The Gap between Visualization Research and Visualization Software (VisGap) (2020)
C. Gillmann, M. Krone, G. Reina, T. Wischgoll (Editors)

Lessons Learned from Large Data Visualization Software Development for the K computer

Jorji Nonaka[1] and Naohisa Sakamoto[2,1]

[1]RIKEN R-CCS, Operations and Computer Technologies Division, Japan
[2]Kobe University, Graduate School of System Informatics, Japan

Abstract

High Performance Computing (HPC) always had a close relationship with visualization as we can remember the landmark report on "Visualization in Scientific Computing", which was credited to have coined the term Scientific Visualization (SciVis). K computer, a Japanese flagship HPC system, appeared in 2011 as the most powerful supercomputer in the Top500 list, and as other similar HPC systems in that ranking, it was designed to enable "Grand Challenge" scientific computing with unprecedented scale and size. RIKEN Center for Computational Science (RIKEN R-CCS) operated and provided the K computer's computational resources to the HPC community for almost 8 years until it was decommissioned in 2019. Considering that most of the scientific computing results were publicly presented in the form of visual images and movies, we can infer that the SciVis was widely applied for assisting the domain scientists with their end-to-end scientific computing workflows. In addition to the traditional visualization applications, various others large data visualization software development were conducted in order to tackle the increased size and amount of the simulation outputs. RIKEN R-CCS participated in some of these development and deployment dealing with several environmental and human factors. Although we have no precise statistics regarding the visualization software usage, in this paper, we would like to present some findings and lessons learned from the large data visualization software development in the K computer environment.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools;*** • ***Applied computing*** → *Physical sciences and engineering;* • ***Computing methodologies*** → *Parallel computing methodologies;*

1. Introduction

RIKEN R-CCS is a leadership-class Japanese HPC Center, established in 2010, and has led the co-development of the two most recent Japanese flagship supercomputers, the *K computer* decommissioned in 2019, and *Fugaku* expected to be operational in 2021. The K computer was a SPARC64-based HPC system, and originated two generations of commercial HPC systems based on such CPU architecture (Fujitsu PRIMEHPC FX-10 and FX-100), which were installed at different HPC sites throughout Japan. RIKEN R-CCS was responsible to operate and to provide computational resources to the designated HPC users, or more specifically, to the Japanese HPCI (HPC Infrastructure) user community. It is worth noting that we just provided the computational resources with no control about how these resources were used. Thus we have no precise statistics about the software utilized by the users. We gathered some information from different sources, and found that several efforts have been done on the porting and development of different large data visualization applications, libraries, and tools. We also participated in the development and deployment of some of them, and we listed, in Fig. 1, some of the external factors that influenced in the visualization software development process.

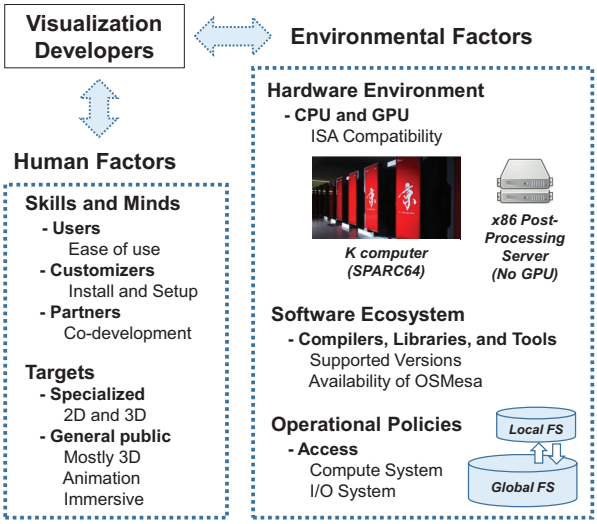

Figure 1: *Some examples of human and environmental factors that can influence the visualization software development.*

