Performance Visualization: 2-D, 3-D, and Beyond*

Daniel A. Reed   Matthew J. Gardner   Evgenia Smirni

{reed,mjgardne,esmirni}@cs.uiuc.edu
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Abstract

During the past ten years, performance data visualization techniques have evolved from static, two-dimensional graphics to dynamic graphics and immersive virtual environments. We sketch the domain of applicability for each visualization technique using analysis of input/output behavior and WWW traffic as example problem domains. With this background, we describe experiences with virtual environment representations of complex performance data and the potential for immersive interaction with complex software.

1 Introduction

Despite the ubiquity of scalable parallel systems and high-performance workstation clusters, achieving substantial fractions of these systems' peak performance for a wide range of applications remains an elusive goal. Although the causes for poor performance are as varied as the range of systems and applications, performance measurement and analysis are prerequisites to performance optimization. The associated performance data presentation techniques have evolved from simple static graphics to include dynamic two-dimensional graphics and virtual environments. Each is best suited to different performance problems and ranges of user expertise.

Building on this thesis, in §2, we describe a range of performance visualization techniques, their usability implications, and their domains of applicability. In §3–§4, we describe two decidedly different performance analysis problems — the input/output dynamics of large, input/output intensive parallel applications and WWW access patterns and performance. Based on these experiences, in §5 we describe virtual environment representations of complex performance data and the lessons learned from this analysis. In §6 we describe related work. Finally, §7 summarizes our work and outlines plans for continued research.

*This work was supported in part by the Advanced Research Projects Agency under ARPA contracts DABT63-94-C-0049 (SIO Initiative), DAVT63-91-C-0029 and DABT63-93-C-0040, by the National Science Foundation under grants NSF IRI 92-12976, NSF ASC 92-12369, and NSF CDA 94-01124, and by the National Aeronautics and Space Administration under NASA Contracts NAG-1-613 and USRA 5555-22.

2 Performance Data Visualization

Although many performance analysis and visualization tools have been developed for scalable parallel systems, all form a few distinct equivalence classes, distinguished by their performance instrumentation and presentation techniques [16]. Within each class, these tools form a continuum of complexity, ranging from simple ones that show statistical profiles of program execution to more sophisticated tools that use graphics to correlate temporal and spatial patterns of application resource demands with system responses.

Underlying both simple and complex tools are a group of standard instrumentation techniques based on sampling (e.g., as used in `gprof`), counting, timing, and event tracing. Each technique strikes a different balance among implementation complexity, behavioral perturbation, and resulting performance data volume. For example, standard procedure profiling requires operating system support for program counter sampling, generates a volume of performance data that is independent of program execution time, and provides only a static snapshot of program behavior. However, profiling rarely perturbs execution enough to mask true bottlenecks. Conversely, timestamped event tracing requires high-resolution clocks and data extraction mechanisms, generates performance data volume proportional to program execution time, and permits detailed behavioral analysis. But, event tracing can greatly perturb execution if instrumentation is placed in frequently executed code.

2.1 User Expectations

Our growing experience base with software tools [22] suggests that simple, easy to use performance tools are used most frequently, with more complex, sophisticated tools used only to identify pernicious and subtle problems. This reflects user pragmatism — the proximate cause of most performance bottlenecks is obvious given a simple textual or graphical profile of computation, communication, and input/output operations. Only when the causes of poor performance are obscure (e.g., due to the interactions among multiple processors and software layers) do more powerful tools provide novel and important optimization insights. Simply put, users are loath to scale the learning curve of complex tools unless there are clear software performance rewards unattainable via simpler tools.

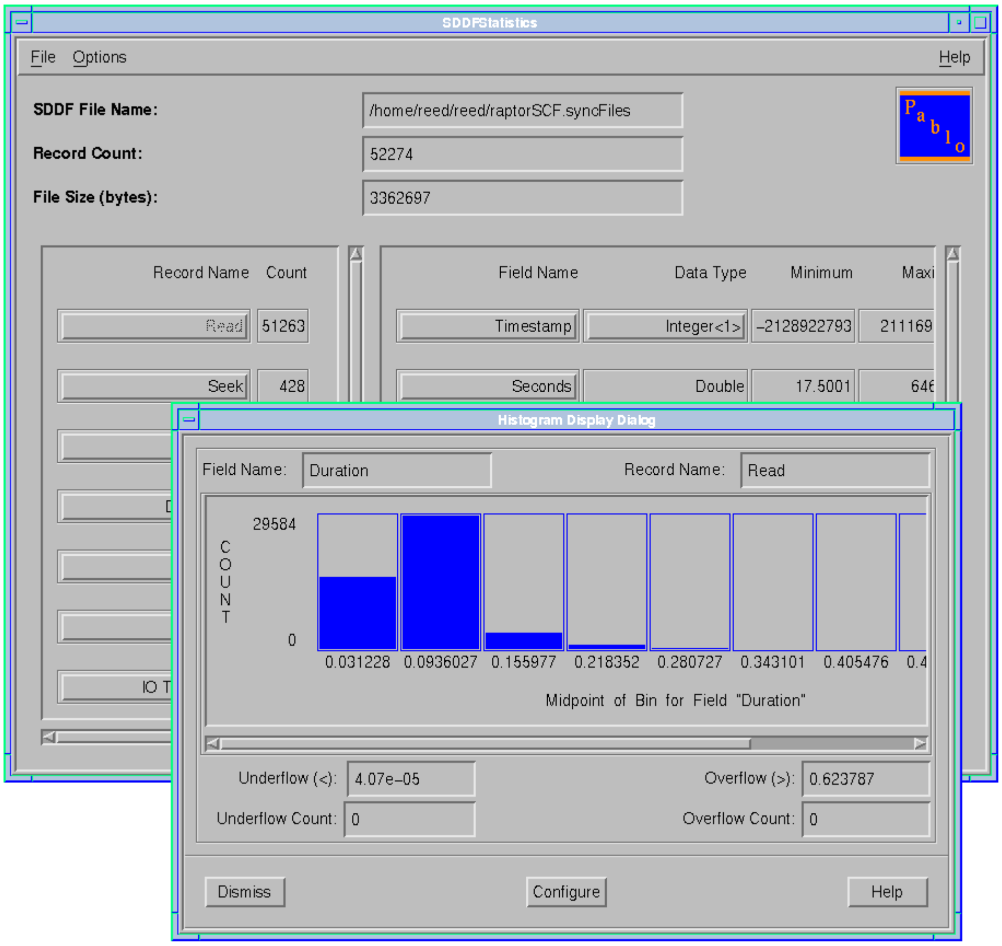


Figure 1: Pablo Static Graphics (Performance Data Histogram)

