Application Development Environment for Supercomputer Fugaku

Kensuke Watanabe Takafumi Nose Kiyofumi Suzuki Shuichi Chiba

The objectives of developing the supercomputer Fugaku (hereafter, Fugaku) are to achieve a system with up to 100 times greater application performance than that of the K computer and to ensure general versatility to support the operation of a broad range of applications. Accomplishing this required an integrated development environment that supports everything from the development to the execution of applications that extract the maximum performance from the hardware, but there were significant challenges in achieving this goal. First, it required development optimized for the CPU used by Fugaku and support for various applications. Moreover, it required support for new hardware and increased memory usage due to the increase in the total number of processes in the message passing interface (MPI) communication library. In addition, it required improvements of the practical usability of application development support tools. This article describes such initiatives for the application development environment.

1. Introduction

Supercomputer Fugaku (hereafter, Fugaku) [1] is equipped with the high performance "A64FX" CPU that adopts and was developed with Arm Limited's instruction set architecture (hereafter, Arm architecture), features general versatility that supports various applications, and achieves massively parallel processing through the Tofu Interconnect D (hereafter, TofuD) described below. The development goal for Fugaku was to achieve a system with up to 100 times greater application performance than that of the K computer [2]. Accomplishing this required an integrated development environment that supports everything from the development to the execution of applications that extract the maximum performance from the hardware. The developed Fugaku has demonstrated a high level of performance by taking first place worldwide in the four categories of TOP500 [4], HPCG [5], HPL-AI [6], and Graph500 [7] in the high-performance computing (HPC) rankings announced at the International Supercomputing Conference (ISC) 2020 [3].

This article introduces the application development environment compiler targeting Fugaku, the message passing interface (MPI) communication library, new features of the application development support tools, and initiatives undertaken to improve performance.

2. Compiler

This section discusses the technological development of the compiler for Fugaku.

The compiler for Fugaku supports three languages, Fortran/C/C++. The development objectives were to achieve up to 100 times greater application performance than that of the K computer for applications relating to social and scientific priority issues that should be selectively undertaken by Fugaku (hereafter, priority issue apps) and to enable the operation of various applications. In order to achieve these objectives, it is necessary to understand the features of the A64FX and to enhance the appropriate features. The following two issues were extracted through cooperative design (co-design) with the applications.

- Enhanced optimization to support the Arm architecture
- Support for the increasing use of object-oriented languages in recent years and optimization for integer type operations

The following subsections explain the initiatives with respect to these issues.

2.1 Supporting Arm architecture

Promoting the application of vector instructions for running applications at high speed and the application of optimizations to increase the parallelism of instructions are essential for creating an advanced compiler.

First, to promote the use of vector instructions, the Scalable Vector Extension (SVE), newly adopted by the Fugaku, was utilized. With SVE, it has become possible to use a predicate register to specify whether or not to execute operations on each element of the vector instructions. As a result, this enables the vectorization of complex loops that include IF statements and makes it possible to execute at high speed.

Next, software pipelining (SWP), which is a strength of the Fujitsu compiler, is important as an optimization for increasing the parallelism of instructions. While the K computer has 128 vector registers, Fugaku has a limited 32 registers. In order to effectively run SWP, which requires many registers even on Fugaku, loop fission optimization was enhanced. Loop fission is an optimization that considers the number of registers and the number of memory accesses for a loop with many statements and fissions the statements into loops to which SWP can be applied.

Figure 1 shows the evaluation results of the physical process kernel and the dynamic process kernel for NICAM [8], which is one of the priority issue apps. The former features loop structures with a lot of computational complexity and branch processing, while the latter features a high rate of memory throughput. As we can see, applying loop fission and SWP even for kernels with different characteristics can achieve a performance improvement of approximately 20%, which significantly contributed to the goal of achieving 100 times greater application performance of the K computer.

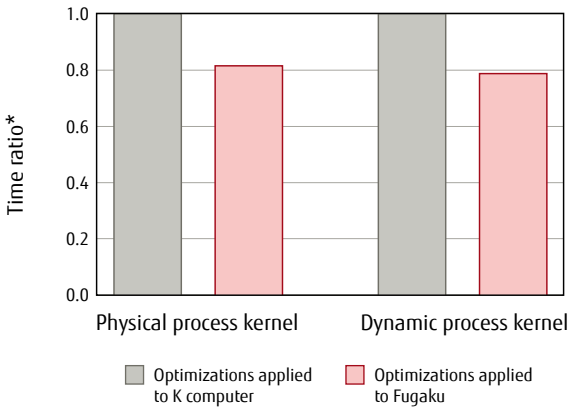

Figure 1
Optimization effects in the NICAM kernel.

