# Streaming visualization for collaborative environments

**M Hereld[1], E Olson[2], M E Papka[1] and T D Uram[1]**
[1] Argonne National Laboratory, Argonne, IL
[2] The University of Chicago, Chicago, IL

E-mail: papka@anl.gov

**Abstract.** Connecting expensive and scarce visual data analysis resources to end-users is a major challenge today. We describe a flexible mechanism for meeting this challenge based on commodity compression technologies for streaming video. The advantages of this approach include simplified application development, access to generic client components for viewing, and simplified incorporation of improved codecs as they become available. In this paper we report experimental results for two different applications developed to exploit this approach and its merits. Using typical datasets under realistic conditions we find the performance for both is satisfactory.

## 1. Introduction

The problem of sharing visualization resources in the form of computational power and data storage is one of the important challenges that all users of HPC face. These resources are scarce and geographically distant from many users and from other resources. Analyzing the results of large simulations must use these resources, and yet the user may not be conveniently near to them. Many approaches to solve this problem (perhaps most) center on connecting remote users to the analysis engines and raw data through tools that present relatively condensed representations locally. Bandwidth and latency prove to be the key limiting parameters that govern the success of the solution. Bandwidth limits the amount of data that can be viewed, while latency limits the responsivity of the control system that enables interaction with the data and analysis.

The need for a new paradigm in handling pixels is being expressed in several areas of ongoing work. At the more traditional end of the scale are the solutions that consider end-point displays as containing graphics horsepower according to the usual workstation standards. Multi-display systems using WireGL [1], Chromium [2], and similar solutions are of this ilk. A central problem for architectures based on this paradigm is that operating system integration with the display adaptor has evolved to provide powerful graphics pipelines connecting the local application to the local pixels: breaking into this pipeline while retaining high performance is difficult. At the other end of the scale, solutions that put pixels in the prime position (as opposed to graphics primitives) are providing perhaps the most flexible and neutral paradigm, typically at the expense of performance. SAGE [3], VNC [4], and the many progeny of VNC are among the approaches that fit into this category. The venerable and ancient X Window System lies between these extremes.

## 2. Our approach

In devising a solution to this problem we are motivated by several characteristics of distributed scientific collaborations:

- The size of scientific data sets has been growing exponentially, and the rate of growth will continue to increase;
- Storage and compute resources are often centralized;
- Collaborations are increasingly multi-institutional, requiring communication among groups of people distributed geographically.

Video codecs designed for low bandwidth, high frame rate applications by utilizing inter- and intra-frame compression and motion compensation appear a practical intermediate format for transmitting visualization data. Streaming video over the network links provides near-immediate views of the data. Streaming over multicast brings the further benefit of one-to-many distribution while accruing no incremental network or processing burden to the sender.

Based on these considerations, our approach is to build interfaces between visualization tools and streaming video tools to allow remote collaborators to see views of the data generated by compute resources at the sender site. We also aim to allow the remote sites to interact with the visualization from a distance. The sections below detail how we have applied this approach to two applications, vl3 [5] and ParaView [6].

### *2.1. vl3 case: client-side stream aggregation*

The vl3 volume visualization application was developed at Argonne National Laboratory. It exploits the latest features of graphics hardware (for example, programmable shaders) to optimize performance, and includes support for cluster execution to scale up to the ever increasing size of scientific data sets.

Typically run on a cluster, vl3 is driven by a head node with rendering nodes communicating over MPI. Each cluster node sends an h.261-formatted video stream to a destination address (figure 1). The receiving client, a modified version of the streaming video tool vic, assembles the streams into a single video for display.

The receiving client has also been modified to accept mouse and keyboard interaction, and to relay those interactions back to vl3 running on the cluster, to effect changes in orientation, color map, and display. This model allows any receiver to control the rendering, subject to network security constraints between the client and the cluster nodes.

By streaming the output frames as video, vl3 enables remote viewing of and interaction with the data, avoiding the common hindrances of transfer of large datasets, access to remote compute resources (possibly across institutional boundaries) and familiarity with complex visualization software.

