White paper
Advanced Software for the FUJITSU Supercomputer PRIMEHPC FX1000

Fujitsu Limited

Contents

Features of HPC Middleware

HPC middleware overview

The HPC middleware *FUJITSU Software Technical Computing Suite* (or simply Technical Computing Suite) developed by Fujitsu provides a exascale system operation and application environment for the K computer(*1) and other supercomputers. The CPUs mounted in the FUJITSU Supercomputer PRIMEHPC FX1000 (or simply PRIMEHPC FX1000) have the ARM(*2) architecture and the versatility to support a wide range of software, including Technical Computing Suite. Fujitsu shares the experience and technology gained from the development of this HPC middleware with the community so that we can improve HPC usability together.
The structure of the HPC middleware in the PRIMEHPC FX1000 is shown in the following figure, with an overview provided below.

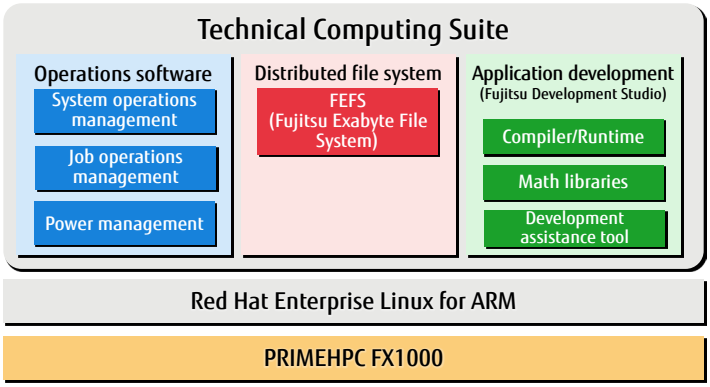

Figure 1 HPC middleware structure

Operating system

Running a Red Hat Enterprise Linux distribution as the OS, the PRIMEHPC FX1000 can be used in the same way as x86 clusters. Operability can be improved using scripts in Python, Ruby, or another language. Furthermore, the PRIMEHPC FX1000 has OS assistant cores mounted for the following purposes.

- Reducing system noise
  System noise (OS jitter) interferes with job execution. For example, in a system with tens of thousands of nodes, the interference increases in proportion to the increase in the number of nodes and may degrade performance.
  In a typical Linux server, the average noise ratio is at the 10^{-3} level, whereas in the PRIMEHPC FX1000, the average noise ratio is at the 10^{-5} level, cutting system noise to one-fifth of PRIMEHPC FX100.
  The PRIMEHPC FX1000 also has improved computational parallelism and throughput for massively parallel programs.

Table 1 Comparison of system noise

	PRIMEHPC FX1000	PRIMEHPC FX100	x86 cluster
Average noise ratio	1.02×10^{-5}	5.10×10^{-5}	1.08×10^{-3}

- Asynchronous communication processing
  Computations can be performed with the MPI non-blocking communication function during MPI communication within a user program (the period of communication is from a communication start function call to a communication wait function call).
  In the PRIMEHPC FX1000, an MPI asynchronous communication thread is allocated to an assistant core, and communication and computation are processed asynchronously. The MPI asynchronous communication thread is processed with high priority even if the CPU is overloaded. This processing is implemented in the Linux kernel as the guaranteed response function for MPI communication.

- Promoting the separation of file I/O processes and jobs
  I/O processes are routed to use the FEFS (Fujitsu Exabyte File System) connected to an InfiniBand or other data transfer network from a compute node that is interconnected through an interconnect. In the K computer and the PRIMEHPC FX10, I/O processes are routed by I/O exclusive nodes (compute nodes that are also relay nodes). But in the PRIMEHPC FX1000, the processes are routed by the assistant cores, which eliminate the need for I/O exclusive nodes. Also, the assistant cores can also process the FEFS client function of compute nodes, so file I/O processes and job communication processes can be separated.

Operations software

Many users can execute their programs concurrently because the operations software allocates computer resources effectively. The software helps the whole system continue to operate even if a single point of failure occurs. The software also helps integrate and facilitate management of a large-scale system.

- System operations management
  This software provides an integrated centralized system management view for a large-scale system that has hundreds to tens of thousands of compute nodes. From the centralized management view, operators can easily control system start and stop, and enable automatic system operations for failure monitoring and isolation of faulty components. As an extension of the base technology for large-scale system management from our experience with the K computer, the software can also manage hybrid configurations of PRIMEHPC FX1000 and x86 clusters.

- Job operations management
  This software makes it possible not only to execute a single job that uses tens of thousands of nodes but also to effectively execute a wide variation of numerous user jobs. Many user jobs can share large-scale system resources among themselves when sharing and job priorities have been set for the job scheduler.

- Power management
  A newly provided power management function leverages hardware power performance to the fullest.

■ Distributed file system (FEFS)

Based on the file system of the K computer, the FEFS is a high-speed file system supporting large-scale systems. The following FEFS features enable high-speed access without delays in a large-scale environment.

- Hundreds of thousands of clients can stably access a file system consisting of thousands of storage devices (OSS/OST)(*3).
- The file system can support large volumes of data up to the exabyte level.
- The massive file I/O of one user does not affect the file I/O of other users.

■ Application development (Fujitsu Development Studio)

This integrated software suite is used to develop (compile, debug, tune, etc.) and execute scientific computing programs written in Fortran, C, or C++. It supports parallelization technology such as automatic parallelization, OpenMP, and MPI.

- Compiler/Runtime, math libraries
  They support instructions in new HPC extension of Arm, Scalable Vector Extension(SVE), and new language standards. Scientific computing programs can perform high-speed parallel processing by leveraging the PRIMEHPC FX1000 hardware functions.
  The upward compatibility option for PRIMEHPC FX10 and PRIMEHPC FX100 applications supports continued use of user assets.

- Development assistance tool
  The application development environment makes available the process cost information for the large page memory allocation function and deallocation function. It can graphically display the information collected from the application layer by the operating system and effectively tune applications for a massively parallel computer.

New Technical Computing Suite initiatives

We are working on the performance improvement to answer the massive exascale operations and the improvement of convenience in operation of private supercomputer system by mid- and small-scale centers and business users.

■ High-speed execution of applications

With improved usability and performance, the development assistance tool has the following features:

- High-speed execution of applications using SVE, as SIMD efficiency has increased due to enhanced compiler optimization
- Improved usability in a de facto standard development environment using Eclipse(*4)

■ Efficient use of parallel computing systems

Functional enhancements for operations software are improving the system utilization rate.

- Users can select the virtual environments optimized for their jobs when submitting the jobs.
- Scheduling performance is 10 times better than with the old system, and responsiveness for job information display has improved.
- The efficiency of maintenance is higher with the rolling update function, which makes possible partial maintenance that does not stop the operation of the whole system.
- The abundant APIs available are compatible with schedulers from other companies. User-specific scheduling algorithms can be installed as add-ons, improving support for the myriad requirements of joint research centers.

■ Power-saving operation with power management functions

Power consumption by jobs can be taken into account as a computational resource in job scheduling. This feature can prevent computer centers from exceeding their power consumption capacity. Also, end users can optimally tune the power efficiency of their applications with the power measurement/control API.

*1 The K computer has been jointly developed by RIKEN and Fujitsu. K computer is a registered trademark of RIKEN.
*2 ARM and the ARM logo are trademarks or registered trademarks of ARM Limited or its affiliates.
*3 The OSS (object storage server) is a server for controlling file data. The OST (object storage target) is a storage device connected to the OSS.
*4 Eclipse is a trademark or registered trademark of the Eclipse Foundation, Inc. in the United States and other countries.

Application Development

Application development environment overview

The PRIMEHPC FX1000 contains Fujitsu Development Studio, in addition to application development environments such as GCC included in Red Hat Enterprise Linux, to maximize PRIMEHPC FX1000 performance.
The PRIMEHPC FX1000 is a hierarchical distributed parallel supercomputer consisting of many cores and combining compute nodes with the Tofu interconnect D to share a memory space.

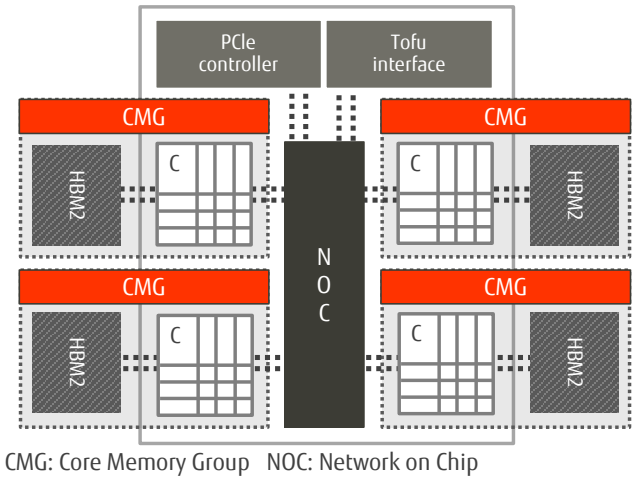

Figure 2 PRIMEHPC FX1000 compute nodes

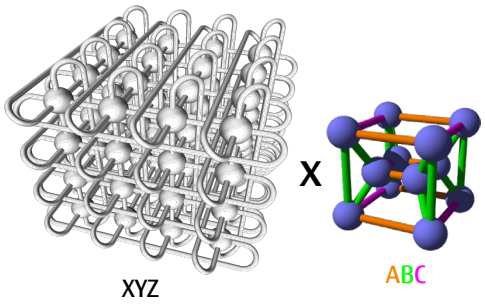

Figure 3 Tofu interconnect D

Fujitsu Development Studio provides Fortran, C, and C++ compilers and runtimes, math libraries, and a development assistance tool. With the tool, Fujitsu Development Studio maximizes PRIMEHPC FX1000 performance by speeding up processing on individual layers (i.e., cores), within a node, and between nodes.