DOI: 10.2312/visgap.20201113

We could perceive that the specialized hardware environment, in addition to the restrictions brought by the operational policy (usually specific to each HPC site), have the potential to impact the smooth visualization software development, thus impeding to provide the required visualization capabilities. For the specific case of the K computer environment, we can cite the CPU architecture (SPARC64) with no ISA compatibility with other traditional CPU; the provided set of customized compilers, libraries, and tools with low version number and without including the OSMesa functionality; the runtime access policy to the compute nodes and I/O system that impedes interactive *in situ* or *in transit* processing capabilities [BAA*16]. We should also take into consideration that different users may have different expectations and goals regarding visualization. We grouped them into three categories: *Users* those just expect the visualization tasks being as simple as possible; *Customizers* as a group of skilled people that works by their own using available software and information; and *Partners* as a group of people who considers the visualization researchers as collaborators for assisting their visualization and analysis tasks.

We probably cannot generalize due to the small universe of samples, but we also observed that the objective of the visualization tasks as well as the target audience of the visualization results have a great influence on the expectations regarding the visualization applications. For instance, there were cases where 2D plots or a simple slice rendering were considered sufficient when targeting academic publications, but CG-like rendering and effects as well as immersive rendering were desired when targeting non-researchers and general public. In this paper, we will focus on the SciVis [McC87] oriented visualization applications for assisting the end-to-end scientific computing workflow, and those that use partially or totally the HPC computational resources [BCH12].

2. K computer Environment

The main characteristic of the K computer is probably the SPARC (Scalable Processor Architecture) [MMMA11] based CPU with no instruction set compatibility with other HPC oriented CPU architectures at that time, such as the IBM PowerPC and Intel x86. The K computer was a massively parallel HPC system composed of 82,944 CPUs (or Nodes) for the computation, and the hardware developer (Fujitsu) provided a customized set of compilers, libraries and tools for the software development, and among them, we can cite the GCC 4.8.5 based Fujitsu compiler suite and the lack of the Mesa library (OSMesa functionality). Instead of providing the Mesa library, Fujitsu opted to provide a "Visualization Library" [OMKO12] by implementing a parallel version of the Object-based Particle Based Rendering (O-PBR) described in [SNKT07], and using the 2-3 Swap [YWM08] parallel image composition (Fig. 2). It provided an AVS field (structured data) and UCD (unstructured data) format data loader for the traditional post-hoc visualization on the K computer, and also provided an in-situ visualization API to enable the integration with the simulation codes. Although we can observe some practical utilization for the in situ visualization, it requires the "Customizers" skills, and the main drawback of this library was that it was provided only as a binary code that impeded further customization or development other than provided via API.

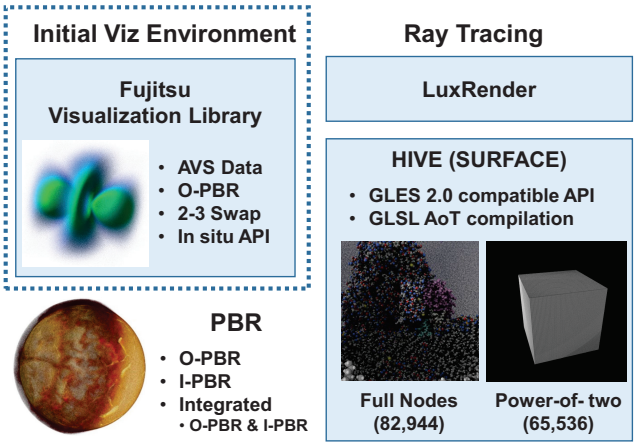

Figure 2: *Initially available visualization environment, and some of the attempted visualization software development.*

3. Actual HPC Users' Needs (Fugaku Environment)

Before discussing some of the large data visualization development done for the K computer environment, we would like to present a more clear picture about the actual Japanese HPC users' needs regarding visualization. It may sound like just an excuse, but we have almost no interaction with the HPC users in daily usage, and the main communication channel was the Help Desk, but the visualization related contacts was almost none, and as a result, we have not much information about the visualization related activities done by the K computer users. However, the Fugaku supercomputer development program made a survey questionnaire, with the Japanese HPCI (High Performance Computing Infrastructure) users, about the list of open source software expected to be available in the future Fugaku HPC environment. Table 1 shows only the visualization related software extracted from the entire list shown in [Ish19]. From this list, we can infer the existence of groups with different expectations and target goals: "Users" group looking for ease of use visualization applications; "Customizers" group looking for libraries and tools for integrating with their simulation codes by taking into consideration the in situ API such as the ParaView-Catalyst [ABG*15], VisIt-libsim [WFM11], and In situ PBR [KNI16]; and the "Developers" group looking for libraries and tools for their own visualization application developments. From this list, we will discuss in the next sections, the PBR and OSMesa based development done for the K computer in addition to the Ray Tracing approach.

