Cactus and Visapult:
An Ultra-High Performance Grid-Distributed Visualization Architecture Using Connectionless Protocols

E. Wes Bethel and John Shalf[†]
Lawrence Berkeley National Laboratory
National Energy Research Scientific Computing Center
University of California
Berkeley, CA 94720

Abstract
This past decade has seen rapid growth in the size, resolution, and complexity of Grand Challenge simulation codes. This trend is accompanied by a trend towards multinational, multidisciplinary teams who carry out this research in distributed teams, and the corresponding growth of Grid infrastructure to support these widely distributed Virtual Organizations. As the number and diversity of distributed teams grow, the need for visualization tools to analyze and display multi-terabyte, remote data becomes more pronounced and more urgent. One such tool that has been successfully used to address this problem is Visapult. Visapult is a parallel visualization tool that employs Grid-distributed components, latency tolerant visualization and graphics algorithms, along with high performance network I/O in order to achieve effective remote analysis of massive datasets. In this paper we discuss improvements to network bandwidth utilization and responsiveness of the Visapult application that result from using connectionless protocols to move data payload between the distributed Visapult components and a Grid-enabled, high-performance physics simulation used to study gravitational waveforms of colliding black holes; The Cactus code. These improvements have boosted Visapult's network efficiency to 88-96% of the maximum theoretical available bandwidth on multi-gigabit Wide Area Networks, and greatly enhanced interactivity. Such improvements are critically important for future development of effective interactive Grid applications.

Introduction
Over the past decade, growth in the memory and data storage capabilities of the largest supercomputing installations in the world has outpaced Moore's law. This has lead to a significant data management and data analysis problem. Even more dramatic growth is expected in the observational sciences as high-resolution data feeds from remotely operated network/Grid-connected observatories and experimental equipment come online [1]. While statistical methods and feature detection/extraction algorithms have been proposed to automate the mining of useful information from these enormous data stores, there is still a strong need for a human component to the visualization process. It is not realistic to expect that every numerical simulation domain will have suitable automated methods to mine and detect features in large data sets. Detection of unexpected phenomena is still most often performed using interactive visualization.

Consequently, supercomputing centers, such as the National Energy Research Supercomputing Center (NERSC), are motivated to provide visualization tools that enable effective interactive analysis of remotely located data stores and remote monitoring of simulation codes runs that can span multiple days, or in some cases, weeks. Such tools must balance the competing interests of interactivity and fidelity in order to fit within the limits imposed by available infrastructure. The emerging Grid infrastructure ties these resources together into a global fabric integrating distributed research teams and virtual organizations with the remote research hardware, software and services used to carry out their work.

The NERSC/LBNL Visualization group has developed the Visapult [2] tool to attack these sorts of "Grand Challenge" problems. Visapult is a distributed, parallel volume rendering application that leverages parallel computation and high performance networking resources that are on the same scale as the supercomputers used to generate the data. Volume rendering is a very important visualization technique

[†] Authors' contact information: jshalf@lbl.gov and ewbethel@lbl.gov, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Mail Stop 50F, Berkeley CA, 94720.

for exploration of complex 3D datasets, but is extremely compute and bandwidth intensive. Traditional serial raytracing volume renderers can take many minutes or hours to render a single frame. Visapult supports volume rendering at interactive rates by employing network distributed components, a high degree of parallelism, and a rendering technique called Image Based Rendering Assisted Volume Rendering (IBRAVR). The IBRAVR algorithm, described more completely in other papers [2,4], allows Visapult to trade additional computational power in exchange for reduced bandwidth requirements. In this paper, we will describe the high performance remote visualization tool called Visapult and our efforts to improve its effectiveness using aggressive network tuning and network protocol modifications. In particular we will describe how we used a new connectionless UDP protocol to improve network utilization efficiency from 25% of line-rate to 88% of line-rate for multi-gigabit networks. Connectionless protocols also dramatically reduce the latency of network event delivery, thereby improving the responsiveness of wide-area distributed interactive graphics applications as compared to TCP stream protocols. We believe that these UDP protocols and development of transport encodings and algorithms that can tolerate loss gracefully will be a fundamental component of future Grid visualization architectures due to their ability to more effectively utilize network bandwidth.

Driving Application (Numerical Relativity)

A motivating application for this kind of Grid-distributed visualization capability is the modeling of black hole collisions. One of the most challenging problems in science is the numerical simulation of Einstein's equations in order to explore Einstein's Theory of General Relativity (GR). These equations are among the most complex in the world of physics; a set of nonlinear, hyperbolic, elliptic coupled equations containing thousands of terms when fully expanded. The General Relativity Group at the Albert Einstein Institute in Potsdam Germany developed the Cactus code [3] to solve these equations from first principles on supercomputers in order to simulate astrophysical phenomena with extremely high gravitational fluxes, such as the collision of two black holes and the gravitational waves that radiate from that event.

