

DIVING DEEP: DATA-MANAGEMENT AND VISUALIZATION STRATEGIES FOR ADAPTIVE MESH REFINEMENT SIMULATIONS

The authors' cosmological applications illustrate problems and solutions in storing, handling, visualizing, virtually navigating, and remote-serving data produced by large-scale adaptive mesh refinement simulations.

Adaptive mesh refinement, developed by Marsha Berger and her colleagues in the 1980s for gas dynamical simulations,¹ is a type of multiscale algorithm that achieves high spatial resolution in localized regions of dynamic, multidimensional numerical simulations. Greg Bryan's excellent article² in the March/April 1999 issue of *Computing in Science & Engineering* describes our cosmological AMR algorithm and how we have applied it to star, galaxy, and galaxy cluster formation. Basically, the algorithm allows us to place very high resolution grids precisely where we need them—where stars and galaxies condense out of diffuse gas. In our applications, AMR allows us to achieve a local mesh refinement, relative to the global coarse grid, of more than a factor of 10^6 . Such resolution would be totally impossible to achieve with a global, uniform fine grid. Thus, AMR allows us to simulate multiscale phenomena that are out of reach with fixed grid methods.

The AMR algorithm accomplishes this by producing a deep, dynamic hierarchy of increas-

ingly refined grid patches. The data structures for storing AMR data are complex, hierarchical, dynamic, and in general quite large. For example, the simulation of an X-ray galaxy cluster shown in Figure 1 used a grid hierarchy seven levels deep containing over 300 grid patches.

Existing visualization, animation, and data-management tools developed for simple mesh data structures cannot handle AMR data sets. Consequently, in the past several years we have been working at the National Center for Supercomputing Applications to overcome this deficit. Here we describe our progress in four main areas: portable file formats, desktop visualization tools, virtual-reality navigation and animation techniques, and Web-based workbenches for handling and exploring AMR data. Although we have applied our work specifically to cosmology, we believe our solutions have broader applicability.

AMR data structures

Within an evolving AMR simulation, the data is organized in a *grid hierarchy data structure*—think of a tree of arbitrary structure and depth (see Figure 2). The AMR algorithm generates this tree recursively and adaptively. Every node and leaf of the tree is associated with a 3D grid patch (hereafter, simply a *grid*); these have various sizes, shapes, and spatial resolutions.

1521-9615/99/\$10.00 © 1999 IEEE

MICHAEL L. NORMAN, JOHN SHALF, STUART LEVY,
AND GREG DAUES

National Center for Supercomputing Applications

In our cosmological simulations, each grid stores arrays of data describing the physical state of the cosmic fluid (density, temperature, and so on), the gravitational potential, and a list of particle positions and velocities for stars and dark matter. The cell size Δx of the grid decreases with depth in the hierarchy as $1/R^{level}$, where R is an integer refinement factor (2 in our application) and $level$ is the level of grid in the hierarchy.

A given grid in the hierarchy overlies and more finely resolves a region in its parent grid, and it can also be the parent grid for a yet smaller, more refined child grid. Every grid, regardless of its level, contains a complete representation of the solution in the region it covers. That is, there are no holes in the parent grid where the child grids exist. Consequently, we can obtain an approximate representation of the global solution at any level of the hierarchy by compositing the solutions at or above that level.

The solutions on the grid hierarchy advance forward in time with individual time steps Δt that decrease proportionally to Δx , starting at the coarsest grid (the root grid), and proceeding to finer and finer grids. When the algorithm reaches the finest level, each grid at that level will advance several times until its time “catches up” with the time of its parent grid. The schedule of operations is similar to the W cycle of classic elliptic multigrid.

Although a tree of grids is a useful way to think about the grid hierarchy at an instant in time, over time the number of levels and the number and location of grids at a given level change. Generally, the deeper into the hierarchy we go, the more grids there are and the

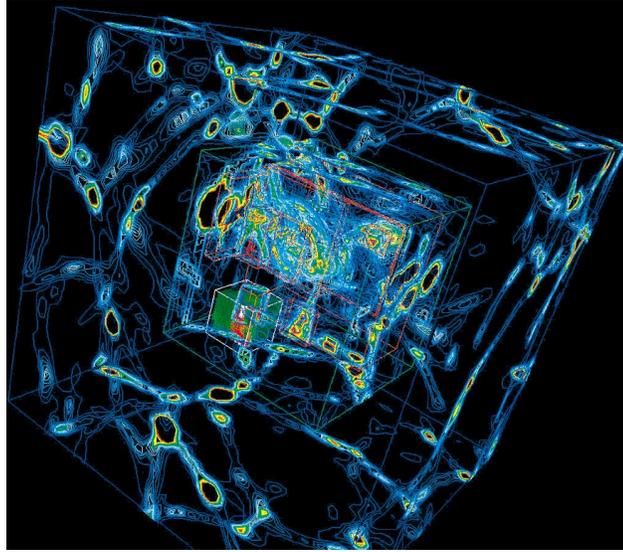


Figure 1. A deep adaptive mesh refinement (AMR) grid hierarchy. Contours of cosmic density in a simulation of an X-ray galaxy cluster are plotted on a grid hierarchy seven levels deep with several hundred individual grids.

more rapidly they change. This becomes important when devising file structures for writing AMR data to disk. The solution we have devised abandons any notion of the hierarchical relationship between the grids and instead just considers a collection of grids that changes over time. Each grid writes its own disk file containing attributes such as spatial domain, level of refinement, and persistence, as well as the field and particle data (for more detail, see the sidebar on FlexIO). In addition, the AMR application writes a single ASCII file to disk with all

