Practical Resource Management for Grid-based Visual Exploration

Karl Czajkowski Alper K. Demir Carl Kesselman Marcus Thiébaux

Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
E-mail: {karlcz,demir,carl,thiebaux}@isi.edu

Abstract

Computational grids are enabling collaboration between scientists and organizations to generate and archive extremely large datasets across shared, distributed resources. There is a need to visually explore such data throughout the life-cycle of projects. Practical exploration of large datasets requires visualization tools that can function in the same grid environment in which the data is created and stored. Resource management interfaces are an important structural component of grid computing environments because they enable uniform access to the wide variety of resources necessary for scientific work. We describe a new advance-reservation system *for graphics resources; and an application of existing grid technology to create general-purpose* active storage *systems. We report our experience with prototype infrastructure and application components, involving experiments coupling end-to-end resources for interactive visual exploration of large data in representative distributed environments.*

1. Introduction

Grid environments [21] are increasingly being used by applications such as particle physics [22], climate modeling [3], or astrophysics [4]. These applications make use of unique, high-end supercomputers and storage-systems and produce large multi-dimensional data sets. This type of work is increasingly being performed in collaborative efforts between geographically distributed scientists and organizations, utilizing shared resources that are also widely distributed [36, 25, 7].

Throughout the life-cycle of such projects, there is a need to visually explore the resulting data. Motivations include *visual validation* during algorithmic prototyping and implementation, *visual evaluation* of results while exploring the parameter space of the simulation, and *browsing and mining* of full-scale results while exploring the coordinate space of the results. To integrate such visual methods into the scientific process, graphics tools must be developed to function with grid resources [18].

There are important motivations for making visual tools grid-aware. Computations often generate and store data at locations remote from the visualization user. For exploratory tasks, it is inconvenient for users to manually transfer remote datasets to their local visualization host prior to browsing. Also, remote datasets in the grid are likely to be large—the user may not be able to store the data locally, and browsing may only require sparse selections of data. It is also important to note that grid-aware tools can also exploit additional capabilities offered by remote resources. In general, programming with grid interfaces enhances the portability of an application, increasing both availability and aggregate capacity [39, 10].

Interactive access to large, remote datasets requires reliable, high-performance interfaces. Grid environments are dynamic, so visual tools must also be easy to switch from one resource to another. For some large-data or high-performance visualization tasks, tools must access remote computers as well as data. Finally, most users have limited desktop resources, and visual tools may need to make use of remote or centralized graphics resources.

To access grid resources, visual tools must utilize *resource management* (RM) interfaces. We have defined and prototyped two new RM interfaces that, when combined with existing infrastructure, permit end-to-end management of distributed visualization tasks. First, we generalize existing notions of application-aware storage, such as ADR [26], to provide an appropriate grid-level *active storage* resource. Second, we provide *advance graphics reservations* to facilitate sharing of centralized graphics accelerators and displays.

In the following sections, we focus on large-scale *visual browsing*—interactive exploration of multi-dimensional data at varying levels of detail—as a representative application that can utilize grid resources. We examine in more detail the kinds of resources and RM interfaces needed to enable end-to-end management of grid-based visual tasks.

Finally, we present experiences from running our own

0-7695-1296-8/01 $10.00 © 2001 IEEE

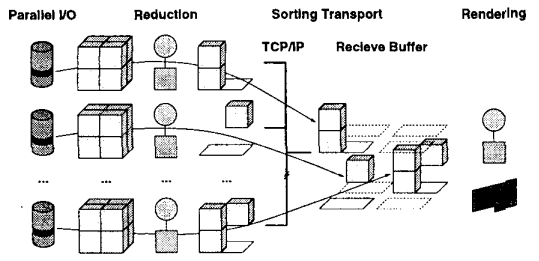

Figure 1. Visual reduction datapath channels parallel I/O and computing into graphics systems.

browsing application for volumetric time-series data in a distributed grid testbed. Our application experience shows that it is feasible to saturate contemporary graphics hardware with data obtained over wide-area networks using distributed resources. This work illustrates the viability of our new RM interfaces in executing distributed visualizations. Observed wide-area performance variation also suggest the need for new RM interfaces supporting network quality-of-service (QoS) in production grids.

