Visualization Research with Large Displays

Bin Wei, Claudio Silva, Eleftherios Koutsofios, Shankar Krishnan, and Stephen North
AT&T Shannon Laboratory

We describe our research at AT&T InfoLab on using large displays to interactively analyze and visualize AT&T's communication networks and services.

Telecommunication networks and services generate tens of gigabytes of data every day. These data record various characteristics of the networks. Visualizing massive data at full scale and at different levels of abstraction provides an effective means of analyzing network, customer, and service behaviors. Interactive visualization tools and high-resolution displays are crucial in presenting this information properly.

Figure 1 is a photograph of a large display wall at the AT&T Global Network Operations Center in Bedminster, New Jersey. Data from the networks are continuously received, processed, and displayed on the wall. The large wall provides the required real estate to display various characteristics of the data coming from different parts of the network.

At the AT&T InfoLab, we've been investigating various visualization techniques for managing and displaying network data. For our experiments, we built a rear-projected display wall, or InfoWall, consisting of a 4 × 2 LCD projector system with a resolution of more than 10 million pixels. We initially configured the InfoWall to be driven by an SGI Onyx with two InfiniteReality[1] graphics pipes. However, to provide a more affordable, scalable architecture, our research currently focuses on building displays using a cluster of PCs.

Applications and software tools

Our experiments focus on visualizing massive data sets from AT&T's networks and services. These networks include long-distance voice networks, multiservice asynchronous transfer mode (ATM) networks, Internet protocol (IP) networks, access and service networks, and wireless and broadband cable TV networks. All these networks generate massive usage data sets. For example, the long-distance network alone issues 300 million call events per day. To generate any kind of real-time visual display of a network requires collecting and processing all this data as quickly as possible. We developed a toolset, called Swift-3D,[2] which provides comprehensive support for data exploration, integrating large-scale data visualization with querying, browsing, and statistical evaluation. It can collect data from many different sources and process the data in real time. Data processing proceeds incrementally and the visualization tools can safely access the data files while they are being updated. To minimize transmission and storage requirements, Swift-3D can compress massive data sets two or three times more than state-of-the-art general-purpose compressors.

All the input data of various sources get converted to a self-describing format, which consists of

1 The large display wall at the AT&T Global Network Operations Center.

0272-1716/00/$10.00 © 2000 IEEE

data schema in its header followed by the actual data. This lets most of the Swift-3D tools remain data independent. The most commonly used tools in Swift-3D are

- filtering,
- aggregation,
- splitting, and
- sorting.

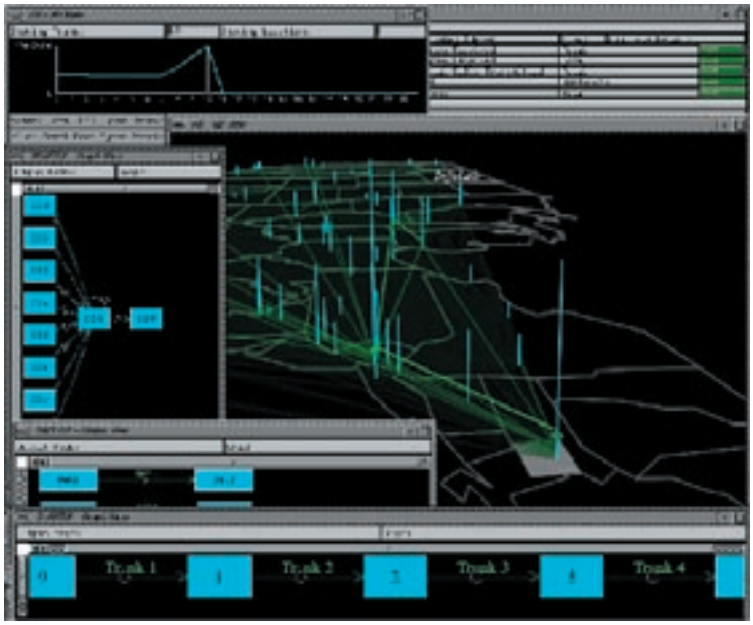
According to the network instantiation and application requirements, these tools can be selectively used to construct data processing pipelines that operate on the data quickly and efficiently and avoid saving temporary data to files. In addition to efficient algorithms, we also use several techniques to make these pipelines run as fast as possible: just-in-time compilation, direct I/O, memory mapping, and multireader pipes. These pipelines can operate incrementally, since we receive the data in real time every day. We use the data these pipelines generate for analysis and visualization.