Given the diversity of user needs and the range of possible performance problems, understanding user expectations during design of user interfaces and data presentation mechanisms is critical to developing useful tools. Gaining user acceptance of more powerful and sophisticated performance analysis and presentation tools is predicated on creation of intuitive user interfaces that hide tool complexity while providing access to sophisticated performance analysis techniques.

Traditionally, graphical techniques for performance data presentation have relied on static graphics (e.g., line or bar plots). More recently, several tools have emerged that exploit dynamic graphics to illustrate temporal and spatial interactions among software and hardware components [10, 5, 20, 16]. Both static and dynamic graphics provide a WIMP (window, icon, menu, pointing device) interface with concomitant menu hierarchies for interaction. In contrast, virtual environments offer the yet unfulfilled promise of direct manipulation of performance data and software components, making the user a part of the software and its behavior [17, 18]. Below, we briefly describe each of these techniques and their domains of applicability.

2.2 Static Graphics

Static, two-dimensional graphics for performance data are based on a long and rich history of map making, plotting, and graphing techniques. This history is complemented by more recent, formal human-computer interaction (HCI) analysis of the effectiveness of data representation alternatives. Standard techniques like bar and line graphs, scatter plots, and pie charts have the advantage that they are well understood and widely used in other contexts. Moreover, because they are best used with a small number of data items, static graphics are ideal for displaying simple performance metrics and high-level snapshots of software and hardware behavior [14].

As an example of the value of simple graphic displays, Figure 1 shows a general purpose histogram display tool based on the Pablo performance analysis software [16]. The tool computes counts, minima, maxima, standard deviations, and histograms of the values of all fields in all records contained in a performance data file represented in the Pablo self-describing data format (SDDF) [20]. As shown in the rear window of Figure 1, users can click on a record to obtain

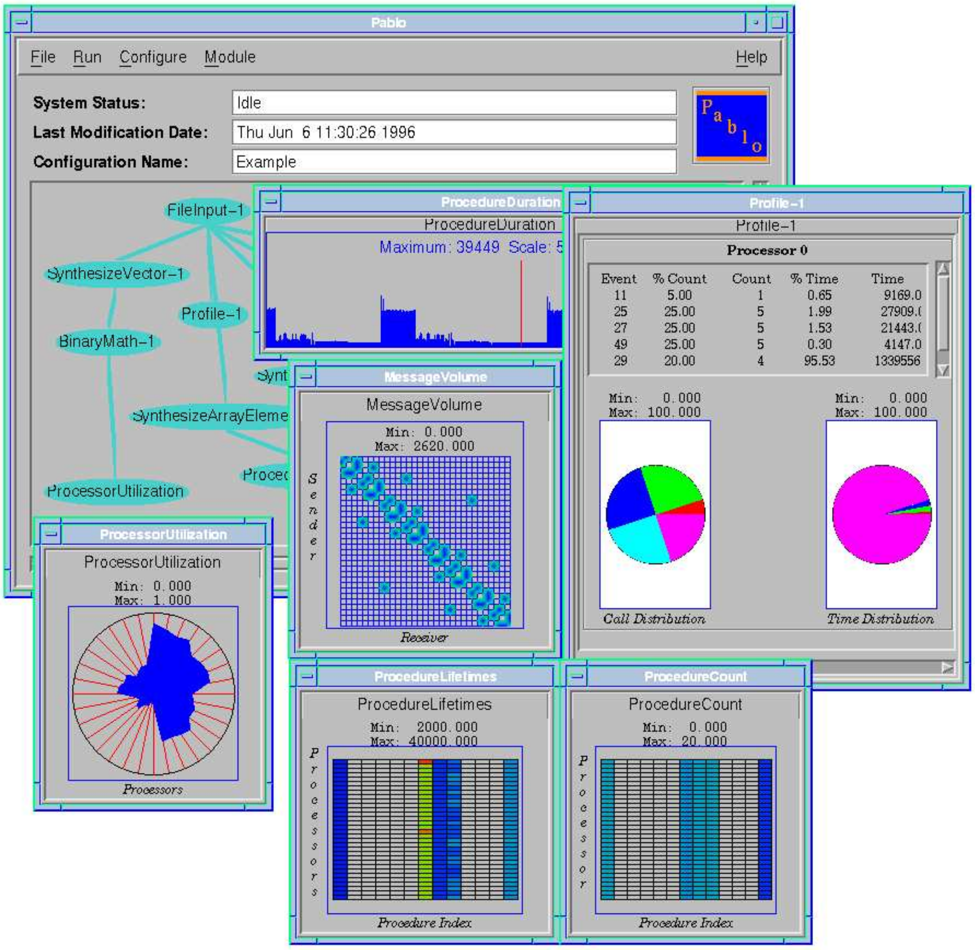


Figure 2: Pablo Dynamic Graphics (Behavioral Snapshot)

data on the fields in the record, and they can click on a field to obtain a histogram of the values associated with a record field. In the figure, the front window shows a user configurable histogram of durations of file read durations using static bar graphs.

Simplicity and generality are the primary advantages of a tool such as that in Figure 1. The tool supports a few readily grasped options, the interface is intuitive, it allows users to understand the distribution of data values, and it can process either profile data or event traces. Finally, because it computes a statistical profile of the input data, the tool can analyze performance data from diverse sources.

