Analyzing the I/O Performance of Post-Hoc Visualization of Huge Simulation Datasets on the K Computer

Eduardo C. Inacio[1,2], Jorji Nonaka[2], Kenji Ono[2,3], Mario A. R. Dantas[1,2]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC – Brazil

[2]RIKEN AICS Advanced Visualization Research Team
Kobe, Hyogo – Japan

[3]Kyushu University
Fukuoka, Fukuoka – Japan

eduardo.camilo@posgrad.ufsc.br, jorji@riken.jp
keno@cc.kyushu-u.ac.jp, mario.dantas@ufsc.br

Abstract. *As computational science simulations produce ever increasing volumes of data, executing part or even the entire visualization pipeline in the supercomputer side becomes more a requirement than an option. Given the uniqueness of the high performance K computer architecture, the* HIVE *visualization framework was developed, focusing on meeting visualization and data analysis demands of scientists and engineers. In this paper, we present an analysis on the input/output (I/O) performance of post-hoc visualization. The contribution of this research work is characterized by an analysis of a set of empirical study cases considering huge simulation datasets using HIVE on the K computer. Results from the experimental effort, using a dataset produced by a real-world global climate simulation, provide a differentiated knowledge on the impact of dataset partitioning parameters in the I/O performance of large-scale visualization systems, and highlight challenges and opportunities for performance optimizations.*

1. Introduction

As the scale of computational science simulations grows to deal with increasingly complex problems, we can also verify a significant increase in the volume of data produced [Roten et al. 2016]. To derive meaningful information from these huge datasets, leading to scientific discoveries and breakthroughs, scientists and engineers rely upon large-scale visualization and data analysis systems [Nonaka et al. 2014]. Such data-intensive applications pose a great pressure on the shared backend storage system of modern high performance computing (HPC) environments. As a result, file I/O becomes a considerable bottleneck.

In order to mitigate performance degradation due to this extreme data movement, approaches have been proposed to execute part or even the entire visualization pipeline in the supercomputer side [Bennett et al. 2012, Dorier et al. 2016]. This scenario is specially verified in HPC environments employing data staging approaches, such as the K computer [Miyazaki et al. 2012]. This approach consists of moving applications' input data to a high throughput file system prior to job execution (*i.e.,* stage-in),

and moving generated output data to users' file system after job completion (*i.e.,* stage-out) [Tsujita et al. 2017].

Focusing on meeting large-scale visualization needs on the K computer environment, a visualization framework, named Heterogeneously Integrated Visual-analytics Environment (HIVE), has been developed [Nonaka et al. 2016]. The HIVE visualization framework offers a scalable approach for both post and in-situ visualization at heterogeneous computing environments, taking advantage of increasing parallelism of modern supercomputers as well as harnessing hardware acceleration capabilities when they are available. An example of the HIVE capabilities is illustrated in Figure 1, in which is presented a projection of a 16K resolution image produced by the HIVE framework on the K computer using a dataset of 1.1 TiB generated by a global climate simulation.

Figure 1. Projection of a 16K resolution image produced by the HIVE framework on the K computer. Data courtesy of JAMSTEC, AORI/The University of Tokyo (HPCI SPIRE3), and RIKEN AICS Computational Climate Science Research Team.

Although results achieved using the HIVE visualization framework have been promising, we have verified file I/O plays a significant role in the execution time. More specifically, when used for post-processing (*i.e.,* post-hoc visualization), normally, a smaller number of compute nodes is allocated for the visualization processing, compared to the number of compute nodes used for simulation, because of the difference in processing power demands of both applications. Considering that a common approach among simulation systems is to output data into multiple files, in a per process, per variable, and/or per time step basis, this difference in scale arises as a particular problem. Mostly because the original simulation output dataset needs to be repartitioned in order to balance the workload among visualization processes [Ono et al. 2014].

In this paper, we report results of an experimental analysis on the impact of dataset partitioning parameters on the I/O performance of the HIVE visualization framework executing on the K computer. Such experimental efforts have previously demonstrated

interesting behavior on parallel storage systems [Inacio et al. 2015, Inacio et al. 2017]. This study is part of an ongoing research work collaboration that focuses on optimizing parallel I/O and storage related parameters for large-scale visualization systems running on supercomputers. While the initial focus has been the execution of the HIVE framework on the K computer, results from this research work are expected to provide helpful insights for the enhancement of the data management features of the HIVE framework on next-generation systems, including the post-K supercomputer.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the HIVE visualization framework on the K computer environment. In Section 3, the dataset partitioning problem is described. The experimental methodology and environment employed in this study is detailed in Section 4, while experimental results are discussed in Section 5. Section 6 concludes this paper with a summary of the observed results and our future research directions.

