A Parallel Visualization System and its Application to the Earth Simulator

T. Takei[a], K. Muramatsu[b], A. Yoshida[a] and S. Doi[a]

[a] *Internet Systems Research Laboratories, NEC Corporation, 4-1-1, Miyazaki, Miyamae-ku, Kawasaki, Kanagawa, Japan (t-takei@da.jp.nec.com, a-yoshida@ap.jp.nec.com, s-doi@cb.jp.nec.com)*

[b] *Center for Promotion of Computational Science and Engineering, Japan Atomic Energy Research Institute, 2-2-54, Nakameguro, Meguro-ku, Tokyo, Japan (muramatu@koma.jaeri.go.jp)*

Abstract: We have developed a parallel concurrent visualization system named PATRAS (PArallel TRAcking Steering system) and functionality necessary to apply the system to the Earth Simulator. The system concurrently performs all the visualization tasks on a computational server (server-side visualization) in the middle of a numerical simulation. Existing distributed data can also be read directly and post-processed on the server. Visualized images are compressed and transferred to a user's terminal. An advantage of this method is that the volumes of data to be transferred over a network and manipulated on the terminal are small. Furthermore, the volumes of data depend only on the size of the image, and are independent of the ever-growing scale of simulation models. Therefore, efficient visualization of large-scale computation results, for which at least several millions of grid points are used, on high-performance parallel supercomputers becomes possible in wide-area network environments, including the Internet. The visualization module of this system has been parallelized for distributed-memory parallel computers by using MPI (Message Passing Interface) and, to minimize data exchange between the processing elements of the server, parallelization is based on the domain decomposition technique that is generally used in parallel numerical simulations. The Earth Simulator is a distributed-memory parallel system that consists of 640 processor nodes, each of which is a shared memory system composed of eight vector processors. Because typical simulations will usually output several TBs of distributed data on the Earth Simulator, conventional post-processing may no longer be applicable and the server-side visualization is the only feasible way of visualizing such a large amount of data. Many processor nodes of the Earth Simulator will be administrated under a batch-processing environment and computation results will be stored in the cartridge tape library. Therefore, we have developed a scenario function and hierarchical data structuring for practical visualization on the Earth Simulator.

Keywords: Concurrent visualization; Server-side visualization; Parallel visualization; Massive data visualization; Earth Simulator

1. INTRODUCTION

Recent progress in large-scale numerical simulations mainly depends on the marvelous development of parallel computers including distributed-memory massive parallel supercomputers. The ASCI Red and the ASCI White have been developed under the ASCI (Accelerated Strategic Computing Initiative) project in the USA, and the Earth Simulator, which will be put into operation in March 2002, is now being developed in Japan [http://www.gaia.jaeri.go.jp/]. The Earth Simulator project will create a "virtual earth" on a supercomputer to show what the world will look like in the future by means of advanced numerical simulation technology. The system includes the world's fastest supercomputer, with 5,120 processing elements (PEs) and 40 Tflops peak performance, and various kinds of software in the fields of atmosphere-ocean variations, plate tectonics, and information science. Because large-scale numerical simulations output huge volumes of data and several TBs of data will be output by the typical simulations on the Earth Simulator on a daily basis, how efficiently the data is visualized and evaluated will be very important.

Conventionally, visualization refers to post-processing simulation results on a client (a user's

terminal including GWS) side after the numerical simulation has been completed. The simulation results have to be output to the computational server's disk and transferred to the user's terminal before they are processed. However, problems appear when visualizing huge volumes of data in conventional post-processing. For one thing, the data that is distributed on many PEs of the computational server has to be combined somewhere into one batch of organized data, the amount of which is too big to be stored on one PE or the user's terminal. Moreover, it is impossible for a network environment, including the Internet, to transfer the data and a large amount of memory space is necessary to manipulate the data on the user's terminal, whose memory is generally much less than the total amount of memory on the parallel supercomputer.

