Visualization Viewpoints

The Grid and Future Visualization System Architectures

John Shalf and E. Wes Bethel

Lawrence Berkeley National Laboratory

The promise of grid computing, particularly grid-enabled visualization, is a transparent, interconnected fabric to link data sources, computing (visualization) resources, and users into widely distributed virtual organizations. The grid's purpose is to tackle increasingly complex problems. However, a wide gulf exists between current visualization technologies and the vision of global, grid-enabled visualization capabilities. Here, we discuss our views of how visualization technology must evolve to offer true effectiveness in a grid environment.

The vision

In our example, Dr. Jane is a geophysics researcher studying the effects of a new material, code-named *zorbs,* that can mitigate the effects of earthquakes on urban structures. The material works by isolating structures from direct exposure to shock damage through direct injection into porous media upon which structures are built. In the course of her studies, she runs experiments at the molecular scale, where she observes molecule to molecule interactions, and at the urban scale, where she induces shock waves into the ground with large equipment. She has access to instrumentation attached directly to urban structures in Shaky City that collects real-time data and stores it at a nearby data center.

Dr. Jane runs large-scale simulations that spawn off subsimulations around the world, wherever compute cycles are available. The subsimulations, when complete, report the results of parameter studies to the master simulation. Because the simulations routinely generate terabytes of results, data caches located close to the computational resource store data for later analysis and visualization. Dr. Jane routinely seeds her simulations with experimental data, and in turn, modifies her experimental parameters with simulation results. When an earthquake hits, she can immediately connect to a vast network of sensors to observe how her handiwork has reduced direct shock damage caused by the pressure and shear waves generated by earthquakes. She can make these observations using a desktop workstation or a handheld telephone. Dr. Jane is part of a large, multidisciplinary team scattered around the world. All collaborators can access the entire collection of experimental and simulation data, which may reside at any one of the member institutions, see Figure 1.

Although this description is fiction, the scope of Dr. Jane's activities is not far fetched, and is reasonably characteristic of many modern research scientists. Indeed, the promise of grids is the ability to weave a collection of disparate resources into a single application in exactly the fashion we have suggested. As visualization researchers and developers who work with people like Dr. Jane on a routine basis, and who design and build visualization tools, it's our opinion that the current state of visualization software is not grid ready. In fact, we believe that a fundamental paradigm shift is needed to create fully global visualization applications. A new grid-aware framework is needed for distributed visualization that's easy to use, modular, extensible, and permits reuse of existing investments in visualization technology.

Getting from here to there

A number of successful, contemporary visualization systems such as VTK, AVS, and OpenDX let users quickly assemble finished applications by combining software components into a processing pipeline. These systems' success is due to their flexible and extensible characteristics. Application developers can use this flexibility by combining components in several ways, enabling a wide variety of visualization tasks. Developers also can benefit from the systems' extensibility by adding new components to the system, extending functionality.

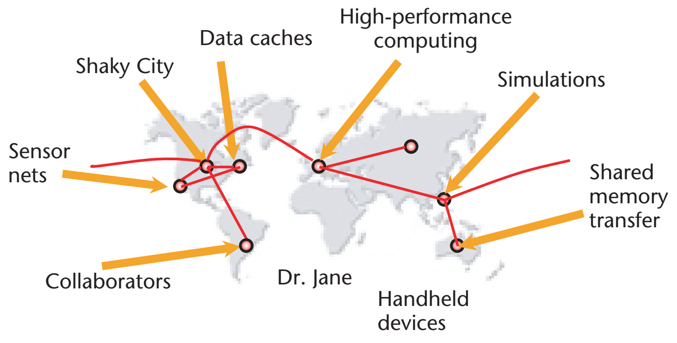


Figure 1. Grid-based visualization and computing applications comprised of heterogeneous resources.

