# Chapter 15

# Advanced Visualization Research Team

## 15.1 Members

Kenji Ono (Team Leader)

Jorji Nonaka (Researcher)

Tomohiro Kawanabe (Technical Staff)

Kazunori Mikami (Technical Staff)

Masahiro Fujita (Visiting Technician)

Naohisa Sakamoto (Visiting Researcher)

Mario Antonio Ribeiro Dantas (Visiting Researcher)

Takashi Shimizu (Student Trainee)

Steve Petruzza (Student Trainee)

Eduardo Camilo Inacio (Intern)

Aya Motohashi (Assistant)

## 15.2 Research Activities

The research and development activities of this team, in these six years of continuous activities, have covered a broad range of the end-to-end simulation pipeline, focusing on the study, design, and development of effective tools and mechanisms for enabling high performance visualization functionalities on the K computer operational environment. The resulting libraries, tools, and applications have been publicly released as open-source software codes via GitHub, and they are listed in the deliverables subsection. In this final year of the continuous activities, we focused on enhancing the functionalities of the HIVE (Heterogeneously Integrated Visual analytics Environment) visualization application, including the input (xDMlib: Data Management Library), output (ChOWDER: Cooperative Tiled Display System), and rendering (PBVR: Particle Based Volume Rendering) capabilities. In addition, we have also worked on a workflow management system (WHEEL), and on a performance measurement and visualization tool (PMlib).

### 15.2.1 Heterogeneously Integrated Visual-analytics Environment (HIVE)

HIVE has been one of the main developments of this team, and was designed to run in the heterogeneous environments of traditional HPC infrastructures, which obviously includes the K computer environment that is composed of the supercomputer itself, the post-processing servers, and a research-oriented visualization cluster. As shown in Fig. 15.1, the HIVE is composed of several modules, most of them were internally developed and some are third-party developed modules, each possessing different functionalities. The HIVE adopted the Client/Server approach, and currently the HIVE Server is able to run on SPARC64fx and x86 hardware systems

running Linux, MacOSX and Windows (via Windows Subsystem for Linux). On the other hand, the HIVE Client only requires a Web browser and an appropriate network connection to the HIVE Server. The aforementioned HIVE modules are loosely-coupled via Lua scripting language, and the communication between the HIVE Server and Client is executed via Websocket connection. The Web-based user interface can serve as a user friendly GUI for selecting the visualization parameters for preparing the "Visualization Scenes" to be used in the large-scale parallel rendering job submissions via Command Line Interface (CLI). The right side of the Figure. 15.1 shows an visualization example of this CLI-based visualization of a large-scale particle-based simulation result rendered by using the data loading mechanism of PDMlib (Particle Data Management Library), and the ray-tracing rendering functionality of SURFACE (Scalable and Ubiquitous Rendering Framework for Advanced Computing Environments). In this rendering, the buildings are treated as particle data and the particle data representing tsunami was converted to polygonal data, via own developed OpenVDB-based polygonal data converter, and rendered as a semi-transparent object.

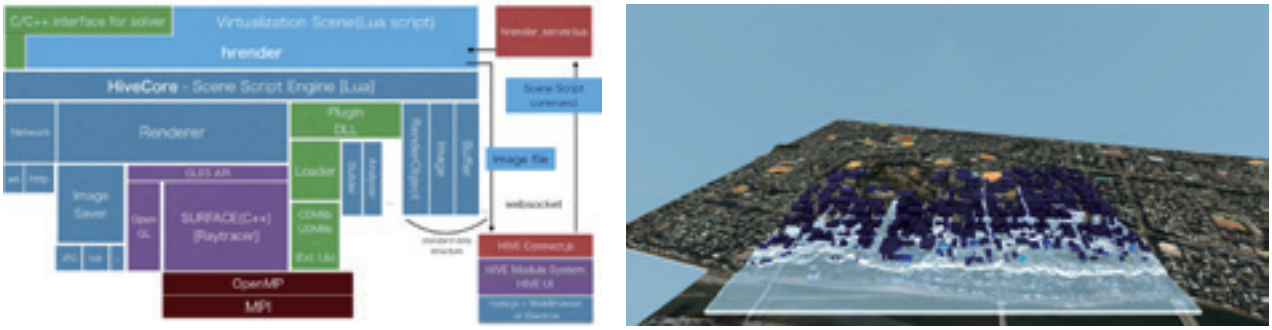


Figure 15.1: An overview of the HIVE software stacks showing the main modules, and a visualization example of a particle-based tsunami simulation rendered by the HIVE (using PDMlib and SURFACE functionalities).

