High-Resolution Remote Rendering of Large Datasets in a Collaborative Environment

Nicholas T. Karonis [a,b], Michael E. Papka [b,c], Justin Binns [b], John Bresnahan [b], Joseph A. Insley [b], David Jones [b], Joseph M. Link [b]

[a] *Department of Computer Science, Northern Illinois University, DeKalb, IL 60115*

[b] *Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439*

[c] *Department of Computer Science, University of Chicago, Chicago, IL 60637*

Abstract

In a time when computational and data resources are distributed around the globe, users need to interact with these resources and each other easily and efficient. The Grid, by definition, represents a connection of distributed resources that can be used regardless of the user's location. We have built a prototype visualization system using the Globus Toolkit, MPICH-G2, and the Access Grid in order to explore how future scientific collaborations may occur over the Grid. We describe our experience in demonstrating our system at *i*Grid2002, where the United States and the Netherlands were connected via a high-latency, high-bandwidth network. In particular, we focus on issues related to a grid-based application that couples a collaboration component (including a user interface to the Access Grid) with a high-resolution remote rendering component.

Key words: MPI, Globus Toolkit, GridFTP, Access Grid, MPICH-G2

1 Introduction

Today's typical simulation involves complex structures and phenomena. To fully visualize all the detail is a challenging task, often as computationally de-

Email addresses: karonis@niu.edu (Nicholas T. Karonis), papka@mcs.anl.gov (Michael E. Papka).

manding as the simulation itself. Scientists often find that standard desktop displays do not have enough resolution to visualize the data at the resolution that is representative of the calculation. Moreover, simulations today are rarely the result of a sole investigator and are more likely the result of a collaboration that may span multiple geographically distributed sites. In response to these needs, we have developed a prototype system that uses the Access Grid to connect remote users across the campus, across the country, or across the world, in a virtual space. The Access Grid is augmented with a shared user-interface for controlling the remote real-time rendering of a low-resolution version of the datasets. Using the Access Grid to view the low-resolution images, the collaborative team decides on views, variables, time-steps, and isovalues, the team also can create a high-resolution rendering of the image for near-real-time viewing or later use.

This paper discusses our experiences in developing the prototype system and in using the system both during the *i*Grid2002 conference and in the weeks after the conference. In Section 2 we give an overview of the system, in Section 3 we describe application, and in Section 4 our performance results. In Section 5 we draw conclusions based our *i*Grid2002 experience, and we outline our plans for enhancing the system.

2 Infrastructure

The prototype system described in this paper uses a wide variety of tools and technology from small software libraries such as XML-RPC,[1] to large software systems such as the Globus Toolkit®[2] and MPICH-G2,[3] to physical infrastructure built from a collection of software and hardware components such as the Access Grid[4]. In the following subsections we present an overview of these resources while using GridFTPwhile using GridFTP.

2.1 Access Grid

The Access Grid is an ensemble of resources that supports group-to-group interaction at a distance (see Figure 1). It consists of large-format multimedia displays, presentation and interactive software environments, interfaces to Grid middleware, and interfaces to remote visualization environments. The

[1] xmlrpc-c.sourceforge.net
[2] www.globus.org. The Globus Toolkit is a registered trademark held by the University of Chicago.
[3] www.globus.org/mpi/
[4] www.accessgrid.org