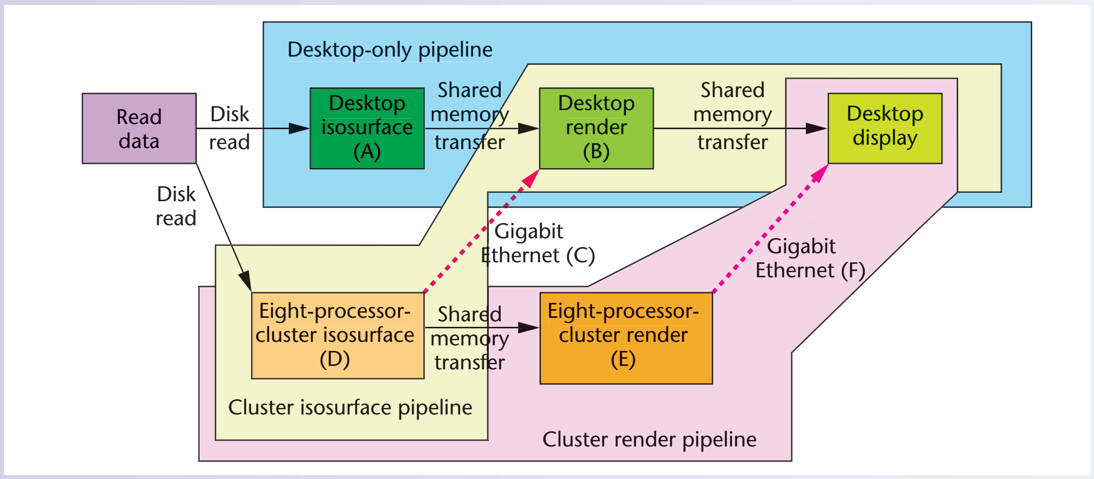
We well understand the characteristics of such systems deployed on a single machine, some of which can run in a distributed manner across a limited number of machines. But deploying such systems onto the grid requires consideration of new conditions not likely anticipated during their initial design and implementation. A distributed visu-

Performance Modeling and Pipeline Partitioning

Most remote and distributed visualization applications use a static partitioning of the visualization pipeline in the hopes that such partitioning works well for all situations. The example in this sidebar illustrates the importance of dynamic partitioning for distributed visualization applications. Figure A shows the components of three potential visualization pipeline configurations that create an image of an isosurface. The links between the blocks indicate data flow.

The three pipeline configurations shown in Figure A use different resources to create the same finished product. The desktop-only pipeline places the isosurface and rendering components on the desktop machine. The cluster isosurface pipeline places a distributed-memory implementation of the isosurface component on the cluster and transfers triangles over the network to a serial rendering component on the desktop. The cluster render pipeline performs isosurface extraction and rendering on the cluster, then transfers images to the desktop for display. The colors in Figure B's bar graph (next page)

continued on p. 8



A Components and data flow for our hypothetical example. Arrows indicate potential data flow paths. There are three potential partitionings in this graph. Letter annotations indicate the component in figures B and C.

alization architecture—a framework suited for distributed, component-based, grid-enabled visualization—can best meet these additional design considerations.

Distributed, heterogeneous components

Building upon the architecture of successful, contemporary systems, future grid-based systems will likely use a component-based architecture. At their core, these distributed components will probably resemble their single-machine cousins—that is, they will perform a well-defined operation on strongly typed input data to produce a result. Currently, modules from one system are often incompatible with those of another or bound to a particular user interface paradigm, creating isolation among users and within the visualization community. Visualization system users want to use the best tool for the job, regardless of its source.[1] Achieving such a goal requires solving a plethora of engineering issues and cultural differences. One issue is the nature of a flexible high-performance-computing component interface, a subject of active research. For instance, the Common Component Architecture (http://www.cca-forum.org) effort seeks to create an Interface Description Language and automatic language wrappers oriented toward high-performance computing application requirements. The Advanced Collaborative Environments Research Group (http://calder.ncsa.uiuc.edu/ACE-grid/main.html) of the Global Grid Forum (http://www.gridforum.org) is defining the security requirements for this sort of component architecture.

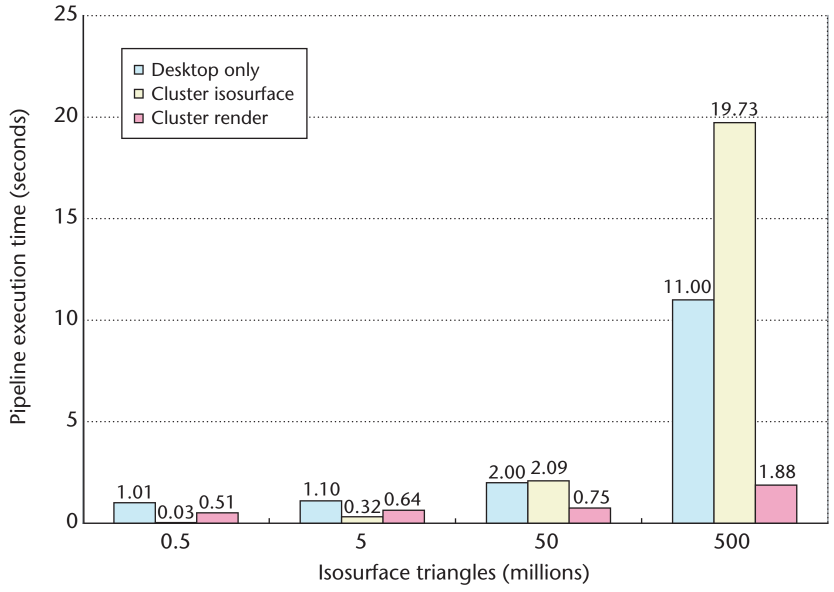
Fundamental graphics and visualization algorithms