2.2 Supporting object-oriented languages and integer type operations

In order to broaden the range of users with Fugaku, the next objective became the ability to execute applications from various fields in addition to conventional simulation programs. Therefore, it was necessary to enhance the optimizations for integer-related operations and object-oriented programs. Accordingly, the Clang/LLVM [9] open source software (OSS) compiler was adopted, which has a proven record of accelerating C/C++ language programs, as the base and ported the optimization technologies for HPCs that Fujitsu has previously cultivated. In addition, the concurrent use of conventional functions was enabled to ensure compatibility with programs used with the K computer and traditional programs to make it easy to port from the K computer.

3. Communication library

MPI is the de facto standard for the communication application programming interface (API) used with highly parallel applications in the HPC field, and the performance of the communication library with MPI implemented significantly impacts the practical usability of an HPC system. With the K computer, Open MPI was used, which is an OSS that has implemented the MPI standard, as the base and added unique improvements to obtain a high level of performance. Fugaku also inherited this library, but there were issues with the support for the new hardware and memory usage due to the increase in the number of processes.

This section discusses the communication library initiative for Fugaku with respect to these issues.

3.1 Supporting new hardware

With the K computer, we adopted the Torus Fusion Interconnect (hereafter, Tofu) [10] which is a node interconnect with a six-dimensional mesh/torus topology that connects many processors in higher dimensions. The Fugaku adopted Tofu Interconnect D (hereafter,

TofuD) [11] which, in comparison with Tofu, enhances the number of simultaneous communications, fault tolerance, and barrier synchronous communication, etc.

The TofuD installed in the Fugaku has an enhanced communication speed of 6.8 GB/s compared to the 5 GB/s of the Tofu used in the K computer. However, because the ratio of improvement for communication speed does not extend to that for computing performance, the conventional collective communication algorithm, which performs data broadcasting and reduction operations involving the coordination of each rank, cannot maximally utilize the computing performance under those conditions. Because the number of links capable of simultaneous communication was increased from four in the K computer to six in the Fugaku, it was important to utilize this to refine the collective communication algorithm to fully utilize the communication performance of the TofuD. In the Fugaku, six instances of simultaneous communication are transmitted via the five collective communication functions consisting of MPI_Bcast, MPI_Reduce, MPI_Allreduce, MPI_Allgather, and MPI_Alltoall. In particular, because the conventional MPI_Bcast algorithm for large message lengths is restricted to three instances of simultaneous communication, increasing that to six significantly improved the communication bandwidth (**Figure 2**).

Moreover, the communication function used for barrier communication [12] acceleration was enhanced in TofuD. By utilizing this enhancement, barrier communication acceleration can also be applied to patterns in which communications occur due to multiple processes within a node. Compared to the K computer, the data length that the TofuD barrier communication (hereafter, Tofu barrier communication) can be applied to has increased from one element to three elements (floating-point numbers) or six elements (integers). Due to the significant acceleration compared to a software implementation, even in a range that exceeds these numbers of elements supported by the hardware, there is a range in which further communication acceleration can be achieved (**Figure 3**) beyond a pure software implementation by repeatedly calling the Tofu barrier communication in the software. By adding an optional function that repeatedly executes Tofu barrier communication to the MPI library, users can use optimization to increase the applicable data length without modifying the program.

