High-Performance and Highly Reliable File System for the K computer

● Kenichiro Sakai ● Shinji Sumimoto ● Motoyoshi Kurokawa

RIKEN and Fujitsu have been developing the world's fastest supercomputer, the K computer. In addition to over 80 000 compute nodes, the K computer has several tens of petabytes storage capacity and over one terabyte per second of I/O bandwidth. It is the largest and fastest storage system in the world. In order to take advantage of this huge storage system and achieve high scalability and high stability, we developed the Fujitsu Exabyte File System (FEFS), a clustered distributed file system. This paper outlines the K computer's file system and introduces measures taken in FEFS to address key issues in a large-scale system.

1. Introduction

The dramatic jumps in supercomputer performance in recent years led to the achievement of 10 peta floating point operations per second (PFLOPS) in 2011. The increasing number of compute nodes and cores and the growing amount of on-board memory mean that file systems must have greater capacity and total throughput. File-system capacity and performance are expected to reach the 100-petabyte (PB) class and 1-terabyte/s (TB/s) class, respectively, as both capacity and performance are increasing at a nearly tenfold rate every year. For this reason, mainstream file systems have been changing from a single-server type to a clustered type.

We have developed a clustered distributed file system called the Fujitsu Exabyte File System (FEFS) with the aim of achieving a file system having a capacity and performance that can handle the computing performance of the K computer,[note)] which is currently the fastest supercomputer in the world.

This paper outlines the K computer's file system and introduces measures taken in FEFS to address key issues in a large-scale system.

2. File system for the K computer

To achieve a level of performance and stable job execution appropriate for the world's fastest supercomputer, We adopted a two-layer model for the file system of the K computer. This model consists of a local file system used as a high-speed, temporary-storage area for jobs only and a global file system used as a large-capacity shared-storage area for storing user files (**Figure 1**). In this two-tiered model, jobs are executed through data transfers between these two file systems using a file-staging control function. The following describes the respective roles of

note) "K computer" is the English name that RIKEN has been using for the supercomputer of this project since July 2010. "K" comes from the Japanese word "Kei," which means ten peta or 10 to the 16th power.

