Fourth Eurographics Workshop on Parallel Graphics and Visualization (2002)
D. Bartz, X. Pueyo, E. Reinhard (Editors)

# Parallel Performance Optimization of Large-Scale Unstructured Data Visualization for the Earth Simulator

L. Chen, [1] I. Fujishiro[1 2] and K. Nakajima [1]

[1] Research Organization for Information Science & Technology, Tokyo, Japan
[2] Graduate School of Humanities and Sciences, Ochanomizu University, Tokyo, Japan

**Abstract**

*This paper describes some efficient parallel performance optimization strategies for large-scale unstructured data visualization on SMP cluster machines including the Earth Simulator in Japan. The three-level hybrid parallelization is employed in our implementation, consisting of message passing for inter-SMP node communication, loop directives by OpenMP for intra-SMP node parallelization, and vectorization for each processing element (PE). In order to improve the speedup performance for the hybrid parallelization, some techniques, such as multi-coloring for removing data race and dynamic load repartition for load balancing, are considered. Good visualization images and high parallel performance have been achieved on Hitachi SR8000 for large-scale unstructured datasets, which shows the feasibility and effectiveness of our strategies.*

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; C.2.4 [Computer-Communication Networks]: Distributed Systems—Client/Server, Distributed Applications; J.2 [Physical Sciences and Engineering]

## 1. Introduction

In recent years, shared memory symmetric multiprocessor (SMP) cluster architecture has become increasingly important in the parallel computing areas, such as multiprocessor clusters from SUN, SGI, IBM, a variety of multi-processor PC clusters, supercomputers like the NEC SX-6 and all Accelerated Strategic Computing Initiative (ASCI)[1] machines. The study on improving parallel performance for large-scale data visualization on SMP cluster machines is very meaningful for advancing the high performance computing research.

In 1997, the Science and Technology Agency of Japan began a 5-year project to develop a new supercomputer named the Earth Simulator [2]. The goal is the development of both hardware and software for predicting various Earth phenomena by numerical simulations using the supercomputer. The Earth Simulator has an SMP cluster architecture and consists of 640 SMP nodes, where each SMP node consists of 8 vector processors with 1 gigaflops speed and 2 gigabyte memory for each processor. GeoFEM is known as an important part of the Earth Simulator Project to develop a large-scale finite element analysis platform for solid earth simulation [3]. This study has been conducted as part of GeoFEM toward developing a parallel visualization subsystem for GeoFEM.

The goal of the parallel visualization subsystem is to provide the users with a visual exploration environment for various types of 3D datasets arising from the analysis subsystems in GeoFEM[4]. Our subsystem can be executed concurrently with the computation subsystems on the same high-performance parallel computer and supports many kinds of parallel visualization techniques, covering scalar, vector and tensor fields. In order to make parallel performance as high as possible on the supercomputers, especially on the Earth Simulator, the following strategies were employed in our implementation:

- Visualization methods are carefully selected to be suitable for parallelization and coupling well with the computation subsystems;
- Three-level hybrid parallelization is employed to fit for the SMP cluster architecture. That means:
  - *Inter-SMP node*: MPI [5];
  - *Intra-SMP node*: OpenMP [6] for parallelization;
  - *Individual PE*: Compiler directives for vectorization / (pseudo) vectorization [7].
- Dynamic load repartition is done to keep load balance.