One solution to such problems may be (semi-) server-side visualization where (a part of or) all of the visualization processes are executed by utilizing the computational server's resources. Some systems have already been proposed for (semi-) server-side visualization. pV3 [Haimes, 1994] developed by MIT generates graphical objects on a parallel computational server and then transfers them to the client side. The graphical objects are rendered by using OpenGL on the client side. CUMULVS [Geist et al., 1997] of ORNL and the "Terascale Visualization" [http://www.llnl.gov/terascale-vis/] of the ASCI project reduce the raw data by using some specialized methods on a computational server and then transfer the reduced raw data to the client side, where remaining visualization processes are done. In PGL [Crockett, 1994] of NASA and RVSLIB [Takei et al., 2001] of NEC, all the visualization processes are done on a computational server and visualized image data is transferred to the client side. In particular, such visualization in PGL and RVSLIB is called server-side visualization.

The authors' parallel concurrent visualization system PATRAS (PArallel TRAcking Steering system) conducts all the visualization processes on a parallel computer and transfers compressed visualized image data over a network to a user's terminal. The volumes of data that are transferred over the network and manipulated on the user's terminal are much smaller than in the conventional visualization methods and large volumes of data are efficiently visualized in wide-area network environments.

2. PARALLEL CONCURRENT VISUALIZATION SYSTEM PATRAS

2. 1 Process Distribution

PATRAS has extended basic visualization functionality of RVSLIB for distributed-memory parallel computers. It is a client-server type system that conducts the server-side visualization mainly because of the volume of data to be transferred over the network and manipulated on the user's terminal. If the number of grid points is n in one spatial direction, the volume is independent of n in PATRAS whereas it is from $O(n^2)$ to $O(n^3)$ in other process distribution methods including semi-server-side visualization. This feature of PATRAS becomes more favorable as the simulation scale becomes larger. However, because the visualization processes consume CPU time of the computational server, it is very important to accelerate them. To achieve this, PATRAS has adopted accelerated visualization algorithms of RVSLIB and parallelized them further.

2. 2 System Configuration

The parallel concurrent visualization in PATRAS is depicted in Figure 1.

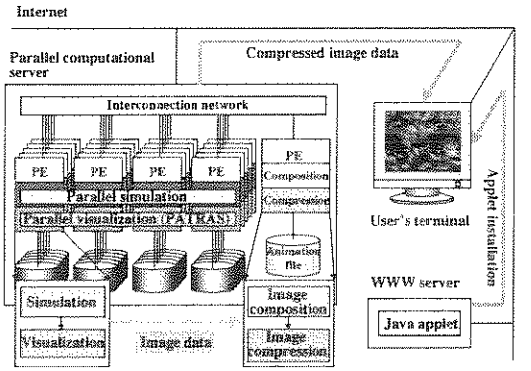


Figure 1. The parallel concurrent visualization in PATRAS.

A part of or all of the computational server's PEs that execute numerical simulation make visualization as well at the same time. In this way, live simulation can be visualized; this type of visualization is generally called concurrent visualization. The simulation program is assumed to be parallelized by using the domain decomposition technique. Each PE generates visualized images of the sub-domain for which it also takes responsibility in the parallel numerical simulation. Here, we refer to this parallelization rule as the "owner computation" rule. The system uses MPI (Message Passing Interface) to exchange necessary information between neighboring PEs for the visualization. The images of all sub-domains are finally collected on one specific PE, where they are integrated into one image that will

be compressed and transferred to the user's terminal. Generated images can be accumulated on the server side to make an animation file. PATRAS can also post-process existing distributed data on a computational server without any data reduction. The accuracy in some visualization such as tracer calculation becomes higher than in conventional methods, where the amount of the raw data is inevitably reduced.

The client module of PATRAS runs on a user's terminal and displays a GUI for operating the system. Because it has been developed as a JAVA applet in which the latest Java2's library JFC (Java Foundation Class) is used, only a Web browser is needed on the terminal. The client module restores and displays the visualized image and sends input visualization parameters to the server, where they are used in the visualization processes at the next time step. Because some of the parameters are also delivered to the simulation program by the server module of PATRAS, they can be used to control the simulation. The communication between the server and the client is HTTP-based.