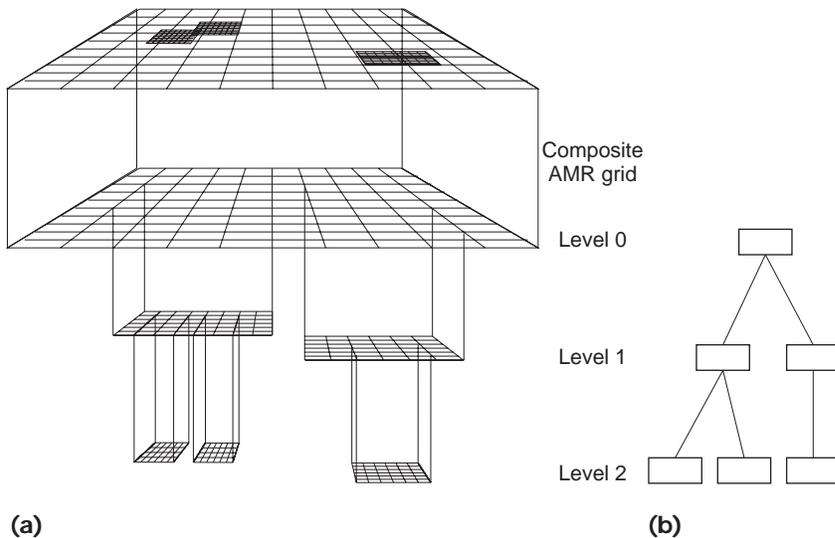


Figure 2. Schematic of an AMR grid hierarchy in 2D: (a) exploded view, showing the level structure, and (b) logical view of tree data structure.

FlexIO: A portable file format for AMR data

The sharing of data and visualization tools across research groups and applications requires common file formats. An AMR file format must be both platform- and application-independent. Platform independence means the file format cannot depend on any particular operating system or processor architecture. Application independence means that the file format must not depend on a particular set of simulation data structures and must be amenable to a wide variety of simulation implementations. Given the great diversity in code implementations and grid hierarchy structures developed so far by the AMR community, application independence has been much more difficult than platform independence. Above all, the file format must be simple enough that it can be easily modified to accommodate new file formats not considered in the initial implementation.

The HDF and NetCDF standards (see the Glossary of abbreviations in the main text) are both popular platform-independent file formats for storing binary simulation data sets. (See <http://hdf.ncsa.uiuc.edu/> for NCSA's HDF home page and www.unidata.ucar.edu/packages/netcdf for information on Unidata's NetCDF.) Because HDF can interoperate with NetCDF, HDF forms the basis for a platform-neutral file representation for AMR data. This is all wrapped in a more abstract API—FlexIO, developed by John Shalf—that is capable of more advanced file operations such as redirecting the I/O through a network socket with little effort on the part of the programmer (see <http://bach.ncsa.uiuc.edu/FlexIO/>). FlexIO is available with bindings for Fortran, C, and C++. It also allows translation to an ancillary file format called IEEEIO, which can be read and written using 100% pure Java or IDL (www.rsinc.com) rather than requiring Java be linked to native methods.

Both NetCDF and HDF store data in a self-describing manner. Each array of data stored in the file intrinsically includes its data type and dimensions. The user can extend the description of the data through attributes—additional named arrays of information that can store information like coordinate systems, time step, and calibration information. Table A and Table B list the standard attributes used in the FlexIO AMRwriter API. We add these NetCDF-style attributes to each

grid to fully describe its position in the hierarchy and thus uniquely identify it.

As you can see, there is a considerable amount of redundancy in these attributes. However, this ensures that the metadata will be in a form that is as convenient as possible for application designers, who tend to make different choices in expressing the same set of concepts. Also, all parameters related to refinement and grid placement are expressed both as integers and in floating-point form. The floating-point representations are necessary because they are most convenient for visualization systems, and they are what most simulations use internally to represent grid placement. However, the numerical precision of floating-point numbers becomes inadequate after about 10 levels of refinement. The only truly accurate representation of placement is integer coordinates. So integer and double-precision floating-point descriptions of refinement and grid placement are necessary to provide both maximum fidelity and convenience of representation.

Typical AMR simulation codes simply write one file per grid so that by the end of the simulation run you have a directory filled with thousands of files. The AMRwriter API enables you to combine all of the grids in the AMR hierarchy into a single file, simplifying data management. In fact, a sequence of time steps can be stored in a single file that takes into account the fact that grids on different levels are updated at different frequencies.

Finally, the FlexIO AMR API suite includes several sample implementations of readers for this file format. The simplest of the readers simply fills out a data structure with metadata information that it reads from the file and presents you with a list of information about every grid in the file. A more sophisticated reader allows you to select grids that are active at a particular time step so you can animate changes in the hierarchy through time. It minimizes data movement from disk automatically by loading only grids that have changed when the time step changes. It also allows you to select particular levels in the hierarchy (or remove intermediate levels that may clutter the scene). Finally, an enhanced version of this second reader automatically converts the AMR hierarchy into

the information required to assemble the complete grid hierarchy from the grid files.

AMR visualization strategies

AMR data structures have required a reevaluation of visualization system architecture. Traditionally, researchers have divided simulation data sets into three primary categories: structured grids, unstructured grids, and particles. AMR data doesn't really fit into any of these categories, leaving us with a gaping hole in our ability to un-

derstand the results of this emerging class of simulations. Consequently, we have invested considerable effort in making visualization of AMR data as sophisticated as that available for the traditional structured and unstructured grids. Of course, our first attempts at building the necessary infrastructure involved recasting the AMR data into a form that our traditional visualization systems could handle.

The most naïve way of approaching AMR visualization is to sample the entire hierarchy into a single structured grid. The primary advantages of

Table A. Attribute specification: Real-valued attributes.

Attribute	Description
origin	Floating-point origin of the data set
delta	Physical spacing between grid points in the data set
min_ext	Minimum coordinate of the data set in physical space
max_ext	Maximum coordinate of the data set in physical space
time	Current real-valued time that this grid represents

Table B. Attribute specification: Integer-valued attributes.