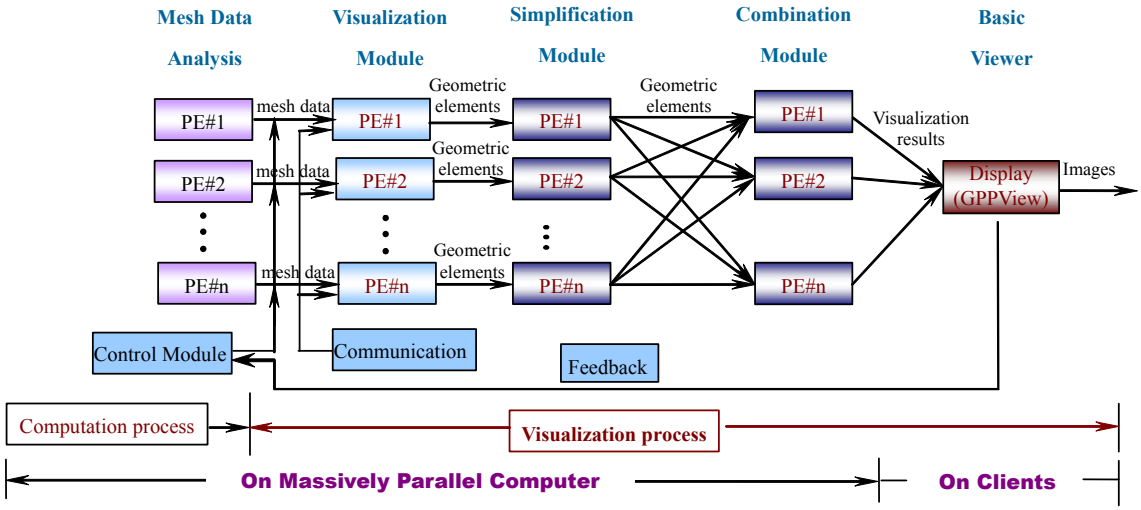



**Figure 1:** *Framework of parallel visualization subsystem in GeoFEM*

This paper will give some detailed description of our strategies for parallel performance optimization. It is organized as follows. First, in Section 2, we give a brief introduction to the visualization subsystem in GeoFEM developed for the Earth Simulator project. Then we will take parallel volume rendering (PVR) module in our subsystem as example, to introduce our basic design of its implementation to take full account of parallel performance and the goal of our visualization subsystem. Next, some techniques for improving hybrid parallel performance are described in Section 4. Finally, dynamic load balancing to improve speedup performance among SMP nodes is introduced in Section 5. The experimental results and future work will be given in Section 6 and Section 7, respectively.

## 2. Parallel Visualization Subsystem in GeoFEM [4]

As a part of the Earth simulator software, our visualization subsystem should satisfy the following requirements:

- Powerful visualization functions;
- The ability to cope with large-scale datasets with a high parallel performance;
- Applicable to complicated unstructured grids;
- Suitable for the architecture of the Earth simulator to achieve the best performance on the supercomputer.

Towards these four targets, we have been developing a visualization subsystem in GeoFEM. The framework of our parallel visualization subsystem is shown in Figure 1.

We implemented our subsystem so as to perform the concurrent visualization with computation on the same high-performance parallel computer. Therefore, we need not save the computational results on the disk or transfer them via network, which can avoid the limitations of storage capacity for large-scale data. Meanwhile, we can make full use of computational server's huge memory to complete visualization. Moreover, once computation modules finish one time-step computation, visualization modules will start immediately. Because visualization is usually much faster than computation process, during each timestep, multiple visualization methods can be done to generate visualization results by different methods and parameters. Two kinds of output styles are provided. One is to output simplified graphics primitives to clients. On each client, the users can set viewing, illumination, shading parameter values, and so on, and display the graphics primitives by the GPPView software, which is also developed by the GeoFEM group[3]. On the computation server, the users only specify in the batch files the visualization methods such as cross-sectioning, streamlines, and related parameters. Moreover, on the Earth Simulator, maybe even the simplified geometric primitives are still too large to transfer. Therefore, we developed the second style to directly output an image or a sequence of animation images to clients.

Our subsystem provides many kinds of parallel visualization techniques, including parallel surface rendering, parallel volume rendering [8], parallel interval volume fitting [9] for scalar data; parallel streamlines, LIC [10], parallel particle tracking, parallel volume rendering [11] for vector data; and parallel hyperstreamline[12] for tensor data. Some feature analysis techniques such as adaptive extraction of isosurfaces/interval volumes based on Hyper Reeb

graph[13], significance map generation based on flow topology analysis[14] have been used to improve the quality of visualization results.

In the following, we will take PVR as example to describe the strategies on improving parallel performance.

### 3. Suitable Design for PVR

Due to the high time and large memory cost of volume rendering method, PVR has been a hot topic in the area of parallel rendering. Many papers for PVR have been presented[15]. There are many kinds of classification methods for PVR, according to traversal order, grid type, composition order, target hardware, and so on. Many good methods can achieve high performance for different cases[16,17,18,19]. When we select volume rendering method, we pay more attention to the possibility of high parallel performance for very complicated grids and huge data size. Our implementation involves partitioning complicated grids by supervoxels, buiding Branch-on-need Octree (BONO)[20] in each supervoxel, rendering on each SMP node, and compositing among SMP nodes finally.

#### 3.1 Partition Complicated Grids by Supervoxels

Because our PVR module need be concurrent with the computational part, it is very important to keep the consistent parallel style and data structure with computation. GeoFEM system adopts an object-space parallelism. The entire data domain is partitioned into distributed local datasets[3], and each partition is assigned to one SMP node. In order to reduce the communication among the SMP nodes, the vertex-based partitioning is adopted in accordance with network theory. As shown in Figure 2, overlapped elements exist at each domain boundary. For this kind of partition, every internal vertex must belong to one subdomain, whereas the external vertex is belong to an overlapped element. Therefore, internal vertices in a subdomain have all of the connected vertex information, which can avoid much communication during the process of visualization, and be convenient for combining all the results generated by each SMP node into the final image.