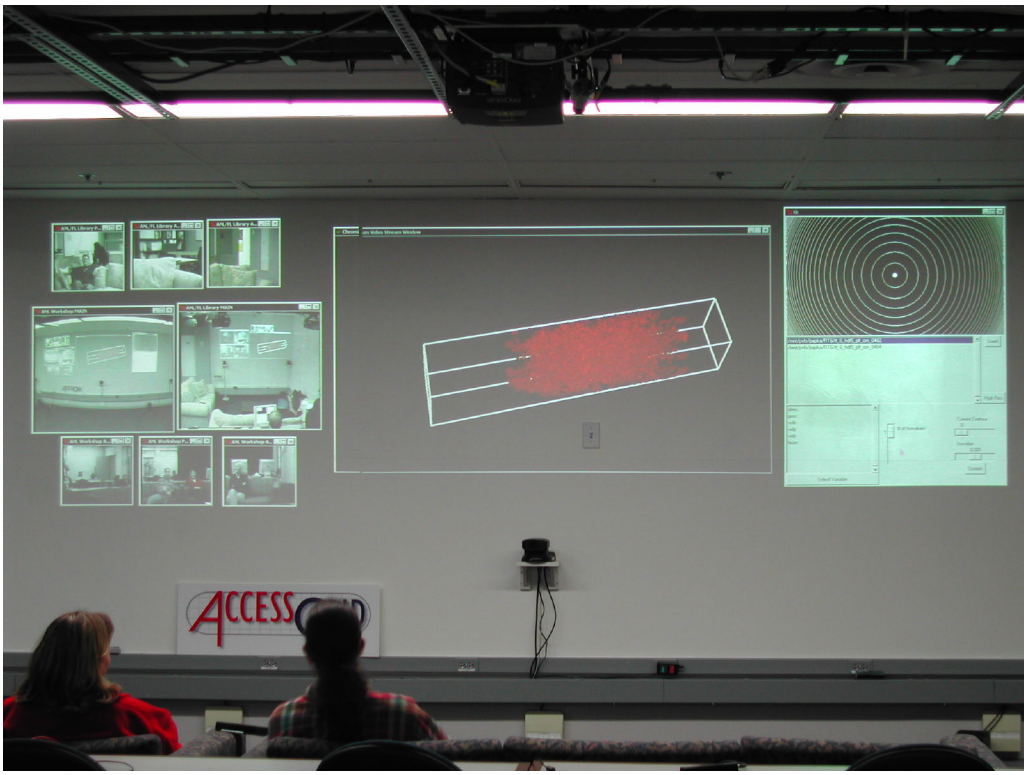

Fig. 1. Photograph of one of Argonne National Laboratory's Access Grid nodes, running the *i*Grid2002 collaborative application. Users are seated on a couch in front of the display and are able to view remote users through video windows and speak to them in a natural manner. Users at each site are able to interact with the low-resolution visualization in order to determine the proper view and values for the generation of a high-resolution version.

Access Grid enables distributed meetings, collaborative teamwork sessions, seminars, lectures, tutorials, and training. The Access Grid design point is small (3 - 20 people per site) but promotes group-to-group collaboration and communication [2]. Large-format displays integrated with intelligent or active meeting rooms are a central feature of Access Grid nodes. Access Grid nodes are designed spaces that explicitly support the high-end audio and visual technology needed to provide a high-quality compelling and productive user experience. Access Grid nodes are connected via the high-speed Internet, typically using multicast video and audio streams.

2.2 The Globus Toolkit

The Globus Toolkit is a collection of software components designed to support the development of applications for high-performance distributed computing environments, or "Grids" [4,5]. Core components typically define a protocol for interacting with a remote resource, plus an application program interface

(API) used to invoke that protocol. Higher-level libraries, services, tools, and applications use core services to implement more complex global functionality. The various Globus Toolkit components are reviewed in [6] and described in detail in online documentation and in technical papers.

We briefly describe one component of the Globus Toolkit, GridFTP [1]. GridFTP is the preferred protocol for transferring data on the Grid. GridFTP starts with RFC 959 (FTP), leveraging the ubiquity of the standard FTP protocol, and then makes backward-compatible extensions to allow for such functions as reliable restarts, performance monitoring, and coordinated multi-host transfersy necessary for successful data transfer over the Grid.