Fujitsu Development Studio configuration

The C and C++ compilers of Fujitsu Development Studio can run in clang mode, which is compatible with Clang/LLVM. Open-source applications can run easily in clang mode.
Furthermore, to support various forms of use, Fujitsu Development Studio provides two compilers: a cross compiler that runs on the PRIMERGY, and the native compiler that runs on the PRIMEHPC FX1000.

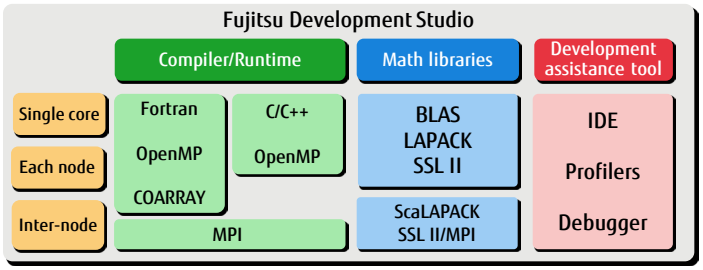

Figure 4 Fujitsu Development Studio configuration

Compliance with the latest standards and industry standard specifications

Fujitsu Development Studio complies with the latest standards and industry standard specifications, making it easy to develop applications based on the latest standards and to run open-source applications on the PRIMEHPC FX1000.

Table 2 Supported standards

Language	Supported specifications
Fortran	Part of ISO/IEC 1539-1:2018 (Fortran 2018 standard) ISO/IEC 1539-1:2010 (Fortran 2008 standard) ISO/IEC 1539-1:2004, JIS X 3001-1:2009 (Fortran 2003 standard) ISO/IEC 1539-1:1997, JIS X 3001-1:1998 (Fortran 95 standard) Fortran 90 and Fortran 77 standards
C	ISO/IEC 9899:2011 (C11 standard) ISO/IEC 9899:1999 (C99 standard) ISO/IEC 9899:1990 (C89 standard) * Extension specifications of the GNU compiler are also supported.
C++	Part of ISO/IEC 14882:2017 (C++17 standard) ISO/IEC 14882:2014 (C++14 standard) ISO/IEC 14882:2011 (C++11 standard) ISO/IEC 14882:2003 (C++03 standard) * Extension specifications of the GNU compiler are also supported.
OpenMP	Part of OpenMP API Version 5.0 OpenMP API Version 4.5
MPI	Part of Message-Passing Interface Standard Version 3.1 and 4.0 (tentative name)

Core speed-up functions

- Vectorization functions

The Fortran, C, and C++ compilers of Fujitsu Development Studio have vectorization functions that utilize the SVE (Scalable Vector Extension) function of the A64FX mounted in the PRIMEHPC FX1000.

SVE can be used to vectorize complex loops that contain IF statements and math functions. Furthermore, vectorization using a SIMD instruction with SVE masking can speed up a loop that has a small number of repetitions, on which vectorization has had little effect in the past. A unique feature of SVE is that the created SIMD binaries with varying SIMD register widths can be transported between machines that have different SIMD register widths.
Figure 5 shows the performance for different data access patterns of a SVE instruction.
The data access speed of the PRIMEHPC FX1000 is 2.0 to 2.3 times better than the speed of the PRIMEHPC FX100.

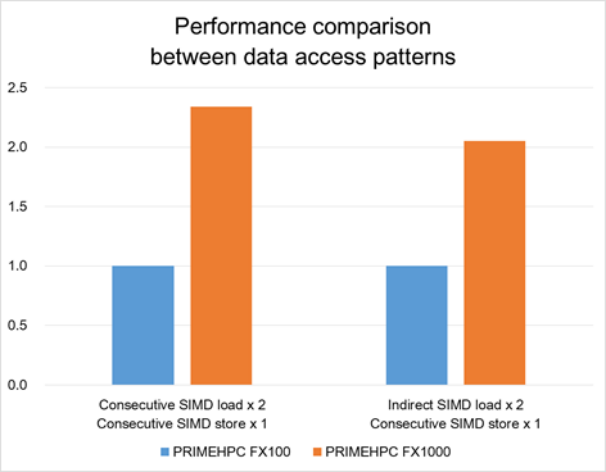

Figure 5 Performance comparison between data access patterns

■ SIMD register utilization according to the data type

The A64FX can make parallel calculations using a 512-bit width SIMD registesr with 8, 16, and 32 elements for double-precision, single-precision, and half-precision floating-point numbers, respectively, and 8, 16, 32, and 64 elements for 8-byte, 4-byte, 2-byte, and 1-byte integers, respectively. The Fortran, C, and C++ compilers of Fujitsu Development Studio use this function to improve the performance of scientific calculation applications, which use mainly floating-point calculations, and a wide range of applications involving integer calculations.

■ Software pipelining function

The Fortran, C, and C++ compilers of Fujitsu Development Studio have a sophisticated instruction-scheduling function called software pipelining.
The software pipelining function arranges the order of the instructions in a loop in a program such that one cycle overlaps the next cycle. This arrangement considers the number of computing units, the latency of individual instructions, the number of registers, and so forth when increasing the parallelism of the instructions in the loop for faster execution.
The Fortran, C, and C++ compilers also have the following functions that promote software pipelining.

- Loop segmentation function
  A greater number of instructions in a loop inhibits software pipelining because there is a shortage of registers. The loop segmentation function promotes software pipelining by segmenting a loop containing many instructions into multiple loops so that software pipelining can be applied to the loops. To do so, the function considers the number of registers, the number of memory access times, cache efficiency, and so forth.
- No-branch type of inline expansion of math functions
  A loop that calls a math function cannot be pipelined because the processing requires conditional branches even if the function is expanded inline. The Fortran, C, and C++ compilers of Fujitsu Development Studio promote software pipelining through inline expansion without branches for trigonometric functions, exponential functions, and square roots. To do so, the compilers use the trigonometric function auxiliary instruction, exponential function auxiliary instruction, and reciprocal approximation instruction.
  Figure 6 shows the performance of basic calculations and math functions. The PRIMEHPC FX1000 has 2.3 to 4.2 times better performance than the PRIMEHPC FX100 for loops that include basic calculations and math functions.

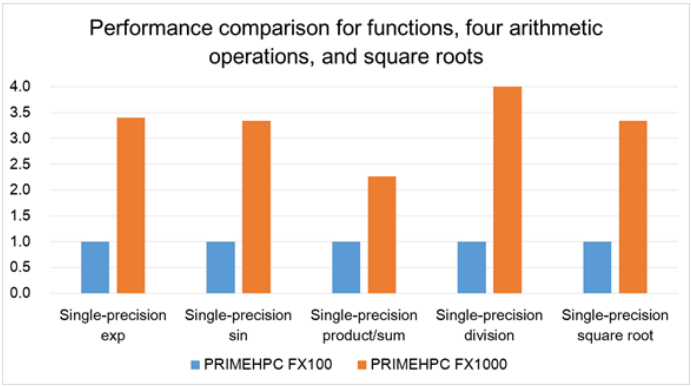

Figure 6 Performance comparison for functions, four arithmetic operations, and square roots

■ Intelligent prefetch function

The A64FX has two prefetch functions to prevent the processing slowdown caused by a cache miss. One is a hardware prefetch function with automatic memory prefetching by hardware for simple memory access patterns, and the other is a prefetch instruction that supports complex memory access patterns.
The intelligent prefetch function of the Fortran, C, and C++ compilers of Fujitsu Development Studio leaves simple memory access patterns to be handled by the high-performance hardware prefetch function. If the hardware prefetch function cannot handle a memory access pattern, this function generates a fetch instruction for that pattern.
This intelligent prefetching includes the function of tuning the prefetch interval for each loop to streamline prefetching.

■ Sector cache instruction function

The sector cache function of the A64FX makes it possible to use part of the cache as high-speed memory.
The Fortran, C, and C++ compilers of Fujitsu Development Studio have the sector cache instruction function, which uses a simple instruction line to specify the data to be stored in the sector cache. The function can be used to store repeatedly used data in high-speed memory to prevent performance degradation.

■ FP16 (half-precision floating-point type) support

The A64FX and the Fortran, C, and C++ compilers of Fujitsu Development Studio support the FP16 type.

For some types of AI algorithms such as those for machine learning, arithmetic precision can be reduced to the FP16 type. The performance of SIMD calculations is two times better with FP16 than with conventional FP32 (single-precision floating-point type). Using FP16 can improve the performance of not only conventional scientific calculation applications but also AI applications.

■ Faster math libraries using internal core speed-up functions

Fujitsu Development Studio provides fast math libraries. Internal core speed-up functions have been applied to these libraries, listed in the following table. BLAS and LAPACK are well-known in the linear algebra field, Fujitsu math library SSL II is used by a wide range of R&D users because its algorithms cover many fields, and the fast quadruple-precision basic-arithmetic library delivers high performance by handling quadruple-precision values in the double-double format. Applications can be faster simply by calling these math libraries.

Table 3 Math libraries (Sequential libraries) (thread-safe)

Sequential libraries (thread-safe)	
BLAS, LAPACK	These linear calculation libraries developed in the United States and released by Netlib are a de facto standard. BLAS has about 80 routines, and LAPACK has about 400 routines. Some BLAS routines support FP16.
SSL II	Library of about 300 Fortran routines covering a wide range of fields
C-SSL II	C interface for SSL II
Fast quadruple-precision basic-arithmetic library	This library represents and calculates quadruple-precision values in the double-double format. There are some thread parallelization routines.

