Distance Visualization: Data Exploration on the Grid

Ian Foster[*+], Joseph Insley[*], Carl Kesselman[@], Gregor von Laszewski[*], Marcus Thiebaux[@]

* Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
+ Department of Computer Science, The University of Chicago, Chicago, IL 60637
@ Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292

Abstract

Scientific visualization has emerged as an important tool for extracting meaning from the large volumes of data produced by scientific instruments and simulations. Increasingly, the visualization process must deal with data sources, end users, analysis devices, and visualization devices that are geographically distributed. Such *distance visualization* scenarios introduce new challenges: for example, security, wide area networks, heterogeneity, and unreliable components. In this article, we explain how new approaches to visualization architecture can allow us to address these challenges in a principled fashion. In particular, we emphasize how infrastructure being developed for emerging national-scale computational "Grids" can be leveraged to both simplify and enhance the robustness, performance, and portability of distance visualization systems. We illustrate this discussion by reference to a substantial distance visualization application: a system for the collaborative, online reconstruction and analysis of tomographic data from remote X-ray sources and electron microscopes.

1 Introduction

As a consequence of our increased ability to model and measure a wide variety of phenomena, we are awash in data. In the immediate future, we anticipate collecting data at the rate of terabytes per day from many classes of applications, such as simulations running on teraflop class computers and experimental data produced by increasingly more sensitive and accurate instruments, such as telescopes, microscopes, particle accelerators, and satellites.

Of course, the generation of data is not an end in itself. Rather, one generates or acquires data in order to obtain insight. While there is a role for data analysis and reduction, in

many situations, understanding is achieved only when the data is interpreted by a human being. The well-documented abilities of the human visual system to recognize and interpret complex patterns has proved to be a vital adjunct to analytical techniques when it comes to detecting meaning—and anomalies—in scientific datasets. Hence, scientific visualization has emerged as an important tool in many areas of science.

Because the process of converting data into accurate and meaningful pictures is difficult, scientific visualization has also emerged as a challenging and important research area in its own right. Advances in algorithms, computer architecture, and visualization systems continue with the goal of allowing ever more sophisticated analysis of larger datasets. Desktop systems capable of interactive exploration of Gigabyte datasets represent the state of the art at the present time. High-resolution and immersive (virtual reality) displays, advanced analyses such as feature detection and tracking, and Terabyte (or even Petabyte) datasets have emerged as major challenges with significant implications for the future of visualization [1]. These and other issues are being addressed in substantial research efforts.

An aspect of visualization that has received less attention is the increasingly pervasive role of physical distribution. As the demands placed on visualization environments increase, it becomes more difficult to address all requirements on a single computing platform or for that matter in a single location. At the same time, high-speed networks and the advent of multidisciplinary science means that the use of remote resources becomes both feasible and necessary. *Distance visualization* hence emerges as a major new concept, with profound implications for both visualization and distributed computing architecture.

In distance visualization, data sources, image consumers, and the visualization engines that translate data into images may be geographically distributed and connected via networks of varying capabilities. For example, in applications that we describe below, data sources are located variously in Illinois and Japan, analysis devices in California and Illinois, and end users in California, Florida, Illinois, and Japan. Geographic distribution introduces significant challenges into the visualization process: one must deal with heterogeneous collections of resources, each with potentially different configurations, local resource management systems, and security requirements. Furthermore, the addition of the network adds additional complexity, as the design space is expanded to include a variety of protocols and communication technologies. Finely, one must deal with traditional distributed systems concerns such as ensuring that an application can robustly respond to resource failure.

In this article, we explore the implications of distribution for the analysis and visualization of data. Using a substantial application—an online, collaborative system for the reconstruction and analysis of tomographic data—to illustrate major points, we explore the motivations for distance visualization, identify major technical challenges, and describe architectural concepts and constructs that we believe can simplify the task of developing distance visualization systems.

Visualization architectures in common use today have not been designed with a view to distribution. In the few cases where some capability for distribution is provided, it is typically integrated into the application at the lowest level, limiting the ability of the application to run in widely distributed, highly heterogeneous environments.

We argue that distance issues must be addressed at a fundamental level in visualization architecture, rather than being treated as an optional add-on to existing architectures; and that the complexity of distance visualization applications can be reduced dramatically—and capabilities increased—by exploiting advanced "Grid Services" being developed in the context of the emerging computational infrastructure known as "the Grid" [2]. In short, we believe that distance visualization requires the development of a new generation of *network-oriented visualization architectures*. In this article, we describe both the current state of the art and expected future developments of such architectures.

2 A Distance Visualization Case Study

We use a detailed case study to illustrate the role of visualization in the scientific process and to motivate subsequent discussion of distance issues. Consider this scenario:

> A scientist is using a scientific instrument to examine a magnified picture of a biological specimen (see Figure 1), using controls to rotate it and zoom in on areas of interest. His colleague sees an interesting feature, and he yields the controls to her to zoom in on the area of interest. They discuss what they are seeing and compare results with another sample seen earlier in the day.

