Case Study: Mantle Convection Visualization on the Cray T3D

James S. Painter *
Advanced Computing Laboratory
Los Alamos National Laboratory

Hans-Peter Bunge †
Institute of Geophysics and
Planetary Physics
Los Alamos National Laboratory

Yarden Livnat ‡
Department of Computer Science
University of Utah

ABSTRACT

The recent years have seen rapid advancement towards viewing the Earth as an integrated system. This means that we have come to understand the interdependence of the major planetary subsystems — atmosphere, biosphere, oceans and the deep earth interior — on a large range of time and length scales. One of the longest time scales of the planet is imposed by solid state convection within the silicate Earth mantle.

Mantle convection modeling, and other earth science modeling efforts, now are producing simulation data on grids that are large enough to strain the memory and processing power of even the largest high-end graphics workstations. Another alternative is to use parallel visualization tools running on the massively parallel computers that generated the data. This is the approach that we have taken for the visualization of mantle convection simulation data.

Introduction to Mantle Convection Simulation

Solid state convection within the Earth's mantle determines one of the longest time scales of our planet. The Earth's mantle, the 2900 km thick silicate shell that extends from the iron core to the Earth's surface, though solid, is deforming slowly by viscous creep over long time periods. While gradual in human terms, the vigor of this subsolidus convection is impressive, producing flow velocities of 1-10 cm/year. Plate tectonics, the piecewise continuous movement of the Earth's surface, is the prime manifestation of this internal deformation, but ultimately all large scale geological activity of our planet, such as mountain building and continental drift, must be explained dynamically by mass displacements within the mantle.

A major problem for researchers in computational mantle dynamics is to resolve the Earth's outer 100 km deep skin, or lithosphere. This lithosphere is an integral part of the mantle and thus a 100 km wide spatial resolution has to be achieved throughout the volume. The resulting computational problem is formidable and numerical discretizations with 1-10 million grid points have to be formulated to resolve the mantle volume on scales of 50 km or less. The resulting computational problem has been largely intractable on conventional sequential computers, as it is necessary to follow the time evolution of pressure, temperature and velocity over the entire volume, requiring gigabytes of memory and computational speeds of many gigaflops. However, such problems are well in reach of modern parallel computers, such as the massively parallel Cray Research, Inc. T3D system.

To address this problem, we use the 3D spherical mantle dynamics code TERRA, which solves the Navier-Stokes equations in the infinite Prandtl number limit using a multigrid approach [1]. Discretization of the spherical shell is based on subdivision of of the regular icosahedron producing a data structure that is well suited for modern parallel hardware using domain decomposition and message passing [2]. A message passing version of TERRA runs on the 256 processor CRAY T3D at the Advanced Computing Laboratory (ACL) at Los Alamos National Laboratory [3]. On this machine our numerical modeling code shows excellent parallel performance, displaying a communication overhead of less than ten percent. The computational memory afforded by the T3D has allowed us to investigate convection employing a numerical grid of more than 20 million finite elements. We are thus able to resolve a large range of dynamical length scales within the mantle.

Parallel Visualization Tools

Visualization of the vast simulation data is a serious challenge, as it is necessary to display the large-scale flow without compromising resolution of small scale structure. Small scale structure is generated primarily in thermal boundary layers, such as the lithosphere at the top of the mantle, but swept around by the large scale flow throughout the mantle. Moreover the temporal evolution must be visualized by displaying long time-series of data, requiring capacity many thousand times that of the individual timestep.

To address this problem, we have developed visualization tools that run on the massively parallel computer where the data was generated. This allows for both a rapid and high resolution display of simulation results too large for visualization on even high-end graphics workstations. In addition, by running the visualization tools on the parallel computers used to generate the massive data, we avoid the need for time consuming and cumbersome data transfers of the simulation results. The ability to display fine simulation details, such as the very localized generation and evolution of thermal structures along major boundary layers, greatly enhances the physical interpretation and presentation of these new high resolution convection experiments.

The parallel visualization tools consist of an isosurface extractor, a parallel polygon renderer, and a parallel slicer that can interpolate arbitrary planar slices through field data. These tools use a message passing and active message programming model [11]. The tools operate directly on the TERRA grid structure. While the TERRA grid is not a structured grid, the recursive subdivision basis of the grid allows the grid geometry to be implicitly represented rather than explicitly stored, saving memory and allowing for efficient geometric queries of the grid.

