

HIVE: A Cross-Platform, Modular Visualization Ecosystem for Heterogeneous Computational Environments

Jorji Nonaka
RIKEN R-CCS
Kobe, Japan
jorji@riken.jp

Kenji Ono
Kyushu University
Fukuoka, Japan
keno@cc.kyushu-u.ac.jp

Naohisa Sakamoto
Kobe University
Kobe, Japan
naohisa.sakamoto@people.kobe-u.ac.jp

Kengo Hayashi
Kobe University
Kobe, Japan
171x219x@stu.kobe-u.ac.jp

Tomohiro Kawanabe
RIKEN R-CCS
Kobe, Japan
tkawanabe@riken.jp

Fumiyoshi Shoji
RIKEN R-CCS
Kobe, Japan
shoji@riken.jp

Masahiro Fujita
LTE Inc.
Tokyo, Japan
syoyo@lighttransport.com

Kentaro Oku
KASHIKA
Tokyo, Japan
oku@kashika.co.jp

Kazuma Hatta
IMAGICA DIGITALSCAPE
Tokyo, Japan
kazuma-h@digirea.com

ABSTRACT

HPC operational environments usually have supporting computational systems for assisting pre- and post-processing activities such as the visualization and analysis of simulation results. A wide variety of hardware systems can be found at different HPC sites, and in our case, we have a CPU-only (x86) large memory server, and there is a plan to replace this with a modern OpenStack-based CPU/GPU Cluster. HPC systems themselves can also be used for executing visualization related processing when applying the in-situ approach, and in our case this will be a SPARC64 fx CPU based HPC system (K computer). It is publicly announced that the current system will be replaced with an ARM based HPC system in a near future. Therefore heterogeneity and scalability are needed to be tackled in order to efficiently use these heterogeneous computational resources for large-scale data visualization on both post-hoc and in-situ contexts. In this poster we present HIVE (Heterogeneously Integrated Visual-analytics Environment), a cross-platform and modular ecosystem for providing visualization service building blocks in such heterogeneous computational environments. Lightweight Lua scripting language is used to glue necessary visualization pipeline related modules, and this loosely coupled modular approach facilitates the long-term development and maintenance.

CCS CONCEPTS

• **Human-centered computing** → Visualization systems and tools; • **Computing methodologies** → Rasterization; Ray tracing;

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SC18 Poster, November 2018, Dallas, Texas USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06.

https://doi.org/10.475/123_4

KEYWORDS

Visualization, Lua, Browser user interface, K computer

1 INTRODUCTION

High performance visualization [1] has played an important role in assisting scientists and engineers to visually explore and get insights into their large-scale simulation results generated on HPC systems. Scientific Visualization [2] has matured over the past three decades, and OpenGL (Open Graphics Library) [3] and VTK (Visualization Toolkit) [15] have become the de facto standard in the visualization community. In addition, for high performance visualization, the VTK-based ParaView [4] and VisIt [5] have become the main choice for post-hoc and in-situ visualization. Ray-tracing based visualization has recently been gaining increasing attention [16], and we can cite the Brayns [12], which is a hardware agnostic ray-tracer that can use heterogeneous hardware resources (CPU/Accelerator and GPU). Mesa3D graphics library is an OSS (Open Source Software)

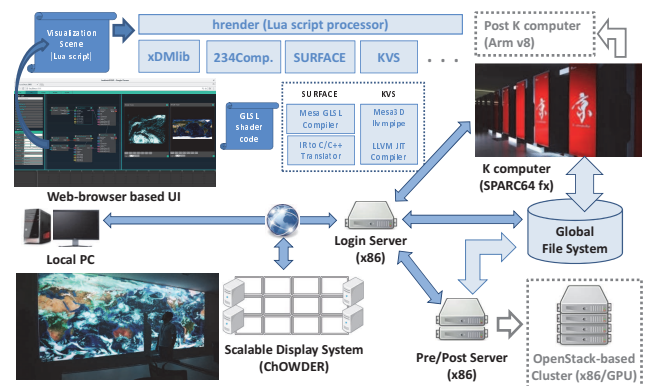


Figure 1: Overview of the HIVE ecosystem on the K computer operational environment (current and planned).

implementation of OpenGL, which has become the de facto standard for enabling the use of OpenGL on CPU-only supercomputing systems. However, Mesa 3D graphics library was not supported on SPARC64 fx based supercomputers such as the K computer and Fujitsu PRIMEHPC FX family of supercomputers. Considering the necessity of running visualization related tasks on such heterogeneous computational environments, we have developed a cross-platform and modular visualization ecosystem, named HIVE [8], which was designed to be easy to enhance and maintain thanks to the loosely coupled modular approach via lightweight Lua scripting language.

