Scalable End-to-End Parallel Supercomputing and Application to Real-time Earthquake Modeling

Hongfeng Yu (Technical lead)[‡] Tiankai Tu (Team lead)[*] Jacobo Bielak[*] Omar Ghattas[†]
Julio Lopez[*] Kwan-Liu Ma[‡] David R. O'Hallaron[*] Leonardo Ramirez-Guzman[*]
Nathan Stone[§] Ricardo Taborda-Rios[*] John Urbanic[§]

Abstract

We demonstrate a new scalable approach to real-time monitoring, visualization, and steering of massively parallel simulations from a personal computer. The basis is an end-to-end approach to parallel supercomputing in which all components — meshing, partitioning, solver, and visualization — are tightly coupled and execute in parallel on a supercomputer. This approach avoids bottlenecks associated with transfer and storage of massive simulation outputs, thereby enabling real-time visualization and steering on supercomputers with thousands of processors. We have incorporated this methodology into a framework named *Hercules*, which targets octree-based finite element simulations. The submitted video demonstrates real-time monitoring and steering from a laptop PC of a 1024-processor simulation of the 1994 Northridge earthquake in Southern California. Because this end-to-end approach does not require moving large data over the network and is completely scalable, our approach shows promise for overcoming the challenges of visualization of petascale simulations.

1 Introduction

As parallel supercomputing moves beyond the realm of the terascale and into the petascale, the size of the data generated by a single scientific simulation can exceed hundreds of terabytes. Beyond the challenges of efficiently executing parallel simulations on thousands of processors lie the—perhaps even greater—challenges of visualization and interpretation of results produced by such massive simulations. While a number of efforts have been directed at facilitating the storage, transfer, and visualization of massive simulation datasets, there remain significant bottlenecks associated with this offline approach for very large-scale problems. A more scalable solution is to visualize the simulation output directly *at simulation runtime*. However, the visualization computations must be as scalable as the simulation to make this online approach viable.

We have developed a new approach that couples all simulation components (meshing, partitioning, solver, and visualization) tightly in a unified framework. All components operate on a shared parallel data structure, execute in parallel on the same set of processors, and avoid intermediary file I/Os. We refer to this new approach as *end-to-end parallel supercomputing*.

A crucial feature that facilitates scalability to thousands of processors and billions of elements is the ability to visualize partial differential equation (PDE) solution data simultaneously as the PDE solver executes. Volume renderings are computed online and in parallel using the same processors that compute the PDE solution; thus, simulation results are retrieved directly from each processor cache or main memory for rendering. As a result, data reduction and summarization take place instantly. Visualization images, which are often several orders of magnitude smaller in size than the corresponding solution fields, are sent to a remote user via (for example) a low bandwidth TCP/IP connection. By contrast, traditional methods use a separate visualization cluster, which requires a sustained network bandwidth equal to the solution output rate in order to support runtime visualization [4].

Our approach therefore avoids the bottlenecks associated with transferring and/or storing large volumes of output data, and is applicable whenever the user's ultimate interest is visualizing the 3D volume output, as opposed to retaining it for future analysis. However, rendering solution data *in-situ* and on-the-fly presents a number of significant computational and networking challenges. First, how can we visualize efficiently on thousands of processors simultaneously with the execution of the solver? Usually, visualization clusters are relatively small (8 to 128 processors). As a result, traditional visualization algorithms involving unstructured finite element meshes seldom scale on more than 512 processors. Second, how can we send an image back to a remote user? Third, how can we support runtime user interaction? That is, when a user specifies a different visualization configuration (for example, a new view angle),

[*]Carnegie Mellon University
[†]The University of Texas at Austin
[‡]University of California, Davis
[§]Pittsburgh Supercomputing Center

how is the control information sent to a supercomputer, and how can the parallel visualization algorithm adjust its rendering and compositing schedule accordingly?

