# White paper Advanced Software for the Supercomputer PRIMEHPC FX10

Next Generation Technical Computing Unit
Fujitsu Limited

PRIMEHPC FX10

Contents

# System Software Overview

## System software structure

Fujitsu developed the first Japanese supercomputer in 1977, and later released the VPP series of distributed-memory vector supercomputers, which offered parallel processing systems. Since then, Fujitsu has continued its efforts to develop and provide system software to perform high-speed scientific simulations efficiently and improve system availability. This software includes parallel program development environments, parallel processing libraries, job operations management functions, and a distributed file system.
The supercomputer PRIMEHPC FX10, which is built from accumulated experience and technologies in conventional computer systems, provides the "Technical Computing Suite" system software, which achieves a high system scalability of up to tens of thousands of nodes. High-speed, large-scale scientific simulations require such high scalability.
The following figure shows the structure of the system software.

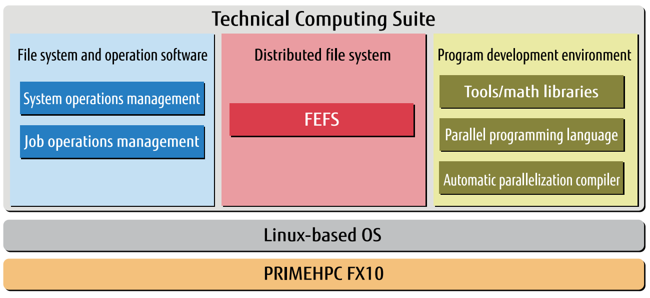

Technical Computing Suite
File system and operation software
System operations management
Job operations management
Distributed file system
FEFS
Program development environment
Tools/math libraries
Parallel programming language
Automatic parallelization compiler
Linux-based OS
PRIMEHPC FX10

**Figure 1 System software structure**

- **Operating system (OS)**
  The operating system for the FX10 is Linux. So many applications can be ported from existing HPC systems. The operating system has been enhanced to maximize the hardware performance of the PRIMEHPC FX10.
- **System operations management, job operations management, and distributed file system**
  The file system and system software enable a large system that includes tens of thousands of computing nodes to be used efficiently.
  - System operations management: This software provides integrated operator interfaces for a large system with hundreds to tens of thousands of compute nodes. Operators are able to control system start and stop with easy operations and also monitor the system status. In addition, the software monitors and isolates faulty components in the system. Thus, the software makes automatic system operation possible and reduces operating costs. It also makes it possible to change the cluster configurations of compute nodes according to operational requests.
  - Job operations management: Not only is it possible to execute a single job that uses tens of thousands of nodes but also a wide variation of jobs can be executed effectively. Large system resources can be shared among many users by setting each user's share or a job priority for the job scheduler.
  - Distributed file system: Many compute nodes can share a large-scale, high-speed file system of one petabyte or more.
- **Program development environment**
  This environment includes automatic parallelization compilers, parallel programming languages, debugging tools, and math libraries. Existing applications can be ported from other existing HPC systems. Furthermore, by leveraging the hardware functions of the PRIMEHPC FX10, the applications can perform high-speed parallel processing.

The above-described system software can easily support the sequences of tasks in application development and scientific simulation execution and analysis at university computing centers and research institutes.

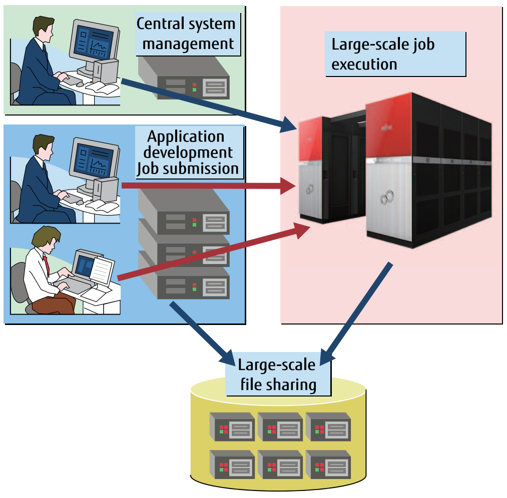

Central system management
Large-scale job execution
Application development Job submission
Large-scale file sharing

**Figure 2 Outline of system software activities**

## Large-scale system implementation issues and system software efforts

For high-speed climate and earthquake simulations, flight analysis, drug analysis, and other scientific computations, the PRIMEHPC FX10 performs parallel processing on compute nodes numbering in the hundreds to hundreds of thousands. Fujitsu has designed and developed system software achieving the following objectives so that many users can use the system quite efficiently:

- Efficient use of large-scale systems
- Easy application development
- Highly parallel simulations in actual practice

- **Efficient use of large-scale systems**
  System administrators and users of the PRIMEHPC FX10 can centrally manage hundreds to hundreds of thousands of compute nodes by using the system management functions and job operations management functions. They can thus use large-scale systems efficiently, as the supercomputer helps to bring various degrees of convenience.

- **High reliability and high availability**
  The system failure rate, including incidents of software faults, increases with hardware scale, so it is especially important in large-scale systems to detect faults early and isolate or recover faulty components.
  The system management software for the PRIMEHPC FX10 can detect hardware and software faults quickly with low overhead by monitoring many nodes hierarchically rather than checking nodes sequentially. It isolates faulty compute nodes from job operations. The job management system automatically re-executes the affected jobs. As a result, users obtain simulation results without awareness of system faults.

- **Sharing by many users**
  With the job management system, users can execute a large-scale simulation on many compute nodes or execute many small-scale simulations handling a wide variety of parameters and input data. The users can share the system fairly since the job scheduler selects the jobs to be executed according to job priority or each user's budget. The job scheduler also assigns compute nodes to jobs so as to utilize maximum system resources.
  On the point of I/O resources, the FEFS (distributed file system with clustered servers) has been extended with a fair share function to prevent particular users from occupying I/O resources.

## Easy application development

Some supercomputers provide a proprietary OS and compilers with limited functionality compared to the standard UNIX functions, so porting (source code modification) and verification of existing applications will be required. In contrast, the PRIMEHPC FX10 provides an OS and compilers that support industry-standard APIs so that many applications on existing UNIX/Linux systems can be recompiled to run on the PRIMEHPC FX10. The OS is Linux-based, and the Fortran, C, and C++ compilers and MPI libraries conform to standard conventions.

## Highly parallel simulations in actual practice

- **Parallel programming model**
  A combination of thread-parallel programs using multiple CPU cores within compute nodes and MPI programs using many compute nodes (hybrid model) can reduce the usage of communication resources to realize high-performance simulations.

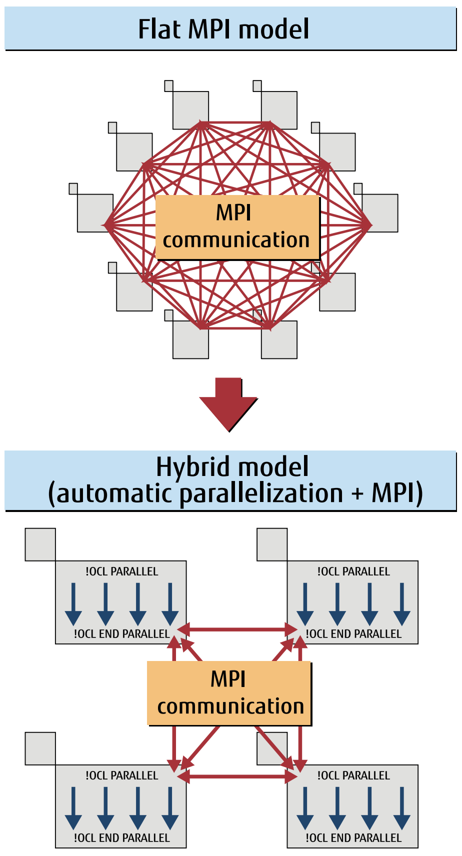


**Figure 3 Flat MPI model and hybrid model**

For parallel simulations that use many CPU cores, it is important to reduce the cost of data communication between the cores. The amount of memory used for communication increases with the square of the number of processes executing the simulation. For example, if the number of processes executing the simulation increases by a factor of 100, the required amount of memory increases by a factor of 10,000. Unlike the flat MPI model where a single process uses 1 core, the hybrid model where a single process uses 16 cores can solve this problem by decreasing the amount of memory required for communication by a factor of 256 (16^2).
The program development environment of the PRIMEHPC FX10 provides compilers that support automatic thread parallelism and OpenMP-based thread parallelism. It also provides MPI libraries that derive the best performance from the Tofu interconnect(*1). So existing applications can be executed easily on the PRIMEHPC FX10 with this high-performance hybrid model, as they are recompiled in this software development environment.

- **Parallel file access**
  High-speed access to input/output data is important to achieving high-speed simulations using many compute nodes. The PRIMEHPC FX10 accomplishes input/output processing of massive amounts of data in a short time by distributing file data to multiple servers and using the MPI-IO parallel access interface.

