A Memory Efficient Image Composition-based Parallel Particle Based Volume Rendering

Yamaoka, Yoshiaki
Hayashi, Kengo
Sakamoto, Naohisa
Nonaka, Jorji

J. Adv. Simulat. Sci. Eng.
Vol. 6, No. 1, 1-10.

A Memory Efficient Image Composition-based Parallel Particle Based Volume Rendering

Yoshiaki Yamaoka[1*], Kengo Hayashi[1], Naohisa Sakamoto[1], Jorji Nonaka[2]

[1]Graduate School of System Informatics, Kobe University, Kobe, Japan
[2]RIKEN Center for Coputational Science, Kobe, Japan

*180x220x@stu.kobe-u.ac.jp

Received: October 22, 2018; Accepted: December 16, 2018; Published: March 15, 2019

Abstract. Large-scale simulations have widely been conducted on modern High-Performance Computing (HPC) systems in a variety of scientific and engineering fields, and scientific visualization has been a popular approach for analyzing and extracting meaningful information from the simulation results. In this work, we focused on Particle Based Volume Rendering (PBVR) method because of its proven effectiveness for handling non-trivial unstructured volume data, which is still commonly used on numerical simulations in the engineering fields. PBVR possesses a visibility sorting-free characteristics thanks to its use of small and opaque particles as the rendering primitives. However, there is a memory cost and image quality trade-off because of the necessity of storing the entire sets of generated particle data before starting the rendering process. In this paper, we present a memory cost efficient parallel PBVR approach for enabling high-quality and high-resolution PBVR of large-scale numerical simulation results. For this purpose, we focused on the image data gathering and processing instead of traditional particle data gathering and processing by using the sort-last parallel image composition approach. We evaluated its effectiveness on the K computer by using the Binary-Swap-based 234Compositor library, and verified its potential for reducing the memory cost while generating high-quality and high-resolution image data.

Keywords: Particle Based Volume Rendering, Sort-Last Image Composition, High Performance Computing

1. Introduction

Direct volume rendering has widely been considered as one of the most powerful visualization technique since it enables the analysis and exploration of the entire volume data by appropriately setting the color and opacity parameters via transfer function. However, volume rendering of unstructured volume data is non-trivial because of the time-consuming visibility sorting process along the viewing ray direction. The continuous increase in the simulation size has proportionately aggravated the problem, and this becomes more serious in the case of complex geometry data composed of heterogeneous cell elements such as the one utilized in this work. We utilized a Computational Fluid Dynamics (CFD) simulation result composed of tetrahedral, hexahedral, and prismatic cells, and it is worth mentioning that this kind of heterogeneous geometric cells is also commonly used on Computational Structure Mechanics (CSM) simulations.

Particle Based Volume Rendering (PBVR) [1] has attracted considerable attention as an efficient and effective visualization method for visualizing this kind of unstructured volume data. PBVR uses the Sabella's particle emission model [2], where a given volume data is represented as a set of small and opaque self-light-emitting particles. These particles are generated by estimating the particle density inside the cell elements taking into consideration the user defined transfer function. These particles are then rendered by projecting onto the image plane, and the most representative particles are selected via depth comparison. This per-particle depth comparison can be executed in any order, thus the traditional visibility sorting in the viewing ray direction becomes no more necessary. Although PBVR has been applied to interactive visualization [3, 4], there is a memory cost and image quality trade-off because of the necessity of storing the entire sets of generated particle data before starting the rendering process. In addition, the problem is aggravated on tightly-coupled in-situ visualization where the simulation and visualization processes share the same memory resource.

In this paper, we present a memory cost efficient parallel PBVR approach for enabling high-quality and high-resolution PBVR of large-scale unstructured volume data. For this purpose, we focused on the image data gathering and processing instead of the traditional particle data gathering and processing by using the sort-last parallel image composition approach. In this method, each rendering node generates a partial image from the assigned sub-volume, and these partial images are then gathered and merged into one image, and at the end these generated set of images are ensemble averaged to produce the final image. In this work, we utilized the Binary-Swap based 234Compositor [5], and verified its potential for reducing the memory cost while generating high-quality and high-resolution image data.