As RFC 959 describes, FTP is a two-channel protocol comprised of a *control channel* and a *data channel*. The control channel is responsible for coordinating the transfer, and the data channel is responsible for shipping the data from starting point to endpoint. An important extension to the RFC 959 provided by GridFTP is the new data channel protocol MODE E. MODE E is a reliable, ordered, parallel-stream, multihost, TCP-based protocol. Whose goal is to provide a fast end-to-end transfer protocol across wide area networks. It delivers this goal by allowing many-to-many multihost transfers and by providing many TCP streams to be used when sending from one host to another.

2.3 MPICH-G2: A Grid-Enabled MPI

MPICH-G2 [9] is a complete implementation of the MPI-1 standard that uses Globus Toolkit services to support efficient and transparent execution in heterogeneous Grid environments, while also allowing for application management of heterogeneity. We briefly describe how MPICH-G2 uses GridFTP.

Once an application has started, MPICH-G2 selects the most efficient communication method possible between any two processes, using vendor-supplied MPI (*v*MPI) if available, or Globus communication (Globus IO) with the option to utilize *parallel sockets* via Globus' GridFTP, with Globus Data Conversion (Globus DC) for TCP, otherwise.

MPICH-G2 applications that transfer large blocks of data in single messages from one process to another over a high-bandwidth network (e.g., visualization data over an optical network) may optionally instruct MPICH-G2 to use a set of parallel sockets between a pair of designated processes to better use the available bandwidth. MPICH-G2 delivers this option to its applications through the use of existing MPI idioms. As an illustration, Figure 2 depicts an excerpt from an MPICH-G2 application in which 64 parallel sockets are established between `MPI_COMM_WORLD` ranks 0 and 1. In that example both processes start by setting values that (a) designate each other

```
#include <mpi.h>

int main(int argc, char **argv)
{
    int numprocs, my_id;
    struct gridftp_params gfp; /* MPICH-G2 structure in mpi.h */

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_id);

    if (my_id == 0 || my_id == 1) {
        /* must set these three fields */
        gfp.partner_rank  = (my_id ? 0 : 1);
        gfp.nsocket_pairs = 64;
        gfp.tcp_buffsize  = 256*1024;

        MPI_Attr_put(MPI_COMM_WORLD, MPICHX_PARALLELSOCKETS_PARAMETERS, &gfp);
    } /* endif */

    /*
     * from this point all messages exchanged between
     * MPI_COMM_WORLD ranks 0 and 1 will be automatically
     * partitioned and transported over parallel sockets
     */

      .
      .

    MPI_Finalize();

} /* end main() */
```

Fig. 2. An excerpt from an MPICH-G2 application that establishes 64 parallel sockets each with a 256 KB TCP buffer between processes ranked 0 and 1.

as partners, (b) request 64 parallel sockets, and (c) request a 256 KB TCP buffer size. After setting these values, the processes place their request for MPICH-G2 by assigning their specified values to the communicator attribute `MPICHX_PARALLELSOCKETS_PARAMETERS`. This technique of using communicator caches to influence MPICH-G2's behavior was used successfully in a similar situation in which putting data into an MPICH-G2 communicator triggered a quality-of-service-enabled line [10]. Once the processes "put" their values into their communicators, MPICH-G2 automatically partitions all messages between them and sends them, using GridFTP's MODE E data channel protocol, over parallel sockets.

3 Application Overview

The application we developed as our prototype system connects two components: a collaborative component and a high-resolution remote rendering component, the two components that can either operate independently or as a coupled pair. The two pieces form an end-to-end prototype for interactively exploring large datasets and the production of high-resolution images in collaboration with colleagues who can be geographically distributed. Our application uses data produced by the University of Chicago's Center fo Astrophysical Thermonuclear Flashes Center[5]. The data is stored in well-defined HDF5 files that represent a multiresolution block structure composed of multiple cells. Blocks tend to be constructed of either 8^3 cells if the data is cell centered or 9^3 cells if the data is vertex centered. Block sizes vary in spatial dimension depending on level of refinement but always contain the same number of cells. This fact can be exploited to produce low-resolution renderings from the same dataset used in the high-resolution cases. Using only the corner data of the blocks for the low-resolution case, one can significantly reduce the amount of data needed for isosurface generation and hence the time to render. For high-resolution isosurfacing and rendering all the data available is used. The following two subsections describe each of these components.