2 HIVE VISUALIZATION ECOSYSTEM

On the K computer operational environment, the hardware vendor (Fujitsu) did not provide the Mesa 3D graphics library, and instead they provided a custom and non-OSS visualization library [10] which implements a non-OpenGL version of PBVR (Particle Based Volume Rendering) [14]. Although it is useful for some specific needs, a more general purpose visualization application was desired to meet a variety of visualization needs. This kind of particular library dependency can severely interfere in the continuous development of an application system. Trying to avoid such interference as much as possible, we have worked on a visualization ecosystem for running on heterogeneous computational environments. Visualization tasks can be expressed as a processing pipeline [6], and usually the core tasks such as data loading, filtering, mapping, rendering, and image compositing can be decoupled from each other and have producer/consumer relationships. Therefore, we focused on a large-scale data visualization ecosystem which can easily be enhanced and maintained by coupling and decoupling the visualization pipeline related modules, as a plugin DLL, via scripting language. Python is probably the de facto standard scripting language for HPC community; however we selected the Lua scripting language due to its portability and especially its compactness. Figure 1 shows an overview of the HIVE ecosystem on the K computer operational environment, which works in server/client model for interactive visualization and in a standalone mode for batch job based large-scale visualization. For the interactive visualization, Node.js is utilized for providing a Web browser based visualization workspace for interactive visual exploration and visualization scene settings.

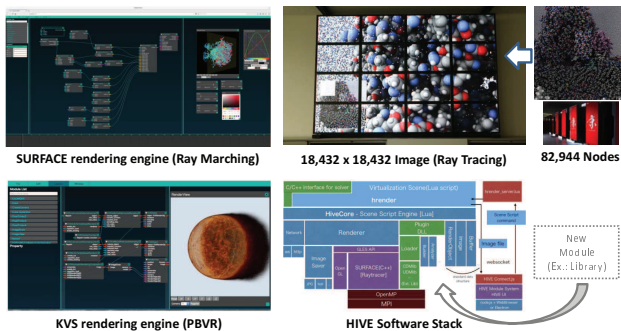


Figure 2: Modular design of the HIVE ecosystem

2.1 Modular Design

HIVE adopted modular design approach for integrating own developed as well as third party tools and libraries in order to facilitate the functionality enhancements as well as the maintainability. Figure 2 shows an overview of the software stack of HIVE with some of the currently integrated tools and libraries. Most of the libraries and tools have been written using C and C++ language, and the visualization pipeline related functionalities are provided to the users as a Lua-based API. JSON has been used to provide the Web browser visualization workspace, and Figure 2 shows two variants of the visualization workspace which uses different rendering engines, SURFACE (Scalable and Ubiquitous Rendering Framework for Advanced Computing Environments) and (Kyoto Visualization System) [13]. Websocket is used for the communication between the HIVE rendering module and the Web-browser based UI, and this enables interactive visual exploration using remotely situated computational resources and datasets. Visualization scenes prepared in the GUI workspace can be exported as a Lua script which can be useful for executing visualization in batch mode. This offline rendering capability can be used to render medium and large datasets on clusters as well as directly on the supercomputers. Figure 2 shows a high-resolution rendering result, which was output as a $18,432 \times 18,432$ image file, using the full computational nodes (82,944) of the K computer.

In this current implementation of HIVE, it utilizes the xDMLib data management library [7, 11] for data loading. It uses lightweight metadata for run-time data migration to support flexible $M \times N$ data loading, where M can be $1 \leq M \leq N$, and N can also be 1. Rendering can be considered the main functionality in a visualization application, and HIVE supports both OpenGL (KVS) and non-OpenGL (SURFACE) rendering. Both rendering engines support GLSL (OpenGL Shading Language) codes and uses Mesa compiler in addition to an own developed translator (intermediate representation to C/C++ code) (SURFACE), or the default LLVM-JIT compiler (KVS). Sort-last parallel rendering requires image compositing to generate the final image, and HIVE utilizes the 234Compositor library [9], a scalable and flexible parallel image compositor framework for massively parallel rendering applications.