2. Visual Browsing Datapath

There are many visualization techniques that can benefit from distributed resources using simple client-server mechanisms. Large-scale parallel renderers can be hosted on supercomputers to serve remote display clients [24]. Similarly, conventional rendering techniques can be coupled with remote storage services to view remotely homed data [18]. To study the limits of grid RM systems, we instead look at a more distributed exploration problem.

A large-data exploration problem can be characterized as a *reduction* datapath wherein bulk data from high-speed storage systems is reduced with parallel algorithms and fed to a graphics system for final rendering (Figure 1). Such datapaths are easily adapted for grid execution because they permit streaming primitives, masking latency over high-bandwidth networks. There are many well known techniques that can be applied during the reduction stage.

We present results using our own distributed volumetric-data browser with grid-based resources, but the utility of a reduction datapath is not unique to the dithering and splatting algorithms used in this application. Examples of common reduction methods include resolution down-sampling, iso-surfacing, field slicing, and even parallel rendering—in which case the final rendering might be a compositing process. Recent work by others has improved the performance of the popular Visualization Toolkit (VTK) [32] on parallel hardware and adapted some algorithms for out-of-core execution [1, 27]. Programmers using VTK commonly construct multi-stage, reduction datapaths.

There are many usage scenarios for a visual reduction datapath, benefitting from different ranges of distributed resources. To name a few:

- Unstructured browsing. A user can operate a GUI and manually control datapath parameters that select regions of data and filtering options. This technique is most sensitive to datapath latency and performance predictability.

- Animated browsing. When one or more browsing parameters are changed in an automated or continuous fashion, the application can predict requests to mask path delays. This variant allows tasks to more effectively pipeline and buffer operations than with unstructured browsing.

- Batch processing. Movie generation and other high-throughput problems can easily utilize distributed resources with very loose synchronization.

In this article we focus on exploratory browsing. As described in Section 1, interactive examination of data is useful throughout the life-cycle of large scientific projects. For such interactive scenarios, responsiveness is important. For large-data visualization, response time is affected by both message latency in the datapath as well as the size of work units.

When tools are stressed to their limits by users with large data, the time it takes to process or render data may dominate the total response time. While large work units allow one to minimize control overhead, a stream of smaller work units permits more effective user interaction. Thus our choice of browsing tasks focuses our interest in distributed resources. We require high-bandwidth resources that can be coupled to produce responsive work streams.

3. Important Resources

In order to distribute the components of our browsing datapath, we need to access a variety of resources. Grid computing environments provide scheduling and allocation for several of these, but highly distributed visualization datapaths also may need to access more specialized graphics hardware for which there are no extant grid interfaces (Figure 2). The resources we are interested in include:

- compute resources
- storage systems
- graphics equipment
- networks and switches

To be accessed by applications, grid resources must be exported through RM interfaces that support resource allocation and configuration. In practice, grid environments must facilitate the *discovery* of resources too, so that application and users need not manually identify every remote resource. However, we focus here on the core allocation and configuration problem.

In this section we describe these resources and survey existing grid RM interfaces, where available.

3.1. General-purpose processors

General-purpose processors are important to execute data reduction tasks in the visualization pipeline. Such desktop and parallel computers have received the most attention in grid environments. There are interfaces to schedule, submit and control jobs remotely with varying levels of generality, security, and other features. Condor [31] provides support for high-throughput tasks, and is mostly suited to movie-generation and other batch visualization problems. Legion [11] and PBS [38] provide support for certain types of automated job placement across multiple resources. We have considerable experience deploying and using resources with the Globus GRAM job-submission interface; GRAM provides a single mechanism to start jobs on a wide range of remote computer types, and permits higher-level components to plan job placement and/or coordinate distributed tasks [14, 10, 15].

3.2. Online storage systems

Random access to large data requires *online* storage systems. These systems are particularly interesting for interactive visual exploration since they allow users to browse large amounts of data in an unstructured manner. Existing remote interfaces to online storage systems include high-level schedulers such as SRB [5] and HPSS [23], as well as block-level interfaces such as DPSS [37], FTP, and NFS. The bandwidth, latency, and request granularity of an online storage interface all affect the responsiveness of applications using their services.

