PerfViz: A Visualization Tool for Analyzing, Exploring, and Comparing Storage Controller Performance Data

Amit P. Sawant[a], Matti Vanninen[b], and Christopher G. Healey[c]

[a, c]North Carolina State University, Department of Computer Science, Raleigh, NC, USA;
[b]Network Appliance Inc., Research Triangle Park, NC, USA

ABSTRACT

This paper presents a technique that allows viewers to visually analyze, explore, and compare a storage controller's performance. We present an algorithm that visualizes storage controller's performance metrics along a traditional *2D grid* or a linear space-filling *spiral*. We use graphical "glyphs" (simple geometric objects) that vary in color, spatial placement and texture properties to represent the attribute values contained in a data element. When shown together, the glyphs form visual patterns that support exploration, facilitate discovery of data characteristics, relationships, and highlight trends and exceptions. We identified four important goals for our project:

1. Design a graphical glyph that supports flexibility in its placement, and in its ability to represent multidimensional data elements.
2. Build an effective visualization technique that uses glyphs to represent the results gathered from running different tests on the storage controllers by varying their performance parameters.
3. Build an effective representation to compare the performance of storage controller(s) during different time intervals.
4. Work with domain experts to select properties of storage controller performance data that are most useful to visualize.

Keywords: information visualization, multidimensional visualization, perception, visual features, file and storage

1. INTRODUCTION

A big challenge in studying file and storage systems is discovering interrelationships between multiple performance metrics. It is difficult to analyze and interpret this abstract collection of multidimensional performance data using traditional non-visual techniques. One possible solution is to use visualization techniques to convert large amounts of data into an image that domain experts can use for exploring, discovering, comparing, validating, accounting, monitoring, identifying faults, and studying the effects of adjusting different performance parameters.

Visualization presents information in a pictorial form.[1] Formally, a dataset D contains n data elements, e_i, such that $D = \{e_1, ..., e_n\}$. A dataset represents a set of data attributes, $A = \{A_1, ..., A_m\}$, $m > 1$. Data elements encode values for each attribute: $e_i = \{a_{i,1}, ..., a_{i,m}\}$, $a_{i,j} \in A_j$. A data-feature mapping converts raw data into visual information. Such a mapping is denoted by $M(V, \Phi)$, where $V = \{V_1, ..., V_m\}$ is a set of m visual features with V_j selected to represent each attribute A_j, and $\Phi_j : A_j \rightarrow V_j$ maps the domain of A_j to the range of displayable values in V_j. Thus, visualization is the process of selecting an appropriate M. An effective M produces images that support rapid, accurate, and effortless exploration and analysis.[2] Effectively mapping data attributes to visual features requires better understanding the properties of the visual features themselves.

The remainder of this paper proceeds as follows. In Section 2, we survey perceptual visual features. Section 3 describes the implementation design of our visualization tool (*PerfViz*). Section 4 provides details on data collection. Section 5 and 6 provide an example of visualizing performance data, and comparing performance data respectively. Finally, Section 7 discusses conclusions and future work.

Further author information: (Send correspondence to Amit P. Sawant)
Amit P. Sawant, E-mail: amit.sawant@ncsu.edu
Matti Vanninen, E-mail: matti.vanninen@netapp.com
Christopher G. Healey, E-mail: healey@csc.ncsu.edu

2. PERCEPTUAL VISUAL FEATURES

A variety of visual features have been used in visualization.[3] The use of color and texture has a long history in the graphics, vision, and visualization literature. Our data-feature mappings are constructed from psychophysical studies of how the visual system "sees" fundamental visual properties in an image.

Color is widely used in many visualization techniques. Simple color schemes include the rainbow spectrum, red-blue or red-green ramps, and the gray-red saturation scale, whereas more complex techniques attempt to control the perceived difference between different colors.[4] Researchers in visualization have combined perceptually balanced color models with nonlinear mappings to emphasize changes across specific parts of an attributes domain, and have also proposed automatic colormap selection algorithms based on an attributes spatial frequency, continuous or discrete nature, and the analysis tasks to be performed. Experiments have shown that color distance, linear separation, and color category must all be controlled to select discrete collections of distinguishable colors.[5]

Scientists have used fundamental perceptual properties of texture such as regularity, directionality, contrast, size, and coarseness to model human vision, to perform texture segmentation and classification, and more recently to visualize multiple data attributes.[6–8] In visualization an individual attribute's values control a corresponding texture dimension. This results in a display that changes its visual appearance based on the data in the underlying dataset. Healey and Enns constructed perceptual texture elements (pexels) that varied in size, density, and regularity.[5] They found that size and density are perceptually salient, but variations in regularity are much more difficult to detect. Later work showed that 2D orientation can also be used to visualize data.

