Distributed and collaborative visualization of large data sets using high-speed networks

Andrei Hutanu [a], Gabrielle Allen [a], Stephen D. Beck [a], Petr Holub [b,c], Hartmut Kaiser [a], Archit Kulshrestha [a], Miloš Liška [b,c], Jon MacLaren [a], Luděk Matyska [b,c], Ravi Paruchuri [a], Steffen Prohaska [d], Ed Seidel [a], Brygg Ullmer [a], Shalini Venkataraman [a]

[a] *Center for Computation and Technology, 302 Johnston Hall, Louisiana State University, Baton Rouge, LA 70803, United States*

[b] *CESNET z.s.p.o., Zikova 4, 16200 Praha, Czech Republic*

[c] *Masaryk University, Botanick 68a, 62100 Brno, Czech Republic*

[d] *Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany*

Abstract

We describe an architecture for distributed collaborative visualization that integrates video conferencing, distributed data management and grid technologies as well as tangible interaction devices for visualization. High-speed, low-latency optical networks support high-quality collaborative interaction and remote visualization of large data.

Key words: Collaborative Visualization, Distributed Applications, Co-scheduling, Interaction Devices, Video Conferencing

---

1 Introduction and iGrid Scenario

The study of complex problems in science and engineering today typically involves large scale data, huge computer simulations, and diverse distributed collaborations of experts from different academic fields. The scientists and researchers involved in these endeavors need appropriate tools to collaboratively visualize, analyze and discuss the large amounts of data their simulations create. The advent of optical

Email address: `Primary contact : ahutanu@cct.lsu.edu` (Andrei Hutanu).

networks opens doors to new low latency, high bandwidth approaches that allow these problems to be addressed, in highly interactive environments, as never before.

This article describes recent work to develop generic, novel techniques that exploit high speed networks to provide collaborative visualization infrastructure for such problems. The particular driving problem is provided by a numerical relativity collaboration between the Center for Computation & Technology (CCT) at LSU, and colleagues in Europe, including the Albert Einstein Institute (AEI) in Germany. The collaborators already hold joint meetings each week using AccessGrid technologies. However, much higher image quality, resolution, and interactivity are needed to support collaborative visualization and deep investigations of data.

Interactive Collaborative Visualization Scenario. We designed and implemented technologies to enable multiple sites, connected by high speed networks, to interact with each other and with a server-based visualization system capable of handling large distributed data sets. The most important characteristics of the technologies supporting image-based collaborative visualization are their latency (at most 200 ms for interaction), overall quality of visualization (resolution comparable to a standard desktop with few to no artifacts) and throughput (minimum of 5 frames per second)

Low latency (i.e. uncompressed) high-definition video and audio transport software and hardware is used to connect sites in an interactive video conference session. One site serves as the host for the visualization application, whose output is converted and transferred to the other participating sites by the video transport system. Remote collaborators follow, interact with and steer the visualization session using custom-made interaction devices deployed at all the sites. The control of the visualization system is fully shared [5].

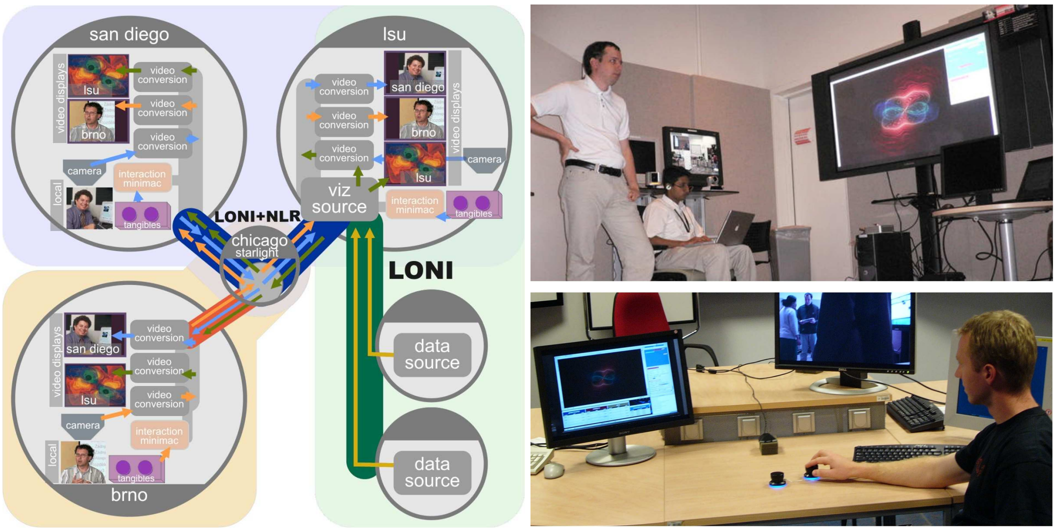

Fig. 1. Left : Illustration of our experiment setup. Right : Remote visualization at iGrid (up), Remote interaction from Brno (down)

