Advanced Visualization Research Team

1. Team members

Kenji Ono (Team Leader)
Jorji Nonaka (Researcher)
Chongke Bi (Postdoctoral Researcher)
Hamed Khandan (Postdoctoral Researcher)
Kazunori Mikami (Technical Stuff)
Masahiro Fujita (Visiting Researcher)
Kentaro Oku (Visiting Researcher)
Naohisa Sakamoto (Visiting Researcher)
Yukiko Hayakawa (Assistant)

2. Research Activities

The purpose of our visualization team is to construct a parallel visualization environment for large-scale datasets, which we named “HIVE”, and also to deliver this system to the users. To achieve this aim, we are developing some fundamental technologies essential to build parallel visualization systems for large-scale datasets, and at the same time, are integrating them into the HIVE system. The HIVE system consists of a parallel rendering kernel named “SURFACE”, some peripheral libraries, and a Web-based user-interface. The main idea of the SURFACE is to exploit the sort-last parallel rendering method, which is divided into the following three tasks: data loading, parallel rendering, and image compositing.

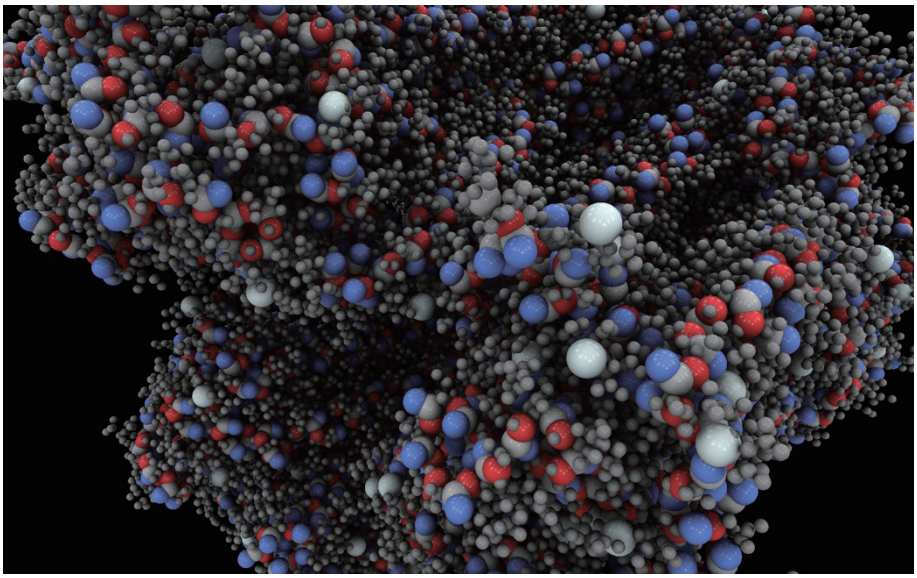


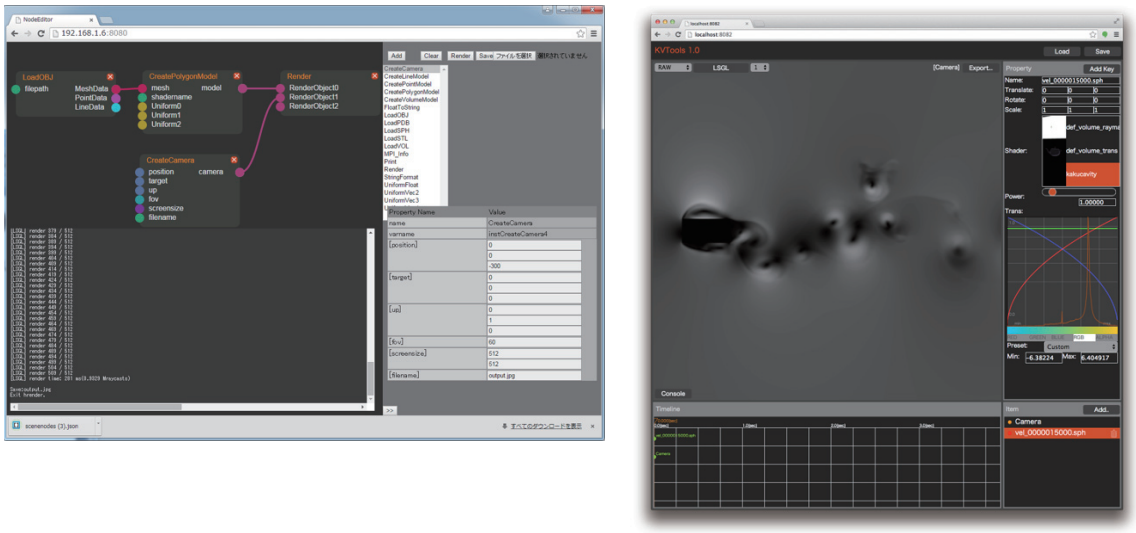
Figure 1. Example of high-quality ray tracing image from a molecular dynamics simulation data. This image was generated using 82,944 nodes on K-Computer.

