

Initial experiences with grid-based volume visualization of fluid flow simulations on PC clusters

David H. Porter, Paul R. Woodward, and Anusha Iyer

Laboratory for Computational Science & Engineering
University of Minnesota

ABSTRACT

Over the last 18 months, our team at the Laboratory for Computational Science & Engineering (LCSE) at the University of Minnesota has been moving our data analysis and visualization applications from small clusters of PCs within our lab to a Grid-based approach using multiple PC clusters with dynamically varying availability. Under support from an NSF CISE Research Resources grant, we have outfitted 52 Dell PCs in a student lab in our building that is operated by the University's Academic and Distributed Computing Services (ADCS) organization. This PC cluster supplements another PC cluster of 10 machines in our lab. As the students gradually leave this ADCS lab after 10 PM, the PCs are rebooted into an operating system image that sees the 400 GB disk subsystems we have installed on them and communicates with a central, 32-processor Unisys ES-7000 machine in our lab. The ES-7000 hosts databases that coordinate the work of these 52 PCs along with that of 10 additional Dell PCs in our lab that drive our PowerWall display. This equipment forms a local Grid that we coordinate to analyze and visualize data generated on remote clusters at NCSA.

The PCs of the student lab offer a 20 TB pool of disk storage for our simulation data as well as a large movie rendering capability with their Nvidia graphics engines. However, these machines do not become available to us in force until after about 1 AM. This fact has forced us to automate our visualization process to an unusual degree. It has also forced us to address problems of security and run error diagnosis that we could easily avoid in a more standard environment. In this paper we report our methods of addressing these challenges and describe the software tools that we have developed and made available for this purpose on our Web site, www.lcse.umn.edu.

We also report our experience in using this system to visualize 1.4 TB of vorticity volumetric data from a recent simulation of homogeneous, compressible turbulence with our PPM code. This code was run on the NSF TeraGrid cluster at NCSA, and the data was transported to our lab at 8 MB/sec over the Internet using the same ipRIO software that we developed to move this data around within our own environment. The movie visualizations generated on the ADCS PCs overnight are viewed on the LCSE 10-panel, 13 Mpixel PowerWall the next day. Smaller trial movies can be generated on the small PC cluster in our lab before submitting an overnight large movie request.

Keywords: parallel and distributed volume visualization; visualization of extremely large datasets.

I. INTRODUCTION

There are several other efforts in volume rendering of very large datasets, such as the Volumizer effort at SGI [Bhaniramka and Demange 2002] and the TRex effort at Los Alamos and Utah [Kniss et al. 2001], which rely principally on large, integrated, multipipe visualization servers. At Utah, Parker [2002] and DeMarle et al. [2003] have developed volume rendering techniques aimed at very large data sets that do not utilize GPU acceleration. The group at Caltech's CACR is working with specialized hardware to build a fast volume rendering pipeline for large datasets [Lombeyda et al. 2001, 2003]. Our approach using commodity PC clusters and GPU acceleration is distinguished from these other efforts in its focus on unattended operation using multiple clusters, both for the movie rendering and for the necessary data distribution. Our approach to the data distribution problem is distinguished from global file system [Elder et al. 2000] and Globus-based [Foster et al. 1997; Allcock et al. 2002, 2004] approaches by the special needs of our environment, as described below.

We begin by describing the structure of the data, since that is fundamental to the way we have chosen to process it.

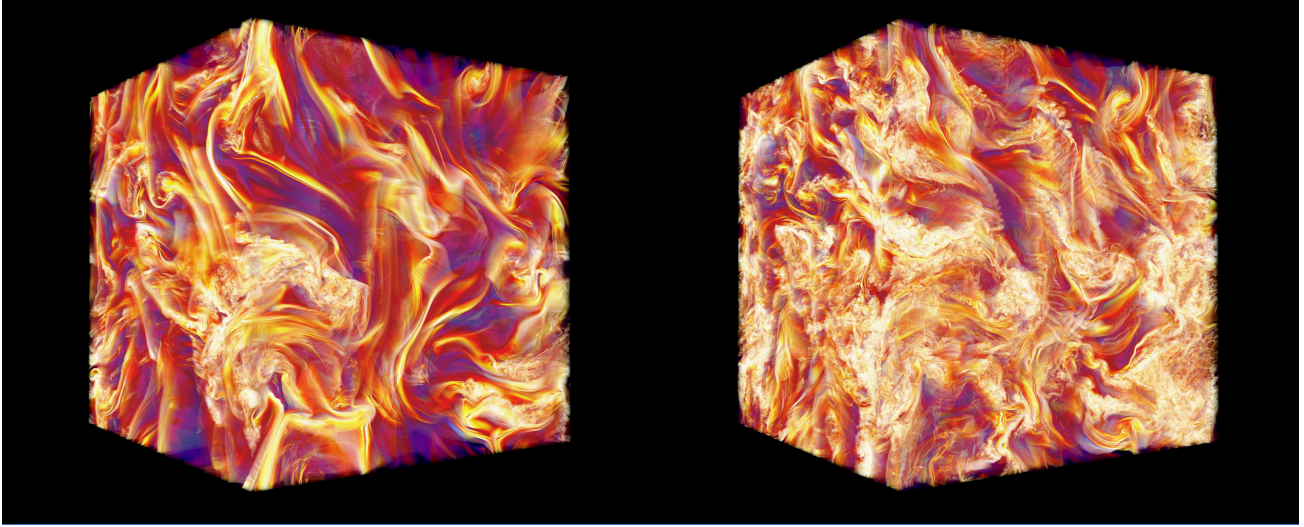


Figure. 1 A & B. Volume visualizations with HVR of the vorticity distribution in a PPM simulation of Mach 1 homogeneous turbulence as some regions near strong shear layers begin to become turbulent. In the second image, on the right, shear layers that were weaker in the previous image and were intensified by compression of the flow in the direction roughly perpendicular to these layers now go unstable.