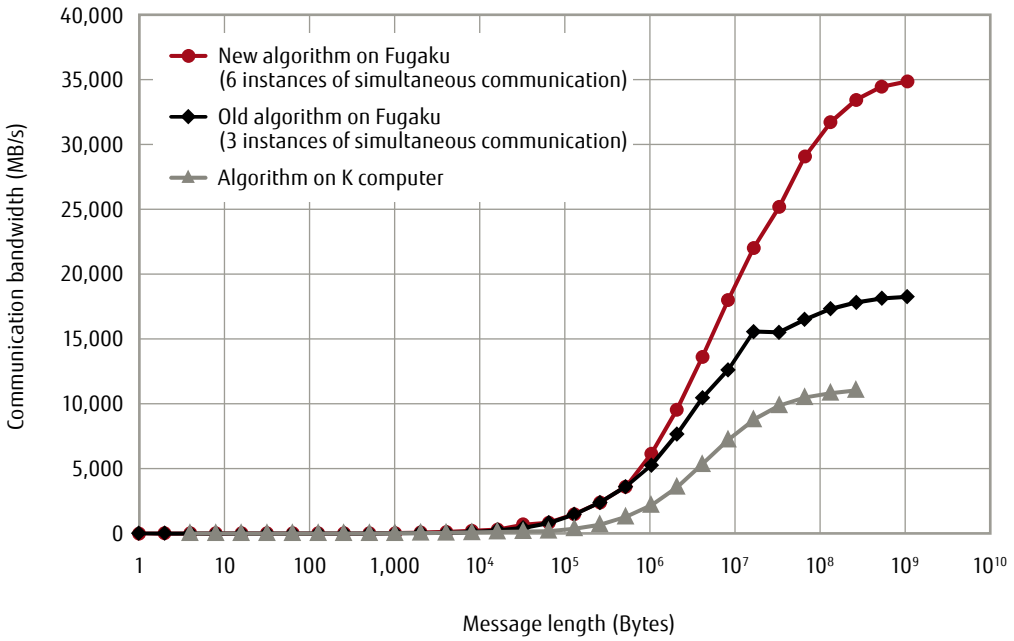

Figure 2
Performance of dedicated collection communication algorithm supporting six directions (Bcast).

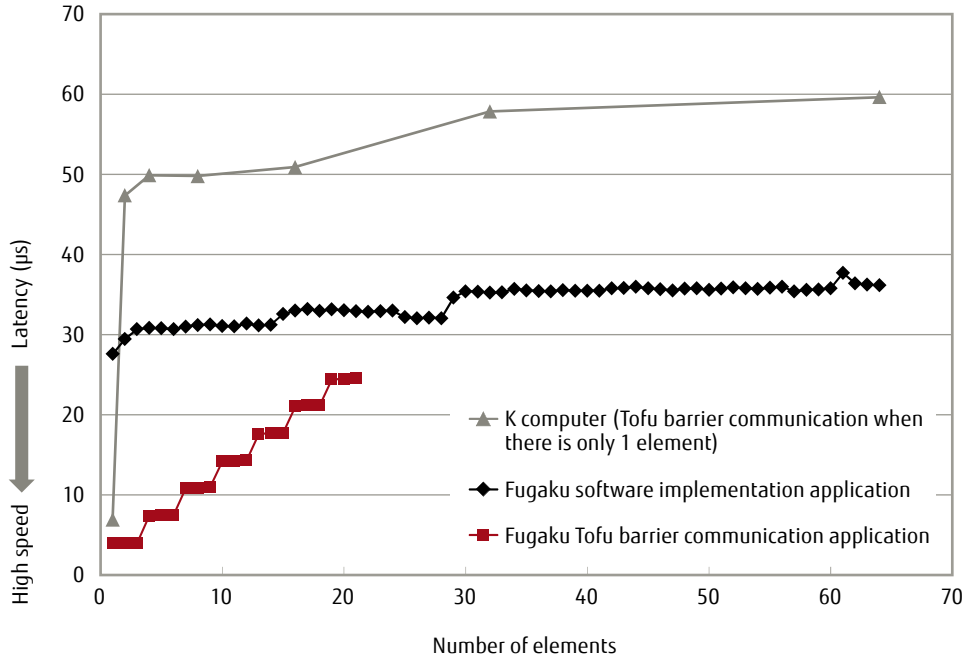

Figure 3
Accelerating barrier communication (MPI_Allreduce).

3.2 Memory usage control

In the K computer, one node is equipped with 16 GiB of memory, and one process monopolizes the entire memory space. However, while the memory capacity for one node has been increased to 32 GiB in the Fugaku, it has been split into four pieces through a structure within the node called the core memory group (CMG). Furthermore, because it is recommended to allocate multiple processes to a separate CMG, Fugaku shares the 32 GiB among the four processes. Therefore, the memory usage per process has decreased to 8 GiB. In addition, when viewed at the scale of the number of processes for the entire Fugaku system, because there are 150,000 nodes that can run four processes on each node simultaneously, we get a total of approximately 600,000 processes, which is a significant increase from the approximately 80,000 processes of the K computer. Generally speaking, because the management information required within the MPI library also increases in proportion to the number of processes that are connection partners of MPI communication, an increase in the overall system scale becomes a factor that places a strain on memory capacity and significantly impacts application performance.

