ADAPTIVE, MULTIRESOLUTION VISUALIZATION OF LARGE DATA SETS USING PARALLEL OCTREES*

LORI A. FREITAG† AND RAYMOND M. LOY‡

Abstract. The interactive visualization and exploration of large scientific data sets is a challenging and difficult task; their size often far exceeds the performance and memory capacity of even the most powerful graphics workstations. To address this problem, we have created a technique that combines hierarchical data reduction methods with parallel computing to allow interactive exploration of large data sets while retaining full-resolution capability. The hierarchical representation is built in parallel by strategically inserting field data into an octree data structure. We provide functionality that allows the user to interactively adapt the resolution of the reduced data sets so that resolution is increased in regions of interest without sacrificing local graphics performance. We describe the creation of the reduced data sets using a parallel octree, the software architecture of the system, and the performance of this system on the data from a Rayleigh-Taylor instability simulation.

Keywords. Interactive Visualization, Multi-Resolution Visualization, Adaptive Visualization, Parallel Octrees

1. Introduction. A critical step in the computational solution of application problems is the interactive exploration and visualization of the resulting data sets. However, in today's computational environment, the data sets produced by simulations performed on giga- and teraflop-scale massively parallel processors (MPPs) often exceed the memory and performance capacity of typical graphics workstations. Currently, interactive performance of high-end graphics workstations is limited to data sets that contain roughly 256^3 data points. Unfortunately, this is an order of magnitude smaller than many data sets produced by large-scale simulations, and the discrepancy between what is easily visualized and what scientists would like to visualize is likely to continue. Thus, the interactive visualization of very large data sets requires either (1) a postprocessing step to reduce the number of data points sent to the visualization environment or (2) sophisticated parallel rendering algorithms that work with the full-resolution data set. Both techniques have been used successfully. In this paper we present an adaptive, multi-resolution data reduction method that combines both approaches.

To create a reduced data set, researchers commonly build a hierarchical, multiresolution representation of the data or geometric model to be visualized (see [1] for an overview of these methods). In these techniques, a series of coarse representations of the data is constructed using, for example, quadtrees or octrees [2, 3], progressive meshes [4, 5], or wavelets [6]. The level of detail in each region is controlled through a variety of mechanisms, such as error tolerance bounds that control fidelity to the original model, or user input, such as field of view. These methods are useful for fast navigation through the data set, but maximum resolution is limited by the memory size and speed of the graphics workstation.

In contrast, parallel rendering algorithms often work with full-resolution data sets either as the computation proceeds or as a postprocessing step [7, 8, 9, 10]. The data is distributed across the processors of the MPP, and derived visualization entities such as isosurfaces or streamlines are computed in parallel and communicated to the graphics workstation for display. The advantage of this technique is that no information is lost in a data reduction process; however, for scientists

* The authors were supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

†Assistant Scientist, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. freitag@mcs.anl.gov.

‡Postdoctoral Appointee, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. rloy@mcs.anl.gov.

who run their parallel applications at remote sites, there are two potential drawbacks to using this method for interactive visualization. First, many MPP supercomputers are accessed through a batch queuing system, making it difficult to predict when the "interactive" job will run. Second, these techniques are useful for interactive requests of derived visualization entities *only* if there is sufficient network bandwidth between the graphics workstation and the parallel computer.

To address these problems, we have developed a system that combines data reduction techniques and parallel computing to obtain fast performance on a local graphics workstation while retaining access to the full-resolution data set. We first distribute the data across the processors of a parallel computer and create a hierarchical representation of it using a distributed-memory octree data structure. Various levels of detail are sent to the local visualization environment for interactive navigation and exploration via a communication mechanism such as the ALICE memory snooper [11] or CAVEcomm [12]. The octree may be easily adapted to reflect changing level of detail required by the visualization; additional detail is obtained by refining the octree, whereas reduced detail is obtained by pruning or truncated traversal. Thus the user can zoom into regions of interest without sacrificing local performance by simultaneously coarsening the data outside the region of interest. The user can evaluate the fidelity of the reduced data set to the original data by visualizing error indicators that are computed and sent to the graphics workstation for display.