Figure 7 shows the performance of a BLAS matrix product routine. The BLAS matrix product routine on the A64FX shows that the single-precision type has two times better performance than the double-precision type and FP16 has two times better performance than the single-precision type.

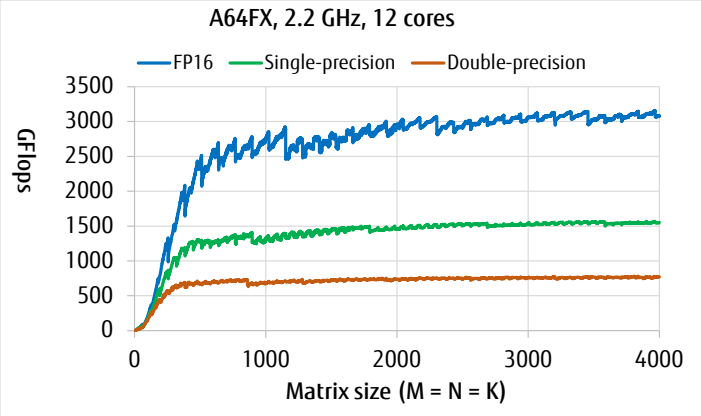

Figure 7 Matrix product performance

Higher speeds with internal node parallelization

■ Thread parallelization function

To use many cores in a PRIMEHPC FX1000 node, the Fortran, C, and C++ compilers of Fujitsu Development Studio support an industry standard specification, OpenMP, which provides automatic parallelization for loops and thread parallelization.

The A64FX has a high-speed hardware barrier mechanism for thread parallelization. The runtime library of Fujitsu Development Studio uses this hardware barrier for high-speed execution of parallelized threads.

■ Faster MPI with support for network layers

The A64FX has four mounted CMGs (core memory groups) and employs the NUMA (non-uniform memory access) architecture to connect the CMGs through a ring bus. The PRIMEHPC FX1000 connects the A64FX through the Tofu interconnect D. As a result, in process parallelization across different CMGs, connections are made on the two network layers of the Tofu interconnect D and ring bus within the A64FX.

In addition to the Tofu interconnect D, the Alltoall algorithm of the Fujitsu Development Studio MPI includes an algorithm optimized to reduce communication path conflicts in the A64FX ring bus, as shown in Figure 8. This algorithm raises A64FX performance by maximizing ring bus performance as shown in Figure 9.

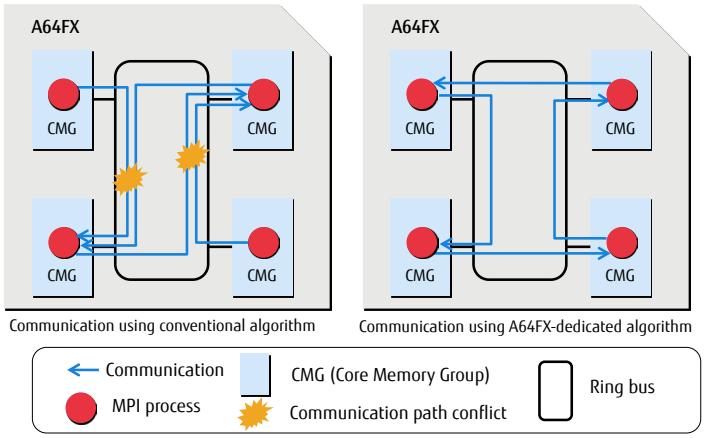

Figure 8 A64FX-dedicated collective communication algorithm

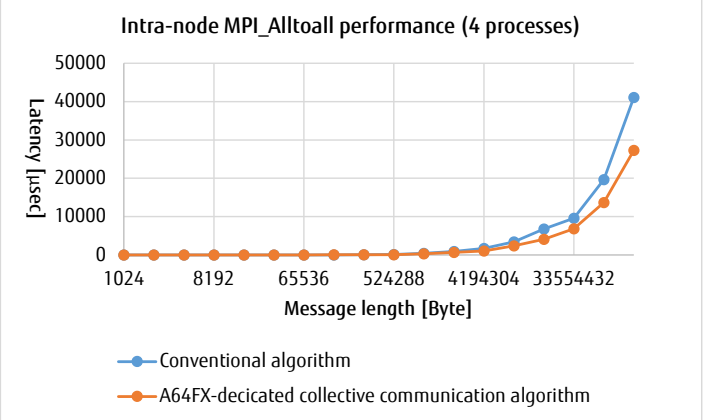

Figure 9 Performance of A64FX-dedicated Alltoall algorithm

■ Faster math libraries using thread parallelization function

Fujitsu Development Studio provides fast math libraries supporting thread parallelization. The libraries are BLAS, LAPACK, and SSL II. Applications can be faster with thread parallelization by calling these math libraries.

Table 4 Math libraries (Thread parallel libraries)	
BLAS, LAPACK	The interface is the same as for the sequential libraries. Also included is PLASMA, which is a task parallel version of LAPACK. They can also be used from Python (NumPy, SciPy).
SSL II thread parallel functions	They are thread parallel versions of about 80 important routines. The interface is different from that of the sequential library SSL II, so they can be used concurrently.
C-SSL II thread parallel functions	C interface for SSL II thread parallelization

Higher speeds with parallelization between nodes

■ Faster MPI using the Tofu interconnect D

The Tofu and Tofu2 mounted in the PRIMEHPC FX10 and PRIMEHPC FX100 have four network interfaces per node, whereas the Tofu interconnect D in the PRIMEHPC FX1000 has six network interfaces. This is a significant enhancement of communication capabilities from four-way simultaneous communication to six-way simultaneous communication.

The Fujitsu Development Studio MPI uses this enhanced feature to improve collective communication algorithms, including Allgather and Bcast, which require a wide communication bandwidth for each node.

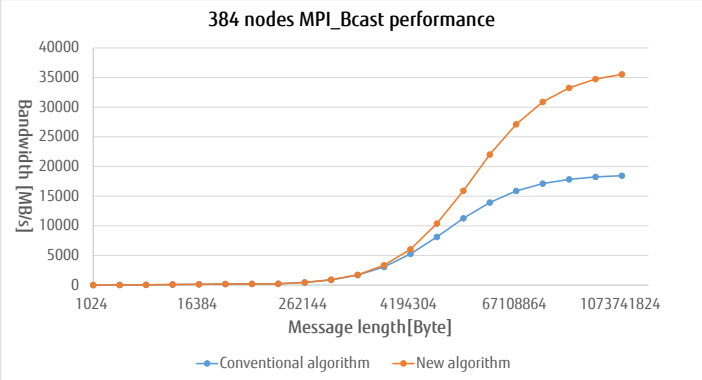

Figure 10 Performance of dedicated collective communication algorithm supporting 6-way communication (Bcast)

■ Faster math libraries using the MPI parallelization function

Fujitsu Development Studio provides fast math libraries having MPI parallelization. The libraries are ScaLAPACK and SSL II/MPI. Applications can be faster with MPI parallelization by calling these math libraries.

Table 5 Math libraries (MPI parallelization libraries)	
ScaLAPACK	MPI parallelization library of BLAS and LAPACK. This library has about 200 routines.
SSL II/MPI	MPI parallelization library with three-dimensional FFT functions

Application development assistance tool

■ Integrated development environment

Fujitsu Development Studio has Eclipse, which is an integrated open-source development environment, and Parallel Tools Platform (PTP), which is an extension plug-in for application development in the scientific calculation field.

With PTP, users can check and compile source code, submit jobs, check the job status, and retrieve/display performance information seamlessly on Eclipse using the job scheduler.

■ Profilers

- Performance analysis procedure

  Depending on conditions, users can employ one of two performance analysis procedures. Performance can analyzed efficiently with the basic profiler, detailed profiler, and CPU performance analysis report shown in the following figure, depending on conditions.

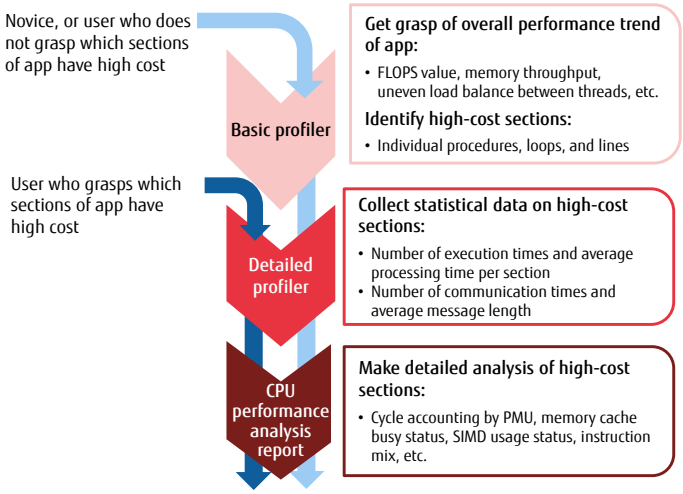

Figure 11 Performance analysis procedure

- Step-by-step performance analysis with profilers

  The basic profiler, detailed profiler, and CPU performance analysis report support performance analysis of applications.

  The basic profiler takes samples of the overall performance trend and cost distribution information for an application. The collected cost distribution information helps a user grasp the overall application performance from displayed costs for individual procedures, loops, and lines.

  The detailed profiler is used to obtain more detailed performance information on hotspots identified by the basic profiler. With the detailed profiler, users can get a grasp of performance conditions and the statistical communication status of the MPI in the specified section.

