

VISFILES

Real-Time Immersive Performance Visualization and Steering



Bill Hibbard
 Space Science and Engineering Center
 University of Wisconsin - Madison

May 00 Columns



Visfiles



My productivity as a programmer has increased by an order of magnitude over the last 30 years, due to improved programming tools. Visualization has a big role to play in future improvements, so I have asked Dan Reed and Eric Shaffer to describe their work with the Virtue system for visualizing distributed programs.

- Bill Hibbard

Eric Shaffer

University of Illinois at Urbana-Champaign
Daniel A. Reed

University of Illinois at Urbana-Champaign

Introduction

The history of software systems is one of evolving complexity. In the latest inflection point, we have seen high-performance computing become distributed, with multidisciplinary applications now running in parallel on heterogeneous systems connected via high-speed, wide-area networks. For example, real-time analysis of radio astronomy data requires capture from remote sites, network transmission, near real-time data convolution and remote instrument control. Tuning the behavior of such distributed applications requires deep understanding of network, parallel system and application behavior.

Although significant research projects [2, 6] have already applied visualization to understanding software behavior, this work has focused on either single processor or homogeneous parallel systems. The added complexity of a networked environment dictates a new breed of visualization tools.

These new tools must support effective visualization software and hardware components that are much more distributed and whose

interactions and performance problems are more subtle and complex. This requires coherent display of data from multiple sources and software levels, as well as intuitive interfaces for interacting with the visualization.

Most importantly, users must be able to steer the application on-line and in real-time. Many modern distributed applications have extremely long execution times or are mission critical, with minimal down time allowed. In either case, it is not feasible to analyze the application performance off-line, make changes to the application and start over. Instead, the visualization tool itself supports on-line steering of the application.

To meet these needs, we have produced a prototype system that integrates collaborative, immersive performance visualization with real-time adaptive control of applications. The Virtue visualization system strives to make the abstract world of software tangible by using three-dimensional data displays and virtual reality technology.

From Immersive Environments to Palmtop Displays

Immersive virtual reality (VR) display systems such as the CAVE [3], driven by high-performance graphics systems (e.g., like the SGI InfiniteReality system), have been widely used to display and analyze complex scientific data. However, they have equal promise as display and analysis systems for information visualization. With the requisite display metaphors, the virtual environment affords one to interact with representations of abstract data just as one might interact with objects from the everyday world.

Not everyone has access to an immersive virtual environment (yet). Moreover, performance optimization of distributed, multidisciplinary applications often requires a team of analysts who are themselves physically separated with disparate computing systems and network connections. Hence, pragmatically, one must support alternate display devices. Therefore, Virtue also executes on SGI and Linux desktop systems and can export graphics and audio across the multicast backbone (MBONE) via an interface to the VIC and VAT videoconferencing tools. This allows palmtop and wearable devices to receive low-resolution Virtue video in real-time via wireless networks.

Visualizing Software Structure and Behavior

Finding intuitive mappings from abstract performance data to physical representations is critical for effective visualization. This task is made more difficult by the diverse objects of interest in performance analysis. From the structure of specific software

modules, to hardware performance counters on a specific platform, the components of the visualization vary widely across systems. Consequently, the most intuitive mappings are likely to be domain dependent and discovered only via trial and error. Virtue accommodates this by providing a language for quickly specifying and exploring data mappings.

The central component of Virtue is a toolkit for the visualization of hierarchical, three-dimensional graphs. Our experience has been that communication network components, application execution and software structure are particularly amenable to graph-theoretic modeling. The use of hierarchy helps organize the performance metrics and cull data from the display when it is not currently of interest.

To construct a visualization, a user creates a text file describing a set of graph structures and associating a particular graph layout with each. Virtue provides a suite of layout algorithms, including cone trees, geographic layouts and a ball and spring data-driven layout. This file also contains a set of data mappings, allowing one to associate data with various visual or sonic attributes of graphs. When Virtue processes data, either from performance measurement files or via real-time performance data streams from network sockets, the attributes of vertices and edges will change based on these mappings.

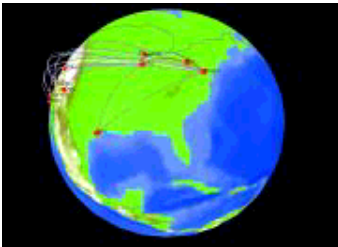


Figure 1: Geographic network display.

