of parallel coordinates has been prevalently used as an effective technique for modeling multi-dimensional datasets in various areas of research [4].

Areas of further research commonly entail the implementation of additional approaches for user interaction or modes of visualization which makes the overall tool more effective pertaining to datasets in a particular area of research. Each specific domain application has its own requirements with each contribution of research, and certain methods that are extremely effective on datasets of one area, may not be effective at all on datasets of another [2] [5] [7] [8]. This section describes a few related works with novel additions to parallel coordinates, designed for datasets of other areas of interest, some more likely useful for modeling data storage system captures as opposed to others.

Data Binning is a preprocessing technique that quantizes data by categorizing the original data points as bins as they fall within specified intervals [11]. Using this technique for large datasets can effectively convey trends, at the cost of less resolution. Novotny and Hauser were effective in illustrating context and outliers by applying a binned data method [7]. This implementation used a preprocess data binning approach to identify common trends between each pair of dimensions, which made it easier to visually comprehend these trends as well as reducing the load on rendering by using parallelograms for each grouping, rather than individual polylines. They proceeded to identify outliers by leveraging their data binning and rendered these with the typical approach of using polylines to ensure their distinction, and rendered the user specified focus (brushing) with the same method as well.

This implementation of parallel coordinates would be effective for any general dataset that requires the identification of trends, outliers, and user focus. As with any approach that decreases data resolution, information can be obfuscated as well. Data binning would be a promising area to investigate for the future work of our tool.

Temporal Depth Cues Johansson et al. investigated the illustration of depth cues in temporal parallel coordinates [5]. Storage system workloads are both spatial and temporal, therefore analyzing the changes across time may produce insights that would be less apparent otherwise. This research produced a temporal window by constructing density maps and utilizing transfer functions. A depth cue visualization was produced based on temporal binning, perception based coloring, and concepts from volume rendering.

This approach does not seem as effective when applied to a data storage analysis capture where informative discoveries lie in the identification of common trends and outliers found in the dataset at all points in time, because the value from an in-depth temporal analysis would illustrate how significant the dataset changes over varying time periods.

Dimension Reordering is another area of parallel coordinates researched to reduce clutter in visualizations [8]. Ordering of dimensions can be important with visualizing multidimensional datasets. It is difficult to manually determine what the most effective arrangement is with many dimensions; this can depend on characteristics of the data being analyzed, as well as the information the user intends to extract from it. Although most implementations of parallel coordinates allow manual reordering, this can be exhaustive when searching for a more effective ordering of dimensions. This particular implementation defined a metric used to measure the amount of clutter, and applied it to all possible arrangements to identify the one with the least amount of clutter. This metric relies on normalized Euclidean distances between data points and a user adjustable threshold to determine the sensitivity of clutter detection. From here, a few algorithms were proposed for optimally searching for the least cluttered arrangement.

While this is an interesting approach, we would like to accumulate more feedback from experts in our data storage system domain to analyze if any trends appear for the arrangement of dimensions that is most effective, and whether it correlates to the amount of clutter observed before incorporating this functionality in our tool. Ultimately, manually analyzing variations in ordering for the nineteen dimensions in our datasets is not feasible, therefore an automated approach such as this would be a beneficial area of future work.

Visualization Enhancing Curves An ambiguity often found in basic implementations of parallel coordinates is when a common point is shared among multiple records, making it difficult to establish which direction each of the records proceeds on towards the adjacent dimension. When following a record preceding the intersecting point on a dimension, the resulting polyline could be any of the polylines stemming from the intersection at the dimension. This particular implementation resolves this conflict with curves instead of polylines, which hints to human visual processing at the resulting direction of each record by following the best fit curve matching to its preceding counterpart [2]. A focus+context feature further spreads these records and includes a bounding box to indicate if a point with multiple records has been spread. The repositioning is calculated by moving the dimensional crossing point proportionally to its average position in the preceding and following dimensions, a technique which can be accomplished without curves as well. The rendering of curves as opposed to polylines comes at the expense of performance, requiring optimizations in data preprocessing as well as repainting algorithms.

This approach seems promising, however it will be left as future work after extensive user feedback to assess the current level of clarity at crossover points across dimensions. From there, it can be determined if the benefits found in this work could extend to modeling data storage system workloads.