A critical decision in visualization is to select appropriate visual metaphors for abstract entities such as networks. Although some aspects of these networks have certain physical representations (such as trunks of a network), many interesting concepts are completely virtual, such as a phone call in a voice network or a virtual circuit in a data network. Visualizing abstract entities is hard to do well, but at the same time, not having to simulate specific physical objects (such as an automobile or an airplane) allows more freedom in coming up with new visual representations.

We experimented with various visual metaphors, ranging from abstract to physical, and found that geography is one of the few physical representations that prove useful in network representations. Figure 2 shows a photograph of the InfoWall. The left half of the display shows a representation of a voice network. The various bars on top of the map represent the volume of calls out of each location. This representation makes it clear where most of the volume comes from—big cities. Many abnormal events on networks also cluster around specific locations. Using the tools in Figure 2, we discovered that many calls out of New York City failed because of busy signals (by observing a large concentration of bars around New York). This occurred because a NYC radio station had a mass call-in from its listeners.

Texture maps, such as the one shown on the right half of the display in Figure 2, can also be used to provide additional geographic clues such as cities, counties, mountains, and lakes.

2 A photograph of the InfoWall showing a representation of a voice network (left) and a texture map of the United States (right).



3 A Swift-3D display of a large-packet data network.

Figure 3 shows a large packet-data network. On the geographic map, the green lines between locations correspond to collections of virtual circuits. Their opacity and intensity correspond to the volume of traffic. In the abstract graph view on the left, we see a layout of the circuits in the selected area drawn as an undirected graph. Queries can be performed through the graphical user interface to explore the connectivity of the various elements, as shown in the two graph views on the bottom.

Showing these large networks in full detail and through multiple views effectively requires displays with a large amount of pixels (resolution). We designed Swift-3D for these large displays. The physical size of the wall plays an important role too. It allows a group of people to collaborate better by looking at a single display. We soon realized that using a conventional mouse for input seems difficult and unintuitive. We're currently exploring 3D pointers and gesture recognition as alternatives.

A goal shared by several research groups is to move from a rear-projection InfoWall driven by an SGI graphics server to a more affordable, scalable architecture using clustered PCs and commercially available tiled flat-panel displays. Some features of SGI graphics

To leverage the cost and performance advantages of commodity PCs, our work focuses on using PC clusters as an alternative architecture for driving high-resolution scalable displays. We mainly use Linux as the underlying operating system.

servers that we rely on aren't yet easily replicated on clustered PCs, even using a special-purpose 2D array of frame buffers (see http://www.dexonsystems.com).

Building scalable displays

We're interested in efficient and cost-effective ways to build the software and hardware for high-resolution scalable displays, such as those shown in Figure 1. Here we present our approach based on a cluster of PCs.

One potential solution is to use a high-end SGI Onyx2. These machines can be configured with a large number of rendering pipes, several CPUs, and lots of memory. Irix supports a shared-memory programming model on these machines, which makes it possible to develop scalable Iris Performer applications. However, this approach has a few drawbacks. First, the prices on SGI Onyx2 start very high, making it hard to develop cost-effective, smaller configurations. For very large displays, we get into a different problem—there is a hard ceiling on the number of InfiniteReality pipes that can go into one Onyx2. Another problem is the fact that the R12K processors currently offered on these machines lag behind the performance of other CPUs, such as the Intel Pentium III and AMD Athlon, which run at over 1 GHz. The same performance trend also appears on the graphics subsystem, where Nvidia graphics chips (see http://www.nvidia.com) approach InfiniteReality performance at a fraction of the cost.[3] Thus an Onyx2 solution has the problems of scalability (both cost and display size) and performance (including CPU and graphics).

