File System and Power Management Enhanced for Supercomputer Fugaku

Hideyuki Akimoto Takuya Okamoto Takahiro Kagami Ken Seki
Kenichirou Sakai Hiroaki Imade Makoto Shinohara Shinji Sumimoto

RIKEN and Fujitsu are jointly developing the supercomputer Fugaku as the successor to the K computer with a view to starting public use in FY2021. While inheriting the software assets of the K computer, the plan for Fugaku is to make improvements, upgrades, and functional enhancements in various areas such as computational performance, efficient use of resources, and ease of use. As part of these changes, functions in the file system have been greatly enhanced with a focus on usability in addition to improving performance and capacity beyond that of the K computer. Additionally, as reducing power consumption and using power efficiently are issues common to all ultra-large-scale computer systems, power management functions have been newly designed and developed as part of the upgrading of operations management software in Fugaku. This article describes the Fugaku file system featuring significantly enhanced functions from the K computer and introduces new power management functions.

1. Introduction

RIKEN and Fujitsu are developing the supercomputer Fugaku as the successor to the K computer and are planning to begin public service in FY2021.

The Fugaku is composed of various kinds of system software for supporting the execution of supercomputer applications. **Figure 1** shows the system software components in the Fugaku. While the OS and application development environment are taken up in separate articles [1, 2], this article focuses on the file system and operations management software. The file system provides a high-performance and reliable storage environment for application programs and associated data. The operations management software mainly provides system management functions and job management functions.

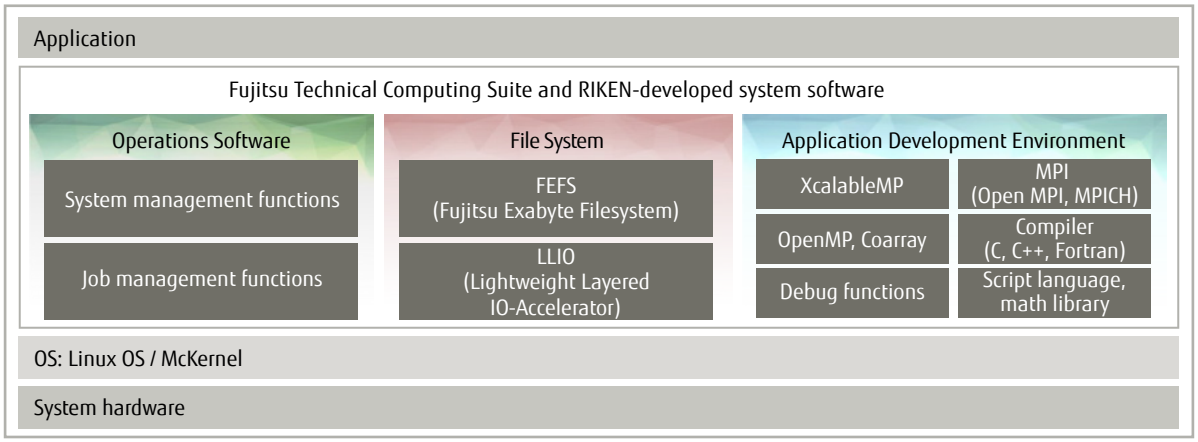


Figure 1
Fugaku software stack (configuration).

While inheriting the software assets of the K computer, the goal with Fugaku was to provide even higher computational performance and to use resources more efficiently. Furthermore, to broaden the range of supercomputer use, improvements and upgrades were undertaken in each of the software components shown in Figure 1 with a focus on flexible operation and ease of use.

In the file system, the Lightweight Layered IO-Accelerator (LLIO) was designed and developed as a new dedicated file system for the job execution area based on the Fujitsu Exabyte File System (FEFS) [3] developed for the K computer. LLIO aims to improve the usability of hierarchical storage and optimize application file I/O. Next, in operations management software, in addition to improving system monitoring and job scheduling performance, the application programming interface (API) for job scheduling and other tasks was enhanced and operations-customization functions for system managers were expanded [4]. Furthermore, to reduce system power consumption and promote the efficient use of power, which are issues common to all ultra-large-scale systems, power management functions in operations management software were newly designed and developed to enhance functionality.