As described in the previous paragraph, two of the most important characteristics of the HIVE are the Server/Client architecture and the loosely-coupled module integration via Lua scripting language. By taking advantage of these two characteristics, in this fiscal year, we focused on enhancing the HIVE functionality by including the Particle-Based Volume Rendering (PBVR), and mechanisms for visual causal exploration. For this purpose, we utilized the functionalities provided by the KVS (Kyoto Visualization System) initially developed at the Kyoto University, and currently maintained by the Kobe University. For the PBVR framework, we utilized the particle generation functionality of KVS, and stored the generated particle data as PDMlib (Particle Data Management library) data. For the visual causal exploration framework, we utilized the causality volume data generation functionality of KVS, and stored as CDMlib (Cartesian Data Management library) data. We also developed some shader codes for rendering these data sets via SURFACE ray-tracing functionalities, and these examples are shown in Fig. 15.2.

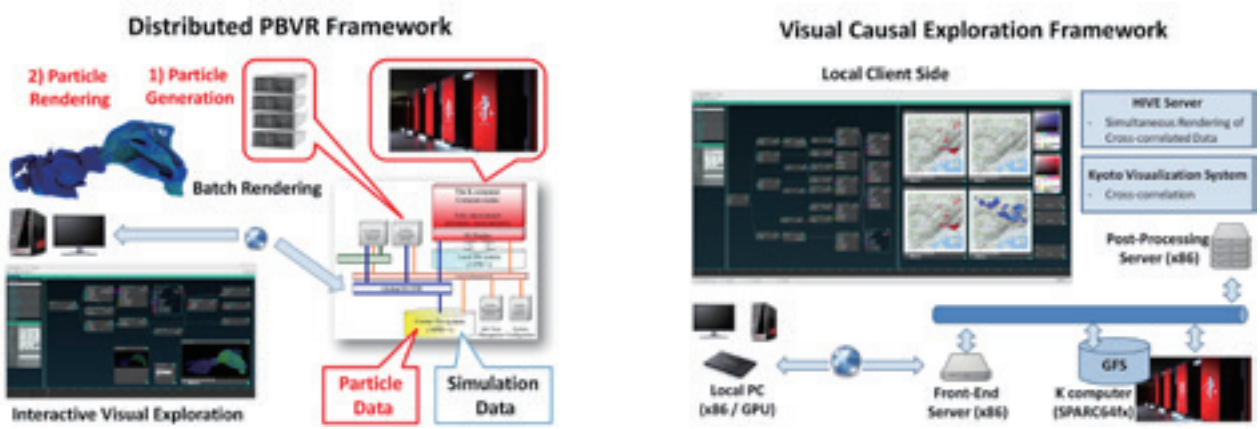


Figure 15.2: HIVE Functionality Enhancements: Distributed Particle-Based Volume Rendering (PBVR) Framework and a Visual Causal Exploration Framework.

### 15.2.2 Data Management Library (xDMlib)

The xDMlib data management library has been designed to support the I/O during simulations, such as the restart with different resolution size or number of computational nodes, and also to make a bridge between the I/O of simulation and visualization applications. The "x" in the xDMlib represents the type of data format: "C" for the Cartesian Data (CDMlib) such as voxel data; "P" for the Particle Data (PDMlib); "U" for the Unstructured Data (UDMlib) such as defined by the CGNS (CFD General Notation System) library; and "H" for the Hierarchical Data (HDMlib) such as the hierarchical block-structured data. One of the main characteristics of this library is the data aggregation and repartition mechanism for providing a flexible data I/O functionality, which can absorb the difference between the number of files ($M$) and the number of nodes ($N$) for reading or writing as shown in the Fig. 15.3.

Considering that the number of nodes available in hardware systems for post processing is usually much smaller than the number of generated files during the large job runs, the $M \times 1$ and $M \times N$ loading mechanism becomes the most important features from the visualization point of view. As shown in Fig. 15.3, the $M \times N$ data loading can be *aligned* or *unaligned* based on the selected partitioning parameters, and it is worth noting that unaligned data partitioning can impact the I/O performance. Another important mechanism of xDMlib is the use of lightweight meta-information for the distributed data management in order to maintain the original simulation data intact while extracting only the necessary data for the analysis and visualization processes. For this purpose, xDMlib utilizes two meta-information for managing the distributed files: information related to the contents of the data (index.dfi); and the information related to the data distribution among the processes (proc.dfi).