3.1 Collaboration Component

The collaboration component (see figure 3) is constructed to work in the Access Grid environment described earlier and make use of a low-resolution remote rendering server to produce the images for the collaborative environment. Interacting with an interface that sends mesages to the remote visualization server, users can manipulate the viewpoint, number of isosurfaces, isosurface values, datasets, and dataset variables viewed. The user interface uses XML-RPC commands issued from any one of the user-interfaces participating in the session. Once a set of instructions is received by the remote rendering server, the appropriate files and variables are loaded. The low-resolution visualization server, built with the Visualization Toolkit (vtk) [11], is currently capable of generating up to 20 different isosurfaces for a given variable. The resulting isosurface(s) are then rendered by using a sort-last parallel renderering scheme with the scene decomposed as if the final image were divided into an n x m set of tiles. This decomposition is done by using the Chromium library [8], which sorts the polygonal data for rendering on the appropriate tile. Chromium is coupled with an Argonne-developed Chromium stream processing unit that creates a set of H261-encoded video streams. These video streams are then

[5] www.flash.uchicago.edu

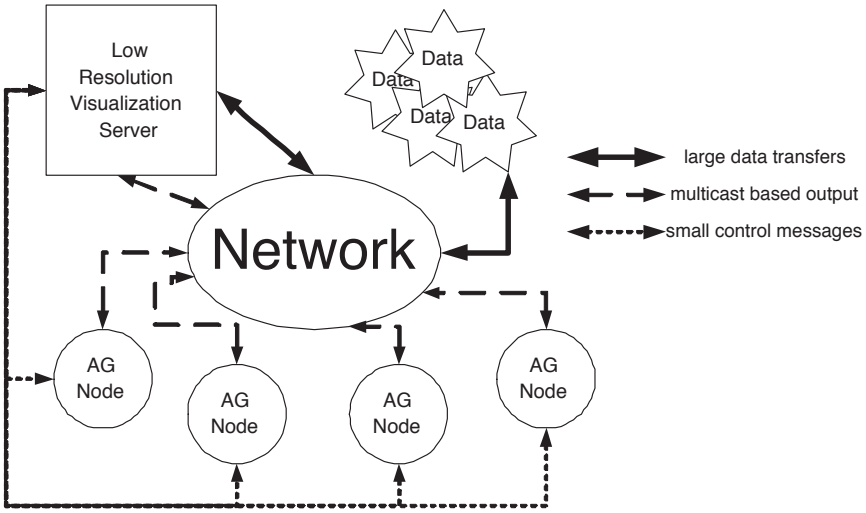

Fig. 3. The collaboration component of the prototype. Access Grid nodes interact via a user interface to select data stored on the network. The selected data is then visualized by using the low-resolution visualization server to generate video output, which is then multicast back to all interested Access Grid nodes.

multicast across the Internet to all participants. The participants then run a video client developed in conjunction with the stream generation to reassemble the multiple video streams into what looks like a single source at the Access Grid node. Both the video generation and video reassembling components can be tiled in any n x m combination, although they must match. When used in conjunction with an Access Grid meeting, a typical configuration is 3 x 2 yielding an overall resolution of 1056 x 576.

3.2 High-Resolution Remote Rendering Component

The high-resolution rendering server is a parallel visualization server prototype that facilitates the generation of isosurface(s) using the full dataset and the generation of images from the surface(s) (see Figure 4). These images can be saved to a file for later viewing or streamed in near-realtime to a tiled display to be viewed when the image is ready. Both of these modes were used, during the *i*Grid2002 conference. The pieces of the final image were sent to Amsterdam along with zbuffer for compositing on the cluster there. As new pieces arrived they were merged with pieces already in place. This strategy gave viewers a progressive final image displayed on a four- tile LCD wall. Subsequent tests have used the "image saved to a file" mode, since the tiled display is no longer in place.