What makes this apparently routine scenario interesting and challenging is that the "instrument" in question is not a conventional microscope but rather a set of complicated electronic equipment. The "output" from this instrument is not visible light but gigabytes of data, which must be processed via a complex reconstruction process performed on a supercomputer before it can be "seen." The comparison with "another sample" involves the retrieval of gigabytes from a remote storage system. And the scientist and her "colleague" are not sitting next to the instrument, or even in the same room: in fact, they may be located thousands of kilometers apart: potentially on different sides of the Pacific Ocean (see Figure 2) [3].

This example illustrates why the decomposition of the data access/analysis/visualization process to allow for geographical distribution of its components can be so important in scientific computing. For one thing, it becomes possible to call upon more computational power than is typically available at experimental facilities. The tomographic reconstruction process used to produce the datasets shown in Figure 1 takes tens of hours on a modern workstation. Using a high-speed network to couple instruments to remote supercomputers can reduce this processing time to tens of minutes, hence allowing reconstructed data to be obtained while a sample is still in the instrument.

A second advantage of distribution in this context is that it can allow scientists to interact with an instrument without leaving their home institution. Today, it is not uncommon that scientists travel long distances to collect data or alternatively rely on a technician or student to collect data for them. Neither approach is ideal and becomes even more problematic when—as is often the case—an experiment involves a large team.

Together, these two capabilities revolutionize the ways in which scientific instruments can be used by allowing the currently batch-oriented processes of data collection and analysis to occur interactively. Because scientists can examine data sets as they are collected, they can verify that the experiment is proceeding correctly and respond to unexpected results while the sample is still in the instrument.

We have considerable experience with scenarios such as those just described because of work we have undertaken over the past two years in the context of a Department of Energy Grand Challenge project concerned with online tomographic reconstruction and analysis. In collaboration with colleagues, we have developed a sophisticated computational framework that supports on-line three-dimensional tomographic image reconstruction—and subsequent collaborative analysis—of data from remote scientific instruments [5]. We describe this framework below.

3 Challenges in Distance Visualization

Online tomographic reconstruction and analysis requires a distance visualization framework that can quickly extract measurement data from a scientific instrument (i.e., from the computer running the data capture software); pass that data to a supercomputer for construction of the three-dimensional data set; render that data so as to produce a three dimensional graphical representation; and enable that representation to be viewed and interacted with by one or more users. As we explained earlier, the instrument, supercomputer, and ultimate consumers of the data may be—indeed, usually will be—located at different sites. (To this list we could add a fifth goal, namely to allow remote operation of the scientific instrument. However, this issue is not related to visualization and so we do not consider it here.)

This is a typical example of a distance visualization application. (See the sidebar for a discussion of other circumstances in which distance issues arise.) In essence, the data-access/analysis/visualization pipeline that would traditionally have been executed within a single computer must be decomposed, with individual components placed on different systems, perhaps placed at different physical locations. Decomposition and distribution introduce new issues that are not encountered in the relatively simple, homogeneous, and hence predictable computing systems on which traditional visualization systems have been developed. These issues, and the techniques required to overcome them, include the following.

Specialized resources. Distribution is commonly employed to access specialized computers, scientific instruments, and archives. The networks used to access these resources may have high bandwidth-delay products or other unusual features. Harnessing these specialized resources can require the use of nonstandard protocols, algorithms, and programming techniques that are often not familiar to application developers: for example, predictive prefetching over networks.

Parallelism. In high-performance applications, parallelism becomes important within computers, networks, disks, and even display devices. Specialized techniques are required in order to exploit this parallelism effectively.

Heterogeneous capabilities. Distance tends to encourage diversity; hence, distance visualization applications must often deal with resources with different capabilities. For example, display devices may range from palmtops to immersive virtual reality systems and networks from multi-gigabit/s networks to dialup lines. Even apparently identical resources may be configured in different ways at different sites or provide different interfaces and services. As the capabilities of remote resources are typically not known to the application developer, applications need to be able to discover and then act upon configuration information.

Policies. Distribution also leads to variation in the policies that govern who can use resources, how resources are paid for, and what resources can be used. As an example of a policy that we may encounter, a site providing a remote rendering capability may limit the amount of network bandwidth any one remote display can consume: unless the requestor is a close collaborator or pays a premium. Applications need to be able to discover the nature of relevant policies and then act upon this information when selecting and using resources.

Lack of trust. Distance visualization brings together users and resource providers that may not trust each other or even have strong prior trust relationships. Mechanisms are required for establishing identity, for controlling who can use what resources when, and for protecting the integrity and confidentiality of data.