II. BLOCK STRUCTURE OF A TYPICAL LARGE FLUID DYNAMICS SIMULATION DATASET

Large fluid dynamical simulations are almost exclusively performed on systems with large numbers of processors and distributed memories. It is therefore natural to archive data from a simulation in data blocks that each describe the fluid state at a particular time level in a simply connected 3-D subdomain of the flow. Each complete snap shot of the fluid state then consists of a collection of hundreds or thousands of separate disk files, and there are hundreds of such snap shots in a typical simulation dataset. We use uniform Cartesian meshes for our flow simulations, which is a luxury that is still both possible and practical for many flow problems of astrophysical research interest. In this paper, we will use as an illustration a 5 TB dataset from a recent simulation by our team of Mach 1 homogeneous turbulence [Woodward et al. 2004] using our PPM gas dynamics code [Woodward and Colella 1984; Colella and Woodward 1984; Woodward 1986; Edgar et al. 1999; Woodward 2005] on a grid of 2048^3 cells. This simulation was run on the TeraGrid cluster at NCSA using a number of dual-CPU machines that varied between 80 and 250 over the course of this 2-month run. We decomposed the problem domain into 1024 grid bricks, each with 128×256^2 cells. As the simulation ran on the cluster at NCSA its data products were streamed to the San Diego Supercomputer Center over the TeraGrid for archiving and a preprocessed subset of this data was streamed to our lab, the LCSE, at the University of Minnesota for analysis and visualization.

2.1. Scale of the Dataset

At relatively frequent intervals, we saved compressed representations of the complete fluid state on a coarsened grid and also a full resolution compressed representation of the magnitude of the fluid vorticity. Thus at each of 208 time levels we saved 1024 “blended dumps” and 1024 vorticity “bricks of bytes.” The blended dumps saved the averaged values of 35 state variables or products of state variables in 4^3 bricks of cells within the 128×256^2 subdomain, using 2 bytes to represent each blended value. Each such file was thus $2 \times 35 \times 32 \times 642$ bytes, or 8.75 MB. Each vorticity brick was 128×256^2 bytes, or 8 MB. In this particular run, we did not start saving the vorticity bricks until dump 68, so that we have only 140 of these 8 GB collections of files while we had 208 of the 8.75 GB collections. Ten times less frequently, we saved standard PPM “compressed dumps,” each consisting of 1024 files, each in turn representing the cell averages of the 5 fundamental fluid state variables in a single subdomain. Using 2 bytes per value, these files are each 80 MB, so that each such compressed dump is a collection of files totaling 80 GB.

Our dataset from this simulation amounts to 1.82 TB of blended dumps, 1.12 TB of vorticity bricks, and 1.6 TB of compressed dumps, for a total of 4.54 TB. This is not an unusual amount of data for a high resolution fluid simulation. Nevertheless, the 376,832 files in this dataset constitutes a significant data management problem. Before bringing the vorticity data to our lab, we simplified our data management task by combining the vorticity bricks in groups of 1024 files into just 140, very much larger hv-files (described below) before they left the NCSA cluster where they were generated. This reduced our overall data management problem to dealing with only 233,612 files, but increased the total size of the dataset to 4.85 TB. We moved these files to the San Diego Supercomputer Center's Storage Resource Broker (SRB), and we moved the hv-files to our lab in Minnesota using our own ipRIO software.

2.2. Structure of a Volume Rendering HV-file

The visualization software we have developed at the LCSE (see www.lcse.umn.edu/hvr) deals with files that we call hv-files. Each hv-file is made up of a hierarchically organized collection of bricks of bytes along with metadata. Each such file gives a complete description of the distribution of a single variable over the entire problem domain, using only one byte per value. For the turbulence dataset we are discussing, we chose 128^3 grid bricks for our hv-file representations. To avoid the generation of image rendering artifacts, these grid bricks overlap, and therefore the hv-file contains some left-over sections that are not grid cubes but instead rectangular solids of grid cells. A single hv-file contains a representation of the variable on the full resolution grid, also on a grid where averages on 2^3 bricks of cells are given, on a grid of averages over 4^3 bricks, etc., with the coarsest representation given in a single brick of 128^3 or less blended values. All these bricks comprising all these different resolution representations of the variable, together with some metadata, are collected into a single hv-file, which for our 2048^3 grid is then 10.2 GB in size. Volume renderings of 2 of these hv-files, made with our utility HVR (cf. www.lcse.umn.edu/hvr), are shown in Figure 1 at 2 time levels during the first half of the transition to fully developed turbulent flow.

2.3. Generating HV-files with A3D

We have written a data analysis utility, A3D, available on our Web site, that can generate an hv-file for any prescribed function of the variables in a compressed or blended dump. The input to A3D is therefore a collection of 1024 files, and a single hv-file is the output. A3D processes the compressed data chunk by chunk, which is well adapted to the requirements of a PC cluster, in which no cluster node contains more than 2 GB of memory. A3D allows the specification, in inverse Polish notation, of any function of any combination of the variables in the compressed dump. It supports operators that include, in addition to addition, multiplication, and division, complicated combinations of these representing, for example, x-, y-, or z-derivative, divergence, curl, and the Laplacian, as well as common mathematical functions such as abs, sqrt, log, exp, arcsinh, tanh, and various scaling operations. Certain frequently used combinations of operators, such as the computation of a scaled vorticity magnitude, are implemented in an optimized form. None of these computations does very much work on the data that is read in from disk. Therefore disk I/O is usually the performance-limiting factor. For this reason, within each of the 1024 files in a compressed or blended dump, the data is blocked by variable. Thus if we desire only the vorticity, we need read in only the 3 velocity components, which amount to 60% of the data in a full resolution compressed dump of 80 GB or only 9% of the data in a blended dump.