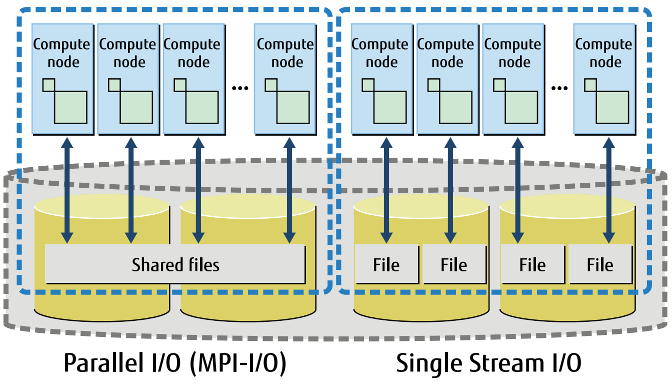


**Figure 4 Parallel I/O and single-stream I/O**

*1 The Tofu interconnect is a high-speed interconnect independently developed by Fujitsu. For details, see *Advanced Technologies of the Supercomputer PRIMEHPC FX10.*

# System Operations Management

## Efforts for system operations management

Recent supercomputer systems, not just our PRIMEHPC FX10, generally tend to consist of between thousands and tens of thousands of compute nodes to improve system performance. For this reason, "system operations management" has become more important than ever before as a function facilitating state management and operation control of such large-scale systems.
Fujitsu has developed system operations management with the following features to handle these large-scale systems.

- **Distribution of hierarchical processing**
  - Distribution of system monitoring and system control loads
  - Distributed processing in installation
- **High-availability system (continuous operation)**
  - Automatic job isolation upon fault detection
  - Redundancy of important nodes and automatic node switching
- **Job execution efficiency improvement**
  - Frequent communication reduced through coordination of the notification process
  - Interruptions eliminated by use of the RDMA (Remote Direct Memory Access) communication function$^{(*1)}$
- **Easy-to-read display (overall view)**
  - Display of summarized contents
  - Standardized system configuration information and state display

The functions implementing the above features of system operations management and the means of implementation are described below.

## Distribution of hierarchical processing

### Distribution of system monitoring and system control loads

System operations management increases processing efficiency by using the following hierarchical structure to distribute monitoring and other processing loads in a large-scale configuration. The entire system is divided into logical units called node groups. Job operations management subnodes handle the part of system operations management processing within the scope of their node group in order to distribute system monitoring and system control loads. This hierarchical structure improves system scalability because any added nodes can be easily managed simply by the addition of a node group.

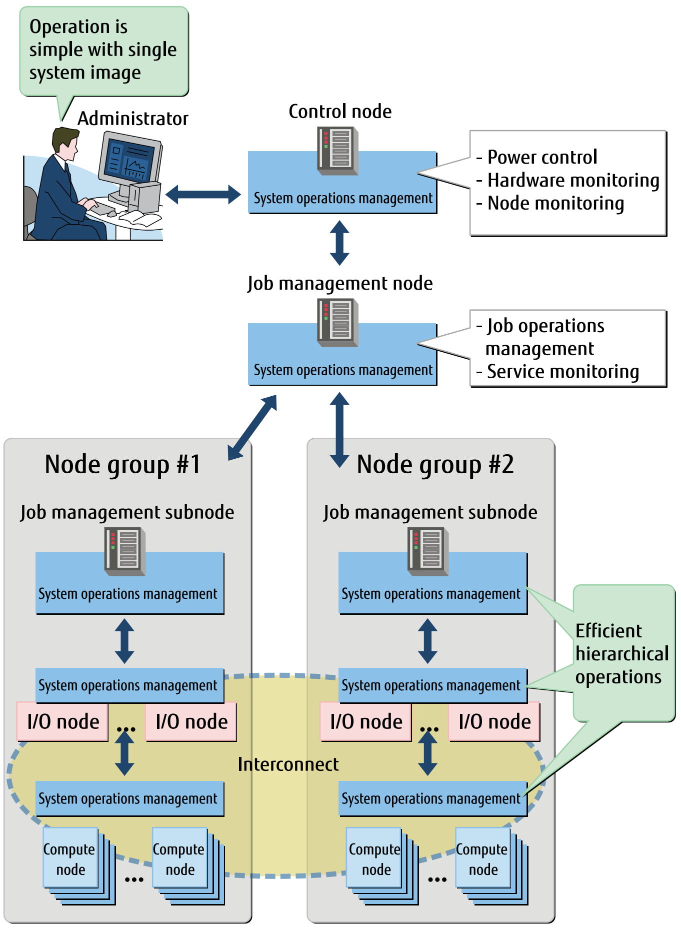

Operation is simple with single system image
Administrator
Control node
System operations management
- Power control
- Hardware monitoring
- Node monitoring
Job management node
System operations management
- Job operations management
- Service monitoring
Node group #1
Job management subnode
System operations management
System operations management
I/O node
...
I/O node
System operations management
Compute node
...
Compute node
Interconnect
Node group #2
Job management subnode
System operations management
System operations management
I/O node
...
I/O node
System operations management
Compute node
...
Compute node
Efficient hierarchical operations

**Figure 1 Outline of system operations management activities**

### Distributed processing in installation

Installation of an OS and different packages on a large number of node groups is extremely time-consuming. Subsequent management of the installed software is also difficult. The PRIMEHPC FX10 provides an installer specially designed for large-scale installation. This installer supports the above hierarchical structure so it can centrally manage the application status of packages and configuration files.
As shown in Figure 2, the control node acts as an installation server and job operations management subnode, and I/O nodes act as intermediate installation servers. The resulting installation structure has three layers to distribute and speed up processing. The installer on the control node centrally manages installation information and the contents of configuration files. While constantly synchronizing the intermediate installation servers, it maintains a unified state within the system. As a result, administrators can install software and manage packages and settings on large-scale nodes without spending much effort and time.

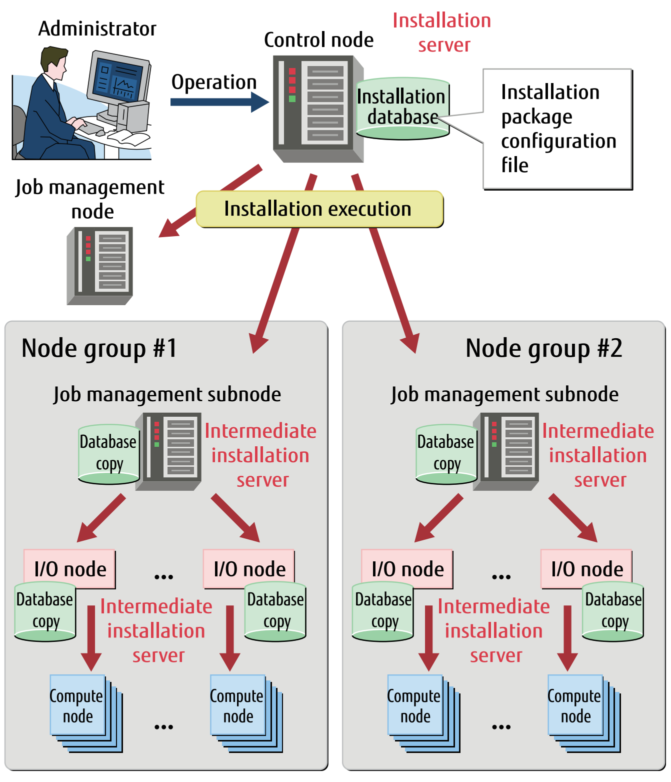


**Figure 2 Installer for large-scale installation**

## High-availability system (continuous operation)

■ **Automatic job isolation upon fault detection**

The PRIMEHPC FX10 can detect faults on nodes by either of two methods. The first method is system monitoring with software. The above hierarchical structure is used to efficiently collect information on the states of nodes and services to detect node failures while distributing loads. The second method is linkage with a hardware-based fault notification function. The PRIMEHPC FX10 is designed to immediately issue notification of any hardware failure concerning a node or interconnect, and node failures are immediately detected from this notification.

After a node failure is detected by one of the above methods, the relevant node is disconnected from operation. At the same time, all job processes running on the node group including this node are terminated and their jobs are automatically re-executed on available nodes so that operation continues.