This article describes the Fugaku file system featuring greatly enhanced functions centered about usability and introduces newly designed and developed power management functions as a functional enhancement of operations management software.

2. File system

To achieve a large-capacity, high-performance storage system, the Fugaku adopts hierarchical storage the same as the K computer [3]. In addition to providing high computing performance, the Fugaku is a system that envisions a broad range of users as a matter of policy. Its storage system must likewise satisfy this policy, and to this end, in addition to improving the usability of hierarchical storage, a function is needed for performance optimization according to file-access characteristics that differ from one application to another.

With these requirements in mind, we developed the LLIO as a new dedicated file system for the job execution area. LLIO provides three types of areas to an application according to file use thereby improving the usability of hierarchical storage and enabling the optimization of application file I/O.

In this section, we first present an overview of the Fugaku storage system. We then describe improvement in the usability of hierarchical storage through three types of LLIO areas and the optimization of temporary-file I/O. Finally, we present the results of LLIO performance measurements.

2.1 Overview of storage system

An overview of the Fugaku storage system is shown in **Figure 2**. The Fugaku hierarchical storage system consists of three layers: the 1st layer that serves as a dedicated high-performance area for job execution, the 2nd layer that provides a large-capacity shared area for use by both users and jobs, and the 3rd layer that provides commercial cloud storage [5]. At the time of this writing (early July 2020), the method for using cloud storage on the 3rd layer was still in preparation, so we here describe the Fugaku storage system with a focus on the 1st and 2nd layers.

To begin with, 1st layer storage uses no specialized server nodes; rather, it uses compute & storage I/O (SIO) nodes each equipped with an NVMe SSD that play the role of file system servers. Here, an assistant core [6] on the SIO node will process any file access request from a compute node. One SIO node exists for each bunch of blades (BoB) grouping 16 compute nodes. The launch of a job triggers the generation of a temporary LLIO file system that uses only the SIO node within the BOB in which that job was allocated. This LLIO file system is used by this job while executing and is released when job execution completes.

As shown in **Table 1**, LLIO provides three types of areas according to file use. The cache area of 2nd-layer storage enables the use of 1st layer storage without having to be concerned about different namespaces between layers, inter-layer data transfers, etc. The shared temporary area is used to store temporary files shared between compute nodes. The node temporary area is dedicated to storing temporary files shared only within a compute node. These areas are described in more detail in subsections 2.2 and 2.3 below.

Connections between layers are made via an I/O network. Within the compute node group, compute & global I/O (GIO) nodes are connected to the I/O network and data transfers between layers are carried out via the GIO node. In addition, 2nd-layer storage is configured with multiple FEFS [3] the same as the K computer.