The representation of the spatial distribution of a variable by an hv-file is so compact that even on grids as fine as 2048^3 we have not yet seen any good reason to decompose hv-files into collections of separate files. Thus from hundreds of thousands of compressed dump or blended dump files we use A3D to generate only hundreds of hv-files for each variable selected for visualization. This represents a simplification in our data management challenge. However, we usually visualize several different variables from each simulation. Thus we still have thousands of files to manage, even if we can ultimately set aside our hundreds of thousands of compressed and blended dump files. For the particular dataset we are focusing on here, the vorticity files were generated by the PPM code as it ran, and collected into hv-files by a separate daemon process. We will see that dealing with these files effectively in a Grid-based visualization context is still a challenge.

III. PARALLEL MOVIE RENDERING.

Our hierarchical volume rendering utility is called HVR. It is available on our Web site at www.lcse.umn.edu/hvr along with a technical description of its detailed operation and a (still incomplete) user's

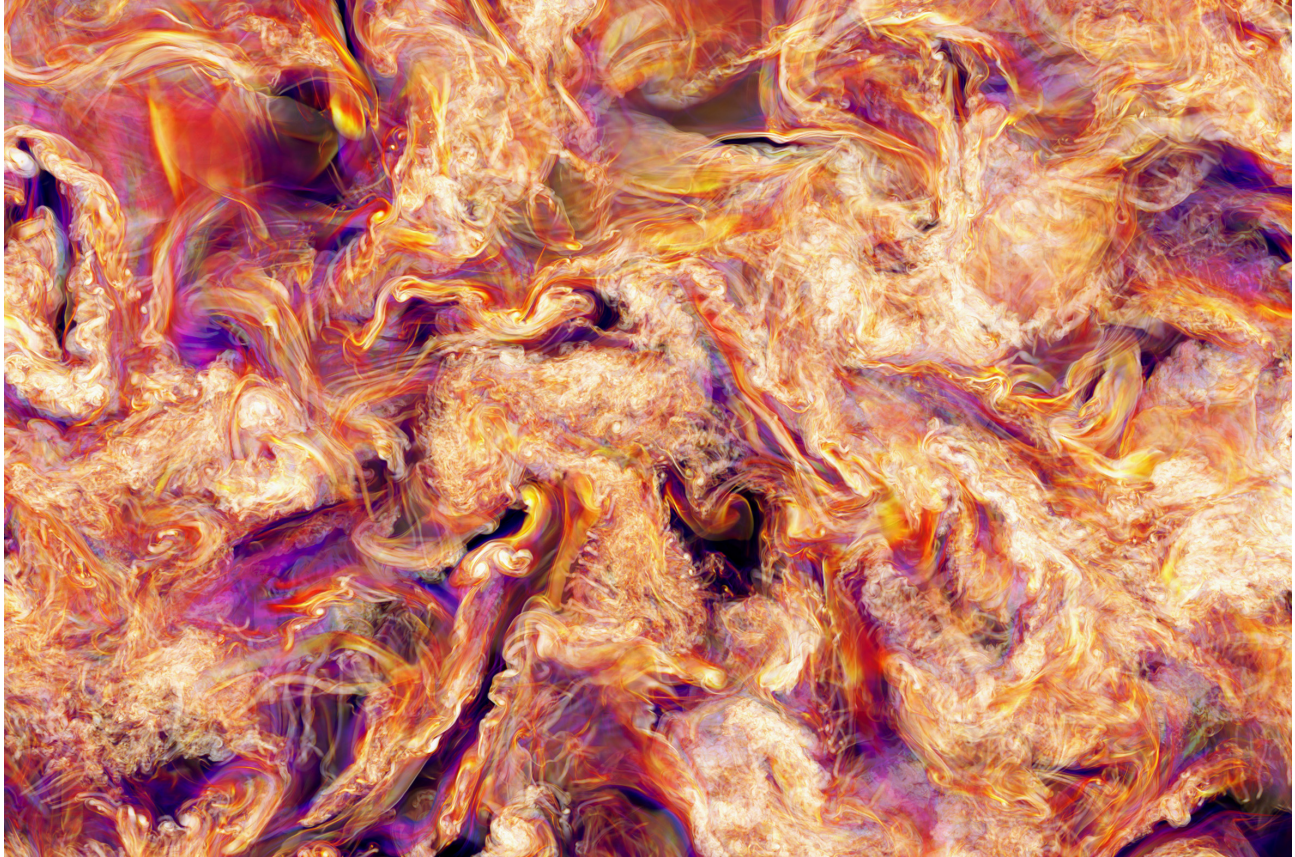


Figure 2. A volume rendering with HVR of a slice of the 2048^3 domain. The magnitude of vorticity is visualized here, with white showing the highest values, usually found in the cores of the many intertwined vortex tubes. Views like this one, from inside the flow, give us a close-up view of the vorticity dynamics when animated on our PowerWall.

guide. It renders an image by processing the bricks of bytes in the hv-file one by one, beginning at the back of the volume and working forward. The metadata in the hv-file is used to inform HVR of the extent of variation of the variable value within each brick of bytes, so that HVR can determine in each particular situation what resolution representation should be used to create the image. Our team at the LCSE has been rendering images in this fashion for over a decade; the new feature is the speed and the low cost of the operation on modern PC equipment. As has been true since the late 80s, this perspective volume rendering operation (we have never used orthographic projections) runs so well on specialized hardware that general purpose CPUs have no chance to beat it. For high resolution data, the compromises in image quality forced by this approach are extremely minor. Therefore we have chosen to accept these compromises in return for the huge acceleration of the rendering speed that the GPU hardware provides. For our very large datasets, this rendering process can easily become I/O limited. Nevertheless, on the inexpensive Dell PCs of the ADCS student lab, we have attached two internal 200 GB ATA-133 disks that, when striped, deliver 70 MB/sec sustained data streams. This is nearly but not quite fast enough to keep up with the graphics card's present maximum rendering speed. These PCs render 13 Mpixel PowerWall images of this 8.6 Gvoxel vorticity data in about 115 seconds, which indicates that they are I/O limited.