Dynamic behaviors. A consequence of the number of resources involved in wide area systems, the fact that many resources (in particular networks) must be shared, and the diversity of policies that govern resource access is that resource behaviors are often dynamic and—from the point of view of the user at least—unpredictable. Specialized mechanisms such as reservations and mirroring may be required for reducing the impact of such unpredictability; in addition, applications must be able to deal with changes in resource characteristics, by selecting alternative decompositions, resources, or algorithms. For example, if a remote hardware rendering engine becomes unavailable, an application might switch to a software renderer, simultaneously reducing frame rate. Or alternatively, it might reconfigure itself so that rendering is done where the data is located and a video stream is sent to remote users.

In summary, decomposition and distribution simultaneously increase the complexity and reduce the predictability of the environment in which visualization occurs. These two factors combine to make the creation of robust distance visualization systems a challenging task. Systems that are specialized to a particular hardware configuration and decomposition strategy are inflexible; systems that make least common denominator assumptions about each resource do not perform adequately. What is required is software that can adapt its behavior to the characteristics of the resources that are available to it at a particular point in time.

Developing software with these characteristics requires new approaches to the structuring of visualization applications. Specifically, we need to achieve a separation of concerns between three separate classes of problem: application-specific issues, issues that are application-independent but specific to distance visualization, and issues that relate to problems inherent in the fact that we are operating in a distributed environment. If this separation of concerns can be achieved, then in principle robustness is improved and the amount of code to be written for a specific application can be reduced significantly to just that required to address application-specific issues. The second set of issues can be handled via distance visualization toolkits while general distributed computing issues such as security and resource management can be handled by enhanced "Grid" middleware—see sidebar.

Clearly, while such a separation of concerns is in general a good thing, it remains to be shown how this separation can be achieved in distance visualization applications and whether indeed significant improvements in complexity and robustness can be achieved. Hence, the next section presents our experiences to date in this area, while the final section presents our views on future visualization architecture.

We note that the issues just listed are not addressed effectively in today's visualization systems. State-of-the-art scientific visualization systems as SCIrun [4] and AVS use sophisticated structures to support the modular construction of analysis/visualization pipelines via the composition of independent components. However, they do not address the distance issues just listed.

4 A Distance Tomography System

We now return to our distance visualization case study and discuss what it teaches us about the design of distance visualization systems. As noted above, this system supports the online collaborative reconstruction and analysis of tomographic data [5]. Tomography is a technique for determining the three-dimensional structure of an object from a series of two-dimensional measurements made at varying angles. (CAT scans work on this principle.) These measurements can be generated by a variety of instruments, including X-ray sources and electron microscopes, which "probe" the sample with X-rays and electron beams, respectively. A single dataset can be many gigabytes in size (e.g., several hundred angles, each involving 2,048 by 2,048 16-bit pixels) and reconstruction is computationally intensive.

The basic structure of our reconstruction and analysis framework is quite conventional: it is a pipeline (see Figure 3), with the major stages being data generation (by a data acquisition computer connected to the scientific instrument), tomographic processing (on a supercomputer or collection of workstations), and display (on high-end immersive displays such as an Immersadesk and/or on less capable display devices). The volume rendering computations required for display may be performed on a graphics computer connected directly to the display device; alternatively, volume rendering can be performed remotely and then sent to a low-end display as video. A master process controls the pipeline, which allows data to be reconstructed and passed to display client(s) via an iterative refinement process, as data is generated at the instrument. A set of collaborative controls allows any participating site to manipulate visualization parameters, including point of view.

This moderately complex application must deal with many of the distance issues introduced previously. The pipeline typically executes across three administrative domains, two computer architectures (often three), multiple security systems, and at least two communication protocols. Hence there are many opportunities for complexity and unpredictability. These complexities were addressed via the following two-pronged approach:

- We take for granted the existence of certain "Grid" services provided by the Globus toolkit [6] (see sidebar): in particular, information, authentication, and resource management.
- We also layer on a number of more specialized but still application-independent distributed visualization services, including some developed in previous work and some developed specifically for this project.

- As we explain in the following, this approach allows us to raise the level of abstraction to which our application is coded, simplifying development and increasing reliability.

4.1 Use of Existing Grid Services

Our implementation exploits three Grid services provided by the Globus toolkit: the Globus Resource Allocation Manager (GRAM) for resource management, the Grid Security Infrastructure (GSI) for authentication, and the Metacomputing Directory Service (MDS) for information look up. We also exploit the Globus communication library, Nexus, for inter-component communication.