2. Method

As shown in Fig. 1, the proposed image composition based parallel PBVR [1] consists of the following four main stages: Particle generation stage; Particle projection stage; Image Composition stage; and Ensemble averaging stage. The "Rank" in the Fig.1 represents each of the MPI (Message Passing Interface) processes, and after their independent data loading the main three stages are repeated until a pre-defined number of repetitions. The particle projection stage produces a set of partial rendered images, which are gathered and processed by the "Image Composition" stage. Therefore, the generated particles become useless right after the projection stage, and they can be eliminated to release the memory space for storing the new set of generated particles. The ensemble averaging stage will progressively generate the image with continuous quality refinements. These main stages will be described in the next sub-sections.

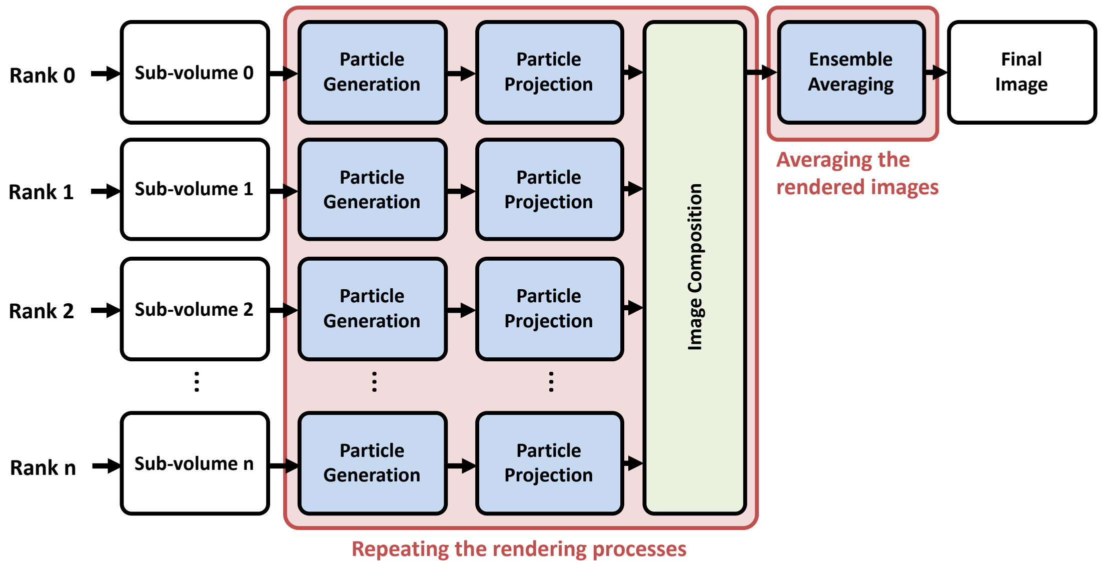

Figure 1: Overview of the proposed parallel particle-based volume rendering.

2.1. Particle Generation

In the proposed method, each MPI process will be responsible for processing a portion (sub-volume) of the entire target volume data. After the data loading, each process will generate the particle datasets corresponding to the loaded sub-volumes. This particle generation process is governed by (1), where the particle density ρ represents the number of particles within a unit volume, which corresponds to a single cell in the unstructured volume data.

$$\rho = \frac{-\log(1-\alpha)}{\pi r^2 \Delta t} \qquad (1)$$

In this equation, the variable α represents the opacity value given by the user specified transfer function; r represents the radius of the particle; and Δt represents the specified integration interval length during the calculating of the Sabella's brightness equation [2]. Since the particle generation process occurs in a cell-by-cell fashion, and since there is no process dependency between the cells, we can also apply intra-node parallelization for accelerating the particle generation process. In this work, we utilized the OpenMP multi-threading in our implementation. The number of generated particles for each of the sub-volumes can be calculated by multiplying the volume of the grids by the particle density ρ. There is also a new particle generation approach, where the number of particles can be controlled via particle size adjustment technique [4].

2.2. Particle Projection and Image Composition

In the already exisitng parallel PBVR method [4], the entire set of generated particles from the sub-volumes were gathered to the master MPI process (Rank 0) node before the particle projection stage. However, this particle gathering and processing approach has suffered from high mermoy consumption problem since there is a need to store the entire particle data sets, corresponding to the pre-defined number of repetitions, before starting the particle projection stage. It is also worth noting that the particle data size can become large depending on the transfer function defined by the user, and in addition since the image quality is directly related to the number of repetitions, thus larger memory will be required for higher quality renderings. To avoid this high pressure on the memory consumption, we propose the application of sort-last rendering approach, by gathering the image data instead of the particle data onto the master MPI process node.