3.3. Graphics accelerators

Graphics rendering algorithms can often take advantage of hardware accelerators. Parallel accelerators have been used to render large data on SGI Onyx2 systems [28], and the Pomegranate [16] system proposes an architecture for scalable (clustered) acceleration. For practical resource sharing, there is a need to mediate access to the hardware by both local and remote users. However, we do not know of any existing grid-level interfaces to manage graphics resources.

In typical graphics server platforms, system administrators must make static decisions about how to logically

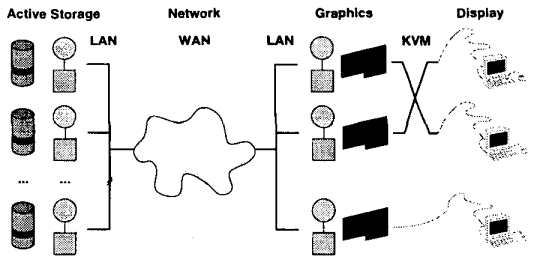

Figure 2. Important resources for grid-based visualization.

partition graphics hardware [30, 34]; graphics *sessions* are scheduled by users approaching a console and logging in. The length of a session is controlled by the user applications that are launched at login time, and may continue indefinitely. Commercial SGI software provides an environment for running OpenGL tasks on an acceleration server with a virtualized display/input device on a remote host within the local organization [29]. In the following sections, we describe the use of our own new graphics RM service.

3.4. Networks

To plan interactive visualization datapaths, we must be able to predict communication performance between resources. In addition to the discovery problem that exists for all resources, we face the problem that the communication network itself is a shared resource. To insulate applications from unexpected performance variations (due to changes in the behavior of unrelated applications), researchers have developed a concept of Quality of Service (QoS) guarantees.

Various mature network QoS allocation schemes exist, such as Intserv/RSVP [9], Diffserv [8] and ATM [35], but they do not provide secure and generic grid interfaces for end-users. The prototype GARA advance-reservation system provides a grid interface to underlying network managers such as Diffserv, as well as other non-network resource managers [20]. The Condor high-throughput scheduler can manage network resources for its jobs but does not interact with underlying network managers to provide service guarantees [6]. Due to a lack of QoS interfaces in production networks, we have been unable to benefit from network service guarantees in our visualization experiments thus far.

3.5. Graphics displays

In resource centers with multiple visualization users, it is common to find shared display devices. These displays may be shared because they are expensive or special-purpose, e.g. CAVE and ImmersaDesk [12] installations, or because

they are attached to expensive, shared graphics accelerators such as are described in Section 3.3. Another common example is video projectors in A/V-equipped conference and seminar rooms. In typical environments, such shared displays are either managed manually by site administrators or made available on a first-come, first-served basis through simple, local mechanisms. We address this issue in the following sections while describing our new graphics RM service.

4. New Interfaces

In the course of our visualization experiments, we have found it useful to define new RM interfaces for parallel *active storage* systems and shared graphics resources.

We obtain active storage systems by combining online storage with general-purpose processors. These systems are capable of storing large amounts of application data in a persistent or scheduleable way, while simultaneously hosting domain-specific or application programs that can be parallelized to scale with the available data size and bandwidth. These systems are realized by clustered computers with distributed disks, or with parallel computers connected to local or system-area storage.

The active storage model enhances the utility of large-scale storage equipment. Our visualization datapath is not hindered by overly-general access protocols that only permit bulk transfer of raw data. By executing our reduction algorithm at the storage system, we are able to optimize the network-bound data for the viewing parameters selected by the user.

Several domain-specific active storage abstractions have been previously published, such as the ADR [26]. Such abstractions do not conflict with our notion of an active storage system; rather the software that responds to domain-specific requests is part of the schedulable application in our model. This simplifies resource deployment, as the operator need only configure base RM components and policy elements to allow multiple application groups to share the resource.