Although HIVE system is still under development, we have already applied this system to visualize some results from large-scale molecular dynamics simulations executed by other AICS teams. Recent investigations have shown us that the HIVE system enables the users to render more than 10 million atoms on a standard computer resource, and to generate extremely high-resolution images (over 16K resolution) in parallel. Figure 1 shows an example of high-resolution image from a molecular dynamics simulation data.

3. Research Results and Achievements

3.1 HIVE and SURFACE

HIVE is an acronym standing for Heterogeneously Integrated Visual-Analytic Environment, which means that the system can be operated on computer systems with heterogeneous hardware architectures and multi-platform software systems. Figure 2 illustrates some of the appearances of the Web-based user interface of the HIVE system. The software architecture adopted by the HIVE is the server-client model thus it enables the users to operate the HIVE system even from their laptop computers. Figure 3 describes the software stack of the HIVE system.



(a) Workspace for visualization workflow design. (b) Viewing window and control area .

Figure 2. HIVE Web-based user interface.

SURFACE is an acronym standing for Scalable and Ubiquitous Rendering Framework for Advanced Computing Environments, and provides the necessary parallel rendering features to the HIVE system. SURFACE employs the ray tracing method, which is able to generate high-quality images, and is especially suitable for large-scale parallel processing.

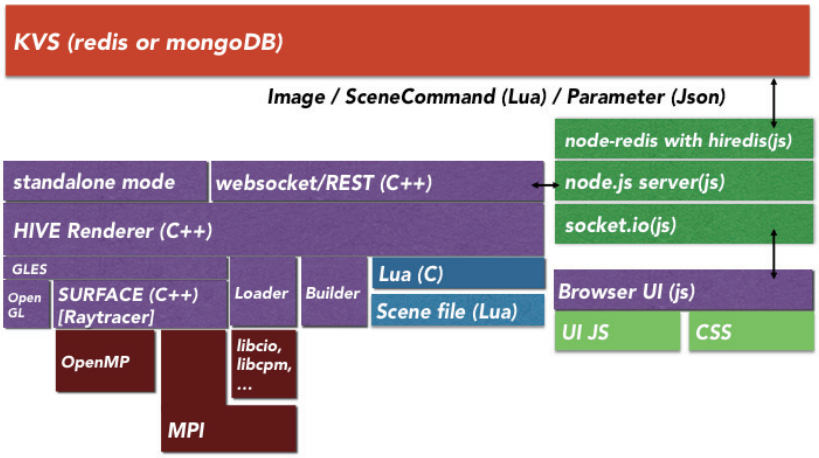


Figure 3. Software stack of the HIVE system.

3.2 Image Compositing

Parallel image compositing corresponds to the last stage of sort-last visualization approach where the parallel rendered images are combined into a single final image. In this year, we worked on an approach to convert non-power-of-two into power-of-two number of compositing nodes in order to enable the use of efficient image compositing algorithms for power-of-two number of nodes. In addition, we worked on an approach to execute large-scale image compositing in multiple steps in order to achieve better scalability.

Efficient parallel image compositing algorithms are worked by exchanging portions of the data, and combining them using appropriate image data merging techniques. It is well known that the exchanging and merging processes work well when the number of compositing nodes is power-of-two. Therefore, different approaches have been proposed to handle non-power-of-two number of compositing nodes. These approaches can be divided into single and multi-stage conversion. The single stage conversion, known as *Reduced* or *Folded* method, converts to the closest power of two ($m = 2^n$) smaller than the number of nodes ($m < m'$) by executing a reduction process known as *2-1 elimination*. We focused on the *2-3-4 Decomposition* approach, shown in Figure 4, for extending the reduction process including the *3-1* and *4-1 eliminations*. The main advantage of this approach is that the resulting number of nodes is smaller (2^{n-1}) compared to the *Reduced* method. This greatly helps to minimize the performance degradation when the number of compositing nodes becomes large. For instance, when the performance degradation between two consecutive power-of-two number of nodes (2^n and 2^{n+1}) is considerable, it becomes possible to achieve faster image compositing compared to the closest power-of-two number of nodes. That is, image compositing between $2^{n+1} + 1$ and $2^{n+2} - 1$ compositing nodes can be faster than using 2^{n+1}

compositing nodes since *2-3-4 Decomposition* reduces the number of compositing nodes to 2^n.

