Development of In-Situ Visualization Tool for PIC Simulation*)

Nobuaki Ohno and Hiroaki OHTANI[1,2)]

Graduate School of Simulation Studies, University of Hyogo, Kobe 650-0047, Japan
[1)]*Department of Helical Plasma Research, National Institute for Fusion Science, Toki 509-5292, Japan*
[2)]*Department of Fusion Science, The Graduate University for Advanced Studies (SOKENDAI), Toki 509-5292, Japan*

(Received 10 December 2013 / Accepted 17 March 2014)

As the capability of a supercomputer is improved, the sizes of simulation and its output data also become larger and larger. Visualization is usually carried out on a researcher's PC with interactive visualization software after performing the computer simulation. However, the data size is becoming too large to do it currently. A promising answer is in-situ visualization. For this case a simulation code is coupled with the visualization code and visualization is performed with the simulation on the same supercomputer. We developed an in-situ visualization tool for particle-in-cell (PIC) simulation and it is provided as a Fortran's module. We coupled it with a PIC simulation code and tested the coupled code on Plasma Simulator supercomputer, and ensured that it works.

Keywords: in-situ visualization, PIC, simulation, scalar field visualization, vector field visualization

DOI: 10.1585/pfr.9.3401071

1. Introduction

As the capabilities of supercomputers are improved, the sizes of computer simulations get larger and larger. In general, visualization is carried out with a researcher's workstation or PC with interactive visualization software after performing the computer simulations. In short, visualization is done as a post-processing. For this case the researchers have to store the simulation results, the raw data, and transfer them from the storages of supercomputers to their workstations or PCs via network. However, it usually needs large space of storages for store and takes a long time for disk I/O and data transfer, if the data size is large. Even if the PC has superb graphics hardware, enough memories, sufficient CPU power and big HDD space, it is impossible to visualize the large-scale data without downsizing the data or extracting their regions of interest from them. There are two other kinds of visualization if classified according to the time when visualization is performed, i.e. visualization as co-processing and in-situ processing. For co-processing, visualization program runs simultaneously with the simulation code on the same computer or other powerful computer. The data is transferred via network or disk. For in-situ processing, visualization codes are embedded in simulation codes, and visualization is carried out in tandem with simulation on the same supercomputers, accessing the data arrays directly. This means that computers for visualization surely have enough CPU power and memory size. In the latter two cases, because the coupled simulation codes or co-processing visualization programs output image files instead of raw data, it is image files to be stored and transferred to researchers. The sizes of image files are generally much smaller than those of raw data. Therefore, the file transfer is not time-consuming. Thus visualization as a co-processing and in-situ processing are considered to be the answers to the problems of visualization and are paid attention to nowadays. The research of these fields have been done for many years, and supercomputer venders have developed in-situ visualization tools (for example, see Ref. [1]). There is even a research using immersive virtual reality system [2]. Yu et al reports in-situ visualization of petascale combustion simulation [3]. Recently there are many researches [4–7] using ParaView and VisIt as visualization engines. ParaView [8] and VisIt [9] are two famous free visualization software and both of them are based on VTK. ParaView and VisIt are basically distributed parallel software (Manta plug-in, a rendering engine of ray tracing, is multi-threaded).

We believe that in-situ processing is more promising than co-processing because visualization code can directly access the data. However, there are problems too. For example, interactive visualization is impossible because visualization is performed as a batch job. A tactic for this problem is reported in Ref. [10]. We have developed a basic and handy in-situ visualization tool, VISMO (VISualization MOdule), for PIC (particle-in-cell) [11] codes which are parallelized based on the domain decomposition method using MPI such as PASMO [12]. Our tool is parallelized by MPI/OpenMP hybrid scheme. This feature of parallelism is suitable for today's supercomputers with hundreds of thousands of CPU cores. The purpose of this development is not only visualizing the simulation data in-situ but also studying in-situ visualization itself.

In the Sec.2, a design of VISMO is introduced. We

author's e-mail: ohno@sim.u-hyogo.ac.jp
*) This article is based on the presentation at the 23rd International Toki Conference (ITC23).

have coupled this tool with two programs and tested them on a supercomputer. Sample images are shown in Sec.3. Finally, a summary is presented in Sec.4.