Attribute	Description
level	Level of this grid in the AMR hierarchy; levels are numbered from 0 to (n levels - 1)
timestep	Integer time step in the file with respect to the evolution of the finest-resolution grid in the hierarchy; in effect, the global simulation time step
level_timestep	Integer time step for the grid with respect to the evolution on its particular level
persistence	Number of time steps the grid persists on this particular level; in terms of the other parameters, the number of <i>timesteps</i> per <i>level_timestep</i>
spatial_refinement	Refinement in each Cartesian direction; a divisor that defines the refinement with respect to the top-level (0th level) grid
time_refinement	Refinement for the time step with respect to the top-level (0th level) grid step size
grid_placement_refinement	Attribute created independently from the <i>spatial_refinement</i> for staggered grids
iorigin	Integer origin of the grid placed in the coordinates of the <i>spatial_refinement</i> defined above

a mesh of hexahedral finite-element cells (removing all grid overlaps in the process). After selecting the desired simulation time step and levels of the hierarchy to show, this third reader gives you an array of points and a connectivity list for an array of hexahedral cells that connect those points. This operates within an AVS module as an example of how to integrate AMR data into existing commercial visualization systems.

In the near future, the HDF group, led by Mike Folk of NCSA, will update the file format to use HDF5, which provides a much richer and more efficient means of storing metadata and intergrid relationships (implied or otherwise). HDF5 also has full support for parallel I/O under HDF and MPI; this makes it much more suitable for wide-area distrib-

uted simulation codes. We are also closely watching the ASCII DMF efforts in enhancing HDF5 with a file storage API based on vector bundles.¹ This is a far more ambitious project than the standard described above, so we expect it will take some time to reach production. However, when it does arrive, it will provide a much more robust data model than we have been accustomed to under NetCDF and HDF4.x.

Reference

1. D.M. Butler and S. Bryson, "Vector-Bundle Classes Form Powerful Tool for Scientific Visualization," *Computers in Physics*, Vol. 6, No. 6, 1992, pp. 576-584.

this approach are that resampling is very easy to implement, and that the visualization-processing algorithms on structured grids tend to give much better performance than their unstructured-grid (finite-element) counterparts. Resampling can take place in full 3D (sampling into a uniform mesh) or by projection of all the data onto a 2D slice. This methodology is clearly not scalable, however, because in many cases an AMR simulation has such a huge range of length scales that there is insufficient memory in the graphics workstation and indeed in the supercomputer to rep-

resent the entire hierarchy. For example, sampling the results shown in Figure 1 to a uniform grid would require a $8,192^3$ grid—far larger than would fit into the memory of the largest supercomputer in the world. This method also results in a lot of redundant work on grid points that have been projected from the hierarchy's coarser grids into the visualization grid's finer mesh. We can compensate for the ratio of scale by specifying small regions of interest, but this leaves us with an incomplete view of the problem domain. Some early tools we implemented using AVS and IDL

Glossary of abbreviations

AMR	Adaptive mesh refinement
ASCI	Accelerated Strategic Computing Initiative
API	Application programming interface
AVS	Advanced Visualization System, a popular graphical visualization tool from Advanced Visual Systems, Maynard, Mass.
CAVE	Computer-Assisted Virtual Environment. Developed at the Electronic Visualization Laboratory of the University of Illinois, Chicago, CAVE is a room-sized virtual-reality environment where the floor, ceiling, and walls project a 3D stereoscopic view of a scene.
CGI	Common gateway interface
DMF	Data management facility
EVL	Electronic Visualization Laboratory at the University of Illinois, Chicago
HDF	Hierarchical data format, a platform-independent binary file format for storing simulation and image data developed by the NCSA HDF group.
HTTP	Hypertext transfer protocol
IDL	Interactive Data Language, a popular interpreted-language-based visualization tool created by Research Systems Inc. of Boulder, Colo.
MPI	Message-passing interface
NCSA	National Center for Supercomputing Applications
NetCDF	Network common data format, a platform-independent binary file format for storing simulation data sets developed by the Unidata group at the University Corporation for Atmospheric Research.
RMI	Remote method invocation
SPMD	Single-program, multiple-data
VE	Virtual environment

were very valuable in the development of the AMR codes (see Figure 3 and the Glossary of abbreviations). But these are reaching their limits of scalability as our AMR data sets ramp up in size and dynamic range.

To capitalize on AMR's multiresolution nature while still fitting it into existing visualization frameworks, an obvious choice is to convert the AMR hierarchy into an unstructured grid composed of hexahedral cells (cubes). This requires removal of overlapping cells in the hierarchy, an operation that can be implemented with $O(n)$ complexity (where n is the number of vertices in the entire hierarchy). This technique gives us graphics output that fully integrates the range of scale from the simulations. However, finite-element or unstructured grids are much less memory-efficient than the structured grids in the original AMR hierarchy because all connectivity in unstructured meshes must be explicit, whereas structured-mesh vertex connectivity is com-

pletely implicit. Furthermore, visualization algorithms on unstructured grids tend to be much slower than their structured-grid counterparts and much more difficult to parallelize efficiently. So, while translating AMR hierarchies into unstructured grids has greatly improved the visualization fidelity, we are still left with the same sort of performance and scalability problems we had when sampling into a uniform grid.

Unfortunately, these previous endeavors into AMR visualization treated AMR grid structures as second-class data types. Visualization systems and libraries are designed to deal only with data that they consider first-class data types such as particles, structured grids, and unstructured grids. All other data structures must be translated to fit into this narrow conception of the computational domain.

The treatment of AMR grids as first-class data structures for visualization computations leads to a variety of performance optimizations. For instance, visualization algorithms can capitalize on each grid's spatial location and data content to remove it from the computation using lightweight tests. A naïve isosurfacing algorithm must search every point in a structured mesh to determine where the surface should be placed. But an algorithm can cull a set of AMR grids based on their precomputed minimum and maximum data values to remove grids whose data content falls outside the range that would intersect with the level surface. An AMR ray tracer can trivially evaluate whether a grid in the hierarchy would resolve to subpixel resolution and thereby choose whether to descend any farther down the hierarchy as it makes progress through the data. A plane slicer need only consider grids that it would intersect. These are just examples of the most obvious set of applicable optimizations. We have barely begun to explore the rich set of possibilities of a grid structure that expresses data locality and level of detail natively as AMR does.