The Cactus simulation code scales well on some of the largest supercomputers in the world, including NERSC's 3000+ processor 5Teraflop IBM SP2 (seaborg). Large-scale MPP's like seaborg are required to reach the target regime of physics they would like to explore; the "spiraling" merger of two black holes. Simulations that require comparatively small amounts of memory (~1Tbyte) allow accurate simulations for simpler phenomena like the Schwartzchild (spinning black hole) and Misner (head-on-collision of two black holes) cases (*see figure 1)*. While simulating of these cases is important for validating and calibrating the solvers used by the code, they are of little consequence to the endeavor of gravitational wave characterization and detection. The spiraling merger of two black holes offers a more compelling and astrophysically relevant source for detectable gravitational waves. At a minimum, 1.5TB of memory is required to simulate a basic spiraling collision using bitant symmetry (reflection along the plane of rotation). A full 3D evolution requires a minimum of 3-5TB of RAM. Still more memory is required to move the outer boundaries far enough away from the event to perform accurate gravitational wave extractions necessary to assist in the signal processing performed on LIGO data.

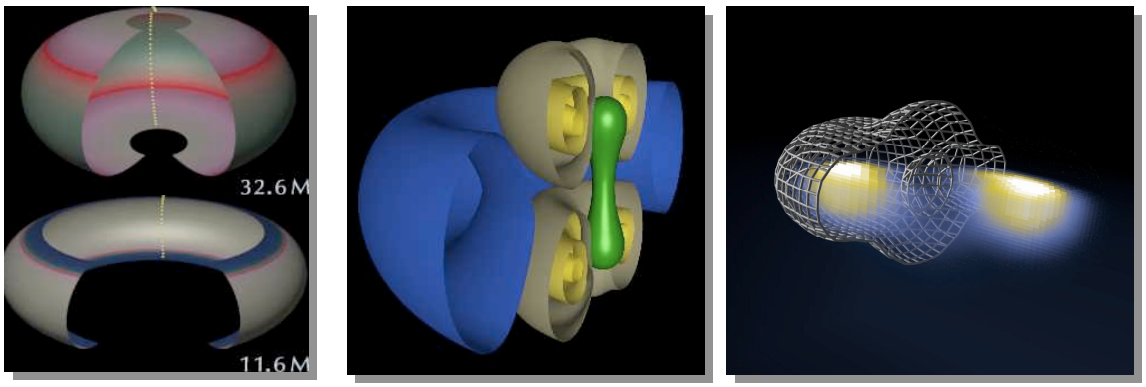

Figure 1: Images of black hole simulations showing the Schwartzchild, Misner, and spiraling merger cases respectively. The first image shows the event horizon of the Schwartzchild spinning black hole colorized by its extrinsic curvature. The Misner visualization in the center shows the real component

of the gravitational potential (psi4 of the Weyl tensor) as blue, grey, and yellow isosurfaces (cut open so you can see inside) and the event horizon of the merging black holes as a green surface. The last image is a volume rendering of the lapse with a cut-open wireframe isosurface of the real part of the gravitational waves emanating from an offset inspiraling merger of two black holes.

Cactus contains some extremely advanced Grid computing capabilities used to reap the benefits of a globally distributed supercomputing infrastructure that serves a globally distributed virtual organization of astrophysicists. In one Grid computing paradigm, the Cactus simulation can be distributed across multiple computing sites to form a virtual supercomputer far larger than any individual machine in the world. Cactus employs runtime adaptive metacomputing techniques that adjust the number of ghostzones used on boundaries that cross the WAN, as well as dynamic adjustment of the compression level used to transport data for the boundary. In one experiment using more than 3000 processors distributed across supercomputing sites on multiple continents, the adaptive techniques were able to improve the Cactus speedup from 15% to nearly 80% of peak. In another Grid application scenario, the Cactus Worm is a nomadic application that can discover more capable resources on the Grid: checkpoint itself and migrate to better resources using Grid standard protocols and information services. The migratory and world-wide distributed nature of this application requires a visualization/analysis infrastructure that is similarly Grid-savvy.

These simulations are so large that it is often impossible to use traditional visualization tools to see and understand their results. Since gravitational physicists do not know, a-priori, the nature of the gravity wave signals they are studying, data mining/feature extraction techniques are inappropriate for this activity. It is not feasible to download all of the data from the supercomputer center to a remote site for later analysis. Consider that a 3-5Terabyte dataset would take 11-18 hours to download if you could reliably sustain a 1Gigabit stream using TCP (very unlikely). Even pre-loading the datasets on your local visualization system is a dubious proposition due to the large size of the computed data sets. Few high end visualization systems have more than a few Gigabytes of memory, or more than a few processors. The platform needed to analyze the data must be similar in scale to that used to produce the data, unless significant data reduction or non-interactive out-of-core methods are employed. Otherwise, some form of real-time dynamic data reduction would be needed before the data gets delivered to the scientist's desktop in order for data analysis to be practical. This sort of data reduction is a natural consequence of many visualization methods.

Typically, visualization is performed at a particular supercomputer in-situ with the running simulation, with the results made available to a remote viewer located, possibly, halfway around the world. Such remote monitoring improves the effective use of supercomputers by allowing the Cactus users to cut short runs that have gone awry, or to steer the code or restart it with more appropriate parameters or initial conditions. The interactive, remote visualization and monitoring methodology requires latency-tolerant visualization algorithms with an emphasis on maximizing Wide Area Network (WAN) throughput and interactivity approximating that of locally hosted applications.