2.3 Dynamic Graphics

Unlike static representations, dynamic graphics allow one to observe temporal evolution (e.g., computation and communication phases) and component interactions (e.g., message passing patterns). However, most dynamic representations require more detailed performance data than static graphics. In consequence, most are based on timestamped event traces [20, 10]. Because the size of these traces grows rapidly with program execution time and with the number of processors, some dynamic graphics displays support real-time extraction and data display [16]; others limit trace size to that obtainable from test executions with small data sets. Although real-time display lessens secondary storage requirements by processing the traces as they are generated, it makes post-mortem analysis and data correlation difficult because one cannot review earlier behavior.

Figure 2 shows a snapshot of the Pablo dynamic graphics display of program execution on a thirty-two processor distributed memory system. All displays are driven by a timestamped trace of procedure calls and message transmissions. The specific displays were created using a coarse-grain data flow model similar to that found in scientific visualization systems like AVS.

Beginning at the top of Figure 2 and proceeding clockwise, the displays show an evolving strip chart of procedure durations across all processors, a dynamic, per processor profile of procedure call counts and durations, a matrix of procedure call counts and total execution time partitioned by procedure and processor, a Kiviat diagram of processor utilization computed by deducting communication delays from the execution time to the current point, and a contour plot of total data transmitted and received by each processor. As the event trace is processed, each display is updated to show the evolution of program execution.

The strength of a graphical tool like that in Figure 2 is its ability to show behavioral dynamics and to prototype new analyses via graphical data flow programming. Conversely, limitations of tools based on dynamic graphics include potentially steep learning curves, the need to identify important behaviors via visual pattern matching, and the mapping of behaviors to application code constructs and system features. In short, two-dimensional dynamic graphics lie at a user interface inflection point. When well designed, they can be powerful tools for understanding execution dynamics. Conversely, user interface complexity can limit their application to either domains where detailed performance optimization is critical (e.g., hard real-time systems) or where bottlenecks are due to software component interactions.

2.4 Virtual Reality

Although animated, two-dimensional graphical displays allow users to explore and understand the evolving relations among abstract, multivariate performance data (e.g., input/output rates, communication latencies, and computation intervals), they strongly separate the user from the parallel system and its behavior (i.e., one *watches* an animation of system execution on a workstation screen rather than being a part of the execution itself). Virtual environments promise to transform users from passive observers of system behavior to active, immersed participants who can directly manipulate software components and behavior. If successful, this direct manipulation of software systems would exploit our real-world abilities to react directly and reflexively based on sensory input. In this mode, users could interact directly with the executing software (e.g., measuring communication latency with a virtual "yardstick").

As an example, Figure 6 shows a "time tunnel" virtual environment perspective of the input/output traffic in an *ab initio* quantum chemistry application.[1] In the figure, each processor's behavior is represented by a timeline along the surface of a cylinder. Each timeline consists of line segments whose colors represent processor behavior during the associated time intervals. In the virtual environment, users can walk about and through the data, changing perspectives based on natural movements.

Data immersion is the primary strength of virtual environment representations for performance data; users can move *inside* the data, seeing and hearing interesting phenomena. This immersion provides a sense of realism and interaction not possible with workstation displays. Following a brief discussion of performance analysis problem domains that could benefit from virtual environment representations, we describe our experiences building virtual environments in §5.

[1] See §3 for additional details on the application.

3 I/O Characterization

There is growing realization that input/output is a major impediment to high performance for an important class of grand and national challenge scientific applications. Although much less is known about the interaction between parallel application input/output demands and file system policies than for other resources (e.g., message passing and task scheduling), preliminary data [6, 23] suggest that input/output performance is strongly sensitive to these interactions. Redressing input/output limitations is the goal of the national Scalable I/O (SIO) initiative.

The Scalable I/O (SIO) initiative [15] is a broad-based attack on the input/output problem that encompasses scientific application performance instrumentation and analysis [17], operating systems, and compiler research groups. The goal of the initiative is to exploit knowledge of application input/output access patterns to develop (a) application programming interfaces (APIs) that encapsulate common input/output access patterns, (b) parallel network and file system policies that can adapt to changing access patterns, and (c) compilation techniques for optimizing low-level input/output accesses from high-level specifications.

To design effective input/output policies, one must first understand time varying application stimuli and file system responses. In turn, this necessitates the capture and analysis of detailed input/output traces from representative applications.

3.1 Application Exemplars

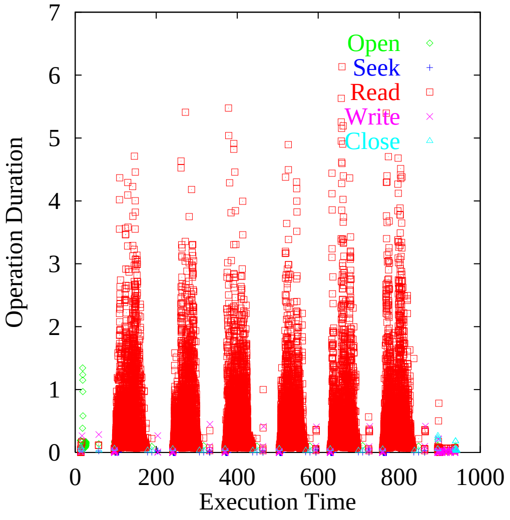
The complete SIO application suite spans a broad range of disciplines, including biology, chemistry, earth sciences, engineering, and physics; thus, space precludes a complete description of all codes and their input/output behavior. Instead, we focus on two quantum chemistry codes with strikingly different behavior.

The first, a Hartree Fock code (HF), computes nonrelativistic interactions among atomic nuclei, electrons in the presence of other electrons, and electrons interacting with nuclei. Atomic integrals, computed over basis functions, are used to approximate molecular densities using a self-consistent field (SCF) method. The second, a quantum chemical reaction dynamics (QCRD) code, computes chemical reaction rates and cross-sections of atoms and diatomic molecules. Both the HF and QCRD codes consist of multiple programs that process large files of numerical quadrature data.