2. HIVE – Post-hoc Visualization on the K Computer

The HIVE visualization framework [Nonaka et al. 2016] was designed to run on heterogeneous hardware platform environments found on traditional HPC infrastructures, such as the K computer operational environment. The development of HIVE was mainly motivated by the demand of visualizing huge datasets generated by large-scale computational science simulations executed on the K computer. Another reason, relies upon the unique architecture of the K computer, that makes it difficult to directly adopt existing visualization systems, such as PARAVIEW [Fabian et al. 2011] and VISIT [Childs et al. 2012].

The design of HIVE follows a client/server paradigm. This approach offers a great flexibility for users to select configurations that better suit their visualization needs. For instance, a client can be executed on a local machine, exploring interactivity, while a server could be deployed at a visualization cluster, fully utilizing abundant hardware resources. A web-based graphic user interface (GUI) permits easily definition of visualization pipelines, which are later exported as Lua scripts, allowing for the automation of the visualization workflow. Moreover, a command-line interface allows for batch execution of visualization workflows, which is particularly helpful for post-hoc visualization in the K computer.

An overview of a post-hoc visualization using the HIVE framework on the K computer environment is illustrated in Figure 2. The shared backend storage of the K computer has two layers: a Local File System (LFS), designed to support high throughput I/O; and a Global File System (GFS), that offers a large storage capacity (> 30 PB) for users' applications and data files; both implemented through the Fujitsu Exabyte File System (FEFS), an enhanced version of the LUSTRE parallel file system (PFS). Since only executing jobs have access to the LFS, simulation output data is available to users after the job completion, when this data is staged-out to the GFS.

In order to execute a post-hoc visualization on the K computer using HIVE, the user must provide a *visualization scene*, which contains the parameters for the visualization processing. This visualization scene can be produced manually, or using one of the editors provided with the HIVE framework. With the simulation output data available at the GFS, the user can submit a job script invoking HIVE rendering command (`hrender`), passing the visualization scene as argument. It is worth noting that, given

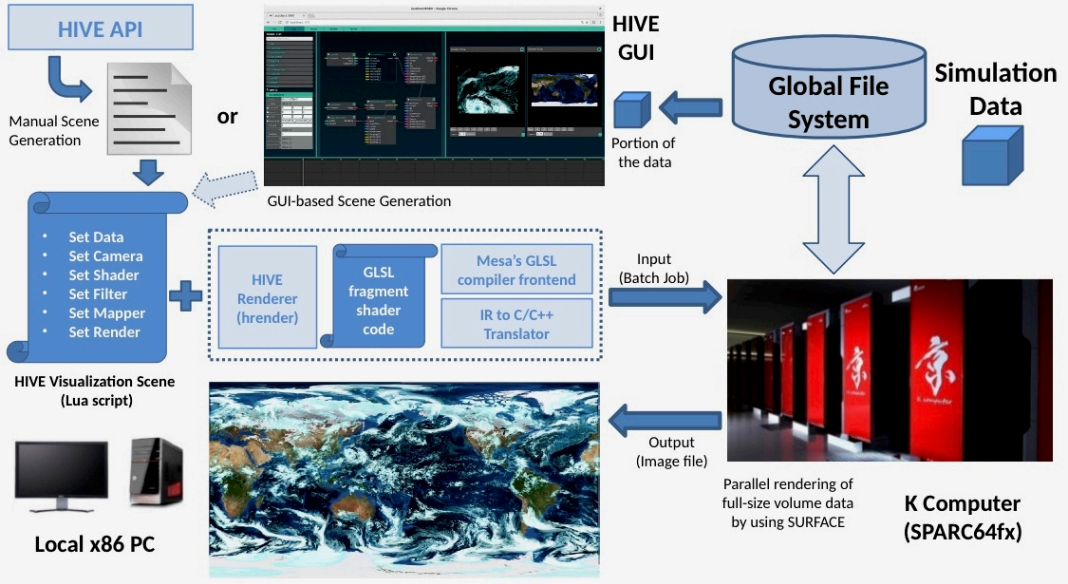

Figure 2. An overview of a post-hoc visualization using the HIVE framework on the K computer environment.

the staging approach of the K computer, simulation output data is staged-in again before the visualization job is launched for execution, which translates into a massive data movement. Once the visualization job is finished, image files stored in the LFS are staged-out, and can be analyzed by users.