3. *PERFVIZ* DESIGN

We adopted the following guidelines proposed by Eick for designing our visualization system:[1] (1) ensure that the visualization is focused on the users needs by understanding the data analysis task; (2) encode data using color and other visual characteristics; and (3) facilitate interaction by providing a direct manipulation user interface. We also used the principle of a visualization framework to build our tool that starts with data management and continues through assisted visualization design, display, navigation, and user interaction.[9]

A number of well-known techniques exist for visualizing non-spatial datasets. These can be roughly classified as geometric projection, iconic display, hierarchical, graph-based, pixel-oriented and dynamic (or some combination thereof).[10,11] We decided a dynamic iconic display was most relevant to our goal of visualizing a storage controller's performance data. Our visualizations were designed by first constructing an object to represent a single data element. Next, the objects are positioned to produce a static visualization of the storage controller's performance metrics. Glyphs are positioned along a traditional *2D grid* or a linear *spiral* embedded in a plane based on scalar ranking attribute(s). In the following subsections we provide a brief discussion on the different glyph representations, placement algorithms, and data-feature mapping.

3.1. Glyph Representation

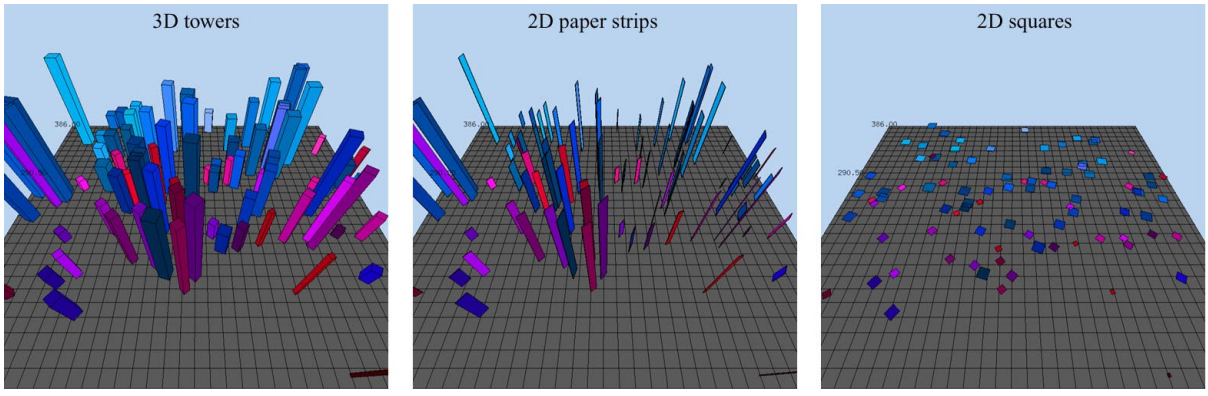

Figure 1. Glyph representation: *3D towers*, *2D paper strips*, and *2D squares*

We use a simple glyph to represent a data element as shown in Figure 1. We implemented glyphs that look like a *3D tower*, *2D paper strip*, and a *2D square*. Each glyph representation has its advantages and disadvantages. A geometric glyph can vary more visual features than a single pixel (*e.g.*, height, density, or orientation).

3D geometric glyph can represent multiple attribute values for each data element. A *3D glyph* also offers more surface area on which to place information, compared to a 2D representation. Psychophysical experiments have shown that viewers can properly identify and compare color and texture properties of 3D objects, as long as they are perceived as being displayed in a 3D environment.[5,12] Visualizing glyphs on an underlying plane is used to reinforce this perception. *3D glyphs* also generate potential disadvantages, for example, occlusion, and a decrease in the total number of glyphs we can display. Our tool supports interactive camera positioning, to further reduce occlusion effects, and glyphs' positions are modified slightly as needed to avoid overlap. We observed that in most situations users find it easier to detect the attribute values mapped to orientation better with *2D paper strip* representation than with *3D tower*.

3.2. Placement Algorithm

Glyphs representing the attribute values embedded in a dataset have to be placed appropriately in order to create an information workspace for visual sense making. We decided to use two layout methods: a traditional *2D grid*, and a *spiral*.

3.2.1. 2D Grid Layout

A *2D grid* layout is very intuitive and a commonly used placement algorithm in visualization systems. A two-dimensional ordering is imposed on the data elements through user-selected scalar attributes.

3.2.2. Spiral Layout

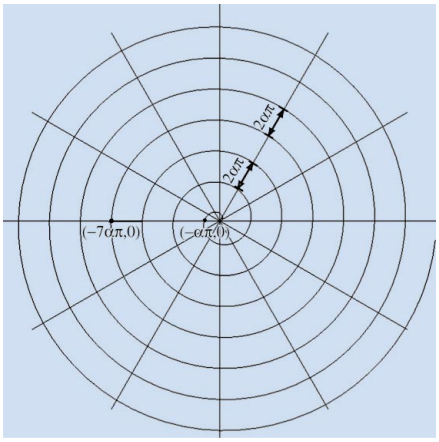

Figure 2. An Archimedean spiral with $\theta_{\max} = 14\pi$ radians producing seven complete turns about the origin

A one-dimensional ordering is imposed on the data elements through a user-selected scalar attribute, or "ranking" attribute. We needed a way to map this ordering to a 2D spatial position for each element. We chose to use a 2D space-filling *spiral* to satisfy this requirement.