Visapult and the Latency-Tolerant Direct Volume Rendering Algorithm

As described in earlier papers [2], Visapult is composed of three components: a raw data source, a viewer, and an MPI-based back end (*see figure 2*). The back end first reads data from the raw data source, which may be either local or remote. Next, the back end domain decomposes the volume of data into smaller units, each of which is direct volume rendered to produce a single image of that particular sub-domain. Relatively little communication is required between the back end processes, so the system scales very well to large numbers of processors. Each of these images is then transmitted to a viewer, where they are texture mapped onto static geometry, and rendered using OpenGL. This process takes advantage of both the computing horsepower of MPP's for partial rendering as well as the benefits of hardware-accelerated rendering available on many low-cost client workstations. It also decouples interactivity on the desktop from the latencies involved in moving data over the network. Our implementation of this technique using large MPPs, high performance networks and desktop viewer is a high-performance extension to the IBRAVR algorithm described in previous literature [4], and was presented in previous publications [2].

Scientific data is read into the Visapult back end in a domain-decomposed fashion, where each PE of the back end volume renders its block of data in software, and sends the resulting image to the viewer. This architecture results in drastically different I/O requirements along the processing pipeline. Whereas O(N^3) bandwidth is required into the Visapult back end, only O(N^2) bandwidth is required between the back end and the viewer. During the NGI project, our objective was to make use of "network-centric" resources, such as network-based storage and network-based compute farms, to solve a single scientific problem. For this reason, we were very much interested in studying the WAN performance of moving data across the WAN into the Visapult back end. The O(N^3) to O(N^2) reduction in bandwidth requirements between data and back end and viewer made Visapult ideally suited for deployment on network topologies in which resources were connected by high-speed links, but links to the viewer were over slower links. The multi-streaming I/O of Visapult's back end reader, combined with the flexibility to extend it to new types of data sources has proven to so effective at utilizing network resources that Visapult, combined with custom code for moving data across networks, has won the SC Bandwidth Challenge two years in a row.

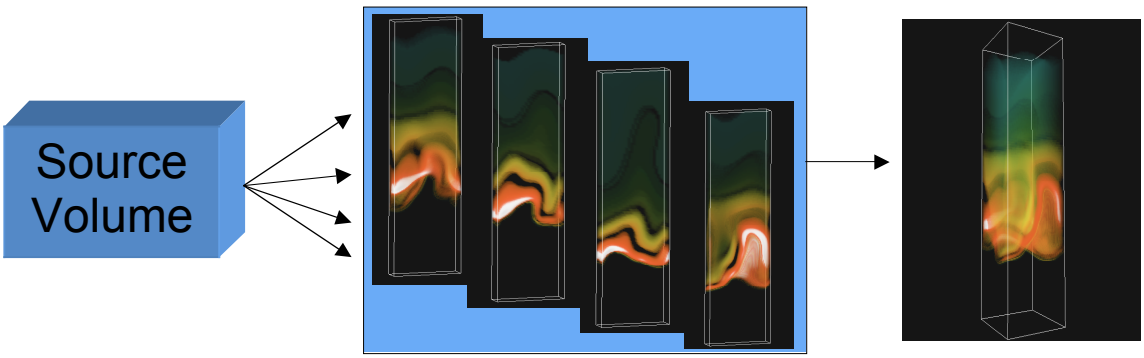

Figure 2*: Diagram of the Visapult processing pipeline. The data source (either a running Cactus simulation or DPSS), feeds the Visapult back-end using multiple parallel network data streams. Each process of the Visapult back-end volume renders its portion of the spatially decomposed domain into an image. These images are in turn sent to a client application running on the user's workstation that uses the texturemapping capabilities of commodity hardware-accelerated graphics cards to composite these images into a single scene at interactive rates.*

Grids, WANs, and TCP Performance

The various components of Visapult must communicate with one-another using IP connections over a LAN or WAN, depending on the deployment of the components. I/O between data source and back-end is performed asynchronously so that it overlaps the volume rendering computations (*see figure 3*). Despite this, the effective performance of the application when operating with maximum fidelity is network-I/O bound unless the network communications are extremely efficient. Visapult has parameters that allow us to reduce I/O utilization in order to fit within the network-imposed limits, but this requires a commensurate trade-off in visual fidelity. Therefore, improvements in the effectiveness of network I/O are critical to maximizing application performance.

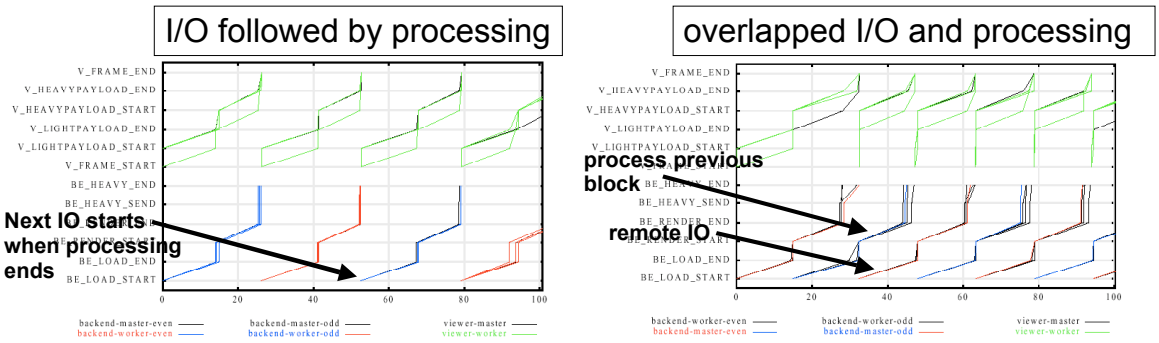

***Figure 3**: By placing network I/O into a background thread, the visapult volume rendering computations could be overlapped with the data transfers, thereby doubling the duty cycle of the application. This overlap fails if network I/O takes longer than the volume rendering calculation, so optimizing network performance is critical to improving overall application efficiency. Netlogger was used for this performance analysis.*

