Performance Classification of the K-Computer Workloads Using Hierarchical Clustering and K-Means

Extended Abstract

Masaaki Terai
RIKEN Advanced Institute for Computational Science
Kobe, Hyogo, Japan
teraim@riken.jp

Riku Kashiwaki
Graduate School of Simulation Studies, Univeristy of Hyogo
Kobe, Hyogo, Japan
sa17k003@sim.u-hyogo.ac.jp

Fumiyoshi Shoji
RIKEN Advanced Institute for Computational Science
Kobe, Hyogo, Japan
shoji@riken.jp

ABSTRACT

In the K computer, the job manager automatically collects various metrics for workloads and stores them in databases. However, most of the data are not fully exploited in our services due to the huge and disorganized data set. As an early study, in order to understand the characteristics of the workloads and extract them without training data set, we make an attempt to classify the workloads using hierarchical clustering and *k*-means. Both techniques are fundamental multivariate analysis methods to divide the entire data set into small groups and easily understand the behavior of the groups. At first, we use typical HPC benchmarks to classify workloads with the criterion, which are grouped seven clusters according to a different set of characteristics. In addition, we classify the total of 363,015 actual workloads by *k*-means with the seven clusters. These classified workloads are further partitioned into groups and having different characteristics as well. Based on these results, we sort out the workloads in terms of performance. Finally, the poster presents that one cluster having a large number of nodes shows lower performance than the mean of all the clusters. The extracted cluster becomes a candidate to examine in more detail about the performance.

KEYWORDS

K-computer, classification, performance analysis, *k*-means, hierarchical clustering

ACM Reference Format:
Masaaki Terai, Riku Kashiwaki, and Fumiyoshi Shoji. 2018. Performance Classification of the K-Computer Workloads Using Hierarchical Clustering and K-Means: Extended Abstract. In *Proceedings of HPC Asia 2018 (HPC Asia),* Masaaki Terai, Riku Kashiwaki, and Fumiyoshi Shoji (Eds.). ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

As part of the operations of our supercomputing center, we currently address a usage survey of our facility that includes understanding workload characteristics with statistical analysis and finding issues from a huge number of workloads in the system. The survey helps not only demand analysis of the procurement process, but also provides insights on how to improve our services.

In the K computer (hereinafter referred to as K/K-computer) [2], the job manager collects various usage metrics (e.g., job name, number of nodes, elapsed time, maximum memory usage size, I/O usage, number of staging files, staging file size, and raw data of hardware counters) for each job, which constitutes a workload. In operation, these metrics are automatically collected and stored in databases. A part of the metrics is directly provided to users by the job manager. Also, some part of the metrics is summarized and reported by administrators. However, most of the data are not fully exploited for analysis to help inform our operations because the amount of data stored in databases is growing every moment and becoming huge size that is difficult to handle them.

Recently, in order to obtain more meaningful information about workload performance from huge system logs, machine learning or some statistical techniques are attracted [1][3]. One of the statistical techniques, clustering is a fundamental multivariate analysis technique to divide the entire data set into small groups and easily understand behavior group by group without training data set. This is an appropriate method for systematically classifying unlabeled workloads in this case.

As a preliminary study, we attempt to classify typical HPC benchmarks (e.g., DGEMM, STREAM and so on) with hierarchical clustering before classifying the real workloads. These well-known benchmarks reveal metrics behavior and correlations. We then determine the number of clusters.

Base on the results of the preliminary study, by using *k*-means, we attempt to classify the massive number of workloads executed on K. Finally, with respect to performance (e.g., FLOPS, memory throughput, and I/O intensive), we analyze the workload characteristics of each group.

All the analyses were mainly performed by SciPy, scikit-learn, and pandas in Python.