  The CPU performance analysis report uses graphs and tables to systematically and clearly display the PMU (performance monitoring unit) counters contained in the CPU. With the CPU performance analysis report, users can get a grasp of application bottlenecks in detail.

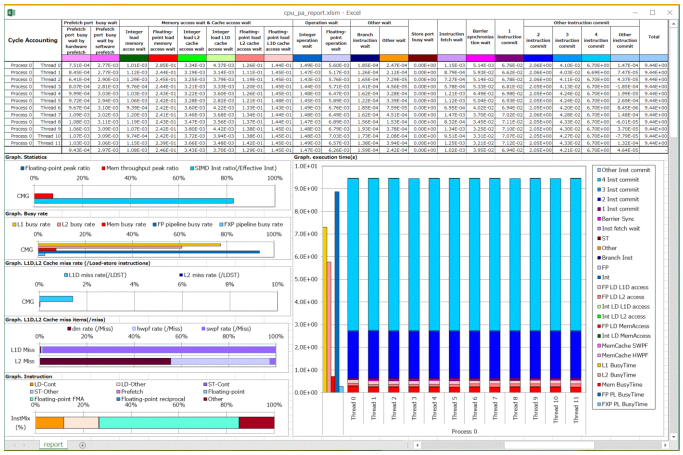

Figure 12 Example of a CPU performance analysis report

- Parallel execution debugger

Fujitsu Development Studio has three debugging functions useful in actual application execution scenarios. They support the investigation of frequently occurring problems (abnormal end, deadlock, etc.) in large-scale parallel processing. The de facto standard GDB is the debugger engine used for the parallel execution debugger.

- Abnormal end investigation function
  This function collects execution information for an application that abnormally ended with a signal during large-scale parallel execution. The collected execution information (bug trace, series of instructions including the signal issuing address, register contents, and memory map) is displayed clearly and succinctly by the function.
- Deadlock investigation function
  An application may not end or not return a response during large-scale parallel execution. This function collects application execution information (bug trace, local/argument variable values for each frame, and memory map) from all the processes of such an application. The collected information is displayed clearly and succinctly by the function.
- Debugging function using command files
  This function enables GDB command files to be used for debugging in the job environment. Different command files can be specified for individual processes, enabling flexible debugging that suits application characteristics.

Job Operations Management

Purpose of developing Job schedulers

Users of the PRIMEHPC FX1000 can run many jobs without concern, even in a large-scale system, because of Fujitsu's continuous efforts to develop batch job execution environments (job schedulers). The PRIMEHPC FX1000 has a job scheduler based on two job schedulers. One is the large-scale job scheduler of the performance-proven K computer, which has approximately 80,000 nodes. The other is the PRIMEHPC FX100 job scheduler, which has an improved system operating ratio in small-scale centers and enhanced operation functions that work together with x86 server jobs. Developed in addition to these features, the PRIMEHPC FX1000 job scheduler has the following job operation functions required for large-scale systems.

- Functions for reliable control of many jobs (large-scale support)
  A large-scale system needs to control many jobs at the same time. And even when so many jobs are running, the large-scale system must not place stress on system users. To meet these requirements, the job scheduler has been enhanced for better performance.
- Support of myriad requirements for a joint research center (operability)
  A number of joint research centers run large-scale systems, with meticulous operation policies on job execution varying slightly from one center to another. For the PRIMEHPC FX1000, Fujitsu has added the following functions for flexible customization of job scheduling reflecting the operation policies on job execution at each center:
  - Job execution environments improving usability for users executing jobs
  - Customization functions that enable fine-grained operation of centers

Joint research centers operate with many users and run many different jobs requiring varying node sizes. The following function was developed to improve the system operating ratio in an environment involving a variety of jobs.

- High operating ratio function: Adaptive elapsed-time job scheduling
  High-priority jobs may cause the system operating ratio to deteriorate. For example, the operating ratio temporarily deteriorates when a large-scale job reserves a large number of nodes. To improve the system operating ratio, job scheduling segments time into small units to allow corresponding shifts in the timing of job execution.

These functional enhancements, discussed below, are characteristic of the PRIMEHPC FX1000.

Functions for reliable control of many jobs (large-scale support)

The improved performance of the job scheduler enables stress-free use of the system even when many jobs are running.

■ Improved job scheduling performance

The improved scheduling algorithm considers situations where many users submit many small-scale jobs. As a result of this improvement, scheduling is finished in a short time—for example, several seconds when 10,000 small-scale jobs are submitted simultaneously.

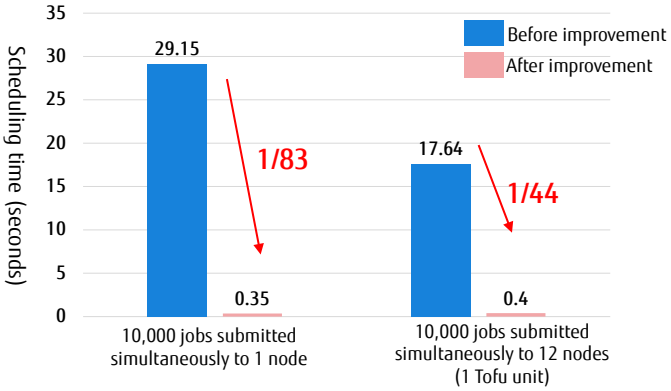

Figure 13 Reduction in scheduling time

■ Improved performance in reading job information

There was a problem with the amount of time taken to read job information when the number of jobs was large. Now, improved information search processing using an OSS (open-source software) database finds information quickly even when there are one million jobs, for example.

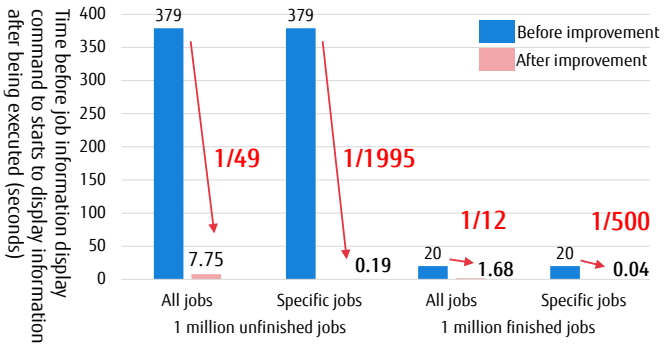

Figure 14 Reduction in job information reading time

■ Improved job operation performance

Improvements in job-related operations (delete, fix, change parameters) enable users to perform the operations without stress even when many jobs are targeted.

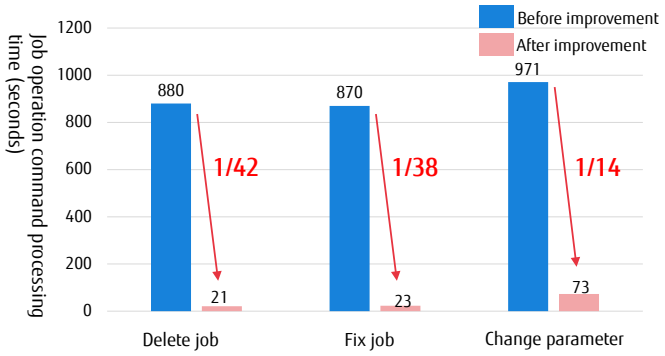

Figure 15 Reduction in job operation time

Job execution environments improving usability for users executing jobs

One function has been developed to deploy job execution environments specially tuned to the applications at individual centers. From the job execution environments made available to the centers and deployed in their systems, users can select and use the suitable environments for their jobs. Another function allows users to use the job execution environments that they themselves have prepared, in addition to the environments already deployed by the administrator. The available job execution environments are as follows.

- Docker mode
  In this mode, all the programs of a job, which include the user's job script, are executed on the container started from the container image specified at the job submission time. Since the OS packaged in the Docker image can be used to execute the job, it is easy to deploy the software environment required for application execution.
- KVM mode
  KVM can run users' job programs. KVM mode uses KVM together with the open-source software QEMU to manage virtual machines. With hardware-based virtualization support, jobs can be executed in this high-speed hardware virtualization environment. This mode is useful for developers of modules on the OS kernel layer and other users who need the privileges and rights to execute programs.

Customization functions that enable fine-grained operation of centers

The PRIMEHPC FX1000 job scheduler provides the following customization functions to optimize operation in user environments:

- Custom resource function for arbitrary resource scheduling
- Hook function that controls the flow of job execution
- Job statistics customization function that enables users to select statistics items and create new statistics items
- Scheduler plug-in function for users to incorporate their own scheduling algorithms
- Command API for users to create their own commands and operation support tools

Here are details on the enhanced job scheduler functions of the PRIMEHPC FX1000.

■ Custom resource function

Users can define any resource, such as a software license or the expected power consumption of a job, with this function. For example, a software license (floating license) that limits the number of concurrently used jobs and processes, can be defined as a custom resource. Likewise, a software license (node-locked license) that limits the number of nodes used, can be defined. Once a user specifies the necessary number of software licenses when submitting a job, the job is scheduled to run in a time slot with an allowable number of compute nodes.

■ Hook function

Available to the administrator, the hook function is a mechanism providing a process (exit process) that is triggered by the occurrence of a specific event during job execution. The administrator can use exit processes to control job execution and change job attributes. For example, the administrator can allow an existing process to check whether the budget allocated to the user executing a job is sufficient and to reject the execution of the job if it is insufficient.

