

Development of Management File Sharing System not requiring File Server Specialized for Server Management System

Mitsuyoshi KITAMURA^{*1}, Tatsuya OKAZAKI^{*2}, Ryoma TANAKA^{*3}, and Kouya ARASHIRO^{*4}

Abstract

In the file servers that are normally used to share management files in server management systems, complicated controls, such as management file sharing methods, are required both as preparation against failure and to ensure file server redundancy. To facilitate such efforts, we propose a management file sharing system (MFSS) for server management systems that does not require a specialized file server. Herein, the detailed configuration of our proposed MFSS, its operation at the times of server failure and recovery, and its overall performance are discussed and evaluated. Then, using various examples, including a high-availability server system and a power-saving server system that utilize different management timings, we demonstrate the utility of our proposed MFSS.

1. Introduction

The spread of new coronavirus infections has various implications around the world. For example, online learning and remote work are spreading rapidly in Japan, and the study methods and working styles of numerous people are changing. These days, cloud services that can provide user-required functions whenever they are needed have become critical social infrastructure. Of these, edge computing [1]-[4], which speeds up the response of cloud services, reduces security risks, and strengthens resistance to failures, is attracting significant attention. Edge computing processes the vast amounts of data generated from devices such as Internet of Things (IoT) terminals on the edge of the cloud by constructing a distributed architecture. In this way, computer systems continue to evolve, and it is expected that social digitalization trends will continue in the future. This, in turn, means cloud computing will become even more important in the future.

In response to these trends, numerous important studies focusing on cloud computing are underway. These include detailed research on data duplication technology in cloud storage and a proposed deduplication scheme [5]; the development of new models and interfaces such as Cyber Compute-First Networking (Cyber-CFN) to generate uniform sampling independent of scalability and network architecture issues [6]; and efforts to analyze cloud server system behaviors mathematically via the Markov modeling process and thus perform availability assessments that consider various other influential factors [7].

Moreover, a new address and routing architecture for cloud service data center networks with regular topology has been reported [8], and a novel load balancing scheduling that considers two-phase energy using virtual machine migration in terms of energy and load balancing, which are important issues in cloud data centers [9], have been discussed. Other advancements include a proposed adaptive and efficient cloud resource allocation method based on actor click deep learning to address the issue of efficient resource allocation in cloud data centers and the development of superior quality of service (QoS) in terms of job delays and rejection rates compared to conventional methods, as well as energy efficiency improvements that have been verified via simulation experiments based on actual data [10].

Research efforts on data sharing also include the proposal of a group key management protocol for file sharing related to safe and efficient cloud storage and data sharing services [11]; the proposal of a split protocol as a protection technique for improving the security, integrity, and availability of data stored in the public cloud [12]; the configuration of a distributed file-sharing platform based on the current mainstream blockchain and a demonstration of its effectiveness against free-riding problems [13]; and improvements to user data file availability that are made possible by considering resource allocation and search efficiency in distributed file systems [14].

Furthermore, since server systems are important components for providing cloud services, failover cluster or load balancer cluster systems are generally adopted as methods for realizing high availability. However, the basic configurations of such systems call for redundant servers that provide the same services, which imposes problems in terms of cost and power consumption. As a countermeasure, a multiple server backup system (MSBS) [15] that can back up the server functions of multiple real servers with one real server has been proposed. The proposed MSBS, which is

^{*1} Associate Professor, Information Technology course, Tokyo polytechnic university

^{*2} second year, Electorionics and Information Technology, Tokyo polytechnic university graduate school of engineering

^{*3} first year, Tokyo polytechnic university graduate school of engineering

^{*4} fourth year, Information technology course, Tokyo polytechnic university

Received Sept. 25, 2023

based on a dynamic backup server system (DBSS) that manages one target server, starts multiple DBSSs on one management server to manage a server system.

Separately, methods that facilitate rapid failure recovery efforts by improving data redundancy before a serious server failure or network error occurs, such as by realizing a highly available server system using Kubernetes and distributed storage [16] and by examining random access memory (RAM), hard disk drive (HDD), and central processing unit (CPU) errors by observing or simulating kernel ring buffer log files on a Linux server [17], are essential. These purposes can be accomplished by detecting impending failures in the servers that comprise high-availability distributed clusters along with other minor network abnormalities that can affect cluster operations [18]. In addition, the adoption of a system interface [19] that simplifies the control programs of power-saving and high-availability server systems during their development, and two power-saving configuration types that consider server system operating environments, have been reported [20].

Although methods in which dedicated management servers monitor all target servers are generally widespread, it is still necessary to design for cases where management servers fail. In such cases, the control method becomes complicated, and the management load increases as the number of target servers increases. In response to those points, a Peer-to-Peer (P2P) server management system [21] that does not require a dedicated management server has been proposed. In this system, since all target servers have management server functionality, it is necessary for all servers to share management data. To accomplish this, a file server is normally used as a file-sharing method, which means it is also necessary to consider potential cases where the file server fails. To resolve this problem, a synchronous editing method (SEM) [22] is proposed as a file-sharing method. However, although it is possible to apply the SEM to a ring-type dynamic backup server system (RDBSS) [23] when managing a real server system that has a management file format like the P2P method, this method cannot be adopted to server management systems with different file processing timings, such as power-saving server management systems.

With those points in mind, this research proposes a management file sharing system (MFSS) that does not require a file server specialized for server management systems and provides detailed information on its configuration and operation at the time of server failure and recovery. In this study, we apply our proposed MFSS to an RDBSS and a power-saving server management system (PSS), which does not require a dedicated management server, to investigate whether our proposed system can be used efficiently. Additionally, a conventional system that shares management files using a file server was adopted as a method for comparison with the MFSS, and an experimental server system that can operate both the proposed and conventional methods was constructed. In the RDBSS, the server function recovery time was measured, while in the PSS, we measured the time required for shifting to the server power-saving configuration. A comparative examination was then performed based on the times measured by the proposed and conventional methods.

The remainder of the present paper is organized as follows. In Section 2, overviews of the RDBSS, PSS, the management file requiring file sharing and their sharing timing, and the startup method of the virtual server required to recover the failed server function, are all described. Next, Section 3 describes the MFSS configuration, its operational outline, malfunction prevention of management file editing, server failure countermeasures, and file sharing time measurements. In Section 4, we describe the configuration and specifications of the experimental system and show that the MFSS operates normally in the RDBSS and PSS. Finally, we provide our conclusions in Section 5.