The remainder of this paper is organized as follows. In Section 2, we describe the creation of the reduced data set including the parallel octree infrastructure. In Section 3, we describe the software architecture of the system, both the envisioned final toolkit and its current instantiation. In particular, we discuss the communication mechanisms that enable the parallel octree code and the visualization environment to run on separate, possibly remote, computers. In Section 4, we show the use of this system to visualize data from a Rayleigh-Taylor instability simulation. We provide performance results for the creation of the octree, the communication of data across an ethernet network, and the visualization frame rates achieved for each reduced data set. Finally, in Section 5 we offer concluding remarks and indicate additional features and results that will be included in the final paper.

2. Creating Reduced Data Sets. Our data reduction technique is based on creating a hierarchical representation of the data using a parallel octree. We first describe the criteria used for inserting field data into the octree, our approach for creating a general interface that can be used with a number of mesh types, and the error indicators associated with the reduction stored on each octant leaf. We then describe the parallel octree infrastructure, including the data structures and techniques used for efficient parallel creation, coarsening, and traversal of the tree.

2.1. Data Reduction. To create the reduced data set, the user must loop through the field data and call the octree code with spatial locations and their field data values. The spatial locations are used to insert the field data value into the appropriate leaf octant; a user-defined criterion indicates whether the octant should divide to create eight new leaf octants. We currently divide an octant leaf when a user-defined maximum number of elements have been inserted into it. This technique maintains approximately uniform fidelity to the solution data for $h-$adaptive meshes in which the error at each data point is roughly constant. For uniform meshes or $p-$adaptive methods, we plan to use error indicator techniques such as energy norms or gradient bounds to determine when octants should be subdivided.

The field data values are averaged or otherwise agglomerated after insertion into the octree. To provide an indication of the error associated with the reduced data set, for each leaf octant we compute and store statistical values such as the standard deviation, σ, and maximum deviation from the mean, e. These values are normalized by the mean to yield σ_n and e_n, respectively, which are included as additional scalar fields to be visualized so that the user has an indication of the fidelity of the reduced data set to the original data set. These measures of error also serve to highlight

potential regions of interest; the cells with a large deviation from the average value are likely to have fine-scale structure that was not adequately captured by the reduction process.

2.2. Parallel Octree Infrastructure. To effectively manage a large distributed data set, the octree must also be distributed across the processors of the MPP. Efficient traversal of the parallel octree data structure is enabled by interoctant links, which may be either local or off-processor. Off-processor parent links are represented by a local root structure containing spatial information to enable local point searches without communication (Figure 2.1). An octree partitioning algorithm [13] together with arbitrary octant migration is used for load balancing of the mesh data and associated octree. If the data reduction were closely coupled to the parallel application, octant migration could be performed to track the location of the application data.

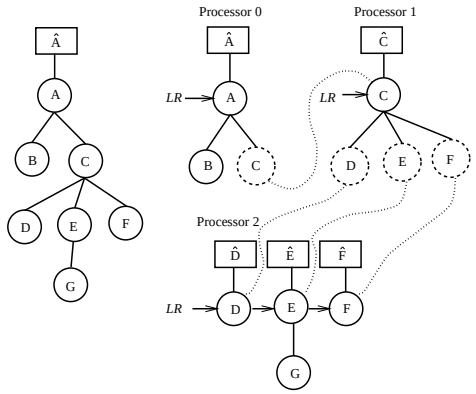

FIG. 2.1. *A portion of an octree (left) and the same tree distributed across three processors (right). Dotted links are nonlocal; squares indicate local root spatial information.*

In most cases, octree insertion, refinement, and coarsening may be performed without the need for interprocessor communication. The reduced data set is easily formed by traversing each local octree and computing the values on each leaf. This can be done on each processor in linear time with respect to the local number of data points. Complex algorithms operating on the reduced data set (such as streamlines or isosurface generation) may require face-neighbor links or a guarantee of maximum level-difference between adjacent octants. Providing this functionality complicates octree insertion but may be done in a highly asynchronous fashion.

3. Software Architecture. The interactive data reduction system described in this paper is comprised of four major components:

- the parallel octree code described in Section 2,
- the local visualization environment, which can consist of externally developed desktop tools such as the General Mesh Viewer (GMV) [14], IRIS Explorer [15], or custom tools built for state-of-the-art display devices such as the CAVE virtual reality theater [16],
- field data input to the parallel octree code through either files or interactive requests to a running application, and
- a portable communication infrastructure that allows the user to make interactive requests for new reduced data sets from the visualization environnment and allows the octree code to obtain new field data from the running application.