* Advanced Computing Laboratory, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, **jamie@acl.lanl.gov**

† Institute of Geophysics and Planetary Physics , Los Alamos National Laboratory, Los Alamos, New Mexico 87545, **bunge@kokopelli.lanl.gov**

‡ Department of Computer Science, University of Utah, Salt Lake City, Utah 84112, **ylivnat@cs.utah.edu**

Parallel Rendering

Many researchers have studied parallel algorithms for polygon and volume rendering in recent years [5, 6]. Molnar *et al.*, provide a useful taxonomy for parallel rendering which classifies parallel rendering methods as sort-first, sort-middle, or sort-last according to where interprocessor communications occurs in the rendering pipeline [9].

Our T3D parallel renderer uses a sort-middle based rendering algorithm. Both the data domain and the image are partitioned evenly among the processors. Each processor first handles the geometric processing for the portion of the data it holds: isosurface extraction, arbitrary slicing and geometric transformation. The resulting geometric primitives are partitioned into scanline segments according to the portion of screen space they cover and sent to the processor responsible for that portion of the image using an active message communications model. Scanline segments are buffered and sent in groups to amortize the cost of a message over several scanline segments.

When the active message arrives at its destination processor, a handler function is invoked that completes the rasterization of the primitives it contains. Opaque scanline segments are directly *z*-buffered. Transparent scanline segments are buffered and handled after all processors complete geometric processing. The transparency segments are first depth sorted via a Newell-Newell-Sancha depth sort then composited front to back [10].

Parallel Slicing

Arbitrary slicing is handled through software based texture mapping. Each slice plane is clipped against a bounding box for each contiguous portion of data held on a processor. The resulting polygons are scan converted with a texture lookup at each pixel. The texture lookup function maps the pixel screen coordinates back to the simulation grid in two steps. First, the screen coordinates are mapped backwards into world coordinates via a four-by-four matrix multiply. These world coordinates are then mapped to a grid cell through a hierarchical grid search. The field values are interpolated within the grid cell and mapped, through a color map, to a surface color. Colored scanline segments are sent, via an active message, to the processor responsible for their scanline, as described earlier.

The mapping from world coordinates to grid coordinates can be done in $O(\log N)$ time * or the TERRA grid because of the subdivision nature of the grid. We have also implemented slices for fields defined over uniform regular grids, which can be indexed directly in $O(1)$ time. While we have not implemented slicing on arbitrary unstructured grids, the texture mapping method could be applied but would require a more elaborate search strategy.

An alternative approach to slicing is to generate a full geometric intersection of the grid with the slice plane and render the resulting polygons using the normal rendering pipeline. Our method has two benefits over this alternative. The texture mapping based slice rendering time grows very slowly with the size of the grid ($\log N$ for the TERRA grids). Further, the slice plane can be interactively changed, since no expensive preprocessing work is done when the slice plane is changed.

Parallel Isosurface Extraction

In traditional isosurfacing, *e.g. marching cubes* [8], the entire data set is traversed in order to extract a single isosurface. This method is not applicable in our case where interactive speed over huge data sets is essential.

The disadvantages of the marching cubes algorithm are two fold. First, the number of cells in a single isosurface are usually very small compare to the the total number of cells. Second, the decision of whether a cell does or does not intersect the isosurface is as expensive as actually computing the intersection, *i.e.* no trivial rejection.

More advanced algorithms employ a pre-process stage where the data set in examined and some key information is retained. Trivial rejection can be achieved by retaining the minimum and maximum values attained in each cell. The first issue, reducing the search time, i.e. checking as few cells as possible against the giving isovalue, is much tougher and many algorithms have been developed to address this issue.

The TERRA data sets can be characterized as both structured and unstructured. The geometry of a cell can be inferred quickly from its index and thus it does not have to be saved explicitly for each cell. On the other hand, the overall structure of the data set makes it difficult to utilizes structured methods such as Wilhelms and Van Gelder's octree [12].

We chose to use the NOISE algorithm [7] which is both near-optimal in time and is flexible enough to handle any geometry. The algorithm has a complexity of only $O(K + \sqrt{N})$ in the worst case, where N is the number of cells in the data set and K is the number of cells which actually intersect a given isosurface. The algorithm is near-optimal in the sense that in practice, K is almost always greater then $\sqrt{N}$. NOISE is based on the projection of the data set onto the *span space*, where each cell is represented by a single point with 2 coordinates: the minimum and maximum values attained in that cell. The projection is perform only once as either an offline stage or, as in our case, in each processor after the data is distributed. In order to achieve a near optimal performance, the algorithm takes advantage of a kd-tree to hold the projected points in the span space. The memory requirement for the kd-tree itself is only a single id per cell. The kd-tree structure is saved implicitly via the order in which these id's are sorted. This property of the algorithm makes it particular useful for very large data sets where both the speed up of the search *and* memory requirements are of great importance. Furthermore, NOISE can be used with or without saving explicitly the minimum and maximum values of each cell. Again, the trade off is between speed and memory. We took advantage of the large amount of memory on the CRAY T3D and aimed at accelerating the search by saving this additional information.