3. The *M* x *N* Dataset Partitioning

The number of processes required by a visualization job for rendering a simulation output dataset is usually smaller than the number of processes used for generating the dataset. Also, computational science simulations executed on K computer usually adopt a file per processes approach when outputting data, mostly motivated by particular optimizations provide by the FEFS for such access pattern [Tsujita et al. 2017]. Consequently, in post-hoc visualization, the problem domain must be repartitioned in order to achieve load balance among visualization processes. As a result, a single visualization process can access data points in multiple files, and a single file can be concurrently accessed by multiple visualization processes.

The *M* x *N* dataset partitioning problem refers to these situations, in which a simulation generates *M* output files, and *N* visualization processes, with *N* usually smaller than *M*, will consume the dataset. The dataset partitioning options, which include the number of visualization processes and the number of partitions in the x, y, and z dimensions (for a Cartesian three dimensional problem space), are provided by the user through the visualization scene script. As previously stated, the way the output dataset is partitioned can have significant impact in the I/O performance of the post-hoc visualization, mainly due to conditions of concurrent access to shared files. Nevertheless, identifying the optimal options for the partitioning parameters can be a daunting task, considering the innumerous options available.

More specifically, the alignment of the visualization dataset partitioning with the

original partitioning of the simulation output could avoid concurrent file accesses conditions. Figure 3 provides an example of an aligned dataset partitioning. The simulation dataset consists of 12 files, partitioning the two dimensional problem domain into 4 x 3 grids. At post-hoc visualization, four processes were used, partitioning the simulation dataset into 4 x 1 grids. Under these partitioning parameters, it is possible to observe that each visualization process would independently access tree simulation output files.

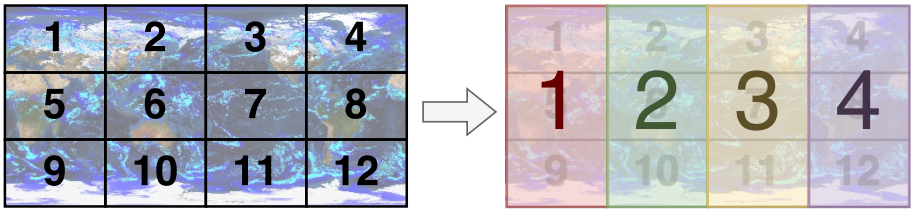

Figure 3. Example of a visualization dataset partitioning aligned with the partitioning used at simulation output files (M = 12, N = 4).

A very different situation is observed when the visualization partition parameters are defined as 2 x 2, even keeping the same number of visualization processes, as demonstrated in Figure 4. This configuration results in an aligned partition of the simulation output. Consequently, beyond visualization process accessing multiple files, it can be verified that some files are concurrently accessed by two processes. In this example, files 5 and 6 are concurrently accessed by visualization processes 1 and 3, while files 7 and 8 are accessed by visualization processes 2 and 4.

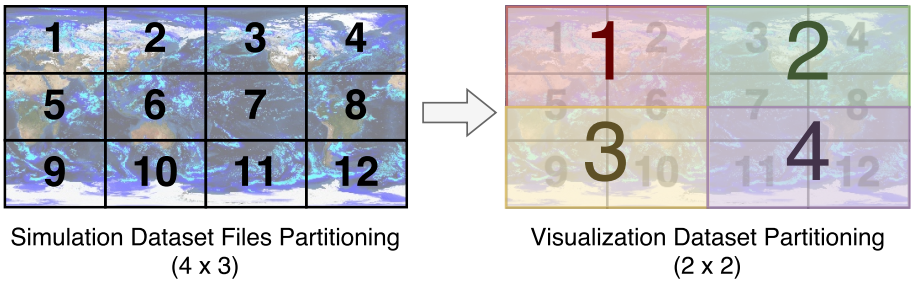

Figure 4. Example of a visualization dataset partitioning unaligned with the partitioning used at simulation output files (M = 12, N = 4).