Common Objective Each of these prior additions to parallel coordinates are novel and intended to more effectively model the datasets from particular areas of specific domain applications they chose to visualize, or modeling large datasets in general. From this research, we are presented with effective methods to enhance parallel coordinates and more clearly visualize large, multidimensional datasets.

When focusing on a specific area with which to obtain a dataset as opposed to attempting to develop a general purpose visualization tool, it is beneficial to consult domain experts in order to gather requirements to address relevant pain points in order to effectively contribute towards that particular area of research. We gathered specific requirements from Western Digital Corporation with respect to modeling captures from data storage systems by observing what their current tools aid them in, and inquiring what they as domain experts would find interesting to be able to observe. We implemented our addition to parallel coordinates from our observations, that addresses their current needs as well as functionality anticipated to be valuable when analyzing their datasets.

3 Data Storage System Visualization

We present an information visualization tool that analyzes data storage system workloads through bus analyzer captures. This tool is intended for storage system designers, administrators, and others interested in gaining insight in these workloads. Our tool allows users to model bus analyzer captures of data storage system workloads, and provides the following feature set.

Users begin the visualization with an overview of all storage commands displayed using traditional parallel coordinates. All axes are shown by default; they can be hidden or revealed at any point, as well as rearranged for more relevant comparisons between axes. Axes can be displayed with either a linear or logarithmic scale, and particular dimensions are defaulted as such in order to display the spread of values more effectively according to the nature of the dimension. At any point, the user can perform transformations on the model including scaling in and out, as well as translations vertically or horizontally.

Discoveries are made by utilizing functionality such as circular histograms, which conveys general trends across the dataset. Focus+context can be applied by applying brushing on specified ranges of data on any dimension, as well as applying a specific focus to crop important ranges of data. Thread coloring is also a valuable function for analyzing storage system datasets, presenting threads by applying a unique color to each thread identified in the dataset.

3.1 Dataset Overview

Our input dataset output is in CSV format, output from bus analyzer preprocessing software. The values to be expected are both integers and floating point numbers for continuous dimensions, and particular character strings for discrete axes, which are hardcoded as constants expected when preprocessing the input data. Each of the nineteen dimensions of these datasets illustrated in our visualization are listed and described below [12], with continuous axes followed by an asterisk which are described in the following subsection.

- **Time *** - a floating point number that represents the time in seconds that the command was received by the storage device.
- **End Time *** - a floating point number that represents the time in seconds that the command was completed by the storage device.
- **Command ID *** - an integer that represents the order of which the command was received by the storage device.
- **End Command ID *** - an integer that represents the order of which the command was completed by the storage device.
- **Intercommand Time *** - a floating point number that represents the time in milliseconds that the host system takes to issue a new command to the storage device.
- **Logical Block Address *** - an integer containing the starting logical block address of the command.
- **Command Length *** - an integer containing the length of the command.
- **Command Completion Time *** - a floating point number that represents the time in milliseconds from when the command is received by the storage device, and the time the command is completed.
- **Queue Command Completion Time** - a floating point number that represents the completion time in milliseconds of command relative to others in the queue.
- **Queue Depth *** - an integer that contains the queue depth when the command is received.
- **End Queue Depth *** - an integer that contains the queue depth when the command is complete.
- **FIFO Position *** - an integer that contains the position in the queue from the host perspective when the command is executed.
- **Stream Number *** - an integer that contains the stream which the command belongs.
- **Queueable** - a boolean value that signals if the command was sent to the storage device as a command that can be queued.
- **Command Type** - a character string that identifies the type of command sent to the storage device.
- **Alignment** - a character string that describes the alignment properties of the command.
- **Forced Unit Access** - a boolean value that signals if the command is a forced unit command.
- **Sequential** - a character string that describes whether the command is sequential to the previous command that was received, part of a sequential stream, or not sequential to the previous command that was received.
- **Cache Hit** - a character string that describes whether the command is a possible cache hit, determined by the command completion time.

3.2 Continuous Axes

Most axes are categorized as containing continuous values, containing numerical values that span across the dimension. When a user focus is applied, the minimum and maximum values for continuous axes are adjusted to those of the focus in order to spread the values specifically in the area of interest. Each of the continuous axes in storage system datasets are identified with an asterisk in the previous subsection.