The parallel version of NOISE takes advantage of the flexibility of the algorithm with respect to the structure of the data set and the sort-middle parallel renderer discussed earlier. The data distribution is left to the renderer and NOISE is applied locally to the data on each processor. Isosurface extraction is then initiated by distributing only the new isovalue. The extracted local isosurface on each processor is temporarily added to the data set to be rendered. It is then left to parallel renderer to distribute the rendering of the isosurface to the other processors, a task which the renderer performs for the rest of its local geometric data.

The integration of the isosurface extraction and the parallel renderer enable us to take advantage of the renderer transparency capability. The use of transparency and the rapid isosurface extraction achieved by the algorithm also enabled the rendering of several isosurfaces at the same time. Each such isosurface is extracted independently from the others. The NOISE algorithm can determine, virtually at no cost, the amount of storage needed for the isosurface *before* it is computed. We take advantage of this capability and refrained from allocating memory by reusing the memory allocated to the previous isosurface if it is large enough.

Image Delivery

Delivering images to the desktop is often the limiting factor for interactive parallel renderers. While a parallel renderer may be able to render frames at interactive rates, it can be difficult to get them

* N is the number of grid cells

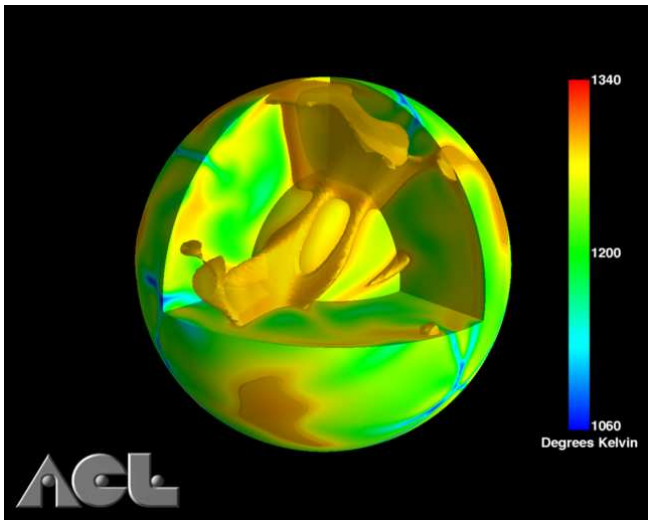

Figure 1: Slicing and Isosurfacing of the Temperature Field

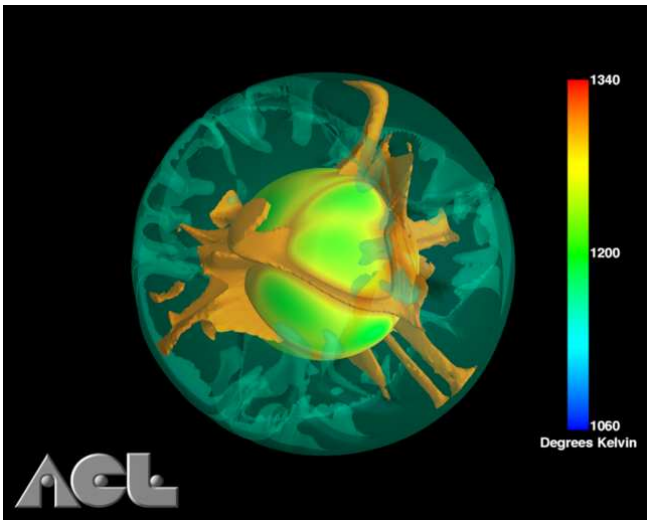

Figure 2: Two Iso-Temperature Surfaces, With Transparency

off the parallel machine and onto a users display quickly enough to allow interaction. In our case, the final image is gathered and displayed using either a HIPPI frame buffer or X11 output to the user's workstation. HIPPI frame buffer output provides the fastest performance: up to 10 1280 by 1024 high resolution frames per second. Our HIPPI frame buffer is switchable to a number of monitors, providing high speed interaction for users with access to one of these monitors.