For many years our team has been dealing with data sets that are far too large to analyze or visualize interactively on any equipment we have been able to afford or have been allowed to access. We have therefore focused on movie generation and display. We use HVR to interactively designate a complex visual tour through our data as a sequence of key frames. In designating this movie path, we use reduced resolution hv-files that contain only the first few levels of voxel bricks. We have written a utility that extracts this coarsened information from a designated set of hv-files and creates a new, local set of very much smaller hv-files that can be interactively viewed on a single PC or laptop machine. Our HVR rendering utility can be informed of the locations on the network of the full-resolution hv-files, so

that if we desire to view a particular key frame at full resolution, the data can be streamed from that file over the network as the full-resolution rendering proceeds. HVR allows the user to generate a test movie from the coarsened data, which can be rapidly rendered even on a laptop machine for validation of the desired pacing as well as the effectiveness of the designated movie path through the data in bringing out particular features under study. This capability greatly mitigates the impact of the long delays that are involved in full resolution movie rendering for a high resolution display such as our lab's PowerWall. We have a complete set of the 1.4 TB vorticity data on the disks of our 10 Dell PCs that drive the LCSE PowerWall. We therefore usually specify our movies using this equipment, but some of the movies have been designed on laptop computers using reduced sets of the data stored on external FireWire drives.

At the LCSE we exploit the fact that rendering a high resolution movie is a lengthy process by implementing this rendering application in the simplest possible parallel fashion. Rather than have multiple PCs labor cooperatively on the rendering of each frame in the sequence, we let individual PCs render entire frames, even when those frames are composed of multiple image panels. This approach is not only simple, but it reduces potential contention on the network for the data needed to render any particular frame in the movie sequence. Not only is our data too big to fit into the memory of a single PC, it is usually also too big to fit onto the disk subsystem of a single PC. In principle, a PC could read the data it needs for rendering a movie frame over the network from the disks of another PC, however, over our Gigabit Ethernet network, with the network adapters and the ATA-133 disk adapters sharing the same 133 MB/sec PCI bus, we can read data from another PC at no more than 35 MB/sec, while we can read from local PC disks at twice this rate. We therefore have adopted the strategy of having PCs only render movie frames for which the necessary data is already located on their local disks. To make this strategy work, for movies that tend often to involve lingering, 800-frame inspections of the flow structures at a single time level, we must carefully lay out the data over our cluster of 52 PCs. The total disk capacity of this PC cluster is 20 TB, so it is generally possible to lay the data out in such a way that there are many copies of any given data file on different PCs. If we are willing to restrict our lingering viewing of particular time levels to, say, levels selected from a special, reduced set, such as every fifth hv-file in the time sequence, then this small set of special files can be replicated much more widely on the system. The data replication on the system not only allows multiple PCs to work on frames that require the same data files but it also allows a movie to be rendered in the student lab even as some of the PCs are being used by students. The lab is open 24 hours per day, and some fraction of the PCs might never be freed up for our use on a particular night.

3.1. Automation of the Movie Making Process

Our movie rendering process at the LCSE is coordinated by a globally accessible MySQL database, which is hosted on a Unisys ES-7000 in our machine room. From HVR, users submit movie description files, which are expanded into individual movie frame requests and entered into this database in order of receipt. When a given PC becomes available for movie frame rendering, its screen saver that implements this function locks and acquires the global information in the database in order to select a movie frame from this list that it will render. At present, if the PC does not have the data on its local disks for any of the requested movie frames, it will unlock the database and do nothing, returning to the database to try again one minute later. Otherwise, the PC will enter into the database its selection of a frame, with a time stamp, return the database information, and unlock the database. This process takes only a few milliseconds, so that it does not become a bottleneck with 52 PCs when movie frames each require from 2 to 200 seconds to render.

Although the 10 PCs in our lab that drive our PowerWall are available to us 24 hours a day, rendering a long PowerWall movie on this cluster alone could take days. By utilizing the additional PCs of the student lab as they become available, we can get this rendering time down to a single night. At intervals during the night, the ADCS lab operators boot PCs not used by students into our operating system image, which makes our disk subsystem visible and which locks out student access. One minute after reboot, the screen saver movie renderer wakes up and accesses our global database to select a movie frame to render. In this fashion we have been able to render a movie of 3200 frames from our 1.4 TB vorticity dataset described above at full 13 Mpixel PowerWall resolution in just about 2 hours. Thus in a given night, many thousands of movie frames can automatically be generated, even using our largest datasets and rendered for our highest resolution displays. This process is not interactive, but it is no longer tedious, since we have automated the process thoroughly. As each movie image panel is rendered, it is automatically written over the