2. 3 Library Style Format

The server module of PATRAS employs a library style format so that its functionality can be invoked by calling the library in a user's simulation program. Because the memory addresses, where the grid data and the computation results for each time step are held, are directly referred to, it is possible for the visualization processes to run at high speeds and for the amount of necessary memory space to be minimized. Moreover, there is basically no limitation to the volume of data that can be visualized and, if concurrent visualization is conducted, it is not necessary to output computation results to a file.

Figure 2 shows a typical calling sequence of the PATRAS library on each PE. Fundamental visualization can be performed merely by calling the depicted four subroutines, which are exactly the same as those in RVSLIB. RVS_BFC obtains the memory addresses of the arrays that have been used in the parallel simulation to store the grid data and the computation results in the sub-domain on each PE. This subroutine is for simulations that use the three-dimensional BFC (Boundary Fitted Coordinate) grid system. If the simulation uses an unstructured tetrahedron or hexahedron grid, RVS_TET1 or RVS_HEX1 should be invoked respectively instead of RVS_BFC. RVS_MAIN communicates with the client and performs all of the parallel visualization processes by using MPI internally. If the simulation program uses a multi-block grid system, it should invoke one of the address specification subroutines for each block and, after finishing one time-step calculation for all the blocks, invoke RVS_MAIN once. RVS_INIT and RVS_TERM are for initialization and termination respectively. When PATRAS is used as a post-processor, parallel one time-step calculation is replaced by reading distributed one time-step computation results in parallel. Specific post-processing modules are also provided for some standard data formats such as netCDF and GrADS.

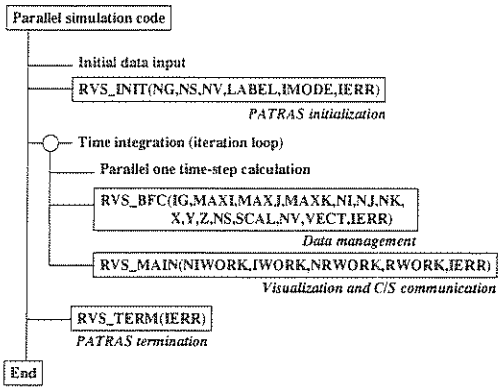
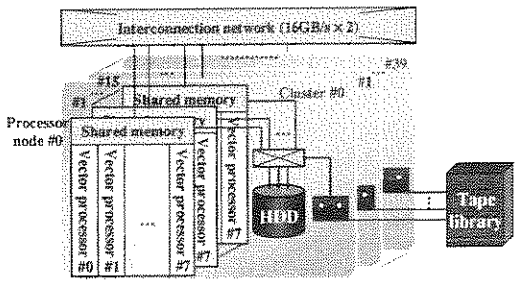


Figure 2. Library style format.

Optional user-defined functions are provided for advanced visualization that uses additional simulation information except for the grid data and computation results. The interfaces of these functions are pre-defined. Users may program their actual contents according to their needs and link them as well to the user's simulation code. RVS_MAIN invokes these functions internally and obtains necessary information. Parallel domain decomposition information, MPI communicator information, obstacle location in the simulation, and so on, will be provided to PATRAS by using the user-defined functions.