The system for collaborative visualization comprises several parts: (i) data visualization; (ii) transport of images to collaborating sites; (iii) tools for remote control of the visualization. The visualization component has a rendering front-end and distributed data feeder back-end which consists of multiple processes running on separate remote supercomputer nodes. A co-scheduling service reserves resources in advance, to ensure that all processes run concurrently, a requirement for an interactive application. Grid technologies facilitate remote job submission to the reserved resources.

iGrid Scenario. For the demonstration in San Diego, CCT/LSU (Louisiana), CESNET/MU (Czech Republic) and iGrid/Calit2 (California) participated in a distributed collaborative session. For the visualization front-end we used a dual Opteron 252, 2.6 GHz, 8 Gbyte main memory, NVidia Quadro FX 4400 graphics (512 Mbyte video memory) at LSU running Amira [10] for the 3D texture-based volume rendering with custom modules for distributed visualization. The visualization back-end (data server) ran on an SGI Prism Extreme (32 Itanium processors, 128 Gbyte shared main memory, 10 Gbit network interface), and an IBM Power5 cluster (14 Nodes, 112 1.9 GHz Power5 processors, 256 Gbyte overall main memory) at LSU. We ran one data process of the data server on the Prism and nine processes each on one node of the P5. The actual data set used, a scalar field produced by a binary black hole simulation, had a size of 120 Gbytes and contained 400^3 data points at each timestep (4 bytes data/point for a 256 Mbyte/timestep).

2 Components

Video Transport System. In order to minimize latency and increase interactiveness for the collaborative visualization, it is important that any non-necessary processing of the image data flow is minimized and, if possible, eliminated completely. Furthermore, it is advantageous to have dedicated network circuits (e.g. optical paths or "lambdas") with very low jitter and close to zero packet reordering, as they allow for eliminating most of the buffering on the receiving side.

For video transmission we opted for using a custom solution based on low-latency uncompressed high-definition video transport described in more detail in an article titled "High-Definition Multimedia for Multiparty Low-Latency Interactive Communication" in this issue. This system captures HD-SDI video with full 1080i resolution (1920×1080 image), encapsulates the data into RTP/UDP/IP packet headers [2] and sends them over the network resulting in 1.5 Gbps per each video stream. The end-to-end latency from capture to display with back-to-back connected computers (thus eliminating latency originating in the network, but including latency from network interfaces of the sending and receiving machines) was approximately 175 ms. The data were distributed to participating sites using UDP packet reflector technology [4] running on machines located at StarLight, Chicago, where all the network links were meeting.

As the HD-SDI capture card did not arrive at LSU before the demo, we deployed compressed high-definition video in HDV format. The price paid for this is a substantial latency increase to approximately 1.9 s and a slightly lower image quality (1440×720). This situation prevented us from sending the visualization images directly using a DVI to HD-SDI converter box. Instead, we used our HD video camera to capture the images off of a monitor.

Distributed Data Management. Responsive interaction is critically important for collaborative visualization systems, and one approach suitable for high speed optical networks is to move appropriate parts of the system to remote computers [9]. We separated the visualization front-end from the data access with a 10 Gbit network connecting the machines running these components. Using the remote machines to pre-cache the large data set to be visualized (or analyzed) improves responsiveness since main memory access plus network transfer over the 10 Gbit network is faster than local disk access.

To optimize the data communication between the two components (front-end and data server) we build upon a remote data access system that separates data selection from data transport [8,6]. This system was enhanced by adding a new data selection interface suitable for data defined on regular grids, as used in the iGrid experiment. We have also implemented the data transport layer based upon the Grid Application Toolkit (GAT) [1] streaming API. This allowed us to experiment with a network protocol interfaced by the GAT, Reliable Blast UDP (RBUDP) [3].

The data server is distributed in order to utilize multiple compute resources for the remote data cache. The distributed server still functions as a whole and for this we developed a component that coordinates the multiple independent processes.

For the setup to work optimally, the remote data servers have to be load balanced to the bandwidth that is available from each of them to the visualization workstation. Currently the load is statically determined in advance and we are working towards automating this process based on the dynamic conditions of the network.

Optimization of the application for a timestep-based data access pattern (the distribution optimization can be easily adapted for other data access patterns) is achieved by assigning a section of each timestep to each node running a data process proportional in size to the available bandwidth from that node to the visualization. In our setup, each of the processes on the P5 cluster was configured to cache at most 12 Gbytes of data whereas the process running on the Prism was configured to cache at most 15 Gbytes of data. In total we were able to cache around 120 Gbytes of data on the remote machines.