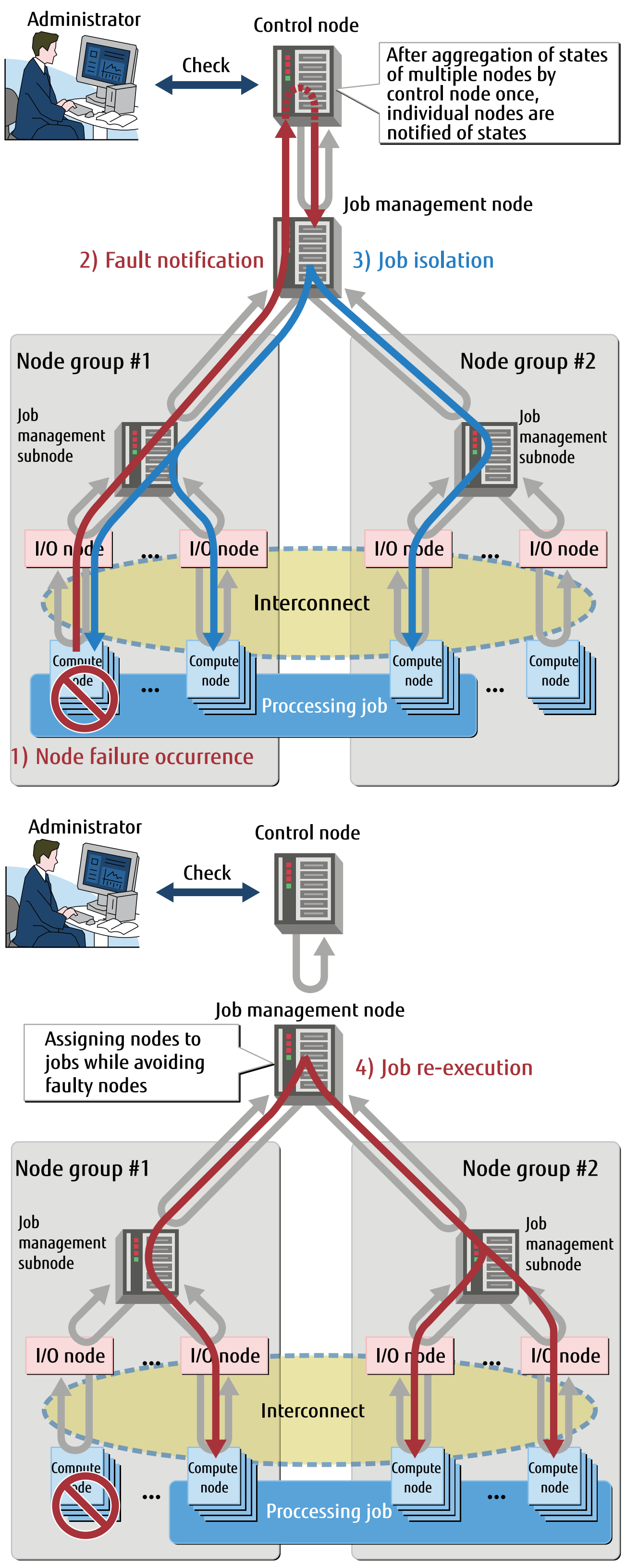


**Figure 3 Automatic job isolation (and re-execution) upon fault detection**

- **Redundancy of important nodes and automatic node switching**
  Control nodes, job operations management nodes, job operations management subnodes, I/O nodes, etc. are important as nodes essential to PRIMEHPC FX10 operations management. If one of these important nodes fails and cannot start, all or part of the system stops operating, resulting in an inability to continue operation. For this situation, the PRIMEHPC FX10 configures all these important nodes redundantly with the nodes paired into active and standby nodes. When a fault is detected on an active node, the node is automatically switched with its standby node so that operation can continue. This makes it possible to prevent operation from being interrupted.

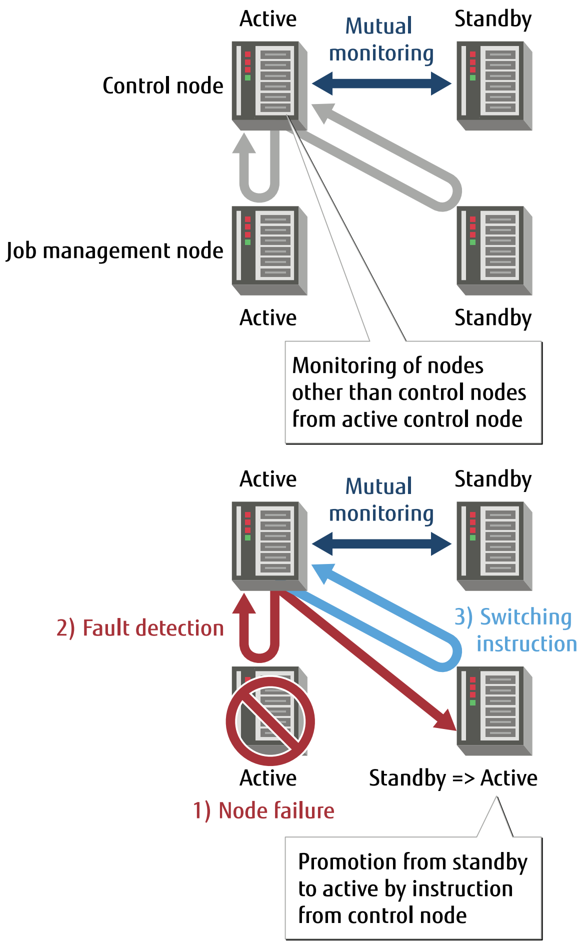


**Figure 4 Redundancy of important nodes**

## Job execution efficiency improvement

- **Frequent communication reduced through coordination of the notification process**
  In a system with a single node taking care of alive monitoring of between thousands and tens of thousands of compute nodes and service state monitoring, an extremely heavy load is placed on this one node. The system management software increases the efficiency of that monitoring by structuring nodes in a hierarchy and using functions unique to the PRIMEHPC FX10.
  Figure 5 shows the monitoring sequence through the hierarchy. Before lower-order compute nodes notify higher-order nodes of faults, a certain number of notifications are combined into a single report, which is then sent to higher-order nodes. Even if the states of many compute nodes suddenly change, this method can prevent frequent transmissions of large amounts of information and ensure efficient notification to higher-order nodes.

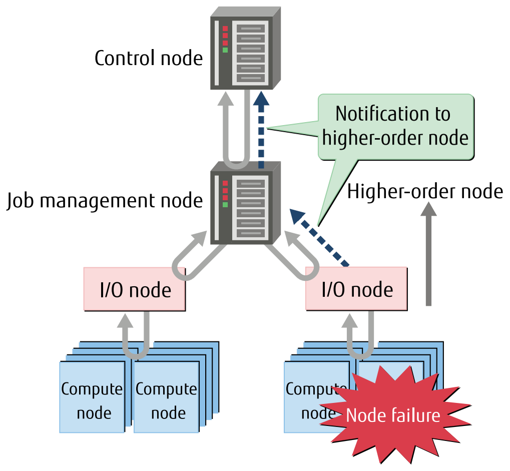


**Figure 5 Hierarchical alive monitoring of compute nodes**

- **Interruptions eliminated by use of the RDMA communication function**
  Alive information for compute nodes is recorded in memory on remote nodes. The RDMA communication function of the interconnect contained in the PRIMEHPC FX10 reads that memory directly for monitoring. As a result, higher-order nodes can regularly monitor compute nodes without affecting the job execution load on compute nodes or committing recognition errors.

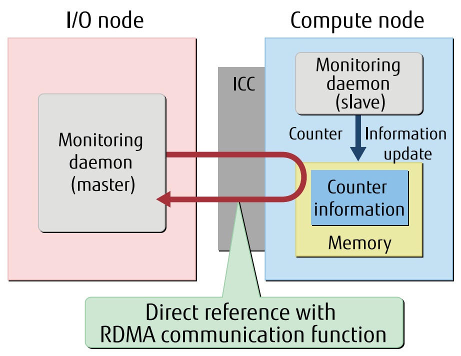

I/O node
Compute node
ICC
Monitoring daemon (slave)
Counter
Information update
Monitoring daemon (master)
Counter information
Memory
Direct reference with RDMA communication function

**Figure 6 Alive monitoring of compute nodes with RDMA communication function**

## Easy-to-read display (overall view)

- **Display of summarized contents**
  If the states of 100,000 target nodes were simply displayed with one node per line on the command-line interface of a terminal, the displayed information would extend over 100,000 lines. Displayed in this way, no overall picture of the system is available so important information could be overlooked, which is a potential problem. For this reason, the default display format of the PRIMEHPC FX10 shows only summary information on the entire system, including the number of nodes in each state category. The options provided to users to get more detailed information include the options for filtering display targets by node group and for displaying information on only the nodes in a particular state. As designed with these options, users can get detailed information in increments from the overall state of the system to the states of individual nodes.

- **Standardized system configuration information and state display**
  Among the variety of information about each node configuring the system, 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 power is on and whether a hardware failure or software defect has occurred.
  As systems become larger in scale, users and system administrators will want to get increasingly diverse information. System configuration information is handled by a variety of commands, depending on the user and use scenario. That may create confusion in users unless command display formats and specification formats are standardized. For this reason, the various software for the PRIMEHPC FX10 has been designed so that system configuration information has standardized forms of expression.

*1 RDMA: Remote Direct Memory Access. The RDMA communication function is provided by the interconnect contained in the PRIMEHPC FX10.

# Job Operations Management

## Efforts to develop job schedulers

We have made continuous efforts to develop and provide batch job execution environments (job schedulers) with each successive supercomputer system up to now. Our conventional job schedulers have the following features.

- **High-speed execution environment that relies on advance reservation of computing resources**
  To get full performance from hardware, each job uses shared computing resources that are reserved in advance by a resource management feature. This feature alleviates performance degradation by preventing jobs from interfering with one another. The execution environment provides unique features not available in competing products, including fine-tuned advance reservation of individual CPUs and high-speed execution with threads/processes attached (bound) to CPU cores in linkage with the language system (runtime).