Single-stream TCP performance on the WAN is often disappointing. Even with aggressive tuning of the TCP window size, buffer sizes, and chunking of transfers, typical performance is still a fraction of the available bandwidth on the WAN on OC-12 or faster links. While there are a number of factors involved, the behavior of the TCP congestion avoidance algorithm has been implicated as a leading cause of this performance deficit. Indeed, the preamble to Sally Floyd's High Speed TCP RFC draft (http://www.ietf.org/internet-drafts/draft-floyd-tcp-highspeed-00.txt) points out that in order to achieve steady-state 10Gigabits/sec throughput using standard TCP implementations with 1500 MTU and 100ms round-trip latency, one must have at most only one congestion event for every 5,000,000,000 transmitted packets. That means one lost packet every 1 2/3 hours in order to reach this bandwidth! This is clearly not practical under any circumstance and yet wide-area distributed supercomputing systems like the Teragrid that use multiple transcontinental 10Gigabit links are already coming online. Alternative methods must clearly be employed to make effective use of these expensive national assets.

TCP Multistreaming

The typical solution to work around TCP congestion avoidance is to use multiple simultaneous TCP connections. This is actually a very old solution that is most commonly employed in web-browsers dating back to xmosaic, and is well described in Stevens' classic book on network programming. This technique works because streams don't share packet loss/window-size information with one-another, so a lost packet will only cause one of the streams to back off rather than all of them. The effect is to make the hypersensitive TCP congestion avoidance algorithm artificially less sensitive to frame loss, but this undoes the "fairness" implicit in the congestion avoidance algorithm. In effect, a multistreaming application becomes a network "bully" that competes rather unfairly with its peers. Moreover, it is difficult to tune the parameters of TCP multistreaming in order to maximize performance. If you wanted to use this technique to reach 10Gigabits, as per the Floyd example, you would have to employ approximately 4000 simultaneous TCP streams to reach full line rate. This kind of protocol arrangement is typically implemented at the application level in an ad-hoc application-level manner, but GridFTP (http://www.globus.org/datagrid/gridftp.html) formalizes this methodology into a standard protocol.

TCP Implementation of Visapult

This multi-streaming method was used for the first implementation of Visapult, developed under the NGI program for the Combustion Corridor project. The Distributed Parallel Storage System (DPSS), developed by LBL's Distributed Computing Group, provides a data source that improved multi-streamed TCP performance by striping the network streams across multiple hosts and network adaptors [5]. The TCP window sizes, buffer sizes, and blocking parameters were carefully tuned to maximize performance. In order to achieve complete overlap of computation and I/O, the network performance must be extremely high; hence the aggressive use of TCP multi-streaming. At SC00 in Dallas Texas, we were able to achieve peak throughput rates of 1.5Gigabits/sec on an OC48, with a sustained rate of about 660Mbps over a 60-

minute window (*see Figure 4*). While this was sufficient to win the SC 2000 Bandwidth Challenge with nearly twice the performance of the nearest competitor, this was still only 60% (peak) and 25% (sustained) of the theoretical line rate, respectively, of the OC-48 link used during the contest. The network throughput was extremely erratic as shown in Fig 4. On a shared WAN, >50% utilization is still admirable, but these were dedicated links that were entirely uncongested. Furthermore, the aggressive tuning of the TCP parameters indicates that we cannot idly blame the "wizard gap" for this performance deficit. So even with tuning and a dedicated link, we cannot efficiently exploit the link bandwidth using the TCP protocol.

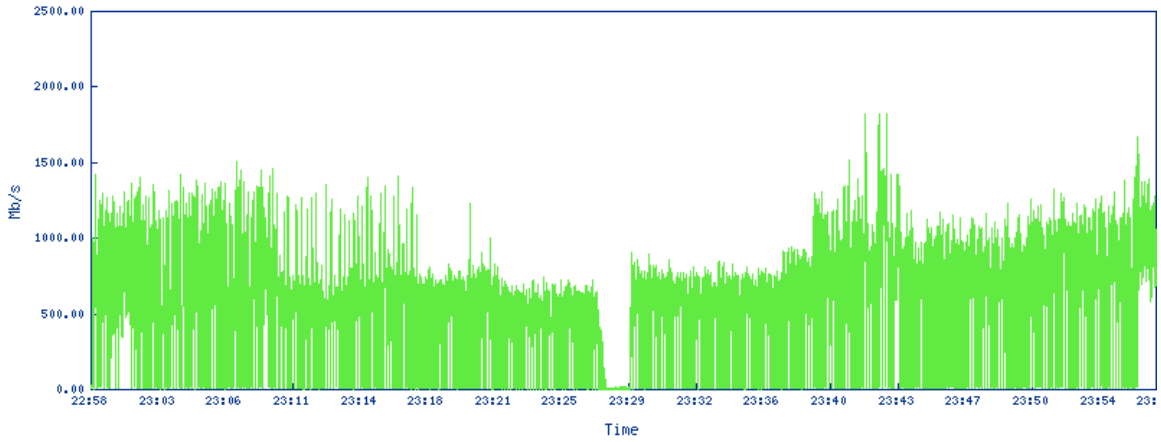

Figure 4*: Graph of network throughput for Visapult during the SC2000 Bandwidth Challenge. The performance was extremely erratic over a sixty minute time span, despite a nominally dedicated network connection and use of 32 parallel TCP streams to mitigate the effects of packet loss. This points to the extreme sensitivity and instability of TCP congestion control in practice.*