3 CONCLUSIONS

Although it has not been tested in a production level environment yet, a HIVE-based scalable visualization environment is being prepared at the operational division of the R-CCS in order to provide visualization services for the K computer users. We are aware that theory and practice can be different; however we expect that the user feedbacks can accelerate the software refinement for improving its use in real usage scenarios. We hope that this visualization ecosystem would be beneficial for preparing the necessary visualization and analysis environment for the Post-K supercomputer.

ACKNOWLEDGMENTS

Some of the results were obtained by using the K computer at RIKEN R-CCS (Center for Computational Science) in Kobe, Japan. This work has been partially supported by the “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” in Japan (Project ID: jh180060-NAH).

REFERENCES

- [1] E. Wes Bethel, Hank Childs, and Charles Hansen. 2012. *High Performance Visualization: Enabling Extreme-Scale Scientific Insight* (1st ed.). Chapman & Hall/CRC.
- [2] Christopher Johnson and Charles Hansen. 2004. *Visualization Handbook*. Academic Press, Inc., Orlando, FL, USA.
- [3] Khronos Group, Inc. [n. d.]. OpenGL Overview. <https://www.khronos.org/opengl>.
- [4] Kitware, Inc. [n. d.]. ParaView. <http://www.paraview.org>.
- [5] LLNL (Lawrence Livermore National Laboratory). [n. d.]. VisIt. <http://wci.llnl.gov/simulation/computer-codes/visit>.
- [6] Kenneth Moreland. 2013. A Survey of Visualization Pipelines. *IEEE Transactions on Visualization and Computer Graphics* 19, 3 (2013), 367–378.
- [7] Jorji Nonaka, Eduardo C. Inacio, Kenji Ono, Mario A. R. Dantas, Yasuhiro Kawashima, Tomohiro Kawanabe, and Fumiyoshi Shoji. 2018. Data I/O management approach for the post-hoc visualization of big simulation data results. *International Journal of Modeling, Simulation, and Scientific Computing* 09, 03 (2018), 1840006.
- [8] Jorji Nonaka, Kenji Ono, Chongke Bi, Daisuke Sakurai, Masahiro Fujita, Kentaro Oku, and Tomohiro Kawanabe. 2016. HIVE: A Visualization and Analysis Framework for Large-Scale Simulations on the K Computer, IEEE Pacific Visualization 2016 (Poster).
- [9] Jorji Nonaka, Kenji Ono, and Masahiro Fujita. 2018. 234Compositor: A flexible parallel image compositing framework for massively parallel visualization environments. *Future Generation Computer Systems* 82 (2018), 647 – 655.
- [10] A. Ogasa, H. Maesaka, Sakamoto K., and S. Otagiri. 2012. Visualization Technology for the K Computer. *Fujitsu Scientific and Technical Journal* 48, 3 (2012), 348–356.
- [11] Kenji Ono, Yasuhiro Kawashima, and Tomohiro Kawanabe. 2014. Data Centric Framework for Large-scale High-performance Parallel Computation. *Procedia Computer Science* 29 (2014), 2336 – 2350.
- [12] The Blue Brain Project. [n. d.]. Brayns: Visualizer for large-scale and interactive ray-tracing of neurons. <https://github.com/BlueBrain/Brayns>.
- [13] Naohisa Sakamoto and Koji Koyamada. 2015. KVS: A Simple and Effective Framework for Scientific Visualization. *Journal of Advanced Simulation in Science and Engineering* 2, 1 (2015), 76–95.
- [14] Naohisa Sakamoto, Jorji Nonaka, Koji Koyamada, and Satoshi Tanaka. 2007. Particle-based Volume Rendering. In *Proceedings of the IEEE Asia-Pacific Symposium on Visualization*. 129–132.
- [15] Will Schroeder, Ken Martin, and Bill Lorensen. 2006. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc.
- [16] I. Wald, G. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Günther, and P. Navrátil. 2017. OSPRay - A CPU Ray Tracing Framework for Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 931–940.