2. Management files in the server management system, their sharing timing, and how to manage virtual servers

2.1. Management file in RDBSS

Figure 1 shows the shared files of the RDBSS [23]. Here, the RDBSS was adopted for the virtual server system and that server provides mail (M), web (W), and file transfer protocol (F) services, while RS1 through RS3 indicate the real servers used to create the virtual server. In the RDBSS, one management server manages one target server. Because it operates as a ring configuration, the group, operation, problem, and heartbeat files must be shared by each management server as management files. The RDBSS adopts a dual monitoring method that uses both internal and external monitoring. The heartbeat file is created during internal monitoring, and a management configuration that predicts failures just before performing external monitoring can be shifted. Here, as shown in the figure, when W1 fails, M1 controls the creation of W1b to recover its server function, and then extends management for F1 in order to maintain the ring management structure. Next, since it is a management file, W1 information is deleted from the group file and moved to the separation file. After that, the operation file is edited based on the edited group file. Note that since the failure status is added to the problem file, it is necessary to share this edited information on each management server.

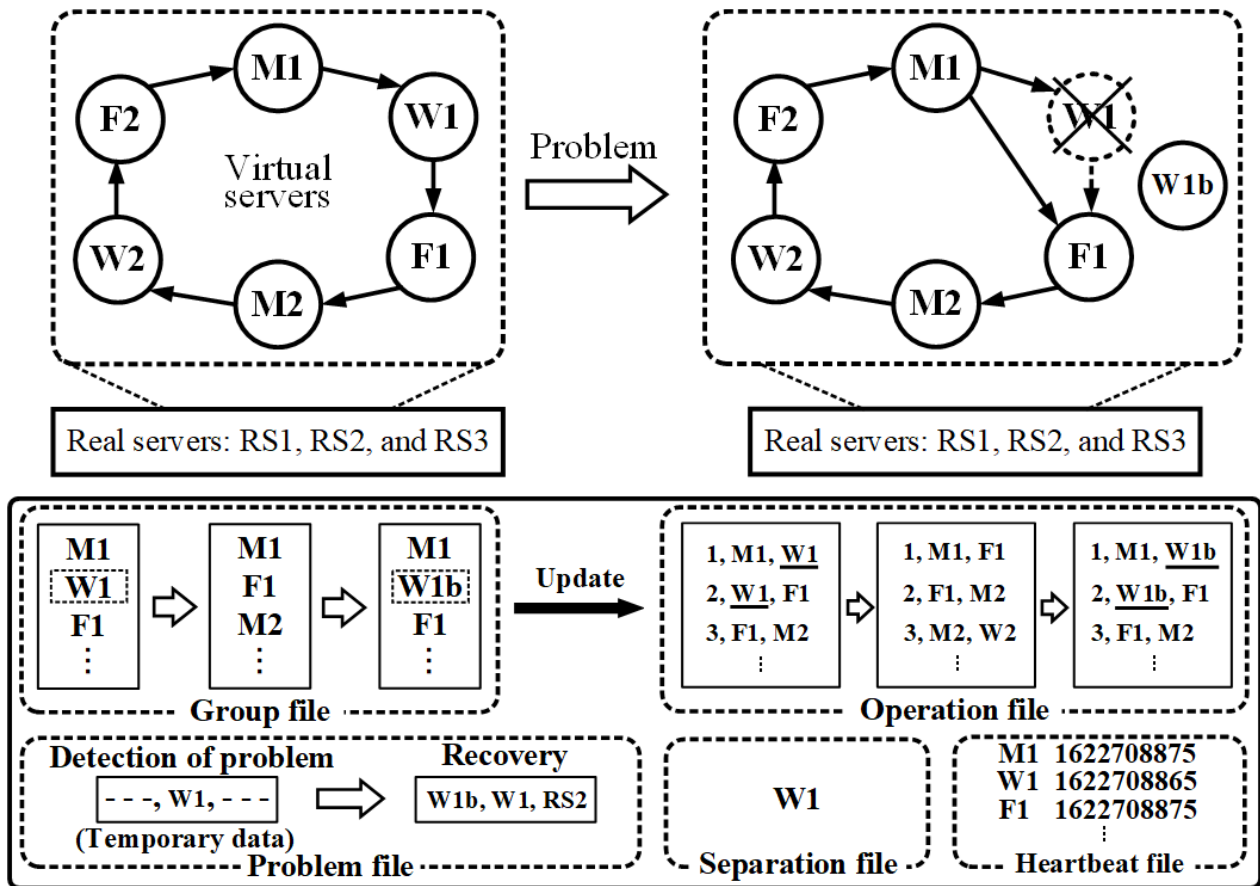


Fig. 1: Management file in RDBSS.

2.2. Management file in PSS

Figure 2 shows the shared file for the PSS [19]. As can be seen in Figs. 2(b) and (c), this system realizes power saving by suspending the real server according to its service provision state. In the figure, in which M, W, and F services are provided to the clients, virtual servers M1 and M2 belong to the M service group; virtual servers W1, W2, and Ws belong to the W service group; and virtual servers F1, F2, and Fs belong to the F service group. Ws and Fs, which are also virtual servers, are spares used for maintaining redundancy in the power-saving state. VIP indicates a virtual IP address. RS1 through RS3 are real servers that are used to create virtual servers. As real servers to be used in the power-saving state, RS1 and RS2 are set as Base Servers 1 and 2, respectively.

If the load on each server is low in the moderate load condition shown in Fig. 2(a), the server system shifts to the light load condition (PS1) shown in Fig. 2(b). If the number of accesses to each server is small in the light load condition (PS1), the server system construction shifts to the light load condition (PS2) shown in Fig. 2(c). The information of the virtual server providing service to clients is recorded in the operation file, the information of the virtual server in the access stopped state from clients is recorded in the access stop file, and the information of the real server in operation is recorded in the real server file. In addition, the load status of each server is recorded in the load file. As shown in the figure, each file is edited if the server system shifts from the state shown in Fig. 2(a) to the state shown in Fig. 2(b) or from the state shown in Fig. 2(b) to the state shown in Fig. 2(c). Please note that it is necessary to share this edited information with each management server.