Based upon these simple examples provided, it is possible to conceive that at larger scales, with hundreds to thousands of processes and files involved, the complexity of the *M* x *N* partitioning problem can increase drastically. Consequently, a significant impact in the overall performance of the post-hoc visualization on the K computer can be attributed to I/O performance degradation associated to dataset partitioning parameters. Next sections details the experimental effort carried out focusing on better understanding the magnitude of such impact.

4. Experimental Environment and Method

In order to observe the impact of using different partitioning parameters into the post-hoc visualization of a huge simulation dataset, several experiments were conducted processing a real-world simulation dataset using the HIVE visualization framework on the K computer. The K computer [Miyazaki et al. 2012] is a Japanese flagship-class supercomputer, developed by Fujitsu in collaboration with RIKEN and currently operated by the RIKEN Advanced Institute for Computational Science (AICS), consisting of $82{,}944$ compute nodes, with a SPARC64fx CPU and 16 GB RAM each, connected through a 6D Tofu interconnect [Ajima et al. 2011]. At full capacity, the K computer is capable of performing 10 Petaflops (10 quadrillion floating-point operations per second), which granted it the top position at the list of the 500 fastest supercomputers in the world for two consecutive times in 2011 [Dongarra et al. 2017], when it started operation.

A dataset generated by a real-world simulation was used in these experiments. This dataset corresponds to a sub-kilometer global simulation of deep moist atmospheric convection using the Nonhydrostatic Icosahedral Atmospheric Model (NICAM) [Miyamoto et al. 2013] executed on the full configuration of the K computer. The NICAM simulation outputs variables in a file per process for each time step. After simulation, variables in the dataset are remapped from the icosahedral grid to a geodesic (latitude-longitude) grid [Satoh et al. 2017]. In this simulation dataset, the x, y, and z dimensions have respectively $11{,}520$, $5{,}760$, and 94 points, and 48 time steps. For this analysis, a single variable and four time steps were considered, resulting in a dataset of 94 GiB. It is worth noting that time steps are processed by the visualization system sequentially. Therefore, using 48 times steps would mainly result in 12-fold larger execution times.

Furthermore, simulation data points were redistributed into 384 files per time step, as if they were generated by a simulation using 384 processes. As a result, files became larger in this configuration than in the original one, keeping the dataset size fixed. This approach was adopted in order to have a baseline performance (*i.e.,* an *M x M* mapping) for comparison of different *M x N* configurations. Executing an *M x M* post-hoc visualization using the full configuration of the K computer (*i.e.,* $82{,}944$) would not only be an unrealistic scenario, but also undesirable from an operational perspective.

Each process (MPI rank) is allocated to a different compute node. Before execution, the dataset is staged-in to the K computer LFS. It is worth mentioning that the K computer resource management system (RMS) allocates Object Storage Servers (OSSs) for a job accordingly to the number and shape of allocated compute nodes. Basically, OSSs in the same racks of allocated compute nodes are made available for jobs. This policy focuses making data closer to processes and mitigating cross-application interferences.

Table 1 presents dataset partitioning parameters considered in the experiments with post-hoc visualization of the NICAM simulation output dataset using the HIVE framework on the K computer. It can be observed that a wide range of parameters were evaluated. Depending on the number of processes used for visualization, which ranges from the same number of files in the dataset up to 16 times less processes, varying aligned and unaligned partitions were considered. The grid size refers to the number of points in each dimension (*i.e.,* x x y x z) per process and per time step.

Table 1. Dataset partitioning parameters considered in the experimental evaluation using the HIVE visualization framework in the K computer.