These four components and their interactions are shown in Figure 3.1. The arrows between the components indicate the communication necessary for an interactive environment. The width of the arrows indicates the relative size of the messages; small messages are needed to request new reduced data sets or field data values; large messages are required to transfer information from the

application to the octree code and from the octree code to the visualization environment. Solid boxes and solid arrows indicate components and interactions that currently exist; dashed box outlines and arrows indicate components and interaction models that are planned for future instantiations of the toolkit. In the visualization tool boxes, the packages listed in roman font are currently supported; near-term support is planned for those in italics. Asterisks indicate extensible toolkits that will support interactivity; the others must be used with file input and output.

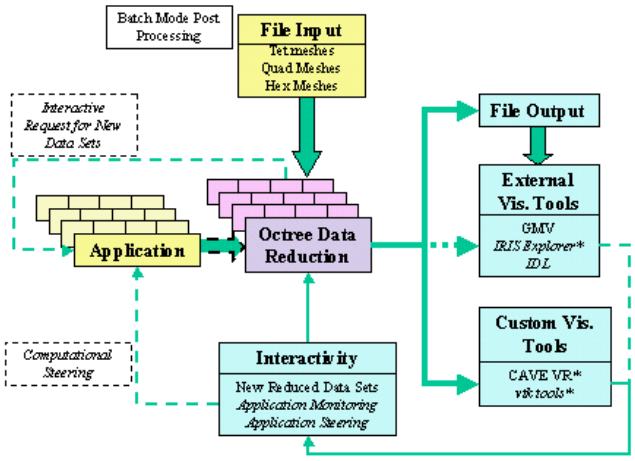

FIG. 3.1. *The four primary components of the data reduction toolkit and their interactions*

In the current instantiation, field data is loaded from a file into the parallel octree code. This step can occur either in a batch processing mode, where the element bound is statically imposed, or interactively via requests from the user. In batch mode, the values computed at the leaf octants are stored to a file; in interactive mode, these values are communicated directly to a graphics workstation for visualization.

We currently support two visualization environments: an externally developed desktop tool, GMV, and a custom CAVE visualization tool developed at Argonne National Laboratory. GMV is a freely available software package from Los Alamos that is distributed in binary form. It is lightweight and easy to use but not extensible to support interactive requests to the parallel octree code. We therefore use it to test the utility of the batch processing mode described above. The CAVE virtual reality theater was developed at the Electronics Visualization Laboratory at the University of Illinois at Chicago. The user manipulates objects in the CAVE using a wand, a three-dimensional analogue of a mouse. We used this functionality to develop an interactive visualization system to test the performance of the dynamic adaptive level-of-detail aspects of our system. User requests and data are transferred directly between the octree code and the CAVE using the communication infrastructure described below.

3.1. The Communication Infrastructure. The communication infrastructure for this software system must meet the following design requirements.

- It must be *portable* and *flexible* so that the parallel octree code and the visualization environment can run on a variety of computer architectures. And ideally, it should be *supported*.
- It must allow *dynamic linking* and decoupling of multiple processes so that (1) the user can periodically monitor a long-running application and (2) multiple scientists can collaborate while visualizing their data.

Several communication packages or specifications meet these design requirements, including Nexus [17], the MPI-2 specification [18] (although dynamic process management has not yet been implemented on most MPP architectures), PVM [19], and various tools built with these communication infrastructures. For our initial tests we use the CAVEcomm library [12] developed at Argonne National Laboratory, which uses Nexus. CAVEcomm uses a client-server model in which a broker mediates the communication between the parallel octree code and one or more CAVE environments. Figure 3.2 shows example interactions required between the CAVEcomm broker, the parallel octree code, and the local CAVE visualization process. After the initial connection is made, the parallel octree code and the visualization environment interact directly without mediation from the broker. The octree code waits to receive a request from the visualization environment for a new reduced data set. Once that request is received, the octree is refined and coarsened as required, and the new octant leaves are sent to the visualization environment for display. This process is repeated until the user has completed the exploration of the data set.