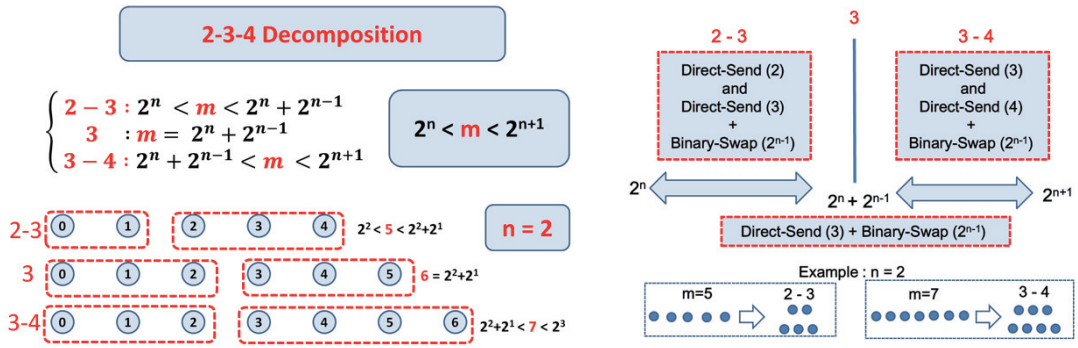


Figure 4. 2-3-4 Decomposition method with an example when n = 2 (Left), and its use in combination with different image compositing algorithms (Right).

Although *2-3-4 Decomposition* can minimize part of the performance degradation on large-scale parallel image compositing, it becomes inefficient when a massive number of compositing nodes in the range of tens of thousands are involved. In the case of K computer, it can reach 82,944 compositing nodes, in Hybrid MPI-OpenMP mode. Therefore, *2-3-4 Decomposition* can reduce it to 32,768 compositing nodes (2^{n-1}). It is worth to mention that the final image collecting is executed through *MPI_Gatherv* collective operation, and there exists a buffer overflow problem on the current MPI implementation for K computer, which does not allow the use of *MPI_Gatherv* with more than 50K nodes. Therefore, by reducing to the range of 32K compositing nodes, it becomes possible to avoid this problem. Although this considerable reduction, it is still in the range of tens of thousands of nodes where the performance degradation is prominent. In order to minimize this degradation, we focused on *Multi-Step* approach, shown in Figure 5, where the entire image compositing nodes are divided into smaller groups, with the size within the range where the performance is not greatly affected. By executing the full parallel image compositing at each of the groups, at the end it will produce partially composited images equal to the number of groups. We can recursively execute this group creation until the number of partially composited images becomes smaller than the group size. Since it only executes parallel image compositing using number of compositing nodes where the performance is not greatly affected, it will result in better scalability. Therefore, the aforementioned *2-3-4 Decomposition* will be greatly benefited.

```
m = Num_Global_Nodes
If ( m != Power-of-two )
  Do Reduction to Power-of-two

Nodes = Num_Nodes_Power of two
Rank = My_Rank_Power of two

While ( Nodes > Max. Group Size )
  Do
    Num_Groups = Nodes / Max. Group Size
    Create MPI Groups based on Rank
      Create MPI Communicator for these Groups
    Create MPI Groups for the Local Roots
      Create MPI Communicator for these Groups
    Nodes = Num_Groups
    Rank  =  New Rank inside the Group of Local Root
End Do
```

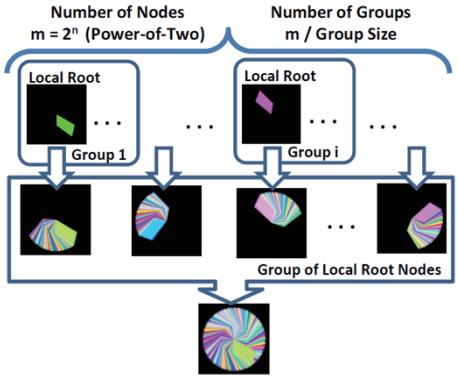


Figure 5. Multi-Step image compositing approach.