Among the several simulation codes running on the K computer, the SCALE (Scalable Computing for Advanced Library and Environment) is one of the representative large-scale simulation codes. This is a computational climate simulation code, and has been generating a vast amount of time-varying, multivariate simulation results. Therefore, a flexible data loading mechanism for the visualization and analysis of the already stored data sets as well as the upcoming simulation results becomes highly valuable. However, the file I/O is based on the NetCDF (Network Common Data From) file format, and can use the HDF5 for storing the compressed data as NetCDF4/HDF5 data. As a result, we extended the CDMlib library by including the necessary functionalities provided by the NetCDF (Version 4.2) and HDF5 (Version 1.8.10 patch 1) libraries. Figure 15.3 shows an overview of the implemented "metadata" generator (netcdf2dfi), and the extended CDMlib API for enabling the flexible data I/O of NetCDF4/HDF5-based computational climate simulation results.

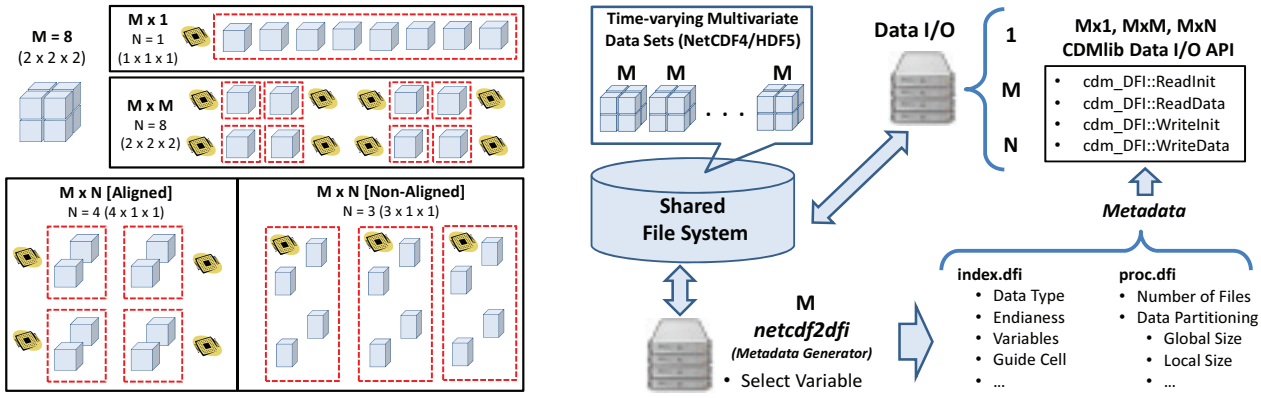


Figure 15.3: An overview of the data loading mechanism and the metadata generation process.

### 15.2.3 Cooperative Tiled Display System (ChOWDER)

ChOWDER (Cooperative Workspace Driver) is a web-based Tiled Display Wall (TDW) system that can be used to enhance collaborative works among multiple sites over the internet. By using the concept of "Virtual Display Area (VDA)", it can dynamically change the size of display area and the magnification ratio of the contents without the constraints on the limitations of the resolution on the physical display side reffig:VDA. When sharing a VDA among multiple sites, even they having TDW with different resolutions or aspect ratios, the ChOWDER can appropriately adjust the display magnification ratio of the VDA for each of the sites.

Besides, the ChOWDER was designed to allow dynamic change of participants, that is, any remote participant can dynamically add or remove their physical displays to the VDA even during the run-time.