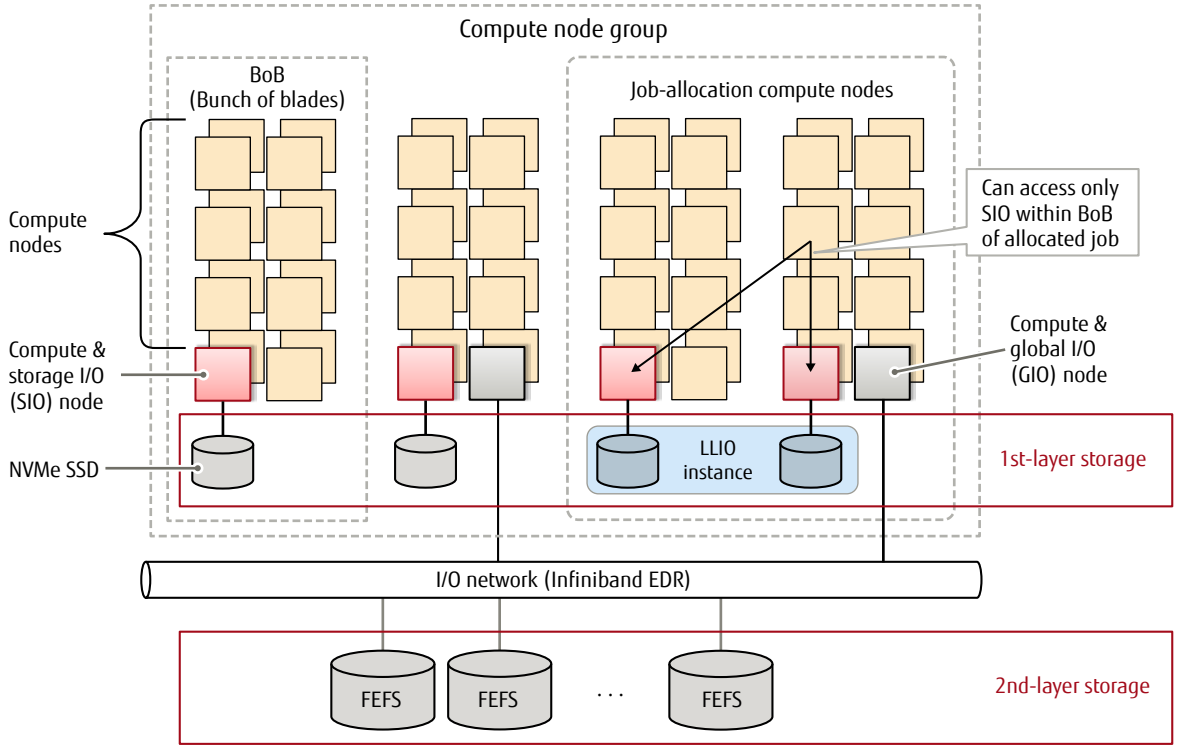


Figure 2
Overview of Fugaku storage system.

Table 1
Three types of areas in LLIO.

Area	Namespace	File Use
2nd-layer cache area	2nd-layer file system translucency	Stored file
Shared temporary area	Intra-job sharing	Temporary file
Node temporary area	Intra-node sharing	Temporary file

2.2 Improved usability of hierarchical storage

One issue in the use of hierarchical storage is the usability of data transfers between layers. Storage in the K computer consists of two layers each configured with an FEFS having a different namespace. For this reason, a staging system was adopted for carrying out data transfers between layers by explicitly specifying any stored files or other files the user needs for job execution [3]. For many users, however, appropriately selecting the files needed for job execution is not a trivial task, and an increase in data transfer time by specifying many unnecessary files has been a problem.

To address this problem, we adopted a cache system in the cache area of 2nd-layer storage in LLIO that improves usability of hierarchical storage by automatically performing data transfers between layers. The cache area of 2nd-layer storage provides the application with the same namespace as the 2nd-layer file system. If the application should issue a file READ request, LLIO will automatically read in and cache file data from 2nd layer storage to 1st layer storage. Meanwhile, if the application should issue a WRITE request, that data will be buffered in 1st layer storage and LLIO will write out that data to 2nd-layer storage asynchronously with application execution. This mechanism enables an application to use high-performance 1st layer storage without having to be concerned about different namespaces between layers, inter-layer data transfers, etc.

2.3 I/O optimization of temporary files

It is not unusual in a supercomputer application to write intermediate results of calculations to a temporary file that is then treated as an input file to subsequent calculations. A temporary file may be of a type that is shared between compute nodes or of a

type that is shared only within a compute node. Such temporary files are needed only during job execution, so using an area specifically for saving temporary files enables file I/O to be optimized. LLIO provides a shared temporary area and node temporary area as areas for saving temporary files.

The shared temporary area is used for saving temporary files to be shared between compute nodes. Here, the namespace of the shared temporary area is shared between the compute nodes to which the job has been allocated. Since no data is written to 2nd-layer storage here, no drop occurs in performance due to conflict between application file I/O and writing to 2nd-layer storage, which could occur in a cache area of 2nd-layer storage. In this way, the shared temporary area enables stable file I/O.