3. APPLICATION TO THE EARTH SIMULATOR

The Earth Simulator is a distributed-memory parallel supercomputer that has 640 processor nodes connected by a 640×640 internode crossbar switch. As shown in Figure 3, the system consists of 40 clusters, each of which has 16 processor nodes and a local disk that is connected to the cartridge tape library. Each processor node has eight vector processors, a 16GB shared-memory system, a remote control unit, and an I/O processor. The peak performance of each vector processor is 8 Gflops. The total number of processors is 5,120 and the total peak performance and the main memory capacity are 40 Tflops and 10 TB respectively. The local disks are only used for



Specifications	
Peak performance / processor	8 Gflops
Peak performance / node	64 Gflops
Shared memory	16 GB

Total number of processors	5,120
Total number of nodes	640
Total peak performance	40 Tflops
Total main memory	10 TB

Figure 3. Hardware configuration of the Earth Simulator.

temporary files, and permanent files such as computation results have to be stored in the cartridge tape library. Therefore, to launch the simulations, necessary input data has to be loaded from the cartridge tape library to the local disks beforehand. Computation results are output to the local disks temporarily and, when the simulation is finished, they are transferred to the cartridge tape library. Although one cluster is a TSS cluster where interactive job execution is allowed, remaining (batch-job) clusters are only for batch processing without any communication with the user's terminal. Therefore, large-scale data visualizations as well as large-scale numerical simulations have to be done in the batch-processing environment on the batch-job clusters.

We have made some functional improvements to PATRAS to construct a visualization system for the Earth Simulator. The configuration of this system is shown in Figure 4. The characteristic functions are the scenario function and hierarchical data structuring, which are overviewed in the following sub-sections.

3.1 Scenario Function

Visualization parameters like the viewpoint location are usually changed from GUI in PATRAS. However, if the process of the parameter change is described as a scenario in advance, automatic visualization will be done by the visualization system for the Earth Simulator. The parameters will be automatically updated according to the scenario as the time step of a simulation proceeds. This function is very useful in the batch-processing environment on the batch-job clusters of the Earth Simulator.

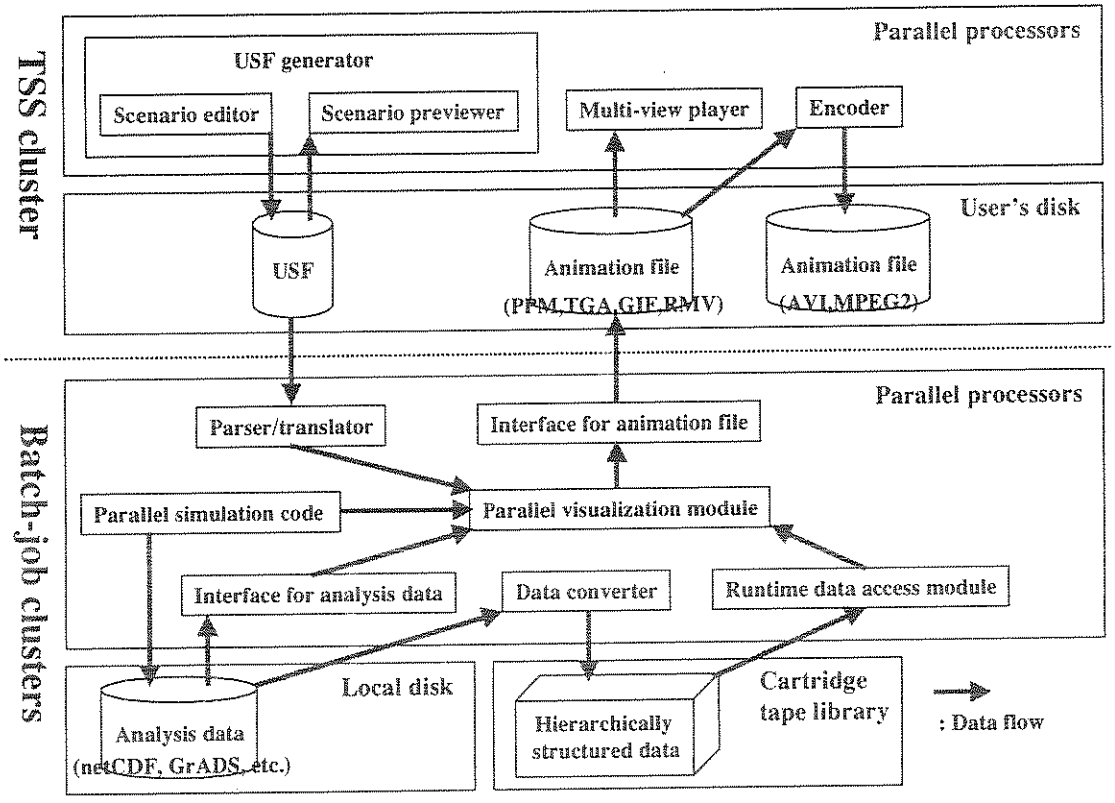


Figure 4. Modules and data flows in the visualization system for the Earth Simulator.