Moving to Unreliable Protocols

For data transfer and replication, data integrity is paramount. Response time and performance is of comparable importance to data integrity for visualization applications. Visualization tools almost invariably use reliable transport protocols to connect distributed components, since there is a general concern that lifting the guarantee of data integrity would compromise the effectiveness of the data analysis. However, visualization researchers find other forms of lossy data compression acceptable, like JPEG, wavelet compression and even data resampling. Such acceptance may be due to the fact that degradation in visual quality is well behaved in these cases. Networking experts have long turned to connectionless/UDP protocols when maximum responsiveness and low-latency is required by an application. An unreliable transport mechanism that deals with packet loss gracefully and doesn't exhibit extreme visual artifacts could compete well with other well-accepted data reduction techniques. Furthermore, when tuned to fit within the available bandwidth of a dedicated network connection, the loss rates for unreliable transport are extremely small - a few tenths of a percent of all packets sent if the packets are paced to stay within the limits of the slowest link in the network path.

Visualization applications also require extremely low-latency transport in order to maintain interactive responsiveness. Consider also that the response time for TCP is 2x(RTT+HostProcessingLatency) at a minimum whereas UDP offers responses that are simply RTT+HostProcessingLatency in the worst case. TCP responsiveness gets far worse when retransmission and data buffering occur. During retransmissions induced by packet loss, stream-oriented protocols will block the data stream until a successful retransmission occurs. Retransmissions result in huge variations in responsiveness and throughput, such as those observed during the SC2000 bandwidth challenge. A well-designed UDP protocol, by contrast, provides the lowest possible latencies over the WAN with little buffering delays and nearly immediate response.

Visapult Implementation Using Unreliable Transport in the SC01 Bandwidth Challenge

Rather than using static data stored on the DPSS as the data source as in our SC00 implementation, we connected Visapult directly to the Cactus code. Cactus' modular code components are referred to as "thorns". We developed a custom "thorn" for Cactus that sends data to the Visapult back end to support visual remote monitoring of executing codes. Combined with the Cactus web-based remote steering interface, Visapult can be used as a visualization component of an interactive remote steering system composed of Grid-distributed components [3].

For the SC01 Bandwidth Challenge, we modified the back end of Visapult to work with our own custom UDP transport protocol. The thesis of our work was that use of a connectionless protocol would produce dramatically more efficient bandwidth utilization as well as more consistent and rapid responsiveness. The Visapult reader used in the SC00 Bandwidth Challenge, using a TCP protocol, requested data from the network in a specific order, and TCP guarantees in-order delivery of data. In contrast, UDP makes no such guarantees, so each UDP packet must contain information indicating the location in the domain-decomposed array where the data payload must be placed by the Visapult back end (*see figure 5)*.This ensures that each packet can be treated independently so that packet ordering and packet loss have minimal effect on destination processing. The Visapult back end was modified to render continuously rather than waiting for all packets in a given frame to arrive. The visual effect of this choice is an immediate response with progressive refinement of the image over time. Data from the previous frame was used to "prime" the receive buffer and fill in gaps where packets were lost.

We contemplated a multi-buffering scheme to allow all possible packets to arrive before rendering. In practice, such a technique would provide improvements in visual quality when the data in successive frames is changes rapidly. However, due to the slow rate of evolution of the data for our remote monitoring application, the single-buffered approach proved effective as a visualization tool. This sort of multi-buffering continues to be an area of investigation.

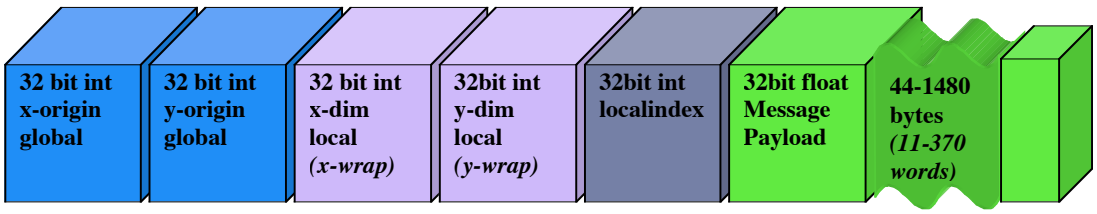

Figure 5*: Diagram of the UDP packet format. A 20-byte header was used to locate the payload of each packet in the destination domain completely independently of one-another. Assuming a full 3D block domain decomposition, the origin and dimension information is superfluous in the z-dimension.* ***X/Y origin global*** *=> the offset in grid coordinates with respect to the global data dimensions of the origin of the domain decomposed block (local domain) in the data source.* ***X/Y dim local*** *=> The x/y dimensions of the local domain decomposed chunk in the data source (used to control wrap-around of data as it is written into the destination data array).* ***Localindex*** *=> The offset in cells counted from the start of the local domain decomposed block in the data source. This source-based data indexing allows the component sending the data to be ignorant of the domain decomposition at the destination, but provides enough information at the destination to support unambiguous data re-assembly.*

