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 are generated from K computer, and to provide the visualization environment for the users. Of course, the development of elemental technologies is included. The following is the objectives of our research in FY2013.

1) Data centric approach for simulation and post-processing (Ono)
2) Development of a visualization technique for large-scale dataset (Nonaka)
3) Data compression for large-scale dataset (Bi)
4) Design of the "A-Cell" portable SIMD framework for heterogeneous computing, development and release of a simulator for visualization and analysis of A-Cell system, and partial development of A-Cell source-to-source compiler (Khandan)

3. Research Results and Achievements

3.1. Data centric approach for simulation and post-processing

Since simulations executed on the K computer often generate large-scale and numerous files, once the data files are generated it is hard to copy or move the data owing to the limitation of the storage. Thus, it is essential for us to manage such many distributed files. We developed an efficient file management library CIOlib using meta-data information so that the library allows us to share the data between different applications such as simulator and post-processing programs. The meta-data is described simple YAML like notation and consists of two types of the information: information related to the parallel process and information related to computed results. This information is described in proc.dfi and index.dfi files respectively. One of the useful functions in developed CIOlib is file loading. The number of execution processes

may vary in some cases, for instance, between simulation and data processing/visualization, or the restart process in simulation. In many cases, file-based data processing after simulation employs much fewer processes than the simulation itself. We consider and provide various file loading patterns for the user's convenience. Figure 1 shows an example of a file loading function that CIOlib provides.

The information included in the proc.dfi can automatically process the following patterns.

a) Normal Loading

The number of processes and the situation of partitioning are the same between the previous and the current session. Most cases choose this pattern.

b) Refinement Loading

In this refinement case as shown in Figure 1, the number of processes is the same; however, the resolution is different. Each process in the current session reads the same file that the process has written in the previous session. However, because the resolution is doubled, interpolation should be performed. User can overload the interpolation method.

c) MxN Loading

M and N denote the number of processes in the previous and current sessions, respectively. Even if M and N are equal but the partitioning is different, the loading pattern becomes MxN loading. Figure 1 shows this pattern for the same resolution, and Figure 1 demonstrates the case of different resolutions and different partitioning. This loading pattern is considerably useful when the user changes the number of processes in a successive simulation process.

d) Staging Assistance

In certain computational environments, the files required for computation need to be transferred from the global file system to the local file system, which is a dedicated disk partition for computations in a batch job, before the calculation begins. This file transfer process is called a stage-in. In the MxN file loading pattern, each current process may need different files that the process wrote in the previous session. In this case, the file that the process needs must be determined prior to submitting a batch job. This procedure is also performed using proc.dfi information. Figure 1 depicts an example. The previous session has M = 9 processes and the current session has N = 4. For instance, the current process of rank 0 (N{0}) demands M{0, 1, 3, 4}. This relationship can be obtained from the information described in proc.dfi.

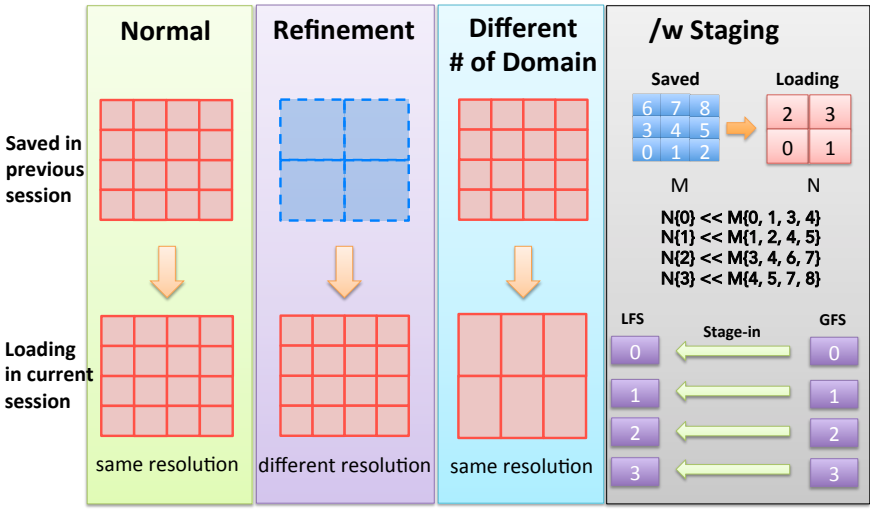


Figure 1. Four file I/O patterns provided by CIOlib.