Opportunities for parallelism

As mentioned earlier, when we scale up the size of AMR simulations, we still face scalability problems. Most visualization algorithms, libraries, and systems are inherently serial because that was sufficient back in the days of vector computing. However, to create visualization tools that can interactively explore the terascale data sets produced on today's massively parallel systems, we must parallelize the visualization computations. That is our only means of fully

exploiting the performance of this new crop of supercomputers. Therefore, from the very start, we have considered opportunities for parallelizing AMR visualization so that it becomes an integral standard as these new libraries develop.

Opportunities for visualization algorithm parallelism abound. There is plenty of research into parallelizing visualization computations on structured and unstructured grids, and we can apply that research to AMR visualization algorithms. For structured grids (often called unigrids), a typical parallelization scheme is a regular domain decomposition of the grid into smaller subcubes of data that processors can operate on in a straightforward data-parallel (SPMD) manner. For unstructured grids, domain decomposition is much more difficult; researchers have proposed a variety of heuristic algorithms to address the problem.³

AMR data offers many more opportunities for parallel execution in the visualization pipeline. We can, for example, partition individual grids for data-parallel computations (treating each as a unigrid computation); however, such domain decomposition incurs a large cost because it requires a sizable communication volume between processors to distribute the grid data. Because the cost of distributing the data often exceeds the benefits of parallelism, this technique has limited utility. On shared-memory architectures like the Origin 2000, we can use an even lighter-weight partitioning, such as handing out grids to different processors in a worker-slaves arrangement. When each slave thread finishes computing on a particular grid, it can request another grid to compute from the master processor. This method raises the same issues as domain decomposition, but it eliminates the costs of copying the data across processors.

Level of detail

Perhaps the ultimate way to improve the user's ability to interact with visualization applications is to reduce or stagger the amount of data that must be computed at any given time. Normally, the stages of the visualization pipeline execute sequentially, forcing the user to wait until the entire pipeline has executed before the image on the screen updates. Even if each stage executes in parallel, the wait can be considerable. With AMR data, we can extend the parallelism in the pipeline direction. Each stage can operate simultaneously to pass the grids through the pipeline in a streaming manner. So the root grid of the hierarchy can pass through to the viewer

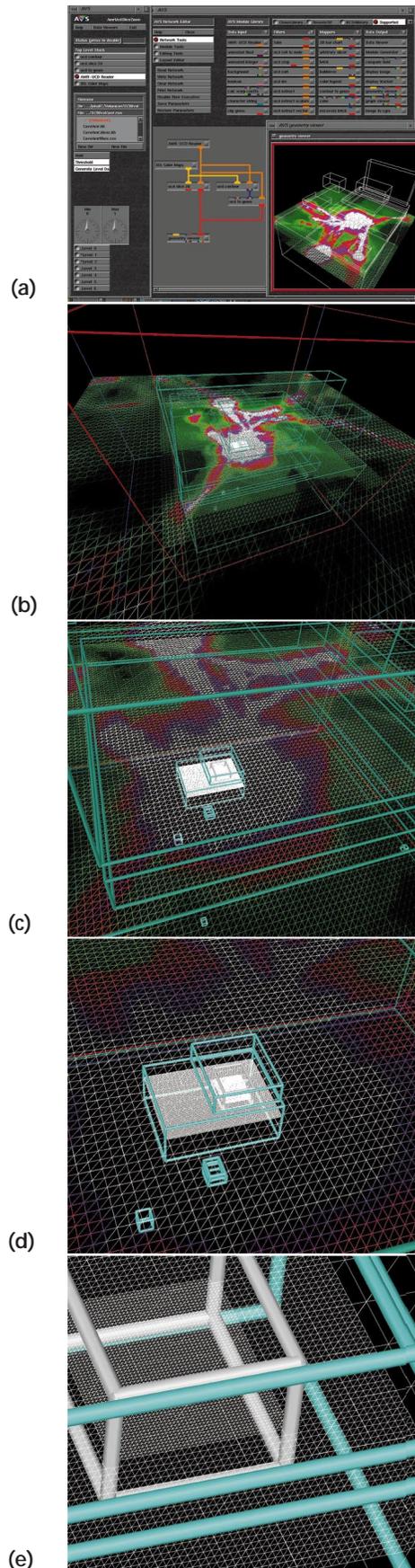


Figure 3. Visualizing AMR data using commercial software: (a) screen shot of AVS pipeline, which automatically converts an AMR grid data structure into an unstructured mesh of hexahedral cells; (b–e) zoom sequence showing the depth of the grid hierarchy.

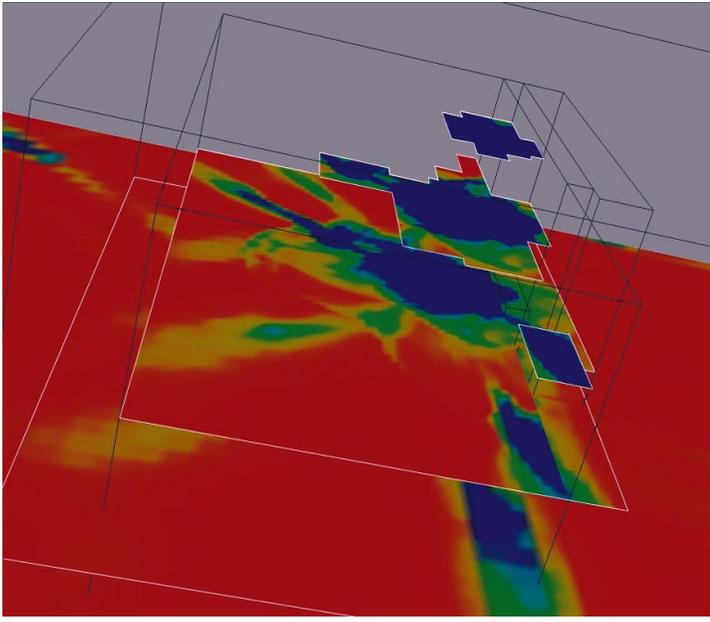


Figure 4. Visualizing AMR data using custom software based on the Visualization Toolkit. Here, the gas density on a slice through the center of a galaxy cluster is colorized (red = low, blue = high). For clarity, “slicelets” are offset from one another by a distance proportional to the level of the grid hierarchy from which the data is obtained. By treating the AMR hierarchical data structure as a first-class data type, VTK makes possible many optimizations that speed up visualization performance.

very rapidly, and over time the higher-detail subgrids make their way through the pipeline to gradually improve the quality of the scene. Using this streaming-AMR-grids paradigm gives the visualization system a natural continuum between interactivity and fidelity.