2 K-COMPUTER

2.1 Overview

K-computer is the first 10-petaflops supercomputer developed by RIKEN and Fujitsu under the Japanese national project. The system has 82,944 compute nodes connected by Tofu high-speed interconnects. Each compute node has a SPARC-V9 based customized chip called SPARC64VIIIfx [4] and is equipped with 16 GB memory for each compute node. Furthermore, a 30 PB global storage and an

HPC Asia, Jan 2018, Tokyo, Japan

11 PB local storage are installed and are transparently accessed by compute nodes through I/O nodes.

2.2 Hardware Counters

The SPARC64VIIIfx chip has built-in hardware counters to store the counts of events (e.g., the number of cycles, floating instructions, load/store instructions, cache misses, and bus transactions in terms of memory and I/O read/write accesses). To precisely measure application performance with a profiler, 56-type counters are to be unlocked to users and extremely low overhead profiling is to be achieved compared with ordinary sampling-based profiling. However, the number of counters in a single execution of a user program is limited by the mechanism and the user program cannot simultaneously use more than eight hardware counters. Besides, users cannot change a set of hardware counters during the execution. Therefore, in this study, we use a single set of counters throughout the duration to consistently compare the same metrics between all targeted workloads.

2.3 Job Record Extraction From the Database

For system efficiency, K-computer employs a job manager, peripheral system software, and tools, which are similar to most supercomputers. The job manager controls submitted jobs as a basic unit of workload and exclusively executes them on compute nodes. At the same time, the job manager records the time stamp, state, and various metrics of the workload in a database.

Also, for usability enhancement, the job manager provides various job submitting methods (e.g., batch and interactive job types, normal, bulk, and step job models) to users. For a preliminary study, we extract only records of the batch job type with the normal model from the database because this type of record is dominant and accounts for more than 75% of all the node-time in the valid records on K. Also, other types of records in the bulk and step job models have a slightly cumbersome structure because of the parent-child relationship. Furthermore, the interactive job spends much time for an idle state because of waiting for the key-in command by a user. (This "interactive" type job provides a command prompt on a compute node. A user can interactively enter commands to start the user's program.) Classifying workloads based on their performance records may make them noise information. Besides, we ignore records with abnormal job termination because these records have missing values. Finally, we use valid records from the database, except for the above conditions.

3 PRELIMINARY

3.1 Job Metrics

The job manager and peripheral tools collect more than 120 metrics for each job. Before the classification of the workloads, to easily handle them, we eliminated the categorical metrics from the original data set and then obtained the 24 metrics as shown in Table 1. One of the metrics, "cycle_counts" is used for the normalization of the hardware counters, but it is omitted from the table.

Also, the database includes particular kinds of metrics not directly related to the workload performance, for instances, regarding the file staging, these metrics refer to the data duplicate process between the local and global storage. This kind of the information may become one of the diagnostic metrics for K in the future. However, as a first step, to reduce the metrics to be used, we omitted the metrics regarding the file staging in this study.

Table 1: Job Metrics

metric name	description
alloc_core_total	num of cores actually allocated to compute nodes
alloc_node_num	num of nodes actually allocated to compute nodes
cpu_mem_read_count_ratio	cpu_ memory_read_counts per cycle_counts
cpu_mem_write_count_ratio	cpu_memory_write_ counts per cycle_counts
elapsed_time	elapsed time in a job
file_io_size	file I/O size
floating_inst_ratio	floating_inst. counts per cycle_counts
flops	num of floating-point op. per elapsed time (FLOPS)
fma_inst_ratio	FMA inst. counts per cycle_counts
io_transfer_size	I/O transfer size
max_cycle_counts	max cycle counts in all threads
max_use_mem	max memory usage in a compute node
mem_th	memory throughput
op_intensity	arithmetic intensity (=flops/mem_th)
read_system_call	num of read system call
req_core_num	num of cores required by a job script
req_node_num	num of nodes required by a job script
simd_floating_inst_ratio	SIMD-floating inst. counts per cycle_counts
simd_fma_inst_ratio	SIMD-FMA inst. counts per cycle_counts
sleep_cycle_ratio	sleep_cycle_counts per cycle_counts
use_core_total	num of cores actually used for a workload
use_node_num	num of nodes actually used for a workload
use_nodetime	product of use_node_num and elapsed_time
write_system_call	num of write system call

In this study, we use the one-year data set collected from Oct. 1, 2016 to Sep. 30, 2017. The number of workloads is 363,015.

