In-Situ Visualization with Membrane Layer for Movie-based Visualization

Kohei Yamamoto and Akira Kageyama[0000−0003−0433−668X]

Department of Computational Science, Kobe University, Kobe 657-8501, Japan

Abstract. We propose a movie-based visualization for High Performance Computing (HPC) visualization. In this method, a viewer interactively explores a movie database with a specially designed application program called a movie data browser. The database is a collection of movie files that are tied with the spatial coordinates of their viewpoints. One can walk through the simulation's data space by extracting a sequence of image files from the database with the browser. In this method, it is important to scatter as many viewpoints as possible for smooth display. Since proposing the movie-based visualization method, we have been developing some critical tools for it. In this paper, we report the latest development for supercomputers to apply many in-situ visualizations with different viewpoints in a Multiple Program Multiple Data (MPMD) framework. A key feature of this framework is to place a membrane-like layer between the simulation program and the visualization program. Hidden behind the membrane layer, the simulation program is not affected by the visualization program even if the number of scattered viewpoints is large.

Keywords: Visualization · Movie-based visualization · HPC.

1 Introduction

The imbalance between processor speed and network bandwidth in High Performance Computing (HPC) systems leads in-situ visualization to provide hope for the future [11, 12, 10, 5, 4]. In general, in-situ visualization deprives users of interactive control of visualization parameters. We cannot change the visualization-related variables after a simulation, unless we adopt special methods for rendering [15, 7]. These visualization-related variables include the applied visualization algorithms, their parameters, and the camera settings. It is not reasonable to resubmit a simulation job when, for example, we just want to observe a phenomenon from a different view position.

We have proposed a way to realize an interactive viewing of in-situ visualization that can be applied with standard rendering [6]. The key point of this method is to apply multiple in-situ visualizations from different viewpoints.As in the bullet-time method used in the movie industry, a number of visualization cameras are placed in the simulation space. Each camera takes a sequence of visualization images as the simulation goes on. In contrast to the steering

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

approach [9, 2], the positions of the cameras are predefined before the simulation. After the simulation, images from each camera are combined into a movie file. Therefore, the output of this in-situ visualization method is a collection of movie files, rather than numerical data files. Labeled with the cameras' locations, the collection of movie files composes a movie database. Here we refer to this approach to HPC visualization as "movie-based visualization", since it is a generalization of image-based visualization [1].

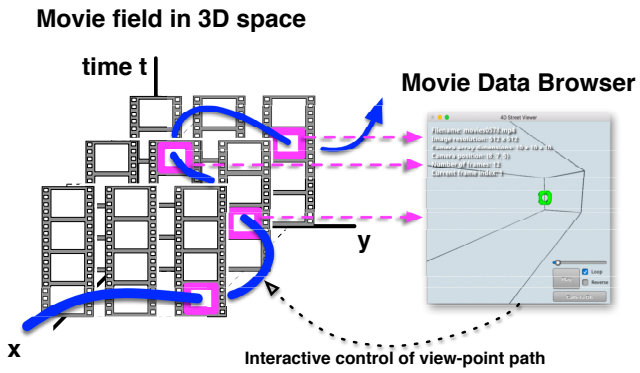

Fig. 1. Interactive exploration of movie database.

After we proposed the idea of movie-based visualization, we developed a specially designed PC application that extracts a sequence of still images from the movie database and shows the sequence as an animation in a PC window. The application, the movie data browser, enables its user to interactively change the viewpoint by dragging the PC's mouse or by keyboard input while a movie is playing forward or backward in time; see Fig. 1.

A key point of this method is to place multiple cameras as densely as possible in the simulation space. In contrast to the bullet-time method, our visualization cameras have no mutual occlusion, regardless of how dense they are, but applying many in-situ visualizations for a simulation is a challenging task.

2 Membrane Layer Method

Major (post hoc) visualization applications are now provided with in-situ libraries or tools, such as Catalyst [3] for ParaView and libsim [18] for VisIt. Various in-situ visualization infrastructures have also been developed for HPC, such as Embree [17] and OSPray [16]. More general framework for high-performance I/O middleware ADIOS [8] is also used in in-situ visualizations. We are also

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

developing an in-situ visualization tool that is based on KVS [14], a general-purpose visualization software tool. We use the off-screen rendering mode of KVS for visualizations on CPU cores of supercomputer systems.