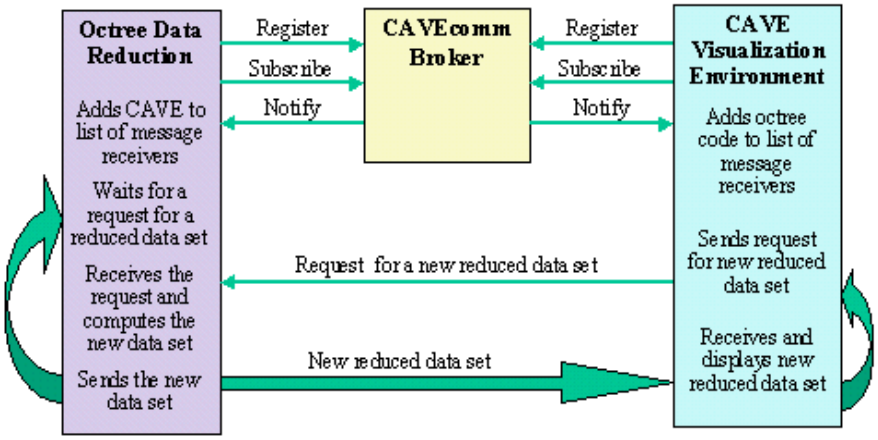

FIG. 3.2. *The communication interactions between the CAVEcomm broker, the visualization process, and the parallel octree code*

As stated earlier, the long-term goal of this project is to dynamically link to and monitor a running application. One concern when building the octree from a running parallel application is ensuring that the data is consistent; that is, we must access information at known synchronization points (for example, between time steps of a time-dependent code). One toolkit that helps manage the synchronization of distributed data sets is the ALICE Memory Snooper [11]. This toolkit uses a client-server model and allows one or more processes to dynamically link to, monitor, and change an application's published memory space at user-defined synchronization points. Thus it can be used both for fulfilling interactive requests for new data sets from a long-running simulation and for modifying or "steering" the simulation. We therefore plan to replace the CAVEcomm library with the ALICE Memory Snooper in the final instantiation of the data reduction toolkit.

4. System Performance. As part of a joint project with the University of Chicago, we are working toward complete simulations of thermonuclear flashes on astrophysical bodies such as neutron stars or white dwarves. The target problem size for these simulations is 1024^3 grid points, which exceeds the limits discussed earlier for graphics workstations. Therefore, data reduction techniques are necessary. One crucial aspect of these simulations is the correct modeling of the flame front as it moves away from the surface of the compact star during the deflagration stage. Because the relatively dense nuclear fuel lies above the nuclear ash, the flame front is subject to Rayleigh-Taylor

instabilities, which can dramatically alter the shape and area of the burn region and, consequently, the duration and strength of the nuclear flash. In Figure 4.1, we show a single-plume Rayleigh-Taylor instability from an adaptive PPM code at two different levels of refinement This code is a descendant of PARAMESH [20] and PROMETHEUS [21]. The figure on the right was computed at a higher resolution than the figure on the left, and we can see a corresponding increase in the fine-scale features evidenced in the numerical simulation. The increase in fine-scale features can dramatically affect the large-scale features, and large problems sizes are therefore necessary to accurately model this phenomenon.

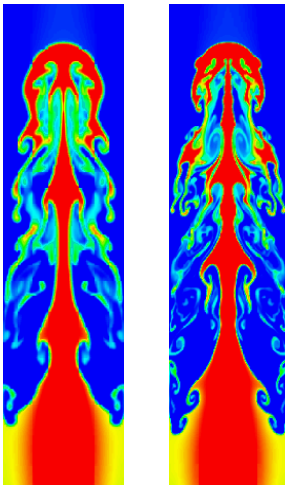

FIG. 4.1. *A single plume Rayleigh-Taylor instability modeled in two dimensions as a compressible fluid with the Euler equations at two different levels of refinements: 14848 mesh elements (left) and 54208 mesh elements (right).*

The data reduction technique of Section 2 was used to preprocess data for display using both GMV and the CAVE. We consider both two- and three-dimensional Rayleigh-Taylor data sets; the two-dimensional one described above, the three-dimensional one from an unstructured discontinuous Galerkin Euler code [13]. The problem mesh types, data sizes, and dimensionalities are given in Table 4.1. Data reduction was performed by an SGI Onyx (4×250 MHz MIPS R4400 processors). The CAVE virtual reality theater and CAVEcomm communications broker were both run on an Onyx2 Reality Monster (16×250 MHz MIPS R10000 processors). These machines were connected by local ethernet with 6Mb/s capacity [22].