The scenario is written in a key-frame format, an example of which is shown below.

```
$beginframe step=300;
  $viewing;
     eye = 1.0 0.0 0.0;
     look_at = 0.0 0.0 0.0;
  $end;
$endframe;
#
$beginmotion;
  interpolation='polar';
$endmotion;
#
$beginframe step=600;
  $viewing;
     eye = 0.0 1.0 0.0;
  $end;
$endframe;
```

The meaning of this scenario is that the viewpoint location ("eye") is rotated from (1.0, 0.0, 0.0) to (0.0, 1.0, 0.0) in the physical space as the time step ("step") of a simulation proceeds from 300 to 600. The center ("look_at") of the rotation is located at the origin and the view direction is always toward this point. Parameter values are automatically interpolated if they are different between contiguous key frames. The interpolation method is specified by the keyword "interpolation". In addition to the polar interpolation shown in the sample scenario, linear and moving average interpolations are also possible.

Although many parameters of the visualization system for the Earth Simulator can be described in a scenario, only parameters that should be changed (e.g., the viewpoint location in the sample scenario) have to be described in the actual scenario. Initial values will continue to be used for other parameters. When a simulation program that invokes the visualization system for the Earth Simulator is executed with a prepared scenario on the batch-job clusters, an animation file will be generated as the simulation proceeds. If more than one scenario is prepared (the maximum is eight), animation is generated independently for each. This is called multi-camera visualization. A multi-view animation player is provided to replay more than one animation file synchronously at each simulation time step.

3. 2 Hierarchical Data Structuring

If computation results are stored in the normal array format, all the results must usually be loaded from the cartridge tape library to the local disks for the visualization on the Earth Simulator, and a great deal of loading time is necessary for large-scale data. Furthermore, the raw data must be randomly accessed many times. However, because image size is limited to the display size, some kinds of graphical tools cannot utilize the original resolution of the raw data. Let us consider the computation results of a global climate simulation where the surface of the earth is divided into a mesh of 10Km×10Km grid cells. Although there are 4000×2000 grid cells in the horizontal plane, the amount of data that is actually used in generating a 512×512-size contour image will be only a few hundredths of the original raw data regardless of the range of the visualized domain. The amount can be further reduced at places where the data changes little.

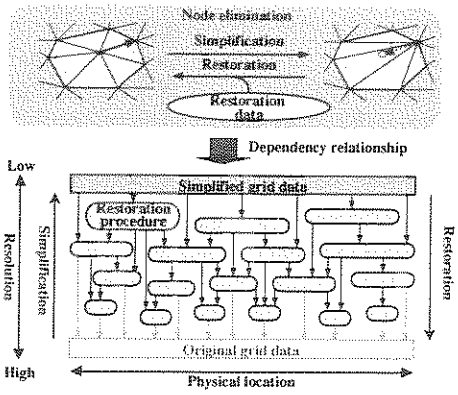


Figure 5. Graphical representation of the hierarchical data structure.