Our algorithm is based on a technique introduced by Carlis and Konstan to display data along an Archimedean spiral.[13] Figure 2 shows an Archimedean spiral represented in its polar form as:

$$\left.\begin{array}{rcl} r & = & \alpha\theta \\ x & = & r\cos\theta \\ y & = & r\sin\theta \end{array}\right\} \; 0 \le \theta \le \theta_{\max} \qquad (1)$$

where θ tracks position along the spiral, r is the distance from the center of the spiral for a given θ, and α is a constant used to control the spacing between neighboring arcs in the spiral (Figure 2).

Visualizations along a spiral have been used with both abstract and periodic data to provide a number of additional properties, including: (1) comparison of values in a neighborhood along a small section of the spiral; (2) comparison of several consecutive cycles of the spiral; (3) identification of periodic patterns in the data; and (4) changes in patterns in the data over the length of the spiral (*e.g.*, different patterns in the inner rings of the spiral versus the outer rings).[14]

3.3. DATA-FEATURE MAPPING

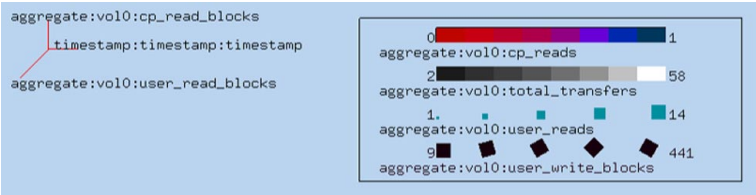

Figure 3. A data-feature mapping - each glyph represents one second of workload to a storage controller, *timestamp* → x position, *cp_read_blocks* → y position, *cp_reads* → hue, *total_transfers* → luminance, *user_reads_blocks* → height, *user_reads* → size, *user_write_blocks* → orientation

When we design a visualization, properties of the dataset and the visual features used to represent its data elements must be carefully controlled to produce an effective result. Important characteristics that must be considered include:[15] (1) dimensionality (number of attributes in the dataset), (2) number of elements, (3) visual-feature salience (strengths and limitations that make it suitable for certain types of data attributes and analysis tasks), and (4) visual interference (different visual features can interact with one another, producing visual interference; this must be controlled or eliminated to guarantee effective exploration and analysis).

Using the knowledge of the above perceptual guidelines, we choose visual features that are highly salient, both in isolation and in combination. We map features to individual data attributes in ways that draw a viewer's focus of attention to important areas in a visualization. The ability to harness the low-level human visual system is attractive, since: (1) high-level exploration and analysis tasks are rapid and accurate, (2) analysis is display size insensitive, and (3) different features can interact with one another to mask information; psychophysical experiments allow us to identify and avoid these visual interference patterns.

Our glyphs support variation of spatial position, color and texture properties, including *x position*, *y position* or *linear radial position*, *hue*, *luminance*, *height*, *size*, and *orientation*. A glyph uses the attribute values of the data element it represents to select specific values of the visual features to display. In consultation with the domain experts we identify the most important attributes and create a default data-feature mapping. The most important attributes should be mapped to the most salient features and secondary data should never be visualized in a way that would lead to visual interference. The order of importance for visual features is *luminance*, *hue*, and then various texture properties. Figure 3 shows an example of a data-feature mapping used to visualize the performance of a volume*.

Figure 4 walks through the steps for generating visualizations with *PerfViz*. We observed that using all the visual features initially and mapping them to different attributes overwhelmed the user. Thus, when the user runs the tool, only the *x position*, *y position* (or *linear radial position*) are mapped to the first attribute in the dataset, and all the other visual features are turned off. Once the user decides which attributes to explore and analyze, the user can map more attributes to the remaining visual features, and thereby gain more insight into the dataset.

For comparing multiple datasets (performance of the storage controller during different time intervals or performance of different storage controllers over a given time period) we use *hue* to represent different datasets (time intervals). In the *2D grid* layout we overlay the glyphs from different time intervals on the grid, and in a *spiral* layout we render the glyphs on spirals that are slightly off-set along the *x-axis* as shown in Figures 5b, d. We found these techniques for both the *2D grid* layout and the *spiral* layout produce better results (gain more insight into the similarities and differences in the datasets) compared to rendering the time intervals sequentially one after the other as shown in Figures 5a, c.