To deal with this problem, the Fugaku MPI library does not allocate memory for all of the communication partners during initialization with the MPI_Init function, but instead dynamically allocates the memory when it first communicates with a communication partner. As a result, it does not increase simply depending on the total number of processes, but is improved by being able to limit the allocation to the necessary minimum depending on the communication pattern. Moreover, by implementing a communication mode utilizing a TofuD function that further reduces the memory consumption, it improves the flexibility with respect to applications that require a balance between process scale and memory consumption. In this communication mode, the maximum memory consumption for 27,648 nodes and 110,592 processes can be reduced by approximately 34%.

4. Application development support tools

This section describes the application development support tools developed for Fugaku.

The application development support tools are composed from three types of functions according to their purpose.

- An easy-to-understand display profiler that collects information for performance tuning
- Debugger for Parallel Applications for supporting the investigation of events such as deadlocks and abnormal termination that occur in large-scale parallel processing
- Integrated development environment that can perform operations such as compiling and submitting jobs through a GUI

The following subsections introduce the features and improvements for Fugaku of the most frequently used profiler.

4.1 Supporting Fugaku

A Fugaku-compatible profiler was developed by inheriting and expanding on the K computer's profiler, which was highly regarded from the perspective of practical usability. As shown in **Figure 4**, it consists of an "Instant Performance Profiler," an "Advanced Performance Profiler," and a "CPU Performance Analysis Report" that can perform efficient performance analysis by using these components in a staged manner depending on the situation.

4.2 Understanding performance trends with Instant Performance Profiler

The Instant Performance Profiler collects cost distribution information for the application with a sampling method to support an understanding of the outline of overall application performance by displaying the information at a procedure, loop, and line level. The hot spots where the costs are concentrated can be identified from the cost distribution.

With the K computer's profiler, there were requests from users who wanted to obtain the collected information in an easy to process and general-purpose format for creating graphs and statistical processing, etc. However, in reality the data was only provided in text and CSV formats, so supporting output formats with a high degree of general usability was an issue with Fugaku. In the