We have addressed these problems within an end-to-end supercomputing framework named *Hercules* [3], which targets octree-based finite element simulations. The breakthrough new capabilities are listed below.[1]

- A new parallel unstructured octree mesh volume rendering algorithm that scales on 1024 processors
- Runtime image delivery to a remote client computer
- Runtime remote user-controlled visualization steering

Note that in our system, volume rendering is performed solely on a supercomputer. A client machine such as a laptop serves only to composite the received image with a background and render the results. When a user steers a visualization, the controls are captured locally and sent to a supercomputer. The heavy lifting, i.e. re-construction of a parallel visualization communication schedule, is carried out on the supercomputer in parallel.

We have applied the extended Hercules framework to simulate the 1994 Northridge earthquake in California. Running the code on the Cray XT3 at the Pittsburgh Supercomputing Center (PSC), we are able to visualize seismic body wave propagation, change view angles, adjust sampling steps, zoom in and out of the domain, modify color maps, and interpret the results—all concurrently with the execution of the simulation on 1024 processors.

2 System Overview

Figure 1 shows the overall software architecture, which consists of three main components: Hercules, PDIO, and QuakeShow.

Hercules [3] is a finite element/octree-based end-to-end parallel simulation framework. It generates and partitions an unstructured hexahedral finite element mesh, solves the governing PDEs, and volume renders the solution results, all in parallel, tightly coupled, and built on top of the same octree data structures. Our previous work has demonstrated the scalability of Hercules on up to 2048 processors for a problem with 400 million degrees of freedom.

In the context of earthquake simulations, the input to Hercules is a material database describing the earth properties in the domain of interest and an earthquake source description. On the PSC Cray XT3, the material database (22 GB) is stored on the Lustre parallel file system. The outputs are images of the propagation of seismic body waves, in either compressed jpeg format (a few hundred kilobyte) or uncompressed tga format (a few megabyte).

PDIO (Portals Direct I/O) [2] is a special-purpose middleware infrastructure that supports external interaction with

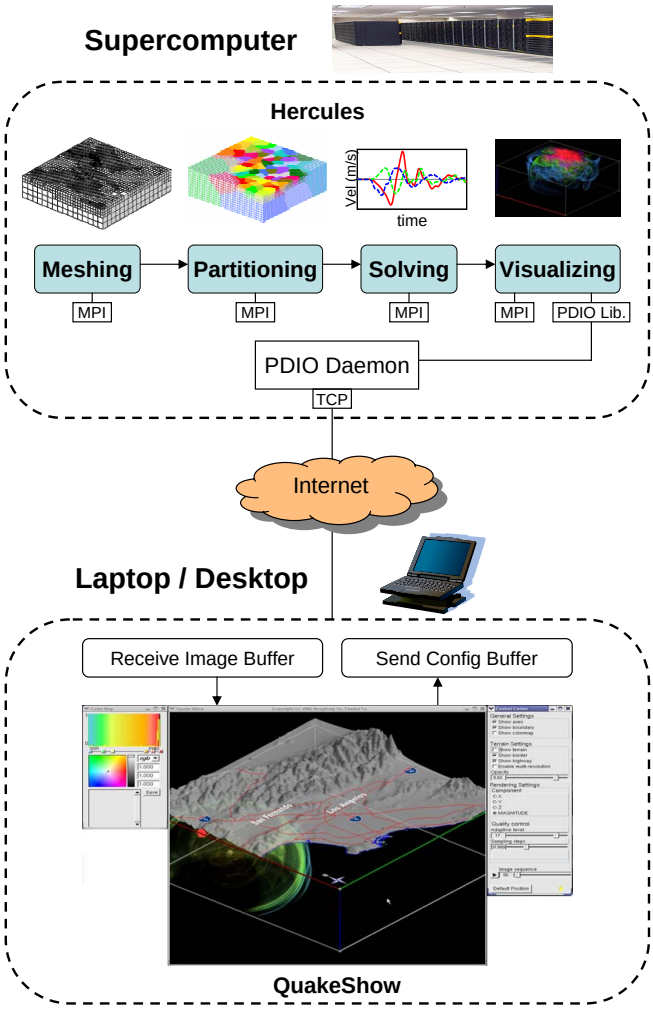

Figure 1: **Extended Hercules System Architecture.**

[1]The results presented in this paper are the newest extensions to the Hercules system.