3.4 Compression of Large-Scale Dataset

M-Swap Method for Parallel POD Compression of Large-Scale Dataset.

In this paper, we presented a parallel data compression approach to reduce the size of time-varying big datasets. Firstly, we employ the proper orthogonal decomposition (POD) method for compression. The POD method can extract the underlying features of datasets to greatly reduce the size of big datasets. Meanwhile, the compressed datasets can be decompressed linearly. This feature can help scientists to interactively visualize big datasets for analysis. Then, we introduced a novel m-swap method to effectively parallelize the POD compression algorithm. The m-swap method can reach a high performance through fully using all parallel computing processors. In another word, no idle processors exist in the parallel compression process. Furthermore, the m-swap method can greatly reduce the cost of interprocessor communication. This is achieved by controlling the data transfer among $2m$ processors to obtain the best balance of computation cost of these processors.

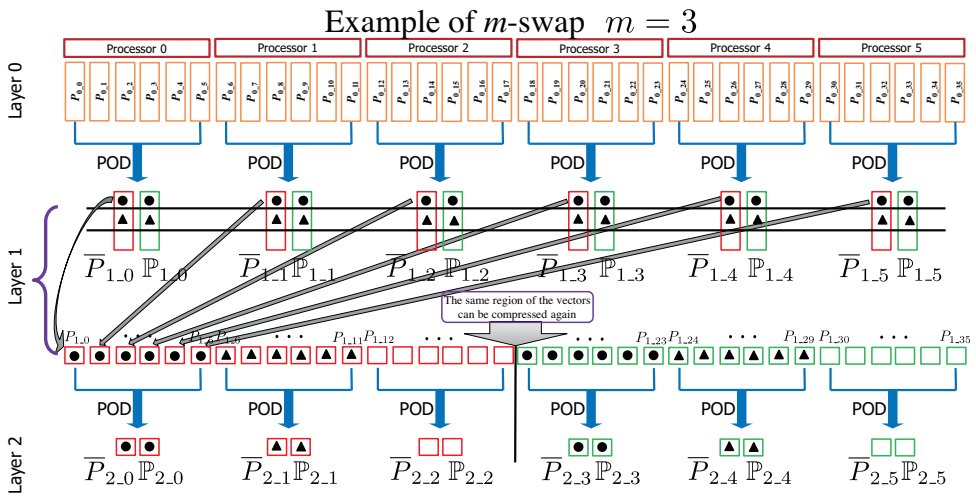


Figure 6. An example of parallel compressing a dataset with 36 time steps.

In our parallel framework, the compressed POD bases and POD mean vectors in different processors will be recursively compressed, respectively. The m-swap algorithm is designed to directly send and receive data among $2m$ (m >= 2) difference processors. Figure 6 shows an example of m = 3. Furthermore, unlike most existing parallel methods that can only deal with the datasets whose time steps number is power-of-two, the presented m-swap method can compress datasets with arbitrary time steps. Note that the binary swap (or 2-3 swap) method cannot be directly used for the POD compression method. Because the size of datasets does not change if two time steps are compressed into one POD mean vector and one POD basis.

Finally, the effectiveness of our m-swap method is demonstrated through compressing several big datasets on the K computer. Figure 7 is one of the results.

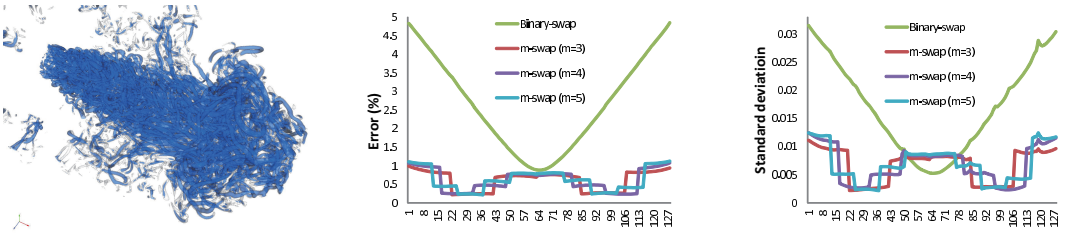


Figure 7. An example of compressing a dataset of the flow simulation in the air jet mixture of a machine. Its size if 300*200*200*128. The rendering result, the error, and the corresponding standard deviation of error are shown from left to right.