In order to make the movie-based visualization method a practical tool for simulations, it is necessary to avoid the speed deterioration of the simulation by the in-situ generation of many movie files. The key idea proposed in this paper is to separate the simulation from the visualization by placing a semipermeable membrane-like layer between them. The two programs are "invisible" to each other.

Both the simulation and the visualization are parallelized. They are basically independent MPI programs with their own `MPI_Init` and `MPI_Finalize` functions. The semipermeable membrane between them is also an independent parallel program. The simulation program sends data to the membrane, and the visualization program receives them from the membrane under a Multiple Program Multiple Data (MPMD) framework.

In fact, the membrane is composed of two independent MPI programs. They correspond to the two sides of a plane; one is the front face, or simulation side, and the other is the back face, or visualization side. Therefore, we have four independent MPI programs in total. When one of the four programs stops because of an error, a signal is sent to the other programs to finalize the whole job. The numerical data flow is basically one way; it is sent from the simulation to the membrane, then to the visualization. The error signal of the visualization program is the only signal that is sent backward, from the visualization to the simulation. That is the reason why we call it a semipermeable membrane.

Owing to the MPMD framework, we can apply multiple in-situ visualizations in an asynchronous way. In contrast to the synchronous cases,we apply multiple in-situ visualizations on other processor nodes of the supercomputer system while the simulation is running.For a larger computational load for the visualization, or for a larger number of viewpoints of the in-situ visualization, we just allocate more computer nodes for additional MPI processes for the visualization program.

Since the membrane's front face is devoted only to receiving data from the simulation, the simulation program can assume that each data transfer is completed without delay. The simulation program does not wait for completion of the visualization for each item of data.

The membrane's back face stores the latest simulation data and passes them to the visualization program when they are requested. If new data are sent from the simulation before completing the visualization of previous data, the stored data in the membrane is overwritten. In other words, it is possible that some image frames are missing in the final product of the in-situ visualization movies when it takes an unusually long time to render an image. We accept it as unavoidable in this framework at present but it could be improved in the future.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

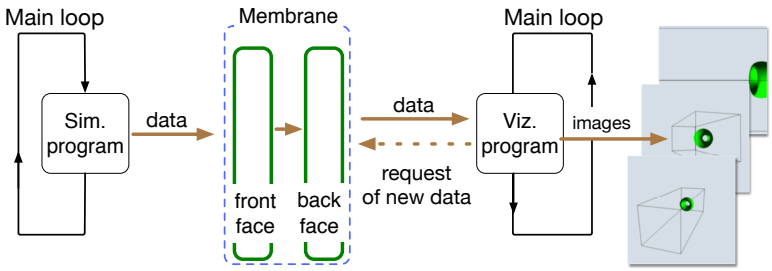

Fig. 2. The membrane layer between the simulation and visualization programs.

3 Experiments

To demonstrate the applicability of the membrane method to in-situ visualization of practical simulations on supercomputers, we have performed a simple computational fluid dynamics simulation on a supercomputer system, Oakforest-PACS (Fujitsu PRIMERGY CX600 M1, 8208 nodes of Intel Xeon Phi) for the well-known smoke-ring formation [13]. The simulation region is a rectangular box of size $L_x \times L_y \times L_z = 30$ m $\times$ 10 m $\times$ 10 m. Periodic boundary conditions are assumed in all (x, y, and z) directions. Compressible Navier-Stokes equations for an ideal gas are solved with a second-order central finite difference method for the spatial derivatives and an explicit fourth-order Runge-Kutta method for time integration. The code is parallelized based on a three-dimensional decomposition with 24 ($= 6 \times 2 \times 2$) domains. An MPI process is assigned to each domain.

In this experiment, we visualize three scalar fields, pressure p, mass density ρ, and enstrophy density $q = |\nabla \times \boldsymbol{v}|^2$, where $\boldsymbol{v}$ is the flow velocity. The front face of the membrane consists of three (same as the number of fields) MPI processes, and the back face of it has the same number of MPI processes. A process of the front face receives one of the scalar fields from the simulation and its counterpart process of the back face passes the scalar data to the visualization program on request. As for the visualization program, we allocate 27 cameras in this experiment, with one MPI process for each camera. In total, we invoke 57 MPI processes (57 processor nodes): 24 (for simulation) + 3 (for front face of the membrane) + 3 (for back face of the membrane) + 27 (for visualization).

Fig. 3 shows visualizations by an isosurface of q by four cameras out of 27 cameras in total. Other visualizations for different physical variables lead to basically the same images. The resulting movie files have sufficiently high temporal resolution (high number of frames per second (FPS)) to analyze the fluid dynamics in detail.