The remote server receives its input of camera values and visualization parameters either from the command line in stand-alone mode or from the collaboration application in coupled mode. Once this information is received, the

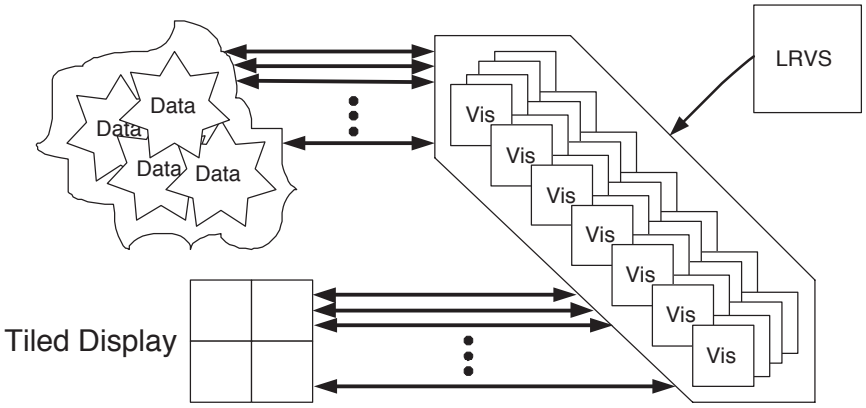

Fig. 4. Diagram of high-resolution component of the application. After users have finished interacting with the low-resolution version of the data, they are able to send a message from the GUI to the waiting high-resolution application in order to initiate the generation of a new image. The new images during *i*Grid2002 were 2048 x 1536. The distributed/parallel visualization server receives data in parallel from the same data location as the low-resolution component. The data was then processed and sent by using MPICH-G2 enabled with GridFTP over the wide-area connection to Amsterdam for compositing and display on EVL's tiled LCD display.

data is loaded and isosurfaced in parallel, then the rendered subpieces and zbuffer sent are to the compositing component. The compositing component can either be collocated with the visualization/rendering component or distributed across the network as done in this case. The compositing component is responsible for merging multiple framebuffers together by looking at the contents of the zbuffer. The prototype does not use the alpha component and therefore does not need to be concerned with render order.

4 Bandwidth Results

The *i*Grid2002 conference provided unique access to a high-bandwidth, high-latency network. We took this opportunity to investigate one strategy to efficiently utilize as much of the available bandwidth as possible. Based on the encouraging results achieved by Allcock et al. [1], we adopted their same general strategy in using GridFTP. Briefly, this strategy involves (1) simultaneously sending many large messages and (2) partitioning each message and sending the pieces over multiple connections each using the Transmission Control Protocol (TCP).

Allcock et al. ran their application over an OC-12 network having a lower latency (40 ms) and lower bandwidth (655 Mb/s) than the network available at *i*Grid2002. Nevertheless, we did not achieve the same relative bandwidth utilization as Allcock et al. instead, at the *i*Grid2002 conference we were able

to achieve only 160 Mb/s with 24 renders talking to 4 compositors and not using the GridFTP infrastructure.[6] After the *i*Grid2002 conference, we were able to achieve approximately 316 Mb/s using same configuration described above but this time using GridFTP. This bandwidth was achieved with 32 parallel sockets per message, a 2 MB TCP buffer, and an individual message length of 21 MB transmitted over 192 messages for a total transfer of 4 GB.