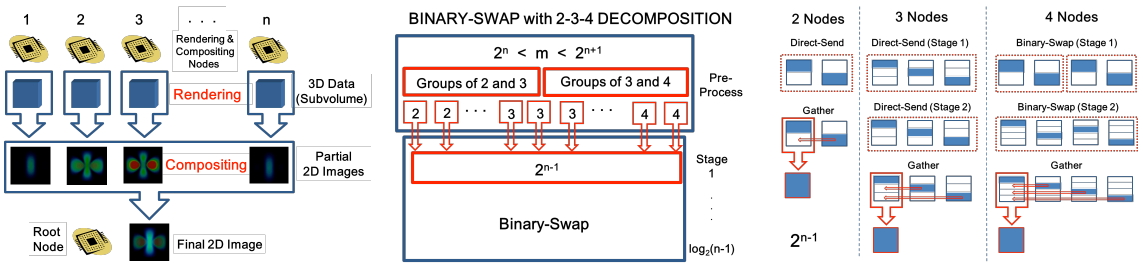
3.2 Development of visualization technique for large-scale dataset

The most popular approach for visualization of large datasets is the sort-last parallel visualization pipeline depicted in Fig. 2a. Most of parallel rendering algorithms are embarrassingly parallel in nature, however the images generated at each rendering node should be composited together to generate the final image. Since it requires interprocess communication among the entire nodes, it usually dominates the total cost of a parallel rendering process. As shown in Fig. 2a, image composition process works coupled with the rendering process, and the same node used for rendering is also used for composition. As a result, the number of rendering nodes determines the number of composition nodes. Although the number of rendering nodes depends on several factors, it is highly important to have a scalable image composition algorithm that works with arbitrary number of nodes.

Parallel image composition has been studied for almost two decades and several algorithms have been proposed so far. Currently, there exist efficient image composition algorithms for power-of-two (2^n) number of nodes. However, when handling non-power-of-two number of nodes, an additional processing is usually applied causing performance penalty. The simplest way is to execute as a pre-processing converting to a power-of-two number of nodes. Since it can be costly, another approach is to distribute this processing overhead to the entire parallel image composition process. As a lightweight pre-processing approach, we investigated the use of 2-3-4 Decomposition method to convert a given non-power-of-two number of nodes to a power-of-two number of nodes. It is worth noting that, in the case of the K Computer, to make full use of the allocated computational resources, multiples of 12 (non-power-of-two) might be

appropriate for job execution.

The 2-3-4 Decomposition works by creating groups of 2, 3 or 4 nodes as shown in Fig. 2b. Considering a non-power-of-two number of composition nodes (m) which is bounded by two power-of-two values, $2^n < m < 2^{n+1}$, when applying 2-3-4 Decomposition, it will create exactly 2^{n-1}, that is, a power-of-two number of groups. By independently compositing each of these groups, as shown in Fig. 2c, it will generate a power-of-two number of images, and by using the local root node at each of these groups it will be possible to apply any of the existing image composition nodes for power-of-two number of nodes. The maximum performance penalty is limited to the overhead of compositing four images. In the case of using Binary-Swap image composition, which is probably the most popular image composition method for power-of-two number of nodes, we verified a stable performance penalty in the range of 1.5 to 1.75 times as those of Binary-Swap on a large-scale image composition environment of up to 32K thousands of composition nodes. It is worth noting that this decomposition method does not depend on any composition parameter to be specified by the user, and has predicable performance behavior. Our ongoing work includes a study to further reduce this performance penalty in certain circumstances.



a) Sort-last parallel visualization pipeline. b) An application of 2-3-4 Decomposition. c) Composition of 2-3-4 groups

Figure 2. 2-3-4 decomposition in sort-last approach.

3.3 Data compression for large-scale dataset

In this year, we have proposed two methods for data compression for large-scale dataset.