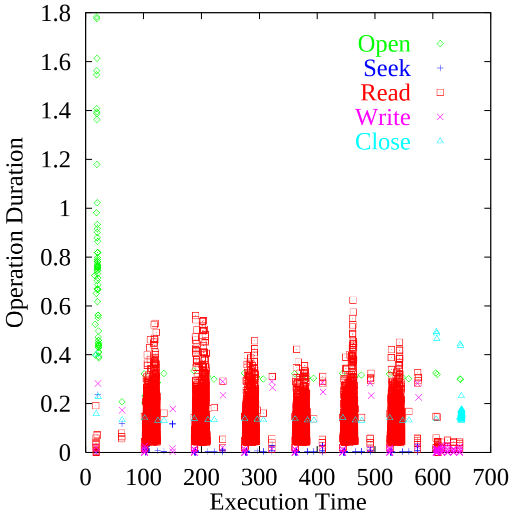
To capture input/output dynamics, we instrumented the codes using a version of the Pablo data capture library that supports unobtrusive logging and extraction of input/output traces [6, 17]. From these traces, one can generate temporal and spatial profiles of input/output activity, simple two-dimensional scatterplots, or immersive virtual environment representations of input/output dynamics.

3.2 Aggregate I/O Behavior

Figures 3–4 show scatter plots of the aggregate input/output patterns in the final phase of the HF code and an intermediate phase of the QCRD quantum chemistry application. The figures show input/output performance for two hardware configurations of the Intel Paragon XP/S, one with 16 I/O nodes, each with a slow RAID-3 disk array, and a second configuration



(a) 16 I/O nodes



(b) 80 local disks

Figure 3: HF SCF I/O Comparison

with 80 faster SCSI-2 disks. In both cases, data are taken from application executions on 64 processors.

As Figure 3 illustrates, the HF code has alternating phases of computation and intensive file reads for retrieval of quadrature data. The larger, faster input/output hardware configuration reduces both the dispersion of file read durations (the large spikes in Figure 3) and their mean values.

In contrast, the QCRD code of Figure 4 is dominated by file seeks and writes. The combination of small writes, data striping across all disks, and frequent metadata updates due to file seeks leads to poorer performance with a larger number of disks.

Although Figures 3–4 capture the gross patterns of input/output, they also illustrate the limitations of static two-dimensional plots. There is severe overplotting from the large number of data points, obscuring the location and distribution of less frequent input/output operations. Moreover, all identification of the operations associated with individual processors is lost, making it impossible to analyze temporal skews. The goal of virtual environment representations like those in Figures 5—7 is intuitive exploration of large masses of data across multiple scales.

4 WWW Characterization

The explosive growth of the WWW is well documented, though much less is known about the data types retrieved and their implications for network protocols and server caching strategies, particularly as the WWW evolves to support streaming audio and video [2]. Thus, understanding extant and evolving request patterns and the loads placed on both servers and networks is key to optimizing both server design and operating system resource management policies. To gain these insights, during the past two years, we have extensively analyzed the NCSA WWW server logs, both statistically [11] and graphically [12, 21].

4.1 Server Logs

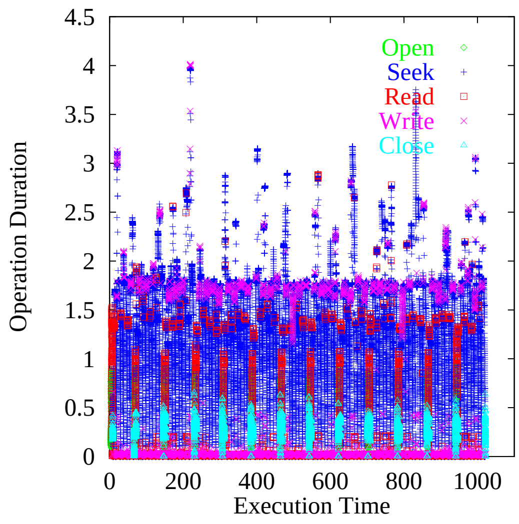
Most WWW servers maintain standard logs of each incoming request. These logs typically contain the IP address of the requesting client, the time of the request, the name of the requested document, and the number of bytes sent in response to the request. Ancillary logs maintained by NCSA contain one minute samples from the UNIX *netstat* and *vmstat* network and virtual memory monitoring facilities.

From the primary and ancillary server logs, we have computed two types of performance statistics. The first, one minute sliding window averages of 48 metrics for each server, includes network and memory traffic, processor utilization, request data types and sizes, and requesting network domains. The second, a geographic distribution, includes the latitude and longitude, city, domain, size, and type of each request. Both types of data are represented in the Pablo performance analysis environment's SDDF data format [20], enabling analysis with extant software.

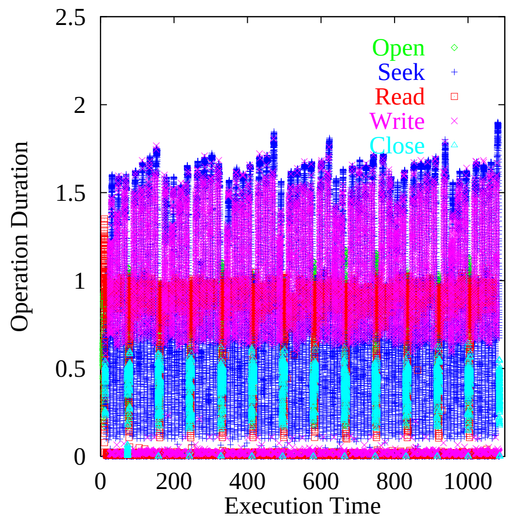
4.2 Data Analysis

Our statistical analysis of the processed server logs [11] has shown that commercial and government use of the server is growing rapidly, and request heterogeneity is increasing, lessening server cache hit ratios. In addition, the data clearly show that increasing use of non-text data (i.e., scientific data sets, imagery, audio, and streaming video) will require different approaches to WWW server design [2].