4.1. Active storage manager

We have defined a minimal grid interface for active storage systems that consists of the Globus GRAM job-submission interface [14] and the normal UNIX filesystem. In the Globus RM architecture, GRAM provides resource access but resource discovery is left to complementary services such as the MDS information system [13, 17, 19, 21], which is an ongoing research effort. Related work on data catalogs and storage-system directories include other services that will simplify grid-based visualization [2]. For the experiments described in this article, we utilized *ad hoc* discovery and dataset management, in anticipation of improving grid information services.

While it had been argued that high-performance storage and computational systems should export rich advance-reservation capabilities in the grid [20], there are interim solutions that allow immediate application deployment. These alternatives enable practical work while new grid capabilities are being developed. For space-sharing job submission, we dedicate disk bandwidth implicitly as part of each processing node; for time-sharing job submission, disk bandwidth can be allocated by a locale-specific interface such as guaranteed-rate I/O (GRIO) [33], though this admits certain scheduling deadlock risks that would be avoided with advance reservation.

Our visual browsing application uses the DUROC co-allocation library [15] to automate interaction with multiple GRAM resources. On clustered storage, the application explicitly manages data and task layout. On centralized storage, the host scheduler places tasks.

4.2. Graphics session manager

We have constructed a session-based grid interface to graphics systems that combines advance reservation with dynamic logical re-partitioning of hardware. A remote client makes a reservation of a particular set of accelerators for a particular period of time, and the resource manager automatically invokes the session software at the selected time. So far, we have identified three typical usage scenarios:

1. Public terminal. The session requires password re-authentication to prevent unauthorized use of the console.
2. Secure terminal. With grid single sign-on security, the session can start directly if the console is physically secured.
3. No terminal. The requested session runs with null device inputs either as a batch program or a remotely-controlled application.

These scenarios are distinguished only by minor variations in how the session software is initialized.

In general one might want to separately reserve an accelerated graphics session, a display device, and the required video data path and then bind them together close to the session time. However, our RM system does not address virtualized displays so the managed displays are always part of the local physical resource pool. We chose to add a "seat" location attribute to the session management interface described in the previous section, so the accelerator, display, and M×N *keyboard/video/mouse* (KVM) switch path are all assigned in one request.

Our session manager extends the GARA generic grid reservation system [20] with the logic needed to control the X Windows server tasks as well as the command-channel of our KVM switch.

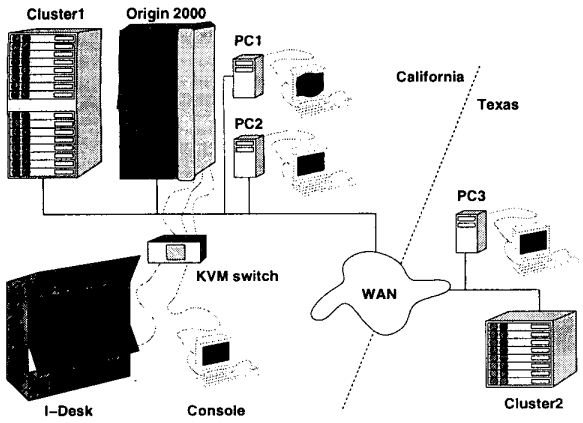

Figure 3. Testbed layout for reported results.

5. Experimental Testbed

In this section we describe experimental resources we have deployed in order to gain experience developing and executing distributed visualization tasks (depicted in Figure 3 and Table 1). These resources combine existing and new RM interfaces described above.

5.1. Active storage servers

We have experimented with multiple active storage servers providing parallel disk-I/O and parallel processing:

- 10-cpu SGI Origin with 300 GB disk array.
- 32-cpu Linux cluster with 1.5 TB of disk.
- 8-cpu demo Linux cluster with 1.5 TB of disk.

The Origin storage is not distributed, so we operate a single GRAM *gatekeeper* that allows the submission of parallel jobs. For the Linux clusters, we operate a GRAM gatekeeper on each node so that our applications can place tasks however desired (see example request in Figure 4). While the 16-node cluster is a permanent installation in our lab, the 8-node cluster was a demo machine operated on the show-floor at the SC2000 conference (Dallas, TX, November 2000).