Firstly, we proposed a parallel data compression method using POD (Proper Orthogonal Decomposition). However, this method can only deal with the cases that both the number of time steps and the number of processors are power-of-two. Therefore, a 2-3-4 combination method was proposed to resolve the problem of non-power-of-two.

3.3.1. A Study of Parallel Data Compression Using Proper Orthogonal Decomposition

The growing power of supercomputers continues to raise scientists' ability to model larger, more sophisticated problems in science with higher accuracy. Equally important is the ability to make full use of the data output from the simulations to help clarify the modeled phenomena and facilitate the discovery of new phenomena. However, as the scale of computing has grown, there has been a corresponding growth in available data. Outputting most of this data is simply not feasible, which defeats the purpose of conducting large-scale simulations. In order to address this issue so that more data may be collected and used for extreme-scale computing, we have developed a scalable parallel data compression solution to reduce the size of large-scale data with low computational cost and minimal error. We use the proper orthogonal decomposition (POD) method to compress data because this method can effectively extract the main features from the data, and the resulting compressed data can be linearly decompressed. Our implementation achieves high parallel efficiency with a binary load-distributed approach (as shown in Figure 3), which is similar to the binary-swap image composition method. This approach allows us to effectively use all of the processing nodes and to reduce the interprocessor communication cost throughout the parallel compression calculations. The results of tests using the K computer indicate the superior performance of the proposed design and implementation.

3.3.2. 2-3-4 Combination for Parallel Compression on the K computer

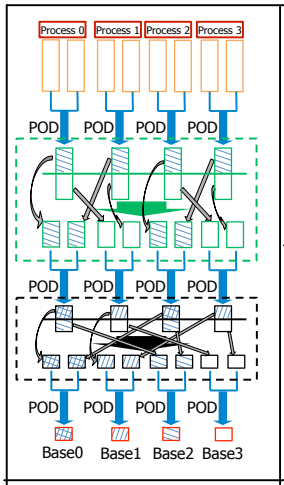


Figure 3 Parallel POD compression using our proposed binary load-distributed method.

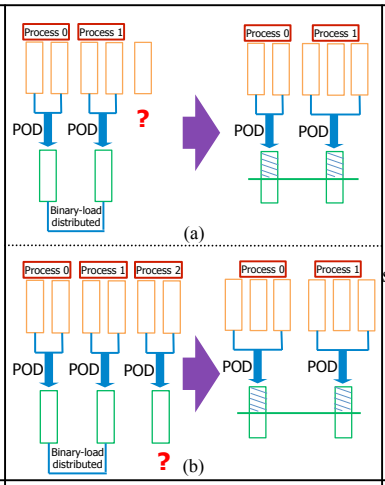


Figure 4 Two cases of non-power-of-two. (a) The number of time steps. (b) Both the number of the time steps and the processors.

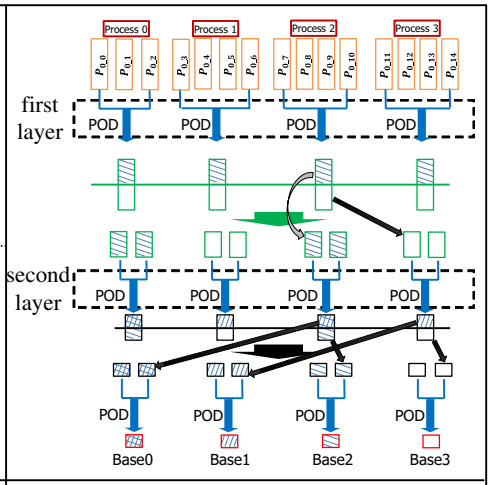


Figure 5 The data with 15 time steps are efficiently parallel compressed using our 2-3-4 combination approach. The best load balance was achieved.