Data volume has proven to be the major limitation of statistical analysis; the NCSA WWW server generates over 150 megabytes/day of performance data. Understanding transient behavior (e.g., displacement of frequently accessed but small text items from the cache by large, but less frequently accessed video clips) is extraordinarily difficult without graphical representations. Likewise, analyzing the geographic distribu-



(a) 16 I/O nodes



(b) 80 local disks

Figure 4: QCRD I/O Comparison

tion of request types and their relation to population centers and network bandwidth is feasible only with geographic display techniques and virtual environment representations such as those in Figures 8—9.

5 Virtual Environment Experiences

In §2, we argued that virtual environments for direct data manipulation and immersion held the promise of realism and interaction not possible with workstation displays. By exploiting the Pablo environment's support for real-time data extraction, we have developed an immersive virtual environment, called Avatar [21, 19], for real-time, three-dimensional exploration of dynamic performance data.

We have successfully used Avatar to study the dynamics of input/output patterns in large parallel applications and to analyze the behavior of WWW servers. Interactions with application scientists have shown that virtual environment exploration is natural and intuitive. Moreover, the ability to walk and fly through the data, to examine it from multiple perspectives, and to change real-time display attributes interactively has provided insights not possible with two-dimensional static or dynamic graphics.

5.1 Infrastructure

Avatar operates with (a) a head-mounted display and tracker, (b) the CAVE [7] virtual reality theater, or (c) a workstation display with stereo glasses. The CAVE is a room-sized cube of high-resolution, rear-projection displays that allows users to walk about unencumbered by a head-mounted display. The lower resolution head-mounted display version of Avatar includes speech synthesis and recognition hardware for voice-directed commands, and both the head-mounted display and the CAVE versions use six degree of freedom trackers for head and hand (three-dimensional mouse) position location. All three versions of Avatar are driven by an SGI Power Onyx with RE2 graphics.

At present, Avatar supports three domain-independent display metaphors that accept real-time data from the Pablo software: a three-dimensional generalization of scatterplot matrices [19], a "time tunnel" view of processor interactions [17], and a geographic data display [12], all with data sonification [13]. Below, we briefly describe the application of each metaphor to input/output and WWW analysis.

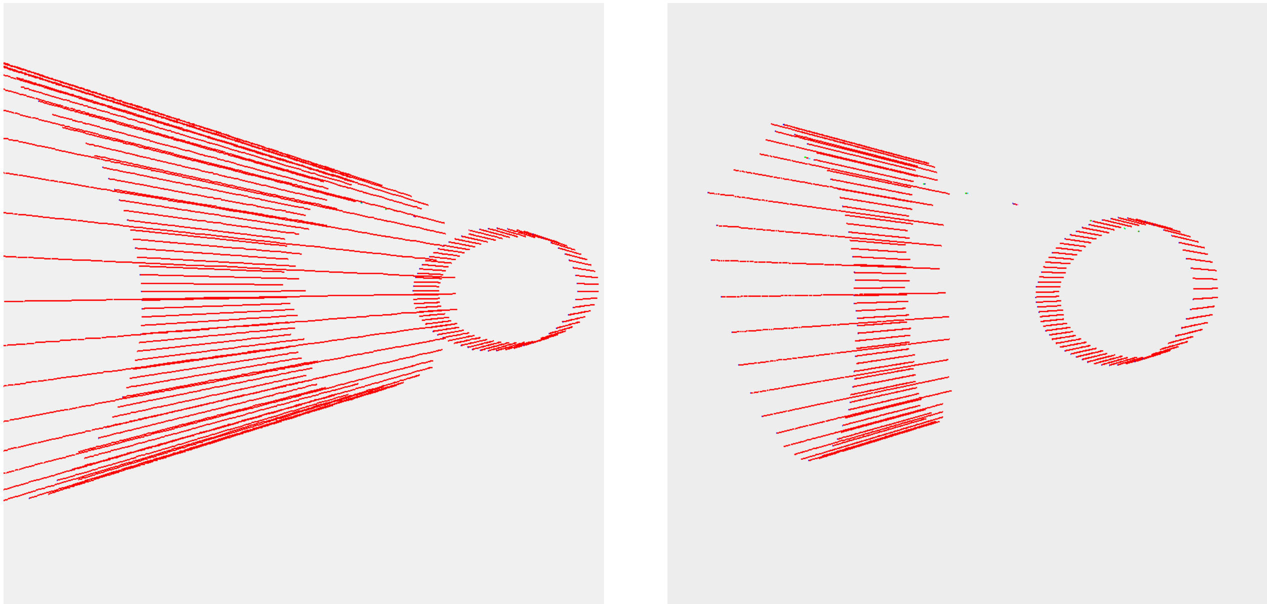
5.1.1 Time Tunnels

The Avatar time tunnel display metaphor captures the time evolutionary behavior of a parallel code via a display consisting of a cylinder whose major axis is time. Along the cylinder periphery, each line is composed of segments, where the color and length of each segment indicate the type and duration of each behavior in a parallel program. Cross-processor interactions (e.g., via message passing) are represented by chords that cut through the interior of the cylinder.

Figures 5–6 show side and end snapshots of the time tunnel for the input/output behavior of the HF and QCRD codes described in §3. In the figures, time increases as one moves forward or to the right from the viewer, with one foot corresponding to five seconds and 0.5 seconds of execution, respectively, for the HF and QCRD codes in the full size displays.

In Figure 5, file reads are the dominant activity, with two of the HF read phases visible.[2] File seeks occur just before each read burst on each processor, though these are not visible in this low resolution snapshot. Comparing Figures 5(a) and 5(b) shows the striking effect of the larger number of faster disks on the total duration of each read phase.

[2]Regrettably, much of the fine scale color detail visible in the virtual environment is lost in the static, gray scale images of Figures 5–9.



(a) 16 I/O nodes

(b) 80 local disks

Figure 5: HF I/O Time Tunnel View

Figure 6 shows time tunnel end views of input/output activity for the QCRD code. Both hardware configurations show a complex mix of file opens, reads, writes, and seeks, with severe resource contention manifest as a lack of regularity in operations across processors. Again, fine detail is lost in the gray scale figures.