3.3 Discrete Axes

Particular axes, namely: Queued, Command, Alignment, Forced Unit Access (FUA), Sequential, and Cache Hit are recognized as axes with discrete values. This is taken into account when parsing input data and displaying the visualization, primarily by hardcoding the character string values and assigning a numerical value when storing the dataset. Discrete axes will display the full range of the possible values found in the dataset regardless of the current user set focus (although the values themselves will not be visible if they are outside of the user set focus). The possible values and descriptions for these axes are listed below.

- **Alignment:** None (undetermined), Aligned End (command starts unaligned, ends aligned), Aligned End Stream (command is part of a sequential stream that starts unaligned, ends aligned), Aligned Start (command starts aligned, ends unaligned), Aligned Start Stream (command is part of a sequential stream that starts aligned, ends unaligned), Aligned Fully (command starts and ends aligned), Unaligned Fully (command starts and ends unaligned)

- **Cache:** Hit (possible cache hit), Miss

- **Command Type:** - (power management command), Flush Cache (signals the storage device to flush its cache), Read (read data from storage device), Write (write data to storage device)

- **Forced Unit Access:** True (access data from a cached copy), False (access data from storage media surface)

- **Queueable:** True (command can be queued), False (command cannot be queued)

- **Sequential:** None (not sequential to the previous command), Sequential (sequential to the previous command), Sequential Stream (part of a stream of sequential commands)

3.4 Circular Histograms

Another defining feature of our visualization is circular histograms, which illustrate point distribution frequencies along discrete axes. Previous attempts at more effectively illustrating the density of one to many values on particular dimensions as well as clarifying crossover ambiguity include the use of curves and point spread [2], context generation [7], and basic overlays of histograms. Initially we implemented traditional histogram rectangles, but found they were hard to observe for particular values with more dense spreads. We instead implement circular histograms with radii corresponding to the proportional ratio for each sum of discrete values in the dataset (constrained to the user specified focus if applied). This approach preserves both the significance of the command values, as well as the illustration of trends across discrete axes, illustrated in Figure 3. From this Figure, it becomes clear that there were more cache hits occurring, a high percentile of accesses were queueable, and that more writes occurred as opposed to reads in this particular dataset. These observations would not be easily perceived without the visual aid of histograms.

3.5 Brushing

Brushing is a common feature in parallel coordinates that allows users to brush a particular range on an axis a distinct color. Users can apply this technique that will extend to other axes and brush all commands that fall under the scope selected. Figure 4 illustrates the use of brushing, and from this we can analyze the accesses which occur in the second half of our capture, and notice that within this subset of data highlighted, queue depth values remain in the lower range in comparison to the rest of the data rendered in black. In Figure 5, we apply brushing to all write commands and observe that the beginning portion of our capture consisted mostly of write commands. Furthermore, we notice that a majority of the write commands in this dataset are definitive cache misses.

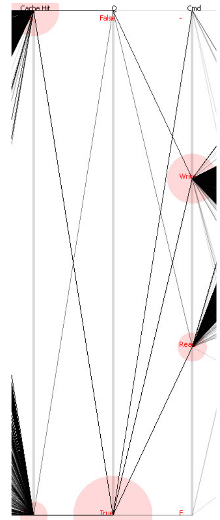

Fig. 3. Our implementation of discrete axes, illustrated with passive circular histograms. Circular histograms are under laid in order to convey trends without interfering with observing dataset records.

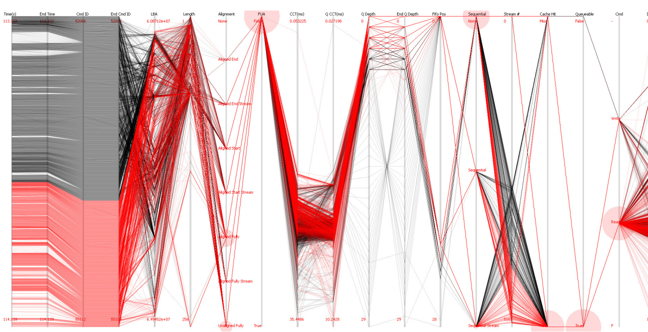

Fig. 4. A focus applied on a particular range of data to filter out all records not within the selected range, as well as brushing applied to all commands within the second half of the dataset.