We can modify the order in which these levels of detail fill in simply by sorting the list of grids at the beginning of the pipeline to fit a particular sorting criteria. For instance, the simplest sort is to fill in the grids by their level in ascending order: coarsest to finest grids on a level-by-level basis. But we can change the sorting algorithm to fill in detail near a point of interest. Alternatively, we can sort the grids so that they fill in preferentially along the line of sight of the current camera view of the scene. Grid streaming opens up a robust set of methods for improving interactive performance and optimizing the method of filling in detail to fit a wide range of requirements.

Desktop visualization tools and workbenches

Back in the 1980s, the supercomputing commu-

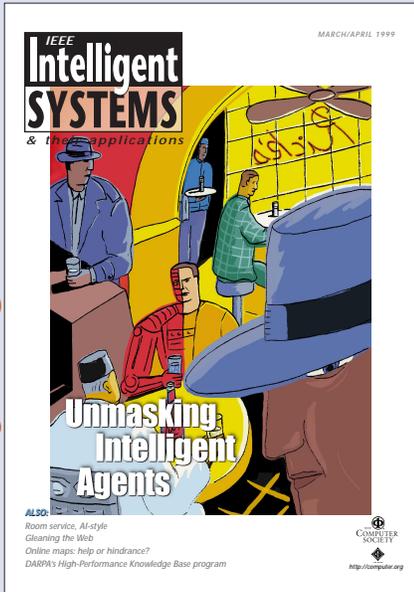
nity began to realize that 3D visualization tools were a basic necessity for understanding the output of supercomputer simulation codes. The Laboratory for Computational Astrophysics⁴ released a tool called 4D2 to provide a free and publicly accessible means of visualizing unigrid data coming from its ZEUS-3D code. This proved a great success among LCA’s community of users. LCA is now developing a portable replacement for 4D2 that will provide a simple-to-use visualization package that handles AMR data in addition to the original complement of unigrid data visualization features.

Our current implementations for native AMR computations are built around the Visualization Toolkit, an extensible, freely available, open-source set of visualization libraries. (See www.kitware.com/vtk.html.) VTK gives us a set of tools that we can modify at a very low level to treat AMR data as a first-class (native) data type. This lets us cast algorithms in a form that takes maximal advantage of AMR’s regularity and multi-scale nature. We anticipate that this will greatly improve the performance and flexibility of visualization systems for AMR data. By building onto VTK, we’re making these techniques widely available; other research groups can incorporate them into their own visualization tools, thus extending them far beyond our initial efforts. Figure 4 is a screen shot of the 4D2-VTK tool with an AMR slicer in operation.

The NCSA Computational Cosmology Observatory

Desktop tools have their limits, however. One of the biggest problems with general-purpose visualization tools is that they give users too many choices. It may seem hard to believe that such flexibility is not necessarily a good thing, but many full-featured visualization tools bewilder scientists with a dizzying array of widgets and menu choices, many of which are irrelevant to the problems they want to solve.

The “workbench” paradigm restricts a tool’s flexibility without making it thoroughly useless. The tool’s designers can optimize the workbench user interface for a particular audience and class of problems, emphasizing visualization paradigms and statistical metrics specific to the targeted research community. This customization can even include details such as the specific nomenclature for units of measurement rather than the more generic terms you would find in a general-purpose tool. Furthermore, a good workbench operates as a gateway to accessing



Special Focuses

Intelligent Information Retrieval...

Intelligent information retrieval—the problem of delivering truly relevant documents matching user needs—has become increasingly central in recent years. This special issue will present creative approaches to solving intelligent information retrieval problems using AI or ML techniques.

...and Constraints

A wide variety of problems can be naturally modeled as constraint-satisfaction and optimization problems, including scheduling, planning, resource allocation, routing, design, configuration, and diagnosis applications in the engineering, manufacturing, and service sectors. The special issue will provide insight into the increasing popularity of constraint technology.

IEEE Intelligent Systems

AI News You Can Use

data archives and running simulation codes. This lets scientists take advantage of sophisticated research codes without themselves being expert computer programmers. Designers of these tools usually implement workbench interfaces as Web tools to maximize flexibility.

The NCSA Cosmology Applications Team—in collaboration with cosmologists at the University of Missouri, the Massachusetts Institute of Technology, and New Mexico State University—is performing roughly 100 extremely high-resolution AMR simulations of X-ray galaxy clusters and placing the results in an online archive called the NCSA Computational Cosmology Observatory. (See <http://sca.ncsa.uiuc.edu>.) Theorists will find this archive useful because they can use the statistical properties of the simulated cluster sample to constrain allowable cosmological models. Observers will be able to conduct simulated observations of the X-ray clusters and their environs to help plan their observing campaigns. Our goal is to make this archive available to the scientific community while shielding users from the size and complexity of the raw AMR data. We also want to give users a variety of data manipulation, export, visualization, and analysis tools.

To accomplish these goals, we developed the Observatory as a workbench-style system that

lets users interact with archived simulation data over the Web. This environment supports data retrieval across several machines at the NCSA and between those machines and the local client systems. The Observatory further supports dynamic and interactive multidimensional visualization and basic analysis tools to extract specified physical attributes of the cluster systems.

Figure 5 shows a diagram of the Observatory. The user begins by selecting a cluster from a catalog. The server then retrieves the raw AMR data from NCSA's mass storage system. Once the data is on the Observatory Web server, the user extracts from the AMR files particle data and field data sampled to a uniform grid of user-specified size and spatial resolution. At this point, the user may export the extracted data as HDF files for local analysis or use the suite of analysis tools provided as a part of the Observatory.