The GRAM, GSI, and MDS services allow us to overcome implementation challenges relating to resource management. In this as in other distance visualization applications, we require mechanisms for discovering appropriate resources, for initiating computation on those resources, and for monitoring and managing those computations. As noted above, these tasks are complicated by the variety of resource types, resource management

systems, security mechanisms, and resource allocation policies that we encounter at different sites. We circumvent these difficulties by coding our application so that authentication is achieved via GSI's single-sign-on capability and resource discovery is performed via calls to MDS functions. Similarly, allocation and management of the computational resources required for data acquisition, reconstruction, and visualization are specified via calls to GRAM functions. The GRAM, GSI, and MDS implementations deployed at individual sites translate these calls into appropriate local mechanisms.

We use the Nexus communication library to similar advantage. Our goal is to create a pipeline that can be distributed in a variety of ways, depending on the characteristics of the computers and networks that are available to us at a particular instance. This goal complicates the implementation of inter-stage communication. For example, in some settings each stage might be placed on a separate computer accessible only via TCP/IP-based communication. Alternatively, a subset of the nodes might be collocated on a parallel computer with special purpose high-performance commutation libraries, such as MPI or shared memory. Coding these alternatives in the application would be complicated. Nexus addresses this problem by providing support for *multi-method* communication, a technique by which a single set of communication primitives is mapped efficiently to a range of underlying communication protocols. Nexus allows us to code the tomography application in terms of a single set of communication operations; the Nexus library maps these operations into the most efficient communication protocol available for the specific communication being performed.

4.2 Specialized Grid Services for Distance Visualization

Efficient execution in distributed environments can require the use of specialized techniques such as latency hiding, compression, and multicast for information sharing. The complexity of these techniques is a significant barrier to effective distance visualization. As noted in the sidebar on Grids, the way to overcome this barrier is to construct Application Toolkits that encapsulate "best practice" in relevant areas.

Our work on the tomography application allowed us to explore the effectiveness of this general approach. In support of this application, we developed three application-independent but visualization specific services, for network data buffering, shared controls, and remote display; these services were then used to simplify the implementation of the tomography application. Our experiences to date with these services have been positive, in that each has been reused in other contexts.

The first specialized service is designed to simplify the rendering of data that is being incrementally generated at a remote site. To this end, we developed a buffering service that receives processed data over the network asynchronously while providing a simple local interface for high-speed local access to arbitrary sub-cubes of the processed data. This interface also allows access to pieces of the data set as they become available, enabling incremental visualization of incomplete data sets at decreased resolution.

The second layered service is used to develop shared controls for collaborative exploration. Building on the ability of Nexus to support reliable multicast, we developed a shared state service that allows the value of a variable to be read or written from multiple locations. We used this service to implement user interface components that support collaborative control of the visualization application from multiple locations.

A unique feature of the visualization component of the tomagraphy application is its ability to display output at both local and remote displays simultaneously. Local displays are handled in a conventional fashion: local graphics hardware is used to perform volume rendering, with the results sent to an attached display. In this context, hardware rendering support is valuable, as it allows us to explore the resulting 3-D data interactively, cutting away portions of the data, rotating it, etc. However, this interactive element makes remote display of the images difficult, as we must also be able to send a sequence of images fast enough to enable interactive manipulation. Hence, we have constructed a special purpose remote display service that uses a video CODEC to process the contents of the local frame buffer and send it to the remote display using standard video conferencing protocols. In future implementations (as discussed below) we would like to offer remote rendering as a generic service, decoupling the views on the local and remote display.

5 Future Distance Visualization Architectures

Our experiences with applications such as that just described convince us that the architectural approaches proposed here can have a significant positive impact on our ability to build and deploy usable distance visualization systems. In current work, we are pursuing these ideas with the goal of defining what we term a Network-Oriented Visualization Architecture (NOVA).

While work on NOVA is still in its early stages we have already identified key areas in which work is required at both the Grid Services and Application Toolkits levels.

Event Service and Adaptation Framework. NOVA must provide sophisticated support for application-level adaptation to changing conditions. To this end, we are working to define a new Grid event service to support the discovery and delivery of the information required to support adaptation. At the Application Toolkit level, we are exploring the feasibility of constructing a visualization-specific adaptation framework that will allow programmers to describe a visualization pipeline, the performance constraints that the pipeline is to satisfy, the tradeoffs that are available for responding to changes in resource availability, and the policies that can be used to make these tradeoffs.

Flow Management. Distance visualization applications frequently involve a complex mix of flows due to the need to support data transfer, control information, and potentially also audio, video, and other collaborative modalities. The mapping of these flows to available communication resources is a complex task that we believe can be made easier by the definition of a flow management service that allows users to register required flow

characteristics and flow priorities. This service can then automate some of the resource management functions and notify applications when requirements cannot be satisfied. The flow management architecture will be most effective if it also integrates other resource management functions, for example the management of specialized hardware, such as rendering engines and video CODECs.