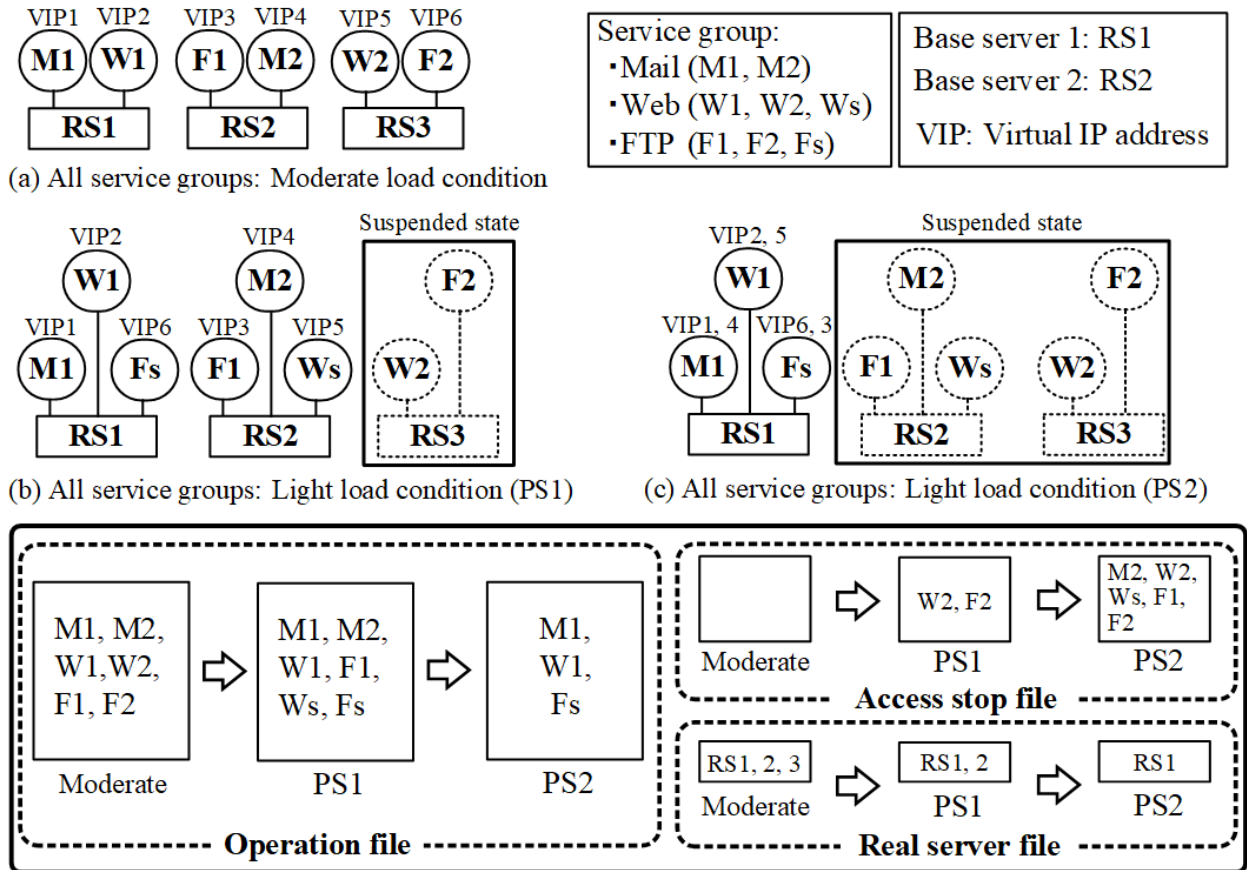
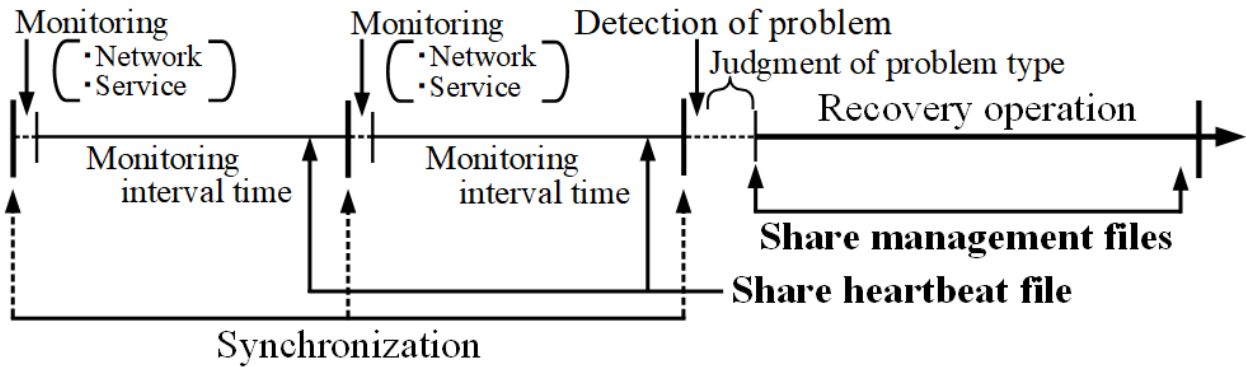


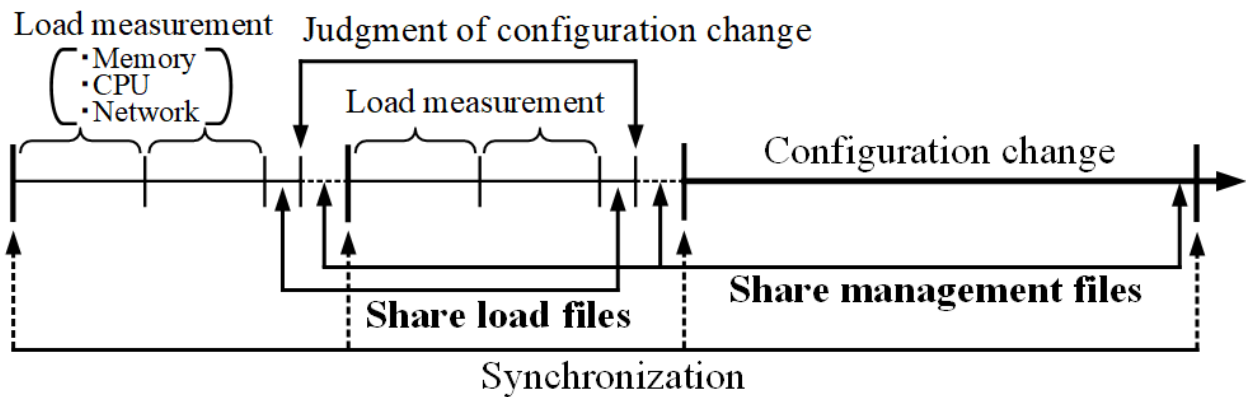
Fig. 2: PSS management file.