However, in the GeoFEM system, the grid is usually very complicated including unstructured, hybrid of tetrahedra, hexahedra, prism and hierarchical. The complicated grids will make image composition very difficult and time-consuming when we adopt ray-casting volume rendering method. Although much work has been done on getting correct composition order for complex data by parallel BSP tree[17,21], it need store a large amount of intermediate results. Very high bandwidth and large memory are needed for huge data size. In order to be available for both hybrid and hierarchical grids and to reduce memory cost, we first transform the original data into BONO[20] data structure by parallel resampling and adaptive refinement process. Some supervoxels are used to

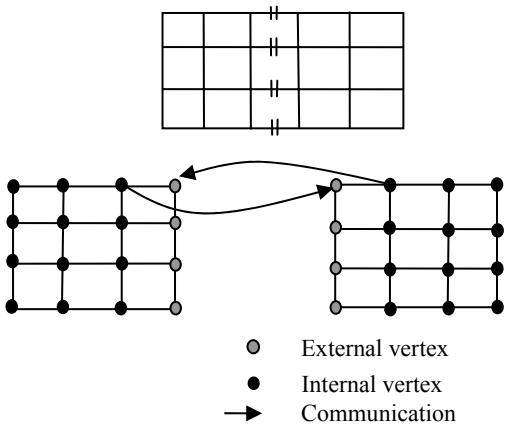



**Figure 2:** *Data structure for parallelization among SMP nodes*

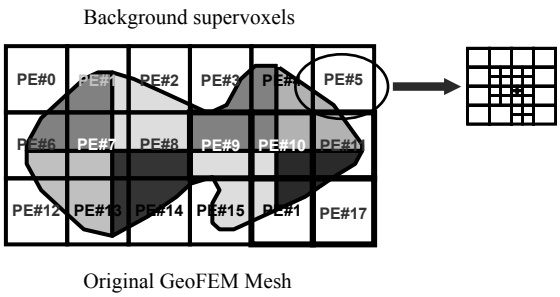



**Figure 3:** *Parallel grid transformation*

divide the original data space into some regular domains uniformly (as shown in Figure 3). Then we assign each supervoxel to an SMP node, and generate the subimage for each supervoxel on each node. Finally we composite these subimages in a correct front to back order of supervoxels according to the position of viewpoint. The final accumulated opacity at each pixel for each supervoxel needs be recorded for the final subimage composition.

Supervoxels can be generated automatically according to the range of the whole dataset, and also can be defined by the users in the batch file. Meanwhile, supervoxels can just cover a local part of the whole dataset as the clipping window. It also can control different display details on the final image by assigning different size and refinement precision for each supervoxel. In a region of high interest, a smaller supervoxel and a high refinement

precision are usually used.

### 3.2 Build Branch-on-need Octree in Each Supervoxel

In each supervoxel, resampling is done to generate regular grids first. Then the BONO structure is built by adaptive combination or refinement according to the difference of data values located on adjacent vertices. It can not only improve the space storage efficiency but also quickly skip over empty or transparent voxels by attaching the minimum and maximum data values on its non-leaf tree nodes [20].

Although it takes extra time for resampling and building a BONO tree, it can accelerate the generation of a sequence of animated images greatly for different viewpoints and opacity transfer functions. Since the computational part usually need spend more than several days to several months to finish a large simulation with a large number of timesteps, and the computation results are too huge to save on the hard disk or transfer to the client machines, it is better to generate many volume rendered images at different viewpoints and parameter values to reveal the whole dataset from different aspects in one time, which can avoid the troubles of selecting suitable settings for volume rendering.

### 3.3 Rendering and Compositing

During subimage generation process for each supervoxel, some traditional acceleration techniques are used, such as early ray termination[22], adaptive raycasting[22], finding the projection area on the screen for each supervoxel first to avoid useless ray casting.

Based on the composition order of supervoxels, the subimages are composed from front to back. Each PE is responsible of compositing almost the same number of the pixels, which is decided by the image resolution and PE numbers.

## 4. Three-level Hybrid Parallelization for PVR

SMP clusters have a hybrid-parallel architecture that consists of a number of nodes which are connected by a fast interconnection network. Each node consists of multiple processors which have access to a shared memory, while the data on other nodes may usually be accessed by means of explicit message-passing (as shown in Figure 4).