The node temporary area, meanwhile, is used for storing temporary files to be used only with a compute node. Here, the namespace of the node temporary area differs between compute nodes. Making the namespace of a node temporary area in one compute node independent of that of another compute node in this way makes a dedicated metadata server unnecessary. As a result, there is no concentration of load at a metadata server due to numerous attempts at file access, which enables high performance throughput proportional to the number of compute nodes.

2.4 LLIO file I/O performance

This subsection describes LLIO file I/O performance measured using 1,152 compute nodes in the Fugaku.

Figure 3 shows WRITE performance and READ performance for each of the three areas provided by LLIO for I/O sizes of 64 KiB, 1 MiB, and 16 MiB when executing the IOR [7] file-I/O performance benchmark. Looking at the results for WRITE performance, it can be seen that high throughput was achieved in each area regardless of I/O size. Next, the results for READ performance show that throughput of the cache area of 2nd layer storage improves as I/O size increases. The reason given for this is that a small I/O size increases software-processing overhead in an assistant core causing bottlenecks to occur.

In the future, we plan to conduct evaluations using multifaceted indicators in large-scale environments while also conducting and releasing measurements in relation to performance improvements in actual applications.

3. Power management functions

Fugaku was ranked No. 1 in the TOP500 list of the world's supercomputers announced at International Supercomputing Conference (ISC) High Performance 2020 Digital held in June 2020 [8]. This achievement verified the high program execution performance of Fugaku [8]. The power consumption on executing this benchmark test was 28,335 kW [9], which corresponds to the power consumed by about 70,000 households given a power consumption of 400 W for a typical household. Operating the Fugaku requires a reduction in unnecessary power consumption (enhanced power savings) as well as the provision of maximum computing power given a limited amount of power that can be consumed. A key element in meeting these requirements is operations management software.

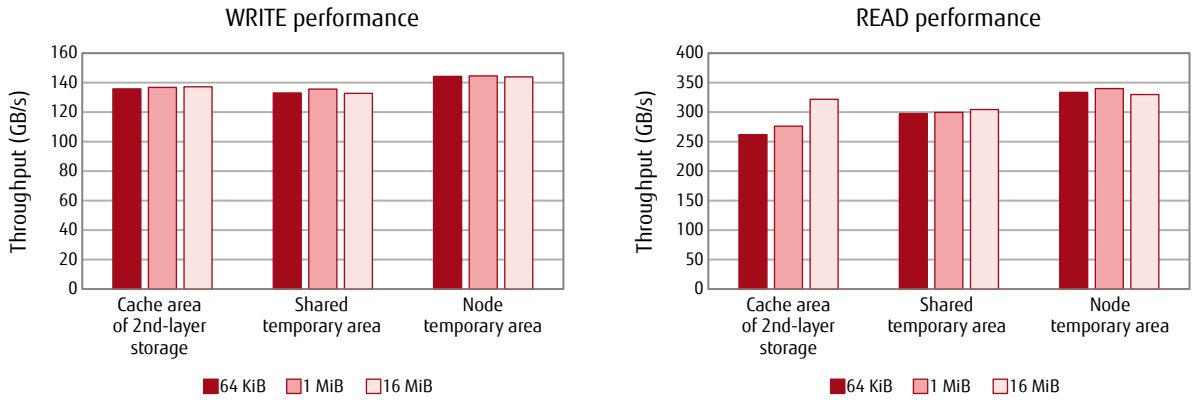


Figure 3
LLIO I/O performance evaluation.

In this section, we first present an overall picture of power management functions that we have newly designed and developed as part of the functional enhancements being made to Fugaku operations management software. We then describe power-consumption measurement functions that lead to power-control and optimization policies in Fugaku together with some measurement results.

3.1 Power management functions in Fugaku

The design and development of the Fugaku aimed to integrate hardware and software and enable effective use of power in the execution of applications.