Each request for a new timestep is automatically processed by the coordinator, distributed into multiple timestep subsection requests, and each request is directed to the process responsible for it. In our implementation, the coordinator is part of the client in order to minimize latency. Using this distributed approach, we were able to reduce the load time from an NFS-mounted file system from 5 seconds and more to 1.4–1.5 s per timestep.

Data transport. For data transport to the visualization, our original plan was to use the GAT streaming API as well as the RBUDP protocol interfaced by the GAT. This would have enabled us to hot swap the used protocol during the application runtime. Unfortunately we encountered a few issues. As described in [11], RBUDP is not suitable for many-to-one communications, and as we found out, it is practically unusable for many-to-one communications when using a single processor for receiving the data. We also had difficulties in interfacing RBUDP with the current GAT API and the performance of RBUDP when interfaced by the GAT is worse than expected, even for one-to-one communication. For this reason we used a TCP implementation of the data channel at iGrid.

In trying various configurations, we found that using more than ten senders with TCP would decrease the overall transfer performance and this led to the decision to use ten jobs for the distributed data server. The measured TCP bandwidth (using iperf) between the ten machines and the visualization machine when using both CPUs for data transfer on the visualization machine is approximately 3 Gbit/s.

Our application only uses one processor for data transmission while keeping the other one available for the visualization. The end-to-end bandwidth observed by the application (including network transfer, data request, endian conversions) was approximately 1.2 Gbit/s.

Interaction. During our experiment, we observed how remote mouse control (e.g., via the Synergy program) can grow practically unusable over high-latency ($>$ 1 second) image-streaming pipes. Even with lower latency, there are major practical challenges in allowing numerous users to collaboratively manipulate a shared visualization via mouse-based interaction (whether with one or many cursors).

In response, we made experimental use of physical interaction devices called “viz tangibles.”.

We planned to deploy both a “parameter pad,” with two embedded knobs; and an “access pad,” allowing the parameter knobs to be rebound to different parameters using RFID-tagged cards. In practice, we deployed only the parameter pads, each statically bound to two parameters: object rotation and global timestep.

For iGrid, we deployed four sets of parameter pads: two in San Diego, and one apiece in Baton Rouge and Brno. Each pad contained two PowerMate USB knobs. These were connected to Mac Mini machines running Linux Fedora Core 4, using routed TCP sockets via 100MB Ethernet. A simple aggregator software used clutching to transform the four sets of incoming wheel updates to one set of Amira parameters (dispatched over Amira's socket control). Graphical feedback was also used to indicate the evolving activity and control over the shared parameters.

Grid Technologies. To execute the distributed visualization application a critical service was developed to co-schedule the required compute and network resources. After the jobs are co-scheduled, the Grid Application Toolkit (GAT), which provides a simple generic job-submission interface, is used to submit the jobs to the compute resources through the chosen Grid resource management system; for this demonstration, Globus GRAM was used to access PBSPro schedulers.

To ensure that all the resources are made available for the same time period, a phased commit protocol is used; we assume that each resource has a scheduler capable of making and honoring advance reservations. [1] The co-scheduler asks each resource to make a tentative reservation for the required time (*prepare*); if, and only if, all resources respond in the positive (*prepared*), are the reservations confirmed (*commit*). In all other situations, the tentative reservations are removed (*abort*).

The simplest solution to this problem is to use classic 2-phase commit, but here the *Transaction Manager* is a single point of failure, and can cause the protocol to block. To avoid this, the co-scheduling software [2] was based upon the application of Lamport's Paxos Consensus Algorithm to the transaction commit problem [7]. Multiple *Acceptors* co-operatively play the role of the Transaction Manager; a deployment with $2n+1$ Acceptors can tolerate failure of n Acceptors. Even with conservative estimates of Mean-Time to Failure and Mean-Time to repair, it is possible to deploy a set of five Acceptors with a Mean-Time to Failure of over ten years.

In our demonstration, three Acceptors were deployed. In addition to scheduling ten compute resources, two Calient DiamondWave switches (one at LSU, the other at MCNC) were also scheduled. [3]

The Grid Application Toolkit (GAT) was used to submit all the jobs to the previously arranged reservations. The Globus middleware (GRAM client) was used to build the underlying adaptor that implemented the job submission functionality of the GAT API. Currently Globus GRAM does not support job submission to advance reservations. To get round this limitation, we altered the PBS JobManager script to accept the name of the reservation in place of a Job queue. [4]

[1] In the case of the Calient DiamondWave switches, this had to be constructed. The scheduler consists of a timetable for each port in the switch; a reservation requests a number of connections which should be active during the reservation.