Fluid Data Compression and ROI Detection Using Run Length Method

Data compression and ROI (Region of Interest) detection are often used to improve efficiency of the visualization of numerical data. It is well known that the Run Length encoding is a good technique to compress the data where the same sequence appeared repeatedly, such as an image with little change, or a set of smooth fluid data. Another advantage of Run Length encoding is that it can be applied to every dimension of data separately. Therefore, the Run Length method can be implemented easily as a parallel processing algorithm. We proposed two different Run Length based methods. When using the Run Length method to compress a data set, its size may increase after the compression if the data does not contain many repeated parts. We only apply the compression for the case that the data can be compressed effectively. By checking the compression ratio, we can detect ROI. Figure 8 shows an example of compressing a 3D fluid data using Run Length method.

(X, Y, Z)	time direction		results of Run Length method
$(0, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(1, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(2, 0, 0)$	$[W\ A\ D\ \cdots\ A\ I]$	$\rightarrow$	$[W\ 1\ A\ 1\ D\ 1\ \cdots\ A\ 1\ I\ 1]$
$(3, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(4, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(5, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(6, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(7, 0, 0)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$
$(8, 0, 0)$	$[T\ H\ R\ \cdots\ E\ E]$	$\rightarrow$	$[T\ 1\ H\ 1\ R\ 1\ \cdots\ E\ 2]$
$\vdots$			
$(121, 361, 181)$	$[A\ A\ A\ \cdots\ A\ A]$	$\rightarrow$	$[A\ 257]$

Figure 8. An example of compressing a 3D fluid data using Run Length method.

4. Schedule and Future Plan

We will continuously improve our visualization framework by ameliorating the components of the software stack, and by aggregating new functionalities. We also have plans to give lectures regarding the visualization framework and its components.

5. Publication, Presentation and Deliverables

(1) Journal Papers

1. Shota Ishikawa, Haiyuan Wu, Chongke Bi, Qian Chen, Hirokazu Taki, and Kenji Ono, "Fluid Data Compression and ROI Detection Using Run Length Method," in Procedia Computer Science, Vol. 35, pp. 1284-1291, 2014.
2. Kenji Ono, Yasuhiro Kawashima, and Tomohiro Kawanabe, "Data Centric Framework for Large-scale High-performance Parallel Computation," Procedia Computer Science, Vol. 29, pp. 2336 – 2350, 2014.

(2) Conference Papers

1. Chongke Bi, Kenji Ono, Kwan-Liu Ma, Haiyuan Wu, and Toshiyuki Imamura, "Proper Orthogonal Decomposition Based Parallel Compression for Visualizing Big Data on the K Computer," in Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, pp. 1-8, June, 2014.
2. Chongke Bi, Kenji Ono, and Lu Yang, "Parallel POD Compression of Time-Varying Big Datasets Using m-Swap on the K Computer," in Proceedings of IEEE International Congress on Big Data, pp. 438-445, June, 2014.
3. Hamed Khandan, "Introducing A-Cell for Scalable and Portable SIMD Programming,"

2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs (MCSoC), pp. 275-280, 2014.

4. Jorji Nonaka, Masahiro Fujita, and Kenji Ono, "Multi-Step Image Compositing for Massively Parallel Rendering," in Proceedings of the International Conference on High Performance Computing & Simulation 2014, Bologna, Italy, pp. 627-634, 2014.
5. Jorji Nonaka, Chongke Bi, Masahiro Fujita, and Kenji Ono, "2-3-4 Decomposition Method for Large-Scale Parallel Image Composition with Arbitrary Number of Nodes," in Proceedings of the First International Conference on Systems Informatics, Modelling and Simulation, Sheffield, UK, pp. 59-64, 2014.
6. Kenji Ono, Shuichi Chiba, Shunsuke Inoue, and Kazuo Minami, "Performance Improvement of Iterative Method with Bit-Representation technique for Coefficient Matrix," 11th International Meeting High Performance Computing for Computational Science, Eugene, Oregon, USA, June 30 - July 3, 2014.
7. Kenji Ono and Yasuhiro Kawashima and Tomohiro Kawanabe, "Data Centric Framework for Large-Scale High Performance Parallel Computation," International Conference on Computational Science, Cairns, Australia, June 10-12, 2014.
8. Hamed Khandan and Kenji Ono, "Knowledge Request-Broker Architecture: A Possible Foundation for a Resource-Constrained Dynamic and Autonomous Global System," 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 506-507, 2014.
9. Kenji Ono and Jorji Nonaka, "Active Subdomain Selection for Efficient Parallel Computation of Internal Flows in Complex Geometry," Books of extended abstracts of 26th International Conference on Computational Fluid Dynamics, pp. 112–113, 2014.