■ Job statistics customization function

The job scheduler statistics function of the PRIMEHPC FX1000 job scheduler records job information (job statistics). The recorded information includes the quantity of nodes, CPU time, memory, and other resources used by a job, the job execution time, and the restriction values specified for job processes.
The job statistics customization function is used to select which items to record as job statistics. Users also use the function to define and record their own items. Statistics items defined by the administrator can be set to any value by the hook function described above. As with ordinary statistics items, users can choose to display their own defined statistics items as relevant information for end users or to record them as job statistics items.

■ Scheduler plug-in function

Using the scheduler plug-in function, the administrator can define and incorporate a scheduling algorithm in the PRIMEHPC FX1000 job scheduler to substitute it for the scheduling algorithm of the job scheduler. The scheduler plug-in function can optimize the job priority evaluation formula and control methods for the operation of a center, making possible more precise control by the administrator.

■ Command API

The command API is a set of interfaces providing functions (job operation and information retrieval) that are equivalent to the end user commands provided by the PRIMEHPC FX100 job scheduler. The administrator and end users can use the command API to enhance the functions of existing commands, create their own commands, and create utilities for operating the system or controlling system behavior.

High operating ratio function: Adaptive elapsed-time job scheduling

This function can specify a minimum execution time for a job when the job is submitted. A job with a specified minimum execution time is guaranteed to run for this minimum time. After the minimum execution time has elapsed since the start of job execution, if the job is not finished and nothing is preventing the job from continuing, the job continues to run.

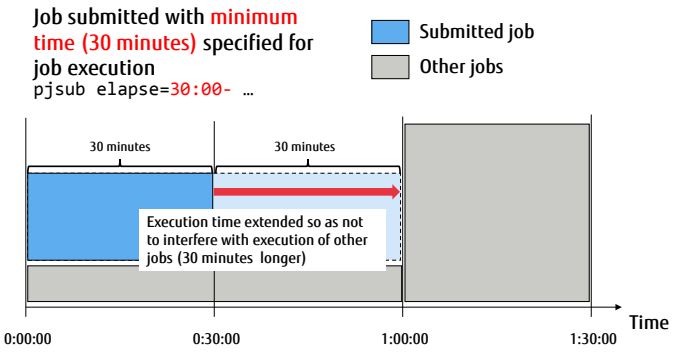

Figure 16 Example of a job with a minimum execution time specified

■ Operating ratio improvement example 1

A job can be completed earlier with the high operating ratio function. The example below shows jobs executing computational tasks that require a total of 3 hours.

(a) Submitting 1 job that takes three hours to execute
(b) Submitting 3 jobs, each taking 1 hour to execute
(c) Consecutively submitting jobs, each with a specified minimum execution time of 1 hour

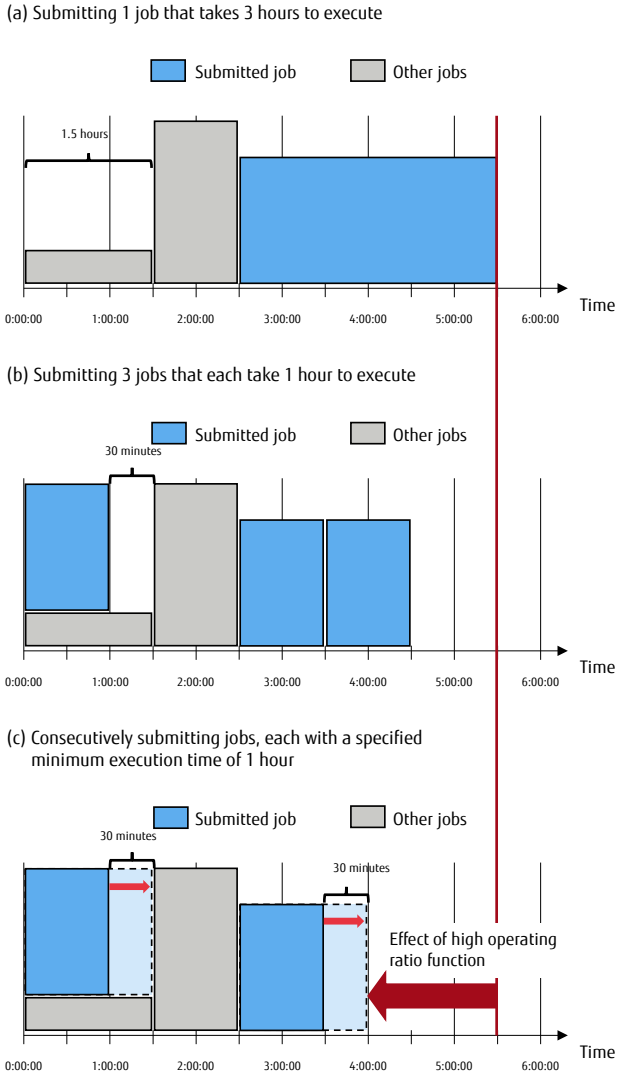

Figure 17 Operating ratio improvement example 1

Figure 17(a) requires an available compute node for 3 continuous hours. The job takes 5 hours and 30 minutes to finish since the start of the job is delayed due to the unavailability of nodes for an amount of time.
In Figure 17(b), one job can start early but the start of the other two jobs is delayed, so the jobs take 4 hours and 30 minutes to finish.
In Figure 17(c), there is no gap between jobs since the jobs can continue running until the start of execution of the subsequent jobs. For this reason, the time taken to complete the series of computational tasks is 4 hours, which is the shortest among the three methods.

■ Operating ratio improvement example 2

The compute nodes in the entire system are expected to have a higher usage rate when many jobs are submitted with minimum job execution times specified.
A usage rate trend analysis on the K computer revealed that a job with a long execution time specified will affect the scheduling of a large-scale job.

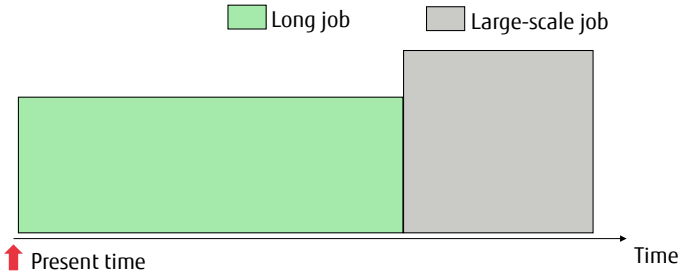

Figure 18 Large-scale job is waiting

In this case, since some compute nodes would be available until the large-scale job begins, the backfill function starts a low-priority job earlier. However, the long job that began before the start of the low-priority job may finish earlier than scheduled, in which case the large-scale job cannot begin until the end of the low-priority job. Consequently, many compute nodes would be left unused for some time.

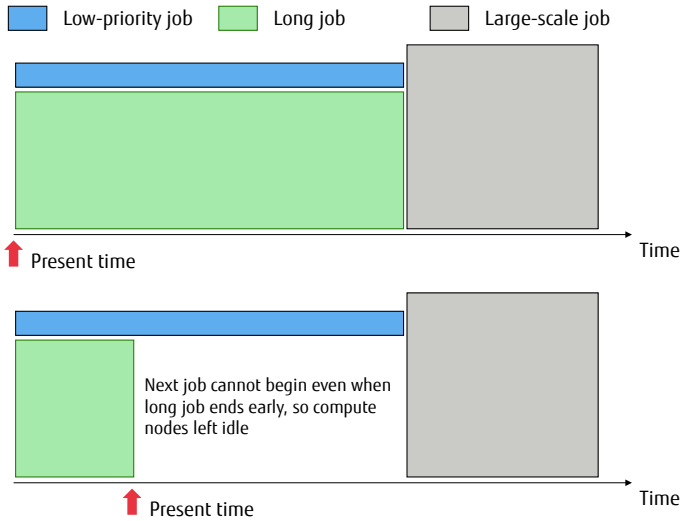

Figure 19 Impact from the early end of a long job

After the backfill function starts execution early for a job that has a minimum execution time specified, execution of that job is suspended at the end of the preceding running job. This allows a large-scale job to start early and avoids leaving many compute nodes unused.

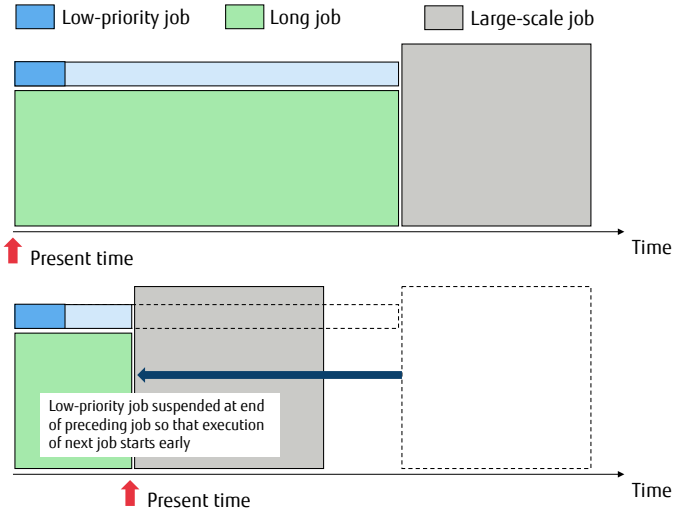

Figure 20 Operating ratio improvement example 2

System Operations Management

Effective management of system operations

System operations management is ever more important as systems grow larger in scale to increase system performance. The latest supercomputer systems, which are the largest-scale systems, tend to consist of thousands to tens of thousands of compute nodes. Due to that size, they need a function facilitating status management and operation control to operate efficiently.
The system operations management function of the PRIMEHPC FX1000 was developed by Fujitsu based on our operations management experience with the K computer, which has over 80,000 compute nodes. This capability to efficiently manage large-scale systems has the following features:

- Flexible and efficient system operations
  - Centralized management of clusters
  - Integrated management from installation to maintenance and routine task monitoring
- High availability and continuous operation
  - Automatic job resubmission upon fault detection
  - Redundancy of important nodes, and automatic node switching
  - Partial maintenance to avoid suspension of a computing center

The features of the system operations management function and how they work are described below.