Because of cooperative policies in our testbed, there is little competition and we obtain essentially dedicated access to the SMP and cluster-based active storage with just immediate-access time-sharing managers. We are participating in the configuration of a much larger cluster that will serve as a testbed for shared access with new grid-based QoS and advance reservation services.

Name	CPU cnt	CPU MHz	RAM MB	Net Mb/s	Disk GB
Cluster1	32	550	4096	2000	1500
Cluster2	8	550	2048	2000	1500
Origin	10	200	4096	1000	300
PC1	2	450	256	1000	n/a
PC2	2	933	384	100	n/a
PC3	2	933	384	1000	n/a

Table 1. Hardware properties of testbed resources.

```
+ (&(manager=node1)
    (executable=filterprog))
  (&(manager=node2)
    (executable=filterprog))
  ...
  (&(manager=nodeN)
    (executable=filterprog))
```

Figure 4. Active storage request to a small cluster.

5.2. Graphics resources

We operate multiple RealityEngine2 accelerator boards in our SGI Origin storage system. Connected to our SGI (via a KVM switch) is an ImmersaDesk virtual-reality display and a wide-screen desktop console. In addition, we operate multiple commodity personal-computers with recent consumer-level NVIDIA graphics accelerators. The SGI and several PCs have gigabit-Ethernet network interfaces, while one PC has an older 100baseT Ethernet interface. These resource serve to represent typical visualization facilities.

Our session manager environment allows users to reserve one or more SGI boards for an X Window session. They also must reserve the desktop or ImmersaDesk seat, or run without a console for off-screen rendering tasks (see example request in Figure 5). The implementation is configured with a table of local resources that are subject to management by the service. A "slot manager" keeps track

```
&(manager=onyx2)
 (pipecount=2)
 (sessiontype=public)
 (seat=idesk)
 (starttime="13:00 PST")
 (duration="2:00")
```

Figure 5. Graphics session request for a local console.

of what resources are assigned over future time intervals. As the current time approaches the beginning of an allocated slot, session construction is triggered. Conversely, as the current time approaches the end of the allocated slot, session destruction is triggered.

Session construction involves configuring the KVM switch and starting the session software. The KVM switch is automated via a serial management port. Session destruction involves resetting the KVM switch and killing the session software to bring hardware back to an idle state.

6. Application Experience

For experiments, we have constructed an interactive browser for volumetric time-series data. The application uses a reduction path as discussed in Section 2, searching the data fields on the active storage system to generate sparse geometry representing the regions of the field matching a user-specified range of value.

Our application automates acquisition of the storage resource, and the user acquires the interactive graphics resource in order to launch the application GUI. The user either reserves the graphics resource in advance using our GARA service or logs into a regular X server console, depending on whether it is our managed Origin or a PC.

6.1. Transport measurement

Our application provides a fully buffered, streaming datapath. The local rendering can run faster than the data transfer rate, allowing motion-study of one dataset while another is being received. Buffered data is nearly always saturating the graphics hardware independent of datapath performance. Transfer rate is a better measure of datapath performance, though it is capped by graphics hardware capability (Tables 2 and 3).

The application GUI allows the user to tune filtering parameters for the desired trade-off of visual complexity for transfer or rendering frame-rate, and in practice it adapts well over burst (5-second average) transfer rates in the range 50 Mbits/s to around 320 Mbits/s, where our current graphics resources become saturated.[1]

6.2. Desktop performance

The application is useful while consuming 100baseT desktop bandwidth (50–90 Mb/s) in both LAN and WAN datapaths. Operation at this level can utilize a wide range of graphics and active storage resources, and due to a parallel transport architecture it can often achieve good bandwidth, without network QoS guarantees and despite competition in the LAN and WAN. Such application instances benefit most from the flexibility of grid access to commodity storage and graphics resources.

[1]Our path synchronization causes a partial *phase lock* between the transfer and render rates, relaxed by variable-depth buffering and ability to render multiple times per transfer. For fixed data sizes, this causes multimodal rate distributions as the application "searches" for the best transfer rate and integer render/transfer multiplier.