### 4.1 Parallel Performance for Our PVR module: Pure MPI Versus Hybrid MPI+OpenMP

For this kind of hybrid architecture machines, *loop directives + message passing* style *hybrid* programming model appears to be very effective when message passing such as MPI[5] is used in inter-SMP node communication, and

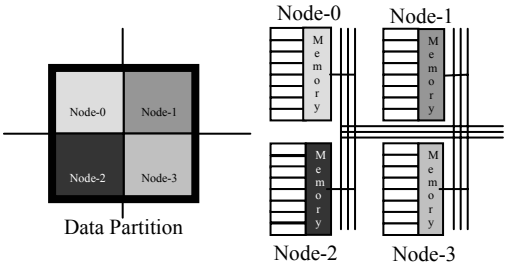



**Figure 4:** *Parallel Visualization on SMP cluster architecture. Each partition corresponds to an SMP node.*

when intra-SMP node parallelization is guided by loop directives such as OpenMP[5], which can make full use of the shared memory available within SMP nodes.

However, sometimes it is difficult to get a high speedup performance for the hybrid MPI+OpenMP parallelization. It is often even worse than pure MPI[23], so up to now most applications on SMP machines still adopted a pure MPI parallel programming model due to the following three reasons:

- Although OpenMP parallel can avoid communication overhead, it has to involve thread creation and synchronization overheads;
- If the MPI routines are invoked only outside of parallel regions, all communication is done by the master thread while the other threads are inactive;
- One thread is not able to saturate the total inter-node bandwidth that is available for each node.

For our PVR method, hybrid programming model is a better choice. The communication happened mainly in the following three cases:

- Gradient computation based on the original mesh: During this process, the boundary vertices need be communicated with other PEs. In the hybrid parallel model, the number of total boundary vertices will be reduced greatly due to the large granularity of data partition. Meanwhile, the communication message size of the boundary vertex information for each MPI process will be larger than that in the pure MPI model, which will reduce the latency overhead in the inter-node communication.
- Domain repartition and resampling process: Obviously, in this case, the hybrid method will have larger domain on each node, which will reduce data movement amount from one processor to another.
- Subimage composition process: In our PVR method, each PE is responsible for compositing one portion of the final image. Therefore, each PE need communicate with all other PEs to get its corresponding composition

pixels. Smaller number of MPI processors will reduce the latency overhead.

In general, because the latency overhead will surprisingly increase as the MPI process number gets larger in MPI communication, and the target of our visualization modules will be used on the Earth Simulator with a very large number of PEs, hybrid parallel is an alternative choice for our implementation.

We employed the three-level hybrid parallel visualization on large-scale unstructured grids based on the SMP cluster architecture.

In order to achieve efficient parallel/vector performance for large-scale unstructured grids, we take full consideration of the following three issues:

- Local operation and no global dependency
- Continuous memory access
- Sufficiently long loops

### 4.2. Multi-coloring to Accelerate PVR

In order to achieve efficient parallel performance for visualization modules, no dependency and no data race are very critical. However, in our visualization subsystem, data race exists in some parts of codes. For example, before ray-casting unstructured grids, the gradient on each vertex based on the shape function of each mesh element should be computed first. The pseudo-algorithm for getting gradients is as follows:

```
#pragma omp parallel
{
   for(i=0;i<num_element;i++) {
      compute Jacobian matrix of shape function;
      for(j=0; j<NUM_VERTEX_ELEMi ;j++) {
      for(k=0; k< NUM_VERTEX_ELEMi;k++)
      accumulate gradient value of vertex j contributed by vertex k;
    }  }
}
```

Obviously, there is a data race on accesses to the shared variable *gradient* because one vertex is often shared by several elements. Although it can be avoided by adding mutual exclusion synchronization or writing to a *private variable* in each thread and then copying to the *shared variable*, they need either more extra time cost or more extra memory cost. In our parallelization, a multi-coloring strategy is adopted[24], which can get rid of data race very easily. In this method, as shown in Figure 5, elements are assigned different colors so that each element is marked by the color different from all the colors of its adjacent elements. After coloring, the elements assigned a same color have no common vertex, so they can be parallelized without any data race by OpenMP. Meanwhile, the elements with different colors should be computed in a serial order.