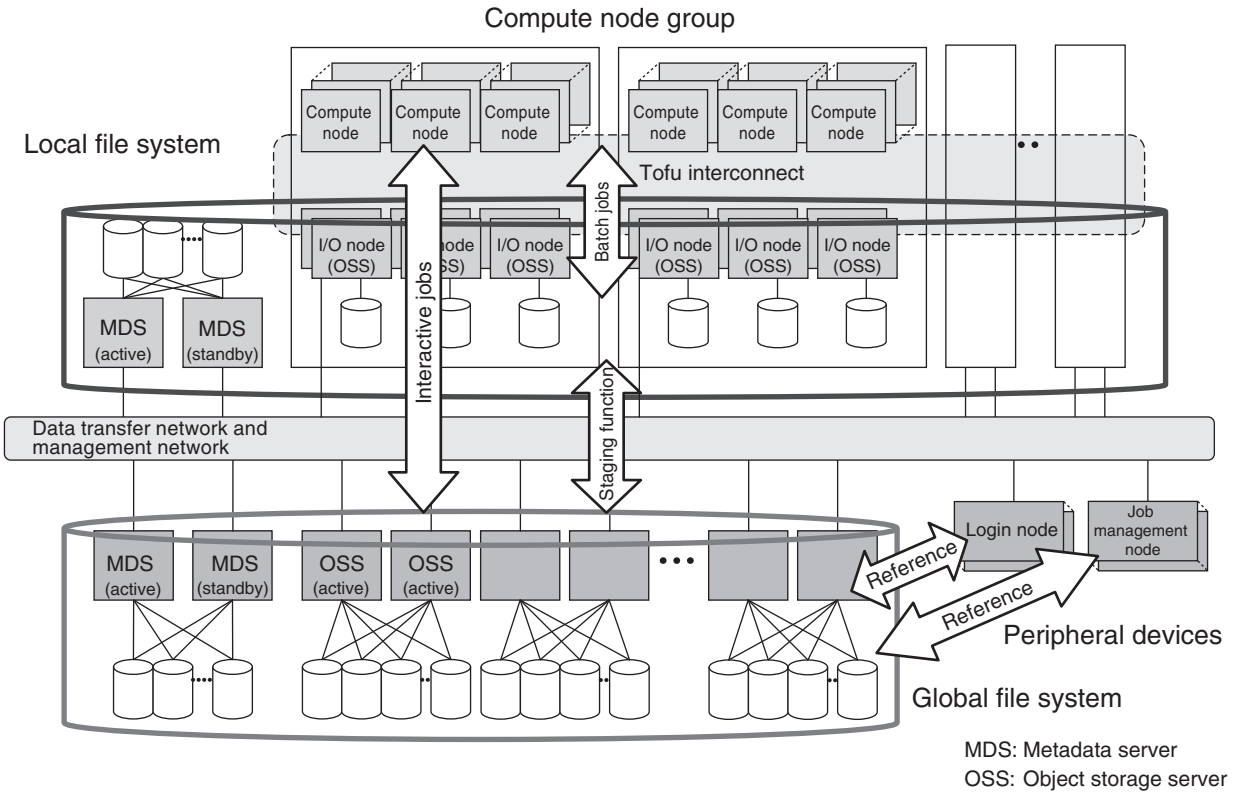

Figure 1
Two-layer file system model.

the local file system, global file system, and file-staging process.

1) Local file system

This is a high-speed, temporary storage area dedicated to job execution for extracting maximum file-I/O performance for applications executed as batch jobs. The local file system transfers input/output files from and to the global file system through the file-staging function and temporarily holds job files that are either under or waiting for execution.

The file server used to access data blocks connects to a compute node through the Tofu interconnect using an I/O-dedicated node (I/O node) within the compute rack through remote direct memory access (RDMA) communications. As a result, low-latency, high-throughput file data transfer is achieved.

2) Global file system

This is a large-capacity, shared storage area external to the compute racks for storing user files consisting of job input/output data and other content. In addition to providing file access through the login node, the global file system enables direct file access from a compute node when executing an interactive job so that a user can perform program debugging and tuning while checking the job output in real time.

A compute node connects to a group of file servers in the global file system through an InfiniBand link mounted on the I/O-dedicated node. In short, an I/O node relays data between the Tofu interconnect and InfiniBand while a compute node accesses file servers via that I/O node.

3) File staging

File transfer between the local and global file systems is performed automatically by the system through the file-staging function, which operates in conjunction with job operation software. Before a job begins, this function transfers the input files for that job from the

global file system to the local file system (stage-in), and after the job completes, it transfers the output files from the local file system to the global file system (stage-out). The user specifies the stage-in and stage-out files in a job script.

3. FEFS: Ultra-large-scale cluster file system

We developed FEFS to provide a local file system and global file system suitable for the top-class performance of the K computer. The following objectives were established in developing FEFS:

- World's fastest I/O performance and high-performance Message Passing Interface (MPI)-IO
- Stable job run times achieved by eliminating interruptions
- World's largest file-system capacity
- Greater scalability in performance and capacity by adding hardware
- High reliability (service continuity and data preservation)
- Ease of use (sharing by many users)
- Fair share (fair use by many users)

To achieve these objectives, We developed FEFS on the basis of the Lustre open-source file system, which has become a worldwide industry standard, while extending certain specifications and functions as needed. As shown in **Table 1**, FEFS extends specifications like maximum file system size and maximum file size to the 8-exabyte (EB) class relative to existing Lustre specifications. File system size can be extended even during system operation by adding more servers and storage devices.

The following sections describe the FEFS measures listed below for dealing with an ultra-large-scale supercomputer system.

- Elimination of I/O contention by using communication-bus and disk splitting
- Continuous availability by using hardware duplication and failover
- Hierarchical node monitoring and automatic switching
- QoS-policy selection by type of operation

4. Elimination of I/O contention by using communication-bus and disk splitting

As a clustered file system, FEFS bundles together a large number of file servers and improves parallel throughput performance in proportion to the number of file servers. However, if access should become concentrated on specific file servers, communications can become congested and disk-access contention can arise, resulting in a drop in I/O performance and a variation in job run times between compute nodes. It is therefore essential that file I/O contention be completely eliminated to ensure stable job execution.

Table 1
Lustre and FEFS comparison.