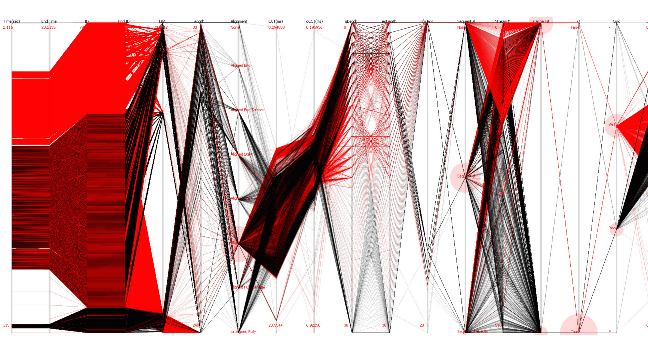

Fig. 5. Viewing the default overview mode of a sample storage system workload dataset, with brushing applied to highlight all write commands in the dataset.

3.6 Focus

Users can apply a focus on the dataset, which crops out all command values outside of the specified range. Applying a focus allows users to examine a smaller, often more manageable subset of data in order to observe occurrences specific to this particular data storage domain, such as interleaved threads and LBA banding. In Figure 7, focusing on a smaller subset of Command ID values and eliminating other axes besides Command ID and LBA clearly exposes thread interleaving, which will be a method of verifying our algorithm for identifying threads in our initial data preprocessing.

3.7 Thread Coloring

Data storage designers we consulted with expressed an interest in identifying interleaved threading, an occurrence where multiple running threads access the storage device in turn, which can lead to LBA banding. In order to illustrate interleaved threads, we implement an algorithm that identifies threads using predefined thresholds for LBA and number of commands. To preserve a visual distinction between threads, we assign a random color to each identified thread. Figure 1 displays an overview of the full dataset with threads rendered in a color unique to the thread.

We suggest the use of thread coloring in conjunction with a user specified focus and beginning with a view showing only the Command ID and LBA axes to observe thread interleaving. An example of this combination of features is illustrated in Figure 6. In this particular screen capture, we can observe a long-running thread colored yellow, which interleaves with the various other threads colored with hues of red, green, and blue. Applying a focus on a smaller subset of the same data, Figure 7 displays the thread interleaving more clearly, with each thread performing a few accesses before trading off with another thread.

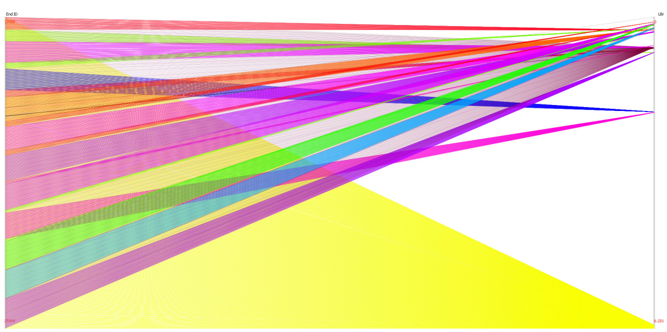

Fig. 6. Applying a focus on a particular range of commands occurring sequentially and observing interleaved threads conveyed by illustrating each machine identified thread with a thread exclusive color.

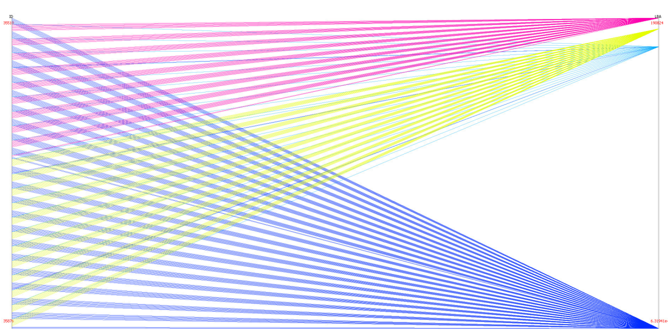

Fig. 7. Focusing on a smaller subset of commands occurring sequentially, which more clearly displays interleaved threads, conveyed by illustrating each machine identified thread with a thread exclusive color.