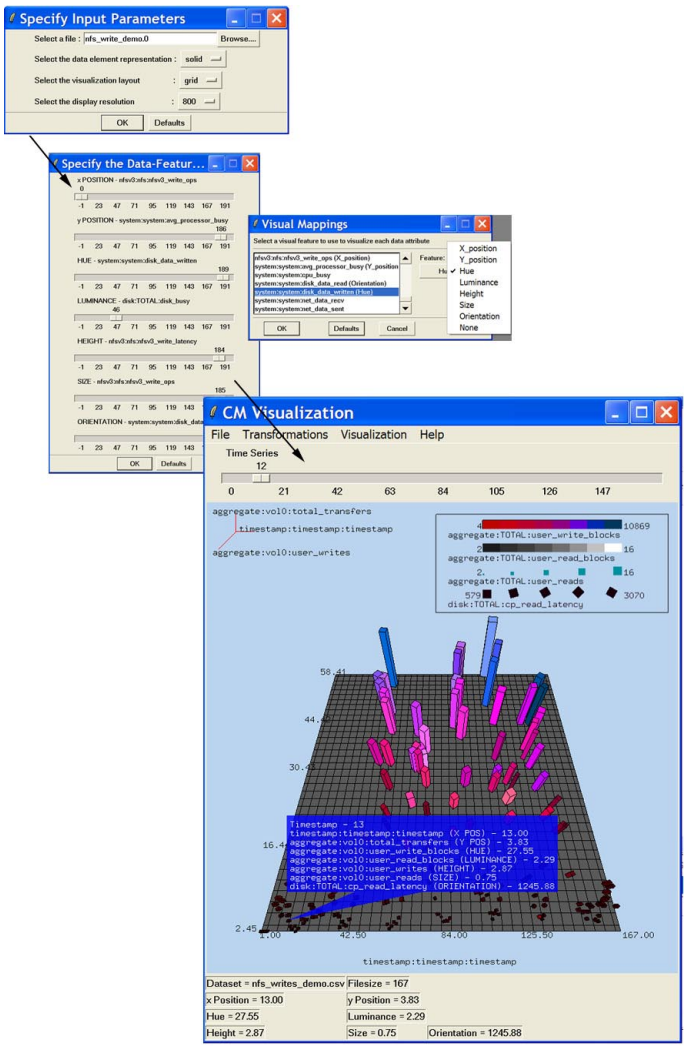

Figure 4. *PerfViz* - Visualization system showing the input dialog, the data-feature mapping sliders and listbox, and the visualization window

4. DATA COLLECTION

The visualization process begins by collaborating with domain experts to identify the storage performance metrics they want to see, analyze, explore, compare, and monitor.

In Data ONTAP (operating system on NetApp storage controllers), there is a layer of performance data management software called Counter Manager (CM) that separates the data collector routines, such as ZAPI, SNMP, and CLI (referred to as clients or consumers of CM) as shown in Figure 6. This layer ensures that any changes to the underlying performance counters do not require any corresponding changes to the clients of CM and at the same time provides a consistent global view of the available performance data.

The CM layer provides an object abstraction to higher layers and groups the performance data with an object-instance-counter hierarchy. An example of such hierarchy is the disk object. This object has a number of instances equal to the number of active disks in the system, and each of these instances has a set of counters associated with it, *e.g.*, number of reads per second, number of writes per second, and so on. The amount of raw data collected by the CM is enormous

*volume is a basic data container unit of WAFL (NetApp's Write Anywhere File Layout).

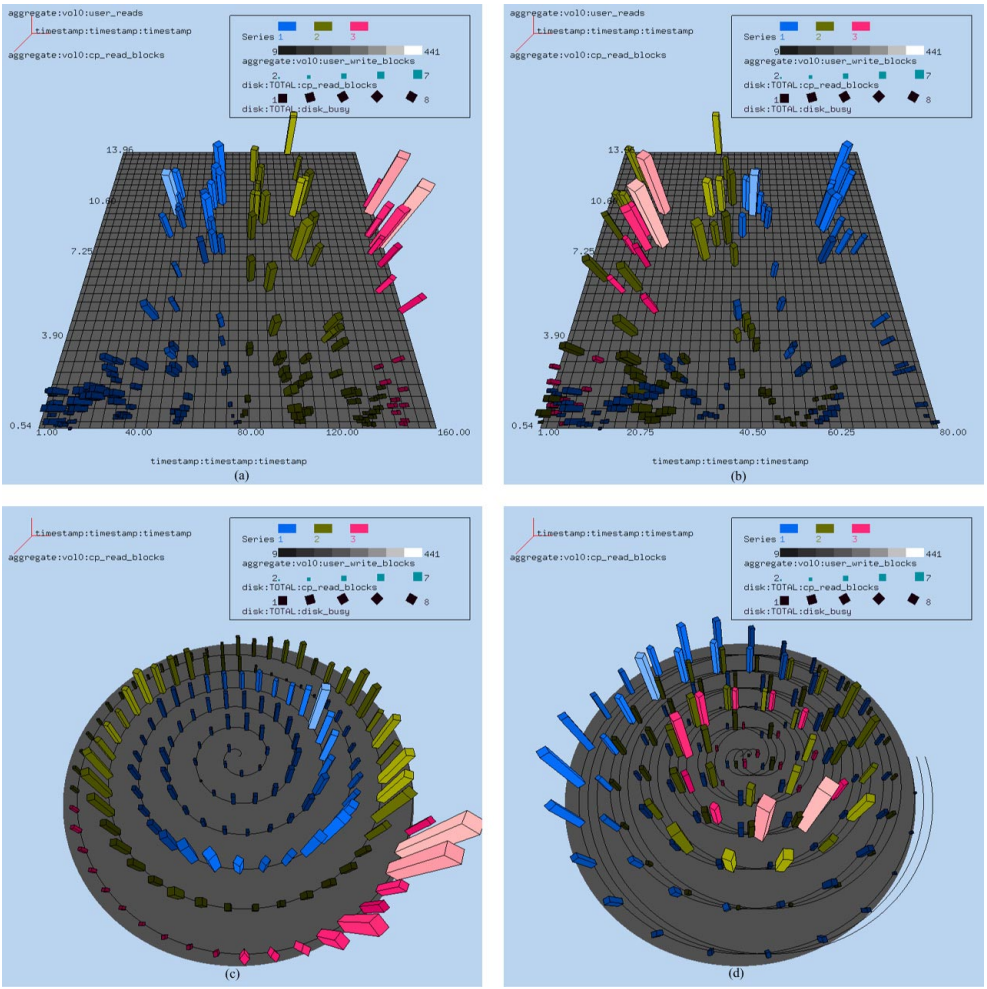

Figure 5. Comparison of storage controller performance data during multiple time intervals - (a) *serial* placement in *2D grid* layout, (b) *parallel* placement in *2D grid* layout, (c) *serial* placement in *spiral* layout, and (d) *parallel* placement in *spiral* layout