Function		Lustre	FEFS
System specifications (max. values)	File system size	64 PB	8 EB
	File size	320 TB	8 EB
	No. of files	4 G	8 E
	OST volume size	16 TB	1 PB
	No. of stripes	160	20 000
	No. of ACL entries	32	8191
Scalability (max. values)	No. of OSSs	1020	20 000
	No. of OSTs	8150	20 000
	No. of clients	128 units	1 000 000 units
Block size (Lustre disk file system) (backend file system)		4 KB	up to 512 KB

In FEFS, file I/O contention is eliminated by separating file I/O access into two stages, namely, in units of jobs and units of compute nodes (**Figure 2**).

At the job level, each job is assigned a separate range of I/O nodes corresponding to file-storage destinations to eliminate file I/O contention on servers, on the network, and on disks. And on the compute-node level within a job, the transmitting and receiving of file data is performed via I/O nodes constituting a minimum number of Tofu-communication hops, thereby minimizing file I/O contention among compute nodes.

5. Continuous availability by using hardware duplication and failover

In addition to high performance, another important requirement of a large-scale computer system is high reliability. A clustered file system consists of many file servers, storage devices, and network devices, and a fault or failure in any of these or even operation suspension for maintenance must not prevent file-system service and system operations from continuing.

However, when configuring FEFS with more than several hundred file servers and storage devices, there are bound to be times when hardware such as network adaptors and servers are in a fault or maintenance state. To ensure that operations across the entire system continue even under such conditions, it is essential that faults be automatically detected and detours taken around faulty locations so that file-system service can continue unimpeded.

For this reason, FEFS duplicates hardware and uses a software-based control process to initiate failover processing at the time of a hardware fault. This processing switches the server and I/O communication paths and restores the service in progress to its normal state. This prevents a single fault from halting system services and enables system operations to continue (**Figure 3**).

6. Hierarchical node monitoring and automatic switching

A large-scale computer system must have a mechanism for detecting faults without human

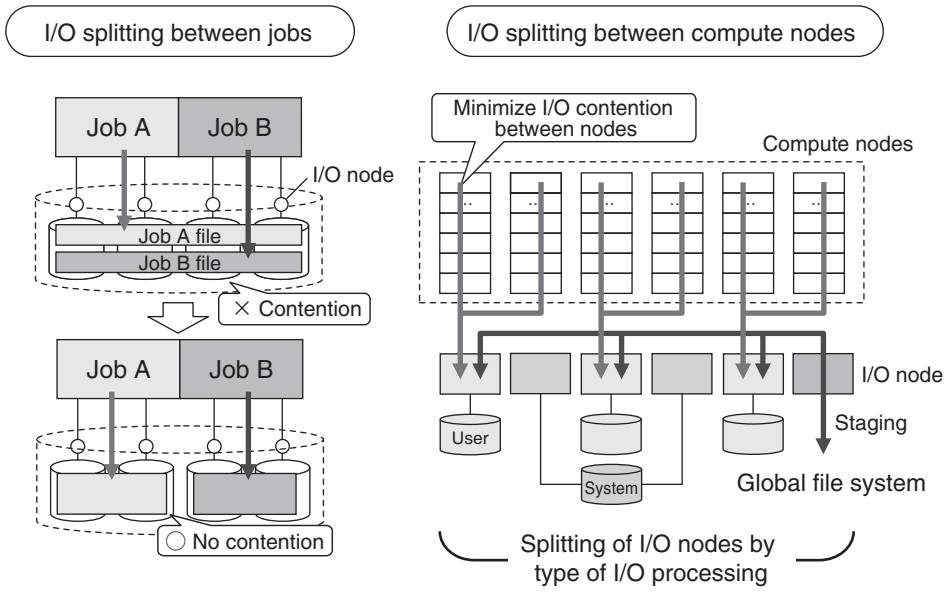

Figure 2
Elimination of contention by I/O splitting.