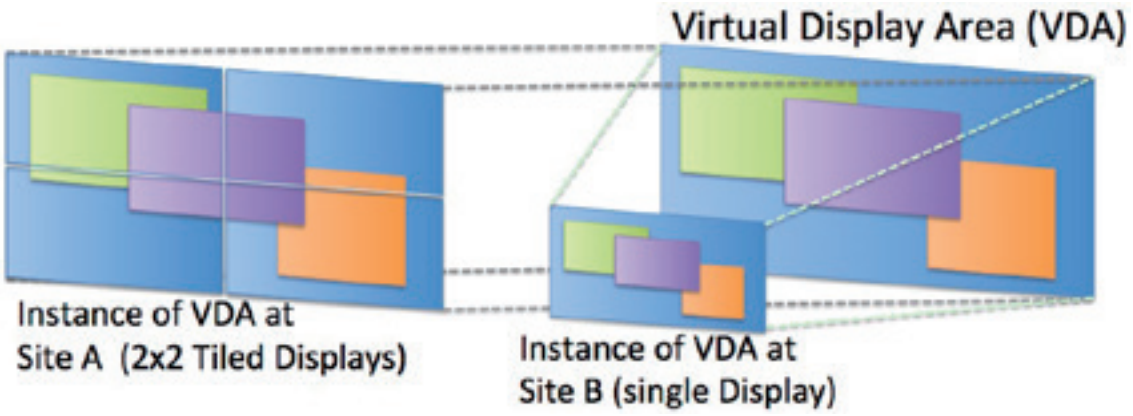


Figure 15.4: The concept of Virtual Display Area (VDA).

In this fiscal year, we developed some additional functionalities to the ChOWDER for enabling contents sharing of Movies, Screens, and Webcams. These functionalities are built using the P2P data transmission protocol called WebRTC, and the contents data is sent directly from the sender device to the display devices where the contents will be displayed. As a real-world usage scenario, we evaluated the effectiveness of these new functionalities by setting up an collaborative work scene using the TDWs placed at the Team's Laboratory, here in Kobe, and at the RIIT (Research Institute of Information Technology) at Kyushu University, in Fukuoka, as shown in Fig 15.5.

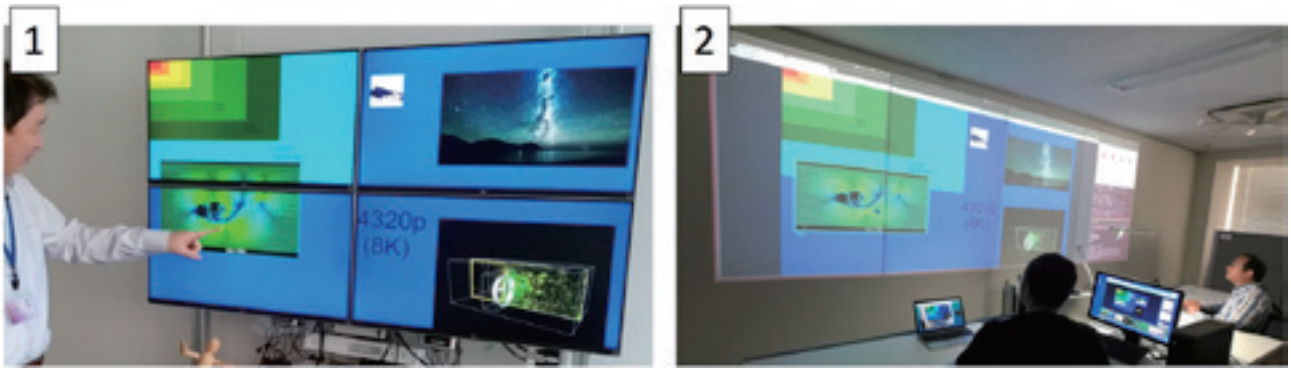


Figure 15.5: Distributed collaboration between Kyushu University (1) and RIKEN R-CCS (2) using ChOWDER.

### 15.2.4 Workflow Management System (WHEEL)

Typical end-to-end processing workflow on the HPC systems is just a repetition of a routine involving pre-processing and post-processing. The repetitive execution of this kind of workflow may waste the time of researchers and engineers and reducing the time for analysis and understanding of the simulated phenomenon, and interfering in the knowledge acquisition process. In order to solve this problem, this team has worked on a workflow management system named WHEEL (Workflow in Hierarchical distributEd parallEL), and is supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) as one of the "Priority Issue on Post-K computer".

The WHEEL is based on another workflow management system named SWF, jointly developed by our team and the Kyushu University in last fiscal year. The developments have been focused on the refinement of the web-based user interface, and the addition of new functionalities. The WHELL can be defined as a multi-platform automated job preparation environment, including pre- and post-processing, for the efficient job

execution on the HPC systems. Since it uses web-based GUI, and run on web browser, thus it does not depend on the executing hardware platform. The main processing is handled by the *Node.js*, which is a cross-platform runtime environment, and enables its use on the Linux, Mac, and Windows hardware platforms. Users can define the "tasks" for each of the processing units via this web-based GUI, and can define the execution order by connecting these tasks under the GUI workspace, as shown in Figure 15.6.