Table 1: *Summarized list of the desired visualization oriented open source software for the Fugaku environment.*

Probably "Users"	Probably "Users" and "Customizers"	"Developers"
GrADS	ParaView	VTK
GNUPlot	VisIt	OSMesa
ImageMagick	PBR	MesaGLUT

4. Development for the K computer

During the K computer lifetime, we could observe different attempts to enable large data visualization on this HPC environment, and we can group them into the following three approaches:

- PBR (Stochastic Rendering)
- Ray Tracing
- OSMesa

4.1. Particle Based Rendering (PBR)

For people outside Japan, the presence of the PBR, in Tab. 1, may be a surprise, but the wide adoption of Particle Based Rendering (PBR) [SK12] is probably one of the uniqueness of the Japanese HPC visualization community. PBR is an order-independent stochastic rendering technique suitable for structured and unstructured volume data as well as semi-transparent polygons, and depending on the particle generation approach, it can be divided into O-PBR (Object-space PBR) and I-PBR (Image-space PBR). PBVR (Particle Based Volume Rendering) is the main representative of O-PBR, and SPT (Stochastic Projected Tetrahedra) is the main representative of I-PBR. There is also an integrated approach which combines the O-PBR and I-PBR. We will not enter in the technical details, but the main characteristic of PBR is the use of tiny and opaque particles (without *alpha* or transparency information) which enables order independent processing, thus making highly suitable for massively parallel processing.

Table 2: *PBR-based Applications for the K computer*

Name	Developer
Visualization Library	Fujitsu
Remote Visualization System	JAEA (Japan Atomic Energy Agency)
Distributed PBVR Fully Parallel PBVR	RIKEN R-CCS and Kobe University

Fujitsu explained in [OMKO12] that they implemented the initial version of the PBVR, which is based on the sub-pixel technique. Although we could observe some practical usages, the target users were limited to those working with AVS data format, and requiring only volume rendering. In order to increase the visualization capabilities and functionalities, a more recent O-PBR method based on the repetition technique were implemented by JAEA and RIKEN R-CCS (in collaboration with Kobe University) as shown in Tab. 2. The PBVR-based Remote Visualization System [KIM*15] is a client-server based large data visualization application where the K computer can be used for the particle generation and the local PC is used for interactive visual exploration. We have also worked on a client-server Distributed PBVR system [NSS*17], but we also developed a fully parallel PBVR [YHSN19b] by using the OSMesa functionality [NMS*18] and 234 Compositor [NOF18], a Binary-Swap based image composition with 2-3-4 scheduling.

4.2. OSMesa Approach

Although Mesa library was not officially supported on the SPARC64 system, we found two groups of "Customizers" that have worked on this library in order to enable the use of in-situ visualization via VisIt-libsim (K computer), and ParaView-Catalyst (Fujistu FX-10). We were also interested to use the OSMesa functionalities in order to use the Kyoto Visualization System (KVS) [SK15], a general purpose visualization framework for both post-hoc and in situ visualization. The Fujitsu compiler provided for the SPARC64 users were based on GCC 4.8.5 and was only capable of compiling the legacy *swrast* driver present in the Mesa 9.2.5 as shown in the Tab. 3. However, the *swrast* driver only provides the fixed graphics pipeline functionalities, and the programmable graphics pipeline features are only provided in recent versions (Gallium *softpipe* and *llvmpipe*), and the *llvmpipe* provides better performance thanks to the multi-threading, and also higher capabilities by supporting more recent OpenGL features.