parallel programs running on Portals-enabled compute nodes, as is the case on the Cray XT3.[2] An application, for instance an MPI program, calls PDIO client library functions to communicate with PDIO daemons running on the I/O nodes (of the same supercomputer). When a parallel process writes to or reads from the PDIO client library, the data is routed via the internal Portals network directly to or from a PDIO daemon, without the intervention of other application libraries such as MPI.

A PDIO daemon runs on an externally connected I/O node. It receives Portals messages from clients on the compute nodes, aggregates them into optimally-sized buffers in memory and asynchronously routes them over the Internet via parallel TCP/IP data streams to remote receivers. Each process utilizes one or more multi-threaded ring buffers accessed by Portals and TCP/IP.

QuakeShow, a client program, runs on a remote user's computer and communicates with a PDIO daemon using a TCP socket. Besides receiving images and sending visualization configurations, QuakeShow also blends the images with a background, provides various user controls, and captures mouse movements. Figure 2 and Figure 3 are two snapshots extracted from the animation submitted along with this paper.

In addition to the three main components described above, we have also defined an **application communication protocol** to support message exchanges (images and visualization configurations) between a client machine (QuakeShow) and a supercomputer (Hercules). PDIO delivers messages on behalf of both sides without interpreting the semantics. This design choice has not only simplified the implementation of the PDIO library and daemon, but also guaranteed the robustness of the whole system.

3 Enabling Techniques

This section briefly describes the enabling techniques that have made very large scale runtime visualization steering possible. We highlight important new features and omit the technical details, which are described in [3].

3.1 A New Parallel Volume Rendering Algorithm

The original visualization algorithm in Hercules renders volume using a parallel splatting method. It supports adaptive rendering based on the level of details specified. But its Achilles' heel is that it takes several seconds to several minutes to build a parallel visualization communication schedule due to an inherently sequential component that cannot be removed.

To overcome this obstacle and enable real-time user interaction, we have designed and implemented a new parallel ray-casting algorithm for volume rendering. Similar to the original one, our new algorithm supports adaptive rendering. Unlike the original one, the new algorithm is able to build (and rebuild when required by the user) a communication schedule in parallel much faster.[3] The key idea is to make use of the underlying distributed octree structure to perform efficient parallel sorting. As a result of this improvement, we are able to change view angles, adjust sampling steps, and zoom in and out, without noticing any interruptions of the incoming image stream when running large-scale problems

Another improvement in the new algorithm is a better compositing scheme that not only balances the workload of compositing but also reduces network traffic.[4]

3.2 PDIO Read

When initially developed in 2005 to support streaming data out of a supercomputer such as Cray XT3, PDIO supported only write operations. That is, an MPI program can call a PDIO write function (`pdio_write`) to send data to a PDIO daemon running on an I/O node, which in turn sends data to a remote user. To support runtime user steering, we have augmented the PDIO library and daemon to support the read function (`pdio_read`).

Whenever Hercules finishes sending an image by calling `pdio_write`, it attempts to read a new visualization configuration by calling `pdio_read`. A PDIO daemon checks the first cached message (stored in a FIFO) received from the client side. If there is no message available, the return value of `pdio_read` is 0, indicating there is no new configuration available. If the message size does not match the requested `pdio_read` size, an error has occurred in the application communication protocol between QuakeShow and Hercules. The message is discarded and an error code is reported to Hercules. If the sizes match, the first cached message is returned to Hercules. In case there are multiple cached incoming messages, Hercules drains the message queue (maintained by the PDIO daemon on a I/O node) to obtain the most recent visualization configuration by repeatedly calling `pdio_read` until it returns 0.