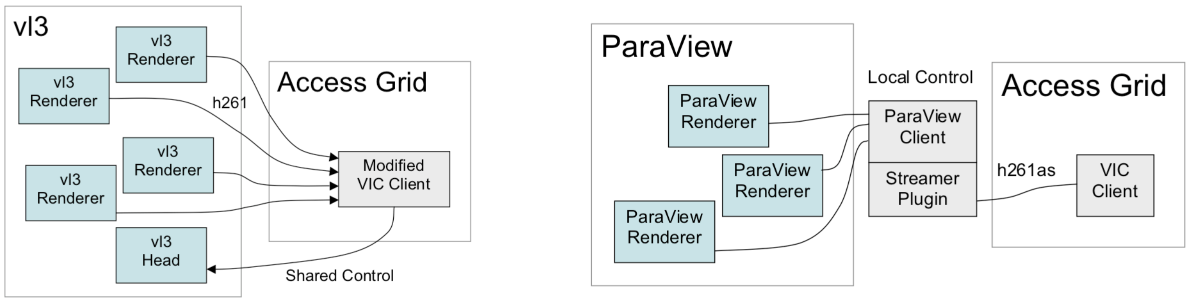

**Figure 1.** Comparison of the vl3 and Streaming ParaView application architectures.

*2.2. Paraview streamer: a remote visualization plugin*
ParaView consists of a client application for viewing and controlling the visualization, and one or more optional servers. The render server can be run on a single machine, or on a cluster of machines communicating over MPI. Because ParaView is a mature and feature rich application, and is widely used for visualizing data from many disciplines, it was a natural target for extension into collaborative scientific visualization.

The ParaView plugin architecture provides a natural way to add streaming video for remote visualization, producing data streams in a generally accessible format and at lower bandwidth. When loaded, the plugin adds controls to the ParaView toolbar to allow the user to specify the destination address to which frames should be streamed, and to start and stop streaming. Additionally, ParaView can be launched directly from within the Access Grid, where it will automatically set the destination to the current venue [7].

The streaming plugin subscribes to render events in the main ParaView render area, updating an internal buffer. A separate thread streams frames from the buffer at native resolution at 30 frames per second, independent of ParaView's rendering. The format used for the video stream is h.261 with modifications to support arbitrary size frames, known informally as h261as. Support for this codec is included in recent releases of the streaming video tool vic; this support is also included in recent releases of the Access Grid, allowing users to receive and display these streams with no additional effort. Because the h.261as codec includes support for motion vectors and interframe compression, the stream bandwidth drops appreciably when the rendering is unchanged.

## 3. Measurements
To study the performance of these two applications we devised an experiment to capture data rates for all streams as a function of time and activity. In both cases, we used a dataset that typified our use of the system. While one member of the team manipulated the interactive controls to rotate, translate, and zoom the visualization, the other annotated the activities with single letter codes added to a running log with automatic time stamps at 1-second intervals. The client code reported the bandwidth data at 1-second intervals as well. The experimental