Flexible and efficient system operations

■ Centralized management of clusters

The PRIMEHPC FX1000, x86 servers, and storage groups with disk devices are considered to be clusters. Operations are managed per cluster.
One cluster may consist of the PRIMEHPC FX1000 and an x86 server, making up a flexible job execution environment enabling end users to submit jobs from one login node to one compute node.

■ Integrated management from installation to maintenance and routine task monitoring

From the management node, the administrator can centrally manage tasks from installation to maintenance. Here are a few examples.

- Software installation using a distributed processing installer
  Generally, installation of the OS and various packages on a large number of nodes is extremely time-consuming. The subsequent management of the installed software is also difficult. Therefore, the PRIMEHPC FX1000 provides an installer specially designed for large-scale installation. This installer enables centralized implementation of the initial settings required as packages and job execution environments.
- Standardized system configuration information and status display
  Users and administrators want to get a wide range of information. The PRIMEHPC FX1000 offers a great variety of information about each node in the system configuration, including hardware and software configuration information. The hardware information includes the number of installed CPUs and amount of installed memory on the node, and the software configuration information includes the assigned IP address and role of the node. There is also node state-related information, such as whether the node power is on and whether hardware has failed or software has a defect.
  The larger the system scale becomes, the wider the range of information that users and administrators want to know. The system configuration information is handled by a variety of commands, depending on the user and use scenario, which have a standardized command display and specification formats to prevent confusion among users. All the various software designed for the PRIMEHPC FX1000 have standardized forms of expression for the system configuration information.

High availability and continuous operation

■ Automatic job resubmission upon fault detection

The PRIMEHPC FX1000 can detect node failures and remove the failed nodes from operation. The PRIMEHPC FX1000 detects the failures in two ways.
The first method is system monitoring by software. Using the hierarchical structure of node groups, the software efficiently collects the status of nodes and services to detect node failures while distributing the monitoring load. The second method is linking with failure notification by hardware. Through the internal PRIMEHPC FX1000 mechanism for notification of node- and interconnect-related hardware failures, node failures can be instantly detected.
The failed nodes detected by those methods are excluded from the operation targets. To continue operation after a node fails, the PRIMEHPC FX1000 terminates the processing of all the jobs running on the node, and the jobs are automatically restarted on available nodes.

■ Redundancy of important nodes, and automatic node switching

The important nodes essential to operations management in the PRIMEHPC FX1000 include the system management node and compute cluster management node. If any of these nodes cannot start due to failure, all or part of system operations are suspended, and operation cannot continue. To prevent that situation and continue operation, the PRIMEHPC FX1000 can configure all of these nodes in a redundant configuration as active and standby systems. The detection of a failure in the active node will result in automatic switching of the system to the standby system to continue operation.

■ Partial maintenance to avoid suspension of a computing center

To perform maintenance on bundles of software, large-scale systems will stop operation longer than other systems if all nodes must be

restarted to update settings. However, some bundles do not necessarily require node restarts, and other bundles can run with the existing version, depending on the repairs in the software. A rolling update implements partial maintenance on these latter bundles without stopping job operations of an entire cluster. During the update, job operations continue on some of the compute nodes in the cluster. rpm in each system software package provides additional package management information per bundle. The information includes the availability of rolling updates and the necessary post-repair system operations (node restart, service restart, what is not required, etc.). Armed with this information, administrators can visualize the work required in their software maintenance and minimize the operation suspension time of computing centers.

Distributed File System

"FEFS" is the name of the distributed file system that provides the high reliability required for supercomputer-level processing power. The FEFS is also stable, since file system stability is directly related to the stability of a supercomputer. The implemented file system also delivers high I/O parallel performance as it is important to minimize the file I/O time of tens of thousands of nodes. Otherwise, the supercomputer cannot make full use of its computing capabilities.
As discussed below, the FEFS achieves both high performance and high reliability.

Cluster-type file system based on Lustre

FEFS stands for Fujitsu Exabyte File System, which is a cluster-type distributed file system. Originally based on the open-source Lustre 2.10, which is the de facto standard for HPC file systems, the FEFS has not just inherited the excellent performance and functions of Lustre, such as high parallelization and scalability, but gone further with enhancements featuring high performance, stability, and usability. It can be widely adapted to massive-scale systems, such as the K computer, and to medium- and small-scale center operations.

Many successes with large-scale systems

Inheriting the technology gained through the development and operation of the K computer, the FEFS proved itself to be a success with the K computer and then achieved further success with Fujitsu's x86 cluster systems and PRIMEHPC FX10/FX100, helping promote the stable operation of the systems.

■ High scalability of over a terabyte per second

The FEFS is a cluster-type file system that can scale out total throughput in proportion to the number of object storage servers used. To obtain the required throughput, it is necessary to prepare the number of units appropriate to the required performance.

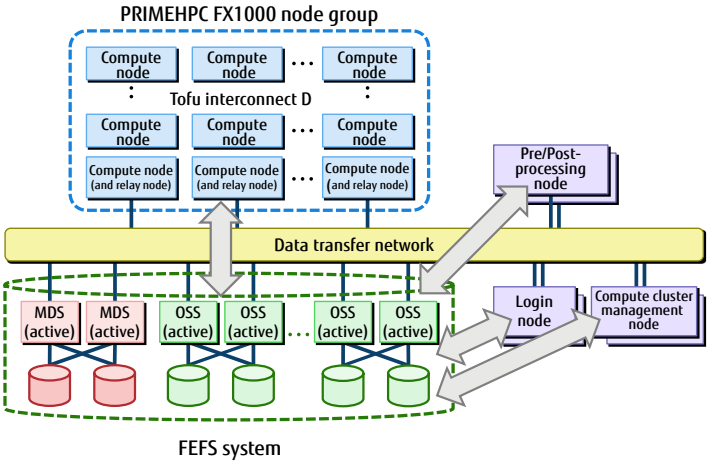

Figure 21 PRIMEHPC FX1000+FEFS system configuration

The PRIMEHPC FX1000 has up to 8 compute nodes that are also relay nodes per rack, and it has a mounted InfiniBand adapter for access to the FEFS file servers. The number of nodes equipped with InfiniBand can be selected flexibly according to performance requirements, and throughput performance can be scaled out in proportion to the number of nodes.

■ Elimination of interference on parallel applications

An important factor to getting the best super-parallel MPI application performance with tens of thousands of nodes is to eliminate interference from the system daemons. The start of a system daemon delays the synchronization process between parallel processes and extends the application runtime. The FEFS has completely eliminated file system daemon processes that run periodically, reducing the impact on the application runtime.
The PRIMEHPC FX1000 has 48 compute cores and 2 or 4 assistant cores for the OS. The system daemons for I/O processing run on the assistant cores. By making applications exclusively use the compute cores, the system has successfully eliminated interference to job execution.

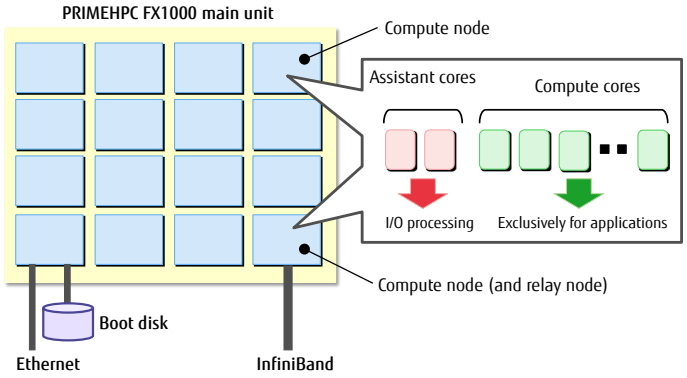

Figure 22 I/O processing and compute processing separated utilizing assistant cores

■ Load distribution by multi-MDS in metadata access

Metadata access tends to be a major bottleneck to system performance. A file system consisting of multiple pairs of MDS and MDT can distribute the metadata access load. By increasing the number of multi-MDS pairs, the file system increases the number of managed files.

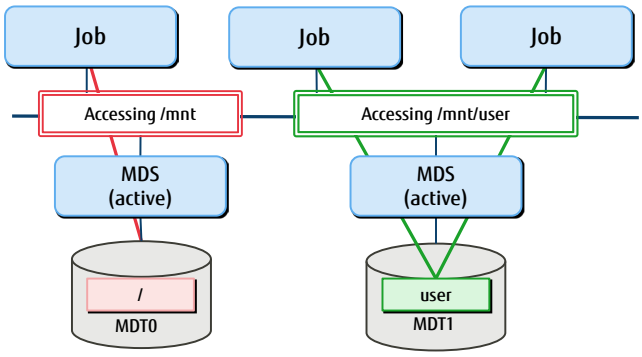

Figure 23 Metadata management by subdirectory

Reliability of continuous operation when hardware fails

High reliability as well as performance is essential to stable operation of a large-scale system. In a cluster-type file system consisting of many file servers, storage devices, and network devices, system operation is required to continue even when part of the system fails or stops. So it needs to be fault tolerant. The FEFS improves fault tolerance through hardware redundancy. Also, by using node monitoring and automatic switching in link with system management software, the file system can continue in service even during maintenance or when a single point of failure occurs.

