HIVE: A Visualization and Analysis Framework for Large-Scale Simulations on the K Computer

Jorji Nonaka; Kenji Ono†
Chongke Bi, Daisuke Sakurai
RIKEN AICS
Kobe, JAPAN

Masahiro Fujita†
Light Transport Entertainment
Tokyo, JAPAN

Kentaro Oku‡
IMAGICA DIGITALSCAPE
Tokyo, JAPAN

Tomohiro Kawanabe¶
The University of Tokyo
Tokyo, JAPAN

ABSTRACT

In this poster, we will present a software framework for visualization and analysis of large-scale simulation data generated by the K Computer, a Japanese flagship-class supercomputer installed at RIKEN AICS. This framework was named HIVE (Heterogeneously Integrated Visual analytics Environment) by taking into consideration the heterogeneous hardware and software environment found on traditional HPC (High Performance Computing) infrastructure consisting of supercomputers, visualization-oriented clusters, and front-end machines. HIVE has been designed to be capable of running on SPARC64fx-class processors, used by the K Computer and derived commercial versions. In these HPC systems, the software implementation of OpenGL graphics library, such as the well-known Mesa is not officially supported, and to enable and facilitate the cross-platform execution, HIVE adopted an OpenGL ES 2.0 (OpenGL for Embedded Systems) compatible API, and a loosely coupled architecture for the module integration. In the poster, we will present an overview of the software architecture showing the already implemented features and possible future directions.


1 INTRODUCTION

Flagship-class supercomputers have been designed to achieve the maximum performance in floating-point operations, and can have specialized hardware architecture and software environment. In addition, each supercomputer system may have different operational policies and restrictions, inherent to the installed site, which can bring some difficulties even to run the same application at different sites. The K Computer [5] is a massive CPU-only supercomputer with more than 80,000 SPARC64fx-class processors interconnected with a specialized 6D Mesh/Torus topology network named Tofu (Torus fusion). It uses separate file system, for offline data storage and simulation runs, which imposes some restrictions to access the intermediate data during the job execution, thus making difficult to implement some of the traditional In-situ processing scenarios.

Although there already exist two derived commercial versions of the K Computer (Fujitsu PRIMEHPC FX10 and FX100), this system still remains as the fastest supercomputer in Japan, and has been used to run large-scale simulations in a wide range of application domains in science and engineering. The In-Situ processing [1] becomes more a necessity than just an option for tackling these large simulation results for the visualization and analysis purposes. In such situation, the use of supercomputer resource for executing the entire visualization pipeline [6] should be taken into consideration. However, in the case of the K Computer, software implementation of OpenGL library, such as the well-known Mesa 3D graphics library is not officially supported, and the promising OpenSWR software rasterizer has been designed for x86-class processors and accelerators. The hardware developer (Fujitsu) has only provided a so-called Visualization Library [8], which only supports a limited set of data formats, and solely a rendering technique based on Particle-based Volume Rendering [9]. To meet the visualization and analysis needs from the K Computer users, we have been developing a visual analytics framework capable of running directly on the K Computer, which will be described in the next section.

2 HIVE

HIVE has been designed to be capable of running on the heterogeneous hardware environment of traditional HPC infrastructures, which can include visualization-oriented clusters and front-end machines, along with the supercomputers, such as shown in Figure 1. Considering the possible combinations of the aforementioned hardware systems for the In-Situ processing scenarios, the components of the visualization pipeline (For example: Read, Filter, and Render) must be capable of running on different hardware systems at different scales. It includes the batch-mode visualization on the supercomputer side (In-situ, Co-processing approach) as well as the interactive mode on the local machine side (In-situ, Concurrent and Hybrid approaches). To facilitate the implementation of both execution modes, HIVE adopted a loosely coupled architecture for the module integration by using the Lua scripting language [3]. Figure 1 shows an overview of the software stack, where we can verify some of the main modules, which includes the Loader, Builder, SURFACE, Browser UI, and HIVE Renderer.

Figure 1: Overview of the target running environment and the HIVE software stack including the modules of the visualization pipeline.