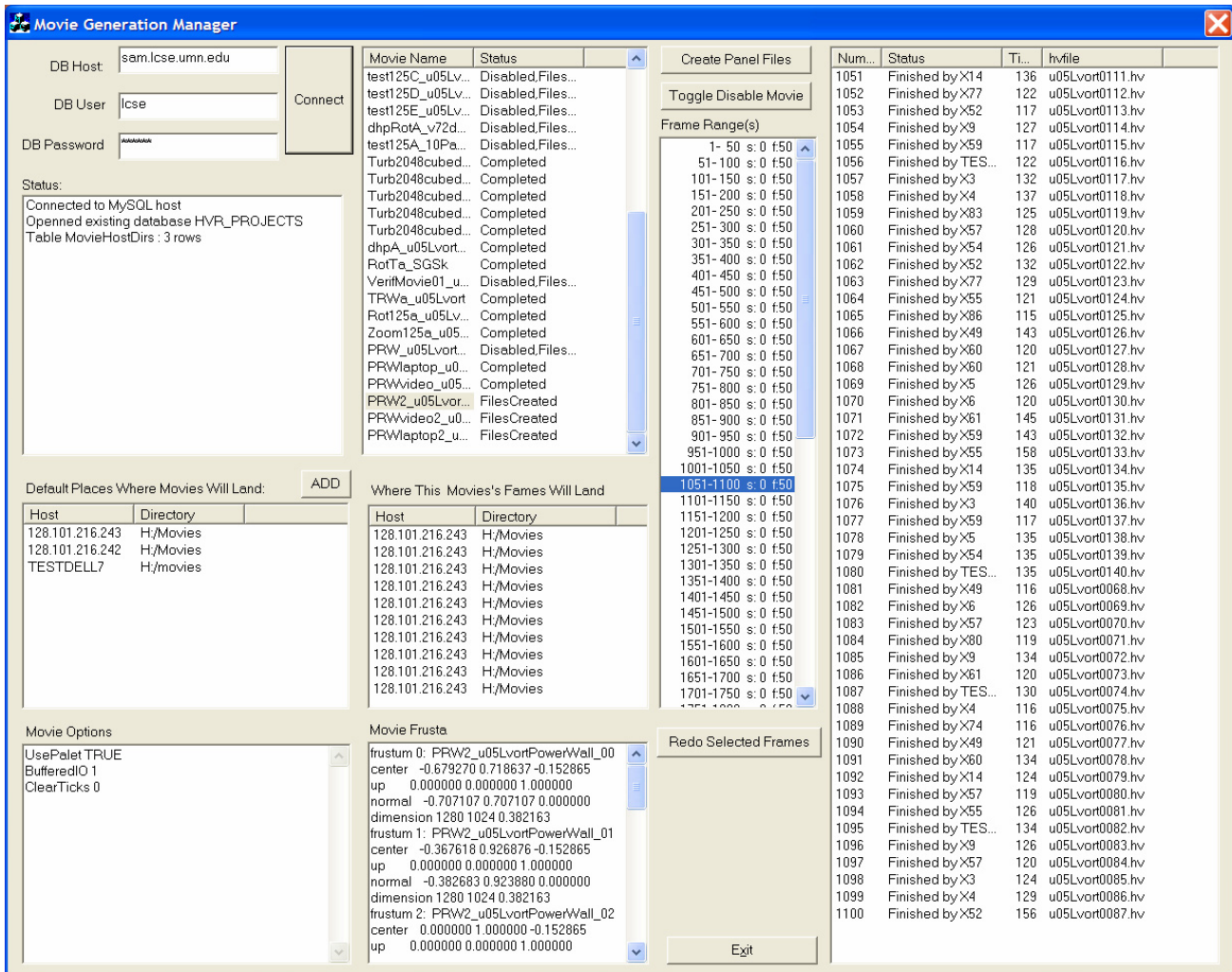


Figure 3. The Movie Generation Manager, MGM, allows the user to manage movie rendering jobs.

network to the disks of the PC that will later be used to display it. Thus a user can walk in the next morning and see the movie immediately.

The utility routine MGM provides a user with a view of the movie generation database. A typical view of this database is shown in Figure 3. A movie being generated for display on the PowerWall has been selected in the “Movie Name” window near the top center of the form. Ten image panels, each 1280×1024 pixels, will have movies generated, and these will later be synchronously displayed by the 10 PCs that drive the PowerWall display. The machine on which these 10 movies are being placed is listed in the window near the center of the form. There are 3277 frames in the requested movie, as is shown in the “Frame Range(s)” window on the form. The disk files for the 10 movies are created first, so that they will not be fragmented, and then the individual frames are written into these files as the image rendering proceeds. The status of any range of 50 frames can be viewed in the window at the right of the form. Here the frame number, status, rendering time (in seconds), and data file used are displayed. In the particular range shown it is clear that several machines have cooperated in generating the movie. The “redo frames” button allows selected frames to be redesignated as “new” so that they will be rendered by the next machine to access the database that has the necessary files on its disks. This function allows individual frames to be repaired in a movie in response to any of a number of possible error conditions.