The hallmark of grid-based deployments is a focus on remote visualization. Typical past approaches to remote visualization have focused on one of two broad methods. The first approach performs all visualizations on the server and sends image data (or X-protocol streams) to the client. This approach works best with a large data set. The alternative approach transfers subsets of the large scientific data from the server to the client workstation, where visualization occurs completely on the desktop machine. This approach works best when interactivity is most important. However, these requirements can change dynamically at runtime, as we show in the "Performance Modeling and Pipeline Partitioning" sidebar. New trends in graphics and visualizations have produced a new group of algorithms that will become increasingly important for future grid-based visualization systems. These latency tolerant algorithms seek to maintain interactivity on the desktop, while performing visualization of large data sets. Visapult[2] is an exam-

continued from p. 7

correspond to the groupings of components shown in Figure A. To measure performance for each of these pipelines, we make the following optimizing and simplifying assumptions:

- All triangles produced by the isosurface component are triangle strips. A single vertex represents each incremental triangle of the strip and consumes 24 bytes (six 4-byte floats comprising vertex and normal information).
- All graphics hardware can render 50 million triangles per second. An eight-node system can render 400 million triangles per second.
- Isosurface creation takes 1 second on the desktop, and 0.125 seconds on the eight-node system.
- The image transfer assumes a 24-bit high-definition 1920/1080p common-intermediate-format frame buffer.
- Interconnects use a 1-Gbit network with perfect performance.
- The performance model does not consider the cost of data reads, scatter-gather operations, or displaying cluster-rendered images on the desktop.



B Absolute runtime for each pipeline. Desktop-only performance is the sum of components A and B (letters refer to annotations in Figure A); cluster isosurface performance is the sum of components D, C, and B; and cluster render performance is the sum of components D, E, and F.

Whereas Figure B shows the absolute runtime of each pipeline for varying numbers of triangles, Figure C presents the relative runtime of all compo-

ple of pipelined-parallel image-based rendering-assisted volume rendering,[3] while the TeraScale Browser[4] uses multiresolution methods and pipelined data caching for interactive, latency-tolerant visualization. More algorithms of this type should be integrated into a common framework to perform interactive, desktop visualizations of large, remotely located data sources using distributed visualization components.

Distributed execution and dynamic scheduling

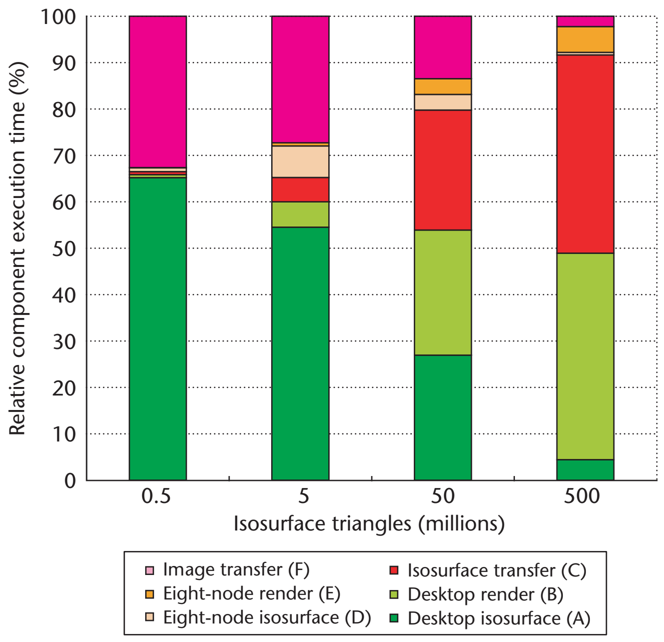
Most contemporary component-based visualization systems, which manage component launching and data movement, are typically suited for single platform use. As such, it's likely these systems won't prove effective in grid environments. A common practice is a manual staging of components on machines into an execution pipeline. This approach is impractical in grid environments where a single application might use hundreds or thousands of resources. A better approach is a database or directory service that distributes and tracks components—both executables and running instances—on heterogeneous resources. Services of this sort could exist in grids based on the Globus architecture's monitoring and discovery service (http://www.globus.org/mds), which is essentially a lightweight-directory-access-protocol-based hierarchy of resources for a given virtual organization.