2.3. File sharing timing in server system management

Figure 3 shows an overview of file sharing timing in server management systems. In this figure, “synchronization” indicates the process at the monitoring start of each server in the RDBSS and at the examination start in the PSS. The heartbeat file is regularly shared in the RDBSS to disseminate the internal monitoring results. In addition, when a problem is detected in the target server, the management file sharing process also occurs because the management file is edited, as shown in Fig. 1. The sharing processes for load and management files in the PSS occur during each examination period, as shown in Fig. 2. Management file sharing also occurs when the server system configuration is changed.



(a) Operation timing for RDBSS



(b) Operation timing for PSS

Fig. 3: File sharing timing in server system management.

2.4. Original server method for creating virtual server and its automatic replication system

Here, we indicate the original server method [22], which places system data of a virtual server used to recover server functions within the management server, and the system that automatically replicates it [24]. In the RDBSS, if the target server fails, a virtual server that can provide the same service as that server is created, and the function of the failed server is recovered. However, because the management server that creates this virtual server is determined according to the load status, it is not possible to allocate the system data of the virtual server on a specific management server. Therefore, the original server method, the operational outline of which is shown in Fig. 4, was proposed. In this method, S1, S2, and S6 are real servers, while S2b is a virtual server used for recovering the server function of S2. The original server system data are the virtual server system data that can provide the services of all target servers, and the installation procedure file is required to make the original server capable of recovering the failed server functions. This procedure file contains information such as the installation procedure used to construct the service program provided by the failed server and the startup settings of that program. In the figure, S1 detects the failure of S2, controls S6, and simultaneously creates an original server using the system data for the original server saved in S6. After that, to operate the original server as S2b, the “Install procedure file (S2b)” is transferred and installed.

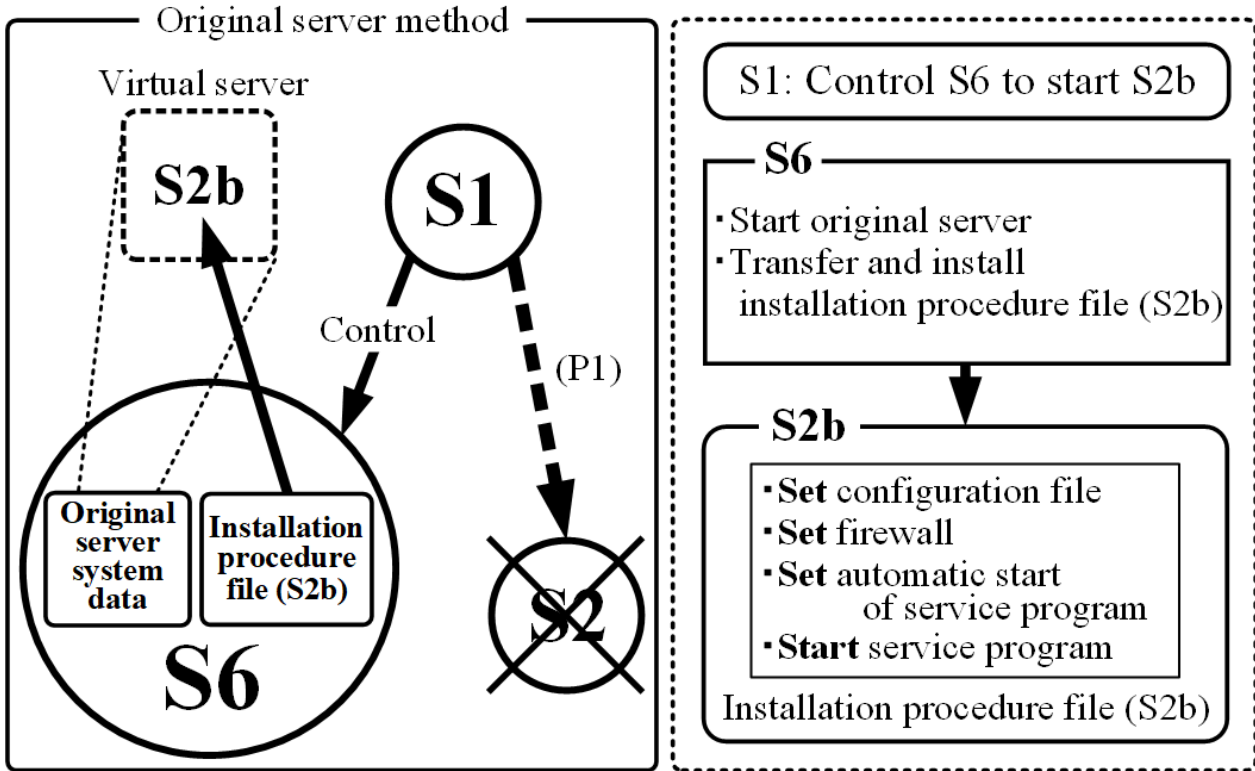


Fig. 4: Operation outline of original server method.

In the case of a virtual server system, if the real server fails, it is necessary to create a backup server that recovers the functions of the failed virtual server on the remaining real server. However, the number of backup servers varies depending on the situation, which means it is not possible to determine the number of original server system data to allocate on each real server in advance. To address this issue, the original server automatic duplication system shown in Fig. 5 has been proposed. In this system, an available original server file, a reservation file, and a real server file are used as management files. The available original server file refers to this file when using the original server. The system data name of the original server to be duplicated is recorded in the reservation file, while the information of the real server in operation is recorded in the real server file, and the information of the real server judged to be abnormal is deleted from this file.

In the figure, S1 through S6 are virtual servers that provide services to clients, S1D through S6D are virtual server system data, OD1 through OD12 are original server system data, and ODS1 through ODS3 are the system data sources used to clone the original server system data. The number of original server systems allocated on each real server is the average number of virtual servers created on the real server. As shown in the figure, if RS3 fails, S3b and S6b are created on RS1 and RS2 to recover the functions of S3 and S5 created by RS3. OD1 and OD3 are used when creating S3b and S6b, respectively. Because the average number of virtual servers created on a real server is three, system data of the two original servers are automatically created as a shortfall.