3.8 Thread Identification Algorithm

We apply preprocessing to identify and assign unique colors to threads, using predefined thresholds for LBA and number of commands. The algorithm maintains a collection of recently identified threads (recent_threads), a default threshold for LBAs (lba_threshold), and a default threshold for number of commands (command_threshold). We iterate through each record in the dataset in the order they appear, and determine if they belong to a recently identified thread by calculating the delta (lba_delta) between the current LBA and each LBA of recent threads, and maintaining a count variable (time_to_live) for each thread, which decrements with each record iterated that does not belong to the thread. This algorithm is detailed below:

for each *record* in *records* **do**
  while $thread\_iterator.hasNext()$ **and** $lba\_delta > lba\_threshold$ **do**
    {Iterate through each thread in recent_threads until the last thread has been observed, or we find a thread that this record belongs to}
    $thread \leftarrow thread\_iterator.next()$
    $lba\_delta \leftarrow |thread.lba - record.lba|$
  end while
  if $lba\_delta \leq lba\_threshold$ **then**
    {This record belongs to a recently identified thread}
    $thread.lba \leftarrow record.lba$
    $thread.time\_to\_live \leftarrow command\_threshold$
  else
    {This record belongs to a new thread}
    $new\_thread.lba \leftarrow record.lba$
    $new\_thread.time\_to\_live \leftarrow command\_threshold$
    $new\_thread.color \leftarrow generateNewColor()$
    $recent\_threads.insert(new\_thread)$
  end if
  Decrement the time_to_live for all recent threads, remove threads from recent_threads if the time_to_live value is zero
end for

Unique colors are generated by manipulating Hue Saturation Brightness (HSB) values, setting saturation and brightness as maximum constants, and incorporating a random number generator to determine hue. Applying this method ensures that each thread will be a different color that is relatively distinguishable from others without the need to consider differing levels of saturation and brightness for the same hue values. Random numbers are generated with a hard coded seed, ensuring that each usage of the tool assigns the same colors for each thread across multiple runs of the tool.

3.9 Implementation

Languages and Frameworks This visualization tool was written in C++, OpenGL, and utilizing the Qt framework. Our tool uses Qt because it is a mature and widely used framework for developing applications for both Microsoft Windows and Linux operating systems [6]. Support for these platforms is necessary for the industry of data storage systems, our primary intended users.

Rendering the data is currently a single-threaded process, however we acknowledge the future performance improvements and usability methods possible with exploring multi-threaded rendering implementation.

Usability Most functionality is presented to the user through context menus, as shown in Figure 8. Here we can see the context menu for the End Time axis, providing the user with options to remove the dimension from the visualization, apply a brush to a range of data, create a focus on a selected range of data, or set the equation of the axis scale for this particular dimension. Although this is not ideal for certain operations such as rearranging axes, it serves as a relatively intuitive method to interact and manipulate parallel coordinates axes through the use of a computer mouse and keyboard.

Defining ranges of data is accomplished through the use of sliders and optional input fields, in order to provide the user with both coarse and fine-grained methods. Figure 9 is an example of the usage

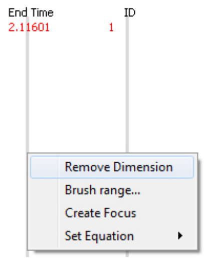

Fig. 8. The use of a context menu to allow the user to interact with and manipulate parallel coordinates axes, as well as apply focus+context features to the visualization.

of sliders and input fields to select a range to brush the range of data to analyze and differentiate, allowing the user to utilize the sliders for coarse values, and the input fields for more specific values. Means of transformations are optimized again for mouse input, allowing translations through clicking and dragging the visualization, and scaling using a slider at the bottom of the application window.

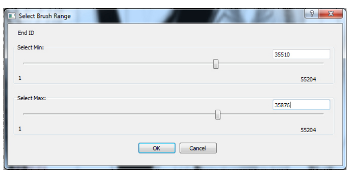

Fig. 9. The use of sliders and input fields to provide coarse and fine-grain methods for selecting ranges of data when applying a focus or brushing.