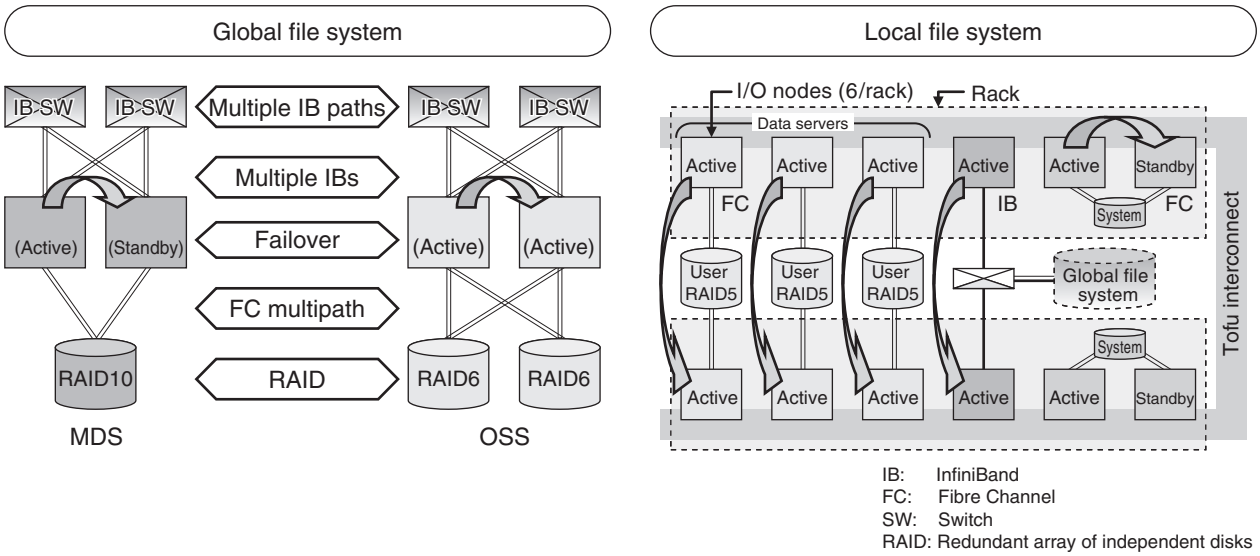

Figure 3
Fault tolerance through hardware duplication.

intervention and for automatically notifying nodes in the affected range that node switching is being performed. One method for monitoring node states is to have compute nodes and file servers exchange monitoring packets, but this can generate a large number of packets exponentially proportional to the scale of the system, which can hinder MPI communications among compute nodes and data communications between compute nodes and file servers.

As a countermeasure to this problem, FEFS performs hierarchical node monitoring and switching control in conjunction with system management software to minimize the load on system communications. This approach achieves efficient and automatic switching by performing tree-based monitoring consisting of monitoring within a compute rack, monitoring in units of node groups by grouping multiple racks, and monitoring higher-order node groups (**Figure 4**).

7. QoS-policy selection by type of operation

In a large-scale system used by many users as in datacenter operations, large-capacity file I/O performed by some users should not affect the operations of other users. Likewise, file access from jobs on compute nodes should not affect system response for users on login nodes.

In FEFS, these problems are solved by using an inter-user fair-share function for file I/O and a response-guarantee function in a time sharing system (TSS) environment, as described below.

1) Restriction on I/O requests issued by a single user

In FEFS, the number of I/O requests issued on the client side and processed on the server side is controlled so that a specific user does not monopolize I/O resources (I/O network, servers, disk equipment), as shown in **Figure 5.**

On the client side, FEFS controls the number of I/O requests that can be issued simultaneously by a single user to prevent that user from issuing a large quantity of I/O requests and monopolizing I/O resources.

It is also possible for I/O resources to be monopolized if I/O requests are issued by an application belonging to the same user from multiple clients such as compute nodes. Thus, on the file-server side, FEFS controls the server processing capacity that can be used by a single user to prevent I/O resources from being monopolized by the I/O requests from that user.