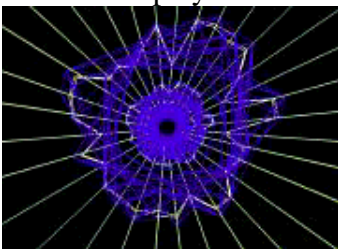


Figure 2: Time tunnel graph layout display

Graph Layout

As an illustration of Virtue's generic data mapping capability, we present some images captured from a performance visualization of a large parallel application. The visualization demonstrates a three-tiered hierarchy of network activity, processor-level execution metrics and software structure.

As shown in Figure 1, a geographic network display allows graph vertices placed at specific locations on a map, in this case a globe. The appearance of the links can encode specific performance metrics. In this example, color is related to communication latency. Selecting a specific site allows a user to "drill down" to a process view of performance data from a system at a specific site.

At the selected site, parallel execution on a single system can be presented as a time tunnel. In this graph layout, the timeline of activities in each task is color coded and laid out along the periphery

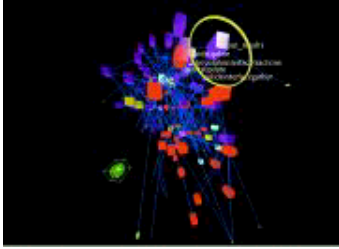


Figure 3: Call graph, where vertex color corresponds to time spent executing that code, and vertex size represents the number of invocations.

activities in each task is color coded and laid out along the periphery of a cylinder. Chords connecting the timelines in the cylinder interior denote intertask communication. Figure 2 illustrates both types of behavior. At this level, one can select a specific task and drill down further to examine the structure of the software executing on that task

This final level creates a call graph view of the code, where graph vertices represent specific code functions and links represent the static pattern of function calls. The vertical placement of nodes is used to indicate the direction of the call. Like all Virtue graphs, this display can be augmented with other data. Figure 3 shows a call graph where vertex color corresponds to time spent executing that code, and vertex size represents the number of invocations.

User Interfaces

Immersive virtual environments present unique demands on user interface design. Typing and 2D mouse interaction, the standard mechanisms for desktop software control, must be replaced by methods that allow users to move freely and select objects in a three-dimensional space.

Virtue uses off-the-shelf speech recognition software to support voice com-mands. These com-mands allow users to perform common functions such as navigating the graph hierarchy and pausing graph animation of performance data streams. Complementing voice commands, selection is typically accomplished via a 3D "mouse" that uses ray shooting to pick objects. Once an object is selected, the mouse can be used to move or otherwise activate it.

Virtue also supports the Virtual Technologies CyberGlove[®] as an alternative to the mouse. In addition to supporting more natural selection mechanisms (such as simply grabbing an object), the CyberGlove provides tactile feedback through micro-vibrators when interacting with objects. Both the 3D mouse and CyberGlove can be used to issue commands via a simple glyph-drawing interface. This can be quite useful in noisy environments where accurate speech recognition is difficult.

To aid data exploration, Virtue includes a set of virtual tools for manipulating and interrogating visual data representations. In keeping with our model of using intuitive representations, all the tools are virtual analogues of common real-world objects. The Magic Lens [1] seen in Figure 3, appears as a magnifying glass. The lens allows a user to see hidden data that is associated with specific objects. In the context of Virtue's call graph visualization, the lens displays text labels that associate a source code file name with a graph node. Other tools include a cutting plane that computes metrics on data associated with a set of bisected graph edges (e.g., computing

a maximum network latency given a set of latencies from the bisected network links) and a dipstick for displaying the exact data values associated with a selected object.

Supporting Teamwork

Virtue exports streaming video of the immersive visualization to remote collaborators via the MBONE, along with a Java interface for remote control of an immersive visualization. However, collaboration frequently requires both asynchronous and distributed interaction among team members.

For asynchronous interaction, Virtue supports multimedia annotations. By selecting a Virtue object and issuing a command, users can attach an audio and video record of their insights about the displayed data. Visually, annotations are represented by wireframe bounding boxes that enclose the object in question. An example annotation is visible in Figure 3.

The annotation system is based on modified versions of the VIC and VAT videoconferencing tools. When a record command is issued, these tools generate RTP (Real-time Transfer Protocol) audio and video files that are stored within an annotation server's database. When the annotation is opened for replay, the audio and video is multicast across MBONE. Virtue displays received video through windows floating in the virtual environment.

The same mechanisms used in the receipt of multimedia annotation streams allow Virtue to support live multicast audio and video streams. This enables videoconferencing between the immersive display and remote desktop or palmtop devices.