4 Results

We analyzed various storage system captures with our tool, including corporate production workloads that were successfully obtained on the premise of utilizing our newly developed tool for the purposes of analyzing workload data and offering feedback for increasing storage system efficiency. We then conducted a user study that assisted in identifying areas of our visualization that were effective in comparison to tools currently used, and areas which need improvement to better serve our users.

4.1 Tool Comparison

In order to validate our tool, a comparison between our visualization and current tools were used to analyze both sample and production datasets. Currently, analysis of these datasets consisted of parsing through these records with the aid of spreadsheet software, graphing software, and combining multiple approaches to make discoveries or answer hypotheses. The graphs generated in this process are limited to three dimensions. With our tool, users are able to view the full dataset at once, and navigate through the model with one encompassing tool which provides the following visually enhancing features beyond the capabilities found in current approaches: the ability to visualize all nineteen dimensions of workload captures, focus+context features to focus on subsets of data and highlight areas of interest, circular histograms to quickly convey general trends, and the ability to suggest threads to the user identified with unique colors.

Our tool is further validated by making the same discovery of LBA banding, first observed through the use of traditional analysis methods. We obtained a production data center capture from a enterprise email delivery service, for the purposes of analyzing their workloads and identifying areas for improving drive utilization. We analyzed these datasets with both traditional analysis tools and our parallel coordinates tool with the assistance of domain experts at Western Digital Corporation and made similar discoveries. Figure 10 displays the output of a current tool utilized to identify the occurrence of LBA banding, the act of alternating between two LBA bands. LBA banding introduces seek overhead as the drive read-write head must travel back and forth between each LBA band, possibly leading to waiting for full disk revolutions prior to performing the requested operation. This occurrence can lead to a low queue environment, equating to low throughput, and consequently the potential for poor reliability as the drive is subject to adjacent-track interference (ATI), and an uneven spreading of drive bearing lubrication [1].

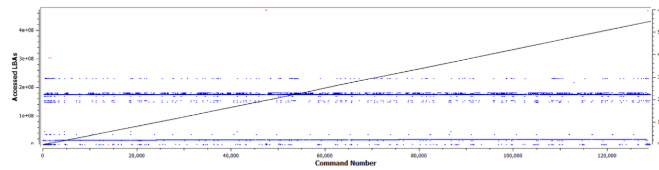

Fig. 10. Output of a tool currently used by our collaborators to identify the occurrence of LBA banding, plotting Accessed LBAs on the left, Command Number on the bottom, and Time on the right [1].

We display the same dataset in Figure 11, which also exhibits the same density of accesses across the LBA axis when compared to the prior figure, captured with current tools. In both figures, there are a few accesses occurring at the highest LBA values, such as 4.72653×10^8, while a majority of accesses across the capture occur at LBA bands in the lower half of LBA values in this particular dataset. Furthermore, we create a focus between commands 62885 and 63127 arbitrarily in Figure 12, which further illustrates the occurrence of LBA banding at block addresses 1.0×10^7 and 1.8105×10^7. This displays LBA banding more clearly with the use of colors to display the various threads involved, and the alternating between two LBA bands through the time sequential commands (not to be confused with LBA sequential) of the capture.

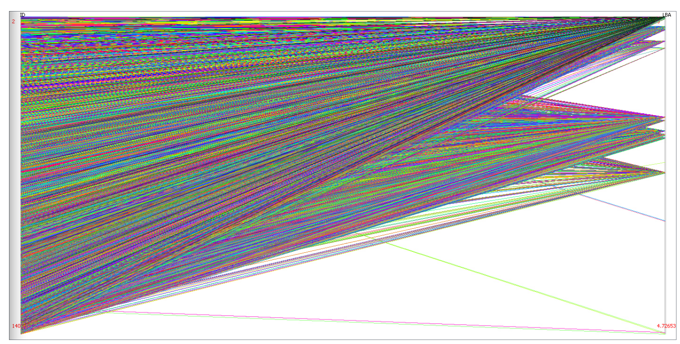

Fig. 11. Output of our tool identifying the occurrence of LBA banding, plotting Command ID on the left and LBA on the right with thread coloring displayed. LBA banding is discovered by observe the behavior of commands on the left accessing various bands of LBA values on the right [1].