The advantage of this retrieval procedure is that we can decouple the execution of Hercules from QuakeShow and the delay of TCP/IP network transmission. A lock-step communication protocol (i.e. one image out, one visualization configuration in) could stop the execution on the supercomputer if a client machine is slow or if there is congestion in the TCP/IP network.

[2] Portals is a low level communication layer on top of which MPI is implemented.

[3] Note that the parallel communication schedule used by the visualization algorithm is unrelated to the one used by the solver.

[4] Here, we are referring to the traffic on a supercomputer's internal interconnection network (i.e. a 3D torus network) rather than the traffic on the public TCP/IP network.

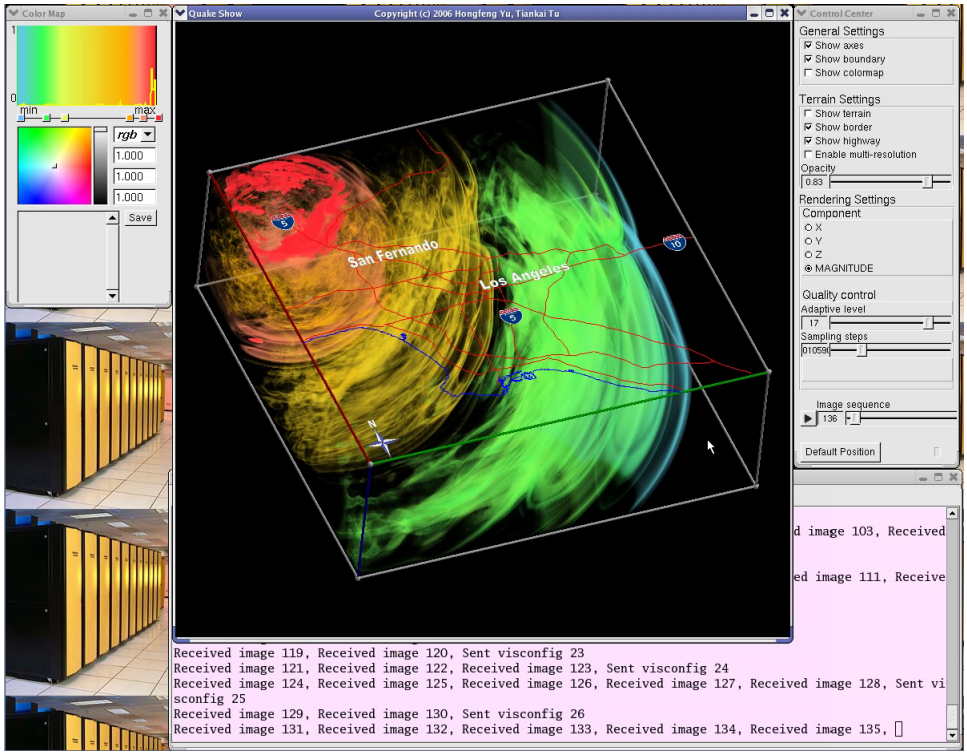

Figure 2: **Snapshot One of QuakeShow during an earthquake simulation.**

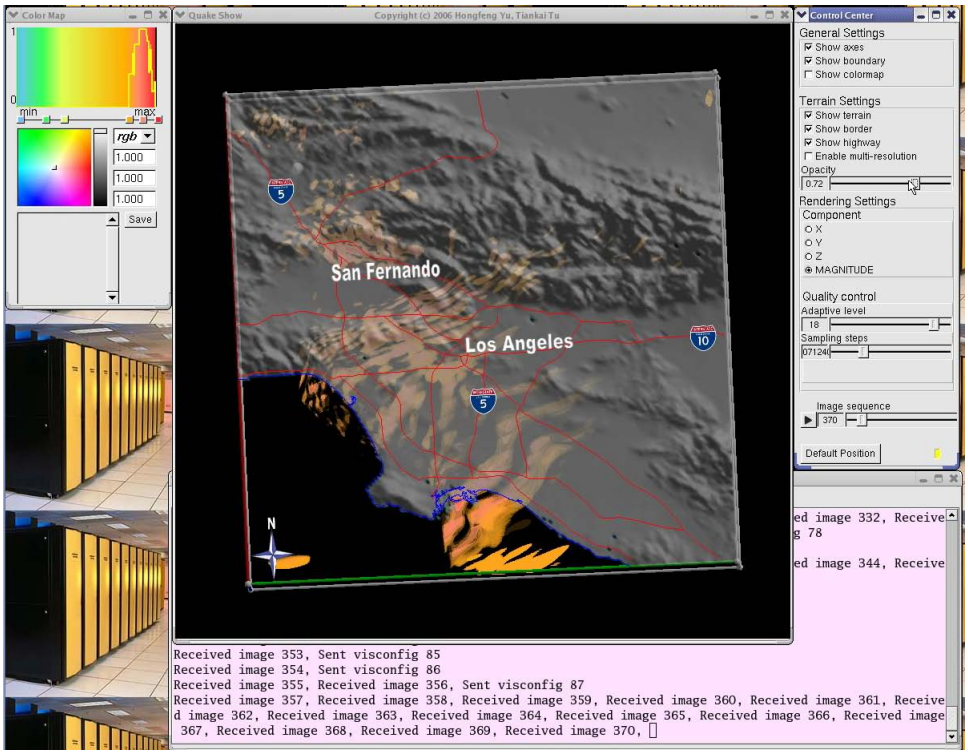

Figure 3: **Snapshot Two of QuakeShow during an earthquake simulation.**

Frequency	0.23 Hz	0.5 Hz	0.75 Hz	1 Hz	1.5 Hz	2 Hz
Elements	6.61E+5	9.92E+6	3.13E+7	1.14E+8	4.62E+8	1.22E+9
Nodes	8.11E+5	1.13E+7	3.57E+7	1.34E+8	5.34E+8	1.37E+9
Max leaf level	11	13	13	14	14	15
Min leaf level	6	7	8	8	9	9

Figure 4: **Summary of the meshes of different resolutions for earthquake wave-propagation simulations.**

3.3 QuakeShow

QuakeShow is instrumental for providing a powerful user interface. A user interacts with QuakeShow through mouse movements and clicks. Each click results in a request that is either serviced locally or sent to the remote supercomputer (a PDIO daemon). Local service renders the geographical context of