Visualization Communication Libraries. The flexible communication layer that we developed for the tomography application played a significant role in its success. However, the protocols and data formats used were developed specifically for this application. We believe that it is possible to develop communication libraries that are tailored to, and optimized for the transfer of data between components of the visualization pipeline. In doing so, we can simplify the process of application development while providing a standard mechanism for data exchange between visualization components. For example, we are investigating a communication library that is optimized for transporting four basic data types: structured and unstructured meshes and structured and unstructured point sets. These routines have been carefully designed for performance, for example enabling data to be removed from the network and fed directly into rendering hardware without reformatting.

Specialized Communication Protocols. Continuing in this vein, we also must consider enabling the use of alternative protocols. For example, in some cases, better overall interactive and display performance may be achieved by the use of "semi-reliable" communication protocols in which subsets of the data may not actually be delivered. Delivery of MPEG interopolation frames is an example of such data. Similarly, remote display of rendered visualization data may benefit from specialized coding and transfer schemes that can make tradeoffs between observed latency and available network bandwidth. These examples imply that NOVA can benefit from a flexibly means of protocol selection that can be driven by the application requirements as well as available resources.

Collaboration Services. Toolkits that enable collaboration of the visualization experience will also be important. The CAVERNsoft environment is an example of such a toolkit [7]. Building on basic Grid services, CAVERNsoft provides the application developer with a high level set of abstractions that are designed to support shared viewing, telepresence, and collaborative exploration. We expect next-generation collaborative tools to provide improved capabilities by leveraging more generic NOVA services such as those just described.

6 Conclusions

In this article, we have explored the symbiotic relationship between advanced visualization and the Grid infrastructures that are currently being created. We have used the term distance visualization to evoke what we believe is an exciting vision, namely that the "visualization engine" that we use to extract meaning (or at least pictures) from

data need no longer be restricted in its capabilities to what we can place on our desktop. Instead, we can create virtual visualization engines that integrate resources distributed across a machine room, institution, or the globe. These resources can then be used to perform analyses and to share information in ways not previously possible.

Needless to say, achieving this vision requires that we overcome many challenges. We must learn how to match the problems that we wish to solve to the resources that are available to use and we must discover how to knit those resources into an integrated environment. However, with this challenge comes opportunity, as this rich, distributed environment enables us to tackle problems that would simply not be possible with today's technology.

Acknowledgments

We are grateful to our colleagues at Argonne National Laboratory and the University of California at San Diego with whom the microtomographic applications described here were developed: in particular John Bresnahan, Mark Ellisman, Peter Lane, Ian McNulty, Mark Rivers, Mei Su, Steve Tuecke, and Steve Wang. 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; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation; and by the NASA Information Power Grid program.

6.1.1.1.1 Sidebar #1: Why Distance?

In the case study presented in the text, distance arises because the data source in question is a specialized instrument and because the analysis process requires another specialized instrument, a supercomputer. But we should not conclude that distance is only an issue when such unique resources are involved. In fact, distance can play an important role at every stage in the visualization pipeline (see Figure 4).

Data acquisition. Data sources encountered in distance visualization range from scientific instruments to data archives and supercomputer simulations. The introduction of distance allows other stages in the visualization pipeline (and/or the user) to be located remotely from the data.

Analysis and Interpretation. Analysis translates raw data into derived quantities required to answer user questions; intrepretation generates a drawable representation of the process data. Analyses can range from simple data management tasks (e.g., culling, subsampling) to computationally intensive tasks such as feature extraction. Distance allows analysis functions to call upon more computing power than is available at the data sources.

Rendering and Rasterization. Rendering converts a drawable representation into a graphics language, such as GL; rasterization then generates pixels. When viewers are remote, it can be useful to locate various combinations of the analysis, interpretation, rendering, and/or rasterization functions remotely from the data. Determining factors include the number of viewers, viewer requirements (e.g., same or different perspectives and resolutions), data sizes at various stages in the pipeline, network capabilities, and the availability of specialized devices (e.g., rendering engines). For example, with fast networks and hardware rendering engines at viewer locations, we can transmit analyzed data and perform rendering remotely. Alternatively, if networks are slow, analyzed data is large, and/or rendering engines are not available remotely, we may prefer to transmit rasterized data.

Display and Interface. Collaborative exploration and control can frequently motivate the need for distance in both the display and interface/control functions. Furthermore, distance in the display function can be introduced by the use of remote rasterization, as described above.

SIDEBAR #2: The Grid and the Globus Toolkit

A decade of experimentation with advanced network applications has demonstrated a clear need for services beyond those provided by today's Internet. Pervasive authentication/authorization, information, resource management, instrumentation, and other services are essential if application development costs are to be kept manageable and if applications are to operate robustly in dynamic networked environments.