- **Extended job concept, a job running across multiple nodes**
  The distributed parallel execution environment provided with the PRIMEHPC FX10 treats parallel programs as one job so that multiple compute nodes can execute them reliably in a batch. The typical distributed parallel execution environment involves problems such as the following: the abnormal end of a parallel program leaves processes remaining on some servers; or information, such as program output results and statistical information, is not output in a batch. Our products are free from these problems because job control is implemented at the OS level.

- **Fulfillment of various requirements at joint research centers**
  Joint research centers are often running supercomputer systems, with the operation policy on job execution varying slightly from one center to another. From our dealings with customers, we have implemented functional enhancements by reflecting their operating requirements in our products as necessary.

To develop the job scheduler for the PRIMEHPC FX10, we built upon the above-described features while focusing our efforts on the following features, keeping in mind the challenges of guaranteeing high system scalability for tens of thousands of nodes and taking full advantage of PRIMEHPC FX10 hardware characteristics.

- **Job operations on a very large-scale system**
  The number of jobs handled by the job scheduler increases significantly as the system scale increases. The PRIMEHPC FX10 can implement a very large-scale system containing tens of thousands of compute nodes. The system is expected to handle over one million jobs. Therefore, it is necessary to provide a user interface that can control many jobs reliably and allows end users and system administrators to handle many jobs easily.

- **High-speed job execution on a very large-scale system**
  As the system scale increases, runtime deviations due to I/O contention between jobs are expected to increase. The characteristics of the PRIMEHPC FX10 hardware configuration themselves are the solution to the I/O contention problem.

- **High-performance execution environment that takes full advantage of hardware characteristics**
  The high-speed execution environment provided with the PRIMEHPC FX10 takes full advantage of the PRIMEHPC FX10 hardware performance. This environment is built from know-how gained from conventional products and makes full use of Tofu interconnect characteristics.

- **Simplified operations management**
  An issue involved with conventional products was that the operations management cost spent on system administrators was higher due to complicated operation configurations, despite the fact that the products were equipped with a rich array of functions. We reviewed the settings required for job operations, divided them into settings related to the whole system and settings related to individual jobs, and redesigned them with the aim of achieving simple operation settings. The redesigned settings can be individually defined.

## Support for very large-scale systems

The PRIMEHPC FX10 is configured with tens of thousands of compute nodes, which when performing operations on a large number of jobs should not create stress in users. So the efforts to develop the job scheduler for the PRIMEHPC FX10 were focused on the following.

- **Improved response to job input and other operations**
  We reviewed and redesigned the processes spanning the life cycle of a job from acceptance to end so that each process is multiprocessing-aware and multithreading-aware. The resulting job input performance represented by response time is one-tenth of that of conventional products. As a result, many users can input a large number of jobs and not feel that anything is wrong. In addition, the bulk job function is supported for computation techniques like a parameter study. The function makes it possible to input a large number of jobs that require the same processing and differ only in parameter values.

- **Job scheduling performance improvement**
  If the job scheduler has a high processing overhead, the utilization rate of the entire system stays low because executable jobs cannot be executed immediately. To execute jobs, the job scheduler for the PRIMEHPC FX10 parallelizes processes to select the optimum computing resources from an enormous amount of computing resources with high processing costs. The result is cost reductions.

- **System utilization rate improvement**
  The mix of jobs executed at joint research centers include different scales of jobs, from jobs requiring only one node to very large-scale parallel jobs that sometimes require tens of thousands of nodes. In this environment, the utilization rate of the entire system may drop unless computing resources are efficiently scheduled. The job scheduler of the PRIMEHPC FX10 supports backfill scheduling as a solution to this problem. Before the expected execution start time of a large-scale job, backfill scheduling uses computing resources to first execute small-scale jobs that are supposed to end before the start time. The system utilization rate can thus be improved.

- **Design of display interfaces**
  Since the system handles an enormous number of jobs, the display of job states must be designed so that users can easily understand the current status. The basic display of job states is a summary display. Users select to display detailed information only as needed, which simplifies the display of information.

## Response to I/O contention (file staging function)

In an environment where a large number of jobs run concurrently, I/O contention between jobs is expected to increase runtime deviations. The supported file staging function deals with this situation by transferring the various files required for job execution to compute nodes before the nodes begin to execute the jobs. After the end of job execution, it retrieves the execution result files.

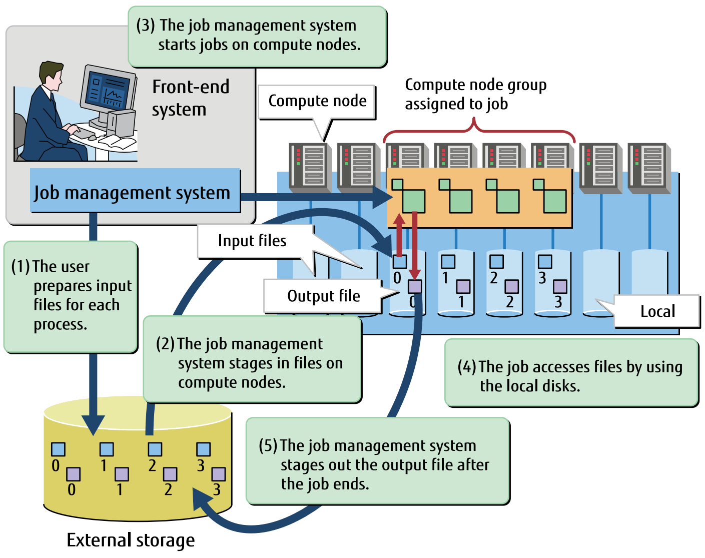

(3) The job management system starts jobs on compute nodes.
Front-end system
Compute node
Compute node group assigned to job
Job management system
Input files
Output file
Local
(1) The user prepares input files for each process.
(2) The job management system stages in files on compute nodes.
(4) The job accesses files by using the local disks.
(5) The job management system stages out the output file after the job ends.
0
1
2
3
External storage

**Figure 1 File staging function**

A general file staging function transfers files as part of the jobs. This approach may waste computing resources as the computing resources required for job execution remain reserved even during file transfer. The file staging function of the PRIMEHPC FX10 prevents this waste of computing resources by transferring files separately from jobs. One of the characteristics of the PRIMEHPC FX10 system configuration is that it has I/O nodes independent of compute nodes. Utilizing this characteristic, the file staging function processes file transfers on I/O nodes asynchronously with job execution. This eliminates job execution wait time due to file transfer, which in turn prevents the utilization rate of the entire system from dropping.

## Provision of a high-performance execution environment

- **Built from conventional technologies**
  The high-performance job execution environment provided with the PRIMEHPC FX10 is built from conventional technologies, such as those for advance assignment of computing resources to jobs and binding of job processes and threads to cores.

- **Effective use of the PRIMEHPC FC10 interconnect**
  Making full use of the characteristics of the PRIMEHPC FX10 Tofu interconnect, the scheme for advance assignment of the computing nodes to be executing jobs does the following.

  - Assign a group of nodes in advance for guaranteed performance in communication with adjacent nodes and prevention of interruptions caused by communication initiated by other jobs.
  - Control unassigned compute node groups (free nodes) so that they are always adjacent (contiguous) to each other in order to facilitate advance assignment of adjacent node groups.

## Flexible adaptation to operation policies

Joint research centers have their own operation policies, so our systems must be able to flexibly adapt to the various operation policies differing among the centers.
The PRIMEHPC FX10 assumes the following two categories of operation policies at joint research centers:

- Restrictions (upper and lower limits of the quantity of resources required for job execution, default value assumed when a specification is omitted, etc.) that apply to a job when the job is executed
- Policies on overall job operations, such as which jobs have priority for execution

The job scheduler for the PRIMEHPC FX10 uses a function called "job ACL" for the restrictions described above in the first category. The job ACL can define each restriction applying to individual jobs. The job scheduler uses a function called "scheduling policy" for the operation policies described above in the second category. The scheduling policy can define operation policies affecting the whole system. In conventional products, there were a variety of parameters for these settings. In the PRIMEHPC FX10, however, the settings are aggregated into these two functions to reduce the management workload on operations administrators.

Operations administrators also need to monitor and improve various operation parameters to set suitable values in response to the current utilization status. As part of efforts to optimize system operation, we are also studying tools that accept the job scheduler activity logs as input and simulate the operation status subsequent to operation parameter changes.

# Distributed File System

## Efforts to improve file data processing

The performance of supercomputers is increasing at a remarkable pace, with computing performance reaching several dozen petaflops in 2011. Associated with the increases in the total numbers of compute nodes and cores as well as in the installed memory amount, the capacity and total throughput performance of file systems have increased, so that in the near future, file systems are expected to have a capacity in the order of 100 petabytes and performance in the order of 1 terabyte per second. Both the capacity and performance have dramatically increased nearly tenfold in one year. To keep up, the major file systems are changing from single-server file systems to clustered file systems.
With the PRIMEHPC FX10, Fujitsu successfully provides the best computing performance in the world. To achieve the world's best data processing performance in the field of file systems, Fujitsu developed the FEFS, a clustered distributed file system, aiming at the following objectives:

- World's fastest I/O performance and high-performance MPI IO
- Job runtime stabilized by interruption prevention
- World's largest file system capacity
- Improvement with added hardware in scalable performance and capacity
- High reliability (service continuity and data integrity)
- Ease of use (sharing by many users)
- Fair share (fair use by many users)

Before we could achieve these objectives, we had to overcome many challenges, including those relating to performance. Of those challenges, those that are particularly critical to large-scale systems are listed in the following table. The table also includes an outline of measures implemented in the FEFS.