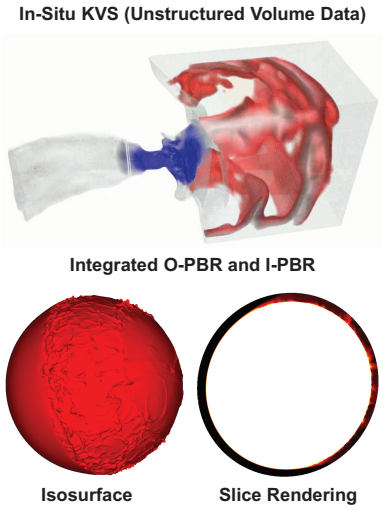

Figure 3: *OSMesa-based post-hoc and in situ KVS visualization examples of unstructured volume data (hexahedra and prism cells).*

Table 3: *Mesa drivers for SPARC64 architecture*

Driver	Version	Compiler
Legacy swrast	9.2.5	Fujitsu
Gallium softpipe	13.0.6	GCC 6.4.0
Gallium llvmpipe	13.0.6	GCC 6.4.0 LLVM 3.9.1

The turning point occurred close to the end of the K computer life cycle, when we could secure a budget for external software development, and increase the list of open source software capable of running on the K computer. This includes the GCC 6.4.0 and LLVM 3.9.1 which enabled the use of Mesa with the *softpipe* and *llvmpipe* drivers. For the latter, the LLVM was necessary since it uses the JIT (Just-in-Time) compilation mechanism for handling the user specified shader codes written in GLSL (OpenGL Shading Language). We can say that this Mesa driver availability facilitated the "Customizers" to work with ParaView-Catalyst and VisIt-libsism after integrating their simulation codes. In addition, it also helped us to develop in situ visualization appli-

cations [HSN*18] [YHSN19a] by working with "Partners". It is worth noting that only the porting task was conducted, and the necessary extensions for taking advantage of the SIMD vectorization provided by the CPU were not implemented.

4.3. Ray Tracing Approach

Another important large data visualization software development for the K computer environment was the Ray Tracing approach. Initial attempts were started before the availability of OSMesa functionality, and there is a written report about the porting and scalability analysis of the parallel *LuxRender* [FOR], and we can cite the *SURFACE* [FNO14], which was developed at the RIKEN R-CCS by focusing its use on the K computer environment that also includes the x86-based post-processing server (Fig. 1). In order to facilitate the cross-platform compatibility, it was developed by using OpenGL ES 2.0 compatible API. Since the LLVM-JIT functionality required for processing the GLSL shader code was not available during the development period, it utilized ahead-of-time (AOT) compilation approach by employing the Mesa 9.0.1 GLSL compiler as the front-end in order to generate the Intermediate Representation (IR) codes. After that, the own developed source-to-source translator produces the equivalent C code to be processed by the back-end Fujitsu compiler to generate a native machine code.

A large data visualization application named HIVE (Heterogeneously Integrated Visual-analytics Environment) was later developed by using the SURFACE rendering engine, and as shown in Fig. 4. It provided a Web browser based workspace for the interactive design of visualization pipeline that can be exported as Lua script, and this can be used for the batch-based large-scale parallel rendering on the K computer. Visualization examples of using the full set of compute nodes (82,944) and the largest power-of-two (2^{16}) nodes are shown in Fig. 2. Although the substantial amount of effort spent in the optimization of the SURFACE and HIVE for the SPARC64 CPU, especially for taking advantage of the HPC-ACE SIMD functionality, unfortunately it seems that the SPARC architecture will no longer be used on the mainstream HPC systems.

5. Some Lessons Learned

Each HPC system and facility can have its own uniqueness and peculiarity, and we probably cannot generalize our observations and findings for the environmental factors that can influence the visualization software development. Talking specifically on the K computer environment, we learned that you should basically work with the tools and libraries provided in the beginning of the operational life cycle, unless you can obtain additional budget for the software development. We understand that the delivery of scientific visualization to users of the K computer would have been more successful had there been early investment in porting standard graphics libraries, which could have been used to more easily port a larger collection of visualization software. However, differently to the traditional procurement based HPC system acquisition, people who are not actively participating in the supercomputer R&D project group have simply no access to the hardware and software system being designed and developed. We would not enter in the merit of the choices made by the project group, and as also said in the beginning, we learned that we should be content with what was provided, and also to take in mind that there is no guarantee about the continuity of the hardware architecture.