On the hardware side, we designed and implemented mechanisms for dynamic power control and power measurement for CPUs, memory devices, etc. within compute nodes. Here, power control means dynamically changing the state of a device from software within a compute node according to the characteristics of the application (job) being executed to reduce power consumption and optimize computing performance per unit amount of power. Power measurement, on the other hand, aims to confirm and evaluate how power consumption in a compute node or device actually changed as a result of those power control measures.

Next, on the software side, we adopted Power API [10] as a power-control API for using the above mechanisms from within a compute node and designed and implemented associated software.

As part of Fugaku's operations management software, we designed and implemented power management functions that use Power API to link with job operation and control and perform power measurements. **Figure 4** shows the operation nodes of power management functions and their configuration.

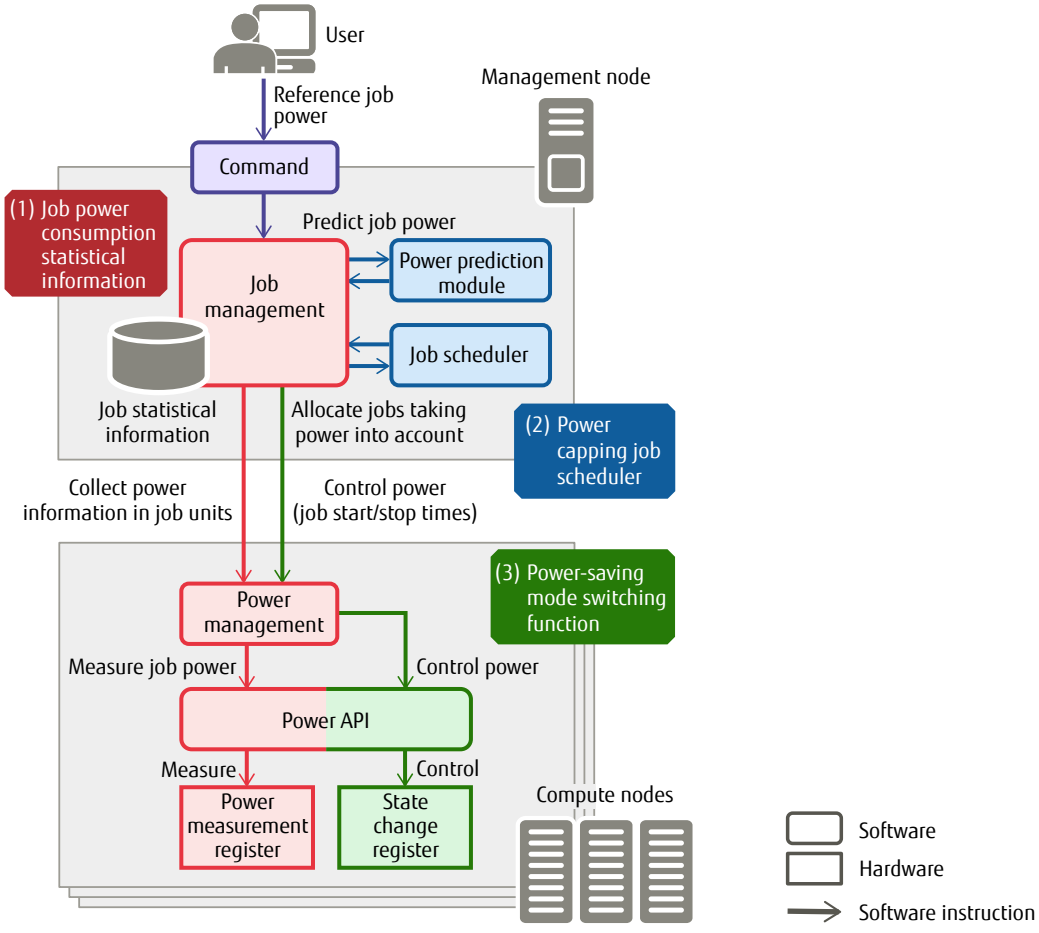


Figure 4
Operation nodes of power management functions and their configuration.