Table 1 File system challenges and measures

| Category | Challenge | Outline of FEFS measure |
|---|---|---|
| Interruption prevention (performance) | Job runtime variations due to I/O contention between users or jobs | 2-layer hierarchy that prevents jobs from being interrupted |
| | Job processing delays due to I/O contention between jobs or compute nodes | Communication path and disk splitting to eliminate I/O contention |
| High reliability | Services stopped by single point of failure | Continuous operation ensured by hardware duplication and failover |
| | Monitoring of tremendous number of nodes | Hierarchical node monitoring and automatic switching |
| Ease of use | Lots of I/O by single user causes response time deterioration for other users | Restriction on number of I/O requests issued by single user |
| | User response time deterioration due to job I/O | Guarantee of response at login nodes |

The FEFS measures taken to address the above-described challenges are discussed below.

## Two-layer file system that prevents interruptions

To ensure stable job execution, the FEFS can operate with a two-layer file system model consisting of a local file system and a global file system. The local file system is used as a high-speed temporary area dedicated to jobs. The global file system is used as a large-capacity shared storage area for user files.

### Local file system

As a high-speed temporary area dedicated to jobs, the local file system is intended to maximize the file I/O performance of applications executed as batch jobs. Using the staging function described below, the FEFS transfers input/output files between the local and global file systems. The local file system temporarily keeps the files for jobs that are running or waiting to be executed.

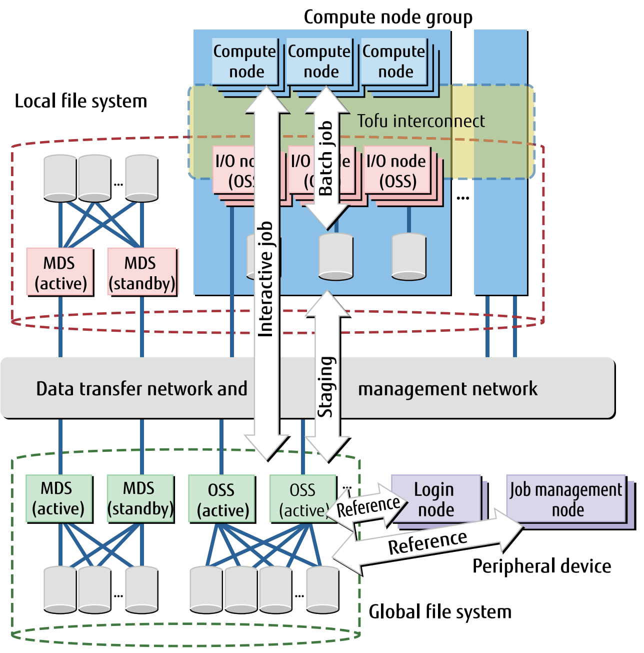


**Figure 1 Two-layer file system model**

File servers that access data blocks are I/O-dedicated nodes (I/O nodes) inside the PRIMEHPC FX10 rack. I/O nodes are connected with compute nodes via the Tofu interconnect to achieve low-latency, high-throughput file data transfer using RDMA communication function.

### Global file system

The global file system is located outside the PRIMEHPC FX10 and serves as a large-capacity shared area that stores job input/output data and other user files. When using an interactive job or performing another operation in addition to accessing a file from the login node, a user can perform debugging or tuning while checking job output by directly accessing the file from a compute node.

The PRIMEHPC FX10 is connected with the file servers of the global file system via the QDR InfiniBand installed on I/O nodes. The I/O nodes relay file data transfers between the Tofu interconnect and InfiniBand, whereas the compute nodes access file servers via I/O nodes.

- **File staging**
  The system automatically passes files between the local and global file systems by using the file staging function.
  The file staging function works in linkage with job operation software. Before a job starts, the file staging function transfers (stages in) the input files from the global file system to the local file system. After the job ends, the function transfers (stages out) the output files from the local file system to the global file system. The user can specify the stage-in and stage-out files in a job script.

## Communication path and disk splitting to eliminate I/O contention

The FEFS, which is a clustered file system, scales out parallel throughput performance by bundling many file servers into groups. Concentrated access on a specific file server causes communication congestion and/or disk access conflict, which degrades I/O performance and results in job runtime variations among compute nodes. The complete elimination of file I/O contention is critical to stable job execution.
The FEFS eliminates file I/O contention by splitting file I/O at two levels: job level, and level of compute nodes per job. At the job level, I/O nodes are grouped so that each I/O node group stores files for one job, thereby preventing file I/O contention that would otherwise occur on the servers, networks, or disks in the group.
At the level of compute nodes per job, file data is exchanged via I/O nodes with the minimum number of hops to minimize file I/O contention between compute nodes.

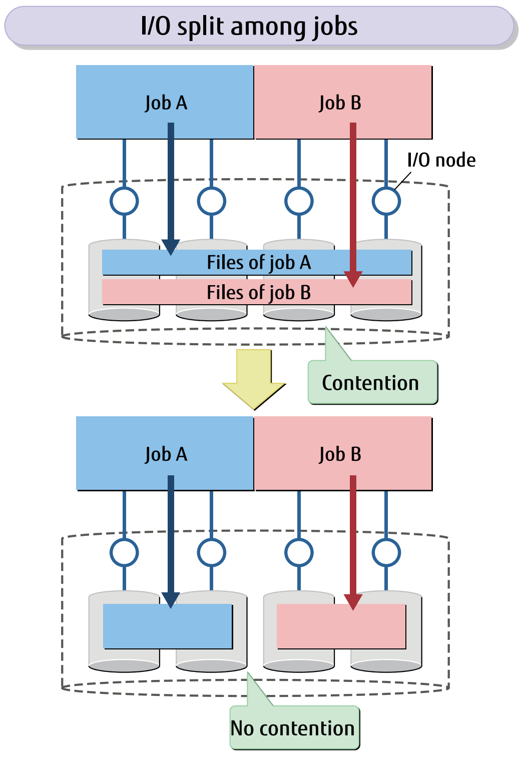


**Figure 2 I/O split among jobs and among nodes (1/2)**

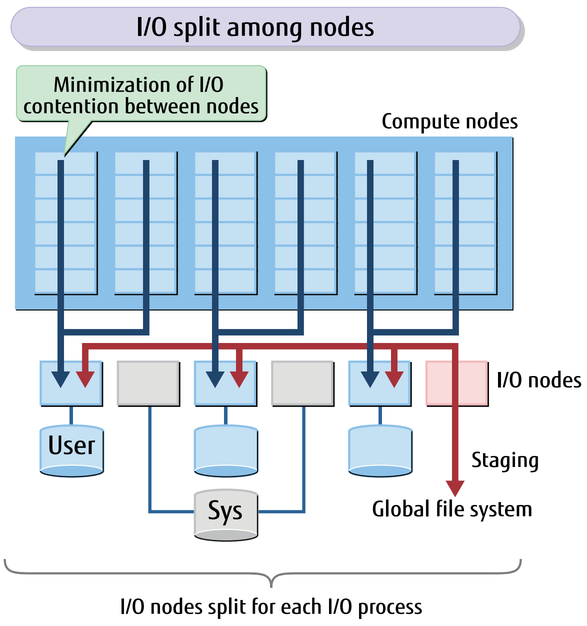

I/O split among nodes
Minimization of I/O contention between nodes
Compute nodes
I/O nodes
User
Sys
Staging
Global file system
I/O nodes split for each I/O process

**Figure 2 I/O split among jobs and among nodes (2/2)**

## Continuous operation ensured by hardware duplication and failover

Reliability is as important as performance in large-scale systems. A clustered file system consists of many file servers, storage devices, and network devices. Even if one or several of them become faulty, go down, or are under maintenance, the services of the file system must continue in order to ensure that operation of the whole system is not interrupted.
The FEFS improves fault tolerance by duplicating hardware components and controlling node monitoring and switching with software so that services can continue even if a single point of failure occurs.

- **Fault tolerance**
  A large-scale file system consisting of more than hundreds of file servers and storage devices is always likely to encounter a network adapter, server, or other hardware fault that forces a wait for maintenance. To ensure that the entire system continues operating even in this condition, it is important that the file system automatically detect faults and continue services by bypassing the fault location.
  The FEFS can provide continuous services 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.