Lastly, we present Figure 13 to illustrate the consequences of LBA banding by analyzing the following axes: Time, Command ID, LBA, Command Completion Time (CCT), Queue Command Completion Time (qCCT), Queue Depth (qDepth), End Queue Depth (eqDepth), FIFO Position, and Queueable. LBA banding can lead to a low queue environment; in this dataset, the queue utilization was nonexistent as seen by observing the zero values of all records on the Queue Depth and End Queue Depth dimensions. There is no variation in the values between Command Completion Time and Queue Command Completion Time without queue utilization. Furthermore, every command in this dataset is identified as being queueable. The lack of utilizing command queueing to optimize the order in which access commands are executed indicates that unnecessary drive read-write head movement is likely occurring, which results in a decrease in performance, and consequently increased wear on storage media for workload environ-

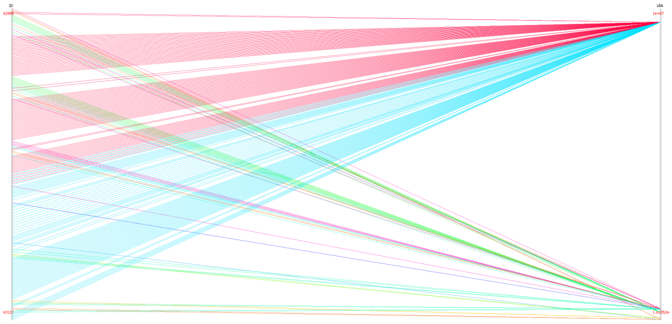

Fig. 12. Output of our tool identifying the occurrence of LBA banding, plotting Command ID on the left and LBA on the right with thread coloring displayed, and a focus applied to a subset of command IDs that occurred sequentially [1].

ments consisting of multiple simultaneous read and write requests, a frequent occurrence in server-type applications [3].

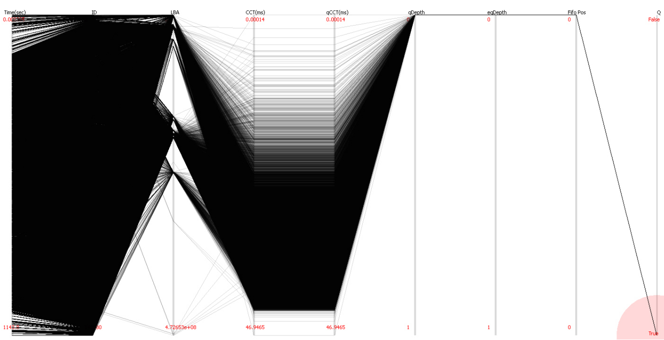

Fig. 13. Our tool identifying the occurrence of LBA banding, displaying the command completion time, queue command completion time, and queue depth dimensions to illustrate a lack of queue utilization in overview mode [1].

These results contribute to the validation of our tool as an effective approach for extracting valuable information from data center captures in order to identify inefficiencies in drive utilization of data storage systems. Furthermore, we introduce our tool to domain experts, academic professors, and graduate students to determine the usability of our visualization application.

4.2 User Study

We conducted a user study in order to determine the effectiveness of our visualization, as well as gather feedback regarding areas of future work. Our user study primarily focused on the ability to convey information from the workload datasets, illustrate general trends and outliers, and the usability of navigating the tool and manipulating the data. We designed our study to incorporate both quantifiable and open-ended channels of feedback in order to refrain from limiting users from expressing ideas for future work.

Our user group consists of a domain expert from Western Digital Corporation, and several academics in the field of computer science and electrical engineering. We provided our users with a sample dataset, the visualization tool, documentation, and both quantifiable and open-ended questions. Questions instructed users to perform a specific functions of the tool on a provided dataset, rate the effectiveness of the particular use case, and provide free-form observations and areas for improvement [9].

Overall, the feedback was generally positive regarding both the effectiveness and usability of our tool. The limitations expressed in this study are identified as our known limitations. Quantifiable questions yielded a 3.5 average regarding the clarity of labeling in the tool, and a 5.0 average regarding the understanding of our thread coloring visualization, both on a 5.0 scale.