containing large number of data series, where each data series is often characterized by multiple counters. In this paper counters and attributes are used interchangeably.

5. VISUALIZING PERFORMANCE DATA

We provide an example of visualizing NFS writes workload to a storage controller. NetApp storage controllers cache writes, enabling immediate acknowledgment to clients and low overall response times. Dirty data is periodically, asynchronous flushed to disk in an event called a consistency point (CP). A CP increases CPU and disk utilization. The glyphs representing the NFS workload to a storage controller can be displayed along a traditional *2D grid* or a space-filling *spiral*.

5.1. Layout: *2D Grid* Algorithm

Interpretation.

1. From Figure 7, the low, dark glyphs at the bottom right indicate samples when conditions are ideal, that is, higher throughput (*x position*, *size*), lower response time (*height*), lower CPU utilization (*y position*), and lower disk utilization (*orientation*). In contrast, the tall glyphs at the top left indicate periods of lower throughput, higher response

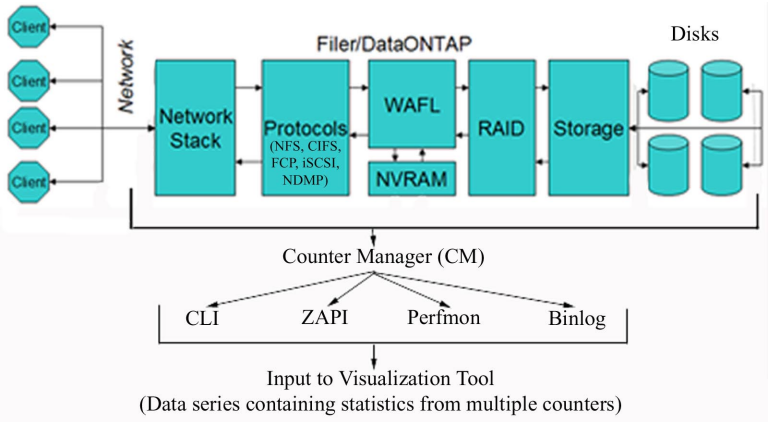

Figure 6. Block diagram of a NetApp Filer System: ONTAP - Open Network Technology for Appliance Products; WAFL - NetApp's Write Anywhere File Layout; ZAPI - management API; CLI - Command Line Interface

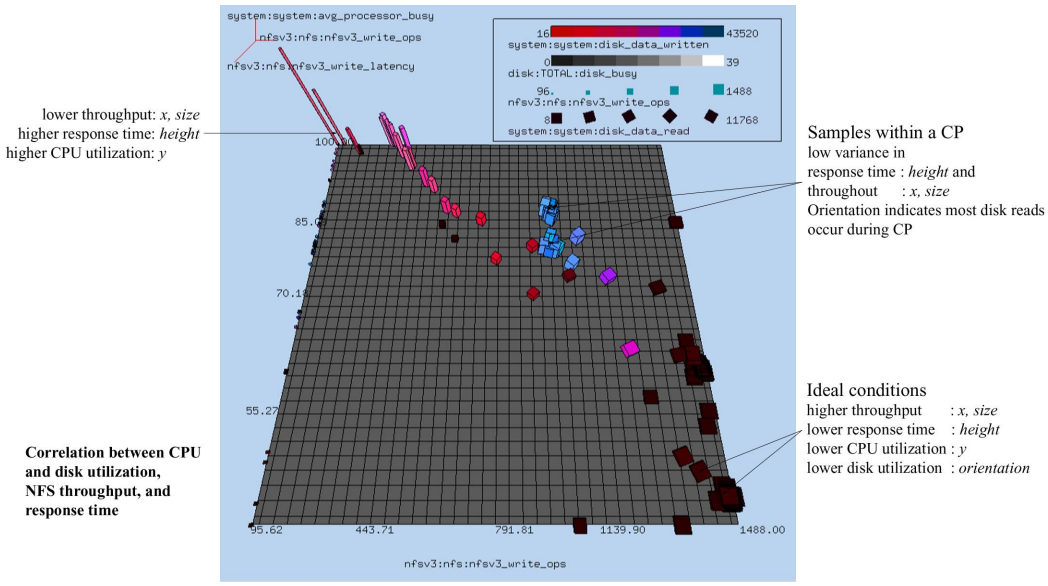

Figure 7. *2D grid* layout - each glyph represents one second of an NFS write workload to a storage controller; *nfs_write_ops* → x position, *avg_processor_busy* → y position, *disk_data_written* → hue, *disk_busy* → luminance, *nfs_write_latency* → height, *nfs_write_ops* → size, *disk_data_read* → orientation

time, and higher CPU utilization. The visualization clearly shows the correlation between CPU and disk utilization, NFS throughput, and response time.