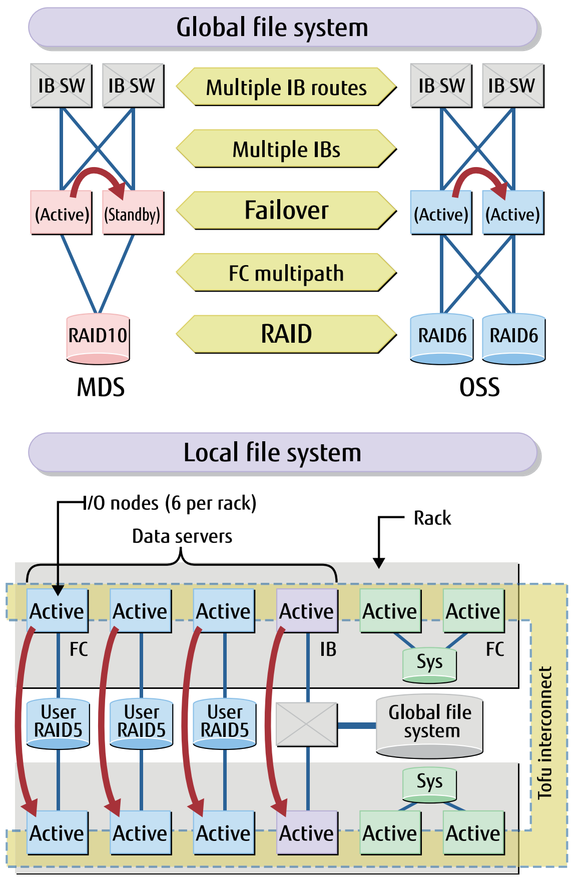


Figure 3 Fault tolerance through hardware duplication

## Hierarchical node monitoring and automatic switching

Large-scale systems require a scheme that can detect failures and automatically notify the affected nodes of replacement without human intervention. One such scheme used so far is node state monitoring based on monitoring packet exchange between compute nodes and file servers. However, one problem involved with this scheme is that the number of generated monitoring packets is very high. The number is exponentially proportional to the system scale. Such heavy packet transmissions hamper MPI communication between compute nodes and data communication between a compute node and file server.
The FEFS minimizes communication loads through hierarchical node monitoring and control of switching between nodes in linkage with system management software. The FEFS monitors the nodes represented in a tree consisting of multiple levels in the following hierarchy: nodes inside the PRIMEHPC FX10 rack, node groups each consisting of multiple racks, and higher-order node groups other than the preceding node groups.

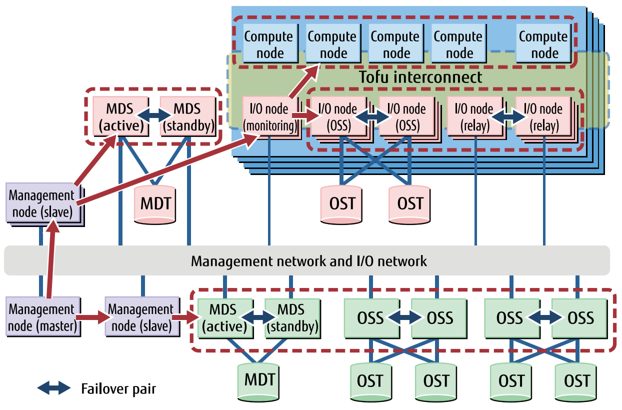


Figure 4 Hierarchical node monitoring and automatic switching

## Selecting a QoS policy appropriate to operating conditions

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 a fair share function for users and a function guaranteeing TSS response.

## Restriction on the number of I/O requests issued by a single user

The FEFS limits the number of I/O requests issued or processed by each client or server in order to prevent I/O resources from being occupied by a specific user.
The client side limits the number of I/O requests that can be issued at the same time by a single user, thereby preventing the situation where numerous I/O requests issued from a single user occupy I/O bandwidth and server resources.
Suppose that a user application running on multiple clients issues I/O requests at the same time, behaving as though it were running on compute nodes. Consequently, there may be a risk that file server resources will be occupied. To prevent this, file servers can control the server processing capacity that can be allocated to a single user so that I/O requests from one user do not occupy the server resources.

## Guarantee of response at login nodes

Users judge the usability of a system directly from the response received when they access the system, so response to user access is more important than response to job access.
To guarantee response to access by a user using TSS on the login node, the FEFS is equipped with a function for assigning server resources to process I/O requests from login nodes. This can ensure good response for users who manipulate files on the login nodes even during file I/O by jobs on compute nodes.

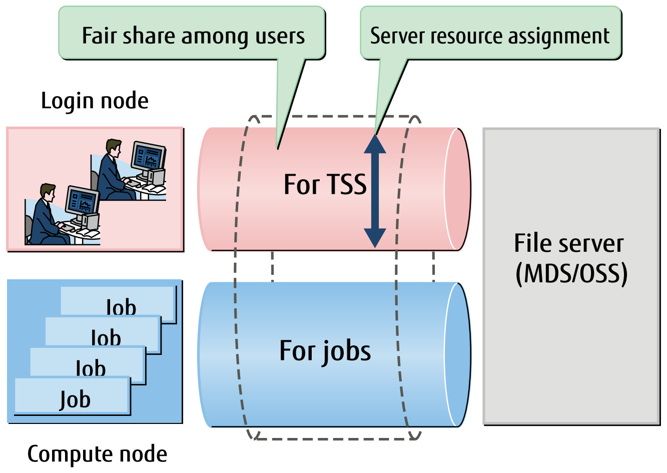


Figure 5 Guaranteed response for login nodes

Furthermore, the FEFS also supports best-effort operation so that all server resources can be used effectively. When there is no I/O request from any compute node, login nodes use all server resources. When there is no I/O request from any login node, compute nodes use all server resources.

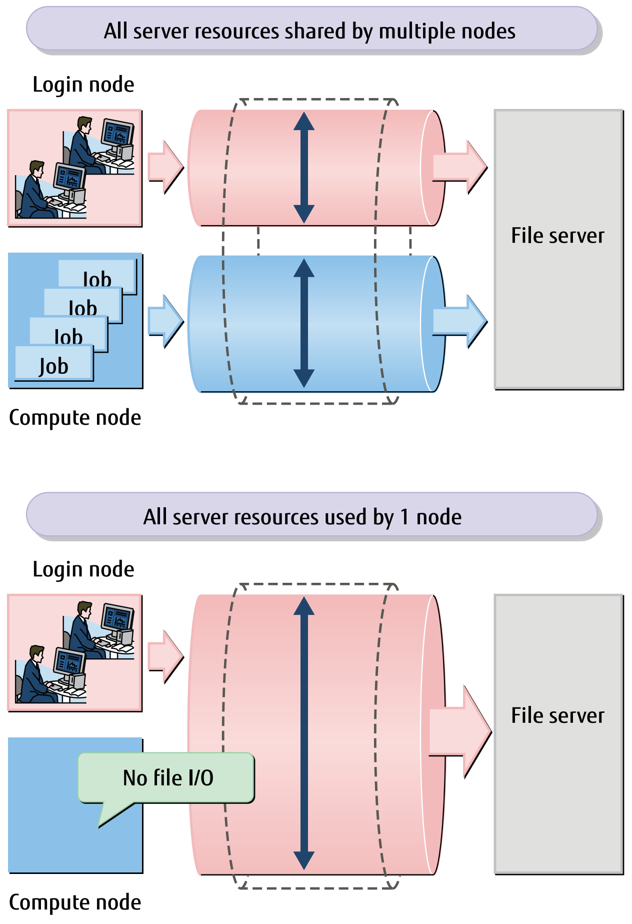


**Figure 6 Best-effort file server operations**

## Comparison with other file systems

Our clustered distributed file system is based on Lustre, an open-source distributed file system. Superior technologies from Lustre are joined together with extensions for large-scale systems in our system, which maintains client compatibility with Lustre. This makes it possible to flexibly build file systems according to performance, capacity, and operation policy requirements.
The following table at the end of this section compares our file system with Lustre(*1) and GPFS(*2) in terms of the main specifications for customers(*3).

Table 2 Comparison with competing file systems

| Item | Fujitsu | Lustre | GPFS |
|---|---|---|---|
| Maximum file system size | 8 EB | 64 PB | $2^{99}$ Bytes |
| Maximum number of files | $2^{63}$ | $2^{32}$ | $2^{64}$ |
| Maximum file size | 8 EB | 320 TB | NA |
| Maximum number of stripes | 20,000 | 160 | NA |
| Maximum block size | 512 KB | 4 KB | NA |
| POSIX ACL<br>Maximum number of entries | Available<br>8,191 | Available<br>NA | Available<br>NA |
| Disk quota | Available | Available | Available |
| Directory quota | Available | Not available | Not available |
| Fair share between users and TSS response guarantee | Available | Not available | Not available |

*1 http://wiki.lustre.org
*2 http://www-03.ibm.com/systems/software/gpfs/
*3 These specifications are as of June 2011 and may change in the future.