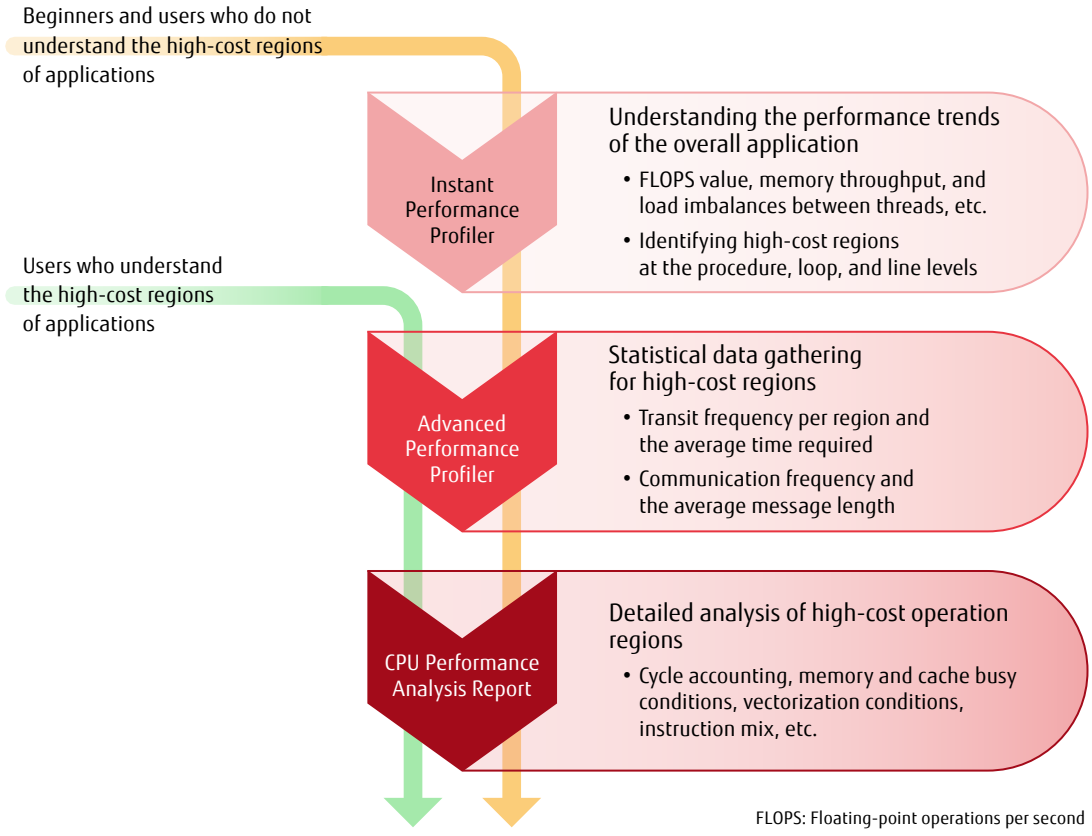

Figure 4
Performance analysis procedure.

Fugaku profiler, this issue was addressed by supporting output in XML format in both the Instant Performance Profiler and the Advanced Performance Profiler.

4.3 Understanding detailed information with Advanced Performance Profiler and CPU Performance Analysis Report

The Advanced Performance Profiler and CPU Performance Analysis Report are used to obtain more detailed performance information for hot spots identified from the results of the Instant Performance Profiler.

The Advanced Performance Profiler collects MPI communication cost information for specific sections and CPU performance analysis information to enable a detailed performance analysis of hot spots. The CPU Performance Analysis Report displays the information of Performance Monitoring Unit (PMU) counter, which is built into the CPU and measures the CPU operating conditions regarding operation performance, in a structured and easy-to-understand manner using graph and table formats (**Figure 5**). Analyzing PMU counter information makes it possible to understand the operation performance bottlenecks in detail.

A function that is similar to the Fugaku CPU Performance Analysis Report was achieved in the K computer with a precision PA visualization function in Excel format. However, the application had to be run seven times to gather the information, making the low level of convenience an issue. Moreover, in the case of the PRIMEHPC FX100 commercial unit that applied the K computer technologies, the application had to be run 11 times.

With the Fugaku CPU performance analysis report, a report can be created after an application is run 1 time, 5 times, 11 times, or 17 times, which means that the volume of information increases with the number of runs. Therefore, the CPU Performance Analysis Report can be used in a flexible manner to change the number of runs according to the required volume of information while also utilizing the newly available types of PMU counter information to achieve a more detailed analysis.