In our previous research, a parallel compression method has been proposed. However, in this algorithm, interprocessor communication can only be carried out in the case that both the number of time steps and processors are power-of-two (the case in Figure 3). Obviously, it is

impossible for the general cases, Figure 4 shows two non-power-of-two cases. A method that can fully resolve this problem with low computational cost will be very popular. In this research, we proposed such an approach called 2-3-4 combination approach, which can be simply implemented and also reach high performance of parallel computing algorithms. Furthermore, our method can obtain the best balance among all parallel computing processors. This is achieved by transferring the non-power-of-two problem into power-of-two problem to fully use the best balance feature of the binary load-distributed method proposed in our first research. Figure 4 shows how to transfer the two non-power-of-two cases into power-of-two. Figure 5 is an example to effectively compress the data with 15 time steps through using our 2-3-4 combination method. We evaluate our approach through applying it to the parallel POD compression algorithm on the K computer.

3.4 A-Cell

An important mission of advanced visualization research team is to create reusable program modules that could be used by AICS and the rest of HPC community. Our projects include physics simulation and rendering. In addition we have lunched the new trend of knowledge discovery in simulation data. In order to make our solutions available for larger portion of the community our solutions should be completely portable and scalable to be more reusable. Therefore we decided to focus on the fundamental problem of portability and scalability. Other than the K computer, the visualization team uses a specific-purposed visualization cluster with heterogeneous nodes with two high-end CPUs and one GPU per node. It is desirable that our programs to be executable on both K computer and visualization cluster without modification or additional setup. Furthermore, it is desired that our programs to be executed without modification or additional setup on the next generation exa-scale supercomputer which its exact architecture is not currently determined yet.

Unfortunately, available solutions like OpenACC, OpenMP, PGAS, and Unified Memory in CUDA C only make the problem deeper. These technologies try to present the programmer with an abstraction of the system that looks and feels like a sequential programming environment. Then, the compiler will be responsible for creating a parallel program out of user code. However, most compilers fail to perform that task properly. In addition, compile-time parameters differ from one machine to another. That makes the program less portable. Furthermore, none of these technologies are portable between multi-core to many-core architectures and vice versa.

We came up with a promising solution that we call A-Cell (asynchronous cells). A-Cell is a high-level abstraction of fine-grained parallelism specifically designed to be applicable to all range of parallel devices from super computers based on CPUs or GPUs, to network of

embedded devices. To achieve this, A-Cell adopts a programming model called "connectionist computing" and with that takes a leap step farther from Turing programming model compared to aforementioned solutions. Also, in contrast with those other solutions that are based on holistic philosophy, the philosophy of A-Cell is reductionist. An A-Cell encapsulates a fine-grained task with its related variables and data linkage. A source-to-source compiler translates the program to a set of programs that are compilable to the target devices. Execution of the task is through massive instantiation of an A-Cell. The runtime environment takes the responsibility of distributing A-Cell instances between all available nodes, CPU cores or GPU multiprocessors (MPs). The runtime environment also assures synchronization and consistency of data between A-Cells.

With A-Cell two important inherent properties emerge, namely, Spatial Elasticity and Temporal Elasticity. Spatial Elasticity makes A-Cells to fit in any heterogeneous computing environment while Temporal Elasticity improves execution time by compensating for a part of interconnect latency and the time wasted by unexpected system interrupts. Development of A-Cell compiler based on LLVM/Clang compiler system has been started and it is about 20% through. A simulator is developed as a proof of concept for A-Cell scalability and its better time performance. The simulator is released to public and is available as an open-source project.

When the A-Cell compiler, runtime system becomes available, we will be able to produce visualization and knowledge discovery solutions that are portable between varieties of HPC environments and scalable to arbitrary number of nodes.

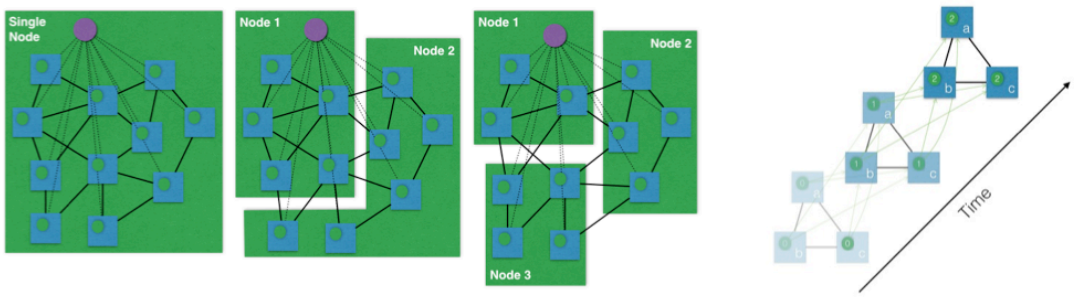


Figure 6. Left: Spatial Elasticity, Right: Temporal Elasticity in A-Cell.