2. The cluster of bright blue glyphs near the center indicate samples within a CP, when data is being flushed to disk. The tight concentration suggests that storage controller response time and throughput have low variance during CP, and are slightly worse than the best case, during a CP. The orientation of these samples indicate that most disk (RAID parity) reads occur during this period.

3. The lone low, dark glyph near the top left is an anomalous outlier. This indicates a sample where CPU utilization was high and throughput was low, but response time was not correspondingly high and disks were not busy. This may indicate that the storage controller CPU resources were consumed by some background activity outside of a CP.

4. The bright red glyphs near the center represent metadata which must be loaded at the start of a CP.

5.2. Layout: *Spiral* Algorithm

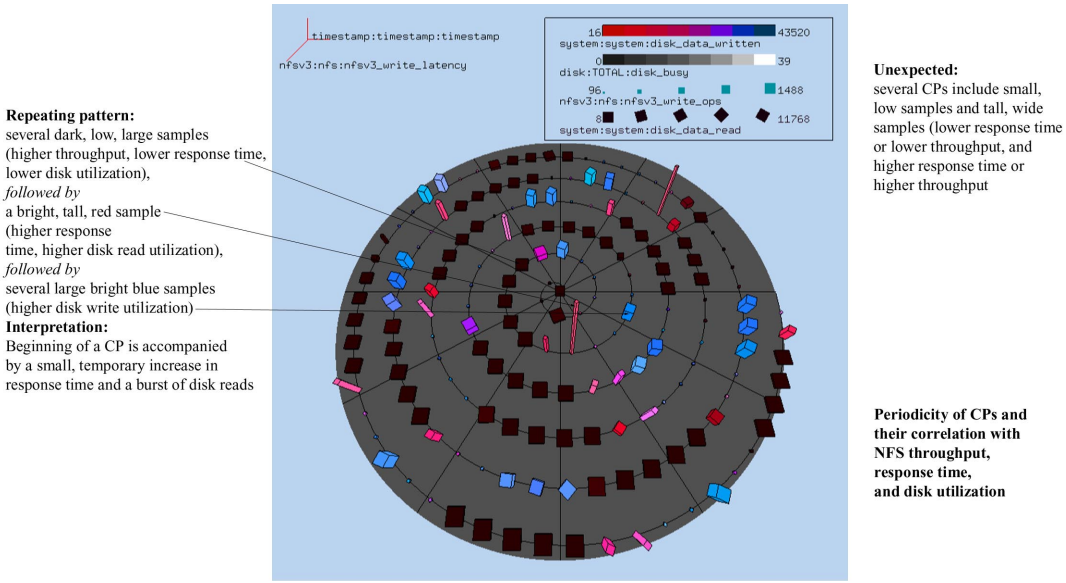

Figure 8. *Spiral* layout - each glyph represents one second of an NFS write workload to a storage controller; *timestamp* → linear radial position, *disk_data_written* → hue, *disk_busy* → luminance, *nfs_write_latency* → height, *nfs_write_ops* → size, *disk_data_read* → orientation

Interpretation.

1. From Figure 8, in the *spiral* time series visualization, the periodicity of CPs and their correlation with NFS throughput and response time, and disk utilization is evident. A repeating pattern of several dark, low, large samples (higher NFS throughput, lower NFS response time, lower disk utilization), followed by a bright, tall, red sample (higher response time, higher disk read utilization), followed by several large bright blue samples (higher disk write utilization) is evident. This suggests that the beginning of a CP is accompanied by a small, temporary increase in NFS response time and a burst of disk reads, probably for metadata needed to complete the CP. The majority of the CP is spent doing disk writes as buffered data is committed to disk.

2. Several unexpected artifacts are also evident from the visualization. Most notably, several CPs include small, low samples and tall, wide samples, indicating lower response time/lower throughput and higher response time/higher throughput respectively. We expected an inverse relationship between these attributes, so this may be indicative of variance in the workload being presented by the client.

6. COMPARING PERFORMANCE DATA

We provide an example of visualizing two different time intervals of NFS reads and writes workload to a storage controller. The first time interval is represented by blue hue, and the second time interval is represented by red hue. The *serial* and *parallel 2D grid layout*, and *serial* and *parallel spiral layout* are used for comparing performance data, and they both together provide an overall insight into multiple time intervals of the storage controller.

6.1. Layout: *Serial* and *Parallel 2D Grid*

Interpretation.

1. From Figure 9a, in the *serial 2D grid* layout we can conclude that the overall performance trend of the storage controller during both the time intervals was similar.