As we considered what type of visualization software to provide for clients' local machines, our critical requirements were that the software be platform-independent and thin (small and quick to download). These characteristics are especially important for 3D visualizations. We have successfully addressed these concerns in two ways. First, the Observatory renders 3D objects on the server side. Client applets down-

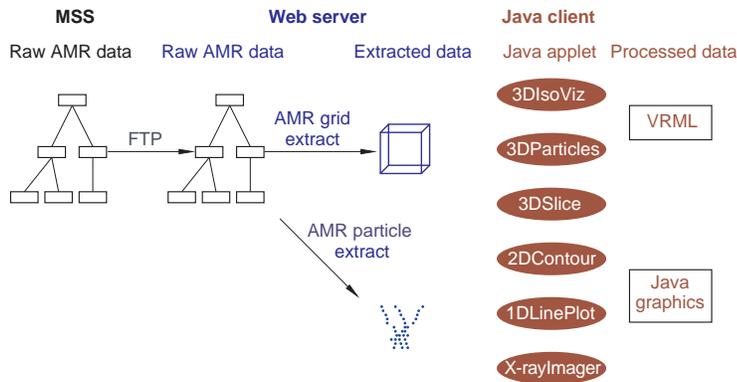


Figure 5. NCSA Computational Cosmology Observatory.

loaded from the site let the user specify which cluster to visualize, which field quantities to download, and the type of visualization, such as an isosurface. The applet then connects back to a CGI script via the HTTP protocol, which allows the user's applet to post its input to the CGI script. The script then takes the user's preferences and launches a server-side VTK-based rendering program. Using the server for rendering achieves the goal of having a thin client on the user's local machine.

Second, although visualizations created by VTK are not platform-independent, VTK does provide classes to convert the visualizations into a VRML 2.0 format and then write them to a file. However, instead of writing to a file, our CGI scripts write the contents to the standard output, and the server sends them back to the applet. The applet then simply opens a new browser window and specifies the content type (VRML), and the browser loads the appropriate VRML-viewer plug-in. Because browsers running under many different operating systems can display VRML-based graphical representations, this gives us a platform-independent way of displaying complex 3D visualizations. In short, the site leverages widely and freely available technology to deliver high-quality 3D visualizations to a variety of platforms without overwhelming the processing power and storage capacity of a client's local machine.

We also meet the requirement of lightweight client visualization software with Observatory site tools that implement lower-dimensional visualizations through Java graphics. Specifically, the client applet allows the user to launch CGI scripts that sample subsets of large 3D HDF files. The server retrieves these large files from the archive storage, and they remain on the server side. Only the smaller subsets of the data are

transferred over the network for the client software to process.

One capability of the client applet is the construction of 2D color contour plots of the sampled data. Figure 6 is a screen shot of a working session of the Observatory implementing these plots. In keeping with the consideration of processing speed and memory on the client side, the client applet allows users to control the size and number of images generated. We are also developing archive tools for other forms of lower-dimensional visualizations—such as 2D projections of 3D data and 1D line plots.

Future developments of the Observatory site will focus on implementing remote method invocation (RMI) to establish a more persistent and unified computing environment, allowing more finely controlled access of objects on the server. We will also add links to other electronic literature and develop software to interact with existing digital libraries of observed data. This will mean adding support for more data types and formats, as well as constructing additional analysis tools to allow direct comparison with observed data.

Multiscale virtual navigation and animation

Virtual environments that provide dynamic views covering the full human visual field promise insights not otherwise easily achieved into the rich variety of spatial and temporal structures arising from AMR calculations (see Figure 7). However, given the bulk of AMR data, creating graphical realizations at acceptable frame rates is a major problem even for high-end graphics systems. Virtual environments are more demanding in this respect than desktop systems: Although desktop users might tolerate a few seconds' delay while an image is computed, VE users lose their sense of immersion and virtual user interfaces become unusable if frame rates fall below a few frames per second.

It becomes essential, in the exploration process, for the VE user to focus attention on subsets of the data. Fortunately, the AMR paradigm offers natural ways to do this. The system can cheaply present data from the coarsest grids. Furthermore, the very distribution of the AMR grids indicates active areas in the computation. In a cosmological simulation, viewing animated grid-bounding boxes alone can reveal locations

of filaments and forming galaxy clusters.

From such cues, the user could indicate a volume of interest, perhaps by sweeping it out directly with a gesture. Sizes of interesting regions in AMR data might well span several orders of magnitude as exploration proceeds, but they could still be naturally described by the virtual users, who would be able to adjust their scale at will relative to the data. The visualization system could then concentrate its effort on, say, the coarsest few AMR grids spanning that region, extending outward (for context) and inward (for better resolution) as computation and rendering time permit. Applying the streaming-AMR-grid schemes described earlier will be crucial here. Much experimentation will be necessary to determine how the focusing process should work and how to design the balance between user specification and automatic adaptation.

Focusing attention, of course, isn't merely a way to accommodate the limitations of available graphics systems; it will be important also in extricating relevant features from what can easily become a visually cluttered mass of detail. Maximal realism is not always the best way to display scientific data!

Still, there will be a need for views of higher quality than those that can be generated at interactive rates. It seems promising to build a hybrid system that would decouple part of the rendering from the high-frame-rate virtual environment: The user would plant a virtual camera in space. A separate system (possibly a remote supercomputer) would render clearer views of the camera's scene at leisure and display the resulting images within (or alongside) the virtual environment. Meanwhile, the user would continue to interact with the virtual world.

Design of suitable user interfaces is another major issue for virtual AMR. Describing just what should be shown, and when and how, will call for plenty of knobs to tweak. While immersive VEs make direct spatial interaction easy, they lack good counterparts to the keyboard and pointer that make the familiar desktop widgets so efficient to use.