TABLE 4.1
Rayleigh-Taylor data set characteristics

Dimension	Mesh Type	Data Location	Data Size	Display Device
2-D	Block Structured	Cell-Center	5.4×10^4	GMV, CAVE
3-D	Tetrahedral	Cell-Center	3.3×10^5	CAVE

Let I denote the maximum number of field data points allowed in a leaf octant. For the experiments presented here, the mesh data set is read from a file and inserted into a coarse initial octree using the criterion $I = 1024$. This initial construction took 2 seconds for the two-dimensional data set and 15 seconds for the three-dimensional data set. The level of detail in the entire domain is adjusted by traversing and adapting the existing tree to conform to the new criterion.

For the two-dimensional Rayleigh-Taylor data set, we show three levels of detail in Figure 4.2 corresponding to $I = 128$, 32, and 8. In each case we show the reduced data set image, the error

plots associated with that level of detail, and the slice through the associated octrees. The light-colored octant leaves in the middle row of figures indicate regions in which the error associated with the data reduction is large. These regions correspond to the sharp density interface between the two incompressible fluids. As the user requests finer detail, more leaf octants are created, the regions of large error reduce in size, and more fine scale features are visible.

In Table 4.2, we give the performance results for the cases shown in Figure 4.2 as well as for three levels of detail for the three-dimensional set. Recall that σ_n is the normalized standard deviation, and e_n the normalized maximum deviation from the mean. For each level of detail, we give the number of leaf octants generated, N; the percentage of the full data set to which N corresponds, P; the maximum σ_n over all octants; the maximum e_n over all octants; the average σ_n ; and the average e_n. The first two values give the worst-case scenario for fidelity to the original data set; the latter two values give a measure of the overall quality of the data reduction. We also include the time in seconds to revise the octree, T_R; the time to transfer the new reduced data set to the visualization environment, T_C; and the frame rate, F_R, for the CAVE visualization tool (measured in frames/second).

TABLE 4.2
Results for two- and three-dimensional Rayleigh-Taylor simulation data sets

N	P	Max. σ_n	Max. e_n	Avg. σ_n	Avg. e_n	T_R	T_C	F_R
2-D Rayleigh-Taylor								
820	1.6	0.397	1.081	0.134	0.294	0.48	0.17	225.50
3280	6.2	0.396	0.932	0.081	0.154	0.57	0.49	85.83
13120	25.0	0.382	0.569	0.042	0.050	1.10	1.54	24.27
3-D Rayleigh-Taylor								
9677	2.6	0.090	0.232	0.025	0.053	1.75	1.02	17.00
19933	5.3	0.088	0.206	0.019	0.035	2.28	2.21	8.26
43977	11.8	0.085	0.153	0.014	0.023	3.27	4.95	3.69

As before, our criterion for creation of the reduced data sets in two dimensions is $I = 128,\ 32$, and 8 and in three dimensions is $I = 128,\ 64$, and 32. Thus in each successive case, N is quadrupled in two dimensions and doubled in three dimensions. The largest errors in both two and three dimensions are located along the discontinuity between the two fluids simulated in the Rayleigh-Taylor instability. Because these features are much smaller than the leaf octants, the maximum σ_n (about forty percent in two dimensions and nine percent in three dimensions) only slowly decreases as N increases. The average σ_n decreases more significantly as N increases; the biggest gains in accuracy of the reduced data are achieved in the first increase of N, and fairly accurate representations of the data sets (eight percent error in 2-D; two percent error in 3-D) are achieved for $P \approx 5$. To further illustrate the relationship between the error measures and N, we plot average σ_n and e_n values in both two and three dimensions for criterion of $I = 2 \ldots 1024$ in two dimensions and $I = 4 \ldots 1024$ in three dimensions in Figure 4.3. As N increases, the corresponding decreases in σ_n and e_n both approach zero as well as each other. The times, T_R and T_C, vary linearly with N, and the inverse frame rate $\frac{1}{F_R}$ also varies linearly with N. We note that T_R grows more slowly than T_C and, for large values of N, the value of T_R is less than T_C demonstrating that, as expected, the bottleneck of this procedure is communication.