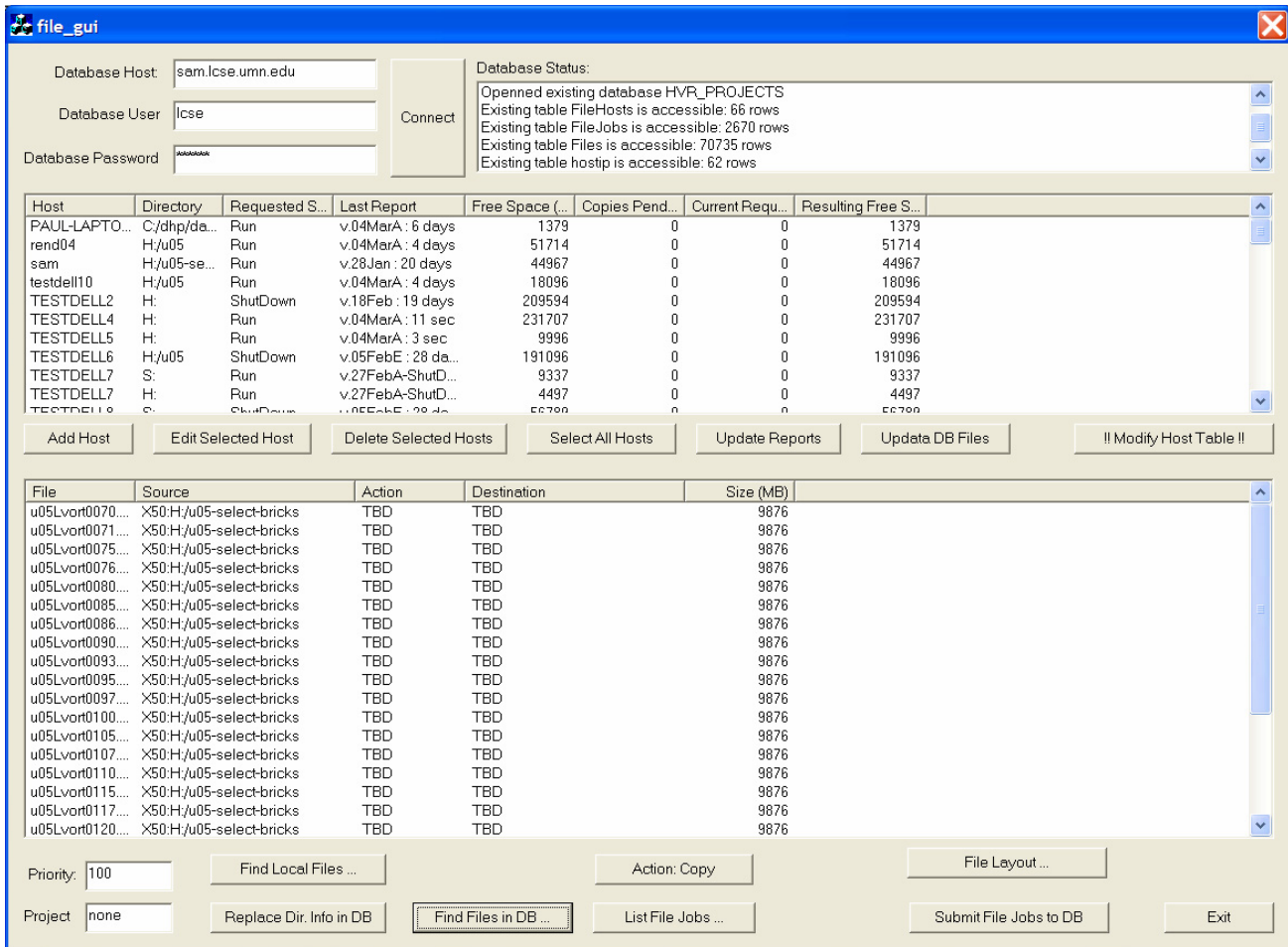


Figure 4A. The File_GUI user interface to the data distribution service database. In the upper window, machines and their drives or directories that are entered in the database appear, along with their present status. In the lower window, the user has caused all the files in machine X50's H:/u05-select-bricks directory beginning with "u05Lvort" and ending with ".hv" to be listed. Any or all of these can be selected, and a destination machine and directory selected, the "Action Copy" button and then the "Submit File Jobs to DB" button pressed, and the corresponding file copy requests will be entered into the database.

At present, each machine in the cluster, upon being booted into the LCSE rather than the ADCS operating system image, accesses the database and looks for the first movie frame designated as "new" for which this PC has the data needed to render the frame. Upon selecting such a frame, it marks the frame as "started," puts its machine name and start time into the database record for this frame, and unlocks the database. When there are no frames left to be assigned for a given movie, PCs accessing the database check for frames marked as "started" for which an inordinate time has elapsed without the frame being marked as "finished." Any such frames are then taken for re-rendering. This process makes the movie rendering fault tolerant. Machines for which problems occur can be reset or otherwise attended to by the operator of the ADCS lab, and thus brought cleanly back into the movie rendering process.

3.2. Staging the Data for Effective Movie Rendering

One aspect of the movie generation process that is very tedious and that we are now working to automate is the staging of the data. Consider our example of the 140 vorticity hv-files from the PPM turbulence simulation on a 2048³ grid. These 10.2 GB hv-files were brought to our lab in Minnesota over Internet-2 at 8 MB/sec using our own data transmission software, ipRIO (see www.lcse.umn.edu/shmod). Staging this data to Minnesota was difficult,

File	Source	Action	Destination	Size (MB)
u05Lvort0180...	X81.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0177...	X81.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0175...	X80.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0170...	X80.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0167...	X79.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0165...	X79.H/u05-select-bricks	copy:NEW	X87.H/u05-select-bricks	9876
u05Lvort0160...	X80.H/u05-select-bricks	copy:Started	X87.H/u05-select-bricks	9876
u05Lvort0157...	X80.H/u05-select-bricks	copy:FAILED	X87.H/u05-select-bricks	9876
u05Lvort0155...	X80.H/u05-select-bricks	copy:FAILED	X87.H/u05-select-bricks	9876
u05Lvort0150...	X80.H/u05-select-bricks	copy:FAILED	X87.H/u05-select-bricks	9876
u05Lvort0147...	X80.H/u05-select-bricks	copy:FAILED	X87.H/u05-select-bricks	9876
u05Lvort0145...	X80.H/u05-select-bricks	copy:FAILED	X87.H/u05-select-bricks	9876
u05Lvort0208...	X76.H/u05-select-bricks	copy:NEW	X86.H/u05-select-bricks	9876
u05Lvort0203...	X83.H/test	copy:NEW	X86.H/u05-select-bricks	9876
u05Lvort0200...	X82.H/u05-select-bricks	copy:NEW	X86.H/u05-select-bricks	9876
u05Lvort0194...	X83.H/test	copy:NEW	X86.H/u05-select-bricks	9876
u05Lvort0190...	X82.H/u05-select-bricks	copy:NEW	X86.H/u05-select-bricks	9876
u05Lvort0185...	X82.H/u05-select-bricks	copy:NEW	X86.H/u05-select-bricks	9876

Priority: 100 Find Local Files ... Action: Copy File Layout ...

Project: none Replace Dir. Info in DB Find Files in DB ... List File Jobs ... Submit File Jobs to DB Exit

Figure 4B. In the lower window of the File_GUI user interface to the data distribution service database, the user has listed all the file copy requests, with the most recent listed first. As the appropriate machines become available, copies marked as “NEW” are started preferentially. If no such copies are needed, then copies marked as “FAILED” are retried. Finally, copies marked as “Started” but that have not finished in more than 3 times the normal time interval for that size of file will be retried.

because we have not yet automated this process, and since the viability of the data path changed frequently due to reconfigurations beyond our control. This data however, was generated at NCSA over a 2 month period, so we had plenty of time to get it to Minnesota. However, once we had this vorticity data in our lab, mirrored on two pairs of PCs, we were faced with the problem of staging it onto the 52 Dell PCs of the ADCS student lab. Although 12 fibers run from our machine room into this lab, at present cabling problems at the ends limit this number to just 4. Nevertheless, only two of our PCs in the LCSE can at present be on the network of the ADCS student lab, because of network management and topology issues and because of security concerns, all of which we are in the process of addressing. The ADCS lab thus looks like a swimming pool for data that we must fill through just two garden hoses. This picture, however, is not correct, because we want to replicate the data once we get it into the ADCS lab. Moving all the data into the ADCS lab was not difficult, because we could fit it all on just 4 machines, and we set scripts going that could run all night. It is the proper distribution of the data over the ADCS PCs that is the most difficult step. This involves many hundreds of file copies, each lasting, on average, 8 to 15 minutes, on machines that may not be available until after 1 AM.