3.2 Preprocessing

Most of the metrics are integer-type variables. However, part of the metrics (e.g., cycle_counts and sleep_cycle) easily become a large number represented by a 128-bit integer. Also, their counts depend on the duration of a workload. To equally compare metrics with other workloads in view of three types of workloads: arithmetic, memory access, and I/O intensive workload not depending on the number of nodes, we normalized the eight hardware counters by using "cycle_counts."

In addition, a classification process sufficiently works with lower accuracy than the accuracy of the actual measurement. Therefore, we treated all the metrics as a 32-bit floating-point number.

Also, the metrics have substantially left-skewed distribution. To be spread between records more uniformly, we apply the logarithmic transformation defined by $w = \ln(v)$, where $v = u + 1 (u \geq 0)$ and u is an actual measurement value of a metric. Then, we apply standardization to the log-transformed value given by $w_{std} = (w - w_{mean}) / w_{sd}$, where w_{mean} is the mean value and w_{sd} is the standard deviation for w.

Finally, after the rescaling process, we obtained the preprocessed data set.

4 CLUSTERING FOR BENCHMARKS

To understand the characteristics of metrics, we classify the typical HPC benchmarks (e.g., DGEMM, STREAM, IOR, Intel MPI Benchmarks, and NasParallel Benchmarks) before classifying the real workloads on K.

Figure 1 is the result of using a hierarchical clustering method with Euclidean distance and Ward variance minimization algorithm as a linkage method. In this figure, standardized metric values are shown. The standardization processes were applied to each column.

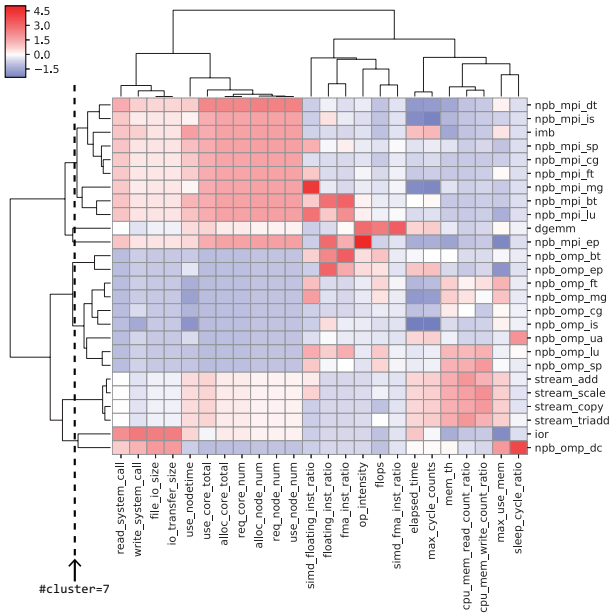


Figure 1: A classification of the benchmarks by hierarchical clustering. The columns mean the metrics. The rows mean the benchmarks. A heat map with dendrograms uses the standardized values for each metrics. "dgemm" is matrix-matrix multiplication. "stream" is memory bandwidth measurement benchmark. "ior" is file I/O intensive benchmark. "imb" is Intel MPI Benchmarks. "npb_omp_*" are NasParallel Benchmarks, OpenMP threaded version. "npb_mpi_*" are NasParallel Benchmarks, MPI version.

The white color represents the mean value in a column. The red and blue colors describe the values higher and lower than the mean values, respectively. Also, on the top and left sides of the heat map, dendrograms are shown.

To determine a minimum number of clusters, we used a criterion that divides intensive workload ("dgemm" and "stream") into different groups. Based on the criterion, we obtained seven clusters from the hierarchical clustering.

As shown on the heat map, "dgemm" obviously has the largest "flops" value in all the benchmarks. "Stream", "npb_omp_lu", and "sp" reflect the memory throughput behavior. Also, "ior" and "npb_omp_dc" have high metric values regarding I/O. Furthermore, a group of "npb-omp" and "npb-mpi" have divided two large groups based on the number of nodes. We found that the behavior of workloads reflecs most of the metrics.