Virtual AMR in the Virtual Director

We are using the NCSA Virtual Director as a host for developing virtual AMR visualization tools (<http://viridir.ncsa.uiuc.edu/viridir/>). The Virtual Director is a CAVE-based software framework that wraps around visualization applications, supporting navigation through space and time, virtual camera choreography and recording, and networked collaboration in shared virtual spaces.⁵ Its

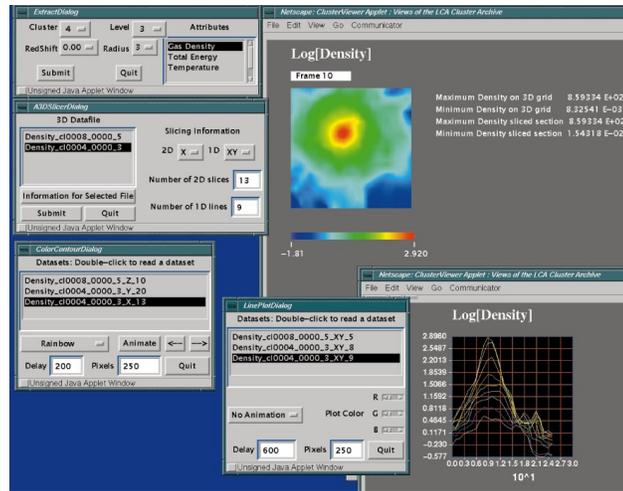


Figure 6. Screen shot of a working session on the Cluster Archive site illustrating Java-based 2D visualization. The components of the Java applet allow the user to specify parameters for data retrieval, sampling of the data into lower-dimensional subsets, and 2D color contour visualization with multiple color maps and zooming features. Here, the user sees a plot of the gas density for a slice through the core of a typical cluster.



Figure 7. The authors explore AMR data from a cosmological simulation of primordial star formation using the NCSA Virtual Director. They are standing in front of an ImmersaDesk—a virtual environment akin to a drafting table invented at the Electronic Visualization Laboratory of the University of Illinois, Chicago.

purpose is to provide simple creation of animations—either by automating a series of virtual-reality screen snapshots while driving the virtual camera along user-designed animation paths, or by exporting the path for use by external rendering software. Figure 8 illustrates the use of camera path for multiscale exploration. We have programmed the camera to adjust scale as it approaches the forming protostar.

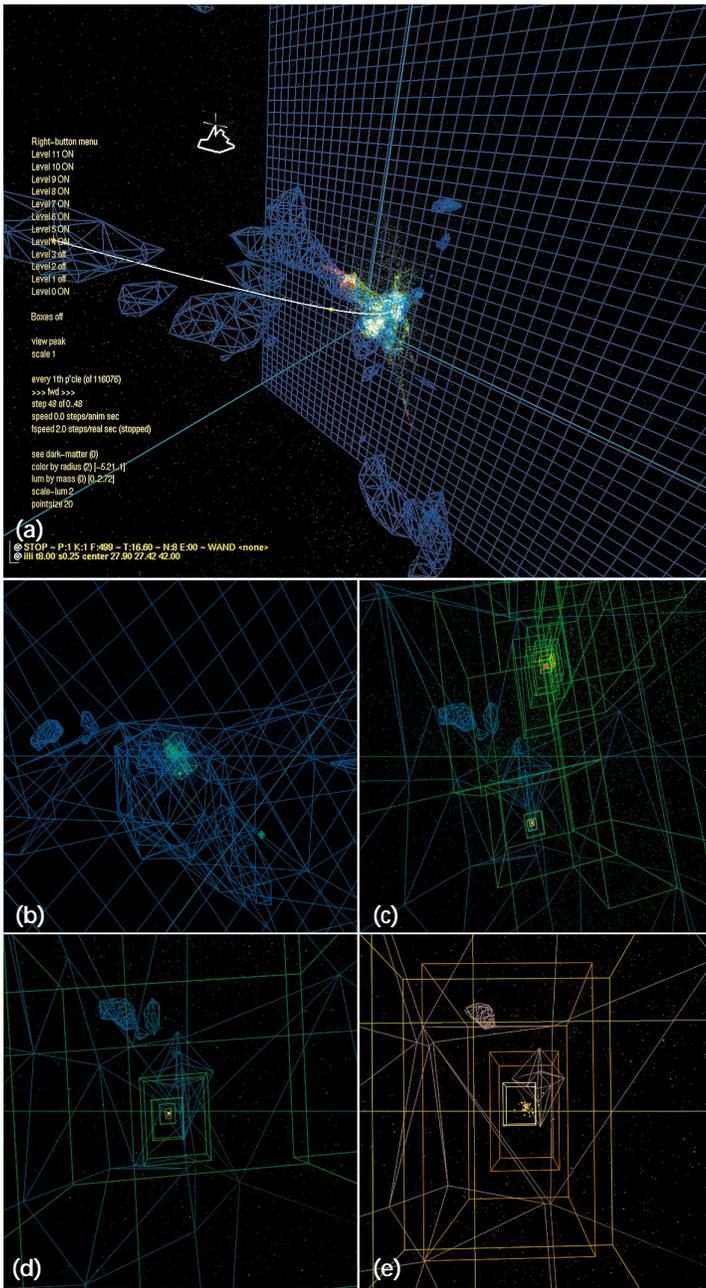


Figure 8. Diving deep using the NCSA Virtual Director. With wand and voice commands, the user interactively defines a camera path that passes near a forming protostar. A range of scales in excess of in excess 100,000 is resolved with a 13-level AMR grid hierarchy. (a) overview and (b–e) camera-eye view as it traverses the levels. Wireframe isosurfaces and colored points show the distribution of gas density in the neonatal cloud.

Users invoke many of the Virtual Director’s functions through a combination of 3D gestures and text-based commands. Using text makes it convenient to express commands using voice, and indeed users often control the Virtual Director using a commodity PC-based voice recognition system—a partial solution to the user-in-

terface conundrum. Likewise, it is easy to send commands as network messages, a facility that proves useful in collaboration sessions.