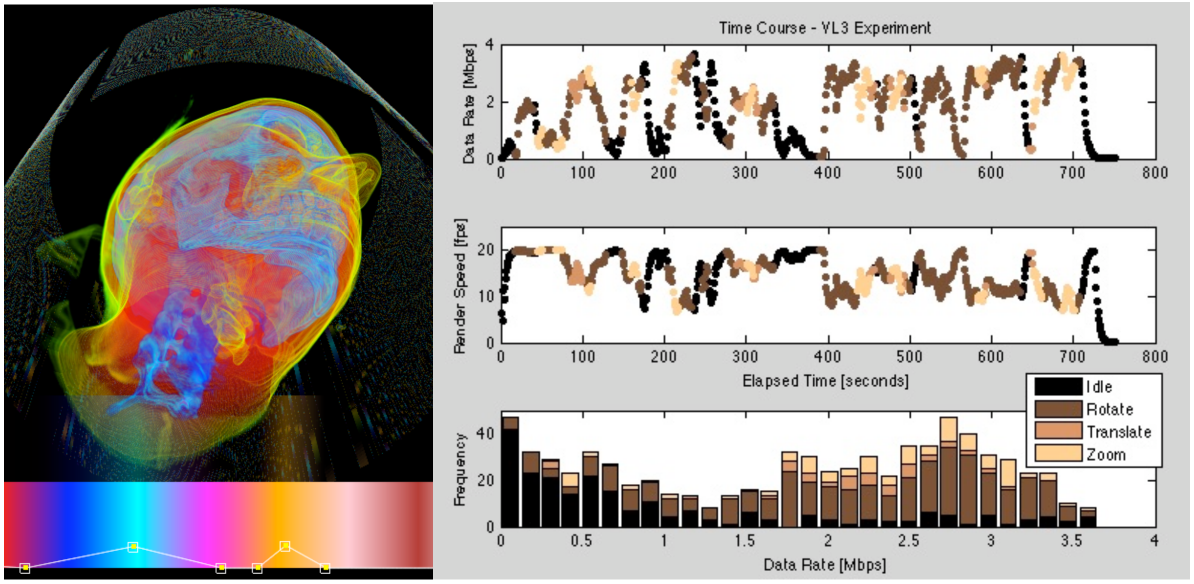

**Figure 2.** The vl3 streaming experiment. At left is a snapshot from the data browsing experiment. At right is a summary of the performance data from a timed exploration. (Data courtesy University of Chicago)

design emphasizes measurement of a realistic use case over a significant time period – in excess of 10 minutes of recorded data were collected. In settling on this methodology, we were driven by the observation that the instantaneous performance depends on the complexity of interactions between the data, the streaming codec, the POV, and the rate of change of the POV. Significant variation in the measured performance is to be expected from moment to moment. Quantitative modeling would be extremely difficult and of questionable value.

The results from both experiments are presented in figures 2 and 3. Each synopsis includes an image taken of the data in a typical pose to give an indication of both the visual complexity of each and of the variety of data in common use. The annotated performance data is reported on the right hand panel. Annotations show when the user was idle at the controls (perhaps studying the image), or when he was manipulating the view with rotate, translate, or zoom. Each created a log of at least 10 minutes of interactive browsing. In the vl3 experiment, four separate streams from rendering processes were bonded at the client to form the final image. The data rate reported in the figure is the sum of the individual stream data rates. In the ParaView experiment, the visualization was generated by one render process and the values reported in the figure are for the h.261as stream as received by the vic client.

A few observations can be made about the results shown in figures 2 and 3.

- The frame rate varied by a factor of two or so over the course of both experiments. The bandwidth varies more, but its impact is capped by the maximum frame rate.
- Except for a large peak from the low bandwidth required during idle time, the distribution in both cases is not particularly remarkable. Judging from the stacked distributions in the histogram, different activities do not seem present significantly different demands: for example, rotation seems no more or less taxing than zoom.
- Frame rates dip below 10 fps at times for the rendered data in the vl3 test, presumably due to a combination of the relative complexity of the image structure and fraction of the image area it fills.