In the proposed approach, a parallel image compositing library is used to gather the set of resulting images, from the particle projection stage, which are composed of color and depth information. Since the representative particle selection processing is done by comparing their distance (depth) from the viewing point and selecting the closest particles, we can use any parallel image composition library which possesses per-pixel depth comparison functionality. In this work, we utilized the 234Compositor [5] that was proven to run on the SPARC64 fx CPU environment such as the K computer. Although the order independent characteristics of the PBVR gives us the flexibility for executing the sub-image gathering in an asynchronous manner, the 234Compositor requires the presence of the entire set of sub-images for starting the image composition process. During the image composition stage the sub-images are gathered and merged by taking into consideration the depth information of the pixels, and at the end, only the closest color information will remain in the generated merged image (Fig. 2).

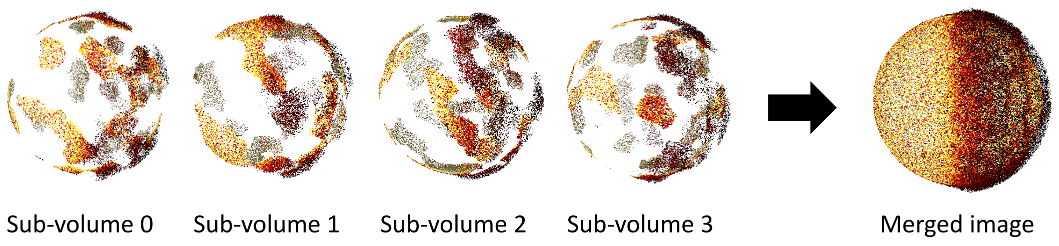

Figure 2: Example of a set of rendered sub-images, and the resulting merged image.

2.3. Ensemble Averaging

The merged image shown in Fig. 2 represents a single ensemble image, and by repeating the particle generation, projection, and image composition stages as shown in the Fig. 1, a set of ensemble images, corresponding to the number of repetitions, will be obtained. In the already exisiting parallel PBVR method [4], the final image is calculated by averaging these ensemble images in the master MPI node as shown in the Fig. 3. However, to reduce the memory cost for storing the entire set of the ensemble images before starting the ensemble averaging process, we propose the use of a progressive approach for this ensemble averaging process. As shown in (2), by considering $P_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$ as a pixel value at position (i,j) on an ensemble image, the $P_{i,j}^{k} = (R_{i,j}^{k}, G_{i,j}^{k}, B_{i,j}^{k})$ will then be the pixel value obtained with the number of ensemble averaging processes of k, and as a result, the following $P_{i,j}^{k+1} = (R_{i,j}^{k+1}, G_{i,j}^{k+1}, B_{i,j}^{k+1})$ pixel value can then be progressively calculated.

$$\begin{cases} R_{i,j}^{k+1} & = & \frac{k}{k+1} R_{i,j}^{k} + \frac{1}{k+1} R_{i,j} \\ G_{i,j}^{k+1} & = & \frac{k}{k+1} G_{i,j}^{k} + \frac{1}{k+1} G_{i,j} \\ B_{i,j}^{k+1} & = & \frac{k}{k+1} B_{i,j}^{k} + \frac{1}{k+1} B_{i,j} \end{cases} \tag{2}$$

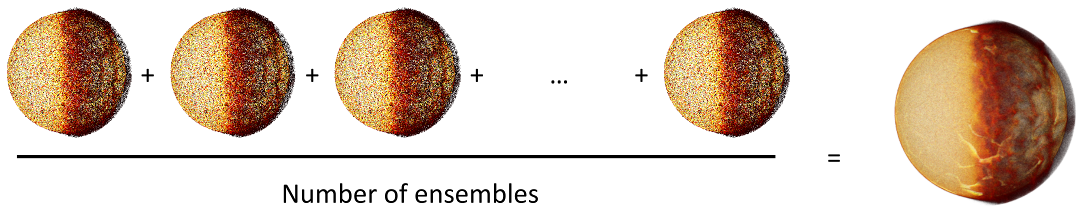

Figure 3: Example of ensemble averaging for each of the merged images.