# Processes	# Partitions	Grid Size (Pts)	File Size	Alignment
384	32 x 12 x 1	360 x 480 x 94	62 MiB	Aligned
192	32 x 6 x 1	360 x 960 x 94	124 MiB	Aligned
	6 x 32 x 1	1920 x 180 x 94	124 MiB	Unaligned
	16 x 12 x 1	720 x 480 x 94	124 MiB	Aligned
	12 x 16 x 1	960 x 360 x 94	124 MiB	Unaligned
96	16 x 6 x 1	720 x 960 x 94	248 MiB	Aligned
	6 x 16 x 1	1920 x 360 x 94	248 MiB	Unaligned
	8 x 12 x 1	1440 x 480 x 94	248 MiB	Aligned
	12 x 8 x 1	960 x 720 x 94	248 MiB	Unaligned
48	4 x 12 x 1	2880 x 480 x 94	496 MiB	Aligned
	12 x 4 x 1	960 x 1440 x 94	496 MiB	Unaligned
	8 x 6 x 1	1440 x 960 x 94	496 MiB	Aligned
	6 x 8 x 1	1920 x 720 x 94	496 MiB	Unaligned
24	8 x 3 x 1	1440 x 1920 x 94	992 MiB	Aligned
	3 x 8 x 1	3840 x 720 x 94	992 MiB	Unaligned
	4 x 6 x 1	2880 x 960 x 94	992 MiB	Aligned
	6 x 4 x 1	1920 x 1440 x 94	992 MiB	Unaligned

In the context of this research work, an experiment run consists of the dataset loading phase of the HIVE visualization framework, using one of the partitioning parameters presented in Table 1. I/O performance metrics (*i.e.,* response variables) evaluated in this research work are the time each process takes to load the required data for each time step, and the time required to load the complete dataset, including all time steps. Each experiment run was replicated three times to account for experimental variance. The execution order of the experiment runs was completely random to assure response variables are independent and individually distributed.

5. I/O Performance Analysis

As discussed in Section 3, depending on the partitioning parameters, the visualization processes may need to read from multiple files. Furthermore, multiple processes may simultaneously access the same file, which can result in a contention translated into an I/O performance degradation. To verify such behavior, an analysis was carried out on the time that each process takes to load its designated data points per time step.

Figure 5 presents the load time per process according to different dataset partitioning parameters. Boxes delimits the first and third quantiles (*i.e.,* the 25th and 75th percentiles), thus, comprising 50% of the measurements. The horizontal line inside boxes denotes the median value (*i.e.,* 50th percentile), while vertical lines, known as whiskers, extend to largest and smallest values no further than $1.5\times$ the inter-quartile range (IQR) above or below boxes, respectively. Data points beyond whiskers denote outliers. Red boxes denote unaligned partitions, while blue boxes denote aligned partitions.

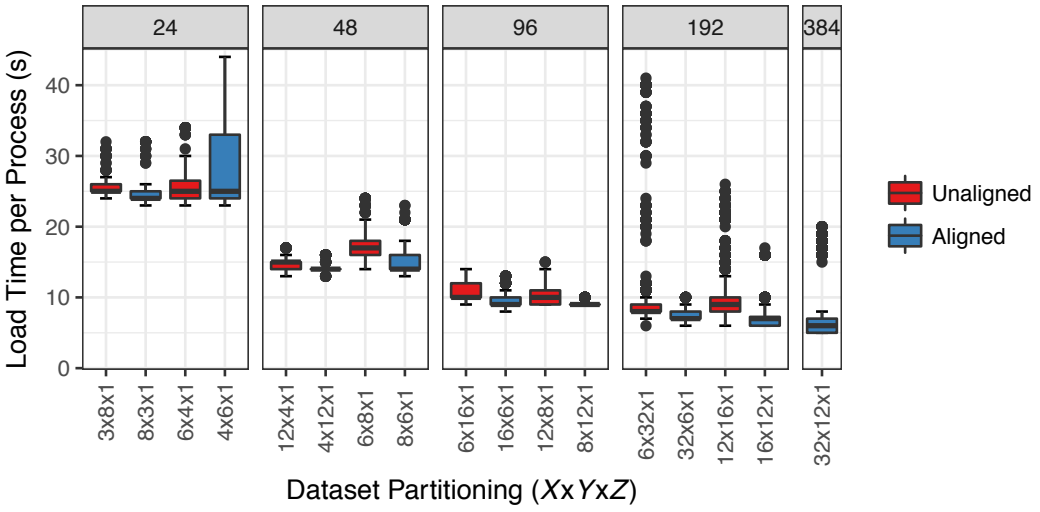

Figure 5. Load time of a single time step of the simulation dataset per process using different number of processes and partitions.