We next confirm the effect of the membrane as a barrier against increased load for the visualization program. We compare two different styles of in-situ visualization: one with the membrane-based asynchronous visualization method

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

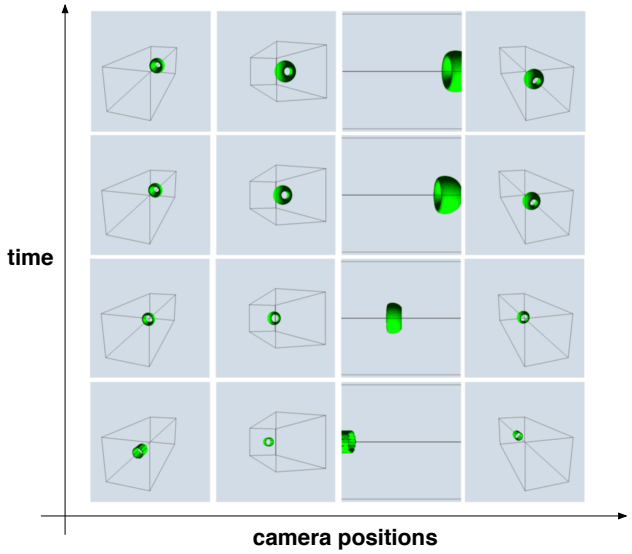

Fig. 3. Snapshot sequences of the propagation of a vortex ring, visualized from different viewpoints. It shows four of the 27 viewpoints. The vortex ring is visualized by an isosurface of the enstrophy density.

and the other a fully synchronous visualization (i.e., in-situ visualization embedded into the simulation code).

We perform asynchronous and synchronous in-situ visualizations on the same supercomputer system with the same simulation set-ups and physical parameters as in the previous experiment. We measure execution time for the simulations with different numbers of in-situ visualization cameras. As in the previous experiment, we solve the vortex ring formation with 24 MPI processes, applying in-situ visualizations every 100 simulation loops.

The numbers of cameras considered are 8, 16, 32, 64, and 128. We compare the elapsed time in each case for $10,000$ simulation steps. In the embedded synchronous in-situ case, the total number of MPI processes is fixed, while in the membrane layer method, the number of MPI processes varies.

The result of the comparison is shown in Fig. 4. The elapsed time presented in this figure is defined as the average of five runs. This figure shows that the membrane method is always faster than the synchronous method and, more importantly, it keeps the execution time for the simulation constant even if we increase the number of in-situ visualization cameras.

4 Summary

We have proposed a movie-based visualization method for HPC [6]. In this method, the viewer interactively explores a database of movie files, rather than a

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

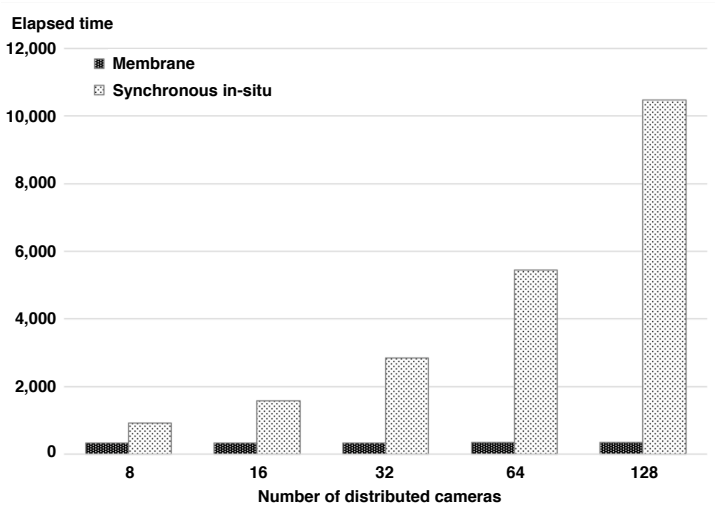

Fig. 4. Elapsed time for the vortex ring simulation with and without the membrane layer.

numerical dataset, as in a standard post hoc visualization. The movie database is a collection of movie files that are labeled with the spatial coordinates of the viewpoint of each movie.

In this paper, we have proposed a framework for in-situ visualization on HPC systems that enables simulation researchers to apply a large number of in-situ visualizations with different viewpoints without slowing down the simulation. The key idea is to place a semipermeable membrane layer between the simulation and visualization programs. The membrane is implemented by two MPI programs that correspond to two faces, the front face and the back face of the membrane.