To address the data staging problem, we have built a data movement orchestration utility that allows users to specify extremely large and complex sequences of file movements that can be carried out automatically overnight using our iPrio service, coordinated by a global MySQL database. This utility, file_gui, is illustrated in Figure 4 above. It allows a user to view all of his or her files of interest on the entire cluster system, even when the cluster machines are not available. Extremely large and complex sequences of file moves (at present, we copy and then have a separate utility for the dangerous, but necessary file deletions) can be scripted at a single sitting from a machine anywhere on the network. Once these file copy requests are entered into the database, they will automatically happen as soon as the necessary resources become available. When the copies are completed, the file location database is automatically updated to reflect the change. At present, only lengths of the files are entered into the database, because interrupted copies is a frequent cause of error, and it can be detected by short file lengths. We are now implementing automatic check-summing, since this can be done at essentially zero additional cost. This will allow us to automate file deletions following provably successful file copies. Without this sort of utility, the tedium of file distribution is a high barrier to productivity on any large cluster system. In our case, where the cluster is never available during normal working hours, this sort of file movement utility is essential. Other efforts in a similar direction, by, for example, the Condor and Gryphyn teams, are known to us, but did not, in our view, meet our performance or network topology requirements. Our file_gui utility is simple, it provides much of the functionality of a personalized shared file system, and it is targeted at our specific needs, for which a great many simplifying assumptions apply.

Our file movement utility is based upon a remote I/O service, ipRIO, that we developed initially for use by the gas dynamics simulation code itself. This service makes possible use of disks at the network nodes of a cluster as a shared memory, and we distribute ipRIO along with our SHMOD (Shared Memory On Disk) framework for parallel computation (see www.lcse.umn.edu/shmod). The ipRIO server will upon request read a file on its local disks, and forward that to be written onto the local disks of the requesting node. Also, it can take forwarded data and write a file on its local disks. ipRIO is built upon TCP/IP, and it uses large frames to achieve up to 90 MB/sec sustained data transfers over Gigabit Ethernet on appropriate systems. For the Dell PCs of the ADCS student lab, where both the disk controllers and the Gigabit Ethernet cards share the same slow PCI bus, such data transfers average about 35 MB/sec. Multiple ipRIO data streams over the same Gigabit Ethernet link can utilize about 90 MB/sec of aggregate bandwidth, and a single ipRIO stream from NCSA, in Illinois, to the LCSE achieves about 8 MB/sec.

An unexpected advantage of ipRIO for us is its utilization of a single, specific port. This feature allows it to operate on machines that have firewalls that essentially eliminate external access. The ADCS lab personnel have levels of security concerns that are extreme, arising no doubt from the tremendous difficulty of keeping a thousand PCs operating reliably and free of virus infections with a large student user community. Their staffing levels assume a nearly impenetrable wall against intrusions from the Internet. ipRIO helped us to gain their confidence that their PCs could operate free of virus infection while in constant communication with machines in our lab. In essence, what we are trying to do with our file transfer service is functionally indistinguishable from hacking; we seek automated remote control of large numbers of machines. ipRIO has features that allow us to hack these machines for scientific benefit while at the same time locking out any real hackers.

It is important to point out one additional feature of our disk file management software. When a PC is rebooted each evening to bring it into our environment, the first thing it does is to pull from our Unisys ES-7000 machine a fresh copy of our software, and then it updates the listings of file system directories that were requested for inclusion in the file database. This feature keeps version control issues, which plagued us at first, manageable, and it helps to keep the state of the file location database up to date.

IV. DIRECTIONS FOR FUTURE WORK

We are incorporating into our file movement utility an automatic check-summing capability that will enable it to perform deletions as well as copies without threatening our confidence in the outcomes of very complex, scripted file redistributions. We are also adding a capability to take a prescribed initial set of files and locations, and to have this utility develop from its program a complex series of file transfers to achieve one of a number of generic file layout patterns. We would also like to integrate our various utilities together, so that they can operate in a pipelined fashion without human intervention, covering the entire simulation data analysis and visualization pipeline from the data generation by the fluid dynamics code to the viewing of the data by the user. The plan for such a design involves many of the same elements that we have implemented already in more restricted contexts, such as a set of globally accessible databases that both log and coordinate the activity of flocks of different, cooperating servers across a network. These services are proactive, in that they aggressively consult the databases searching for work to perform. The user can run one or more user interface applications that allow access to these databases and to the data products of the simulation as well as services, such as image rendering, that operate on these data products. Finally, we are working to eliminate latencies in our data analysis and visualization pipeline by more tight integration and more complete automation, so that we will be able to generate volume rendered images interactively from a dynamic stream of data from a simulation on a remote supercomputer or remote cluster.

ACKNOWLEDGEMENTS

We would like to thank the TeraGrid team and especially Rob Pennington's team at NCSA for their cooperation with the large PPM turbulence run, which was carried out under a friendly user period in the fall of 2003. Our work on data analysis and visualization has been supported by the NSF PACI program through NCSA (grant ACI-9619019, subaward 786) and by the Department of Energy through the Office of Science (DE-FG02-03ER25569) and the ASCI program. Our rendering cluster has been built under support from an NSF CISE Research Resources grant (CNS-0224424), drawing on our pre-existing systems and generous local support through the University of Minnesota's Minnesota Supercomputer Institute, Digital Technology Center, and Academic and Distributed Computing Service.