The direct explanation of our poor bandwidth utilization can be found in the network interface card (NIC), the implementation of TCP in our version of Linux, and the TCP protocol itself. High-latency, high-bandwidth networks ("long fat pipes") like the one at *i*Grid2002 induce very large congestion windows for optimal bandwidth utilization. The TCP protocol starts by exponentially increasing the sender-side congestion window until it encounters its first congestion event, which it uses as a trigger to cut the size of the current congestion window in half. It then enters congestion avoidance mode, where it increases the window only linearly (by only 1,500 bytes for a standard TCP connection) per round-trip message. After *i*Grid2002 we discovered that the Linux implementation of TCP encountered a congestion event triggered by a depletion of network NIC buffers. While the depletion of NIC buffers is not a congestion event as defined by TCP (i.e, dropped packet), it is reasonable for Linux to treat the case as such.

Nevertheless, the poor bandwidth utilization we experienced cannot be entirely blamed on the NIC or the Linux TCP implementation. At some level TCP itself, specifically its congestion avoidance strategy, makes it unsuitable for high-latency, high-bandwidth networks. In [3] Floyd addresses this fundamental limitation of TCP by proposing a modification to TCP's congestion control mechanism for use with TCP connections with large congestion windows.

5 Conclusion

Multiple lessons can be learned from the experience of building this prototype. MPICH-G2, with its recent extension of GridFTP support, has demonstrated its improvement in the utilization of a network pipe in a wide-area environment. Along with this lesson is a more subtle lesson of the need to support a mechanism for transparent characterization of the network that can be done automatically based on current network conditions and can adjust the application without user intervention.

[6] During the week of *i*Grid2002 we were unable to use the GridFTP infrastructure because of a bug in the application code.

Perhaps the most important lesson is that the congestion avoidance mechanism in TCP makes it a poor choice for high-latency, high-bandwidth networks because of the large congestion windows required for efficient bandwidth utilization induced by such networks. However, the basic strategy used by GridFTP of partitioning large messages and simultaneously sending pieces over multiple streams is still worthy of investigation. This same general technique has proved successful when using the *User Datagram Protocol (UDP)* [7,12]. We plan to modify MPICH-G2 so that it too sends large messages over multiple UDP streams; we will deliver that capability to applications in the same convenient manner as we did with GridFTP, that is, by setting values of a few fields in a structure and then using existing MPI idioms (i.e., communicator cache and `MPI_Attr_put`) to configure a point-to-point link.

Acknowledgments

We thank Jason Leigh and members of the Electronic Visualization Laboratory at the University of Illinois at Chicago for use of their tiled display and cluster during the *i*Grid2002 demonstrations; the Access Grid teams at both SARA and EVL for the use of their Access Grid node during the *i*Grid2002 demonstrations; and Rick Stevens and members of the Futures Laboratory for numerous useful discussions and support; and the members of the University of Chicago's Center for Astrophysical Thermonuclear Flashes for providing data used in this work.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the NSF Middleware Initiative (NMI) Program of the National Science Foundation.

References

[1] B. Allcock, J. Bester, J. Bresnahan, A. 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*, 2001. No page numbers available.

[2] L. Childers, T. L. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens. ActiveSpaces on the Grid: The construction of advanced visualization and interaction environments. In B. Engquist, L. Johnsson, M. Hammill, and F. Short, editors, *Parallelldatorcentrum Kungl*

Tekniska Hgskolan Seventh Annual Conference (Simulation and Visualization on the Grid), volume 13 of *Lecture Notes in Computational Science and Engineering*, pages 64–80, Stockholm, Sweden, 1999. Springer-Verlag.

[3] S. Floyd. High-speed TCP for large congestion windows. *Internet Engineering Task Force Internet Draft*, 2002.

[4] I. Foster and C. Kesselman. The Globus Project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–18. IEEE Computer Society Press, 1998.

[5] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[6] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[7] E. He, J. Leigh, O. Yu, and T. A. DeFanti. Reliable Blast UDP: Predictable high performance bulk data transfer. In *IEEE Cluster Computing*, page 317.

[8] G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski. Chromium: A stream processing framework for interactive graphics on clusters. In *Proceedings of SIGGRAPH 2002*, pages 693–702, 2002.