We show a dendrogram of the metric clustering shown on the top side of the heat map. While "flops" and "simd_fma_inst_ratio" have similar behavior, "floating_inst_ratio" and "fma_inst_ratio" are not directly tied in "flops." We think that these instructions have the tendency to be exclusively issued in the same workloads. Also, in the metric range from "use_core_total" to "use_node_num," we found extremely similar behaviors. In other words, we can select one of the metrics for feature selection.

5 CLUSTERING FOR REAL WORKLOADS

We classified all the targeted workloads on K, using k-means with seven clusters based on the result of the previous chapter. We used this classification instead of hierarchical clustering because it is difficult to apply them because of the complexity $O(N^2)$, where N is the number of workloads.

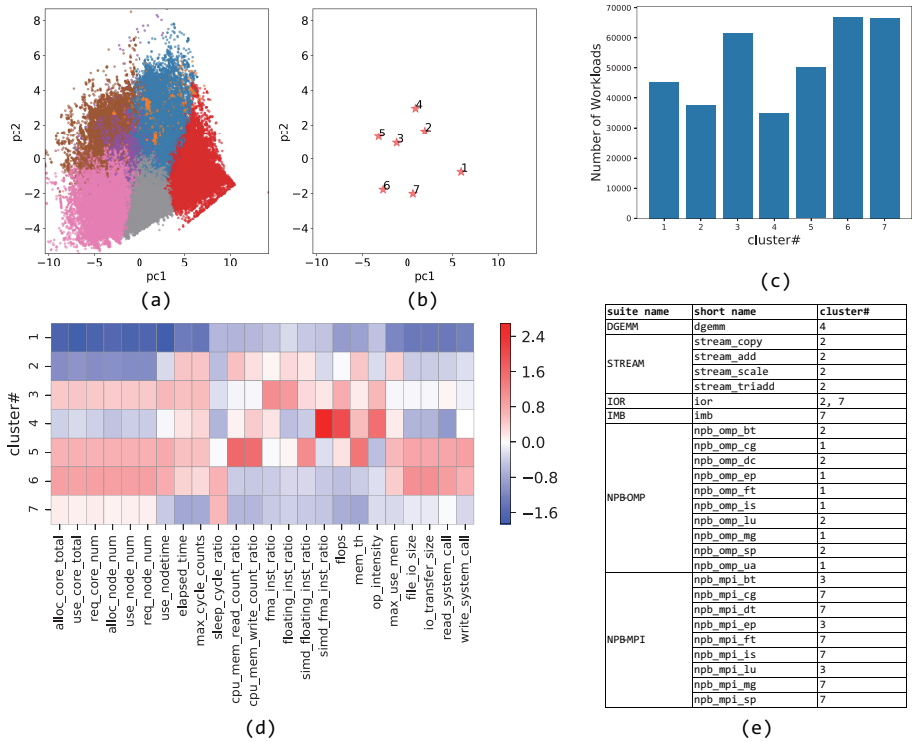
To depict the classification result by k-means, we used scatterplot on the principal component analysis (PCA) subspace as shown in Figure 2 (a). Each color represents a cluster group classified by k-means. Figure 2 (b) shows the centroids for each cluster. Figure 2 (c) shows the number of workloads in each classified group. A well-known feature of the k-means method is that it makes approximately same size groups. Actually, we can see that each cluster has a tendency to equally partition into seven groups. However, we intuitively think that the cluster size is not equal; therefore we will attempt to use other methods in our future work.

Figure 2 (d) shows a heat map using standardization by each cluster. In cluster-1, all the metric values are lower than the mean. Also, these workloads used a small number of nodes in less time. In other words, the workloads are ignorable.

In cluster-2, while these workloads used a small number of nodes as well, the workloads have higher values in the several metrics including "mem_th." However, in Figure 2 (e), "stream" was classified into cluster-2 even though it does not have the highest "mem_th" value in all the clusters. We think that the number of nodes in benchmarks used smaller than the real workloads. In other words, the number of nodes strongly affects the classification.