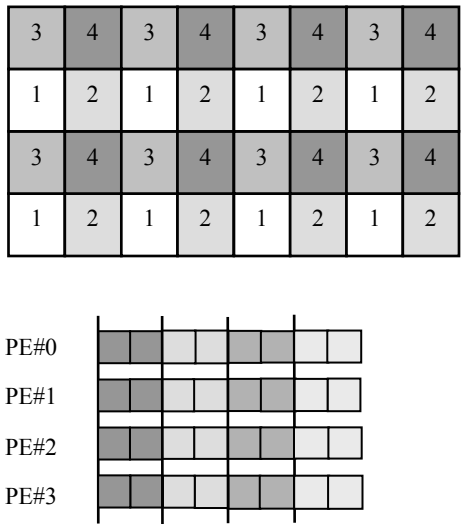



**Figure 5:** *Multi-coloring for removing data race.*

There is a minimum color number to satisfy the above multi-coloring condition. Although the number of colors can be any one larger than the minimum one, the load balance is more difficult to keep with the increase of color number. The minimum color number is the best one for getting the high parallel performance in the intra-SMP node parallelization by OpenMP because it can make the loop length in vertex traversal as long as possible.

Although it takes some extra time for multi-coloring, it can improve parallel performance much for large time-step dataset visualization. The multi-coloring process only need be executed once in the first time-step.

### 4.3. Improve Vector Performance for Each PE

Although vectorization can be done automatically by compiler, sometimes it is very difficult to reach a high vector performance if we do not organize the codes well to make it suitable for vector performance. For example, the sufficient long loop is very critical for the vector performance. We make this kind of loops as possible as we could by the following ways[25]:

(1) Combine some short loops into one long loop by reordering;

(2) Exchange the innermost and outer loop to make the innermost loop longer;

(3) Minimize load/store latency as possible as we could.

## 5. Dynamic Load Repartition Among SMP Nodes

During visualization, rendered voxels often accumulate in small portions of the field and their number changes greatly, which leads to very unbalance load distribution. Therefore, load balancing is very important for parallel performance.

In PVR, the number of rendered voxels on each SMP node should keep almost equal for load balance. It is decided by several factors, such as the number of non-empty voxels, opacity transfer function and viewpoints. Since these factors are usually changed at each timestep for time-varying datasets, the load repartition should be a dynamic process.

A scattered decomposition was adopted by some previous methods [26], which can get a very good load balance. However, this method lost data coherence, which would slow down parallel performance on the SMP cluster machine. Moreover, this method is based on a sort-last architecture, which should send a huge number of stencil collections for further sorting in the compositing nodes. It is not suitable for huge-scale datasets. Content-based load balancing is also a very good method presented by Silver, et al. [19] However, the number of slabs in their method is too large, which is not suitable for our case.

In order to keep the load balance, dynamic load repartition among the initial supervoxels is done in our method. During the building of BONO structure, the number of rendered voxels is counted. Then according to the number of rendered voxels on each SMP node, move a subvolume of a supervoxel from one SMP node with a larger number of rendered voxels to another SMP node with a smaller one. Due to the limitation of regular supervoxel shape, the absolute balance cannot be achieved in our method. The default threshold in our module is one-twentieth of the average number of rendered voxels on each SMP node.

After the load repartition, the number of supervoxels on some SMP nodes maybe increases. The new supervoxels should be added into supervoxel list for deciding the final composition order.

## 6. Experimental Results on SR8000

We have applied our parallel modules to some unstructured large datasets generated by GeoFEM computation modules on SR8000 (8 PEs, 8GFLOPS peak performance and 8GB memory for each node. 128 nodes (1024 PEs), 1.0TFLOPS peak performance and 1.0TB memory for the total system).

**Test 1**: Parallel volume rendering an Earthquake dataset (Data courtesy of Mikio Iizuka in GeoFEM)

As shown in Figure 6, the parallel volume rendering module was applied to reveal the distribution of the magnitude of displacement attribute. From the image, we can see the mantle structure and the data distribution on the fault surface very clearly. On SR8000, with 8 SMP nodes, it took about 5.52 seconds to generate a single 400-by-190 pixels image, and about 12.41 seconds to generate 16 animation images when the viewpoint is rotated along y axis.

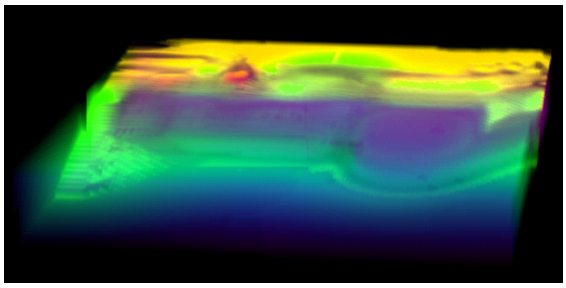
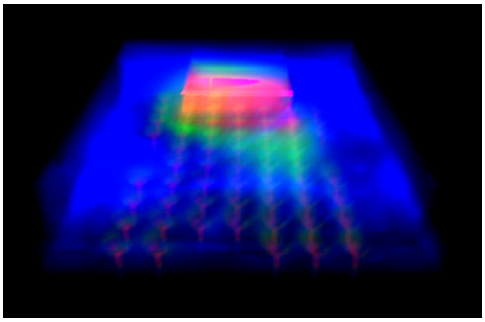