2. Basic Design

The basic design of VISMO is described in this section. The PIC codes usually have data of particles, scalar and vector fields. This tool provides visualization methods for the three kinds of data. It can be used for MHD simulations as well if they are parallelized in the same way because the incorporated visualization methods for scalar and vector fields are also suitable to visualization of them. This tool does not require graphics hardware in order to be used on supercomputers. And it is provided as a Fortran's module. We chose Fortran as the programming language because many simulation codes are written in it and the researchers can use this module by only adding Fortran's statements in their simulation codes. We believe that it is important for this kind of tools not to be complex to use and to be user-friendly, so that it is designed not to require them to change their simulation codes greatly.

2.1 Target computers

The target computer of this tool is a distributed memory parallel computer with no graphics hardware such as Plasma Simulator at NIFS (National Institute for Fusion Science) in Japan. This is an ordinal environment of supercomputers and this condition does not restrict the use on shared memory systems. Besides, this tool does not require any special libraries except MPI library (and Fortran Compiler, of course) so that it may be used with a little or no modification on various supercomputers and PC clusters.

2.2 Visualization methods

PIC simulations produce three kinds of data, i.e. particle, scalar field and vector fields. The general visualization methods for those kinds of data are incorporated into VISMO such as "spheres" for particles, isosurfaces, slices and volume rendering for scalar fields, tubed streamlines and "arrows" for vector fields. All the visualization methods are implemented by software rendering because graphics hardware may not be available on the supercomputer. Tubed streamline is drawn as a collection of spheres and cylinders, and an arrow is drawn as a collection of cylinder and cone. Spheres, cylinders and cones are drawn by ray casting methods using their equations like a ray tracing software POV-Ray [13]. This means that unlike much visualization software, three-dimensional objects are not divided into many polygons, say triangles. This ensures the output images are more beautiful when objects are near viewing points.

2.2.1 Spheres

"Spheres" is a visualization method for particle data. This method displays spheres where there are particles such as ions and electrons. However, displaying all the particles is not a good strategy because current PIC simulations handle billions of particles and the image gives no insight. The visualization of such huge number of particles is a future work.

2.2.2 Isosurface, slices and volume rendering

Volume rendering [14] and all the other scalar visualization methods are implemented by ray casting method. On isosurface visualization, it is judged if the ray goes across the isosurface by comparing the two values of scalar data on the two sequential sampling points. For slices, the positions of the two sequential sampling points are compared to judge if the ray goes across the slices. Thanks to drawing by ray casting, semi-transparent isosufaces and slices can be drawn. The ray casting is enhanced by the traditional ways, such as early ray termination [15] and empty space skipping.

2.2.3 Streamlines and arrows

Streamlines are drawn based on the 4-th order Runge-Kutta integration. At the cost of MPI synchronization, streamline can extend through multiple subdomains. "Arrows" are stationed on a plane specified by users. The density of arrows can be also specified.

2.3 Camera and projection

Multiple cameras, i.e. viewing points, are supported so that images drawn from various viewing points with the same visualization parameters can be produced. Both perspective and parallel projection are supported.

2.4 Interface

VISMO is provided as a Fortran's module and used by embedding it into simulation codes. Users need to add Fortran statements of VISMO in their simulation codes. This tool requires all the MPI process to know the global coordinates information in addition to its local coordinates. This may lead to a little modification to users' codes.

Visualization parameters such as isosurface levels, camera, light and so on are specified in text files, not in the simulation codes. By this feature, when a user wants to try other visualizations, he/she need not modify and compile the simulation codes again. He/she only needs to prepare another text files in which new visualization parameters are specified.

2.5 Parallelization

VISMO is parallelized based on MPI/OpenMP hybrid scheme. The target simulation code is parallelized PIC codes based on the domain decomposition method by MPI. If the MPI processes of PIC code are parallelized by OpenMP, this tool also works as a hybrid code. Each MPI process has its own local data. They visualize their local

data using visualization methods parallelized by OpenMP and render a subimage. Then all the images are composited into a final image. Direct send method [16], which can be used on an arbitrary number of rendering processes, is employed for the image composition. Finally the master process of MPI outputs the final image on the storage.