(3) Invited Talks

1. Kenji Ono, "Design of Practical Framework to utilize Big Data on Exascale Computer," ORAP Forum 33, CNRS, Paris, France, April, 2014.
2. Kenji Ono, "Scalable and Ubiquitous Visualization in Extreme-Scale Computational Environment," Extreme Performance Computational Science French-Japanese Conference, Embassy of France in Japan – Department for Science and Technology, Tokyo, Japan, April, 2014.

(4) Posters and presentations

1. Chongke Bi, "An m-Swap method for Parallel POD Compression on the K Computer," in Proceedings of Sparse Modeling High-Dimensional Data-Driven Science, pp. 165-166, December, 2014.
2. Chongke Bi, "In-situ Visualization of Big Data Using Sparse Modeling," in Proceedings of

Conference on Sparse Modeling, June, 2014.

3. Jorji Nonaka, Masahiro Fujita, Kenji Ono, Naohisa Sakamoto, Koji Koyamada, “A Study on Parallel PBVR for Large Data Visualization,” 第 42 回可視化情報シンポジウム, Tokyo, Japan, 2014.
4. Jorji Nonaka, Masahiro Fujita, Kenji Ono, “SURFACE: A Visualization Framework for Large-Scale Parallel Simulations,” Oral presentation at Ultrascale Visualization Workshop 2014, New Orleans-LA, USA, 2014.
5. Masahiro Fujita, Jorji Nonaka, and Kenji Ono, “LSGL: Large Scale Graphics Library for Peta-Scale Computing Environments,” Poster presentation at High Performance Graphics 2014, Lyon, France, 2014.
6. Kenji Ono, “CFD Applications for Industrial Design in Peta-Scale Computing Environments,” International HPC Summer School 2014, Budapest, Hungary, June 2-6, 2014.
7. Kenji Ono, “Bit-Representation of Boundary Conditions for High Performance Incompressible Thermal Flow Simulation,” 11th World Congress on Computational Mechanics (WCCM XI), Barcelona, Spain, July 20-25, 2014.
8. Shigueho Noda, Kazuyasu Sugiyama, Yasuhiro Kawashima, , Kenji Ono, Shu Takagi, , and Ryutaro Himeno, “Development and Applications of the Parallel Computing Middleware for the Life Science Simulations,” 11th World Congress on Computational Mechanics (WCCM XI), Barcelona, Spain, July 20-25, 2014.
9. Ken Uzawa, Kenji Ono, and Takanori Uchida, “Validation of Local SGS Models for High Reynolds Number Flow,” 11th World Congress on Computational Mechanics (WCCM XI), Barcelona, Spain, July 20-25, 2014.
10. Kenji Ono, “Challenges and Strategy to Tackle the Extreme-Scale Fluid Simulation for Engineering Process,” Japan/United States Exascale Applications Workshop, Gatlinburg, Tennessee, USA, September 5-6, 2014.
11. Jyunya Onishi, and Kenji Ono, “A Cartesian Grid Method for Simulating Two-Phase Flow in Complex Geometries,” 2nd International Conference on Numerical Methods in Multiphase Flows, Darmstadt, Germany, April, 2014.

(5) Patents and Deliverables

All released software products from this team are delivered via GitHub, and are available in the following repository: https://github.com/avr-aics-riken/

1. PMlib

   Performance Monitor library
2. SURFACE

Scalable and Ubiquitous Rendering Framework for Advanced Computing Environments

3. HIVE

   Heterogeneously Integrated Visualization Environment

4. CDMlib

   Cartesian Data Management library

5. PDMlib

   Particle Data Management library

6. JHPCN-DF

   Data compression library based on Jointed Hierarchical Precision Compression Number - Data Format

7. Text Parser

   Text Parser library that enables us to handle YAML-like simple text parameters

8. 234Compositor

   A parallel image composition library being developed to perform sort-last image composition on massively parallel environments.