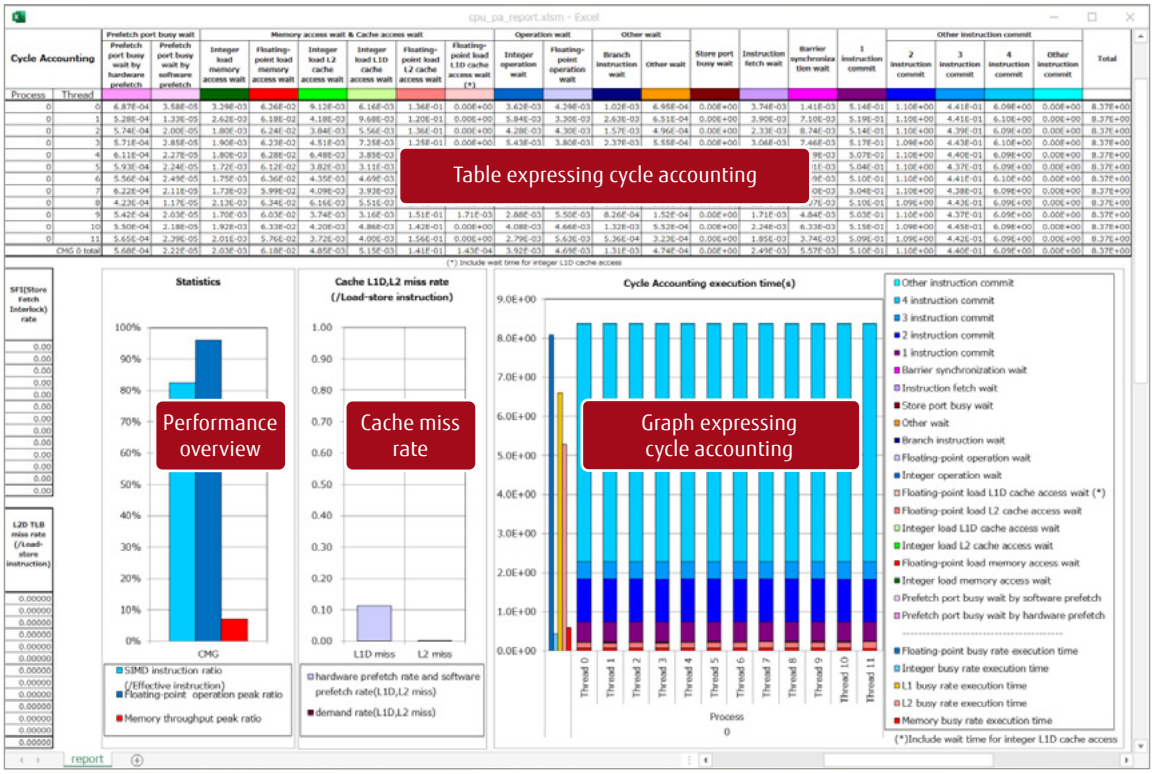

Figure 5
Example of a CPU Performance Analysis Report.

5. Conclusion

This article discussed the Fugaku compiler, MPI communication library, functions developed with the application development support tools, and initiatives to improve performance. Starting with the K computer and its commercial successors such as the PRIMEHPC FX10 and FX100 and continuing up to the Fugaku, changes were made to the architecture and the hardware, which required support for major software changes every time. However, resolving these issues by incorporating clever solutions in each software version significantly contributed to achieving the Fugaku development objective of reaching 100 times greater application performance than that of the K computer.

Fugaku has taken the top spot on the TOP500 and other categories at ISC 2020. Going forward, software improvements will be essential in order to use Fugaku and commercial units such as the PRIMEHPC FX700 and FX1000 as cutting-edge research and development platforms. While placing importance on user convenience, the developers of Fugaku would like to provide high-performance functions and tie those characteristics to innovative results in science and technology.

References and Notes

[1] The official name of the post-K computer decided by RIKEN in May 2019.
[2] The official supercomputer name decided by RIKEN in July 2011.
[3] ISC 2020.
https://www.isc-hpc.com/
[4] Ranking which takes the execution performance of the LINPACK program for solving simultaneous linear equations with a matrix calculation as an index and uses those results to rank the top 500 fastest computer systems.
[5] Benchmark test ranking that uses the conjugate gradient method, which is a computational method for solving simultaneous linear equations composed from a sparse coefficient matrix and is frequently used in actual applications.
[6] Benchmark test ranking which improves on the LINPACK benchmark and implements it with low-precision operations.
[7] Benchmark test ranking which involves the analysis of large-scale graphs, which indicate the relationships between data through peaks and branches.
[8] Abbreviation of Nonhydrostatic ICosahedral Atmospheric Model. Adopted as a priority issue application for Fugaku. In a dynamic process, it solves for the wind velocity, air temperature, etc. on a grid and features structured grid stencil calculations and a loop structure with high memory requirements. In a physical process, it solves for cloud phase changes and other phenomena and features a loop structure with a lot of computation and branch processing.
[9] Clang/LLVM.
https://clang.llvm.org/
[10] Y. Ajima, et al.: "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers." IEEE Computer, Vol. 42, No. 11, pp. 36–40 (2009).
[11] Y. Ajima et al.: The Tofu Interconnect D. IEEE International Conference on Cluster Computing, pp. 646–654 (2018).
[12] Communication for the synchronization of multiple MPI processes. With the TofuD, it not only performs synchronization, but is also capable of simultaneously performing small amounts of Reduce and Allreduce operations.

Kensuke Watanabe
Fujitsu Limited, Platform Software Business Unit
Mr. Watanabe is currently engaged in compiler development for the supercomputer Fugaku.

Takafumi Nose
Fujitsu Limited, Platform Software Business Unit
Mr. Nose is currently engaged in MPI communication library development for the supercomputer Fugaku.

Kiyofumi Suzuki
Fujitsu Limited, Platform Software Business Unit
Mr. Suzuki is currently engaged in the development of application development support tools for the supercomputer Fugaku.

Shuichi Chiba
Fujitsu Limited, Platform Software Business Unit
Mr. Chiba is a manager of software development for the supercomputer Fugaku application development environment.