3. Experimental Results and Discussions

In order to verify the effectiveness of the proposed parallel PBVR method, we executed some experiments for comparing the memory cost of the proposed parallel PBVR approach and the already exisiting parallel PBVR [4]. We utilized an unstructured volume data from numerical simulation results of Magnus force acting on a rotating ball placed in a uniform flow [6]. This simulation investigates the influence of the uniform flow and the rotation of the sphere around the axis perpendicular to the flow field in the surroundings and the lift force. This unstructured volume data is distributed into 256 separated files, and is composed of 18,899,767 prism elements with 15,321,546 nodes. We implemented parallel PBVR applications for both approaches, and measured the rendering performance on the K computer system, which was jointly developed by Fujitsu and RIKEN [7]. In this work, we used the OpenGL-based KVS visualization library running on top of the customized Mesa 3D llvmpipe driver for the SPARC64 fx CPU present on the K computer system [8] in order to implement the parallel PBVR applications. Figure 4 shows some visualization results obtained by using our proposed parallel PBVR method when applying different pre-defined repetitions (number of ensembles).

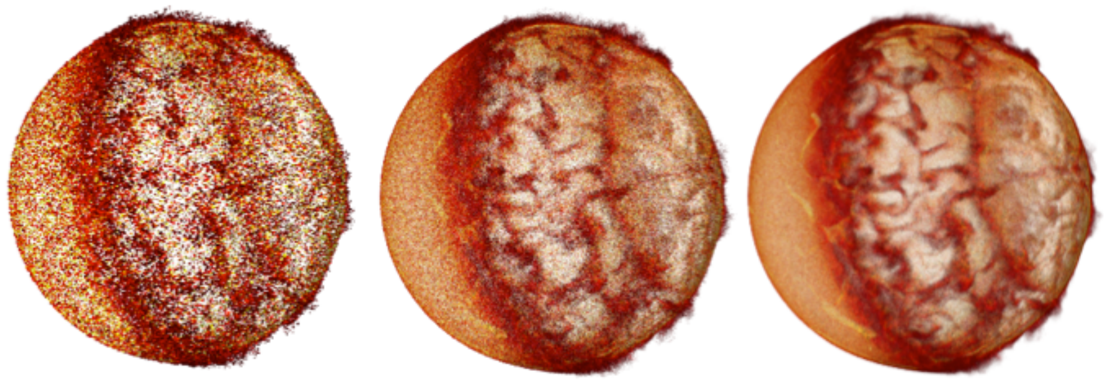

Figure 4: Rendering results with different numbers of ensumbles: 1 (left), 10 (middle), and 100 (right).

The memory cost can be estimated from the sizes of particle and image data. Each particle requires 28 bytes, which includes the coordinate position (4 bytes × 3 components = 12 bytes); the corresponding normal vector (4 bytes × 3 components = 12 bytes); and the RGBA color information represented as a scalar value of 32-bit (4 bytes). Terefore, when the number of particles is N, the memory cost of the particle data can easily be obtained by multiplying by 28, that is, by calculating 28 × N. On the other hand, each pixel requires 7 bytes, which includes the RGB color value (1 byte × 3 components = 3 bytes); and the corresponding depth value represented as a floating point data (4 bytes). Therefore, the memory cost of the entire image data will be 7 × width × height.

Table 1 shows the measured memory consumption (one image data and the entire set of particles) for the already exisiting parallel PBVR method [4] utilized for the comparison. We can easily verify that the memory consumption proportionately increases as the number of ensembles increases since the entire set of the particles is needed to be stored before starting the particle projection stage. Since the entire set of particles is initially gathered onto the master MPI node, the required memory cost is correspondent to the entire set of the generated particles, thus it becomes independent to the utilized number of MPI processes. We could also verify from the experimental results that the memory cost for the image data, when the number of ensembles is equal to 1, it represents almost 34% of the entire memory consumption. However, when the number of ensembles is 100, the image data size only represents 0.6% of the entire memory consumption.

Table 2 shows the measured memory consumption (two image data and a set of particles for the corresponding ensemble phase) for the proposed parallel PBVR method. As shown in this table, we can verify that the memory cost remains almost constant independent to the pre-defined number of ensembles (repetitions). The small variations in the memory cost for different number of ensembles can be explained by the stochastic particle generation that can produces some variations in the final number of particles even using the same sub-volume. We can also verify that the memory cost decreases as the number of MPI processes increases. In a well load-balanced scenario, the theoretical memory consumption for the particle generation and rendering can be calculated by dividing the total number of generated particles by the number of MPI processes. In this experiment, the number of generated particles becomes well load-balanced thanks to the utilized user-defined transfer function.