The Cactus/Visapult thorn (AlphaThorns/ShmServ) buffers data in a shared memory staging area that is created when the code starts up. A set of background worker processes (NetWorkers) are spawned at startup time that are dedicated to reading data out of the shared memory region and sending it over the network to the Visapult back end. We chose to implement the NetWorkers as processes rather than threads in order to get around the performance problems and scheduling overhead associated with some particularly poor vendor implementations of pthreads. The NetWorkers require a dedicated CPU so that they don't interfere with scheduling and execution of the primary simulation code's processes. The NetWorkers have parameters that allow the code set a fixed packet rate that can be tuned to prevent

oversubscription of resources and thereby minimize packet loss. At least one asynchronous NetWorker is required per network interface card (NIC), but we also found that multiple NetWorkers per NIC were required to maximize GigE NIC utilization on larger SMP's like the 16-way IBM Nighthawk SP nodes. [Fig 6]. For our Dual-CPU Linux hosts, one processor was dedicated to network I/O while the other performed computing functions. Under this arrangement, we could achieve 960Megabits/full-line-rate per node. In July 2002, we demonstrated full utilization of a 10Gigabit Ethernet pipe using a modestly sized (12 node) Linux cluster for the Visapult backend component (http://www.supercomputingonline.com/nl.php?sid=2252)

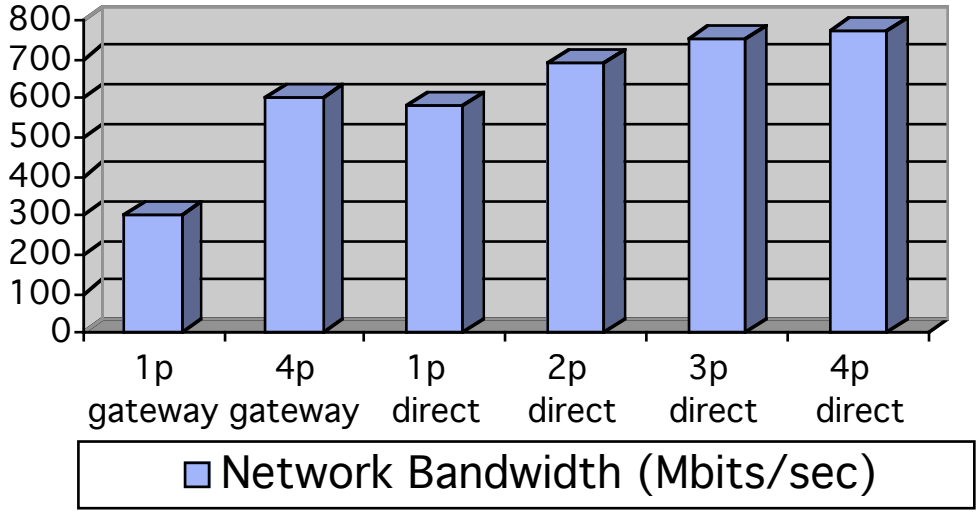

Figure 6: UDP performance on the SP2 using the internal network (sending through GigE on the Gateway node (1p-gatway and 4p-gateway mean 1processor on one node and 4 processes on 4 nodes routed through the gateway node respectively) and 1-4 NetWorker processes (1p-4p direct) on 16 CPU Nighthawk nodes with directly-attached GigE NICS. There is considerable benefit in using directly attached GigE NICs and some moderate benefit to feeding the network interface using multiple processes on each node.

SC01 Bandwidth Challenge Results

This past year at SC01 in Denver, we conducted another high performance Visapult run as part of the Bandwidth Challenge. This run made use of the new UDP transport protocol between the data source and the back end of the Visapult application. For the data source, we ran a binary black-hole merger calculation using the Cactus code on 6 nodes of NERSC's IBM SP-2 system. We also ran a related apparent horizon finder on a 128node Origin 2000 system at NCSA. At NERSC, the GigE interfaces on 5 of the SP-2 nodes connected to a dedicated OC-48 link provided by Qwest and the remaining NIC connected to an OC-12 link provided by ESNet. The Visapult application itself ran on an 8-node Linux cluster on the show floor that was connected to a 10Gigabit Force10 switch. At NCSA, a single GigE on the host fed the NCSA OC-12 uplink and fed a 32-way Sun Starfire SMP system in the Sun booth on the showfloor.

The NERSC system alone was able to reach throughput rates of 2.38Gbps on the OC-48. That is 96% of its theoretical capacity and reached this rate without the typical ramp-up associated with TCP-based applications (*see figure 7*). The total aggregate throughput of the application from all sources was 3.3Gigabits, and was the winning performance number at the SC01 Bandwidth Challenge. The visual quality and interactivity of the application was greatly improved by the high average throughput sustained by the application's data feed. The visual quality of the volume visualization was not compromised by the unreliable transport method (*see figure 8*). The gains in interactive performance were very dramatic. While there were some lost packets, the loss rate was easily tailored to be less than 1% by throttling the

packet-sending rate. The resulting artifacts when packets were lost were rather innocuous, but obvious to users who might study the image for them. However, the extremely high bandwidth efficiency resulted in rapid recovery from these artifacts when they did indeed occur. Overall, the move to unreliable transport greatly enhanced the effectiveness of the application for remote analysis of extremely large/dynamic datasets.