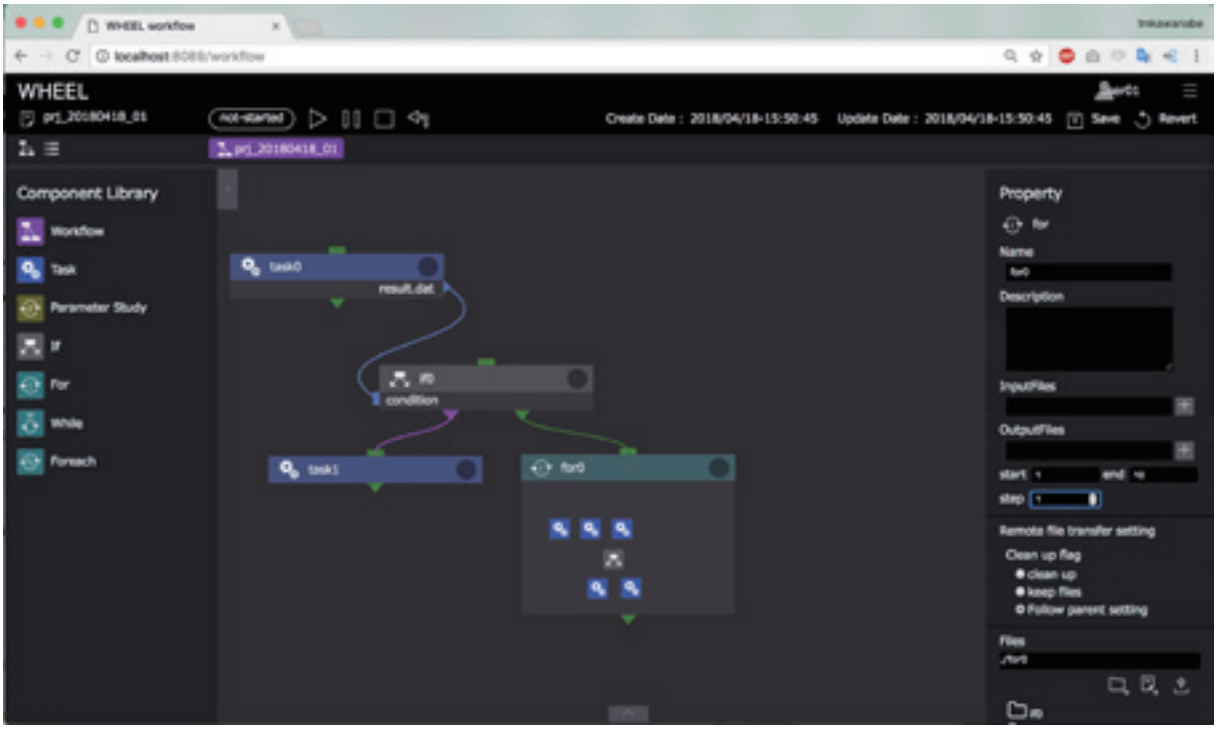


Figure 15.6: An overview of the WHEEL and its web-based GUI.

One of the distinctive features of the WHEEL is its parameter study execution component. By editing the parameter file of the simulation software inside the text editor incorporated in the WHEEL, and designating the sweep range of arbitrary parameter value, the simulation jobs for the specified parameter combinations are automatically executed. After the job execution, the users will receive the result data files that the users specified in advance for receiving after the execution. Compared to the SWF, some components have been added: Iterative processing; Conditional branching; and Sub-workflow. By using these components, in a combined manner, users can also build multi-objective design exploration workflow by using evolutionary calculation module based on genetic algorithm. In this fiscal year, we built and evaluated the operation of this workflow, using the evolutionary calculation module named "cheetah", which is provided by the JAXA.

### 15.2.5 Performance Monitoring Library (PMlib)

PMlib is designed to monitor the computational performance of scientific applications and to visualize their characteristics. The computational performance information such as flops and/or bandwidth can be obtained from the actually executed system workload which is automatically read through the hardware performance counters embedded in the modern CPUs. The alternative performance information can be obtained based on the theoretical requirement by the source program. The PMlib accepts the explicitly provided user arguments for such purposes.