5.1.2 Scattercubes

The time tunnel provides a behavioral view of activity, but no mechanism to correlative this activity with other application or system performance metrics. Scattercube matrices, a generalization of simple scatterplot matrices [3], allow one to study such metric correlations by showing all three-dimensional projections of the N-dimensional data. In each scattercube, the coordinate axes correspond to three of the N performance metrics, and the time-varying position of each processor in the scattercube is determined by the current values of the associated performance metrics.

Geometrically, the behaviors of the p processors define a set of p curves in an N-dimensional performance metric space, with each scattercube showing a different three-dimensional projection of this trajectory. To help analyze data point trajectories, one can interactively enable history ribbons to mark the movement of selected points and toggle display of the limits of multidimensional clusters via bounding, translucent volumes. Moreover, one can fly through the projections to explore the data space and interactively rescale the axes; sonification complements the visualization by mapping the centroid of the data in each scattercube to pitch.

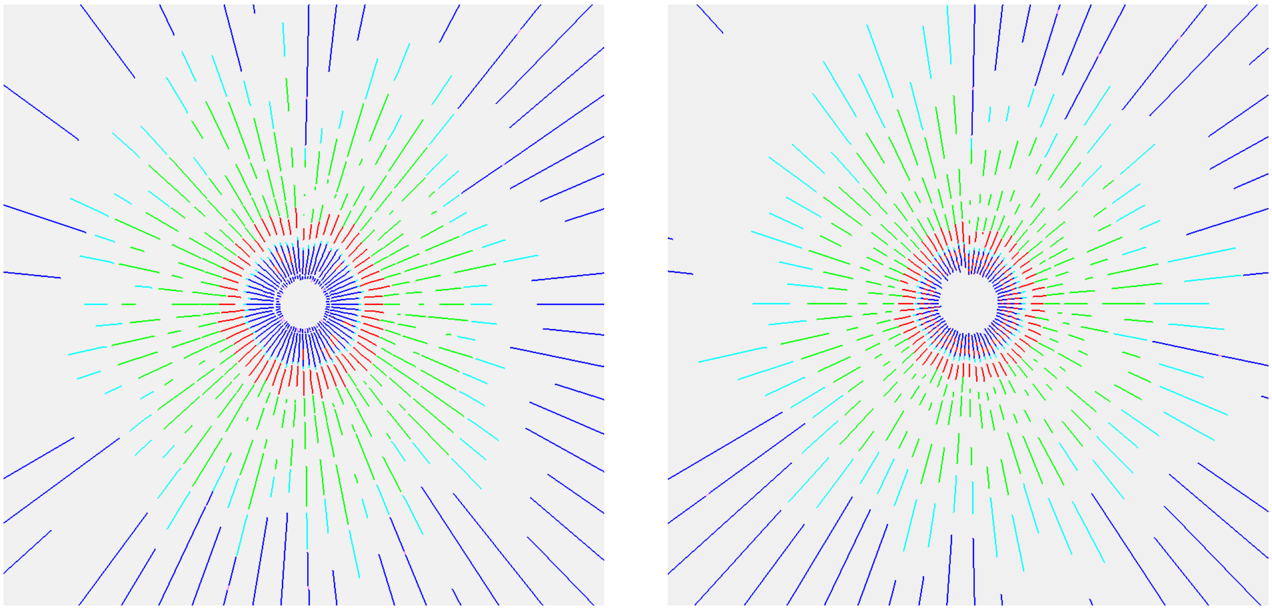
Figure 7 shows the interior of two scattercubes that contain input/output data from the HF and QCRD chemistry codes; other scattercubes are visible through the translucent walls. In each scattercube, the three axes are file seek, write, and read operations, with the current position of each processor in the metric space denoted by a blue (gray) octahedron, and the limits of each cluster marked by a purple (gray) bounding bubble. All data are five second sliding window averages of operation durations.

In Figure 7(a), the processors cluster along the read axis, reflecting the dominance of read operations in the HF code. Periodic seeks are manifest in the history ribbon trajectories along the left (seek) axis. Figure 7(b) shows the wide variability of operations and durations in the QCRD code, with a much larger bounding cluster volume.

Figure 8 illustrates application of the scattercube representation to a different domain — analysis of WWW data from NCSA [11]. In the figure, the three axes correspond to one minute sliding window averages of three of the 48 statistical performance metrics describing WWW request patterns and server behavior: the number of bytes of data transferred to satisfy requests for video clips, bytes transferred for text requests, and total number of requests. Each colored ribbon represents the trajectory of one NCSA WWW server in the metric space. The clustering of history ribbons in Figure 8 indicates that the round robin algorithm used by NCSA to balance the server load is very effective.

5.1.3 Geographic Displays

As we noted in §4, WWW performance data is amenable to both statistical and geographic displays.



(a) 16 I/O nodes (b) 80 local disks

Figure 6: QCRD I/O Time Tunnel View

The latter are particularly appealing when analyzing the distribution of WWW requests and types based on point of origin. To date we have relied on a "globe in space" view for global perspective. The globe consists of a texture map of the world on a sphere whose surface includes altitude relief and political boundaries. Illumination based on the local time of day provides temporal perspective. On the globe, geographic data are represented as stacked bars sited at appropriate locations, with bar height and color bands conveying data attributes. Rotation, together with the ability to walk around and inside the globe, allows one to explore data from a variety of perspectives.

Figure 9 illustrates use of the geographic display to explore the characteristics of requests to WWW servers [12]. Here, each bar is placed at the geographic origin of a WWW request, with the bar heights showing attributes of the requests from that location. The menus in the background of Figure 9 allow users to choose the types of data displayed. Options include display by server number, by Internet domain, by time since request, and by data type; each option can in turn be displayed based on the total number of requests or the aggregate number of bytes. In Figure 9, the bars represent the number of bytes of data requested from each location, partitioned by type of data (i.e., scientific, video, audio, image, text, or other).

Users can interact with the display of Figure 9 by walking around and inside the globe, rotating the globe along any axis, changing the types of displayed data, and using an interactive "dipstick." Via the dipstick, one can select and identify geographic point. After selection, the point's identity and the current numerical values of the data associated with that point are displayed.