[2] Available for download at http://cosched.sourceforge.net/

[3] At the time of the demo, these were not connected to any relevant resources

[4] Although this is simple for PBSPro, which creates a queue for each reservation, the Globus PBS JobManager script by default only accepts submissions to queues which were defined in the scheduler when Globus was *installed*.

3 Conclusions

We have developed a collaborative application that exploits high speed optical networks for interactive, responsive visualization of huge data sets, over thousands of kilometers, with high image quality. Co-scheduling of network and computing resources guaranteed availability. While currently the data transfer does take most of the update time when changing a timestep (1.4 s compared to 0.35 s for transfer to video memory), further optimizations in the networking implementation might reverse this situation. Also, as the data size/timestep increases beyond the capacity of a single video card, investigating distributed rendering front-ends for the visualization becomes a necessity.

One of the lessons learned while using the GAT as well as the BSD socket API was that a streaming API is not fully optimal for the block-wise type of data transfer we are doing. Future efforts will lean towards defining and incorporating message-based APIs as well as related network protocols.

Further investigations in our network implementation are necessary. Immediate next steps include attempting to use multiple processors for data transfer while monitoring the behavior of the visualization and automatically adapting network transmission to changes in the network condition.

4 Acknowledgments

We would like to thank all the people who helped us make this possible : Boyd Bourque, Fuad Cokic, Jiří Denemark, Peter Diener, Lukáš Hejtmánek, Ralf Kaehler, Gigi Karmous-Edwards, Olivier Jerphagnon, Michael Lambert, Lonnie Leger, Honggao Liu, Charles McMahon, Sasanka Madiraju, Andre Merzky, Yaaser Mohammed, Seung Jong Park, Jan Radil, Tomáš Rebok, Sean Robbins, Brian Ropers-Huilman, Rajesh Sankaran, William Scullin, John Shalf, Jeremy Songne, Steve Thorpe, Cornelius Toole, Isaac Traxler, Alan Verlo and Sam White.

This work was supported by the Center for Computation and Technology at LSU, the Enlightened project (NSF grant 0509465); an NSF MRI (grant 0521559); and the Louisiana Board of Regents. The Czech authors were supported by the CESNET research intent (MŠM 6383917201). The loan of two 10GE T210 network cards from Chelsio is highly appreciated.

References

[1] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schütt, E. Seidel, and B. Ullmer. The grid application toolkit: Towards generic and easy application programming interfaces for the grid. *Proceedings of the IEEE, Special Issue on Grid Computing*, 93(3), 2005.

[2] L. Gharai and C. Perkins. RTP payload format for uncompressed video. IETF Draft, Internet Engineering Task Force, Feb. 2004. `http://www.ietf.org/internet-drafts/draft-ietf-avt-uncomp-video-06.txt`.

[3] E. He, J. Leigh, O. Yu, and T. A. DeFanti. Reliable blast udp: Predictable high performance bulk data transfer. In *CLUSTER '02: Proceedings of the IEEE International Conference on Cluster Computing*, page 317, Washington, DC, USA, 2002. IEEE Computer Society.

[4] E. Hladká, P. Holub, and J. Denemark. An active network architecture: Distributed computer or transport medium. In *3rd International Conference on Networking (ICN'04)*, pages 338–343, Gosier, Guadeloupe, Mar. 2004.

[5] G. Johnson. Collaborative visualization 101. *Computer Graphics*, 32(2):8–11, 1998.

[6] R. Kähler, S. Prohaska, A. Hutanu, and H.-C. Hege. Visualization of time-dependent remote adaptive mesh refinement data. In *Proc. IEEE Visualization '05*, 2005.

[7] L. Lamport and J. Gray. Consensus on transaction commit. Microsoft Technical Report MSR-TR-2003-96, Jan. 2004. `http://research.microsoft.com/research/pubs/view.aspx?tr_id=701`.

[8] S. Prohaska and A. Hutanu. Remote data access for interactive visualization. In *13th Annual Mardi Gras Conference: Frontiers of Grid Applications and Technologies*, 2005.

[9] L. L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, and P. M. Papadopoulos. The optiputer. *Communications of the ACM*, 46(11):58–67, 2003.

[10] D. Stalling, M. Westerhoff, and H.-C. Hege. Amira: A highly interactive system for visual data analysis. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*, pages 749–767. Elsevier, 2005.

[11] X. R. Wu and A. A. Chien. Evaluation of rate-based transport protocols for lambda-grids. In *HPDC '04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*, pages 87–96, Washington, DC, USA, 2004. IEEE Computer Society.