Known Limitations We have been able to identify the following limitations in usability and overall effectiveness with the participants in our user study, especially domain experts. It would be more intuitive to perform axis rearranging with drag and drop actions, rather than through the current use of context menus to address usability. Similarly, for selecting the range of data to create a focus on, or brush and highlight, it would be useful to also have the option to click and drag to select ranges within a bounding box.

We gathered that the ability to set and compare multiple focus ranges simultaneously would be a valuable feature in order to increase the effectiveness of our tool. The choice of function and particular axes naming should be explored as well, taking into account recommendations of domain experts of whom this tool is intended for. A few other limitations were expressed, which classify as both usability and effectiveness, namely the ability to undo sequential user commands in the tool, and the ability to save and reload states of the visualization model while using the tool. Some of these limitations were understood prior to the user study and acknowledged as areas that can be addressed with advances in rendering performance, left as future work.

4.3 Conclusion

This paper discusses the value for a comprehensive visualization tool aimed to specifically model data storage system captures. We found parallel coordinates as a widely used, effective approach for modeling multidimensional data, and described other related works that have implemented variations tailored towards particular domain applications. We then presented our implementation, which fully models data storage system trace captures and provides a range of features for user manipulation in order to focus on particular areas of interest. Through our user study, we found that our tool effectively conveys trends and outliers, and serves as a valuable tool for analyzing storage system datasets in comparison to current tools available. We also discovered through feedback that there are some limitations to our tool concerning usability as well as additional functionality.

In the industry of data storage system design and administration, this analysis tool provides insight into the design and deployment of small to large-scale storage systems. As the trend in mass data storage utilization continues to grow, it will be increasingly valuable to identify inefficiencies in storage system design and deployment regarding storage access algorithms, in order to minimize costly failures or underutilized storage media.

4.4 Future Work

This tool can be further developed to address limitations identified in our user study regarding usability and effectiveness through additional features. Some of the limitations identified are dependent on rendering performance; therefore future work entails refinement of the rendering algorithms used, including implementations that utilize parallel computing.

The analysis of large datasets is a difficult task. Incorporating research in both data mining and information visualization as complementary approaches ensures that information can be gained from these datasets with both the statistical and exhaustive characteristics of machine learning, and the cognitive ability to intuitively analyze visual representations through human interaction.

Acknowledgments

The authors wish to thank Western Digital Corporation for their collaboration efforts as domain experts.

References

[1] C. Lupo, J. Oliver, and Western Digital Corporation. Data storage system research collaborative. A collaborative effort between Cal Poly and Western Digital intended to analyze various workloads of corporate data centers in order to address areas of improvement in drive utilization.

[2] M. Graham and J. Kennedy. Using curves to enhance parallel coordinate visualizations. In *Proceedings of the Seventh International Conference on Information Visualization*, pages 10–16, Washington, DC, USA, 2003. IEEE Computer Society.

[3] A. Huffman and J. Clark. Serial ATA Native Command Queuing. White paper, Intel Corporation and Seagate Technology `http://www.seagate.com/content/pdf/whitepaper/D2c_tech_paper_intc-stx_sata_ncq.pdf`, July 2003.

[4] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st Conference on Visualization '90*, VIS '90, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[5] J. Johansson, P. Ljung, and M. Cooper. Depth cues and density in temporal parallel coordinates. In *EuroVis*, pages 35–42, 2007.

[6] Nokia Corporation. Qt - A cross-platform application and UI framework. `http://qt.nokia.com/products/`.

[7] M. Novotny and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(9):893–900, Nov. 2006.

[8] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. *IEEE Symposium on Information Visualization*, 24(4):89–96, Jan. 2005.

[9] S. Pungdumri. An Interactive Visualization Model for Analyzing Data Storage System Workloads. Master's thesis, California Polytechnic State University San Luis Obispo, Mar. 2012.

[10] S. Olds, W. Vinson, and Western Digital Corporation. Understanding drive performance: Performance architecture, modeling, and algorithms. Internal document provided through Cal Poly Data Storage System Research Collaborative.

[11] T. J. Fellers, K. M. Vogt, and M. W. Davidson. CCD signal-to-noise ratio. `http://www.microscopyu.com/tutorials/java/digitalimaging/signaltonoise/index.html`.

[12] Western Digital Corporation. Internal conversations regarding internal software tools currently used to analyze storage system workloads.