*e-mail: jorji@riken.jp †e-mail: keno@riken.jp ‡e-mail: syoyo@lighttransport.com ¶e-mail: kentaro-o@digirea.com §e-mail:tkawanab@iis.u-tokyo.ac.jp
As shown in the Figure 2, the HIVE Renderer is responsible for calling and dispatching the necessary modules during the batch and interactive operational modes. The main components of the HIVE will be detailed in the following sub-sections.

2.1 xDMlib (Data Management Library)

The diverse simulation results generated by the K Computer are the primary target of the HIVE, and for this purpose, we have been developing a data management library named xDMlib. The “x” portion represents the different categories of the data formats, which include Cartesian (CDMlib), Unstructured (UDMlib), Hierarchical (HDMLib), and Particle (PDMlib). The main characteristics of this library is its data migration functionality which enables flexible distributed data load/save including 1-to-N, N-to-1, M-to-N, and N-to-M configurations along with the traditional N-to-N configuration, when using N data loaders/savers. This feature greatly facilitates the data handling when the number of data readers do not match with the number of distributed files generated by the simulation. In order to minimize the data replication, only a lightweight metadata, with the data migration information, is required for the on-the-fly data repartitioning of single or distributed simulation data stored in spatial, temporal or spatiotemporal configuration.

2.2 SURF ACE (Raytracer)

After the data loading, the functionalities corresponding to the Filter nodes, in the visualization pipeline, will be executed by the Builder components of the software stack. Currently, only a small set of functionalities have been implemented, and the loosely coupled architecture of the HIVE module facilitates the integration of new modules including the user developed ones. The Render node in the visualization pipeline corresponds to the raytracing based rendering module named SURF ACE (Scalable and Ubiquitous Rendering Framework for Advanced Computing Environments). This module has been developed from the LSGL (Large-Scale Graphics Library for Peta-Scale Computing Environments) [2], which has been optimized to run on SPARC64fx processors. To facilitate the cross-platform compilation and execution, SURF ACE adopted an OpenGL ES 2.0 compatible API, and to enable the user extendibility of visual representation, JIT (just-in-time) compilation of fragment-level GLSL (OpenGL Shading Language) shader codes has also been implemented. Figure 2 shows some rendering results applying the user-level custom fragment shaders.

For the parallel visualization, HIVE adopted the Sort-last rendering approach where the parallel rendered images distributed among the rendering nodes have to be merged to generate the single final image. Considering that the SURF ACE is capable to render using the full computational nodes of the K Computer (82,944 nodes), the scalability requirement for the parallel image compositing was also in the similar order of magnitude. Taking this into consideration, we have developed a scalable parallel image composition, named 234Compositor [7], which is based on the Binary-Swap [4] parallel image composition approach. It includes some extensions to enable the usage with non-power-of-two number of nodes, and some optimizations to maximize the usage of multi-core architecture by taking advantage of the Hybrid MPI-OpenMP parallelism, and optimizations to minimize the performance degradation of the final image gathering process. The scalability was confirmed using the full 82,944 computational nodes, as the image composition nodes, in the Hybrid MPI-OpenMP mode.

2.3 Browser-UI (User Interface)

HIVE adopted the Web-based UI to facilitate the multi-platform access and usage. It can be used in both standalone and client/server modes, and currently the HIVE provides two applications as shown in Figure 2. The Scene Node Editor enables an intuitive manipulation of the visualization workflow, and the HIVE_UI can assist the preparation of key-frame animations. Both applications have the functionalities to export the visualization scenarios (Scene Files) as Lua script files, which can be interpreted by any other HIVE Renderers running at different machines. Since it is just a Lua script, the users are allowed to modify the exported scene file to meet the visualization and analysis needs.

3 Conclusion

In this poster, we outlined a software framework for large data visualization and analysis which can run on a HPC infrastructure, which includes supercomputers with SPARC64fx-class processors. The adopted loosely coupled architecture is expected to facilitate further module integration to enhance and enrich the visualization and analysis capabilities to meet the K computer users’ needs.

Acknowledgements

Part of the results was obtained by using the K Computer at the RIKEN AICS (Advanced Institute for Computational Science) in Kobe, Japan.

References