End-to-end performance

The practice of static pipeline configuration for distributed resources offers no guarantee of better performance than when running all components on a single machine. While an individual component's performance in a system is straightforward to quantify, we need to model overall system performance for distributed components to effectively use all resources. To select a reasonable pipeline arrangement, we need to predict performance of candidate pipelines before we launch them. Such modeling is nontrivial, and will have a profound impact on future system architectures. Because underlying resources change dynamically in typical grid computing environments and visualization algorithm performance depends on parameter and data choices, distributed visualization architectures must continuously monitor runtime performance to dynamically select resource uses, work distributions, and algorithm alternatives. The Grid Application Development Software project (http://hipersoft.cs.rice.edu/grads/gradsoft.htm) is grappling with performance prediction for distributed applications. Other efforts like the EU GridLab (http://www.gridlab.org) project seek to create application programming interfaces that hide the complex, dynamic nature of grid resources from application developers. The impact of such dynamic selection on performance is illustrated in the sidebar.

nents for varying numbers of triangles. Each vertical bar in Figure C sums the absolute runtime of all components, producing 100 percent, and the size of each colored segment in the bar shows relative time consumed by that particular component. Each component's execution time is normalized by the sum of all component execution times so we can quickly determine which components, or network transfers, will dominate the execution time. (Some components are used in more than one pipeline configuration.) The vertical bars in Figure C are color-coded by component: green shades indicate a desktop resident, red shades indicate network transfers, and orange shades indicate a cluster resident. We've arranged components vertically so you can visually integrate groups of adjacent components into their respective pipeline partitionings. In the 500,000 triangles case, the cost of desktop isosurface extraction dominates in the desktop-only pipeline. In contrast, the cluster isosurface pipeline would perform about six times faster. In the 500 million triangles case, the cluster render pipeline is about five times faster than the desktop-only pipeline, and about eight times faster than the cluster isosurface pipeline.

This example illustrates how optimal pipeline partitioning can change as a function of a simple parameter change. Creating and deploying such dynamic repartitioning strategies on the grid requires consideration of many more variables than discussed here and a more accurate performance model. Grid-based services help measure and provide such performance information to applications, but they don't perform the kind of dynamic partitioning needed to support grid-based visualization. A substantial body of new infrastructure work is required to support effective grid-based visualization tools.



C **Relative performance components in the three pipelines. Execution time is normalized for all configurations. (Letters in the key refer to the component annotations in Figure A.)**

Conclusion

Extending current visualization tools to grid-enable them is a daunting challenge due to the breadth and depth of issues. At one end of the spectrum, careful performance analysis and optimization of the visual pipeline is needed to effectively use remote and distibuted resources, especially in dynamic environments. At the other end of the spectrum, we need new fundamental graphics and visualization algorithms to implement latency-tolerant tools suited for remote and distributed visualization settings. We have only briefly mentioned a few of the numerous issues related to implementing and deploying effective grid-based solutions. The approach we favor leverages on lessons learned from successful, contemporary visualization packages. Specifically, the best solution is a new framework that hides the complexity of implementing fully grid-enabled distributed visualization tools built from software components. Our vision is a distributed visualization architecture that supports high-performance, latency tolerant, and heterogeneous components to promote growth of visualization technology into grid environments and unity within the visualization and computational science communities. ■

Acknowledgement

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division, US Department of Energy under contract no. DE-AC03-76SF00098.

References

1. B. Hamann et al., *NERSC Visualization Greenbook*, Lawrence Berkeley National Laboratory, 2002, http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf.
2. W. Bethel et al., "Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization," *Proc. IEEE/ACM SC2000*, CD-ROM, IEEE CS Press, 2000.
3. K. Müller et al., "IBR-Assisted Volume Rendering." *Late Breaking Hot Topics, Proc. IEEE Visualization*, ACM Press, 1999, pp. 5-8.
4. V. Pascucci and R. Frank, "Global Static Indexing for Real-Time Exploration of Very Large Regular Grids," *Proc. IEEE/ACM SC2001*, CD-ROM, ACM Press, 2001, http://www.sc2001.org/papers/pap.pap114.pdf.

Readers may contact John Shalf or E. Wes Bethel at Lawrence Berkeley National Laboratory, 1 Cyclotron Rd., Mail Stop 50F, Berkeley, CA 94720, email {jshalf, ewbethel}@lbl.gov.