The development effort during this fiscal year was focused on expanding the support of new CPU products such as the Intel Skylake and SPARC64 XIfx (Fujitsu PRIMEHPC FX100) which have new hardware instructions and corresponding event sets. By covering those event sets, the PMlib provides useful information regarding the effective computational performance for the specific micro architecture such as wide SIMD computation. As an example, the net computational performance of the loop body of the following basic kernels will be shown.

```
do i=1,n ; c(i)=a(i)+b(i) ; end do
do i=1,n ; c(i)=a(i)+b(i)*d ; end do
do i=1,n ; c(i)=b(i)/a(i) ; end do
do i=1,n ; c(i)=sqrt(a(i)) ; end do
```

All of the computations are performed with a unit stride with the innermost loop length of $n = 1, 2, 3, .., 50$. The SIMD instructions are generated by the compilers for such non-recurrent loops. Figure 15.7 shows the performance results on FX100 with the variables defined as double precision. The curves show non-monotonic increase, which are often observed on SIMD implemented CPU architectures. The performance jumps according to the SIMD width (SIMD bit width / data bit width), and on the FX100, these jumps occurred at loop length of $256/64 = 4$ 's multiple. This fig 15.7 indicates that a careful programming practice will lead to the significant performance difference in short loops. Figure 15.8 shows the results of the same kernel with variables defined as single precision. A major impact was observed for the loop lengths which coincides with multiple of $256/32 = 8$ .

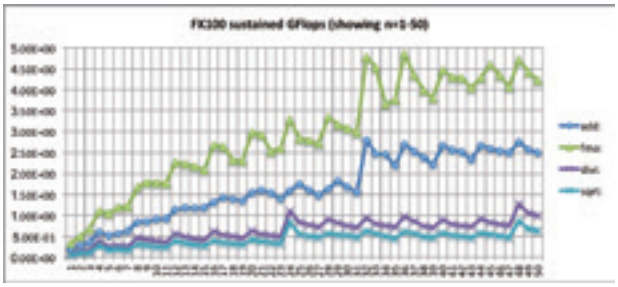

Figure 15.7: PMlib-fx100-R8

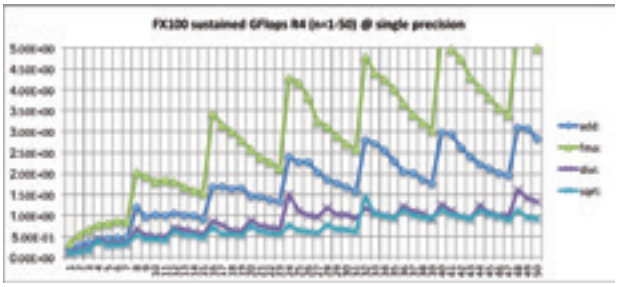

Figure 15.8: PMlib-fx100-R4

We also made some enhancements to the Web-based visualization package, named TRAiL, and designed for the PMlib. In addition to the default PMlib report, users can now produce the time dependent performance trace files in the open trace format, which can be visualized using the TRAiL. Figure reffig:TRAiL-512MPI shows a partial TRAiL view of a 512 process PMlib job.