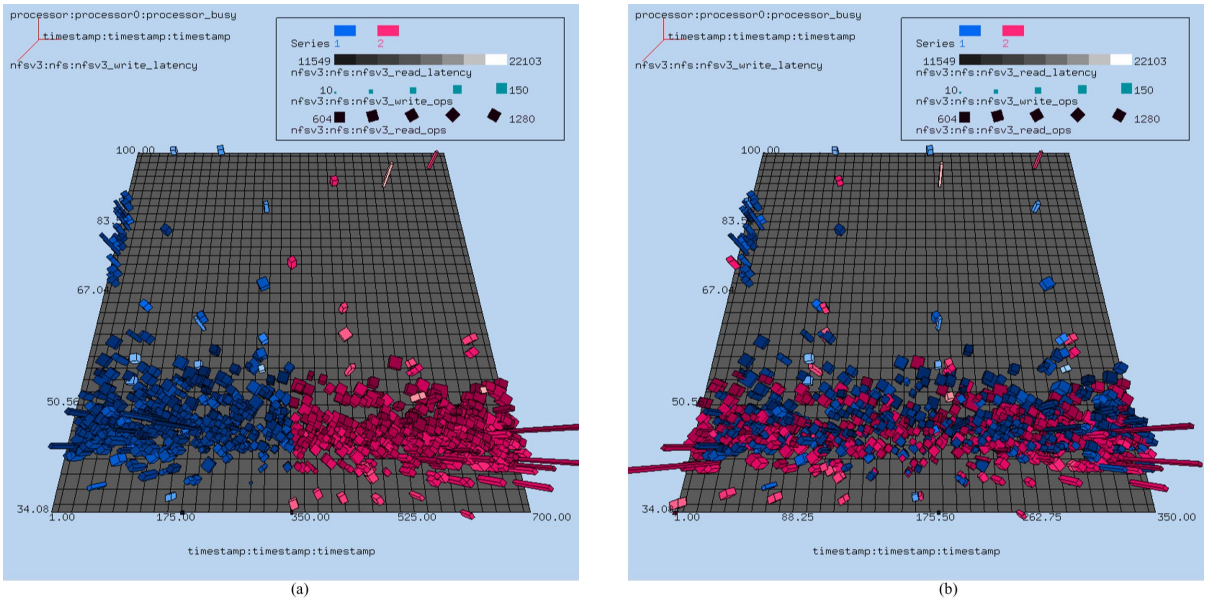

Figure 9. (a) *Serial 2D Grid* layout and (b) *Parallel 2D Grid* layout - each glyph represents one second of an NFS read and write workload to a storage controller; blue hue represents time interval 1, and red hue represents time interval 2; *timestamp* → x position, *processor_busy* → y position, *nfs_read_latency* → luminance, *nfs_write_latency* → height, *nfs_write_ops* → size, *nfs_read_ops* → orientation

2. The tight concentration of the glyphs in each time interval suggests that storage controller response time (*height*) and throughput (*size*) have low variance.

3. In Figure 9b the *parallel 2D grid* layout provides a representation for detailed comparison as the glyphs that belong in the same sequence of the time intervals are placed along the same *x position*.

4. In the initial stages of the first time interval, the CPU utilization (*y position*) was higher compared to the second time interval.

5. The low variance in the *orientation* represent almost constant NFS reads.

6.2. Layout: *Serial* and *Parallel Spiral*

Interpretation.

1. In the *serial spiral* time series visualization we can observe that the number of spikes representing a sudden increase in write response time (*height*) in the second time interval (*red hue*) are more compared to the first time interval (*blue hue*) as shown in Figure 10a.

2. From Figure 10b, in the *parallel spiral* layout, low variance between write response time (*height*) and throughput (*size*) is evident.

7. ACKNOWLEDGMENTS

This work is supported by Network Appliance, Inc.