Data Sink	Data Source	High Mb/s	Low Mb/s	Norm. Range
PC1	Cluster1	170	110	35%
	Origin	125	90	28%
PC2	Cluster1	96	60	38%
	Origin	40	12	39%
PC3	Cluster1	240	90	63%
	Cluster2	320	200	38%
Origin	Cluster1	190	120	37%

Table 2. Observed browser throughput with real rendering.

Data Sink	Data Source	High Mb/s	Low Mb/s	Norm. Range
PC1	Cluster1	380	215	43%
Origin	Cluster1	300	250	17%

Table 3. Observed datapath throughput without rendering limit.

6.3. High performance

The application sometimes saturates our graphics resources when it is connected to gigabit-class networks. In our SC2000 demo environment, the local show-floor cluster provided more predictable (and slightly faster) performance when we had dedicated access, but when we could not get dedicated access we were able to fall back on remote clusters. Our remote cluster in California sometimes achieved 75% of the dedicated, local cluster performance despite traversing multiple networks with best-effort service (Table 2). However, wide-area performance varied widely during the day. Simulating an increase in graphics hardware performance that shifts the path bottleneck, we are confident the application can utilize close to 400 Mbit/s (Table 3).[2]

7. Conclusion

Visual exploration of large data can benefit from grid-based resource environments. Grid interfaces give applications access to redundant resources, improving availability

[2]We simulate this increase by disabling the "draw" calls in the application loop, but continuing to perform the normal data buffering. We have not yet tried to tune TCP/IP or Ethernet for gigabit performance since rendering remains our primary bottleneck.

in the face of contention and failure. Applications can exploit these same interfaces to obtain higher aggregate resource capabilities, e.g. larger storage spaces, higher I/O bandwidths, and higher computational throughput. We have illustrated a simple visual reduction paradigm that, paired with any of a number of existing visualization techniques, can adaptively utilize distributed resource environments.

In addition to typical grid services, we have identified graphics accelerators and application-extended active storage as important resources deserving of grid RM interfaces. We have presented a unique grid-based advance reservation capability for high-end graphics accelerators that can both simplify usage of localized resource-sharing environments and assist in planning of distributed visualization runs. Experimentation with a grid-based visual browsing application validates the use of standard RM interfaces for active storage systems. We have found it practical to build resilient visualization environments by replicating data on multiple remote storage systems and utilizing replicas when the primary choice is unavailable.

While parallel transfer methods achieve acceptable performance for typical desktop systems in best-effort networking environments, we have seen that the lack of grid-level network QoS guarantees hinders the end-to-end management of high-performance WAN datapaths. Best-effort performance over gigabit-class WANs is not predictable on a level comparable to that of the storage and graphics resources available for distributed applications, reducing the accuracy of distributed datapath prediction and planning.

Acknowledgements

We appreciate many helpful discussions with colleagues from the Globus Project and the larger Grid community. In particular we thank Alain Roy for guidance regarding the GARA code-base, upon which part of this work is based.

This effort is sponsored by the Lawrence Livermore National Laboratory and the U.S. Government, under subcontract B505215. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon.

References

[1] J. Ahrens, C. Law, W. Schroeder, K. Martin, and M. Papka. A parallel approach for efficiently visualizing extremely large, time-varying datasets. Technical Report LAUR-00-1630, Los Alamos National Laboratory, 2000.

[2] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *IEEE Mass Storage Conference*, April 2001. (to appear).

[3] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-performance remote access to climate simulation data: A challenge problem for data grid technologies. Submitted for publication, 2001.

[4] G. Allen, W. Benger, T. Goodale, H.-C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, and J. Shalf. The Cactus code: A problem solving environment for the grid. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, 2000.

[5] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In *Proc. CASCON'98 Conference*, Toronto, Canada, November 1998.

[6] J. Basney and M. Livny. Managing network resources in Condor. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, 2000.

[7] J. I. Beiriger, H. P. Biven, S. L. Humphreys, W. R. Johnson, and R. E. Rhea. Constructing the ASCI computational grid. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, pages 193–199, 2000.

[8] D. Black, S. Blake, M. Carlson, E. Davies, and Z. Wang. An architecture for differentiated services. Internet RFC 2475, December 1998.