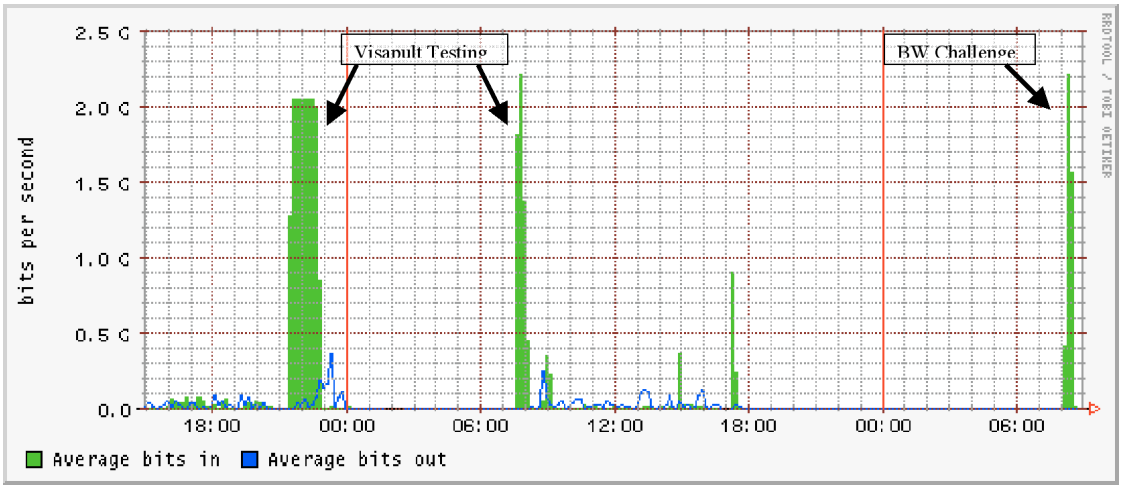

Figure 7: *The daily network traffic log from SciNET Core1 router during the Bandwidth Challenge that monitored the OC-48 traffic from NERSC. All three network traffic peaks that exceeded 2.0 Gigabits are from Visapult testing (the last being the actual bandwidth challenge run). This traffic, combined with another OC-12 from NERSC and an OC-12 from NCSA via Abilene resulted in a sustained performance exceeding 3 gigabits.*

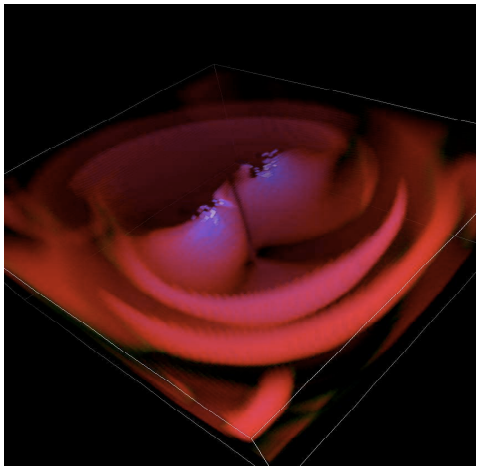

Figure 8:*Volume rendering of the real component of gravitational potential (psi4 of the Weyl tensor).*

Discussion / Future Directions

The UDP method we have described so far relies on pacing of packets to meet, but not exceed, network capacity. The are some concerns that selecting an appropriate packet rate for UDP-based methods is too tedious to be practical. However, the same tuning must occur even for multi-stream TCP implementations. Using too many TCP streams can quickly lead to the same congestion situations that occur with UDP [8], and unlike the UDP method, there is no straightforward method to regulate the bandwidth utilization of multi-stream TCP flows. Even with UDP, we can incorporate TCP-friendly methods described by Mahdavi & Floyd [7] to provide guidelines for appropriate packet rates that limit impact on other network users.

There is no reason that these methods cannot be applied to reliable transport protocols like TCP as well. We refer to TCP/reliable transport without congestion control as "TCP Brooklyn" (a TCP implementation that doesn't back-off in the face of packet loss). It is clear that fixed-rate implementations of both reliable and unreliable protocols would be disruptive to commodity networks, and are most appropriate for use on dedicated network links, Private Virtual Circuits (PVC's), Experimental Networks, or even for scheduled access. Rate-limited flows will prove essential for Data Visualization Corridors and Ultrascale Grid visualization architectures of the future. Such dedicated bandwidth is more typical in high performance networks for the sciences like ESNet, Abilene, and CANARIE. We believe that Grid-based bandwidth brokers such as those proposed for the Quanta QoS project [20] can be used to expand fixed-rate transport services to a wider variety of configurations; even posting QoS feedback on an Globus MDS (Metacomputing Directory Service).

More dynamic environments require continuous adjustment of the data rates. Multi-streaming TCP techniques offer no advantage over UDP protocols in this regard as their performance advantage comes from responding slowly to packet loss; even in cases of actual congestion. We argue that it is possible to craft a congestion control algorithm that models the behavior of multi-stream TCP using a only single stream by simply modifying its response to loss or, as in the case of Web100 Work-Around-Daemon (WAD), using artificially large virtual-MTU's to recover more quickly from packet loss [6]. Ideally, we would like to see our network switching fabric provide detailed QoS hints through informational packets to the endpoint hosts to indicate ideal send rates. Only the switching fabric can provide the necessary information to help the end-points differentiate congestive from non-congestive packet loss. We could, for instance, have a TCP or UDP implementation that uses these hints to ignore packet loss if the switching fabric says that it is non-congestive, but defaults to the standard congestion avoidance algorithm when no such hints are available. Even without intelligent switching fabric, we could create a system of peer-to-peer feedback/auto-negotiation by having end-points multicast their path and current packet-rate information on a fixed set of designated paths. This allows hosts to negotiate amongst themselves for appropriate packet rates rather than involving a third-party, like a bandwidth broker or the switching fabric itself.