Image delivery over X11 is useful for users remotely located from the T3D. High performance image display over X11 is difficult. A full color high resolution (1280x1024) frame uses 5 MB, uncompressed, requiring 50MB/sec of bandwidth, much higher than is available across 10Mb ethernet or over long haul internet connections. To alleviate these bandwidth needs, our parallel renderer compresses each frame to be display, in parallel. The compressed image data is gathered to a single processor, and then sent, via a socket, to a display process running on the users workstation. We use **zlib**, a general purpose lossless compression library. Interactive control is provided by a graphical user interface (GUI) running on the user's workstation. Typical frame rates on 64 T3D nodes are 2-10 frames per second for 1280x1024 HIPPI output, including slicing and isosurface generation on each frame. X11 output is slower and highly depends on network speed, however we have observed greater than 2 FPS at 640x512 resolution between our T3D at LANL and remote workstations across the country.

Results

Figures 1 and 2 illustrate two example stills from the TERRA visualization tools. These figures and the accompanying video are rendering a static data set at a resolution of 1.25 million grid cells.

Figure 1 shows the temperature field over the spherical model of the Earth. In this simulation the viscosity (or stiffness) of the mantle fluid increases with depth by a factor of 30, as suggested by geophysical observation. This one parameter change results in a dramatic difference in convection structure displaying elongated downwellings from the upper surface instead of pointlike patterns typical for isoviscous flow [4]. Such elongated downwellings are analogous to Earth's linear subduction zones where plates dive under one another.

The outer surface of the sphere is a spherical radial shell cutting through the grid structure. Note that the sphere is hollow and an inner shell is shown as well. The simulation model covers only the outer 50% of the Earth's diameter where the mantle exists. A "wedge" has been removed by 3 slicing planes. Within the wedge opening an isosurface has been extracted with a "hot" temperature value. The isosurface and grid slices give insight on how hot material convects upwards through the Earth mantle.

Figure 2 again shows the temperature field with two isosurfaces over an inner spherical radial shell. The outer blue transparent isosurface used a relatively cold temperature, indicating where cold material moves back toward the interior of the mantle. The inner orange opaque surface is a relatively high iso-temperature surface and again illustrates hot material moving outwards.

The accompanying video illustrates an exploration of a single time step snapshot of the TERRA simulation results. While the video was rendered in batch mode, the rendering rate was still over 3 frames per second, excluding image write time.

Conclusions and Future Work

Parallel visualization tools have proven to be an invaluable aid in interpreting mantle convection simulation results. Because of the grid sizes involved, commercial visualization software running on high end graphics workstations was too slow and used too much memory to be an effective interactive exploration tool. A much faster and more convenient visualization system is possible for these "massive data" problems by running the visualization tools directly on the massively parallel processor where the data was generated.

Current and future work with TERRA is aimed at incorporating the information of the geological record into the mantle convection simulations, primarily the effects of continents and tectonic plates at the surface of the Earth. These new simulations are expected to improve substantially our understanding of the temporal evolution of Earth's mantle.

References

[1] J.R. Baumgardner. Three dimensional treatment of convective flow in the Earth's mantle. *J. Stat. Phys*, 39(5-6):501–511, 1985.

[2] J.R. Baumgardner and P.O. Fredrickson. Icosahedral discretization of the two sphere. *Siam J. Numer Anal.*, 22(6):1107–1115, 1985.

[3] H.-P Bunge and J.R. Baumgardner. Mantle convection modeling on parallel virtual machines. *Computers in Physics*, 9(2):207–215, 1995.

[4] H.-P Bunge, M.A. Richards, and J.R. Baumgardner. Effect of depth-dependent viscosity on the planform of mantle convection. *Nature*, 379:436–438, 1996.

[5] IEEE Computer Society. *1993 Parallel Rendering Symposium Proceedings*. ACM SIGGRAPH, October 1993.

[6] IEEE Computer Society. *1995 Parallel Rendering Symposium Proceedings*. ACM SIGGRAPH, October 1995.

[7] Yarden Livnat, Han-Wei Shen, and Christopher R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996.

[8] W.E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[9] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, July 1994.

[10] J. Newell, R. Newell, and T. Sancha. A solution to the hidden surface problem. In *Proceedings ACM National Conference*, pages 443–450, 1972.

[11] J. Painter, P. McCormick, M. Krogh, C. Hansen, and G. Colin de Verdière. The ACL message passing library. *EPFL Supercomputing Review*, 7, November 1995.

[12] J. Wilhelms and A. Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.