To leverage the cost and performance advantages of commodity PCs, our work focuses on using PC clusters as an alternative architecture for driving scalable displays. The basic hardware components of such a system include

- *Workstation-class PCs*. For our purposes, a workstation PC consists of a machine with support for accelerated graphics port (AGP) and peripheral component interconnect (PCI) boards, high-bandwidth memory, one or more high-end Intel Pentium IIIs or AMD Athlons, and a high-end graphics card (such as Nvidia GeForce).
- *Fast interconnect network*. For instance, Giganet's cLAN is a good example of a fast network. The Giganet cLAN runs at over 1.25 Gbps, and since it uses the Virtual Interface Architecture (VIA; see http://www.viarch.org) as the communication protocol, it can bypass TCP/IP (transmission-control protocol/Internet protocol) overhead and achieve point-to-point bandwidths upwards of 100 Mbytes per second (essentially limited by a 32-bit PCI).

It's important to notice the subtle difference between PC clusters for graphics and other Beowulf systems. A Beowulf system[4] is designed to run high-performance computing tasks with a cluster of PCs, interconnected by a private high-speed network. In addition, a PC cluster for graphics needs to have AGP support and high-end graphics accelerators to achieve high-performance graphics rendering. Most Beowulf configurations don't have such options. In fact, all the clusters currently sold by major vendors don't suit graphics well. In addition, most Beowulf systems are not suitable for graphics applications because of their real-time high-bandwidth requirements.

We're primarily using Linux as the underlying operating system for our development. We chose Linux mainly because it provides a stable platform and lets us leverage several advantages of open-source, particularly access to kernel and low-level networking. Giganet provides open-source Linux drivers for its equipment. Also, a commercial message-passing interface (MPI) implementation runs on top of Giganet hardware.

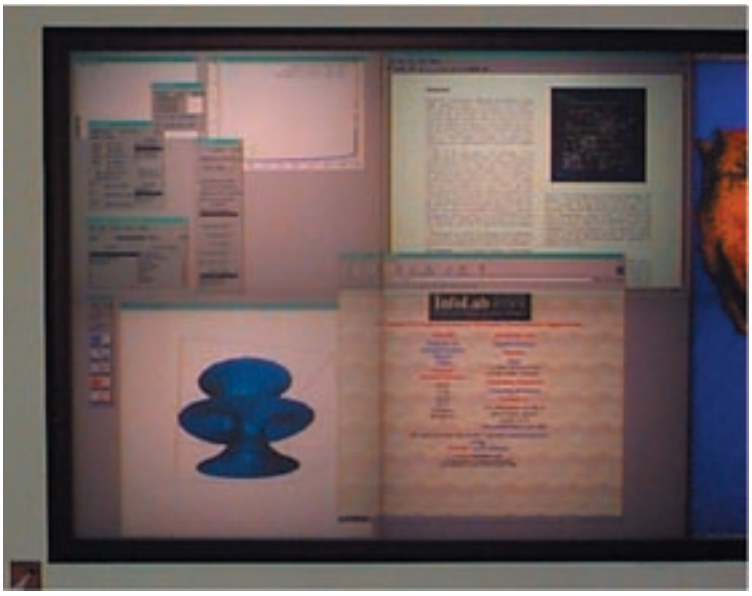
On the graphics end, XFree86, a freely redistributable implementation of the X Window system, is available with hardware-accelerated support for several accelerator cards, including the Nvidia GeForce. (XFree was recently redesigned for faster 3D graphics support.) SGI has also made high-end graphics available in Linux. They released (as open source) several key components that help bring state-of-the-art graphics technology to Linux. They released both the GLX (OpenGL extensions to X) and the OpenGL sample implementation. It's also important that Iris Performer supports Linux (actual support is only available on SGI Linux machines, but it runs fine on other machines that have OpenGL support), although no source-code distributions exist.