The primary area of growth in considering custom UDP protocols is in the development of fault-tolerant/fault-resilient encoding techniques. The simplest approach provides fault-tolerance by copying data from the previous time step to fill in lost data. A more advanced methodology, could use a wavelet or frequency domain encoding of the data so that any loss is hidden in missing spatial frequencies (similar to JPEG compression). For transport of geometric models, we can look at packet encodings that support progressively refined meshes using triangle bisection [10]. Such techniques make packet loss less visually distracting and eliminate the need for data retention on the sending side. Any reliable technique requires data to be retained at the source until its receipt is acknowledged. Given the large bandwidth-delay-products involved for future "terabit" networks, the window sizes necessary for reliable transport will be considerable. The buffering required to support TCP retransmission and large windows creates noticeable lag in the responsiveness of remote visualization applications and produces low bandwidth utilization rates. Fast response times are essential for creating the illusion of locality so low-latency connectionless techniques will be essential for Grid visualization and collaborative interfaces. Overall, there are many avenues to consider for information encoding that make performance enhancing, unreliable delivery methods offer graceful degradation of visual quality in response of packet loss rather than simply settling for degradation in interactivity.

Conclusions

The movement to well-behaved fault-tolerant UDP-based protocols is a significant area of exploration for the future development of effective Grid-enabled visualization tools and distributed visualization component architectures. Such aggressive methods are necessary to overcome the limitations of the aging TCP protocol for high throughput applications on high speed Wide Area Networks. Use of such techniques have produced a 300% improvement in I/O efficiency over the best available tuned, multi-streaming TCP methods. While packet delivery is not guaranteed, the results are comparable to other lossy data reduction techniques commonly employed in visualization. In addition, connectionless methods offer much lower latency and better responsiveness than TCP streams under the same conditions. Ultimately, the architects of the Grid must re-evaluate exclusive reliance on TCP-based reliable transport for distributed interactive

applications like visualization because it is greatly impeding our ability to exploit high performance network-interconnected resources on the Grid.

Acknowledgements
This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. The Cactus Team: Especially Ed Seidel, Gabrielle Allen, and Peter Diener; Chip Smith for setting up the cluster. Force10 Networks for making such an awesome network switch and Raju Shah for making it work. Mike Bennett and John Christman of LBLNet for finding and testing this switch. Jim Ferguson, John Towns and Wayne-Louis Hoyenga at NCSA-University of Illinois for use of their O2k array. ESNet and Qwest for setting up the OC-48 for the bandwidth challenge. SciNET and Eli Dart who worked tirelessly to rewire the NOC (multiple times) over the course of Supercomputing 2001. And finally to NERSC for use of their enormous SP2 supercomputer and David Paul for helping fix the system to do our bidding.

Bibliography

[1] T. DeFanti, M. Brown editors, "Report to the National Science Foundation Directorate for Computer and Information Science and Engineering (CISE) Advanced Networks Infrastructure & Research Division," Technical Report, Chicago, Dec 2001. Text of the final report available at http://www.evl.uic.edu/activity/NSF/final.html

[2] W. Bethel, B. Tierney, J. Lee, D. Gunter, S. Lau.,"Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization" (LBNL-45365). In "Proceedings of SC00", November 2000.

[3] Gabrielle Allen, Werner Benger, Tom Goodale, Hans-Christian Hege, Gerd Lanfermann, André Merzky, Thomas Radke, Edward Seidel, John Shalf, "Cactus Tools for Grid Applications", Cluster Computing **4**, 179-188, 2001. (http://www.cactuscode.org/Showcase/Publications.html , Cactus info at http://www.cactuscode.org)

[4] K. Meuller, N. Shareef, J. Huang, and R. Crafis, "IBR-Assisted Volume Rendering," Proceedings of IEEE Visualization 1999, Late Breaking Hot Topics, October 1999.

[5] B. Tierney, J. Lee, B Crowley, M. Holding, J. Hylton, F.Drake," "A Network-Aware Distributed Storage Cache for Data Intensive Environments", Proceedings of IEEE High Performance Distributed Computing conference (HPDC-8), August 1999, LBNL-42896

[6] Web100 Concept Paper: http://www.web100.org/docs/concept_paper.php

[7] J. Mahdavi, S. Floyd, " TCP-Friendly Unicast UDP Rate-Based Flow Control," Technical Note, Jan 8,1997. (http://www.psc.edu/networking/papers/tcp_friendly.html)

[8] T. J. Hacker and B. D. Athey and B. D. Noble. "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network". To appear in the *Proceedings of the 16th International Parallel & Distributed Processing Symposium*, April, 2002, Fort Lauderdale, FL. (http://mobility.eecs.umich.edu/papers/ipdps02.pdf)

[9] Quanta Testbed: QoS on high performance optical networks: http://www.evl.uic.edu/cavern/teranode/quanta.html

[10] H. Hoppe, "Progressive Meshes," *Proc. 23rd Int'l. Conf. on Computer Graphics and Interactive Techniques SIGGRAPH '96*, ACM, New York, NY, 1996, pp. 99-108.