multi-resolution terrain data, cities, borders, and highways. Remote requests are triggered whenever a user changes view angles, adjusts sampling steps, zooms in or out, or modifies the transfer function. QuakeShow does not send a new visualization configuration to the PDIO daemon until it receives an image.

The current implementation of QuakeShow is not multi-threaded. As a result, a user may experience some jitter while moving the mouse at the moment an image is being received.

4 Scientific Applications

The extended Hercules framework has been used to execute earthquake simulations that model seismic wave propagation during historical and postulated earthquakes in the Greater Los Angeles Basin, which comprises a 3D volume of $100 \times 100 \times 37.5$ kilometers. The earth property model is the Southern California Earthquake Center 3D community velocity model [1] (Version 3, 2002).

Figure 4 summarizes characteristics of the underlying octree meshes for a series of earthquake simulations we have run that are characterized by increasing maximum resolved seismic frequency [3]. The meshes range in size from 0.6 million to over 1.2 billion elements. Since the earth is highly heterogeneous, the largest elements are 64 times larger than the smallest ones (the difference between "max leaf level" and "min leaf level").

The submitted animation was generated by a screen capturing program running on a laptop computer (1.7 GHz Pentium M, 1 GB memory) where QuakeShow was running. No simulation data was pre-processed or stored before the screen-capturing program started. In other words, the animation is equivalent to a live demo.