We would also like to acknowledge the generous donation by Unisys of a 32-processor ES-7000 which now is the core system in our lab.

REFERENCES

1. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A. L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnal, D., Tuecke, S. 2002. Data Management and Transfer in High Performance Computational Grid Environments, *Parallel Computing Journal*, **28**, 749-771.
2. Allcock, W., Bresnahan, J., Kettimuthu, R., Link, J. 2004. The Globus eXtensible Inpput/Output System (XIO): A Protocol Independent IO System for the Grid, Argonne Natl. Lab. Technical Report ANL/MCS-P1130-0204, Feb., 2004.
3. Anderson, S. E. 2003. SHMOD library. www.lcse.umn.edu/shmod.
4. Bhaniramka, P., and Demange, Y. 2002. OpenGL Volumizer: A Toolkit for High Quality Volume Rendering of Large Datasets, Proc. of the 2002 IEEE Symposium on Volume Visualization and Graphics.
5. Colella, P., and Woodward, P. R. 1984. The Piecewise-Parabolic Method (PPM) for Gas Dynamical Simulations, *J. Comput. Phys.* **54**, 174-201.
6. DeMarle, D. E., Parker, S., Hartner, M., Gribble, C., Hansen, C. 2003. Distributed Interactive Ray Tracing for Large Volume Visualization, Proc. IEEE Symposium on Parallel Visualization and Graphics, Seattle, Wa., Oct., 2003, pp. 87-94.
7. Edgar, B. K., Woodward, P. R., Anderson, S. E., Porter, D. H., and Dai, Wenlong, 1999. PPMLIB home page at <http://www.lcse.umn.edu/PPMLib>.
8. Elder, A., Ruwart, T. M., Allen, B. D., Bartow, A., Anderson, S. E., and Porter, D. H. 2000. The InTENSity PowerWall: A Case Study for a Shared File System Testing Framework, Proc. 17th IEEE Symposium on Mass Storage Systems / 8th NASA Goddard Conference on Mass Storage Systems and Technologies, March, 2000.
9. Foster, I., Kohr, D., Krishnaiyer, R., and Mogill, J. 1997. Remote I/O: Fast Access to Distant Storage, Proc. Workshop on I/O in Parallel and Distributed Systems (IOPADS), pp. 14-25.
10. Green, K., 2001. Going with the Flow, NCSA *Access Magazine*, Fall 2001, available at www.ncsa.uiuc.edu/News/Access/Stories/itaniumflow.
11. Kniss, J., McCormick, P., McPherson, A., Ahrens, J., Painter, J., Keahey, A., Hansen, C. 2001. Interactive Texture Based Volume Rendering for Large Datasets, *IEEE Computer Graphics & Applications* **21**, 52-61.
12. LCSE movie animations of the run reported here and similar SHMOD computations can be viewed at www.lcse.umn.edu/AVIs. NCSA MEAD Expedition (Modeling Environment for Atmospheric Discovery) home page, www.ncsa.uiuc.edu/expeditions/MEAD.
13. Lombeyda, S., and McCorquodale, J. 2003. The VolumePro Volume Rendering Cluster: A Vital Component of Parallel End-to-End Solution, Technical Report CACR-2003-198, Jan., 2003. Available at www.cacr.caltech.edu/Publications/techpubs.
14. Lombeyda, S., Moll, L., Shand, M., Breen, D., and Heirich, A. 2001. Scalable Interactive Volume Rendering Using Off-the-Shelf Components, Technical Report CACR-2001-189, available at www.cacr.caltech.edu/Publications/techpubs.
15. Parker, S. G. 2002. Interactive Ray Tracing on a Supercomputer, in *Practical Parallel Rendering*, eds. A Chalmers and E. Reinhard.
16. Sytine, I. V., Porter, D. H., Woodward, P. R., Hodson, S. H., and Winkler, K.-H. 2000. Convergence Tests for Piecewise-Parabolic Method and Navier-Stokes Solutions for Homogeneous Compressible Turbulence, *J. Computational Physics* **158**, 225-238.
17. Woodward, P. R., Anderson, S. E., Porter, D. H., and Iyer, A. 2004. Distributed Computing in the SHMOD Framework on the NSF TeraGrid, LCSE Technical Report, Feb., 2004, available at www.lcse.umn.edu/shmod.
18. Woodward, P. R., and Colella, P., 1984. The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks, *J. Comput. Phys.* **54**, 115-173.

19. Woodward, P. R., 1986. Numerical Methods for Astrophysicists, in *Astrophysical Radiation Hydrodynamics*, eds. K.-H. Winkler and M. L. Norman, Reidel, 1986, pp. 245-326.
20. Woodward, P. R., 1993. Interactive Scientific Visualization of Fluid Flow, *IEEE Computer*, **26**, Issue 10, 13-25.
21. Woodward, P. R., 1994. Superfine Grids for Turbulent Flows, *IEEE Computational Science & Engineering*, Vol. **1**, No. 4, pp. 4-5+cover illustration.
22. Woodward, P. R., 2005. The PPM Compressible Gas Dynamics Scheme, preprint available at www.lcse.umn.edu/PPMscheme.
23. Woodward, P. R., Porter, D. H., Sytine, I., Anderson, S. E., Mirin, A. A., Curtis, B. C., Cohen, R. H., Dannevik, W. P., Dimits, A. M., Eliason, D. E., Winkler, K.-H., and Hodson, S. W. 2001, Very High Resolution Simulations of Compressible, Turbulent Flows, in *Computational Fluid Dynamics*, Proc. of the 4th UNAM Supercomputing Conference, Mexico City, June, 2000, edited by E. Ramos, G. Cisneros, R. Fernández-Flores, A. Santillan-González, World Scientific (2001); available at www.lcse.umn.edu/mexico.