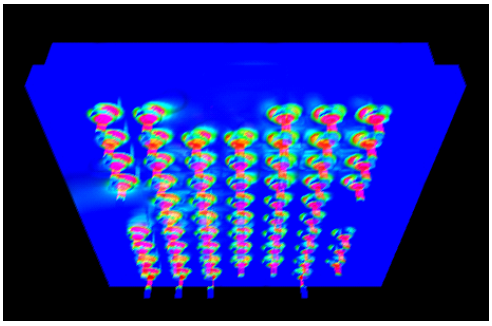
**Figure 6:** *Parallel volume rendering image for a large dataset to simulate the earthquake occurred in southwest Japan in 1944, with 2,027,520 unstructured grid elements. The displacement data attribute is mapped to color. On SR8000, with 8 SMP nodes, it took about 5.52 seconds to generate a single image (Image resolution: 400*190 pixels), and about 12.41 seconds to generate 16 animation images with a rotation of the viewpoint along y axis.*

**Test 2:** Parallel volume rendering a Pin Grid Array (PGA) dataset (Data courtesy of H. Okuda and S. Ezure in The University of Tokyo).

As shown in Figure 7, the parallel volume rendering module was applied to reveal the distribution of the equivalent scalar value of stress. This 3D unstructured dataset has 7,869,771 vertices and 7,649,024 grid elements. We tested it on 1 node (8PEs), 2 nodes (16 PEs), 4 nodes (32 PEs), 8 nodes (64 PEs), 16 nodes (128 PEs) by MPI+OpenMP hybrid parallel module and three-level hybrid parallel module, and compared with the results of pure MPI on the same PE number with the same data size. The speedup performance is shown as Figure 8. To generate the image in Figure 7(a), pure MPI took about 98.33, 48.79, 27.53, 15.39 and 10.40 seconds on 8 PEs, 16 PEs, 32 PEs, 64 PEs and 128 PEs, respectively, whereas MPI+OpenMP hybrid parallel took about 97.96, 48.73, 25.36, 13.34, and 7.48 seconds on 1 node, 2 nodes, 4 nodes, 8 nodes and 16 nodes, respectively. Furthermore, MPI+ OpenMP+ Vectorization hybrid parallel took 35.65, 17.44, 9.28, 5.29 and 3.07 seconds, respectively. We can see hybrid parallelization especially three-level hybrid parallelization can improve parallel performance much on the SR8000 machine.



*(a) Top view*



*(b) Bottom view*

**Figure 7:** *Parallel volume rendering images to show the equivalent scalar value of stress by the linear elastostatic analysis for a PGA dataset with 7,869,771 vertices and 7,649,024 grid elements.*

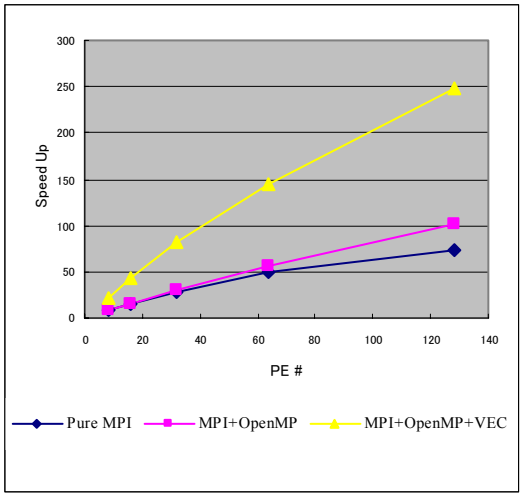



**Figure 8:** *Comparison of speedup performance for the PGA dataset with 7,649,024 grid elements.*

**Test 3:** Parallel volume rendering an underground water dataset (Data courtesy of Kengo Nakajima in GeoFEM).

This 3D unstructured dataset simulated groundwater flow and convection/diffusion transportation through heterogeneous porous media for deep geological disposal of high-level radioactive waste. By using this dataset, we mainly tested the effectiveness of the dynamic load repartition strategy we adopted. In this dataset, many elements have no value in some time-steps. On SR8000 with 8 SMP nodes, for more than 10 million vertices and 100 timesteps, without dynamic load repartition it took about 2.35 seconds for one time-step on average. After dynamic load reparation, it just took about 1.56 seconds on average for one time-step. This demonstrates the effectiveness of our dynamic load repartition method.