In what follows, we describe our work in providing one uniform, seamless display.

X environment for the tiled display

A major stumbling block in using PC clusters for scalable displays is that the vast majority of software we use doesn't work on a tiled display. This proves true for most of the in-house software and for virtually all of the third-party visualization applications (such as Geomview or Wolfram Research's Mathematica). To create the illusion of a single, large screen, we use a software package developed at AT&T Cambridge Labs called Virtual Network Computing (VNC).[5] VNC lets users access their computers remotely, making their console's physical location irrelevant. For this, VNC's developers created a lightweight protocol for sending events (keyboard, mouse, and so on) and images over the network.

However, VNC has no standard feature for tiling displays. Because of this we had to modify the source code. The VNC package comes with a modified X server



4 A snapshot of the InfoWall with a PC cluster running on the left half of the wall. With one seamless display on the PC cluster, we can run existing applications without modification. On this particular screen, we have Geomview, Gnuplot, Netscape, and Adobe Acrobat running.

(based on Xfree source code), Xvnc, that runs with a virtual frame buffer. That is, you can allocate a screen that's as large as the server's memory allows. To see the current state of Xvnc's frame buffers, you must use the VNC protocol to query the Xvnc server. The protocol is quite straightforward. First, the client handshakes with the server and agrees on a set of allowed image formats. Then the client specifies to the server a particular region of interest of the overall screen, which the server sends incrementally to the client every time a change occurs. A particularly nice feature is that the server only encodes portions that have changed and transmits them to the client, thus lowering the necessary network bandwidth. On each client machine, we run a modified VNC client, which updates its portion of the display.

Table 1. Data (in bytes) and packets received by the four Linux PCs per frame on average.

Node	Data	Number of Packets	Data per Packet
1	47,771	11.17	4,276
2	46,800	10.96	4,267
3	46,857	8.78	5,338
4	47,534	8.96	5,306
All	188,962	39.87	4,796

Node 1 is the top-left quarter portion of the screen; Node 4 is the bottom-right quarter portion of the screen.

Figure 4 shows the InfoWall's left 2×2 projectors driven by four Pentium II PCs running Linux. It provides a unified and seamless display for user applications out of multiple machines. However, it requires a fast communication network and protocol to accommodate the application, which generates results and displays them on remote machines. For example, if we use 2×2 PCs with a total resolution of 2,560 by 2,048 and run a full-screen 3D animation program in Geomview, the pixel data that must be transferred across the network (we used Ethernet as the underlying network for our preliminary experiments) from the server to a display machine can be over 3.8 Mbytes per frame. Clearly, the network bandwidth can't meet such requirements.

To reduce the network traffic, we also ran the Geomview 3D animation application by taking advantage of the most efficient compression scheme, Hextile RRE32 (Runs-and-Run-length Encoding), available in the VNC protocol.[5] We measured the size and the number of packets transferred from the server to each client PC for 30 frames. Table 1 lists the average data across the network to each PC.

If the update rate is 10 frames per second, the network control must sustain the bandwidth of at least 1.8 Mbytes per second with about 400 4- to 5-Kbyte packets. Even if we had a machine with negligible computation cost for compressing and decompressing data, the capacity of Ethernet still can't transfer the amount of data required for moderate-sized 3D applications.

Future work

One of the problems with a PC cluster system is that an application that starts running on one machine is responsible for the distribution of relevant work to other PCs for display. This application node serializes the application processing. A challenging issue is to develop a parallel API that uses multiple PCs to achieve better performance.