Cluster-3 is similar to cluster-2 in several metrics but used a larger number of nodes.

Cluster-4 used a smaller number of nodes than the mean but has arithmetic intensive characteristics remarkably. DGEMM belongs to this cluster. Also, the values of "use_nodetime" and "elapsed_time"



suite name	short name	cluster#
DGEMM	dgemm	4
STREAM	stream_copy	2
	stream_add	2
	stream_scale	2
	stream_triadd	2
IOR	ior	2, 7
IMB	imb	7
NPB-OMP	npb_omp_bt	2
	npb_omp_cg	1
	npb_omp_dc	2
	npb_omp_ep	1
	npb_omp_ft	1
	npb_omp_is	1
	npb_omp_lu	2
	npb_omp_mg	1
	npb_omp_sp	2
	npb_omp_ua	1
NPB-MPI	npb_mpi_bt	3
	npb_mpi_cg	7
	npb_mpi_dt	7
	npb_mpi_ep	3
	npb_mpi_ft	7
	npb_mpi_is	7
	npb_mpi_lu	3
	npb_mpi_mg	7
	npb_mpi_sp	7

Figure 2: A classification of the real workloads on K by k-means with #cluster=7. (a) Scatterplot of workloads (horizontal and vertical axes are the 1st and 2nd component of the PCA subspace, respectively.) (b) Centroids plotted on the PCA subspace. (c) The number of workloads for each cluster. (d) A heat map of the standardized values in the metrics vs. cluster#. (e) Belonging to the cluster number for the benchmarks used in the previous chapter.

are not small. These workloads of the cluster have an interesting characteristic.

Cluster-5 has I/O intensive characteristic with a larger number of nodes. Also, this cluster has the highest "max_use_mem" and high "flops" in all the clusters. We are impressed that the cluster exploits more compute resources than others.

Cluster-6 is also I/O intensive workload with a large number of nodes but has lower "flops" and "mem_th" values than the mean. We think that the cluster is a kind of data processing workloads.

Cluster-7 has values similar to cluster-6 except for "use_nodetime" and I/O metrics. Also, IOR unexpectedly belongs to this cluster. We think the same reason for "dgemm" in cluster-2 causes this result. We expect that actual workloads' I/O usage on K is greater than the benchmarks' I/O.

Finally, in terms of workloads' performance, we think that the characteristics of cluster-6 need to be examined in more detail because these workloads of the cluster used a node-time larger than the mean and have lower values in terms of arithmetic and memory access. In addition, the number of workloads in the cluster cannot be ignored; they are approximately 50,000. We think that the struggle to understand actual usage and check the performance of the workloads can help improve our operations and services.

6 FUTURE WORKS

Based on our study, we expect that a classification of the workload will help our screening process for detecting undeveloped applications. Our study will also provide insights to improve our system and operations. We will attempt to classify the workloads using other methods and metrics and introduce the results on the poster.

ACKNOWLEDGMENTS

Part of the results is obtained by using the K computer at the RIKEN Advanced Institute for Computational Science.

REFERENCES

[1] Fei Xing et al. 2014. HPC Benchmark Assessment with Statistical Analysis. *Procedia Computer Science* 29, Supplement C (2014), 210 – 219. https://doi.org/10.1016/j.procs.2014.05.019 2014 International Conference on Computational Science.

[2] K. Yamamoto et al. 2014. The K computer Operations: Experiences and Statistics. *Procedia Comp. Sci.* 29, Supplement C (2014), 576 – 585. https://doi.org/10.1016/j.procs.2014.05.052

[3] O. Tuncer et al. 2017. Diagnosing Performance Variations in HPC Applications Using Machine Learning. In *High Performance Comp. - 32nd Inter. Conf., ISC High Performance 2017, Frankfurt, Germany, June 18-22, 2017, Proc.* 355–373. https://doi.org/10.1007/978-3-319-58667-0_19

[4] Fujitsu Limited. 2010. SPARC64VIIIfx Extensions. (April 2010). Retrieved October 15, 2017 from http://www.fujitsu.com/downloads/TC/sparc64viiifx-extensions.pdf