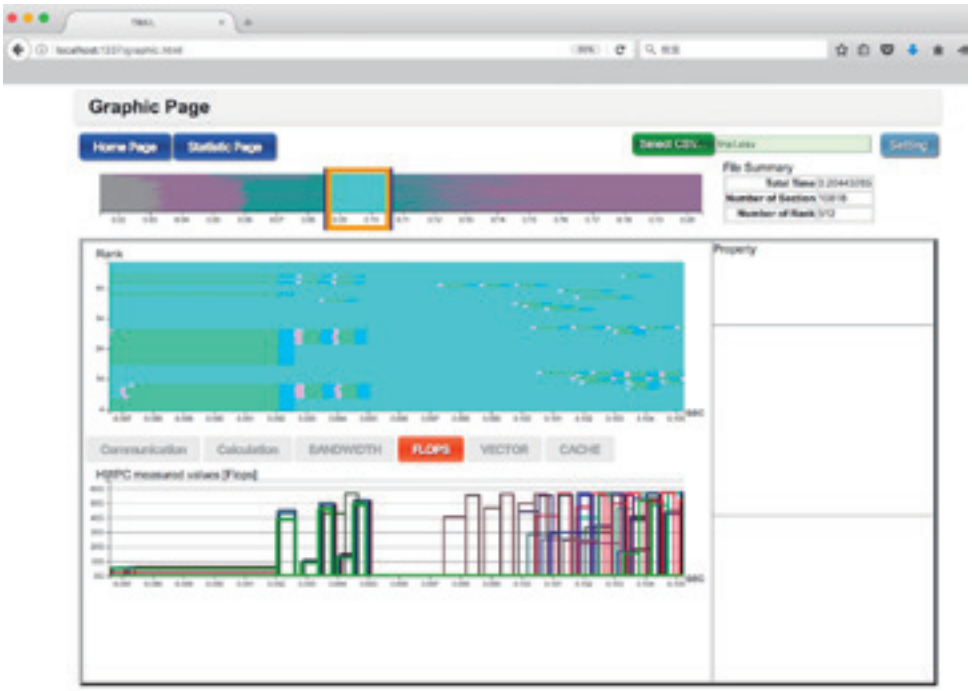


Figure 15.9: PMlib-TRAiL visualization

### 15.2.6 Summary

The Advanced Visualization Research Team has actively conducted the research and development of visualization related techniques from 2012 to 2018. The team has aimed to derive and promote scientific discovery and explore the essence of the simulated phenomenon inside the large-scale data sets generated from huge computational resources such as the K computer, and hence, to assist the knowledge creation and improvement of industrial design. Visualization and data processing are interdisciplinary research area among the information science, computational science, and computer science. Our team has been intensively focused and researched the methodologies to efficiently handle, analyze, and visualize data from various application fields. Therefore, our research area covered a wide range of topics, e.g., visualization of large-scale dataset, data compression with sparse modeling, parallel data management, execution environment for large-scale parallel computation, performance monitoring during the application run-time, performance visualization, and the parallel time integration technique. The Advanced Visualization Research Team can be characterized not only by the conducted research activities, but also for being conscious of the importance to feedback the research results to the society, and has conducted a wide range of activities, and our main contribution can be summarized as follows:

[1] We developed some core technologies and an integrated system which is capable of efficiently assist the visualization and data analysis on large-scale HPC environment such as the K computer environment, and potentially on the coming Post-K computer.

[2] The developed visualization system has been applied to the post-processing of the simulation results generated from various applications, and contributed to the outreach of the K computer.

[3] The developed visualization system has been continuously enhanced and maintained, and the users has also benefited from the visualization support and assistance.

[4] The team has contributed to the human resource development, and our team fostered over eight researchers, two technical staffs, and five student trainees, including some from overseas.

In addition to the R-CCS internal budget, our team successfully obtained more than five competitive external funds from MEXT and JSPS. Some collaborative research projects have been conducted with academia and private companies, including the University of Utah, in the USA, and the Federal University of Santa Catarina, in Brazil.

## 15.3 Publications

### 15.3.1 Journal Articles

[1] Jorji Nonaka, Kenji Ono, and Masahiro Fujita. “234Compositor: A Flexible Parallel Image Compositing Framework for Massively Parallel Visualization Environments”. In: *Elsevier Future Generation Computer Systems*, Vol. 82, pp. 647-655, May 2018.
[2] Jorji Nonaka, Eduardo Camilo Inacio, Kenji Ono, Mario Antonio Ribeiro Dantas, Yasuhiro Kawashima, Tomohiro Kawanabe, Fumiyoshi Shoji. “Data I/O Management Approach for the Post-hoc Visualization of Big Simulation Data Results”. International Journal of Modeling, Simulation, and Scientific Computing, https://doi.org/10.1142/S1793962318400068 (Onlilne Ready).
[3] Kazunori Mikami, Kenji Ono. “Performance Measurement and Performance Visualization Method using PMlib (in Japanese)”. IPSJ-SIG on HPC, 2018-HPC-163, 2017.

### 15.3.2 Conference Papers

[4] Jorji Nonaka, Naohisa Sakamoto, Takashi Shimizu, Masahiro Fujita, Kenji Ono, and Koji Koyamada. “Distributed Particle-based Rendering Framework for Large Data Visualization on HPC Environments” In: *The 2017 International Conference on High Performance Computing & Simulation (HPCS 2017)*, Genoa, Italy, 2017.
[5] Jorji Nonaka, Naohisa Sakamoto, Yasumitsu Maejima, Kenji Ono, Koji Koyamada. “A Visual Causal Exploration Framework / Case Study: A Torrential Rain and a Flash Flood in Kobe City”. In: *ACM SIGGRAPH Asia 2017 Symposium on Visualization*, Bangkok, Thailand, 2017.