3. Test Runs

We tested VISMO coupled with two programs on Plasma Simulator (Hitachi SR16000 Model M1, POWER7). They worked and outputted visualization images successfully. Sample images are shown in the following subsections. In addition, we measured the additional costs for and caused by the coupling.

3.1 Data visualization

We prepared a program which made particle, scalar and vector data in memory using Fortran's mathematical functions. It is parallelized based on MPI/OpenMP hybrid scheme. We coupled VISMO with this program and tested it on Plasma Simulator to make sure if all the visualization methods worked. The number of MPI processes is 8 and that of OpenMP threads is 4. The domain $(101 \times 81 \times 121)$ is divided into 8 subdomains in x, y and z directions. The total number of particles is 400.

To draw Fig. 1, all the visualization methods are used. Green and brown clouds are scalar data visualized by volume rendering. The streamlines successfully crossed among the data.

3.2 PASMO with VISMO

We tested in-situ visualization by coupling PASMO with VISMO. PASMO is parallelized by MPI and compiler's auto-parallel function.

3.2.1 Small scale of PASMO

The domain $(130 \times 129 \times 34)$ is divided into 4 subdomains in the z direction. The number of MPI processes is 4 and that of threads is 16. This tool also uses the same numbers of MPI processes and OpenMP threads as the simulation code. Figure 2 is an image drawn by the coupled simulation code. To visualize data, isosrface, slice and arrows are used.

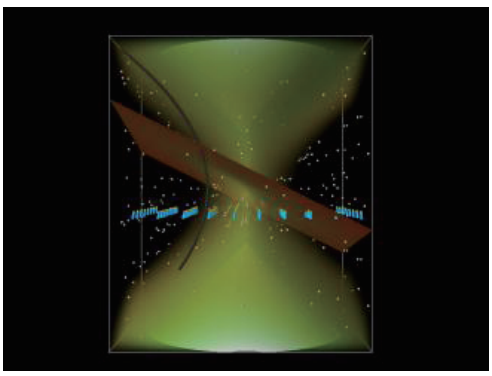

Fig. 1 Volume rendering, slice, isosurface, arrows, streamlines and spheres are used to visualize the data.

3.2.2 Middle scale of PASMO

The domain $(514 \times 257 \times 130)$ is divided into 64 subdomains in the z direction. The number of MPI processes is 64 and that of threads is 32. Totally, 2,048 CPU cores were used in both the simulation and visualization. The number of ions and electrons is about 650 million each during this test run. Figures 3 (a), (b) and (c) are images drawn by the coupled simulation code.

3.3 Costs for VISMO

To couple PASMO with VISMO, we had to add Fortran statements in the code. Totally, 24 lines were added in the PASMO code including declarations of variables and some computation for coordinates settings, and the target attribute were given to the data arrays to be visualized. This may not be heavy burden for simulation researchers. Other than this, users have to prepare text files specifying visualization parameters.

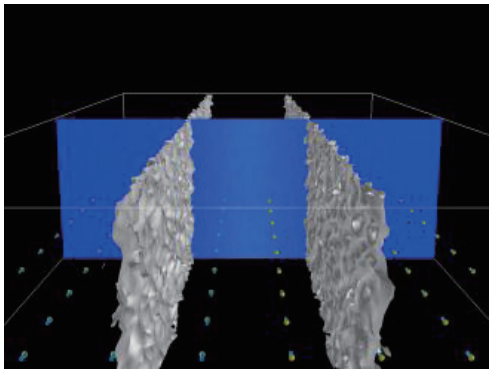

Fig. 2 This picture was drawn by small scale of PASMO, a PIC simulation code, coupled with VISMO.