This realization has spurred interest in the development and deployment of "Grids," virtual private networks that offer such enhanced services to various communities. Early experiments with Grid infrastructure, such as the 1995 I-WAY experiment, demonstrated feasibility; now, several organizations—notably NASA, via its Information Power Grid Program (IPG) [8], and the NCSA Alliance, in its National Technology Grid (NTG) [9]—are deploying production Grid infrastructures to support large communities. These Grid infrastructures provide both enhanced capabilities within end system resources (e.g., advance reservation support on computers, quality of service mechanisms in networks, policy publication mechanisms) and new middleware services within the network: e.g., certificate authorities, information services, instrumentation services, and resource management services. Providing these services as common infrastructure reduces application development costs and the number of services that must be deployed at individual sites.

A current focus of activity within the Grid community is the development of *Grid Toolkits*, enhanced services and tools that build on underlying Grid Services to support a specific class of applications: for example, remote instrumentation, distributed collaboration, distributed supercomputing, parameter studies, and distance visualization. We believe that work on distance visualization should be seen in this context and that the development of a "distance visualization toolkit" (for which we propose an architecture in this article) will be an important step forward.

The application described in the text uses Grid services developed in the Globus project (http://www.globus.org), a leading Grid infrastructure research and development effort [6]. Globus technologies are at the core of IPG and NTG and are also deployed across the Globus Ubiquitous Supercomputing Testbed Organization (GUSTO), an informal international collaboration of Grid researchers. Globus technologies include security, resource management, data management, communications, fault detection, and instrumentation.

For more information on Grid concepts and technologies, see [2] and the web site for the Grid Forum (http://www.gridforum.org), a community organization dedicated to the discussion of best practices and standards in Grid computing.