In Figure 4, function (1) is "job power consumption statistical information" that measures and consolidates the power consumed by a job. The power consumption of a job is recorded as job-related statistical information together with the number of compute nodes used, total execution time, etc. The charge levied on each user can be calculated on the basis of such information. This function is described in more detail in the following subsections. Next, function (2) is the "power capping job scheduler" [11] that controls job execution so that system power consumption does not exceed the target value. Finally, function (3) is the "power-saving mode switching function" that performs device control according to whether or not the compute node is executing a job as a means of power control that does not impact job execution performance. These power management functions are described in detail in a Fujitsu white paper [12].

3.2 Issues in measuring and evaluating power consumption

In system design, if the processing for executing jobs is uniform throughout, it is desirable that job execution exhibits the same performance and power consumption. In Fugaku, however, individual differences between compute nodes and different types of compute nodes will invariably lead to variation in power consumption.

In a supercomputer made up of multiple compute nodes, power consumption differs from one compute node to another due to variations in transistor characteristics originating in the semiconductor manufacturing process of CPUs and memory devices, optimization of operating voltage, etc. In addition, the typical method of using supercomputers is to run multiple jobs simultaneously on the system—it is rare to execute a job that uses all of the compute nodes in the system at one time. As a consequence, the compute nodes to be used whenever executing a certain job will differ depending on what compute nodes are already being used by other jobs, which means variation in power consumption each time that job executes.

In addition, some compute nodes in Fugaku serve a dual role as a compute node and I/O node–such a node will perform file I/O processing for storage or other purposes in addition to computational processing. This variability originates in the unique system design of Fugaku in which compute & I/O nodes and ordinary compute nodes have the following differences.

- Number of assistant cores

An ordinary compute node mounts two assistant cores. On the other hand, a compute & I/O node mounts two more to handle I/O processing for a total of four assistant cores.

- Mounted devices

A compute & I/O node mounts a storage device itself as well as a PCI Express (PCIe) device to connect to external storage.

3.3 Introduction of estimated power

As described above, a job uses different compute nodes every time it executes, so the power consumption of a job cannot be uniquely specified due to variation in the power consumption of compute nodes. To resolve this issue, we are designing and introducing "estimated power" in Fugaku as a power indicator unaffected by variation in power consumption in addition to measured power consumption (measured power) and promoting its use in evaluating job power consumption by the user.

1) Requirements

Our aim in introducing estimated power is to use it in collecting job statistical information in relation to fair power consumption and in optimizing job power consumption by the user based on power consumption trends. With this in mind, the following requirements must be met in the design of estimated power.

(a) The value of estimated power must be determined solely on the content of processing performed by the application program

(a-1) It must not be affected by variation in power consumption due to individual differences between compute nodes

(a-2) Variation in power consumption due to different types of compute nodes must be eliminated

(b) Estimated power must increase/decrease with increase/decrease in measured power

2) Design

To satisfy requirements (a-1) and (b), we check the activity of CPU and memory circuits and calculate estimated power based on those utilization rates. Next, to satisfy requirement (a-2) when calculating estimated power, we eliminate from the total value of the target

compute nodes the power consumed by any assistant cores used in operations other than those of that job and by PCIe devices that may or may not be mounted depending on the type of compute node. Furthermore, given that estimated power is calculated from the activity of various types of circuits, it is also possible to calculate power consumption in units of circuit blocks within a CPU and thereby provide power information of even higher granularity as an additional feature.

3.4 Evaluation of estimated power

We evaluated the validity of estimated power in terms of the following two items:

- Variation in estimated power
- Relationship between estimated power and measured power

In this evaluation, we used benchmarks included in the HPC Challenge Benchmark that measures a more realistic level of performance in a high performance computing (HPC) system. Since the power consumption of a compute node is heavily affected by the utilization rates of CPUs, memory, etc., we used the DGEMM benchmark that performs matrix-matrix multiplication as a routine generating high CPU load. We also used the STREAM benchmark that measures sustainable memory bandwidth by performing array vector operations as a routine generating high memory load. Moreover, to generate an even higher memory load in this benchmark, we changed the MULTIPLY instructions to XOR instructions in the SCALA operation, a vector operation that multiplies each element of a vector by a constant.