The most evident behavior in these results is the decrease of the load time per process with the increase in the number of processes used for visualization. This behavior can be mostly associated with the amount of data read by each visualization process. Considering the dataset size is fixed for the experiment, increasing the number of processes translates into each process loading a smaller number of points, and, thereby, reading a smaller fraction of the dataset. However, it is worth noting that the load time did not reduce in the same rate of the increase of the number of processes. Comparing the mean load time using 24 processes with the baseline case, with 384 processes, the I/O performance was approximately 3.8 times better at the cost of 16 times more compute nodes, an efficiency of 23.5%. For 96 processes, the compared efficiency increases to 35.4%, which can still be considered low. These results shed light upon this important trade-off faced by this large-scale visualization systems executing on the K computer.

Another prominent behavior observed in these results regards the variance of the load time per process across varying partitioning parameters. In general, it is possible to observe a larger variance for unaligned partitions compared to aligned ones; not only by the length of the boxes, but also by the number of outliers observed. Moreover, based upon the shape of the boxes and predominance of upper outliers, it is possible to conceive that the response variable has a skewed distribution with a long tail to the right. This means that most of the measurements are concentrated around smaller values, but with a number of measurements way above the average. Although many factors can contribute to this delay observed in some processes while loading their respective data points from the dataset, it is reasonable to infer that contention due to multiple processes accessing the same file has a non negligible impact in the I/O performance. Precisely controlling the computing environment for isolating such effect in an experiment is challenging, because of the number and variety of elements involved, including other concurrent jobs running at the supercomputer. A further investigation is in progress focusing on addressing this matter.

While the analysis of the time for a process to complete loading data points from a time step of the simulation datasets gives insight about the contention effects resulting from partition parameters, it is also important, from the visualization system perspective, to verify the overall performance behavior, since the delay of some parallel processes may overlap and have their direct impact in performance less perceived. In Figure 6, the time to load the complete dataset considering different partitioning parameters is presented. In other words, this response variable represents the time elapsed for all processes to read their respective data points from all time steps of the dataset in the LFS of the K computer. Red bars denote the mean value for unaligned partitions, while blue bars denote the mean value for aligned ones. Vertical error bars represent the standard error around the mean.

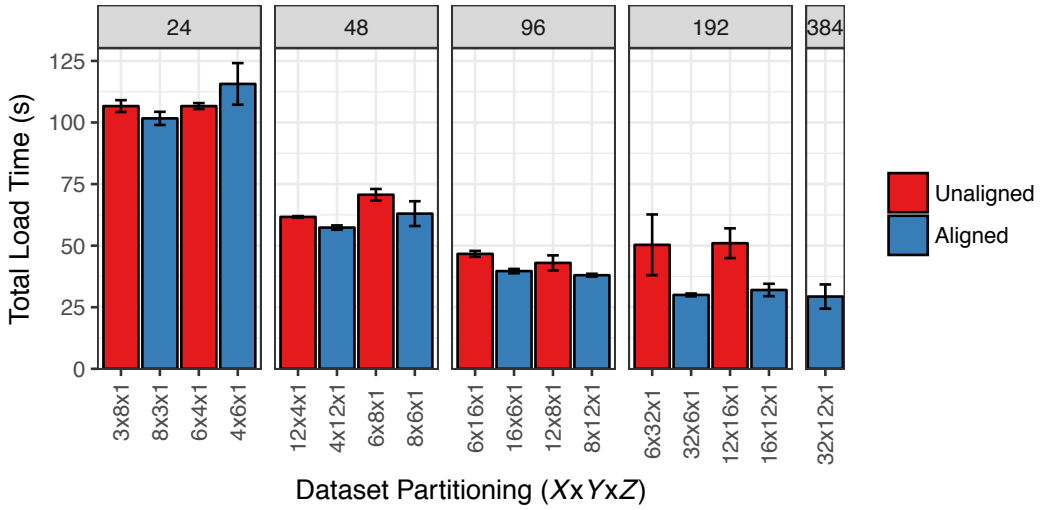

Figure 6. Time to load the complete dataset using varying number of processes and partitions.

Through these results, the comparison of the load time among different number of processes becomes more clear. It is noticeable that experiments with 24 processes had a total load time significantly higher compared to the other experiments. However, it can be observed that for 48, 96, 192, and the baseline case, 384 processes, the mean values became closer for some partitions. For the unaligned partitions 12 x 4 x 1 (48 processes), 6 x 16 x 1, 12 x 8 x 1 (96 processes), 6 x 32 x 1, and 12 x 16 x 1 (192 processes), both mean values and standard error values overlap. This indicates that it is not possible to discern the I/O performance among these partition parameters. In other words, using a partition 12 x 4 x 1 with 48 processes may outcome in the same total load time as using a 6 x 32 x 1 partition with 192 processes. The same interpretation is valid for the aligned partitions 32 x 6 x 1, 16 x 12 x 1 (192 processes), and 32 x 12 x 1 (384 processes). This means that, using half of the number of processes used in the simulation for HIVE post-hoc visualization in the K computer, with proper aligned partitioning parameters, a comparable I/O performance can be achieved, while saving computing resources.