The animation shows the first 10 minutes of executing Hercules on 1024 processors of the PSC Cray XT3 to simulate the 1994 Northridge earthquake at 0.5 Hz maximum resolved seismic frequency. The number of elements and nodes is 9.9 million and 11.3 million, respectively, as shown in Figure 4. The delta time (i.e. the duration of each simulation time step) to ensure numerical stability is 0.008719 (computed automatically by Hercules according to the material properties and resulting wave velocities). The entire duration of the simulated earthquake is 80 seconds, which translates to 9176 time steps. Visualization of solutions occurs every 10 time steps. If we had stored the necessary output for offline volume rendering instead of visualizing at runtime, the combined size of output files would have been 250 GB (= 11.3 million mesh nodes $\times$ 3 doubles per node per visualization step $\times$ 917 visualization steps).

Physical phenomena that are difficult to identify have been visualized effectively. For example, Figure 2 shows amplification of seismic waves in the San Fernando Valley, where the soil is soft (red waves). Figure 3 illustrates the strong residual seismic energy trapped in both the San Fernando Valley and the Los Angeles Basin, while the seismic waves in the nearby Santa Monica Mountains and San Gabriel Mountains have dissipated: a validation that sedimentary basins trap seismic energy during strong earthquakes. Among many other interesting discoveries is the channeling effect of the mountains: the seismic waves travel along the Santa Monica Mountains and into the Los Angeles Basin. (See the animation for details.)

5 Conclusions

We have developed a novel end-to-end scalable methodology for construction, execution, and visualization of large-scale parallel simulations. The Hercules system has enabled real-time, on-the-fly visualization and steering of earthquake simulations on supercomputers with thousands of processors. While some of the techniques presented in this paper are specific to the target class of octree-based discretization methods, the design principle and the overall software architecture are applicable to a wider class of numerical PDE solvers.

We have demonstrated the feasibility and advantages of monitoring, analyzing, and steering the outputs of very large-scale parallel simulations at runtime from a personal computer, thereby avoiding networking and storage bottlenecks associated with massive datasets. Because this end-to-end approach does not require moving large data over the network and is completely scalable, it points the way to integrated simulation and visualization on tens of thousands of CPUs and offers a promising approach to overcoming the challenges of visualization of petascale simulations.

Acknowledgments

This work is sponsored in part by NSF under grant IIS-0429334, by a subcontract from the Southern California Earthquake Center (SCEC) as part of NSF ITR EAR-0122464, by NSF under grant ITR EAR-0326449, by DOE under the SciDAC TOPS center grant DE-FC02-01ER25477, and by a grant from Intel. Supercomputing time at the Pittsburgh Supercomputing Center is supported under NSF TeraGrid grant MCA04N026P. We would like to thank our SCEC CME partners Tom Jordan and Phil Maechling for their support and help. Special thanks to Paul Nowoczynski, Jay R. Scott and Chad Vizino at PSC for their outstanding technical support.

References

[1] H. MAGISTRALE, S. DAY, R. CLAYTON, AND R. GRAVES, *The SCEC Southern California reference three-dimensional seismic velocity model version 2*, Bulletin of the Seismological Soceity of America, (2000).

[2] N. T. B. STONE, D. BALOG, B. GILL, B. JOHANSON, J. MARSTELLER, P. NOWOCZYNSKI, D. PORTER, R. REDDY, J. R. SCOTT, D. SIMMEL, J. SOMMERFIELD, K. VARGO, AND C. VIZINO, *Pdio: High-performance remote file i/o for portals enabled compute nodes*, in Proceedings of the 2006 Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, June 2006.

[3] T. TU, H. YU, L. RAMIREZ-GUZMAN, J. BIELAK, O. GHATTAS, K.-L. MA, AND D. R. O'HALLARON, *From mesh generation to scientific visualization — an end-to-end approach to parallel supercomputing*, in SC2006, Tampa, FL, November 2006.

[4] P. R. WOODWARD, D. H. PORTER, AND A. IYER, *Initial experiences with grid-based volume visualization of fluid flow simulations on pc clusters*, in Proceedings of Visualization and Data Analysis 2005 (VDA2005), San Jose, CA, January 2005.