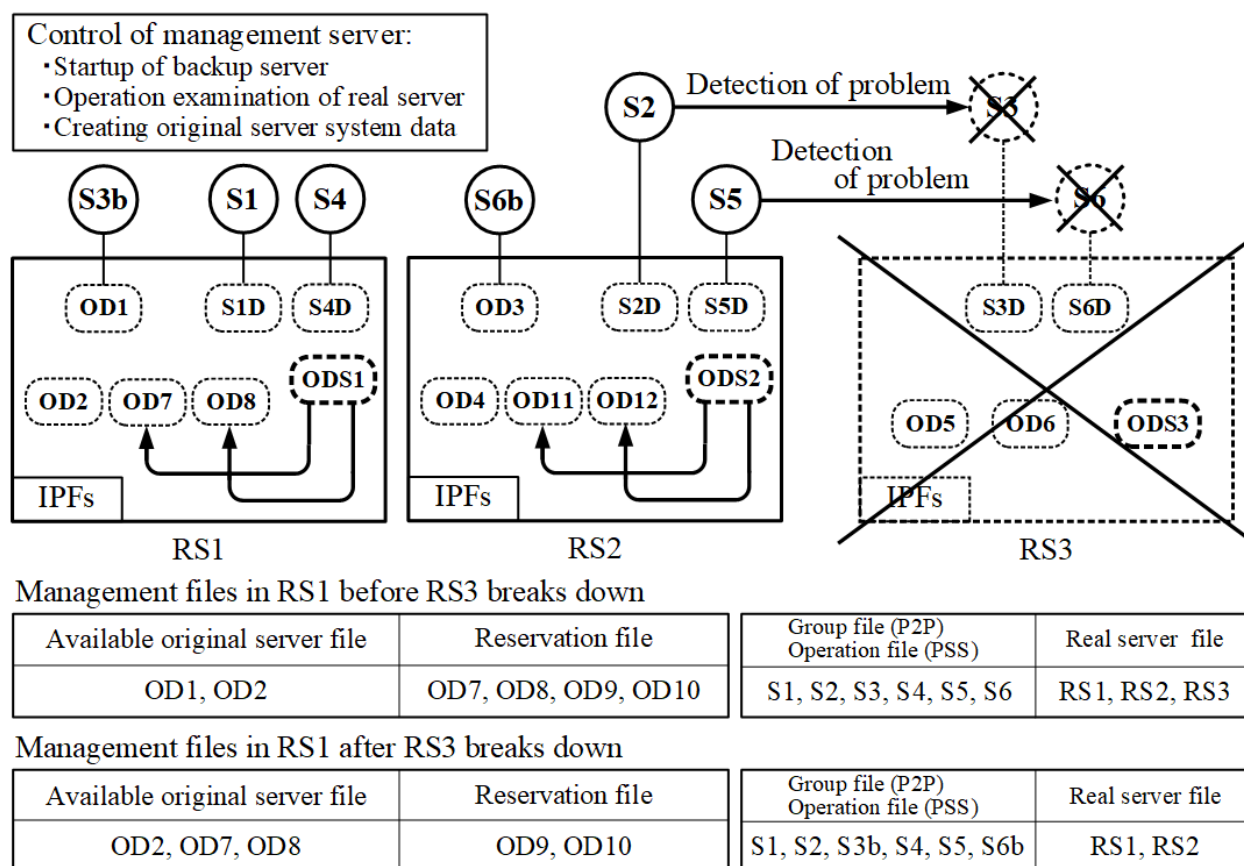


Fig. 5: Operation outline of method for automatic replication of original server.

3. MFSS configuration and performance experiments

3.1. Detailed configuration and operation overview of MFSS

Figure 6 shows the outline of the MFSS, where we can see that S1 through S4 are servers operating the management program of the server system and are connected in a full mesh type by a Secure Shell file system (SSHFS) that can share files via an SSH connection. In the local area of each server, the following areas are set: the master area where the management program creates, deletes, and edits the management file; the backup area where the files with the same contents as the management file in the master area are saved; and the lock area used to prevent file processing conflicts between the management program and the MFSS.

In the S1 through S3 file-sharing configuration shown in the figure, each mount point is allocated on each server and connected to the mount targets of the other servers by the SSHFS. In the MFSS, file monitoring using interrupt processing is performed in the master area and the mount target by using the command “inotify” in UNIX. If the S1 management program edits the management file in the master area, the MFSS examines the difference between the management file saved in the master area and the backup area. After the file editing, control data (Diff.dat) that records the differences is created and saved at each mount point, and the management file edited in the master area is copied to the backup area. Next, S2 and S3 detect Diff.dat created in each mount target, edit the management file in the backup area based on the recorded contents, and then copy the management file to the master area. The same procedure is used to execute file sharing to each server.