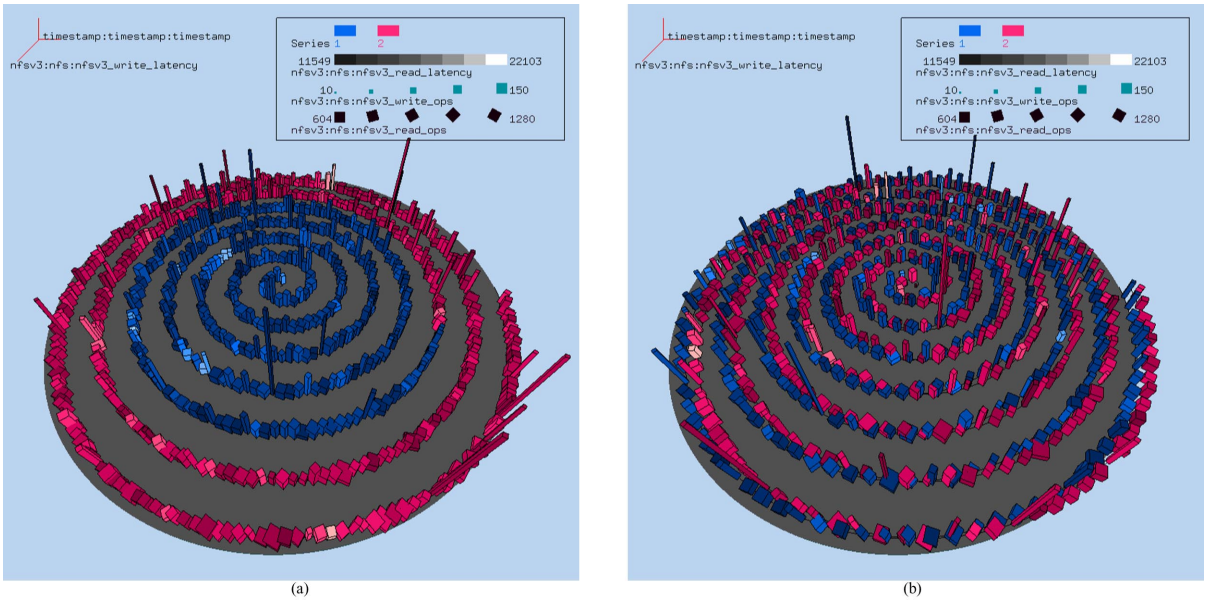

Figure 10. (a) *Serial Spiral* layout and (b) *Parallel Spiral* layout - each glyph represents one second of an NFS read and write workload to a storage controller; blue hue represents time interval 1, and red hue represents time interval 2; *timestamp* → x position, *nfs_read_latency* → luminance, *nfs_write_latency* → height, *nfs_write_ops* → size, *nfs_read_ops* → orientation

8. CONCLUSIONS AND FUTURE WORK

We use rules of human perception to build displays that harness the strengths and avoid the limitations of low-level human vision. Individual data elements are presented using graphical "glyphs" that vary their color, placement, and texture properties to encode the element's attribute values. The result is a display that allows viewers to rapidly and accurately analyze, explore, compare, and discover within a storage controller's performance data. Our technique is not restricted to storage controller performance data. It can be applied in any situation where appropriate ranking attribute(s) can be identified to control glyph placement.

We would like to test the flexibility of our visualization tool on different types of multidimensional datasets. We plan to enhance the comparison of multiple datasets (time intervals) using animation techniques,[16,17] intelligent camera planning, and data summarization techniques. We also plan to conduct simple validation studies to investigate our design choices.

REFERENCES

1. S. G. Eick, "Engineering perceptually effective visualizations for abstract data," in *Scientific Visualization, Overviews, Methodologies, and Techniques*, pp. 191–210, IEEE Computer Society, (Washington, DC, USA), 1997.
2. C. G. Healey, "Formalizing artistic techniques and scientific visualization for painted renditions for complex information spaces," in *Proceedings International Joint Conference on Artificial Intelligence 2001*, pp. 371–376, (Seattle, Washington), 2001.
3. C. G. Healey, "Perceptual colors and textures for scientific visualization," 1998.
4. C. Ware, "Information visualization: Perception for design," pp. 230–239, 2000.
5. C. G. Healey and J. T. Enns, "Large datasets at a glance: Combining textures and colors in scientific visualization," *TVCG* **5**(2), pp. 145–167, 1999.
6. R. M. Haralick, K. Shanmugam, and I. Dinstein in *IEEE Transactions on System, Man, and Cybernetics SMC-3*, **3**(6), pp. 610–621, 1973.
7. R. A. Rao and G. L. Lohse, "Identifying high level features of texture perception," *CVGIP: Graph. Models Image Process.* **55**(3), pp. 218–233, 1993.

8. R. A. Rao and G. L. Lohse, “Towards a texture naming system: Identifying relevant dimensions in texture.,” in *IEEE Visualization*, pp. 220–227, 1993.
9. B. Dennis, S. Kocherlakota, A. P. Sawant, L. Tateosian, and C. G. Healey, “Designing a visualization framework for multidimensional data,” *IEEE Computer Graphics and Applications* **25**(6), pp. 10–15, 2005.
10. D. A. Keim, “Pixel-oriented database visualizations,” *SIGMOD Record (ACM Special Interest Group on Management of Data)* **25**(4), pp. 35–39, 1996.
11. M. Foltz and R. Davis, “Query by attention: Visually searchable information maps,” in *Proceedings of Fifth International Conference on Information Visualisation*, pp. 85–96, London, England, 2001.
12. D. J. Aks and J. T. Enns, “Visual search for size is influenced by a background texture gradient,” *Journal of Experimental Psychology: Human Perception & Performance* **22**(6), pp. 1467–1481, 1996.
13. J. V. Carlis and J. Konstan, “Interactive visualization of serial periodic data,” *ACM Symposium on User Interface Software and Technology* , pp. 29–38, 1998.
14. M. Weber, M. Alexa, and W. Muller, “Visualizing time-series on spirals,” *Proceedings of the IEEE InfoVis Symposium, IEEE Press, Los Alamitos, CA* , 2001.
15. C. Weigle, W. Emigh, G. Liu, R. Taylor, J. T. Enns, and C. G. Healey, “Oriented texture slivers: A technique for local value estimation of multiple scalar fields,” in *Proceedings Graphics Interface 2000*, pp. 163–170, (Montréal, Canada), 2000.
16. A. P. Sawant, “Dynamic visualization of the relationship between multiple representations of an abstract information space,” 2003.
17. K.-P. Yee, D. Fisher, R. Dhamija, and M. A. Hearst, “Animated exploration of dynamic graphs with radial layout,” in *INFOVIS*, pp. 43–50, 2001.