5. Future Work for SC99. For the final paper, we will add two new functionalities to the parallel octree code to improve its flexibility. In particular, we will provide more options for averaging or interpolating the data as it is inserted into the octree. We will also provide a subroutine stub

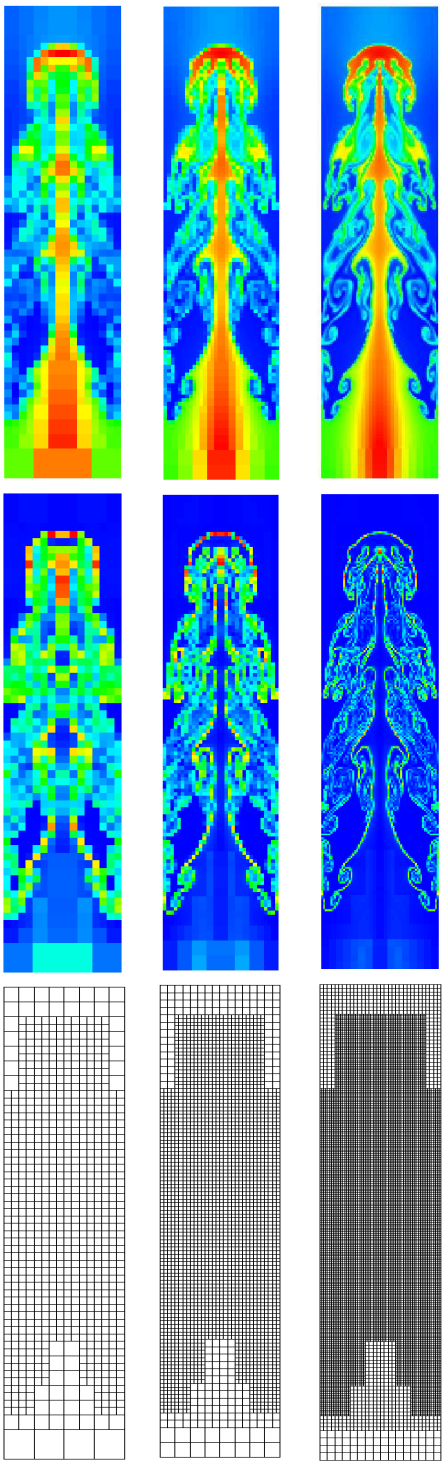

FIG. 4.2. *From left to right, the two-dimensional Rayleigh-Taylor image at three increasing levels of detail. For each level of detail we show density (top), the standard deviation resulting from the data reduction (center), and a slice through the corresponding octree (bottom).*

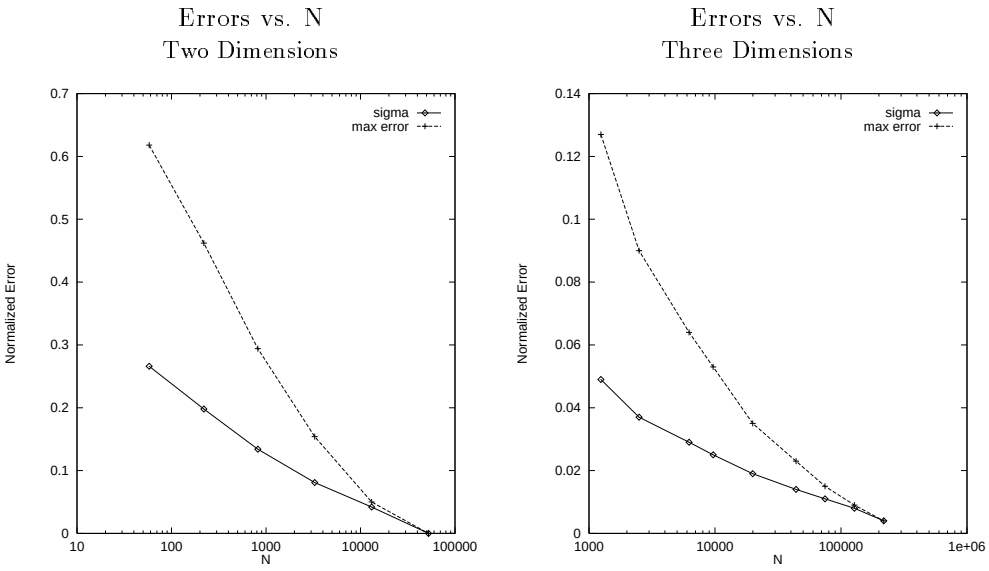

FIG. 4.3. *The relationship between N and the error measures average σ_n (sigma) and average e_n (max error) in both two and three dimensions*

that allows users to custom design the data reduction method most appropriate for their simulation data. Moreover, we will provide more default options for determining when to subdivide leaf octants including gradient tests on scalar fields.