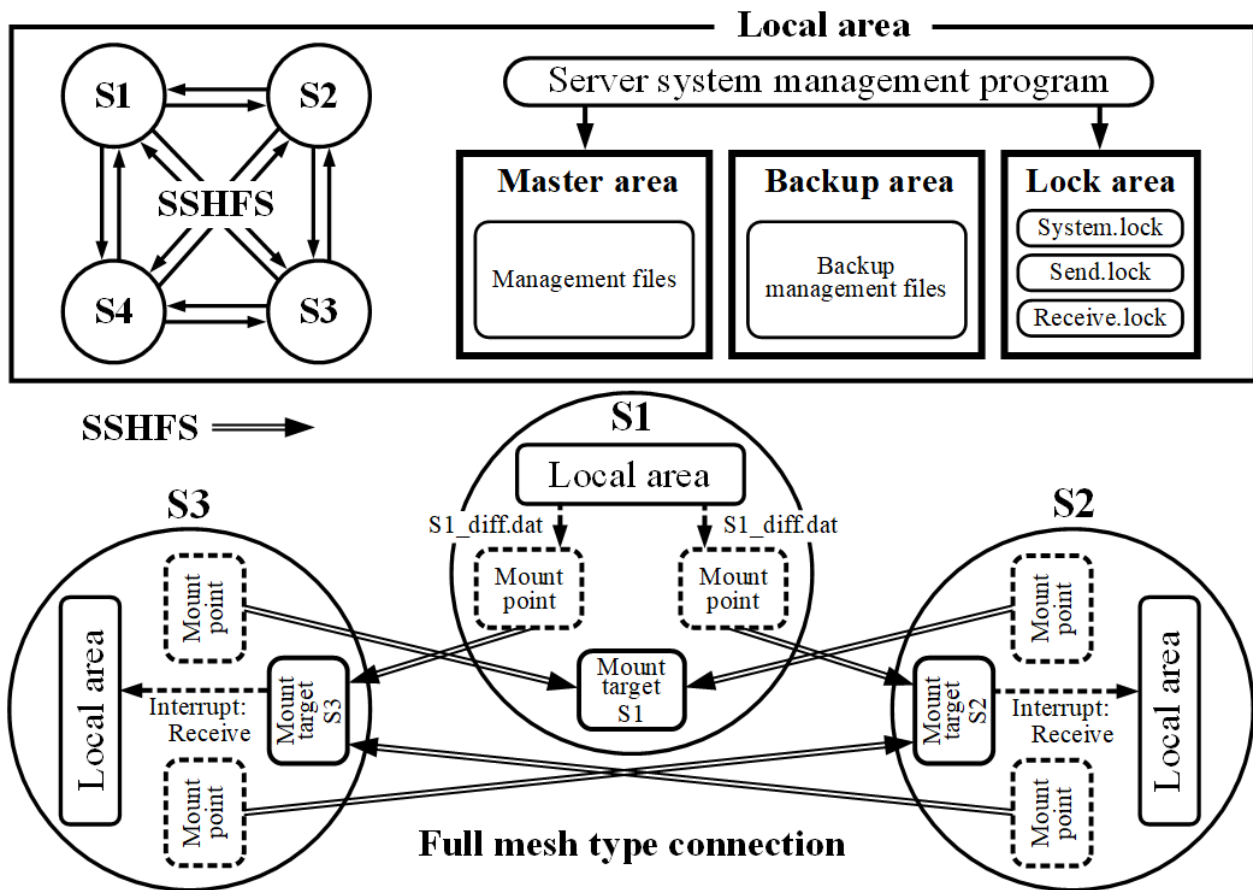


Fig. 6: Detailed configuration and operation overview of MFSS.

3.2. File editing control data (Diff.dat)

Figure 7 shows the contents of Diff.dat used for sharing management files in the MFSS. Because this method deals with the process of creating, deleting, and editing management files, different Diff.dat contents are recorded for each process. For example, when a management file is created, “Directory path for created file, created file name, Create” is recorded on the first line, and the contents of the created file are recorded on the second and subsequent lines. When the management file is deleted, “Directory path for deleted file, deleted file name, Delete” is recorded. When the management file is edited, the first line is “Directory path for edited file, edited file name, Modify” and the results of comparing the edited file in the master area with the unedited file in the backup area by using the command “diff” in UNIX are recorded to subsequent lines. In this case, the server that received Diff.dat uses the command “patch” in UNIX to apply the difference in contents to the corresponding management file.

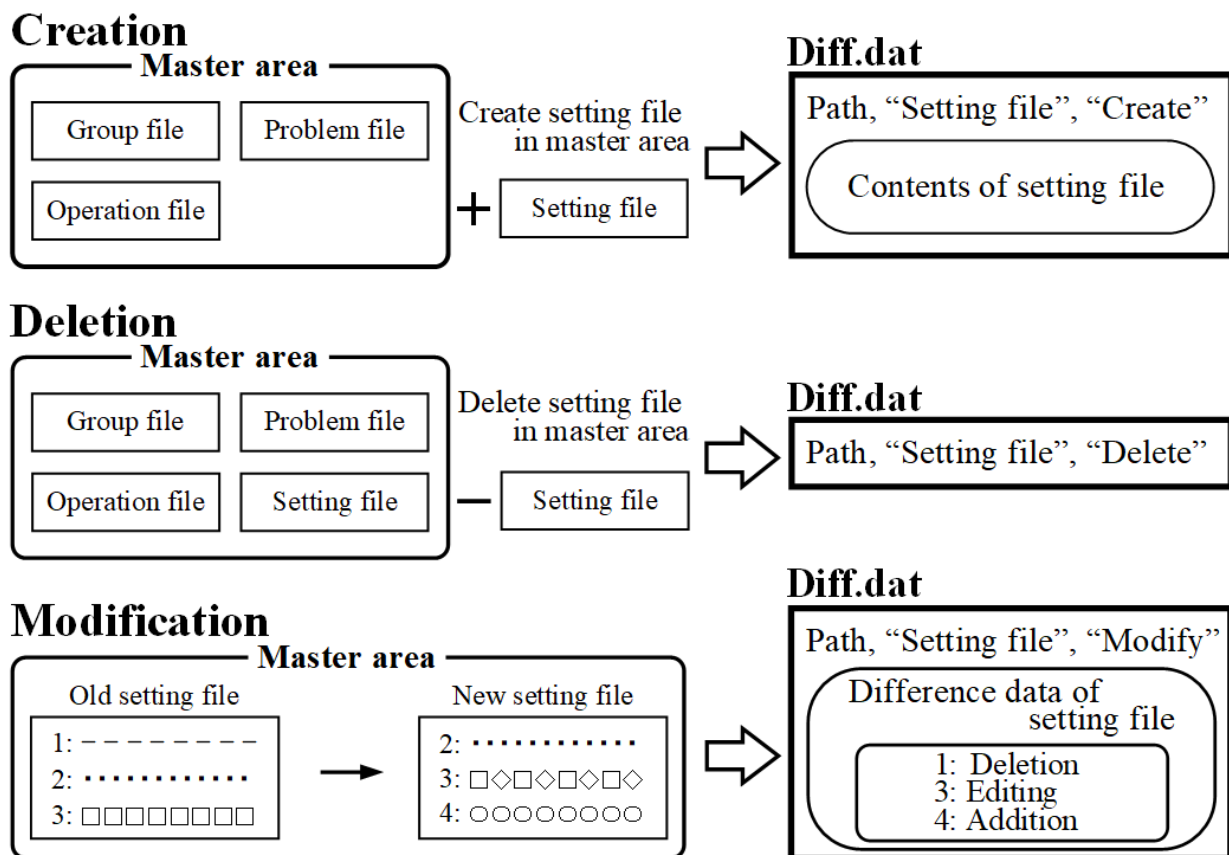


Fig. 7: File editing control data structure.

3.3. Prioritized file locking system and detailed behavior when editing management files

Figure 8 shows the method used to prevent malfunctions when editing management files. In the proposed method, the management and MFSS reception programs access the management file. However, since a malfunction may occur if the management file is edited at the same time, we adopt a prioritized file locking system, as shown in Fig. 8. The management program has a higher priority than the MFSS reception program, and the lock function operates based on these priorities when editing the management file. Note that each program is configured with a lock file for locking the management file.

The lock files created in the locked area shown in Fig. 6 are system.lock for the management program and receive.lock for the MFSS reception program. Taking the MFSS reception program as an example, the operation of the prioritized file-locking system is shown. When Diff.dat is received, receive.lock is created if all lock files do not exist in the lock area. After that, the existence of system.lock, which has the higher priority, is determined. If it does not exist, the MFSS reception program edits the management file. If it exists, receive.lock is deleted, and the program returns to the lock file confirmation status. This prevents simultaneous editing by the management and receiving programs.