■ Fault tolerance

The ability to automatically detect a failure and bypass the fault location to continue file system services is a critical feature for a large-scale file system consisting of over hundreds of file servers and storage devices.

The FEFS can provide continuous service as a file system by duplicating hardware components and using software-controlled switching of servers and I/O communication paths, even if a single point of failure occurs.

In the PRIMEHPC FX1000, if communication is disabled by an InfiniBand error on a compute node for relaying input/output for the shared file system (FEFS) in the FX server, the communication path is automatically switched to use InfiniBand on another of these relay nodes. The result is continued access to file servers and improved fault tolerance.

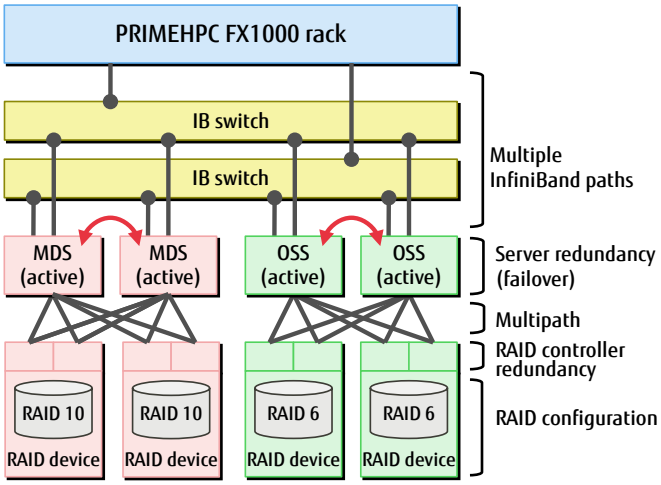

Figure 24 Fault tolerance

■ Hierarchical node monitoring and automatic switching

Large-scale systems require a scheme that can detect failures and automatically notify the affected nodes.

One scheme used so far is node state monitoring based on the monitoring of packet exchanges between compute nodes and file servers. However, one problem with this scheme is the very high number of generated monitoring packets. The number is exponentially proportional to the system scale. The resulting heavy packet transmissions hamper MPI communication between compute nodes and data communication between compute nodes and file servers.

Working together with system management software, the FEFS minimizes communication loads through hierarchical node monitoring and control of switching between nodes. The FEFS monitors the nodes represented in a multi-tier tree with the following hierarchy: nodes inside the PRIMEHPC FX1000 rack, node groups each consisting of multiple racks, and higher-order node groups other than the preceding node groups.

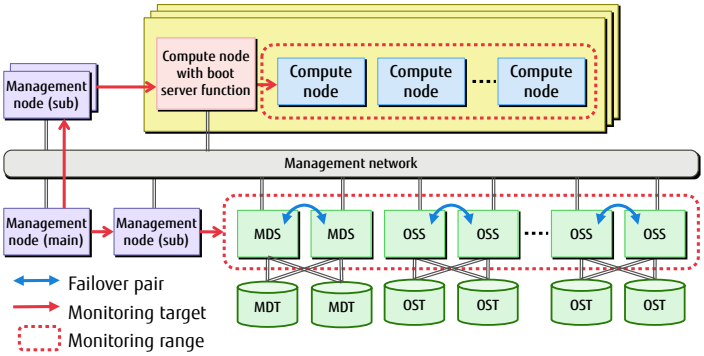

Figure 25 Node monitoring and automatic switching

Functions for improved operability

A large-scale system is used by many users, so the system must ensure that a tremendous amount of file I/O activity by any particular user does not affect other users. It must also ensure that file access by jobs on compute nodes does not affect responses to users on login nodes. The FEFS overcomes these challenges with the fair share QoS function for every user and the response-guaranteed QoS function for login nodes.

■ Fair share QoS function for every user

If many users are using a login node at the same time, a large amount of file I/O by any of the users may drastically slow down the file I/O of the other users.

In the FEFS, to prevent massive I/O requests from a single user and I/O bandwidth occupation, the client side can limit the number of I/O requests that a user can issue.

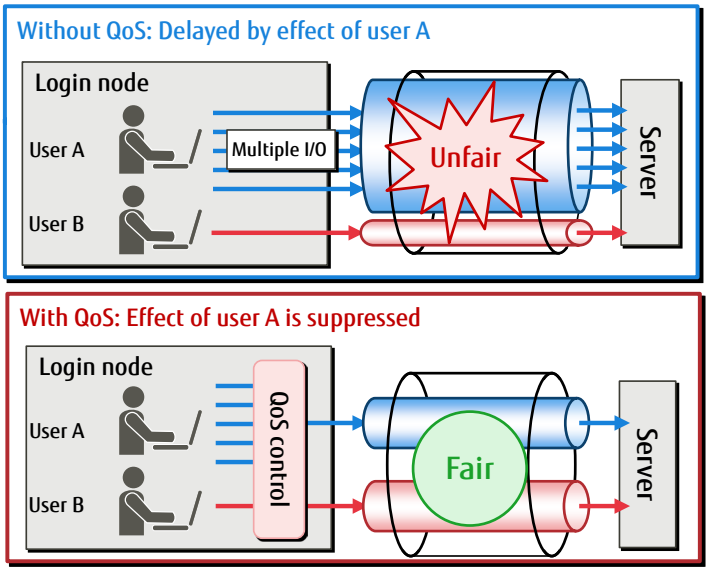

Figure 26 Fair access between multiple users using QoS

■ Response-guaranteed QoS function for login nodes

Access response to users is directly linked with usability, so it is more important than the access response to jobs.

To ensure access response to the users on a login node, the FEFS has a function for allocating server threads that process I/O requests by node group. Thus, even during file I/O by jobs on a compute node, the system can still respond to a user who is accessing the file from the login node.

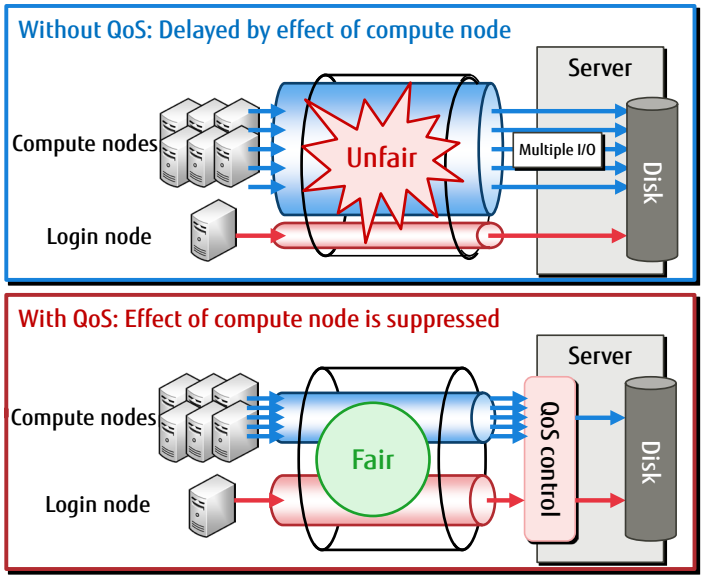

Figure 27 Response guarantee for the login node

Contribution to the Lustre community

An extensive open-source community has helped develop and support Lustre. As a member of the community, Fujitsu contributes to the progress of Lustre.
Through the development of supercomputers, Fujitsu has solved problems as they occurred and provided feedback to the community, leading to higher Lustre performance and consistent quality. These efforts are also aimed at advancing distributed file systems for HPC.

Power Management

Power-saving initiatives

As the scale of supercomputer systems has grown from tens of thousands to hundreds of thousands of compute nodes, power consumption has also increased. Supercomputers are consuming increasing amounts of power amid the rising interest of society as a whole to save energy. So the following challenges need to be addressed.

- Due to system expansion, power facilities cannot supply enough power.
- System power consumption has to be restricted based on the power-saving plans of the installed facilities.

To address the above issues, the PRIMEHPC FX1000 provides a power knob that controls the CPU frequency and memory access to suppress power consumption. However, HPC has certain job execution performance requirements. To manage power and keep job execution performance at HPC levels at the same time, studies on power management in HPC on system operation and job control are actively being conducted.
Aiming to save power without affecting job execution performance, Fujitsu provides the features described below. Figure 28 shows the system administrator reducing power consumption on the operation side and end users reducing power consumption while considering the execution performance of their jobs.

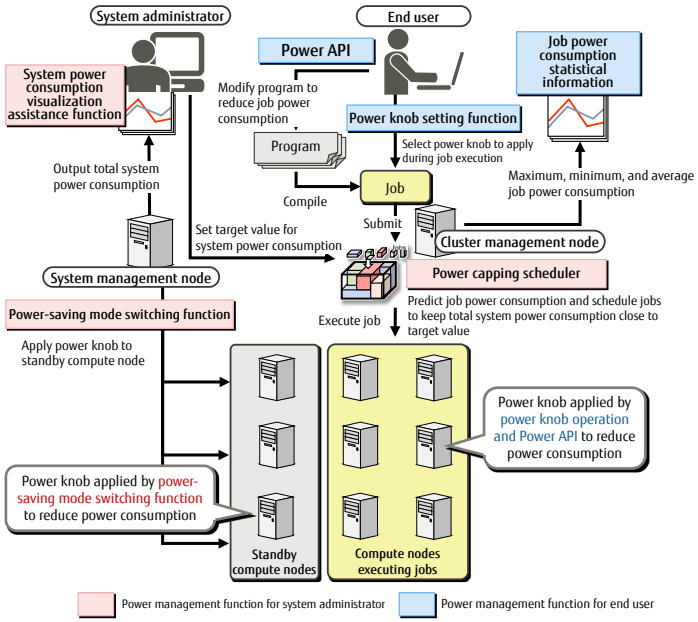

Figure 28 Functional configuration of power management

Power management functions for the system administrator

■ Power-saving mode switching function

Compute nodes consume power even while on standby and not executing any jobs. Standby power is not related to job execution, so it is considered as wasted power consumption and must be reduced.
The power-saving mode switching function automatically switches the CPUs and memory of standby compute nodes to power-saving mode to minimize standby power.
The PRIMEHPC FX1000 offers power knobs for the standby compute nodes not executing any jobs. The power knobs are automatically applied to eliminate wasted standby power consumption. Figure 29 shows the situation of the power-saving mode switching function applying a power knob and the resulting change in power consumption.