We're also extending the VNC protocol to handle 3D primitives. We aim to use the server to execute the X operations in the way VNC currently works and relay the GLX protocol to the client. To accomplish this requires several changes to the software. First, Xvnc has to be upgraded to the new version of Xfree, which has GLX support. We also have to extend the VNC client to

support the new protocols. In doing so, we're almost changing the complete implementation of the VNC client, currently based on the X protocol.

We're also interested in working on resource sharing across multiple display walls. The InfoWall has a dedicated T1 connection to the Display Wall at Princeton University.[6] This provides us a great opportunity for exploring remote collaboration, visualization, and interactions with large displays. ■

Acknowledgments

We'd like to thank James Pelletier from AT&T Labs for his effort in setting up the InfoWall.

References

1. J. Montrym et al., "InfiniteReality: A Real-Time Graphics System," *Proc. Siggraph 97*, ACM Press, New York, 1997, pp. 293-302.
2. E.E. Koutsofios et al., "Visualizing Large-Scale Telecommunication Networks and Services," *Proc. IEEE Visualization 99*, ACM Press, New York, Oct. 1999, pp. 457-461.
3. P.N. Glaskowsky, "Nvidia GeForce Offers Acceleration," *Microprocessor Report*, 1999 Sept. 13., pp. 5-10.
4. T.L. Sterling et al., *How to Build a Beowulf: A Guide to the Implementation of PC Clusters*, MIT Press, Cambridge, Mass., 1999.
5. T. Richardson et al., "Virtual Network Computing," *IEEE Internet Computing*, Vol. 2, No. 1, 1998, pp. 33-38. See also http://www.uk.research.att.com/vnc/.
6. R. Samanta et al., "Load Balancing for Multi-Projector Rendering Systems," *Proc. Eurographics/Siggraph Workshop on Graphics Hardware*, ACM Press, New York, Aug. 1999, pp. 107-116.

Bin Wei *is a senior technical staff member at AT&T Research Labs. His research interests are in the areas of high-performance computer systems, computer graphics, visualization, interactive user interfaces, and digital video. He received a PhD in computer science from Princeton University in 1998. He received a BS in computer science from Tianjin University, China, in 1983 and an MS in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 1989.*

Claudio Silva *is a senior member of technical staff in the Information Visualization Research Department at AT&T Research Labs. His main research interests are in graphics, visualization, applied computational geometry, and high-performance computing. He earned a PhD and an MS in computer science at the State University of New York at Stony Brook in 1996, and 1993, respectively. He received a BS in mathematics at the Federal University of Ceara, Brazil, in 1990. While a student, and later as a National Science Foundation post-doc, he worked at Sandia National Labs, where he developed large-scale scientific visualization algorithms and tools for handling massive data sets.*

Eleftherios Koutsofios *is currently a technology consultant in the Network Services Lab at AT&T Research Labs. His research interests are in the areas of interactive techniques, display and user interaction technologies, and information visualization. He received his PhD in computer science from Princeton University in 1990.*

Shankar Krishnan *is senior technical staff member at AT&T Research Labs. His main research interests are 3D computer graphics, geometric and solid modeling, computer algebra, and robust geometric computation. He received a BS in computer science from the Indian Institute of Technology, Madras in 1991. He received his MS and PhD from the University of North Carolina, Chapel Hill in 1993 and 1997, respectively.*

Stephen North *is Head of Information Visualization Research, a group that he and his colleagues founded in 1998 at AT&T Research Labs to explore applications of novel graphical displays, geometric algorithms, and high-performance computing for displaying and interacting with information about global telecommunication networks and information systems. His background is in software and network visualization, applied computational geometry, and the design of reusable software components. He received a BS from Montclair State College, Upper Montclair, New Jersey, in 1980 and an MA and PhD in computer science from Princeton University in 1983 and 1986, respectively.*

Readers may contact the authors at AT&T Shannon Laboratory, Information Visualization Research, 180 Park Ave., Bldg. 103, Florham Park, NJ 07932, e-mail {bw, csilva, ek, krishnas, north}@research.att.com.