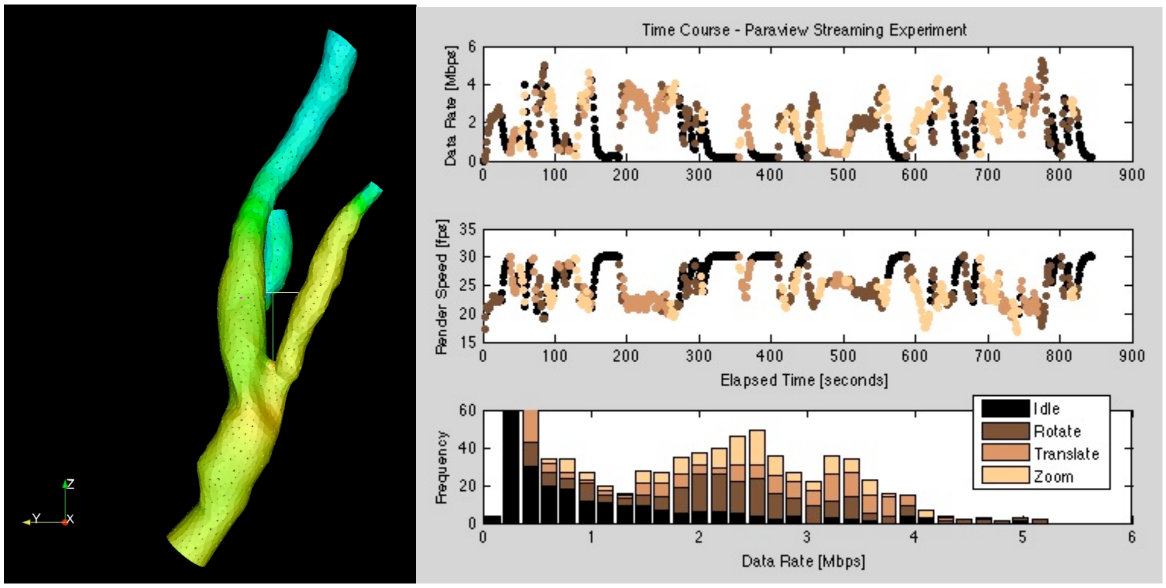

**Figure 3.** The ParaView streaming experiment. At left is a snapshot from the data browsing experiment. At right is a summary of the performance data from a timed exploration. (Data courtesy of Brown University)

## 5. Conclusions

We have described our approach to providing interactive remote visualization to collaborative environments. Based on video streams and widely used non-proprietary codecs, it can provide high quality visualization at low development cost to a range of standard clients. We have described two applications based on the idea and presented experimental results from realistic performance tests.

The experimental results are promising:

- In both applications we verified acceptable performance.
- Not surprisingly, we note significant variation in the bandwidth required – with dependence on data, orientation, scale, and motion.
- The required bandwidth in both cases was no more than several megabits-per-second, a value that is already commonplace in typical research settings.

We plan future work in this area to include incorporation of improved codecs, notably h264, and adding shared control to the ParaView application.

**Acknowledgments**

The authors would like to acknowledge Joseph Insley for his help in this work; the University of Chicago TeraGrid RP support team; and the Australian National University for its modified h261 codec. This work has been supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, United States Department of Energy, under Contract DE-AC02-06CH11357. Additional supported provided by the National Institutes of Health/National Library of Medicine, under Contract N01-LM-3-3508 and the National Science Foundation under Grant No. ANI-01-22296.

## References

[1] Humphreys G, Eldridge M, Buck I, Stoll G, Everett M and Hanrahan P 2001 WireGL: a scalable graphics system for clusters. *Proceedings of the 28th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '01*. ACM, New York, NY, pp. 129-140.

[2] Humphreys G, Houston M, Ng R, Frank R, Ahern S, Kirchner P D and Klosowski J. T. 2002 Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Trans. Graph*. **21**:3, pp. 693-702.

[3] Jeong B, Renambot L, Jagodic R, Singh R, Aguilera J, Johnson A and Leigh L 2006 High-performance dynamic graphics streaming for scalable adaptive graphics environment. *ACM/IEEE Supercomputing 2006 Conference (SC'06)*.

[4] Richardson T, Stafford-Fraser Q, Wood K R and Hopper A 1998 Virtual network computing. *IEEE Internet Computing*, **2**:1, pp. 33 – 38.

[5] Binns J, Dech F, Papka M E, Silverstein J C and Stevens R 2005 Developing a distributed collaborative radiological visualization application. *From Grid to HealthGrid*, pp. 70-79.

[6] Ahrens J, Geveci B and Law C 2005 ParaView: an end user tool for large data visualization. *The Visualization Handbook*, C. D. Hansen and C. R. Johnson, Eds.: Elsevier Inc., pp. 689-716.

[7] Stevens R 2003 Access Grid: enabling group oriented collaboration on the grid. *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Editors, 2nd ed: Morgan Kaufmann, 2003.