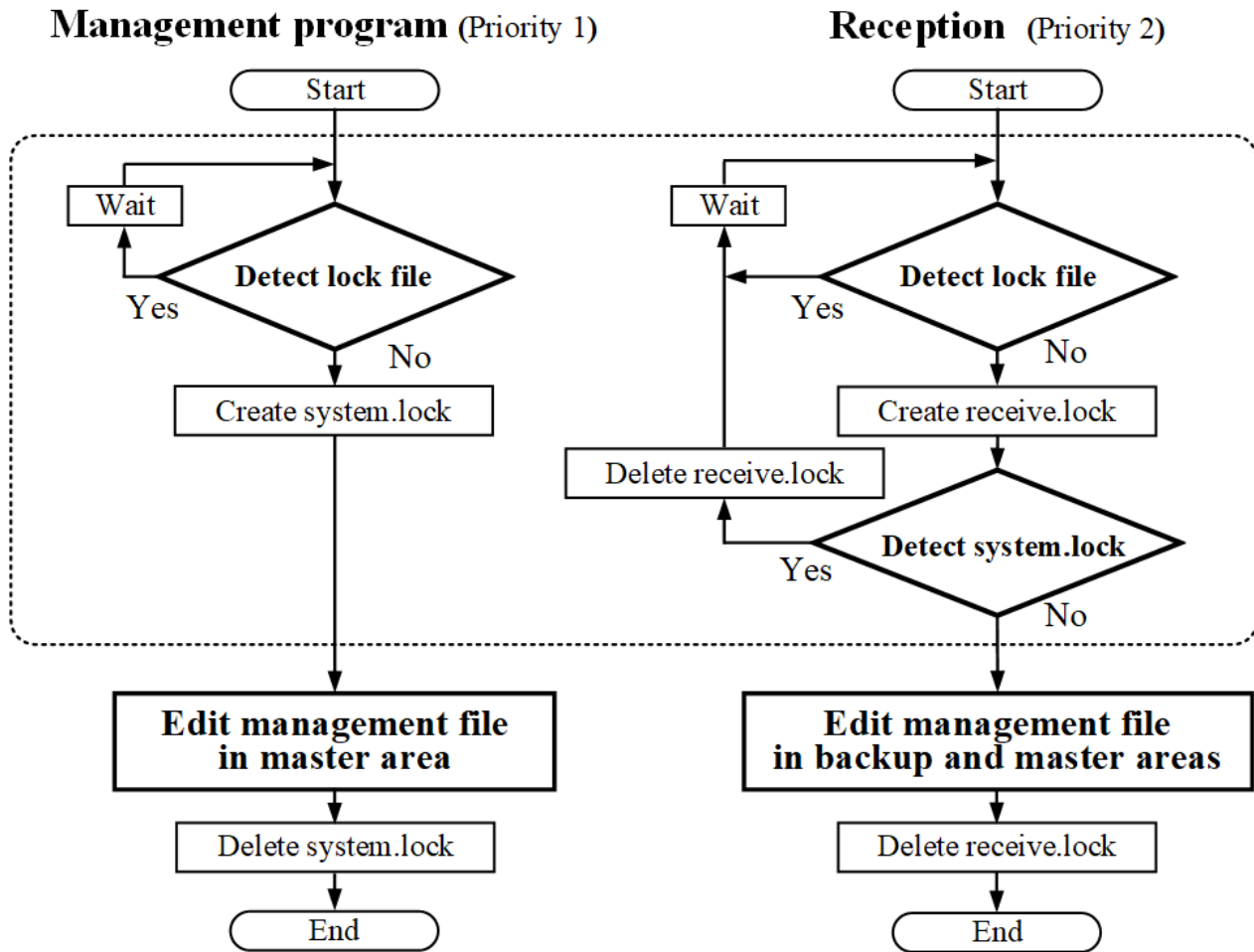


Fig. 8: Prioritized file locking system.

Figure 9 shows the detailed operations made possible by editing the management file along with Diff.dat transmission processing. When the management program edits the management files in the master area, differences with the files stored in the backup area will result, prompting the creation of Diff.dat. The connection status of the mount point is examined, and if a connection exists, Diff.dat is saved to that mount point. However, it is not saved if no connection exists. This processing is performed in parallel for all servers. Next, the edited management file in the master area is copied to the backup area. Note that when the MFSS reception program receives Diff.dat, it edits the management file in the backup area before editing the management file in the master area. This process ensures the management file in the master and backup areas are the same, thus making it possible to prevent erroneous transmission of the received Diff.dat.