One solution to the above-described problems is hierarchical data structuring that enables the necessary part of the data to be efficiently gathered according to the user's requirements in terms of location, range, and accuracy for the visualization. The multi-tetrahedrization method is used in our system to make hierarchically structured data from the raw data (Figure 5). This method expands the dimensions of target data to be processed in the multi-triangulation method [Puppo and Scopigno, 1997], which is generally used to reduce the number of polygonal elements that represent complicated surface geometry such as geographical features without much deformation of the original shape. The multi-tetrahedrization method can be applied to a wider variety of grids than other methods such as wavelet transformation [Wong and Bergeron, 1997] and enables local refinement (node restoration) to be realized. To construct hierarchical data from the raw data, a local simplification procedure (node elimination) is executed node by node. The node elimination generates some deviation from the original data, and nodes are eliminated in the order of smallest deviation. A file stores necessary information for the procedure for restoring the eliminated node, the

amount of deviation generated by the simplification procedure, and the dependency relationships to other simplification procedures. In this way, the original raw data is divided into maximally simplified data and a hierarchical group of restoration procedures, both of which construct the hierarchically structured data. Node restoration procedures necessary to obtain the data with specified accuracy can be done only by gaining the local and relative access to the hierarchically structured data. This type of simplification and restoration are loss-less procedures. We have further improved the algorithm so that the dependency of necessary CPU time on the number of grid elements becomes linear and the data size increase is within several ten percent. Elapsed time for the simplification procedures in an unstructured grid was measured on a PC with a single CPU (PowerPC G3/500MHz), and the result is shown in Figure 6. The original grid comprised 800,000 elements. From the figure, it can be seen that the elapsed time for eliminating a fixed number of nodes is almost constant and is independent of the number of remaining elements.

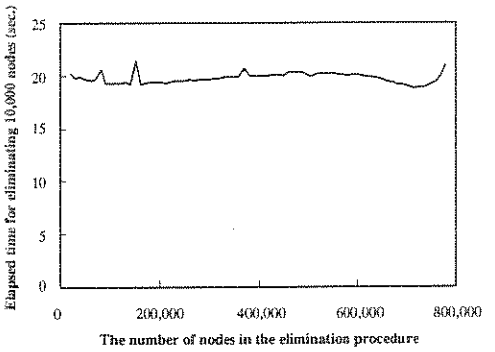


Figure 6. Elapsed time for hierarchical data structuring.

4. CONCLUSIONS

With the increasing popularity of large-scale numerical simulations on distributed-memory parallel supercomputers, for which simulations at least several million grid points are used, problems have arisen in conventional visualization methods. To address these problems, PATRAS performs all the visualization tasks on a computational server and uses compressed visualized image data for the communication between the server and the user's terminal. Consequently, the volumes of data that are transferred over the network are much smaller than the raw data and efficient visualization of large volumes of data in wide-area network environments becomes possible. Furthermore, we have added the functional improvements of a scenario function and hierarchical data structuring to PATRAS, so that massive amounts of distributed data can also be practically visualized on the Earth Simulator. We plan to make further improvements, including parallelization of the hierarchical data structuring, and to open the system to the public in March 2002.

5. REFERENCES

Crockett T.W., Design considerations for parallel graphics libraries, NASA CR-194935, 1994.

Geist G.A., J.A. Kohl, and P.M. Papadopoulos, CUMULVS: providing fault-tolerance, visualization and steering of parallel applications, *International Journal of High Performance Computing Applications*, 11(3), 1997.

Haimes R., pV3: a distributed systems for large-scale unsteady visualization, AIAA Paper 94-0321, 1994.

Puppo, E., and R. Scopigno, Simplification, LOD and multiresolution - principles and applications, *Eurographics'97 Tutorial Notes*, 1997.

Takei, T., H. Matsumoto, K. Muramatsu, and S. Doi, Parallel vector performance of concurrent visualization system RVSLIB on SX-4, *Proc. 3rd Pacific Symposium on Flow Visualization and Image Processing*, 2001.

Wong, P.C., and R.D. Bergeron, Hierarchical representation of very large data sets for visualization using wavelets, In: G.M. Nielson, H. Hagen, H. Muller (eds.): Scientific Visualization Overviews Methodologies Techniques, IEEE Computer Soc. Press, 1997.

6. WEB SITES

http://www.gaia.jaeri.go.jp/
http://www.llnl.gov/terascale-vis/