Regarding the memory consumption, we can verify that the reduction was around 5% to 30% when the number of ensembles is 1, but it is further reduced to around 99% when the number of ensembles becomes 100. Since the memory cost for the two image data is fixed (3.5 MB for the utilized 512x512 image resolution), the variations in the memory cost represent the variations in the paricle data size. From the aforementioned reasons, the memory cost for the already exisiting parallel PBVR method proportionally increases as the number of ensembles increases, and the memory cost for our proposed parallel PBVR method does not depend on this assumption. Therefore, in our method, it is possible to improve the image quality by increasing the number of ensembles without increasing the memory cost during the rendering process. Moreover, the memory cost required for each node can also be reduced by increasing the number of processes.

Table 1: Memory cost of the target parallel PBVR method [4] utilized as the benchmark.

# of MPI procs.	# of ensembles		
	1	10	100
4	5.09 MB	34.9 MB	334 MB
8	5.08 MB	35.1 MB	334 MB
16	5.05 MB	34.9 MB	333 MB
32	5.06 MB	34.9 MB	331 MB

64	5.04 MB	34.6 MB	330 MB
128	5.05 MB	34.7 MB	330 MB
256	5.05 MB	34.7 MB	331 MB

Table 2: Memory cost of the proposed parallel PBVR method.

# of MPI procs.	# of ensembles		
	1	10	100
4	4.42 MB	4.42 MB	4.42 MB
8	4.00 MB	3.99 MB	3.99 MB
16	3.78 MB	3.78 MB	3.78 MB
32	3.69 MB	3.69 MB	3.69 MB
64	3.60 MB	3.61 MB	3.61 MB
128	3.58 MB	3.58 MB	3.58 MB
256	3.57 MB	3.57 MB	3.57 MB

We also made a performance analysis evaluations for the proposed parallel PBVR implementation on the K computer. The particle projection stage, which becomes parallelized in our proposed approach required 18.16 seconds when using 4 MPI processes and 100 number of ensembles. It took 5.50 seconds for 8 MPI processes; 4.41 seconds for 16 MPI processes; 3.89 for 32 MPI processes; 3.64 seconds for 64 MPI processes; 3.54 seconds for 128 MPI processes; and 3.46 seconds for 256 MPI processes. The "projection stage" (Blue colored bar) of the Fig. 5 shows the time difference between the sequential projection stage of the previous (already existing) parallel PBVR and the parallel projection stage in the proposed parallel PBVR approach when using the same number of MPI processes (4, 8, 16, 32, 64, 128, and 256) and number of ensembles (1, 10,100). In this graph, the bar in the negative range represents that the projection stage of the proposed approach is faster than the previous method.

We also compared the particle gathering time required in the previous parallel PBVR approach with the image composition time required in the proposed parallel PBVR approach. The time difference is represented as a orange bar in the Fig. 5, and we can verify that the image composition stage is only faster to the particle gathering for the large number of processes (256) and for small number of ensembles (1 and 10). We can verify that as much the number of ensembles increases, the time difference becomes worse, and the performance gain obtained from the parallelization of the projection stage is not sufficient to neutralize this performance penalty. Although it becomes time constly processing, the proposed parallel

PBVR is capable of generating high-quality, high-resolution volume rendering images compared to the previous approach because of its highly memory efficienciness. As future works, we are planning to exploit the order independency processing during the image composition stage in order to develop and utilize an asynchronous image composition module. In addition, we will investigate some techniques, such as those based on deep learning, in order to minimize the number of ensembles while maintaining high-quality rendering results.