6 Related Work

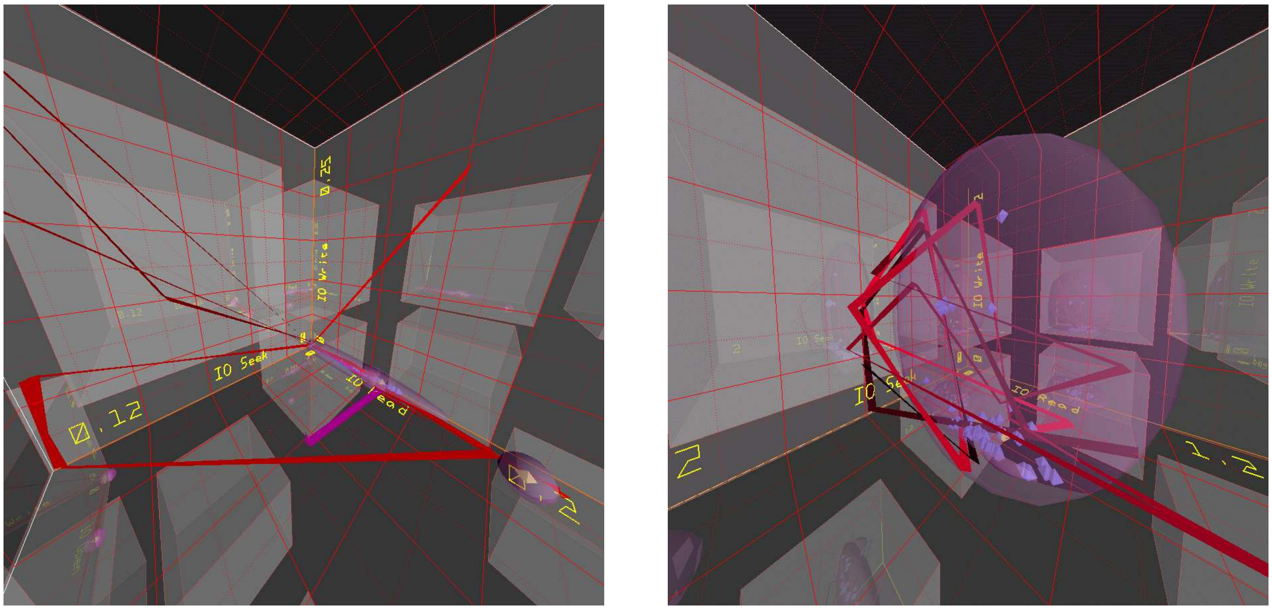
Many researchers are active in virtual reality and its applications to computational science and information visualization. Notable infrastructure efforts include Pausch's development of a toolkit for rapid prototyping of virtual environments [4] and the NCSA/EVL CAVE and its libraries [7]. These toolkits and libraries hide the low-level details of hardware interactions (e.g., tracker polling), reducing the time needed to develop new virtual reality applications like our Avatar environment.

Several groups have developed virtual environment representations of specific problem areas, including interactive computational fluid dynamics [1], molecular docking [9], and financial data visualization [8]. Avatar builds on the interaction lessons gleaned from these environments and from the rich history of statistical graphics [3].

7 Summary and Futures

Performance data presentation techniques have evolved from simple text-based summaries to complex, virtual environments. Our experience with the Pablo performance analysis environment and its progeny, the Avatar virtual environment for performance data analysis, suggests that virtual environments hold great promise for eliminating the artificial separation of software developers from software dynamics and making software a tangible, manipulable entity.

By immersing users in real-time performance data and representations of software and system dynamics, well-constructed virtual environments exploit our real-world spatial, haptic, and kinematic skills. This exploitation makes virtual environment manipulation



(a) HF

(b) QCRD

Figure 7: Input/Output Scattercube View (80 local disks)

intuitive and can allow users to manipulate software structure and behavior directly.

Despite progress, many open questions remain, including development of virtual environment tools for measurement, annotation, and collaborative exploration. In the coming months, we plan to build on the Avatar experience to develop a new virtual environment for direct manipulation of software structure. This environment will allow users to modify the behavior of executing software within the virtual environment and see the effects of those modifications in real time.

Acknowledgments

Ruth Aydt, Dave Kohr, Roger Noe, and Bradley Schwartz all helped design and implement the Pablo performance analysis software. Likewise, Will Scullin, Steve Lamm, Keith Shields, and Luis Tavera made the vision of a virtual environment for performance data immersion and control a reality.

References

[1] Bryson, S. The Virtual Wind Tunnel: A High-Performance Virtual Reality Application. In *Proceedings of IEEE 1993 Virtual Reality Annual International Symposium (VRAIS '93)* (1993), pp. 20–26.

[2] Chen, Z., Tan, S.-M., Campbell, R. H., and Li, Y. Real Time Video and Audio in the World Wide Web. In *Fourth International World Wide Web Conference* (Boston, MA, December 1995), pp. 333–348.

[3] Cleveland, W. S., and MiGill, M. E., Eds. *Dynamic Graphics for Statistics.* Wadsworth & Brooks/Cole, 1988.

[4] Conway, M., Pausch, R., Gossweiler, R., and Burnette, T. Alice: A Rapid Prototyping System for Building Virtual Environments. In *Conference Companion, CHI '94* (Apr. 1994), pp. 295–296.

[5] Couch, A. L., and Krumme, D. W. Monitoring Parallel Executions in Real Time. In *Proceedings of the Fifth Distributed Memory Computing Conference* (Apr. 1990), pp. 1187–1206.

[6] Crandall, P. E., Aydt, R. A., Chien, A. A., and Reed, D. A. Characterization of a Suite of Input/Output Intensive Applications. In *Proceedings of Supercomputing '95* (Dec. 1995).

[7] Cruz-Neira, C., D.J.Sandin, and DeFanti, T. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *SIGGRAPH '93 Proceedings* (Aug. 1993), Association for Computing Machinery.