The power and potential of AMR to solve multiscale problems in computational astrophysics and cosmology excite us. We view AMR as continuing a trend toward simulations of realistic complexity begun in the 1980s with the emergence of supercomputers and accelerated in the 1990s with the advent of massive parallelism. However, even the most powerful parallel computers cannot accommodate uniform grids larger than about $1,024^3$, or a billion cells. With adaptive grids, we have been able to achieve resolutions in excess of 10^6 locally with far fewer cells. Thus, AMR can be viewed as an algorithm for reducing the memory and CPU requirements of simulations for a specified range of length and time scales, or enabling simulations not otherwise possible.

The physical sciences host an abundance of multiscale phenomena that can benefit from the application of AMR. Examples include atmospheric science, fluid dynamics, combustion, materials science, and so forth. Nonetheless, AMR has not been widely applied because of its complexity and the lack of available tools. The development and dissemination of AMR libraries and frameworks⁶ in recent years has begun to reverse this situation. We hope that our work on data formats and visualization strategies will be of use to AMR adopters in other disciplines. 

Acknowledgments

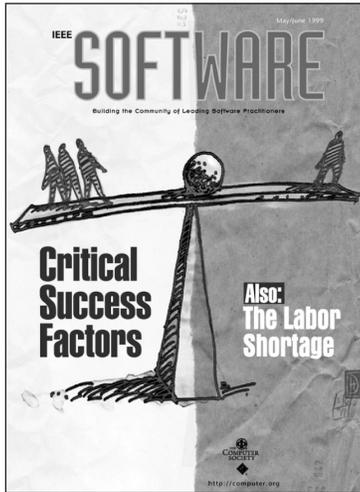
We thank Donna Cox, Bob Patterson, and Matt Hall of the Virtual Director team for allowing us to show some of their unpublished work. We also thank Galina Pushkareva and Brad Miksa for permission to use Figure 4. This work is partially supported by NASA grants NAG5-7404, NAGW-3152, and NAG5-3923.

References

1. M.J. Berger and P. Collela, “Local Adaptive Mesh Refinement for Shock Hydrodynamics,” *J. Computational Physics*, Vol. 82, No. 1, May 1989, pp. 64–84.
2. G.L. Bryan, “Fluids in the Universe: Adaptive Mesh Refinement in Cosmology,” *Computing in Science & Engineering*, Vol. 1, No. 2, Mar./Apr. 1999, pp. 46–53.
3. V.E. Taylor et al., “A Decomposition Method for Efficient Use of Distributed Supercomputers for Finite Element Applications,” *Proc. Int’l Conf. Application-Specific Systems, Architectures, and Processors*, IEEE Computer Society Press, Los Alamitos, Calif., 1996, pp. 12–24.
4. D. Song and M.L. Norman, “4D2 User Guide,” Document TR016, National Center for Supercomputing Applications, Urbana, Ill., http://lca.ncsa.uiuc.edu/lca_intro_4d2.html.

Get a copy of the next issue on Software Certification for \$10 at membership@computer.org

BUILD THE Community OF Leading Software Developers!



Keep track of advances in the software development industry! Access how-to's, tutorials, and experience reports direct from the trenches! Find controversy and disparate opinions!

Be part of the community with a subscription to *IEEE Software*. This widely respected magazine covers crucial advances in software technology, development, and professional issues.

As a subscriber, you'll get coverage on:

- Leading-edge programming practices
 - Internet development
 - Technical project management
- Component-based development
 - Object-oriented techniques
- Living with rapid technology change

EDITORIAL CALENDAR

1999

JULY

Software Certification *
Software Security

SEPTEMBER

Organizational Design *
Architectural Design

NOVEMBER

Cross Pollinating Disciplines *
Defining Software Engineering
as a Profession

2000

JANUARY

Process Diversity

MARCH

Cobol: New Millenium

APRIL

Engineering in the Small

IEEE SOFTWARE

Subscribe now! See <http://computer.org> for special pricing options or contact our customer service department at membership@computer.org

5. D.J. Cox, "What Can an Artist Do for Science: 'Cosmic Voyage' IMAX Film," *Art @ Science*, C. Sommerer and L. Mignonneau, eds., Springer-Verlag, New York, 1998, pp. 53-59.
6. S. Baden et al., "Workshop on Structured Adaptive Mesh Refinement Grid Methods," *IMA Conference Series*, Springer-Verlag, to be published 1999.

Michael L. Norman is a professor of astronomy at the University of Illinois, Urbana-Champaign, and senior research scientist at the National Center for Supercomputing Applications where he directs the Laboratory for Computational Astrophysics, which is devoted to the development and dissemination of astrophysical modeling software. His interests are in simulating astrophysical phenomena, cosmological structure formation, and scientific visualization. He received his PhD from the University of California, Davis, and has been a staff member at the Lawrence Livermore and Los Alamos National Laboratories as well as the Max Planck Institute for Astrophysics. He has recently become a *CISE* area editor for applications. Contact him at NCSA, Beckman Institute D-25, 405 N. Mathews St., Urbana, IL 61801; norman@ncsa.uiuc.edu; <http://lca.ncsa.uiuc.edu>.

John Shalf works for the NCSA Visualization and Virtual Environments group (www.ncsa.uiuc.edu/SCD/Vis) and the StarTap international high-performance networking project (www.startap.net). His background is in computer engineering, but he tends to hang around physicists and computer scientists these days. He has contributed to a variety of projects in networking, distributed computing, and application development frameworks. Contact him at jshalf@ncsa.uiuc.edu; <http://bach.ncsa.uiuc.edu>.

Stuart Levy develops software for the Virtual Director group at NCSA. His background includes work in mathematical computer graphics, data networking, and systems programming. He enjoys amateur astronomy, as well as peering over the shoulders of professionals. Contact him at slevy@ncsa.uiuc.edu.

Greg Daues is a postdoctoral research associate in the Laboratory for Computational Astrophysics of the NCSA. He has a PhD in physics and has been working with Java and VRML for two years in the development of Web-based scientific visualization tools. Contact him at daues@ncsa.uiuc.edu.