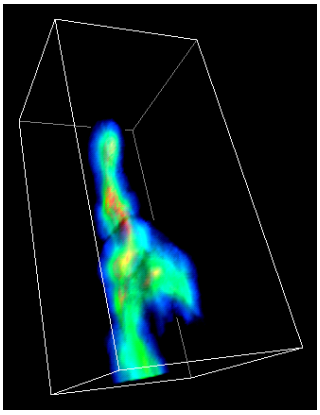


**Figure 9:** *Parallel volume rendering image to show the concentration distribution for an underground water simulation dataset with about 10 million vertices and 100 timesteps.*

## 7. Conclusions and Future Work

This paper reported our parallel performance optimization strategies for large-scale unstructured data visualization on SMP cluster machines. Good visualization images and high parallel performance have been obtained on Hitachi SR8000 by applying the above techniques to some unstructured large datasets in GeoFEM, which shows the feasibility and effectiveness of our method.

The hardware of the Earth Simulator has been successfully finished by NEC company in March this year. Up to now, it is the fastest supercomputer in the world with a peak performance of 35,600 gigaflops. Our software will be installed and run on it this June. Because SR8000 has the similar architecture, hopefully our system also can achieve high parallel performance on the Earth Simulator. The future work will focus on the tests on the

Earth Simulator and further improvements of the parallel performance.

**Acknowledgements**

This study is a part of the Solid Earth Platform for Large-Scale Computation project funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology through Special Promoting Funds of Science & Technology.

Furthermore, the authors would like to thank Kazuo Minami in GeoFEM for providing many good materials on vectorization and Hiroaki Matsui for helpful discussions.

**References**

1. Accelerated Strategic Computing Initiative (ASCI) Web Site: http://www.llnl.gov/asci/.
2. Earth Simulator Research and Development Center Web Site: http://www.es.jamstec.go.jp/.
3. GeoFEM Web Site: http://geofem.tokyo.rist.or.jp/.
4. I. Fujishiro, L. Chen, Y. Takeshima, H. Nakamura, Y. Suzuki. Parallel visualization of gigabyte datasets in GeoFEM, *Journal of Concurrency and Computation: Practice and Experience* (in print).
5. MPI Web Site: http://www.mpi.org.
6. OpenMP Web Site : http//www.openmp.org.
7. Hitachi SR8000 Web Site :http:/ /www.hitachi.co.jp/ Prod/comp/hpc/foruser/sr8000/.
8. M. Levoy. Display of surfaces from volume data. *IEEE CG&A*, **8**(3): 29-37, 1988.
9. I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima. Volumetric data exploration using interval volume. *IEEE TVCG*, **2**(2): 144-155, 1996.
10. B. Cabral, C. Leedom. Image vector field using line integral convolution. In *Computer Graphics Proceedings*, ACM SIGGRAPH, 263-272, 1993.
11. L. Chen, I. Fujishiro and Y. Suzuki. Comprehensible volume LIC rendering based on 3D significance map. *Proceedings of SPIE Conference on Visualization and Data Analysis'02,* San Jose, 2002, pp. 142-153.
12. T. Delmarcelle, and L. Hesselink, Visualizing second-order tensor fields with hyper-streamlines. *IEEE CG&A*, 1993, **13**(4): 25-33.
13. I. Fujishiro, T. Azuma, Y. Takeshima, and S. Takahashi. Volume data mining using 3D field topology analysis. *IEEE CG&A*, **20**(5): 46-51, 2000.
14. J. Helman, L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE CG&A*, **11**(3): 36-46, 1993.
15. C. M. Wittenbrink. Survey of parallel volume rendering algorithms. *Proceedings of International Conference on Parallel distributed Processing Techniques and Applications,* Las Vegas, Nevada, 1998, pp. 1329-1336.
16. R. Yagel. Towards real time volume rendering. In *Proceedings of GRAPH-ICON*, Saint-Petersburg, Russia, 1996, pp. 230-241.
17. K. L. Ma, J. Painter, C. D. Hansen, and M. F. Krogh. Parallel volume rendering using binary-swap compositing. *IEEE CG&A*, **14**(4):59-67, 1994.
18. C. M. Wittenbrink and A. K. Somani. Time and space optimal parallel volume rendering using permutation warp ing. *Journal of Parallel and Distributed Computing*, **46**(2):148-164, 1997.
19. C. Silva. Parallel volume rendering of irregular Grids. *Ph.D. thesis*, State University of New York at Stony Brook, 1996.
20. J. Wilhelms, and von A. Gelder. Octree for faster isosurface generation. *ACM Trans. on Graphics*, **11**(3), 1992.
21. C.R. Ramakrishnan, C. Silva. Optimal processor allocation for sort-last compositing under BSP-tree ordering. *SPIE Eletronic Imaging, Visual Data Exploration and Analysis IV*, 1999.
22. M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, **9**(3): 245-261, 1990.
23. R. Rabenseifner. Communication bandwidth of parallel programming models on hybrid architectures. *ISHPC 2002*, LNCS 2327, pp. 401-412, 2002.
24. K. Nakajima and H. Okuda. Parallel iterative solvers for unstructured grids using Directive/MPI hybrid programming model for GeoFEM platform on SMP cluster architectures. *Journal of Concurrency and Computation:Practice and Experience* (in print).
25. K. Minami and H. Okuda. Performance optimization of GeoFEM on various computer architecture. *GeoFEM Report 2001-006*, RIST/Tokyo, 2001.
26. K.-L. Ma, et al. A scalable parallel cell projection volume rendering algorithm for 3D unstructured data. *In Proc. of 1997 Symposium on Parallel Rendering*, pp. 95-104, 1997.