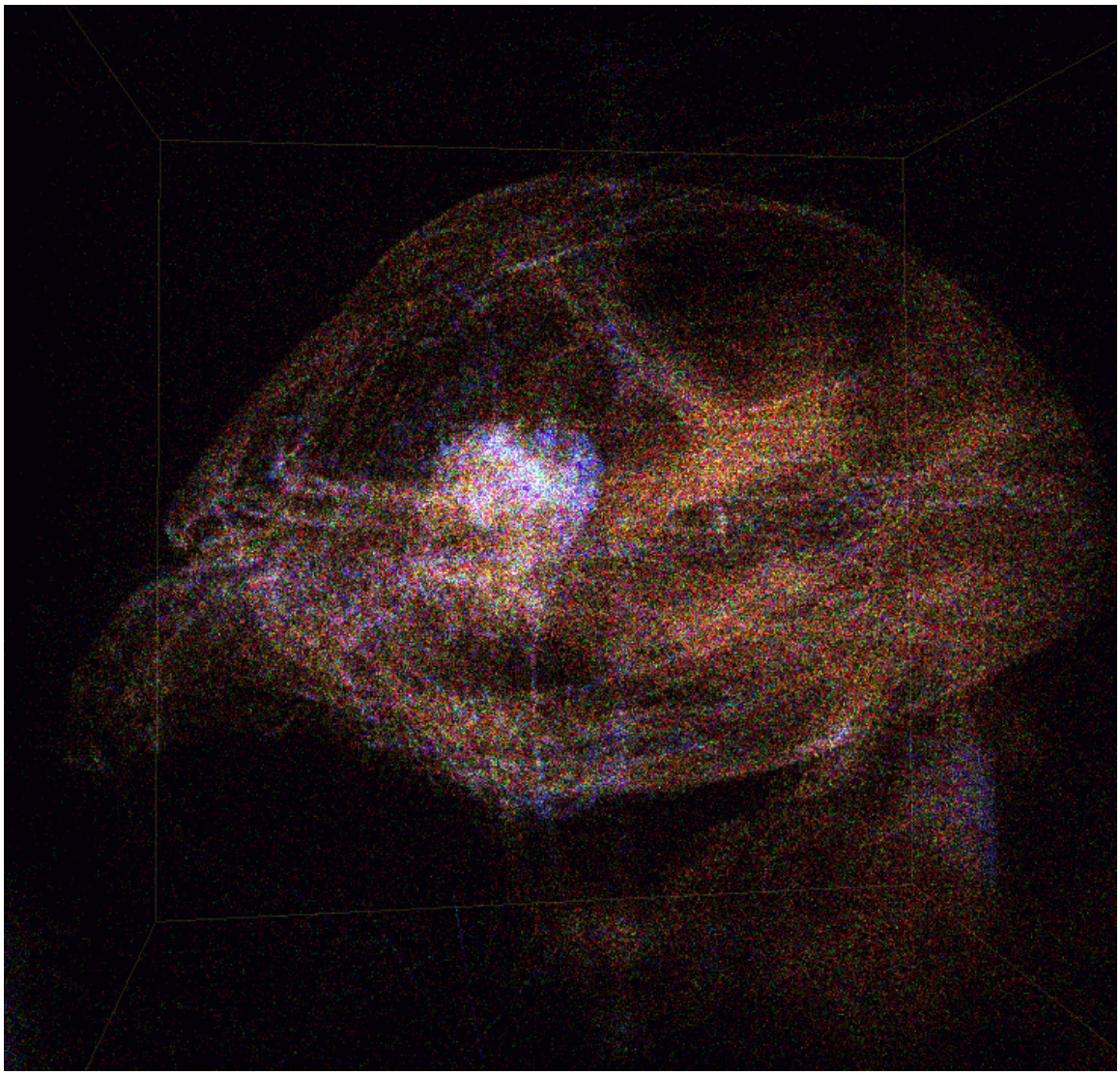

FIGURE 1: A volumetric view of the three-dimensional structure of an ant head. The raw data was collected on the Advanced Photon Source at Argonne National Laboratory and comprised 256 projections of 512 by 512 pixels; the reconstruction output is a 512-cubed 3-D volume. The data set took 15 minutes to collect; the final result took approximately 10 minutes to generate, using 32 processors for the reconstruction. The volumetric visualization was constructed by using a stochastic sampling method to generate three simultaneous isolayers.

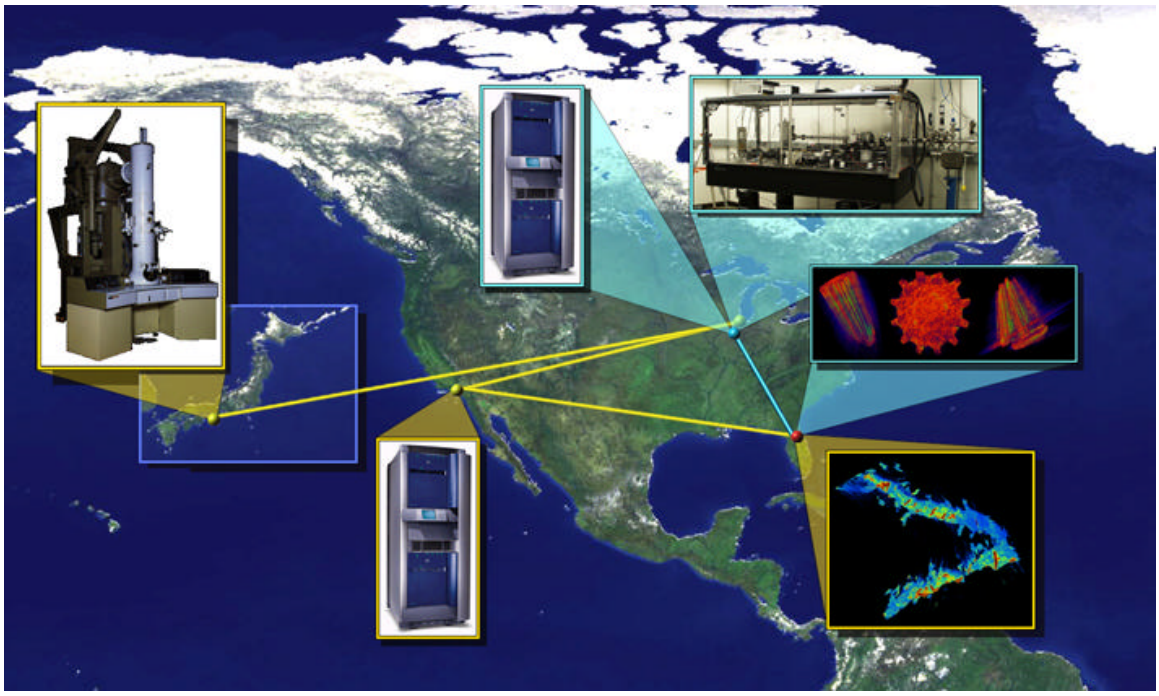

FIGURE 2: The world's brightest high-energy X-ray source is the Advanced Photon Source (APS) at Argonne National Laboratory in Illinois, while the most powerful electron microscope is located at Osaka University in Japan. The online analysis system described in this article couples these instruments to supercomputers at Argonne and USC/ISI and to users located across the U.S. (The configuration illustrated here also includes users at the SC'98 conference, held in Orlando Florida.) Network access to supercomputers and to remote users is provided by DOE's ESnet and NSF's vBNS, while the TRANSPAC connection through STAR-TAP (http://www.startap.net) enables high-speed access to Osaka. The images shown on the right-hand size of the figure are a micromachinery part (top) and a fossil shell (bottom), both imaged at the APS.