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) version 1 functional specification. Internet RFC 2205, September 1997.

[10] S. Brunett, K. Czajkowski, S. Fitzgerald, I. Foster, A. Johnson, C. Kesselman, J. Leigh, and S. Tuecke. Application experiences with the Globus toolkit. In *HPDC7*, pages 81–89, 1998.

[11] S. Chapin, J. Karpovich, and A. Grimshaw. The Legion resource management system. In *The 5th Workshop on Job Scheduling Strategies for Parallel Processing*, 1999.

[12] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Proceedings of SIGGRAPH '93*, pages 135–142, August 1993.

[13] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing*, (to appear) 2001.

[14] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.

[15] K. Czajkowski, I. Foster, and C. Kesselman. Co-allocation services for computational grids. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, 1999.

[16] M. Eldridge, H. Igehy, and P. Hanrahan. Pomegranate: A fully scalable graphics architecture. In *Proceedings of SIGGRAPH 2000*, pages 443–454, July 2000.

[17] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375, 1997.

[18] I. Foster, J. Insley, G. von Laszewski, C. Kesselman, and M. Thiebaux. Distance visualization: Data exploration on the Grid. *Computer*, 32(12):36–43, December 1999.

[19] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl. Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[20] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, 1999.

[21] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Intl. Journal of Supercomputing Applications*, (to appear) 2001. http://www.globus.org/research/papers/anatomy.pdf.

[22] Grid physics network. Internet document, 2001. http://www.griphyn.org.

[23] Basics of the high performance storage system. Internet document. http://www.sdsc.edu/projects/HPSS.

[24] G. Johnson and J. Genetti. Volume rendering of large datasets on the Cray T3D. In *Cray User Group: 1996 Spring Proceedings*, pages 155–159, 1996.

[25] W. E. Johnson, D. Gannon, and B. Nitzberg. Grids as production computing environments: The engineering aspects of NASA's Information Power Grid. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, pages 197–204, 1999.

[26] T. Kurc, Ümit Çatalyürek, C. Chang, A. Sussman, and J. Salz. Exploration and visualization of very large datasets with the Active Data Repository. Technical Report CS-TR-4208, University of Maryland, 2001.

[27] C. Law, W. Schroeder, K. Martin, and J. Temkin. A multithreaded streaming pipeline architecture for large structured data sets. In *Proceedings of IEEE Visualization '99*, 1999.

[28] P. McCormick, J. Qiang, and R. Ryne. Visualizing high-resolution accelerator physics. *IEEE Computer Graphics and Applications*, 19(5):11–13, Sep/Oct 1999.

[29] C. Ohazama. OpenGL vizserver. Internet white paper, 2000. http://www.sgi.com/software/vizserver/.

[30] K. Packard. *X Display Manager*. MIT X Consortium, 2000. Online manual page.

[31] R. Raman, M. Livny, and M. Solomon. Resource management through multilateral matchmaking. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, 2000.

[32] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Prentice Hall, 1996.

[33] Silicon Graphics, Inc. *Guaranteed-rate I/O*. Online manual page.

[34] Silicon Graphics, Inc. *X Window System server for Silicon Graphics Workstations*, 1998. Online manual page.

[35] S. Siu and R. Jain. A brief overview of ATM: Protocol layers, LAN emulation and traffic management. *Computer Communications Review (ACM SIGCOMM)*, 25(2):6–28, April 1995.

[36] P. H. Smith and J. V. Rosendale, editors. *Data and Visualization Corridors, Report on the 1998 DVC Workshop Series*. California Institute of Technology Center for Advanced Computing Research, September 1998.

[37] B. Tierney, J. Lee, B. Crowley, M. Holding, J. Hylton, and F. Drake. A network-aware distributed storage cache for data intensive environments. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, 1999.

[38] Portable batch system. Internet document, August 2000. http://pbs.mrj.com.

[39] G. von Laszewski, I. Foster, J. A. Insley, J. Bresnahan, C. Kesselman, M. Su, M. Thiebaux, M. L. Rivers, I. McNulty, B. Tieman, and S. Wang. Real-time analysis, visualization, and steering of microtomography experiments at photon sources. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 1999.