# Massively Parallel Programming Model and Language Processor

## Massively parallel programming model

CPU development efforts so far have focused on increasing clock frequencies to improve performance. Recently, however, it has become difficult to increase clock frequencies further. That is why the PRIMEHPC FX10 includes two functional enhancements intended to improve CPU performance: one is more cores, and the other is additional instructions specific to HPC (HPC-ACE: High Performance Computing - Arithmetic Computational Extensions). Processing by these cores on the scale of tens or hundreds of thousands running in parallel has improved performance.
Performance improvements in such a massively parallel processing architecture will soon reach a ceiling if the programming model relies only on process parallelism, because such a model tends to increase memory usage and network traffic. To alleviate this problem, we developed a technology named VISIMPACT (Virtual Single Processor by Integrated Multicore Architecture) for curbing increases in communication time during massively parallel processing and adopted this technology in the PRIMEHPC FX10. For programming with an automatic parallelization compiler, VISIMPACT makes it possible to treat a node like a single CPU by sharing L2 caches among cores and implementing hardware barriers between cores. Users can thus use this programming model with the following two benefits. One benefit is easy hybrid parallelism, which means automatic thread parallelism by the compiler for multiple cores within a node together with process parallelism for processing between nodes. The other benefit is a reduced inter-process communication overhead cost due to a decrease in the number of processes. The hardware barriers and shared L2 caches, which are basic features of this architecture, were already implemented in the FX1(*1) and have proved effective. This architecture has been further developed for the PRIMEHPC FX10, with enhanced CPU instructions and more cores within a node.

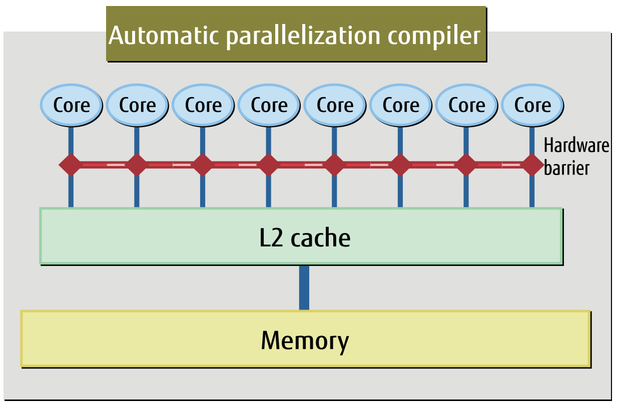


**Figure 1 Compute node architecture**

## PRIMEHPC FX10 language processor

The PRIMEHPC FX10 language processor is equipped with the following functions to enable programming with a VISIMPACT-based hybrid programming model.

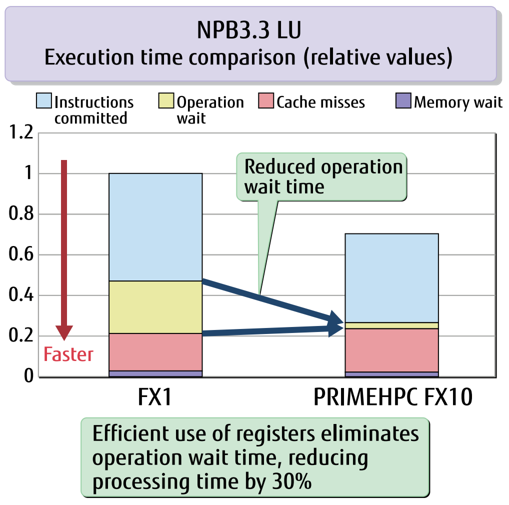
### ■ Compiler

The compiler supports the latest Fortran/C/C++ standards. By supporting not only general standards but also industry-standard language specifications, the compiler can compile widely used open-source software. In addition, the following optimization has been implemented to get new CPU functions and achieve high performance and high scalability:

- Improvement of computing efficiency within a core by expanding the loop optimization range and targets through effective use of expanded registers
- Higher processing speed by utilizing the SIMD (Single Instruction Multiple Data) operation(*2) to reduce the number of instructions executed
- Efficient cache use by providing directives that effectively use a sector cache(*3)

In addition to the above, the PRIMEHPC FX10 supports automatic parallelization, which is superior to vectorization, and OpenMP(*4) 3.0.

The following figure shows an example of performance improvement within one core after the optimization. As shown in the figure, the processing time taken to run the NPB (NAS Parallel Benchmark) program, as measured by the clock speed on a single core, is 30% less than on the FX1 because expanded registers and SIMD instructions were used.



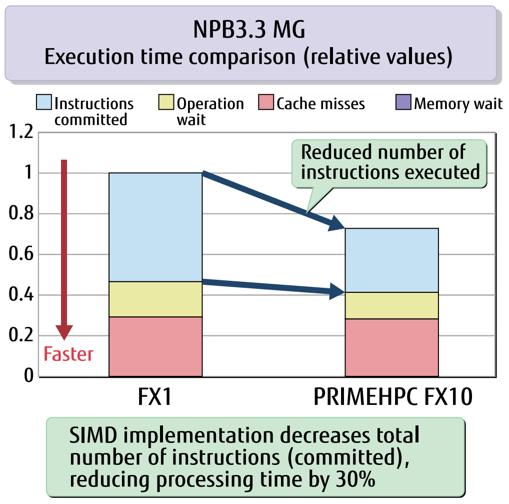


**Figure 2 Single-core performance comparison**

## MPI library

The PRIMEHPC FX10 provides MPIs that draw upon the Tofu interconnect features to enable parallel execution of more than tens of thousands of processes, which is a level of performance required by future HPC programs. To improve point-to-point communication performance, the PRIMEHPC FX10 uses a special type of low-latency path that bypasses the software layer. For further performance improvement, the transfer mode switch is optimized through additional consideration of the length and location of data being exchanged as well as the number of hops. Drawing upon the Tofu interconnect features, a dedicated algorithm has been developed for collective communication performance to control congestion. Frequently used functions (MPI_Bcast, MPI_Allreduce, MPI_Allgather, MPI_Alltoall, etc.) use this special algorithm instead of point-to-point communication. For MPI_Barrier and MPI_Allreduce, processing is faster with the advanced barrier communication feature (implemented in hardware) provided by the Tofu interconnect. The following figure shows the effect of advanced barrier communication for MPI_Barrier and MPI_Allreduce. As shown in the figure, the processing time for 768 processes is 85% to 88% lower than that of the software-implemented schemes.
As the number of processes increases, memory usage is becoming more critical. For the MPI of the PRIMEHPC FX10, we also developed memory saving functions, such as one for a dynamic connection method.

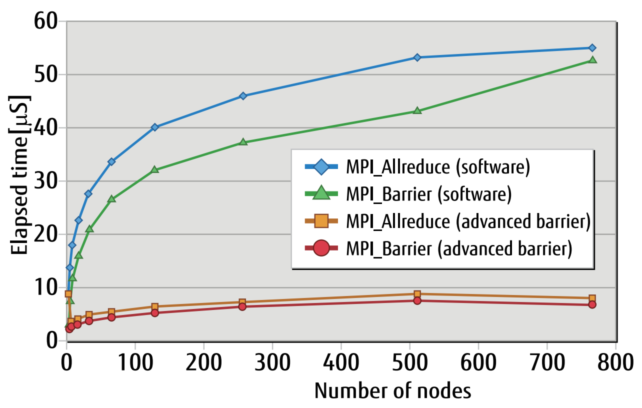


**Figure 3 Effect of advanced barrier communication**

## Tuning tools

In the tuning of highly parallel programs, sequential performance and high parallelization performance must be treated together. The PRIMEHPC FX10 provides tools that can collect appropriate tuning information. They support industry-standard interfaces to work together smoothly with the ISV tools that are familiar to some users. As shown in the following figure, the provided set of tuning tools obtains information from all layers, including the hardware, OS, library, and application layers.

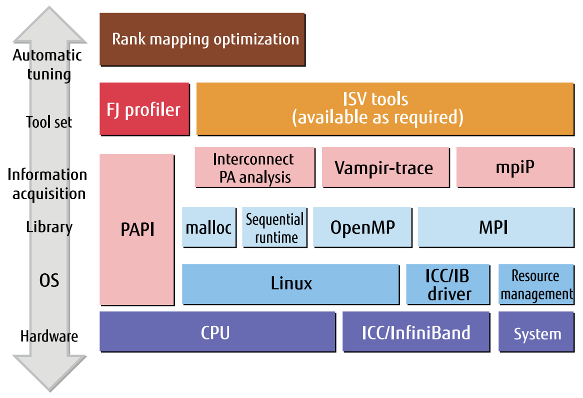
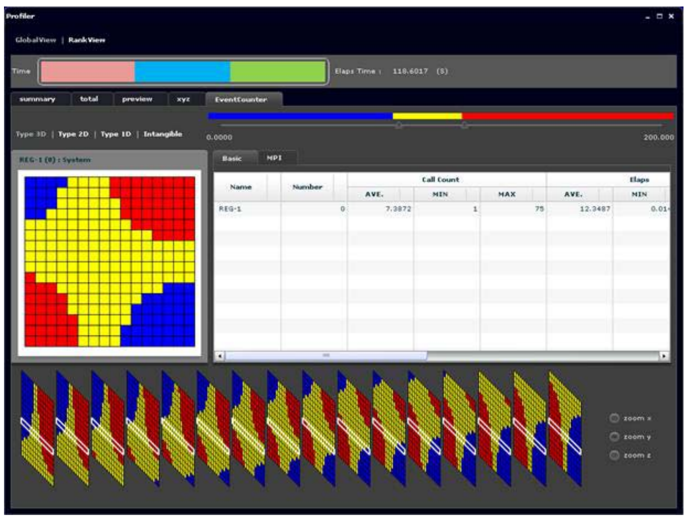


**Figure 4 Tuning tools organized in a stack**

For high parallelism, it is important to understand the behavior of each process. The PRIMEHPC FX10 tuning tools are equipped with functions that can collect PA information, which is provided by the CPU, for each process and graphically display the collected information. Using these functions, you can easily understand the states of processes and take appropriate action. The following figure shows the elapsed execution time of 4,096 processes (16 x 16 x 16) of a program. The colors of the displayed processes depend on the time taken for execution: red, yellow, and blue indicate relatively long, medium, and short times, respectively. The graph shows the relative length of time taken by individual processes, so you can see the performance balance between processes. One grid cell represents one process. You can view information on a process by placing the cursor on the corresponding cell.