[9] N.T. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, to appear 2003.

[10] A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen. MPICH-GQ: Quality-of-service for message passing programs. In *Proceedings of Supercomputing 2000*. IEEE Computer Society Press, 2000. No page numbers available.

[11] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1995.

[12] Tsunami, Advanced Network Management Lab, Indiana University. `http://www.anml.iu.edu`.

Nicholas T. Karonis received a B.S. in finance and a B.S. in computer science from Northern Illinois University in 1985, an M.S. in computer science from Northern Illinois University in 1987, and a Ph.D. in computer science from Syracuse University in 1992. He spent summers from 1981 to 1991 as a student at Argonne National Laboratory, where he worked on the p4 message-passing library, automated reasoning, and genetic sequence alignment. From 1991 to 1995 he worked on the control system at Argonne's Advanced Photon Source and from 1995 to 1996 for the Computing Division at Fermi National Accelerator Laboratory. Since 1996 he has been at Northern Illinois University's computer science department, where he is now an associate professor,

and at Argonne's Mathematics and Computer Science Division as a resident associate guest working on the Globus Toolkit and MPICH-G2. His current research interest is message-passing systems in computational grids.

Michael E. Papka received a B.S. in physics from Northern Illinois University in 1990, an M.S. in electrical engineering and computer science from the University of Illinois at Chicago in 1994, and an M.S. in computer science from the University of Chicago in 2002, where he is currently completing his Ph.D. in computer science. He is the deputy director of the Futures Laboratory of the Mathematics and Computer Science Division at Argonne National Laboratory. He is also a fellow in the Argonne National Laboratory/University of Chicago Computation Institute. He is co-investigator both in the DOE SciDAC Middleware to Support Group-to-Group Collaboration and NSF NMI Extensible Network Services for the Access Grid, as well as senior staff on the DOE SciDAC Fusion Collaboratory and NSF ITR project entitled A Data Intense Challenge: The Instrumented Oilfield of the Future. His main research area is the use of scientific visualization in advanced display environments.

Justin Binns received his B.S. in computer science from Northern Illinois University in 1999. He has been employed at Argonne National Laboratory as a member of the Futures Laboratory in the Mathematics and Computer Sciences Division since 1999, where he is currently employed as a scientific programmer. His research interests include scientific visualization, run-time control of parallel computational systems, collaborative applications, the Access Grid, and active space environments.

John Bresnahan received his B.S. in computer science from Northern Illinois University in 1997 and his M.S. in computer science from Northern Illinois University in 1998. He is a senior scientific programmer with the Mathematics and Computer Science Division at Argonne National Laboratory, where he is currently working with the Globus Project as a member of the GridFTP team, designing and implementing data transfer protocols for the Grid.

Joseph Insley received his B.F.A. in electronic media from Northern Illinois University in 1991, his M.F.A. in electronic visualization from the University of Illinois at Chicago in 1997, and his M.S. in computer science from the University of Illinois at Chicago in 2002. He has been a scientific programmer with the Mathematics and Computer Science Division at Argonne National Laboratory and developer for the Globus Project since 1997. His research interests include distributed computing and scientific visualization.

David Jones received a B.A. in computer science from the University of Chicago in 2002. From 2000 to 2002, he was a student at Argonne National Laboratory working on tiled display software. He is currently an assistant scientific programmer for the Futures Laboratory in Argonne's Mathematics and Com-

puter Science division where he develops software for advanced displays and conducts software experiments for petaflops computing.

Joseph M. Link received his B.S. in electrical engineering from Northern Illinois University in 2001 and is currently pursuing an M.S. in computer science from the University of Illinois at Chicago. He is an assistant scientific programmer with the Mathematics and Computer Science Division at Argonne National Laboratory. He is currently the primary developer for the Globus Project's GridFTP team and also assists in the design of data transfer protocols for the Grid.

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.