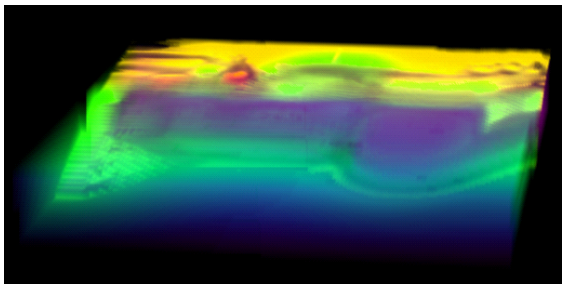


**Figure 6:** *Parallel volume rendering image for a large dataset to simulate the earthquake occurred in south-west Japan in 1944, with 2,027,520 unstructured grid elements. The displacement data attribute is mapped to color. On SR8000, with 8 SMP nodes, it took about 5.52 seconds to generate a single image (Image resolution: 400*190 pixels), and about 12.41 seconds to generate 16 animation images with a rotation of the viewpoint along y axis.*

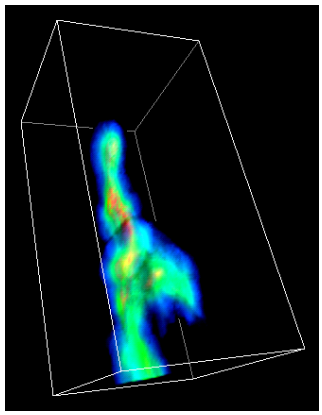
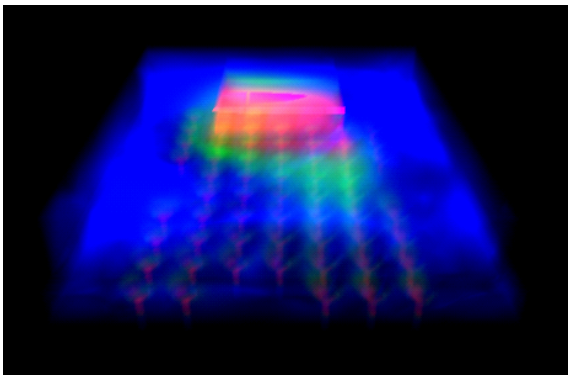
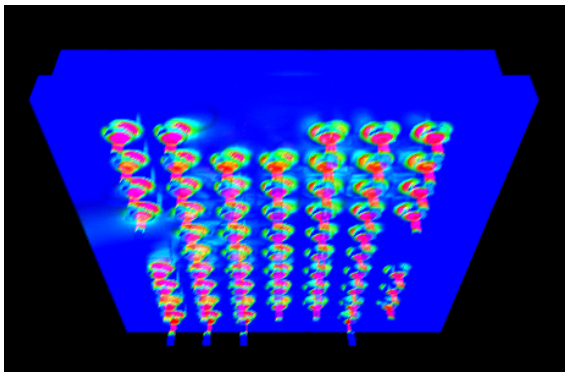


**Figure 9:** *Parallel volume rendering image to show the concentration distribution for an underground water simulation dataset with about 10 million nodes and 100 timesteps. On SR8000 with 8 SMP nodes, for more than 10 million vertices and 100 timesteps, without dynamic load repartition it took about 2.35 seconds for one time-step on average. After dynamic load reparation, it just took about 1.56 seconds on average for one time-step.*



*(a) Top view*



*(b) Bottom view*

**Figure 7:** *Parallel volume rendering images to show the equivalent scalar value of stress by the linear elastostatic analysis for a PGA dataset with 7,869,771 vertices and 7,649,024 grid elements.*