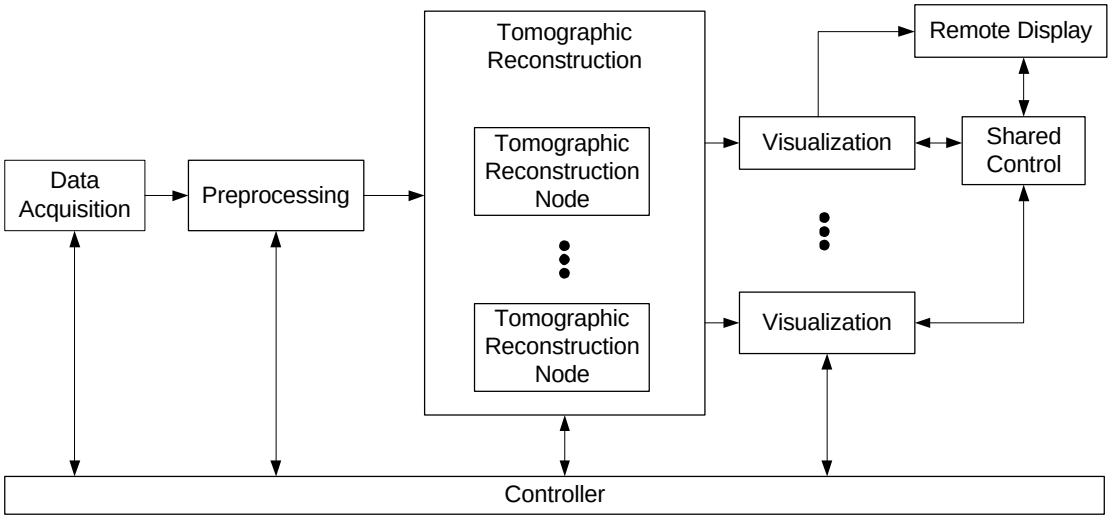

FIGURE 3: The structure of the tomography application described in the text. Pipeline consists of acquisition, preprocessing, reconstruction, and display. Reconstruction can be parallelized and distributed to multiple tomographic reconstruction nodes. Results can be displayed at more then one location, and a remote display can be slaved off of any visualization node. Visualization nodes are coupled by a shared control.

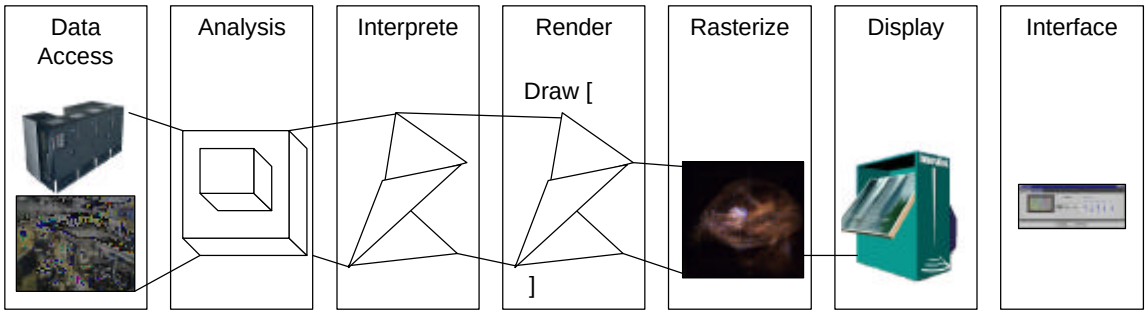

FIGURE 4: As described in the sidebar, distance issues can arise at each stage in the visualization process, which as shown here can be viewed as a seven-stage process.

References

[1] P.Smith and J. van Rosendale (Eds), Data and Visualization Corridors, Technical Report CACR-164, California Institute of Technology, 1998. Also at http://www.cacr.caltech.edu/publications/DVC

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

[3] Paper by Mark Ellisman on remote microscopy. [reference TBD]

[4] Parker, S.G., Weinstein, D.M., Johnson, C.R., The SCIRun computational steering software system, In *Modern Software Tools in Scientific Computing*, Birkhauser Press, pages 1-44, 1997.

[5] Gregor von Laszewski, Ian Foster, Joseph A. Insley, John Bresnahan, Carl Kesselman Mei Su, Marcus Thiebaux, Mark L. Rivers, Ian McNulty, Brian Tieman, and Steve 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.

[6] Foster, I. and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. In [2], pages 259-278. Morgan Kaufmann Publishers, 1999.

[7] Paper by J. Leigh et al. submitted to this same special issue.

[8] W. Johnston, D. Gannon, W. Nitzberg, Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid, *Proc. 8th Symposium on High Performance Distributed Computing*, IEEE Computer Society Press, 1999.

[9] R. Stevens, P. Woodward, T. DeFanti, and C. Catlett, From the I-WAY to the National Technology Grid, *Communications of the ACM*, 40(11):50-61, 1997.