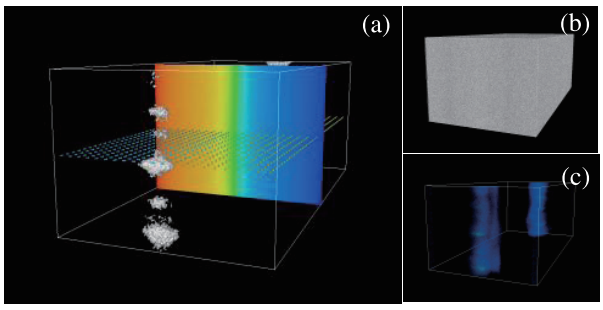

Fig. 3 (a) color slice, isosurface and arrows are used to visualize x-component of magnetic field, electron density and magnetic field respectively. (b) All the electrons are visualized by spheres but this picture gives no insight. (c) Electron density is visualized by volume rendering.

To measure the computational cost of VISMO, we set the coupled middle scale of PASMO described in Sec.3.2.2 to perform visualization once in 5 steps. We carried out 10 steps computation, which did visualization twice, on Plasma Simulator 10 times. The coupled code output an image with the same visualization parameters as Fig. 3 (a) when performing visualization. The resolution of the images is $1{,}024 \times 768$ pixels. The HITACHI's profiler revealed that the average ratio of CPU time spent for visualization was about 38.7%. The required CPU time is about 1.63 times larger than the original PASMO code. For the case that the coupled code is set to output 3 images with the same parameters as Figs. 3 (a-c) for a visualization, the average ratio of CPU time spent for visualization was about 64.2%. The required CPU time is about 2.79 times larger. When VISMO is set to output large number of images or high resolution images, the CPU time needed for VISMO surely becomes large. Those ratios are also influenced by the condition of the system. The researchers have to strike a right balance between simulation and visualization.

4. Summary

We developed an in-situ visualization tool VISMO for PIC simulation. It is provided as a Fortran's module and general visualization methods for particle, scalar and vector data are incorporated. In-situ visualization can be made use of by embedding this tool into simulation codes. By this tool, researchers can investigate the phenomena without moving raw data from the supercomputers' storage to theirs, or storing large raw data in their storages. In addition, this tool can be used to visualize MHD simulations too. This must enhance simulation researches about plasmas and there must be lots of applications in other research fields.

5. Acknowledgements

This work was performed with the support and under the auspices of the National Institute for Fusion Science (NIFS) Collaborative Research Program (NIFS12KNSS027).

[1] A. Ogasa, H. Maesaka, K. Sakamoto and S. Otagiri, FUJITSU Sci. Tech. J. **48**, No.3, 348 (2012).
[2] Y. Tamura, H. Nakamura and N. Ohno, IEEJ Transactions on Electronics, Information and Systems **126-C**, 401 (2006) (in Japanese).
[3] H. Yu, C. Wang, R.W. Grout, J.H. Chen and K. Ma, Technical Report CSE-2009-9, University of California at Davis (2009).
[4] J. Soumagne, J. Biddiscombe and J. Clarke, 5th International SPHERIC Workshop, B. Rogers (Ed.) (2010) pp.186-193.
[5] N. Fabian, K. Moreland, D. Thompson, A.C. Bauer, P. Marion, B. Geveci, M. Rasquin and K.E. Jansen, IEEE Symposium on Large-Scale Data Analysis and Visualization (2011) pp.89-96.
[6] M. Rivi, L. Calori, G. Muscianisi and V. Slavnic, PRACE, whitepaper (2012) pp.1-18.
[7] S. Benjaminsson, D. Silverstein, P. Herman, P. Melis, V. Slavnic, M. Spasojevic, K. Alexiev and A. Lansner, PRACE, whitepaper (2012) pp.1-20.
[8] http://paraview.org/
[9] https://wci.llnl.gov/codes/visit/home.html
[10] A. Kageyama and T. Yamada, Comput. Phys. Commun. **185**, 79 (2014).
[11] C.K. Birdsall and A.B. Langdon, *Plasma Physics Via Computer Simulation* (McGraw-Hill, New York, 1985).
[12] H. Ohtani and R. Horiuchi, Plasma Fusion Res. **4**, 024 (2009).
[13] http://www.povray.org/
[14] R.A. Drebin, L. Carpenter and P. Hanrahan, Comput. Graph. **22**, 65 (1988).
[15] M. Levoy, ACM Trans. Graph. **9**, 245 (1990).
[16] W.M. Hsu, Proc. IEEE Symposium on Parallel Rendering (1993) pp.7-14.