2) Login-node response guarantee

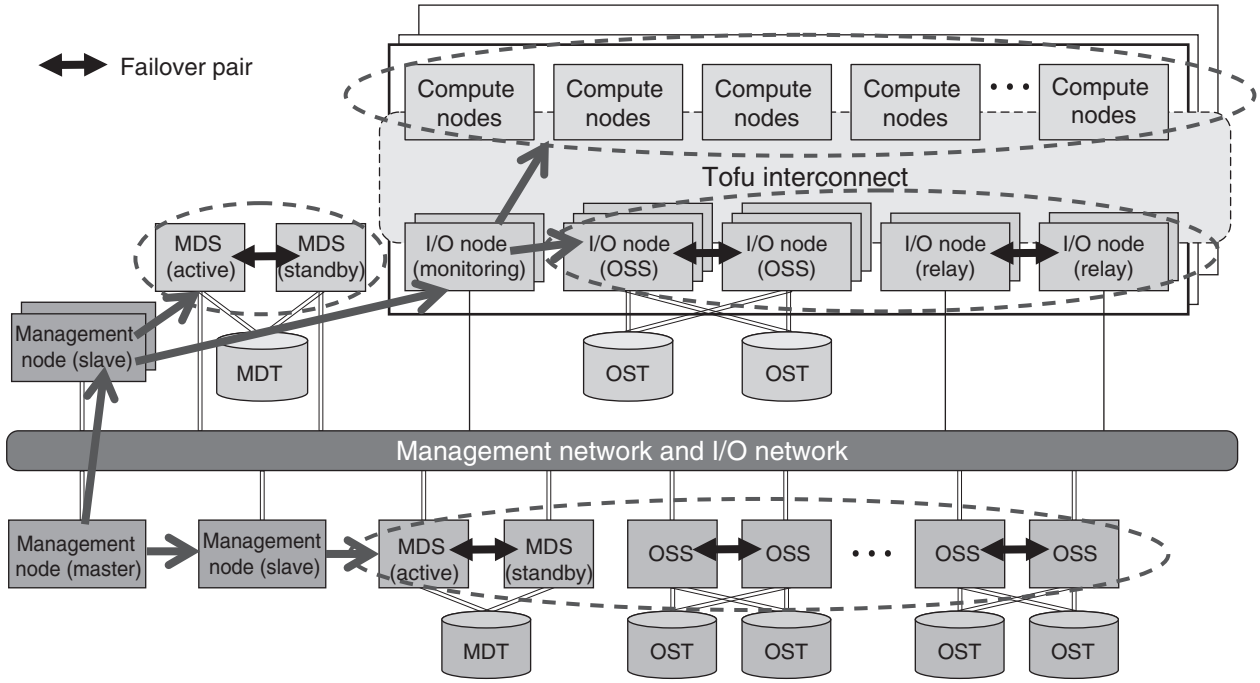

Figure 4
Hierarchical node monitoring and automatic switching.

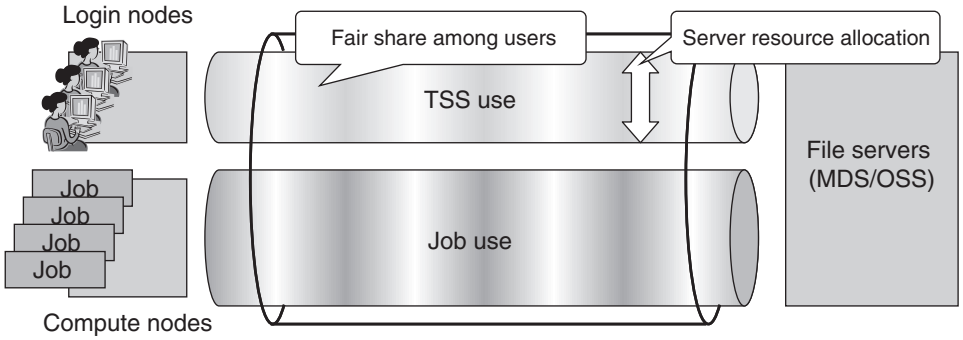

Figure 5
Login-node response guarantee.

Access response felt by users on login nodes is directly related to system ease-of-use and is therefore more important than access response with respect to jobs.

In FEFS, access response for users on the TSS at login nodes is guaranteed by using a function that allocates server resources for processing I/O requests from login nodes. This scheme secures adequate response for users accessing files from login nodes even when jobs on compute nodes are performing file I/O operations.

3) Best-effort use of I/O bandwidth

Best-effort operations have been made possible so that server resources are not used up but effectively used (**Figure 6**). If I/O requests are being issued from both login nodes and compute nodes, all server resources will be shared by these two types of nodes (left portion of Figure 6), but if no I/O requests are being issued from compute nodes, login nodes will use all server resources (right portion of Figure 6).