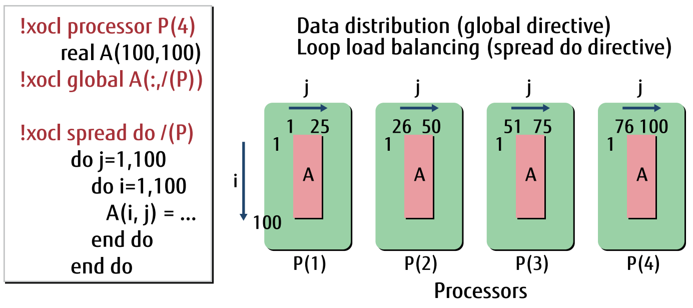


| Name | Number | Call Count AVE. | Call Count MIN | Call Count MAX | Elaps AVE. | Elaps MIN |
|---|---|---|---|---|---|---|
| REG-1 | 0 | 7.3872 | 1 | 75 | 12.3487 | 0.01 |

**Figure 5 Example of PA information visualization**

## Parallel programming language XPFortran

XPFortran (XPF) is Fujitsu's original programming language, a specialized Fortran extension for distributed memory computers. The extension comes mainly in the form of directives that supplement Fortran statements. XPF is designed so that processing can be parallelized stepwise from the stage of sequential programs. The XPF directives provide functions for (1) data distribution, (2) load distribution, and (3) communication and synchronization.
The following figure shows an example of a simple XPF program. Processor (virtual node) P, which is declared in a processor directive, is both a distribution destination for data A (global directive) and a load distribution destination for calculation loop do j (spread do directive). This example assumes use of automatic communication and synchronization.



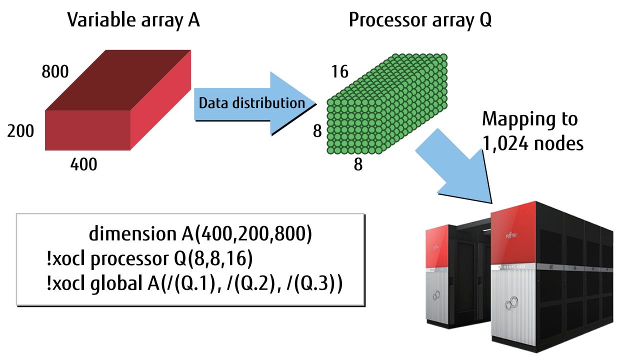
```
!xocl processor P(4)
      real A(100,100)
!xocl global A(:,/(P))

!xocl spread do /(P)
      do j=1,100
         do i=1,100
            A(i, j) = ...
         end do
      end do
```

Data distribution (global directive)
Loop load balancing (spread do directive)
j
1 25
26 50
51 75
76 100
1
i
100
A
P(1)
P(2)
P(3)
P(4)
Processors

**Figure 6 Example of an XPFortran program**

One advantage of XPF over the MPI lies in program development productivity. MPI programming requires programmers to not only rewrite sequential programs into SPMD (Single Program Multiple Data)[(*5)] ones written from the viewpoint of individual nodes but also explicitly write code for communication between nodes. Although the MPI is already in widespread use, writing entire programs relying only on the MPI will become increasingly difficult as hardware parallelism increases and the hardware hierarchy becomes more complex. In the future, mixed use of high-level languages, such as XPF, and the MPI will be more important. XPF can call procedures written in the MPI.
For highly parallel programming assuming parallel execution of between thousands and tens of thousands of processes, a higher importance is placed on multidimensional parallelism, in which a program is parallelized along multiple dimensions. XPF supports multidimensional virtual processor arrays so that multidimensional parallelism is efficiently implemented on the Tofu interconnect. For example, in the following figure, array variable A is a three-dimensionally distributed variable in processor array Q, which has an 8 x 8 x 16 shape as shown. A program that is parallelized along the three axes can be written for variable A. Q is configured with 1,024 processors. With the job scheduler or other such function, the processors can be assigned to 1,024 nodes configuring a virtual three-dimensional torus of the same shape. In this way, even simple programs written in XPF can gain the benefits of high-speed communication between neighboring nodes and regular, high-speed, collective communication from the Tofu interconnect.



```
dimension A(400,200,800)
!xocl processor Q(8,8,16)
!xocl global A(/(Q.1), /(Q.2), /(Q.3))
```

**Figure 7 Applying XPFortran to large-scale parallelism**

*1 The FX1 is Fujitsu's high-end technical computing server using the SPARC64™ VII processors.
*2 The SIMD operation is a technique for performing operations on multiple items of data with a single instruction. For details, see *Advanced Technologies of the Supercomputer PRIMEHPC FX10*.
*3 The sector cache is a new type of cache that can be controlled by software. For details, see *Advanced Technologies of the Supercomputer PRIMEHPC FX10*.
*4 OpenMP is a standardized way for expressing program parallelism.
*5 SPMD is a parallel execution scheme for running the same program on multiple processors with different data on each processor.

# Application Fields

## Supercomputer application fields

The scope of application of supercomputer-based simulations has widened in response to improvements in hardware performance and the progress of applications (simulation programs).
The PRIMEHPC FX10 is designed especially for high-speed execution of simulation programs that use highly parallel algorithms and mathematical calculation schemes. We believe that the PRIMEHPC FX10 will bring about breakthroughs possibly leading to further developments in science and technology as well as enhancing the competitive advantages of businesses.

The PRIMEHPC FX10 can be applied in various fields, which are roughly organized into the following two categories.

- **Basic research (the pursuit of truth, and national strategic research)**
  The central players in this category are the public agencies, universities, and other research institutes that use supercomputers in pursuit of the true nature of the universe, matter, life, etc. to understand universal laws in the natural world. Supercomputers are also used for research critical to national strategies, such as meteorological research (to address food problems and prevent disasters) and research on nuclear and fusion energy (to address energy problems).
  Many research groups develop and run their original simulation programs on supercomputers. Their results have led to great successes that could not be achieved if they relied solely on theories and experiments.

- **Applied research (research for product development or economic forecasting)**
  Private-sector businesses promote using supercomputers to increase the efficiency of developing their products (industrial goods, medicines, service products, etc.) in order to enhance competitiveness.
  For example, when compared with conducting numerous experiments, using supercomputers can greatly cut the time and cost of verifying functionality and quality in the product design phase (shortening the product development cycle).

## Expected application fields of the PRIMEHPC FX10

The following table lists representative examples of the expected application fields of the PRIMEHPC FX10.

Table 1 Expected application fields of the PRIMEHPC FX10

| Field | Representative application examples |
|---|---|
| Life science, medical care, and drug design | Elucidation of organ and cell functions, research on the interaction between proteins and drugs, virus molecular science, new medicine development, etc. |
| New materials and energy | New material development, nuclear safety analysis, fusion energy research, etc. |
| Prevention and mitigation of disasters, and forecasting of global environmental changes | Weather forecasting, meteorological prediction, prediction of seismic wave propagation, exploration of petroleum and other underground resources, prediction of changes in climate conditions (disasters) and marine ecology (fishery resources) caused by global warming, etc. |
| Next-generation manufacturing | Crash safety analysis of automobiles, aerodynamic characteristics analysis of airplanes and other aircraft, analysis of radio wave propagation of electronic equipment, etc. |
| Matter, and origin of universe | Research on elementary particles and atomic nuclei, exploration to reveal origins of astronomical bodies, etc. |
| Economics, finance, etc. | Economic model analysis, portfolio analysis, etc. |

In the future, supercomputers are expected to be used not only in individual fields but also used for applications extending across various fields like the following:

- Development of lightweight, high-strength materials for airplanes, automobile bodies, etc. (in the fields of new materials and manufacturing)
- New energy research and energy plant design focusing on safety and security (in the fields of energy, disaster prevention, and manufacturing)

Supercomputer simulations are becoming more and more important as a third research and development technique comparable to experiments and theories. They are expected to enhance the infrastructure of science and technology, create innovation, and strengthen competitive advantages in industry.

**Reference**

For more information about PRIMEHPC FX10, contact our sales personnel, or visit the following Web site:

http://www.fujitsu.com/global/services/solutions/tc/hpc/products/primehpc/

Advanced Software for the Supercomputer PRIMEHPC FX10

Fujitsu Limited
November 7, 2011, First edition
2011-11-07-WW-EN