4. Schedule and Future Plan

We will release our visualization software around mid. of FY2014, and have a plan to give lectures related our released software. Research themes and developed software will be investigated further in next year continuously, for instance, (1) Completion and release of

KnoRBA + A-Cell precompiler. KnoRBA is an original agent-based software development framework that can be used for development of a scalable, extensible and portable knowledge discovery solution, (2) Implementation of a demo user interface for wall-mounted display, and (3) To proceed with research and implementation of multi-modal knowledge discovery and data minding using KnoRBA and A-Cell.

5. Publication, Presentation and Deliverables

(1) Journal Papers

[1] Ono, K., Kawanabe, T., and Hatada, T., "HPC/PF - High Performance Computing Platform: An Environment That Accelerates Large-Scale Simulations, "Lecture Notes in Computer Science, Vol. 7851, pp. 23-27, 2013.

(2) Conference Papers

[2] Khandan, H., and Ono, K., "Knowledge Request-Broker Architecture: A Possible Foundation for A Resource-Constrained Dynamic and Autonomous Global Network", in *Proceedings of 2014 IEEE World Forum on Internet of Things*, ISBN 978-1-14799-3460-7, March 2014, Seoul, South Korea.

[3] Bi, C., and Ono, K., "2-3-4 Combination for Parallel Compression on the K Computer," in Proceedings of IEEE Pacific Visualization Symposium, pp. 281-285, March, 2014.

[4] Onishi, J. and Ono, K., "Development of a CFD software for large-scale computation," FLUCONME 2013, Nov. 18-23, 2013.

[5] Ono, K. and Mikami, K., "High-performance application development towards exa-scale computation" in Proceedings of 27th Computational Fluid Dynamics Symposium, Dec. 2013.

(3) Invited Talks

[6] Ono, K., "Strategy of Visualization toward Exascale Computing," PC cluster consortium workshop in Kyoto, Jan. 2013.

[7] Ono, K., "Life cycle management of big data for extreme-scale simulation," Big data and extreme-scale computing, Charleston, SC, USA, April, 2013.

[8] Ono, K., "Technology Infrastructure for Scientific Applications with Large-Scale Dataset," The 4th AICS International Symposium: Computer and Computational Science for Exascale Computing, Dec. 2013.

[9] Ono, K., "Challenge for parallel visualization of large-scale dataset and current status," Prospects for Massively Parallel Computing in Science and Technology Calculation - Use, Application and Problems, Tokyo Metropolitan University, March, 2014.

(4) Posters and Presentations

[10] Khandan, H., "Connectionism: How the World Really Computes", Oral and poster presentation at *11th Interdisciplinary Evening*, RIKEN Wako Campus, March 2014.

[11] Bi, C. and Ono, K., "Parallel POD Compression Accelerated by Binary-Swap Compositing," in Proceedings of Symposium on Advanced Computing Systems and Infrastructures 2013, pp. 112, May, 2013.

[12] Bi, C., Ono, K., Ma, K.-L., Wu, H., and Imamura, T., "Proper Orthogonal Decomposition Based Parallel Compression for Visualizing Big Data on the K Computer," in Proceedings of IEEE Symposium on Large-Scale Data Analysis and Visualization 2013, pp. 121-122, October, 2013.

[13] Bi, C. and Ono, K., "POD-Based Parallel Compression for Visualizing Large-Scale Dataset," Proceedings of High Performance Computing Symposium 2013, p.83, 2013.

(5) Patents and Deliverables

[14] CPMlib Cartesian Partitioning Manager library

[15] PMlib Performance Monitor library

[16] TextParser Parameter parsing library

[17] CIOlib Cartesian file I/O library