To confirm the effect of the membrane layer, we have compared two in-situ visualizations, with and without the membrane, for a computational fluid dynamics simulation. Without the membrane, the execution time grows as the number of cameras increases. In contrast, with the membrane, the execution time stays almost constant. This means that the increased computational load for a larger number of cameras is safely absorbed by an increased number of MPI processes for visualization, without affecting the simulation program. The membrane method described in this paper could be regarded as a promising approach to parallel in-situ visualizations, not only for movie-based visualization, but also to in-situ visualizations in HPC in general.

References

1. Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D.H., Petersen, M.: An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55

International Conference for High Performance Computing, Networking, Storage and Analysis, SC, 424–434 (2014)
2. Atanasov, A., Bungartz, H.j., Mehl, M., Mundani, R.p., Rank, E., Treeck, C.V.: Computational Steering of Complex Flow Simulations. In: High Performance Computing in Science and Engineering, Garching/Munich 2009. pp. 63–74 (2009)
3. Ayachit, U., Bauer, A., Geveci, B., O'Leary, P., Moreland, K., Fabian, N., Mauldin, J.: ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. Proceedings of ISAV2015 pp. 25–29 (2015)
4. Bennett, J.C., Childs, H., Garth, C., Hentschel, B.: In Situ Visualization for Computational Science. Dagstuhl Reports **8**(07), 1–43 (2018)
5. Bethel, E.W., Childs, H., Hansen, C.: High performance visualization : enabling extreme-scale scientific insight. CRC Press (2013)
6. Kageyama, A., Yamada, T.: An approach to exascale visualization: Interactive viewing of in-situ visualization. Computer Physics Communications **185**, 79–85 (2014)
7. Kawamura, T., Noda, T., Idomura, Y.: In-situ visual exploration of multivariate volume data based on particle based volume rendering. Proceedings of ISAV 2016 **D**, 18–22 (2017)
8. Liu, Q., Logan, J., Tian, Y., Abbasi, H., Podhorszki, N., Choi, J.Y., Klasky, S., Tchoua, R., Lofstead, J., Oldfield, R., Parashar, M., Samatova, N., Schwan, K., Shoshani, A., Wolf, M., Wu, K., Yu, W.: Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks. Concurrency and Computation: Practice and Experience **26**, 1453–1473 (2014)
9. Long, L.N., Plassmann, P.E., Sezer-uzol, N., Jindal, S.: Real-time visualization and steering of large-scale parallel simulations. In: 11TH INTERNATIONAL SYMPOSIUM ON FLOW VISUALIZATION. pp. 1–12 (2004)
10. Ma, K.L.: In Situ Visualization at Extreme Scale: Challenges and Opportunities. IEEE Computer Graphics And Applications pp. 14–19 (2009)
11. Ma, K.L., Wang, C., Yu, H., Tikhonova, A.: In-situ processing and visualization for ultrascale simulations. Journal of Physics: Conference Series **78**(1), 1–10 (2007)
12. Ross, R.B., Peterka, T., Shen, H.W., Hong, Y., Ma, K.L., Yu, H., Moreland, K.: Visualization and parallel I/O at extreme scale. Journal of Physics: Conference Series **125** (2008)
13. Saffman, P.G.: Vortex dynamics. Cambridge University Press (1992)
14. Sakamoto, N., Koyamada, K.: KVS: A simple and effective framework for scientific visualization. J. Adv. Simulation. Sci. Eng. **2**(1), 76–95 (2015)
15. Tikhonova, A., Correa, C.D., Kwan-Liu, M.: Explorable images for visualizing volume data. IEEE Pacific Visualization Symposium 2010, PacificVis 2010 - Proceedings **D**(VIDi), 177–184 (2010)
16. Wald, I., Johnson, G.P., Amstutz, J., Brownlee, C., Knoll, A., Jeffers, J., Günther, J., Navratil, P.: OSPRay - A CPU Ray Tracing Framework for Scientific Visualization. IEEE Transactions on Visualization and Computer Graphics **23**(1), 931–940 (2017)
17. Wald, I., Woop, S., Benthin, C., Johnson, G.S., Ernst, M.: Embree: A Kernel Framework for Efficient CPU Ray Tracing. Acm Transactions on Graphics **33**(4), 8 (2014)
18. Whitlock, B., Favre M, J., Meredith S, J.: Parallel in situ coupling of simulation with a fully featured visualization system. Eurographics Symposium on Parallel Graphics and Visualization pp. 101–109 (2011)

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_55