To evaluate change in estimated power due to differences in CPU and memory load factors, we used a total of 48 computing cores and varied the number of cores for executing the DGEMM and STREAM benchmarks. In this way, we executed programs with different load patterns each on 192 compute nodes in parallel and measured estimated power and measured power per compute node. We used Power API [10] installed in Fugaku operations management software to measure estimated power and measured power for a CPU operation frequency of 2.0 GHz.

1) Variation in estimated power

Table 2 lists the average value and standard deviation of estimated power and measured power per compute node for each load pattern. In comparison with measured power, these results show that variation in estimated power was kept to about 1/10 for all load patterns. In other words, estimated power can eliminate variation in compute nodes and thereby satisfy requirement (a).

2) Relationship between estimated power and measured power

Figure 5 shows the relationship between average estimated power and measured power across all load patterns. Here, the correlation coefficient R is nearly 1 and estimated power and measured power have a nearly proportional relationship. These results show that estimated power increases/decreases with increase/decrease in measured power thereby satisfying requirement (b).

4. Conclusion

In this article, we described the file system of the Fugaku and power management functions in operations management software as key improvements and enhancements over the K computer.

Table 2
Load patterns and relationship between estimated power and measured power per node.

Load Pattern		Estimated power per node (W)		Measured power per node (W)	
DGEMM No. of cores	STREAM No. of cores	Average	Standard deviation	Average	Standard deviation
48	0	150.3	1.3	166.6	13.6
40	8	169.2	1.2	195.3	14.2
32	16	175.5	1.3	204.3	14.1
24	24	172.2	1.1	200.5	13.4
16	32	171.5	1.1	200.4	13.5
8	40	166.7	1.1	194.3	13.2
0	48	161.3	1.1	180.8	11.7

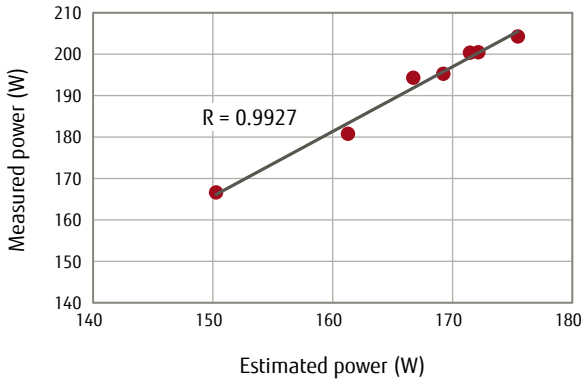


Figure 5
Relationship between estimated power and measured power.

First, for the file system, we described LLIO developed as a dedicated file system for the job execution area with the aim of improving the usability of hierarchical storage and enabling the optimization of application file I/O. The results of evaluating LLIO performance confirmed that this file system achieves high file I/O performance.

We then described the power management functions needed to reduce system power consumption and make efficient use of power/node resources as issues not limited to Fugaku but common to all ultra-large-scale computer systems. We also discussed issues in conducting power evaluations in supercomputer systems consisting of multiple compute nodes and described the results of designing and introducing "estimated power" as a solution to these issues in Fugaku. By using estimated power, a user executing a job can evaluate performance per unit amount of power without regard to the compute nodes on which the job is actually executed. Additionally, from the viewpoint of the operator, the use of estimated power makes it possible to gather statistical information on fair job power consumption independent of the compute nodes used. In this capacity, estimated power shows promise for use in actual operations.

Going forward, we will continue to make software adjustments in an environment based on actual use and operations with an eye to launching general shared use of the Fugaku. In this process, the plan is to evaluate performance and power consumption through actual large-scale applications and to release the results of those evaluations.