In addition to the results in Section 4, we plan to include performance results for the parallel octree code running on several processors of the IBM SP and communicating via the ALICE Memory Snooper to graphics workstations located at both Argonne and the University of Chicago. These results will compare performance of a local ethernet connection, a 10 Mb/s ethernet WAN between ANL and Chicago, and an ATM WAN between ANL and Chicago. In addition, we will demonstrate the use of this tool with at least one Gigabyte data set from the simulation of an explosion on the surface of a neutron star.

Acknowledgments. We acknowledge the FLASH project for providing the motivating application described in this paper. In particular, we thank Mike Singer of the University of Chicago for his help with GMV and for making the two-dimensional Rayleigh-Taylor simulation results available for testing.

REFERENCES

[1] Paul Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proceedings of Graphics Interface 94*, pages 43–50, 1994.

[2] Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-time, continuous level of detail rendering of height fields. In *Computer Graphics Proceedings SIGGRAPH 96, Annual Conference Series*, pages 109–118. ACM, 1996.

[3] Brian Von Herzen and Alan Barr. Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics Proceedings, SIGGRAPH 87*, volume 21, pages 103–110. ACM, 1987.

[4] Hugues Hoppe. Efficient implementation of progressive meshes. Technical Report MSR-TR-98-02, Microsoft Research, Microsoft Corporation, 1998.

[5] Hugues Hoppe. Progressive meshes. In *Computer Graphics SIGGRAPH 96 Proceedings*, pages 99–108, 1996.

[6] Jos Roerdink and Michel Westenberg. Wavelet-based volume visualization. Technical Report IWI 98-9-06, Institute for Mathematics and Computing Science, University of Groningen, 1998.

[7] T. W. Crockett and T Orloff. A MIMD rendering algorithm for distributed memory architectures. In *Proceedings of the Parallel Rendering Symposium*, pages 35–42, 1993.

[8] Thomas Crockett. Beyond the renderer: Software architecture for parallel graphics and visualization. Technical Report ICASE Report No. 96-75, Institute for Computer Applications in Science and Engineering, 1996.

[9] Kwan-Lui Ma. Parallel rendering of 3D AMR data on SGI/Cray T3E. In *Proceedings of the Frontiers 99 Conference*, pages 138–145, 1999.

[10] Robert Haimes. pV3: A distributed system for large-scale unsteady CFD visualization. *AIAA paper*, 94-0321, January 1994.

[11] Ibrahima Ba, Christopher Malon, and Barry Smith. Design of the ALICE Memory Snooper, http://www.mcs.anl.gov/ams, 1999.

[12] T. L. Disz, M. E. Papka, M. Pellegrino, and R. L. Stevens. CAVEcomm user's manual document. World Wide Web Document, 1995. http://www.mcs.anl.gov/FUTURES_LAB/VR/ APPS/C2C/.

[13] Joseph E. Flaherty, Raymond M. Loy, Mark S. Shephard, Boleslaw K. Szymanski, James D. Teresco, and Louis H. Ziantz. Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws. IMA Preprint Series 1483, Institute for Mathematics and Its Applications, University of Minnesota, 1997. To appear, *J. Parallel and Dist. Comput.*

[14] Frank Ortega. General mesh viewer, user's manual, 1999.

[15] IRIS explorer, release 3.5, user's guide, Unix version, 1993.

[16] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *ACM SIGGRAPH 93 Proceedings*, pages 135–142. ACM, 1993.

[17] Ian Foster, Carl Kesselman, and Steve Tuecke. The Nexus approach to integrating multithreading and communication. *Journal of Parallel and Distributed Computing*, 37:70–82, 1996.

[18] MPI-2: Extensions to the message passing interface. http://www.mpi-forum.org/docs/docs.html, 1997.

[19] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM: Parallel Virtual Machine*. MIT Press, 1994.

[20] P. MacNeice, K. Olson, M. Mobarry, R. de Fainchtein, and C. Packer. A parallel adaptive mesh refinement community toolkit. *Submitted to Computer Physics Communications*, 1999.

[21] B. Fryxell, E. Müller, and D. Arnett. Hydrodynamics and nuclear burning. *Max-Planck-Institut für Astrophysik Report 449*, 1989.

[22] D. Diachin, L. Freitag, D. Heath, J. Herzog, B. Michels, and P. Plassmann. Collaborative virtual environments used in the design of pollution control systems. *The International Journal for Supercomputing Applications and High Performance Computing*, 10.2:223–235, 1996.