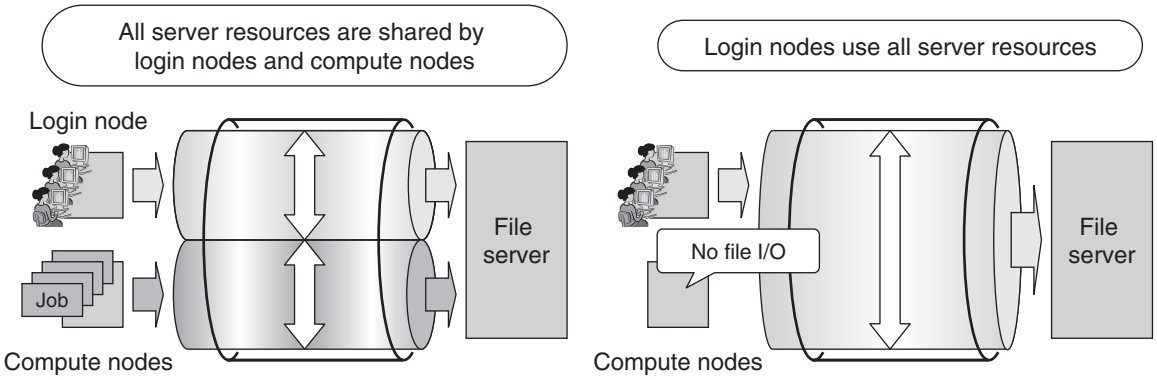

Figure 6
Best-effort use of I/O bandwidth.

Many users are controlled by FEFS, and the file activity of any particular user must not be allowed to affect other users, particularly users on login nodes. The QoS policy is thus selected by type of operation to ensure that each user receives a fair share of the system resources and that each user on the TSS is guaranteed a timely response.

8 FEFS data transfer performance

The K computer is still in the development stage, and FEFS is still undergoing operational tests as the scale of the system is gradually increased. The file I/O performance for two columns of disk racks out of a system total of nine columns is shown in **Figure 7**. These results were obtained using the interleaved or random (IOR) benchmark, a standard used for measuring I/O. They show that FEFS achieves a file-reference performance of 340 GB/s when performing parallel I/O operations on 580 servers. This is presently the top I/O performance in the world.

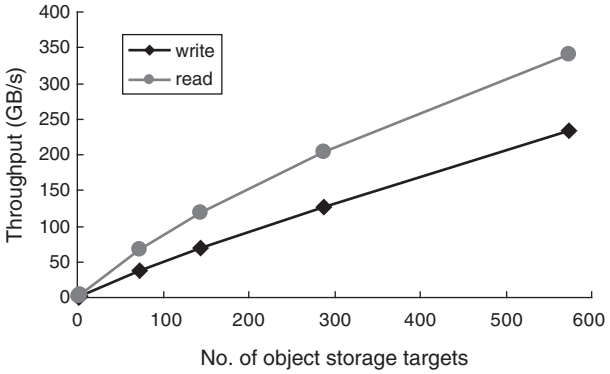

Figure 7
Throughput performance (IOR benchmark).

9. Conclusion

This paper introduced the FEFS clustered distributed file system developed for the K computer. The file system bundles together file servers on a level of several thousand units, achieving a large-capacity system in the 100-PB class and high I/O performance in the 1-TB/s class. It provides efficient node monitoring and automatic node switching requiring no human invention in conjunction with the operation and management software for the K computer. It facilitates file access through a QoS-policy selection function that mitigates contention between users and jobs sharing this large-scale system.

Looking forward, Fujitsu intends to improve FEFS reliability and performance through ongoing development efforts. It also strives for the stable operation of the K computer. It seeks to apply the file-system know-how gained with the K computer to its PRIMEHPC FX10 commercial supercomputer and PC cluster systems and to provide file systems for middle-end to high-end

supercomputers. At the same time, Fujitsu will be working to reflect its file-system achievements on the K computer in future versions of Lustre with an eye toward standardization through collaborative efforts with the Lustre OpenSFS development community and Whamcloud, a Lustre development company.

Kenichiro Sakai
Fujitsu Ltd.
Mr. Sakai is engaged in the development of software for large-scale file systems running on supercomputers.

Shinji Sumimoto
Fujitsu Ltd.
Mr. Sumimoto is engaged in overall technology development for HPC system software involving high-performance communication such as the MPI communication library and cluster file system.

Motoyoshi Kurokawa
RIKEN
Dr. Kurokawa is engaged in development of system as a whole.