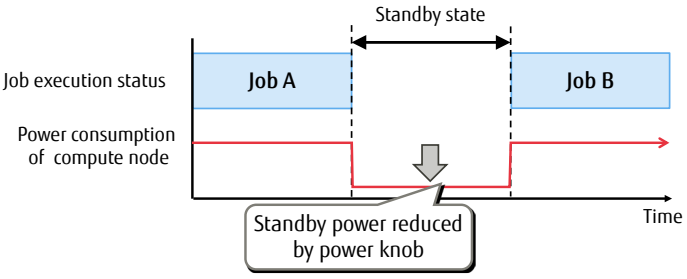

Figure 29 Power consumption transitions due to the power-saving mode switching function

As shown in Figure 29, this function applies the power knob at the same time that the execution of Job A ends, thereby reducing power consumption. Immediately before the execution of Job B begins, the function stops the application of the power knob. Job B can then be executed without being affected by the power knob.

■ Power capping scheduler

Power facilities set power-saving plans with targets for reducing total system power consumption. Once a target value is set for total system power consumption, job scheduling can produce results close to the set target if the power consumption of jobs can be predicted.
The power capping scheduler predicts the power consumption of newly submitted jobs based on the power consumed by jobs executed in the past. The scheduler schedules each job according to the predicted power consumption of the job.
In order to evaluate power management with the power capping scheduler, Kyushu University provided Fujitsu with data on job operations under an actual supercomputer environment. The data was used to simulate transitions in total system power consumption. Figure 30 shows transitions in total system power consumption over 1,000,000 seconds (approximately 11 days) in operations with and without the power capping scheduler. Table 8 shows the average, maximum, and minimum total system power consumption during the simulation period.

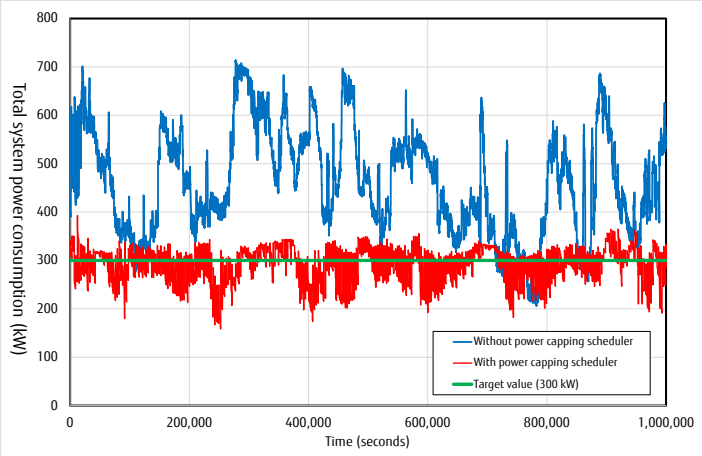

Figure 30 Simulation with the power capping scheduler operating

Table 8 Comparison of total system power consumption

	With power capping scheduler	Without power capping scheduler
Average total system power consumption	283 kW	462 kW
Maximum total system power consumption	393 kW	714 kW
Minimum total system power consumption	159 kW	206 kW

As shown in Figure 30, total system power consumption changes significantly to a maximum of 714 kW when the power capping scheduler is not operating. On the other hand, when the power capping scheduler is operating, total system power consumption does not change significantly but remains around 300 kW, which is the target value for total system power consumption.
Figure 30 also shows that job scheduling by the power capping scheduler keeps the average value in close proximity to the target since the value without the power capping scheduler is 462 kW whereas it stands at 283 kW with the power capping scheduler.

■ System power consumption visualization assistance function

This function monitors power management with the power-saving mode switching function and the power capping scheduler. Using the provided commands and API of this function, the administrator can view the total system power consumption.

Power management functions for end users

The PRIMEHPC FX1000 provides a wide range of power knobs to, for example, control the CPU frequency, restrict memory access, and limit the number of commands issued. From the available choices, end users can select the appropriate power knob for the processing of their own jobs. End users have two methods for applying a power knob: one method is to specify the power knob when submitting a job, and the other method is to specify it within a program.

■ Power knob setting function

When submitting jobs, end users can specify the power knobs they want to apply. The specified power knobs are applied during job execution.

An example of an executed command is shown below.

```
$ pjsub -L freq=1800 -L node=10 run
```

In this example, pjsub is the command for submitting the job. In addition, *freq* instructs the power knob to change the CPU frequency, *node* is the number of compute nodes to execute the job, and *run* is the name of the executed job. The example sets the CPU frequency to 1.8 GHz (1,800 MHz) for the compute nodes (10 nodes) executing the job *run*.
The following figures show the case of the above command executing the job *run*. The job repeats compute processing and I/O processing at a regular interval. Figure 31 shows transitions of processing within the job, and Figure 32 shows transitions in job power consumption.

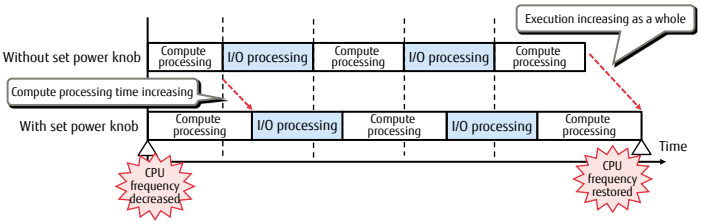

Figure 31 Internal job processing transitions depending on whether a power knob is set

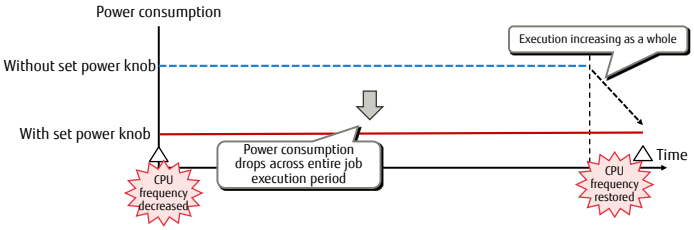

Figure 32 Power consumption transition depending on whether a power knob is set

With the power knob setting function, the specified power knob remains applied from immediately before job execution begins until immediately after job execution ends. If the CPU frequency is 2 GHz normally, the above job command will lower the CPU frequency to 1.8 GHz for the whole job execution period. As shown in Figure 31, the lower frequency does not affect I/O processing, which does not use the CPU much, whereas the processing time of compute processing increases due to the lower frequency and consequently increases the total job execution time. However, as shown in Figure 32, power consumption by the job remains constantly lower due to the lower frequency. Even though this function may increase the total job execution time, end users can easily reduce job power consumption by specifying a power knob in a command option when submitting their jobs.

■ Power API

The PRIMEHPC FX1000 provides a library that enables application of any of the available power knobs at any time. With this library, end users can specify the power knobs they want to apply from the source code of their own jobs. The API specifications of the library are based on the library interface (Sandia Power API) suggested by Sandia National Laboratories.
Using an API provided by the Power API, end users can apply any power knob at any time from the source code of a job.

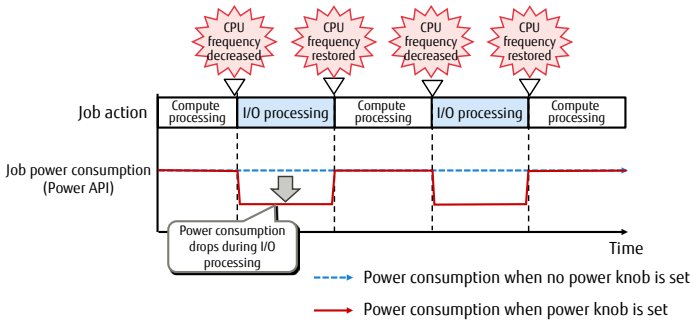

Figure 33 Job power consumption transitions due to the Power API

Figure 33 shows transitions in power consumption with a program modified to lower the CPU frequency only during I/O processing by the Power API in a job that repeats compute processing and I/O processing at a regular interval. In the situation shown in Figure 33, the CPU frequency is assumed to be 2 GHz normally, and I/O processing does not use the CPU much. An end user can lower the CPU frequency during only I/O processing, reducing power consumption in this processing period without increasing the total job execution time. Even though the end user will need to modify the program to specify the appropriate power knob for the correct time by using the Power API, power consumption can be reduced without affecting the total job execution time.

- Job power consumption statistical information

This function enables end users to view statistical information such as the average, maximum, and minimum power consumption of a job. So users can check the status of job power consumption in operations with the power knob setting function and the Power API.

Reference

For more information about the PRIMEHPC FX1000, contact our sales personnel or visit the following website:

https://www.fujitsu.com/global/products/computing/servers/supercomputer/index.html

Advanced Software for the FUJITSU Supercomputer PRIMEHPC FX1000
Fujitsu Limited
First Edition November 12, 2019
2019-11-12-EN