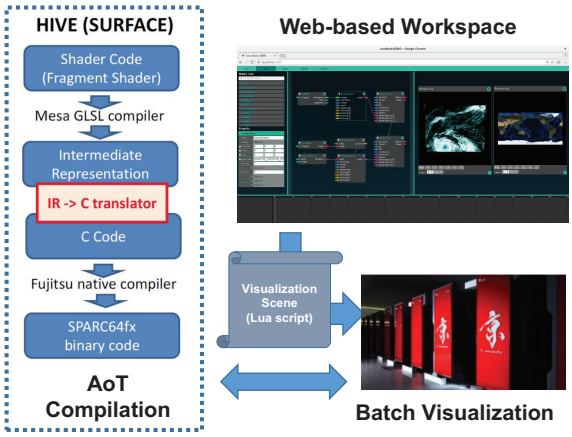

Figure 4: *HIVE large data visualization application.*

Focusing on the human factors, we perceived the difficulty of developing a general purpose visualization solution for the "Users", and some difficulties of working with "Customizers" since sometimes they are not seeing us a source of help, but as a kind of interference. Since each domain scientists can have different desires and needs to solve their problems, we reconfirmed that the most productive way is to create a win-win relationship working as "Partners" in order to develop specific visualization solutions for each case. As we mentioned in the beginning, this is not a generalization nor a critique, but a simple observation. Although the size of the Japanese HPC visualization community is small, we credit the success of PBR due to the participation of "Partners", and we hope it can continue in the Fugaku HPC environment. However, we are also aware that the remaining question is how to convince the "Users" and "Customizers" to adopt the developed software publicly available via GitHub or institutional repositories.

6. Conclusions

In this short paper, we presented some of the observations and lessons learned from the large data visualization software development on the K computer environment. Since its successor, Fugaku supercomputer, will utilize ARM based Fujitsu A64FX CPU with a wider software ecosystem, it is highly expected to minimize the problems faced with the environmental factors from the K computer system. Considering that this CPU will also be used by some models of the Cray supercomputer, and expected to be installed at some HPC sites outside Japan, we hope that we can move our efforts for developing more useful visualization applications by using common tools and libraries. We should also note that the HPC facility itself generates a large amount of data from the electrical and cooling systems, and with the increasing pressure for an energy efficient operation, in addition to the traditional SciVis, we also expect some efforts on the InfoVis and ML based visualization software development in the future.

References

[ABG*15] AYACHIT U., BAUER A., GEVECI B., O'LEARY P., MORELAND K., FABIAN N., MAULDIN J.: ParaView Catalyst: Enabling in situ data analysis and visualization. In *Proceedings of ISAV 2015: In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (2015). 2

[BAA*16] BAUER A. C., ABBASI H., AHRENS J., CHILDS H., GEVECI B., KLASKY S., MORELAND K., O'LEARY P., VISHWANATH V., WHITLOCK B., BETHEL E. W.: In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum 35*, 3 (2016), 577–597. 2

[BCH12] BETHEL E. W., CHILDS H., HANSEN C.: *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, 1st ed. Chapman & Hall/CRC, 2012. 2

[FNO14] FUJITA M., NONAKA J., ONO K.: LSGL: Large-scale graphics library for peta-scale computing environments, HPG 2014: High Performance Graphics 2014 (Poster), 2014. 4

[FOR] FORUM8: HPCI Project ID (hp130034): Building of high speed rendering environment by using photorealistic rendering engine. http://www.hpci-office.jp/output/hp130034/outcome.pdf. 4

[HSN*18] HAYASHI K., SAKAMOTO N., NONAKA J., MATSUDA M., SHOJI F.: An in-situ visualization approach for the K computer using Mesa 3D and KVS. In *High Performance Computing. ISC High Performance 2018. Lecture Notes in Computer Science, vol 11203.* (2018), Yokota R., Weiland M., Shalf J., Alam S., (Eds.), pp. 310–322. 4