Real-Time Data Acquisition and Steering

Optimizing distributed applications requires capture of performance data from multiple sites and networks, diverse computer systems and software components written in a variety of languages. Because these distributed applications execute in a network-connected environment, with shared resources, their execution context is rarely repeatable. Hence, off-line performance analysis may not enable one to optimize the application for future executions in new contexts. Ideally, performance analysis software should allow users to alter runtime parameters and resource allocations in response to changing conditions during application execution.

As Figure 4 shows, Virtue relies on the Autopilot toolkit [4] for distributed measurement of executing software. Autopilot provides a set of low-overhead sensors that can be inserted in application code

or into performance daemons that periodically gather hardware and software performance data and that can be inserted via the SvPablo [5] toolkit. In real-time performance visualization, sensors directly report their values to Virtue for real-time display.

Much of the power of Virtue derives from the ability to perform closed-loop steering of applications. Control of application code is via Autopilot actuators. These routines are inserted into application software just like sensors, but as Figure 4 shows, they alter application or system parameters. Virtue provides a set of virtual controls, 3D sliders and buttons, forming an interface to these actuators.

Real-time steering of an application is made as simple as pushing a button to reset a buffer size, or sliding a lever to alter a variable. The ability to make adjustments on-line rather than off-line can result in tremendous timesavings when optimizing long running applications.

Future Challenges

Although immersive environments typically use large displays, most current systems have resolution no higher than that available on a desktop. Our experiences with Virtue suggest that large-scale performance visualization would benefit from higher resolution displays that fill the user's field of view. Such displays would allow multiple, linked information structures to be viewed simultaneously.

We have just completed construction on an 11'x24' display wall powered by 18 LCD projectors, similar to Li's scalable display wall [7]. The projectors are driven by a cluster of 18 Pentium III class PCs running Linux, with a Myrinet interconnection network. Moving Virtue to the display wall opens a rich set of research topics. We plan to explore new information presentation techniques that allow us to exploit the increased screen space, as well as what graphics performance gains can be made by exploiting the distributed rendering environment. Finally, we are interested in how large-scale displays can best support effective collaboration within a prototype "smart space," an active office environment made possible through pervasive computing.

References

1. Bier, Eric A., Maureen C. Stone, Ken Pier, William Buxton and Tony D. DeRose. "Toolglass and Magic Lenses: The See-Through Interface," SIGGRAPH 1993 Conference Proceedings, pp. 73-80, 1993.
2. Calzarossa, M. et al. "Medea: A Tool for Workload Characterization of Parallel Systems," IEEE Parallel and

Ken Brodlie, Stuart
Lovegrove and Jason
Wood
School of Computer
Studies
University of Leeds
Leeds, UK, LS2 9JT

Bill Hibbard
Space Science and
Engineering Center
1225 W. Dayton Street
Madison, WI 53706

Tel: +1-608-253-4427
Fax: +1-608-263-6738
Website

- Distributed Technology, pp. 72-80, winter 1995.
3. Cruz-Neira C., D Sandin and T. DeFanti. "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," SIGGRAPH 1993 Conference Proceedings, pp. 35-42, 1993.
 4. DeRose, L. et al. "An Approach to Immersive Performance Visualization of Parallel and Wide-Area Distributed Applications," Proceedings of the 8th IEEE International Symposium on High-Performance Distributed Computing, IEEE CS Press, Los Alamitos, CA, pp. 247-254, 1999.
 5. DeRose, L., Y Zhang and D. Reed, "SvPablo: A Multi-Language Performance Analysis System," Computer Performance Evaluation Modeling Techniques and Tools, Lecture Notes in Computer Science, Vol. 1,469, R. Puigjaner, N. Savino and B. Serra, eds., Springer-Verlag, New York, pp. 352-355, 1998.
 6. Heath, M.T. and J.A. Etheridge. "Visualizing the Performance of Parallel Programs," IEEE Software, pp. 29-39, September 1991.
 7. Samanta, Rudro, Jiannan Zheng, Thomas Funkhouser, Kai Li and Jaswinder Pal Singh. "Load Balancing for Multi-Projector Rendering Systems," SIGGRAPH/Eurographics Workshop on Graphics Hardware, August 1999.

The copyright of articles
and images printed
remains with the author
unless otherwise
indicated.

Bill Hibbard's research interests are interaction techniques, data models and distributed architectures for numerical visualization. He leads the SSEC Visualization Project and is primary author of the Vis5D and VisAD systems. He has degrees in mathematics and computer science from the University of Wisconsin - Madison.