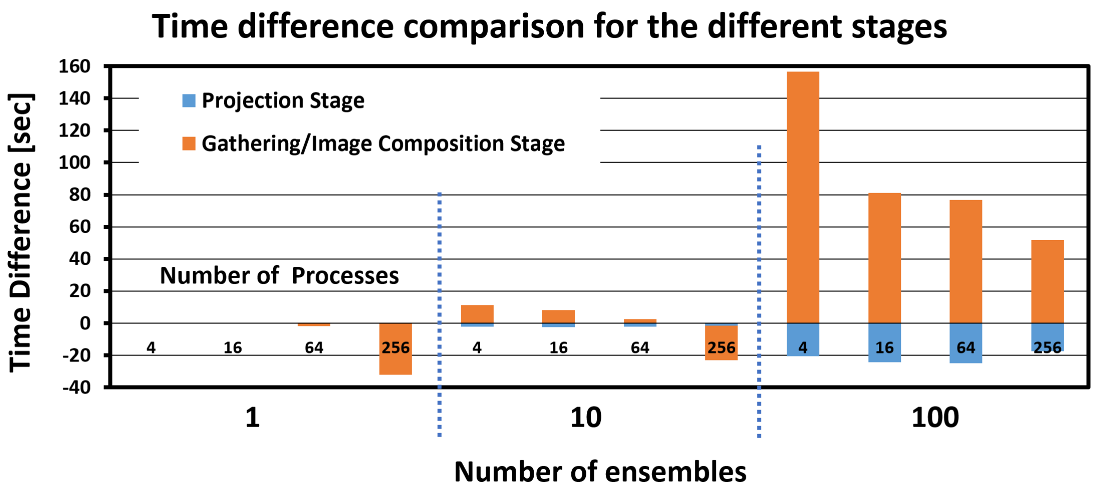

Figure 5: Time difference comparison for the different stages of the already existing and proposed parallel PBVR approaches (sequential particle projection and parallel projection stages; particle gathering and image composition stages).

4. Conclusion

In this work, we proposed a memory-efficient parallel PVBR using sort-last image composition approach for enabling high-quality, high-resolution rendering of large-scale unstructured volume data. In the already exisiting parallel PBVR method, there is a need to store the entire particle data sets before starting the paraticle projection processing. Therefore, for high-quality PBVR which requires large number of repetitions (ensembles), depending on the user-defined transfer function the rendering itself can become impossible if there is no enough memory space to store the particle data. To avoid this particle storage requirements, we propose the use of depth comparison enabled image composition, and we utilized the 234Compositor which is capable of running on the K computer, which was the target hardware system in this work. This memory efficiency become highly attractive for in-situ visualization on such HPC systems, since the simulation and visualization processes should share the same memory space. Although the proposed approach enabled high-resolution and high-quality volume rendering of large-scale unstructured volume data, the overwall performance was not attractive since we utilized a normal parallel image composition library which was not developed to take advantage of the order independency processing of the PBVR. Therefore, as a future work, we are planning to develop and utilize a PBVR oriented asynchronous image composition module. In addition, we will also investigate some techniques, such as those based on deep learning trying to minimize the required number of ensembles while maintaining high-quality rendering results.

Acknowledgement

Our research was carried out under the auspices of the Ministry of Education, Culture, Sports, Science and Technology's entrusted work "Climate Change Adaptation Technology Society Implementation Program (SI-CAT)" and Grant-in-Aid for Scientific Research (JP 17K00169). The unstructured volume data was provided by Mr. Masaya Muto of Meijo University. This work is also partially supported by the "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infastructures" in Japan (Project ID:jh180060-NAH).

References

[1] N. Sakamoto, J. Nonaka, K. Koyamada, S. Tanaka: Particle-based Volume Rendering, in *Proceedings of Asia-Pacific Symposium on Visualization*, Sydney, 2007, 129-132.

[2] P. Sabella: A rendering algorithm for visualization 3d scalar fields, in *ACM SIGGRAPH computer graphics*, Atlanta, 1988, 51-58.

[3] K. Zao, N. Sakamoto, K. Koyamada: Using Interactive Particle-based Rendering to Visualize a Large-scale Time-varying Unstructured Volume with Mixed Cell Types, in *Proc. of IEEE Pacific Visualization 2017 (VisNotes)*, Seoul, 2017, 185-189.

[4] K. Hayashi, T. Shimizu, N. Sakamoto, J. Nonaka: Parallel Particle based Volume Rendering using Adaptive Particle Size Adjustment Techninque, in *SIGGRAPH ASIA Symposium on Visualization(SA17)*, Bangkok, 2017, 11:1-11:8.

[5] J. Nonaka, K. Ono, M. Fujita: 234Compositor: A flexible parallel image compositing framework for massively parallel visualization environments, *Future Generation Computer Systems*, (2017), 647-655.

[6] M. Muto, H. Watanabe, R. Kurose, M. Tsubokura: The effect of surface heating on the drag of a sphere at the critical Reynolds number, in *Proceedings of the 4th International Conference on Jets, Wakes and Separated Flows*, Nagoya, 2013.

[7] H. Miyazaki, Y. Kusano, N. Shinjou, F. Shoji, M. Yokokawa, T. Watanabe: Overview of the K computer system, *Fujitsu Scientific and Technical Journal*, 48:3, (2012), 255–265.

[8] K. Hayashi, N. Sakamoto, J. Nonaka, M. Matsuda, F. Shoji: An In-Situ Visualization Approach for the K computer using Mesa 3D and KVS, *ISC Workshop on In Situ Visualization 2018 (WOIV2018)*, Frankfurt, 2018.