[Ish19] ISHIKAWA Y.: System software for Armv8-A with SVE, 2019. Open Source HPC Collaboration on Arm Architecture - Linaro Workshop, held in conjunction with HPC Asia 2019. URL: https://static.linaro.org/event-resources/arm-hpc-2019/slides/SystemSoftwareforArmv8-AwithSVE2.pdf. 2

[KIM*15] KAWAMURA T., IDOMURA Y., MIYAMURA H., TAKEMIYA H., SAKAMOTO N., KOYAMADA K.: Remote visualization system based on particle based volume rendering. In *Visualization and Data Analysis 2015* (2015), p. 93970S. 3

[KNI16] KAWAMURA T., NODA T., IDOMURA Y.: In-situ visual exploration of multivariate volume data based on particle based volume rendering. In *Proceedings of ISAV 2016: In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (2016), pp. 18–22. 2

[McC87] Visualization in scientific computing. In *ACM SIGGRAPH Computer Graphics*, McCormick B. H., DeFanti T. A., Brown M. D., (Eds.), vol. 21(6). 11 1987, p. 15–21. 2

[MMMA11] MARUYAMA T., MOTOKURUMADA T., MORITA K., AOKI N.: Past, present, and future of SPARC64 processors. *Fujitsu Scientific and Technical Journal 47*, 2 (2011), 130–135. 2

[NMS*18] NONAKA J., MATSUDA M., SHIMIZU T., SAKAMOTO N., FUJITA M., ONISHI K., INACIO E. C., ITO S., SHOJI F., ONO K.: A study on open source software for large-scale data visualization on SPARC64fx based HPC systems. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region* (2018), HPC Asia 2018, pp. 278–288. 3

[NOF18] NONAKA J., ONO K., FUJITA M.: 234Compositor: A flexible parallel image compositing framework for massively parallel visualization environments. *Future Generation Computer Systems 82* (2018), 647 – 655. 3

[NSS*17] NONAKA J., SAKAMOTO N., SHIMIZU T., FUJITA M., ONO K., KOYAMADA K.: Distributed particle-based rendering framework for large data visualization on HPC environments. In *The 2017 International Conference on High Performance Computing & Simulation (HPCS 2017)* (2017). 3

[OMKO12] OGASA A., MAESAKA H., K. S., OTAGIRI S.: Visualization technology for the K computer. *Fujitsu Scientific and Technical Journal 48*, 3 (2012), 348–356. 2, 3

[SK12] SAKAMOTO N., KOYAMADA K.: Stochastic approach for integrated rendering of volumes and semi-transparent surfaces. In *SC Companion: High Performance Computing, Networking Storage and Analysis (UltraVis2012)* (2012), pp. 176–185. 3

[SK15] SAKAMOTO N., KOYAMADA K.: KVS: A simple and effective framework for scientific visualization. *Journal of Advanced Simulation in Science and Engineering 2*, 1 (2015), 76–95. 3

[SNKT07] SAKAMOTO N., NONAKA J., KOYAMADA K., TANAKA S.: Particle-based Volume Rendering. In *Proceedings of the IEEE Asia-Pacific Symposium on Visualization* (2007), pp. 129–132. 2

[WFM11] WHITLOCK B., FAVRE J. M., MEREDITH J. S.: Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization* (2011), EGPGV '11, pp. 101–109. 2

[YHSN19a] YAMAOKA Y., HAYASHI K., SAKAMOTO N., NONAKA J.: In situ adaptive timestep control and visualization based on the spatio-temporal variations of the simulation results. In *Proceedings of ISAV 2019: In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (2019), p. 12–16. 4

[YHSN19b] YAMAOKA Y., HAYASHI K., SAKAMOTO N., NONAKA J.: A memory efficient image composition-based parallel particle based volume rendering. *Journal of Advanced Simulation in Science and Engineering 6*, 1 (2019), 1–10. 3

[YWM08] YU H., WANG C., MA K.-L.: Massively parallel volume rendering using 2-3 Swap image compositing. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (2008), pp. 1–11. 2