[8] Feiner, S., and Beshers, C. Visualizing n-Dimensional Virtual Worlds with n-Vision. In *ACM SIGGRAPH Computer Graphics* (Mar. 1990), vol. 24, pp. 37–39.

[9] Fredrick P. Brooks, J., Ouh-Young, M., Batter, J. J., and Kilpatrick, P. K. Project GROPE — Haptic Displays for Scientific Visualization. In *ACM Computer Graphics* (Aug. 1990), vol. 24.

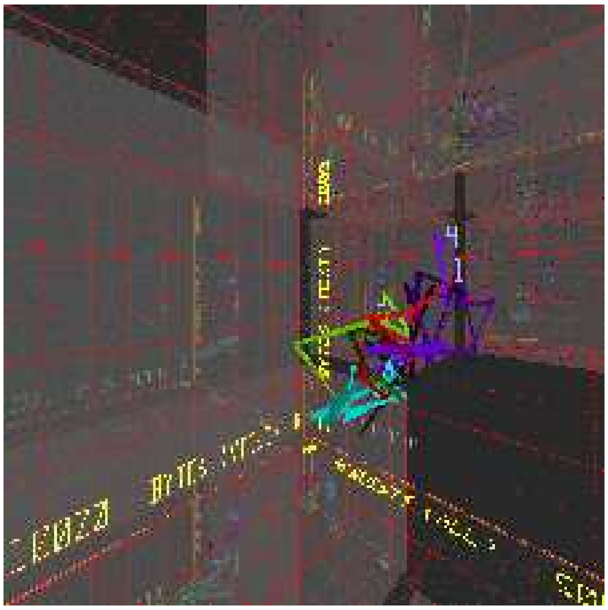


Figure 8: WWW Server Metrics (Scattercube View)

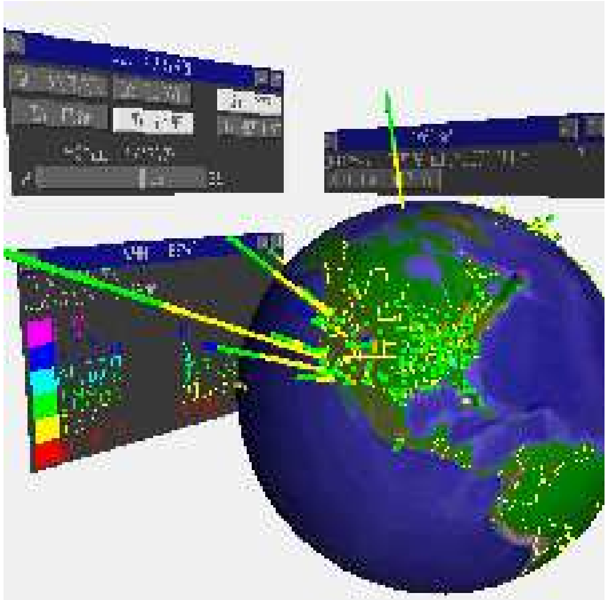


Figure 9: WWW Traffic (Geographic View)

[10] Heath, M. T., and Etheridge, J. A. Visualizing the Performance of Parallel Programs. *IEEE Software* (Sept. 1991), 29–39.

[11] Kwan, T. T., McGrath, R. E., and Reed, D. A. NCSA's World Wide Web Server: Design and Performance. *IEEE Computer* (Nov. 1995), 68–74.

[12] Lamm, S. E., Scullin, W. H., and Reed, D. A. Real-time Geographic Visualization of World Wide Web Traffic. In *Proceedings of the Fifth International World Wide Web Conference* (May 1996).

[13] Madhyastha, T. M., and Reed, D. A. A Framework for Sonification Design. In *Auditory Display: Sonification, Audification and Auditory Interfaces* (1992), Addison-Wesley.

[14] Miller, B. P., Callaghan, M. D., Cargille, J. M., Hollingsworth, J. K., Irwin, R. B., Karavanic, K. L., and Newhall, T. The Paradyn Parallel Performance Measurement Tools. In *IEEE Computer* (Nov. 1995), vol. 28.

[15] Poole, J. T. Scalable I/O Initiative. California Institute of Technology, Available at `http://www.ccsf.caltech.edu/SIO/`, 1996.

[16] Reed, D. A. Experimental Performance Analysis of Parallel Systems: Techniques and Open Problems. In *Proceedings of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (May 1994), pp. 25–51.

[17] Reed, D. A., Elford, C. L., Madhyastha, T., Scullin, W. H., Aydt, R. A., and Smirni, E. I/O, Performance Analysis, and Performance Data Immersion. In *Proceedings of MASCOTS '96* (Feb. 1996), pp. 1–12.

[18] Reed, D. A., Elford, C. L., Madhyastha, T. M., Smirni, E., and Lamm, S. E. The Next Frontier: Interactive and Closed Loop Performance Steering. In *Proceedings of the 1996 ICPP Workshop on Challenges for Parallel Processing* (Aug. 1996).

[19] Reed, D. A., Shields, K. A., Tavera, L. F., Scullin, W. H., and Elford, C. L. Virtual Reality and Parallel Systems Performance Analysis. *IEEE Computer* (Nov. 1995), 57–67.

[20] Ries, B., Anderson, R., Auld, W., Breazeal, D., Callaghan, K., Richards, E., and Smith, W. The Paragon Performance Monitoring Environment. In *Proceedings of Supercomputing '93* (Nov. 1993), pp. 850–859.

[21] Scullin, W. H., Kwan, T. T., and Reed, D. A. Real-time Visualization of NCSA's World Wide Web Data. In *Symposium on Visualizing Time-Varying Data* (Sept. 1995).

[22] Simmons, M. L., Hayes, A. H., Brown, J. J., and Reed, D. A., Eds. *Debugging and Performance Tuning for Parallel Computing Systems.* IEEE Computer Society Press, 1996.

[23] Smirni, E., Aydt, R. A., Chien, A. A., and Reed, D. A. I/O Requirements of Scientific Applications: An Evolutionary View. In *Proceedings of the Fifth IEEE Symposium on High-Performance Distributed Computing* (1996).