Comparing aligned and unaligned partitions inside a fixed number of processes, it is possible to verify how the partitioning alignment can affect the total load time in the post-hoc visualization in the K computer. Except for one case, namely, the partition 4

x 6 x 1 (24 processes), unaligned partitioning resulted in a larger load time. The scale of the differences varies significantly, though. Results suggest the difference is affected by the number of concurrent processes in a file, which is a result from the partitioning parameters. For instance, in the 6 x 32 x 1 partition (192 processes), each file in the original dataset is accessed by at least three visualization processes, while some files are accessed by six processes. Based upon these results, it is possible to conceive the importance of properly choosing the partitioning parameters for post-hoc visualization combined with a more efficient use of the computing resources of the K computer.

6. Conclusions and Future Works

In this research work, we present a contribution that could be understood as the analysis of experiments carried out using the HIVE framework to visualize a huge dataset produced by a real-world computational science simulation on the K computer, and how significant was the impact of dataset partitioning in the I/O performance of a large-scale visualization system. The research presented in this paper will contribute to advance a collaborative research work in progress that aims at optimizing parameters available at the parallel I/O software stack of modern supercomputers for large-scale visualization purposes. Moreover, these results can provide insights on visualization requirements for the design of next-generation supercomputers, including the post-K computer.

Analyzing the time taken by each process to load its data points from the simulation output dataset, it was observed a strong influence of the volume of data transferred in the I/O performance. However, while increasing the number of process effectively reduced the load time per process, results indicate the efficiency of this approach can be low, pointing out an important trade-off between load time and computation resource utilization. On extreme cases, the load time was reduced by 3.8 times using 16 times more processes. Such observation provides compelling reasons for further research in the subject in order to more efficiently explore the available computing resources Furthermore, the variance of the load time across processes suggests while most processes complete loading a time step of the simulation output within a particular time window, a number of processes get significantly delayed. Based on the conditions of the experiment, we argue that such behavior can be associated, in part, to contention in file I/O access due to partitioning parameters.

Considering the time taken by HIVE to load all the simulation dataset, it was observed that the difference in I/O performance becomes less prominent. It is reasonable to consider that differences in load time per tasks get overlapped by parallel processing. Such observation gives insight on real opportunities for executing post-hoc visualization on K computer using a smaller number of processes than what was used for simulation execution, with comparable I/O performance. Nevertheless, dataset partitioning parameters can play a significant role in these situations, resulting in performance degradation if not properly addressed.

As part of this ongoing research work, we intend to, in short-term, perform a more detailed analysis, instrumenting the HIVE code to obtain fine-grained measurements of I/O operations performed during dataset loading. Furthermore, experiments should be conducted to assess the impact of factors related to the parallel I/O software stack of the K computer, such as file striping options of the FEFS, which could also be leveraged for

improved visualization performance. After this characterization effort, an optimization approach for the dataset management of the HIVE visualization system will be designed and developed, focusing on providing efficient large-scale visualization for extreme-scale supercomputers.

Acknowledgements

We would like to thank the Brazilian Federal Agency CAPES for supporting this research. Results reported in this paper were obtained by using the K computer at RIKEN Advanced Institute for Computational Science (AICS) in Kobe, Japan.

References

Ajima, Y., Takagi, Y., Inoue, T., Hiramoto, S., and Shimizu, T. (2011). The Tofu Interconnect. In *HOTI '11 Proceedings of the IEEE 19th Annual Symposium on High Performance Interconnects*, pages 87–94. IEEE.

Bennett, J. C., Abbasi, H., Bremer, P.-T., Grout, R., Gyulassy, A., Jin, T., Klasky, S., Kolla, H., Parashar, M., Pascucci, V., Pebay, P., Thompson, D., Yu, H., Zhang, F., and Chen, J. (2012). Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE.

Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Pugmire, D., Biagas, K., Miller, M., Harrison, C., Weber, G. H., Krishnan, H., Fogal, T., Sanderson, A., Garth, C., Bethel, E. W., Camp, D., Rübel, O., Durant, M., Favre, J. M., and Navrátil, P. (2012). VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In Bethel, E. W., Childs, H., and Hansen, C., editors, *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, pages 357–372. Chapman & Hall/CRC, 1 edition.

Dongarra, J., Meuer, H. W., and Strohmaier, E. (2017). TOP500 Supercomputer Sites.

Dorier, M., Sisneros, R., Gomez, L. B., Peterka, T., Orf, L., Rahmani, L., Antoniu, G., and Bougé, L. (2016). Adaptive Performance-Constrained In Situ Visualization of Atmospheric Simulations. In *CLUSTER '16 Proceedings of the IEEE International Conference on Cluster Computing*, pages 269–278. IEEE.

Fabian, N., Moreland, K., Thompson, D., Bauer, A. C., Marion, P., Gevecik, B., Rasquin, M., and Jansen, K. E. (2011). The ParaView Coprocessing Library: A scalable, general purpose in situ visualization library. In *LDAV '11 Proceedings of the IEEE Symposium on Large Data Analysis and Visualization*, pages 89–96. IEEE.

Inacio, E. C., Barbetta, P. A., and Dantas, M. A. R. (2017). A Statistical Analysis of the Performance Variability of Read/Write Operations on Parallel File Systems. *Procedia Computer Science - Special Issue: International Conference on Computational Science, ICCS 2017*, 108:2393–2397.

Inacio, E. C., Pilla, L. L. L., and Dantas, M. A. R. (2015). Understanding the Effect of Multiple Factors on a Parallel File System's Performance. In *WETICE '15 Proceedings of the 24th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 90–92. IEEE.

Miyamoto, Y., Kajikawa, Y., Yoshida, R., Yamaura, T., Yashiro, H., and Tomita, H. (2013). Deep moist atmospheric convection in a subkilometer global simulation. *Geophysical Research Letters*, 40(18):4922–4926.

Miyazaki, H., Kusano, Y., Shinjou, N., Shoji, F., Yokokawa, M., and Watanabe, T. (2012). Overview of the K computer System. *Fujitsu Scientific and Technical Journal*, 48(3):255–265.

Nonaka, J., Ono, K., Bi, C., Sakurai, D., Fujita, M., Oku, K., and Kawanabe, T. (2016). HIVE: A Visualization and Analysis Framework for Large-Scale Simulations on the K Computer. In *PacificVis '16 Proceedings of the IEEE Pacific Visualization - Poster Session*. IEEE.

Nonaka, J., Ono, K., and Fujita, M. (2014). Multi-step image compositing for massively parallel rendering. In *HPCS '14 Proceedings of the International Conference on High Performance Computing & Simulation*, pages 627–634. IEEE.

Ono, K., Kawashima, Y., and Kawanabe, T. (2014). Data Centric Framework for Large-scale High-performance Parallel Computation. *Procedia Computer Science - Special Issue: International Conference on Computational Science, ICCS 2014*, 29:2336–2350.

Roten, D., Cui, Y., Olsen, K. B., Day, S. M., Withers, K., Savran, W. H., Wang, P., and Mu, D. (2016). High-Frequency Nonlinear Earthquake Simulations on Petascale Heterogeneous Supercomputers. In *SC '16 Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE.

Satoh, M., Tomita, H., Yashiro, H., Kajikawa, Y., Miyamoto, Y., Yamaura, T., Miyakawa, T., Nakano, M., Kodama, C., Noda, A. T., Nasuno, T., Yamada, Y., and Fukutomi, Y. (2017). Outcomes and challenges of global high-resolution non-hydrostatic atmospheric simulations using the K computer. *Progress in Earth and Planetary Science*, 4(1):24.

Tsujita, Y., Yoshizaki, T., Yamamoto, K., Sueyasu, F., Miyazaki, R., and Uno, A. (2017). Alleviating I/O Interference Through Workload-Aware Striping and Load-Balancing on Parallel File Systems. In Kunkel, J. M., Yokota, R., Balaji, P., and Keyes, D., editors, *High Performance Computing*, volume 10266 of *Lecture Notes in Computer Science*, pages 315–333. Springer.