References and Notes

[1] L. Zhang et al.: OS Enhancement in Supercomputer Fugaku. Fujitsu Technical Review, No. 3, 2020. *https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article06.html*

[2] K. Watanabe et al.: Application Development Environment for Supercomputer Fugaku. Fujitsu Technical Review, No. 3, 2020. *https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article07.html*

[3] K. Sakai et al.: High-Performance and Highly Reliable File System for the K computer. FUJITSU Sci. Tech. J., Vol. 48, No. 3, pp. 302–309 (2012). *https://www.fujitsu.com/global/documents/about/resources/publications/fstj/archives/vol48-3/paper08.pdf*

[4] A. Uno et al.: Operations Management Software of Supercomputer Fugaku. Fujitsu Technical Review, No. 3, 2020. *https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article10.html*

[5] RIKEN Center for Computational Science: Fugaku System Configuration. *https://postk-web.r-ccs.riken.jp/spec.html*

[6] R. Okazaki et al.: Supercomputer Fugaku CPU A64FX Realizing High Performance, High-Density Packaging, and Low Power Consumption. Fujitsu Technical Review, No. 3, 2020. *https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article03.html*

[7] GitHub: IOR. *https://github.com/hpc/ior*

[8] Fujitsu: Fujitsu and RIKEN Take First Place Worldwide in TOP500, HPCG, and HPL-AI with Supercomputer Fugaku. *https://www.fujitsu.com/global/about/resources/news/press-releases/2020/0622-01.html*

[9] Top500.org: TOP500 LIST - JUNE 2020. *https://www.top500.org/lists/top500/list/2020/06/*

[10] Sandia National Laboratories: High Performance Computing Power Application Programming Interface (API) Specification. *https://powerapi.sandia.gov/*

[11] H. Akimoto et al.: Toward Job Operations Software for the Post-K Supercomputer Recognizing Upper Limit of System Power Consumption. IPSJ SIG Technical Report, Vol. 2015-HPC-152, No. 1 (2015) (in Japanese).

[12] Fujitsu: White Paper – Advanced Software for the FUJITSU Supercomputer PRIMEHPC FX1000. *https://www.fujitsu.com/downloads/SUPER/primehpc-fx1000-soft-en.pdf*

Hideyuki Akimoto
Fujitsu Limited, Platform Software Business Unit
Dr. Akimoto is currently engaged in the development of power management functions in operations management software of the supercomputer Fugaku.

Takuya Okamoto
Fujitsu Limited, Platform Software Business Unit
Mr. Okamoto is currently engaged in the development of the file system for the supercomputer Fugaku.

Takahiro Kagami
Fujitsu Limited, Platform Software Business Unit
Mr. Kagami is currently engaged in the development of power management functions in operations management software of the supercomputer Fugaku.

Ken Seki
Fujitsu Limited, Platform Development Unit
Mr. Seki is currently engaged in the development of system hardware for the supercomputer Fugaku.

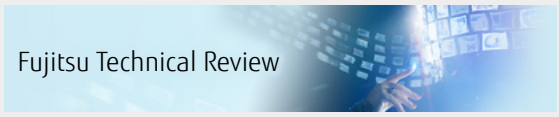
Kenichirou Sakai
Fujitsu Limited, Platform Software Business Unit
Mr. Sakai is currently engaged in the development of the file system for the supercomputer Fugaku.

Hiroaki Imade
Fujitsu Limited, Platform Software Business Unit
Dr. Imade is currently engaged in the development of power management functions in operations management software of the supercomputer Fugaku.

Makoto Shinohara
Fujitsu Limited, Platform Software Business Unit
Mr. Shinohara is currently engaged in the development of operations management software for the supercomputer Fugaku.

Shinji Sumimoto
Fujitsu Limited, Platform Software Business Unit
Dr. Sumimoto is currently engaged in the development of operations management software and language software for the supercomputer Fugaku.

This article first appeared in Fujitsu Technical Review, one of Fujitsu's technical information media. Please check out the other articles.



https://www.fujitsu.com/global/technicalreview/