[6] Kengo Hayashi, Takashi Shimizu, Naohisa Sakamoto, Jorji Nonaka."Parallel Particle-based Volume Rendering using Adaptive Particle Size Adjustment Technique". In: *ACM SIGGRAPH Asia 2017 Symposium on Visualization*, Bangkok, Thailand, 2017.
[7] Jorji Nonaka, Motohiko Matsuda, Takashi Shimizu, Naohisa Sakamoto, Masahiro Fujita, Keiji Onishi, Eduardo Camilo Inacio, Shun Ito, Fumiyoshi Shoji, Kenji Ono. "A Study on Open Source Software for Large-Scale Data Visualization on SPARC64fx based HPC Systems". In: *International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2018)*, Tokyo, Japan, 2017.
[8] Eduardo Camilo Inacio, Jorji Nonaka, Kenji Ono, Mario Antonio Ribeiro Dantas. "Analyzing the I/O Performance of Post-Hoc Visualization of Huge Simulation Datasets on the K Computer". In: *XVIII Brazilian Symposium on High Performance Computational Systems*, Sao Paulo, Brazil, 2017.
[9] Kazunori Mikami, Kenji Ono. "Performance Measurement and Performance Visualization Method using PMlib (in Japanese)". IPSJ-SIG on HPC, Ehime, Japan, 2017.

### 15.3.3 Oral Talks and Posters

[10] Jorji Nonaka, Kenji Ono, Tomohiro Kawanabe. "TDW-based Visualization using HIVE-ChOWDER (Oral Talk)". *NIFS (National Institute for Fusion Science) Research Meetings for Advanced Visualization Technique*, Toki, Japan, 2017.
[11] Kengo Hayashi, Yoshiaki Yamaoka, Naohisa Sakamoto, Jorji Nonaka. "Parallel Particle-based Volume Rendering with 234Compositor for Large-Scale Unstructured Volume Data Visualization (Poster)". *The 8th AICS International Symposium 2018*, Kobe, Japan, 2018.
[12] Kazuki Koiso, Naohisa Sakamoto, Jorji Nonaka, Fumiyoshi Shoji. "Development of a Visual System for Failure Analysis using HPC Log Data (Poster)". *The 1st Visualization Workshop*, Yokohama, Japan, 2018.
[13] Yoshiaki Yamaoka, Kengo Hayashi, Naohisa Sakamoto, Jorji Nonaka. "Parallel Particle Based Volume Rendering using 234Compositor (Poster)". *The 1st Visualization Workshop*, Yokohama, Japan, 2018.
[14] Yusuke Imoto, Daisuke Tagami, Mitsuteru Asai, Yoshitaka Watanabe, Kenji Ono, Naoto Mitsume, Daisuke Nishiura, Jorji Nonaka. "Accuracy Improvement of the Particle-based Method and its Application to Large Scale Fluid Simulator (Poster)". *The 9th JHPCN Symposium, Joint Usage/Research Center for Interdisciplinary Large-Scale Information Infrastructures.* Tokyo, Japan, 2017.

### 15.3.4 Deliverables

[15] *HIVE (Heterogeneously Integrated Visual-analytics Environment).* http:// avr-aics-riken.github.io/HIVE
[16] *ChOWDER (Cooperative Work Driver).* https://github.com/SIPupstream Design/ChOWDER
[17] *PMlib (Performance Monitoring Library).* http://avr-aics-riken.github.io/PMlib
[18] *CDMlib (Cartesian Data Management Library).* http://avr-aics-riken.github.io/CDMlib
[19] *Cutlib (Cut Information Library).* https://github.com/avr-aics-riken/Cutlib
[20] *PDMlib (Particle Data Management Library).* http://avr-aics-riken.github.io/PDMlib
[21] *HDMlib (Hierarchical Data Management Library).* https://github.com/avr-aics-riken/HDMlib
[22] *UDMlib (Unstructured Data Management Library).* http://avr-aics-riken.github.io/UDMlib
[23] *Polylib (Polygon Management Library).* http://avr-aics-riken.github.io/Polylib
[24] *TPLib (Text Parser Library).* http://avr-aics-riken.github.io/TextParser
[25] *V-Isio (VCAD Visualizer).* http://avr-aics-riken.github.io/V-Isio
[26] *FFVC (FrontFlow Violet Cartesian).* http://avr-aics-riken.github.io/ffvc_package
[27] *LPTlib (Lagrangian Particle Tracking Library).* http://avr-aics-riken.github.io/LPTlib
[28] *JHPCN-DF (Jointed Hierarchical Precision Compression Number - Data Format).*
http://avr-aics-riken.github.io/JHPCN-DF