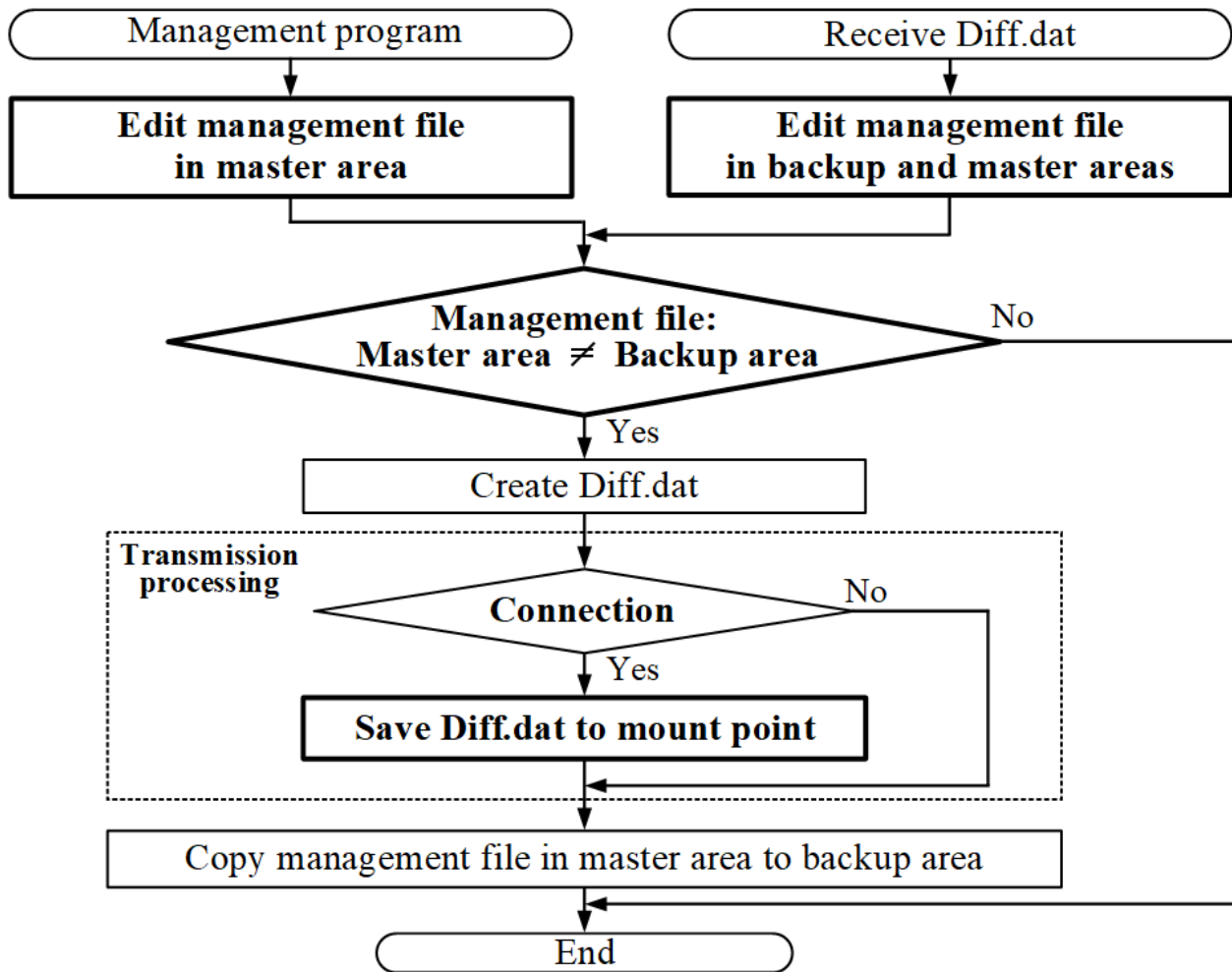


Fig. 9: Detailed operation by management file editing and Diff.dat transmission processing.

3.4. Processing during server failure and recovery

In Figure 10, which shows the processing at the time of server failure and recovery in the MFSS, we see a case where S2 failed while editing the management file in S1. Note that, in S1, the management file is edited by the server system management program, after which Diff.dat is created and then saved to the mount point for the connected server. In the figure, Diff.dat is only sent to S3 because S2 is in a failed state. Because S3 is operating normally, the Diff.dat sent to the S3 mount target is received by interrupt processing, and the management file is edited based on the contents of Diff.dat. The connection with S2 is automatically dropped by the SSHFS settings.

When S2 recovers, an SSHFS connection is performed to the mount target of each running server, which allows the connection to S2 from the server that established the link to be controlled. In S2, this processing establishes a full mesh connection. In addition, since the management file may be updated by another server during the failure, S2 acquires the latest management file by copying the management file saved in the local area of the server specified by the management program for the server management system.

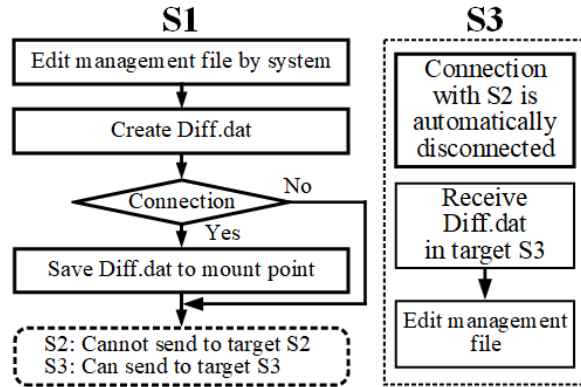
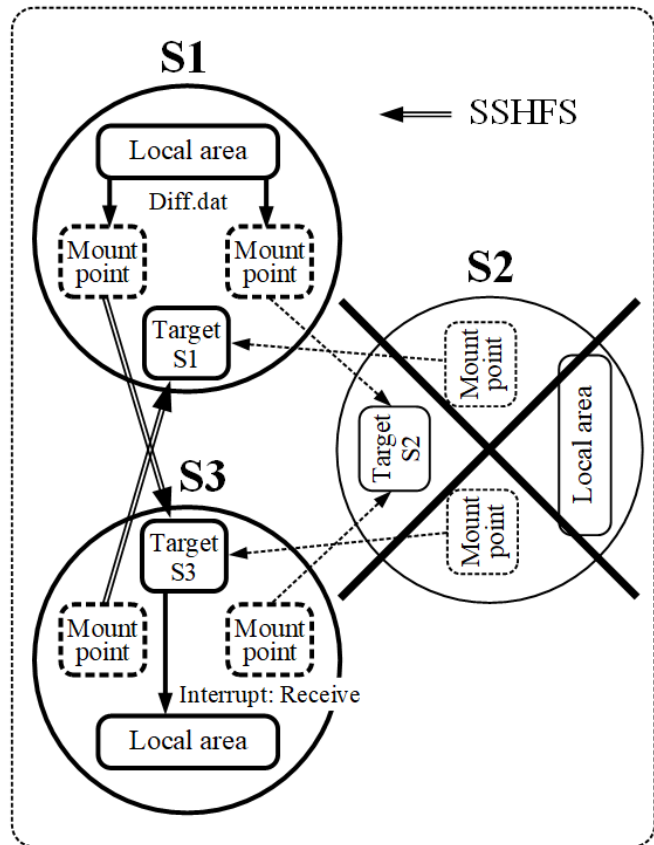
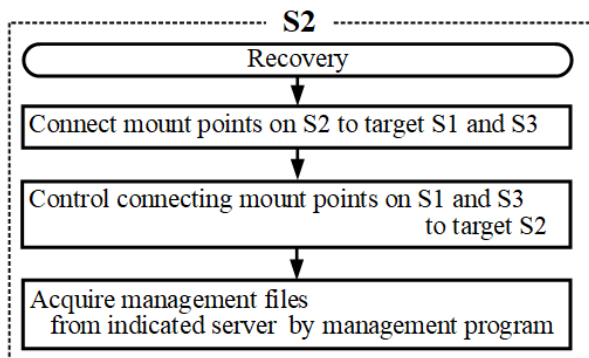
Problem countermeasure**Recovery operation****Fig. 10: Processing during server failure and recovery.****3.5. File synchronization time measurement system and fundamental examination**

Figure 11 shows the time measurement system for examining the synchronization time of management files and their measurement operation flow in the MFSS. As shown in Fig. 11(a), this system consists of three real servers (RS1 through RS3) and six virtual servers (S1 through S6), which are created on the real servers. Figure 11(b) shows the specifications of this system. Because each real server creates multiple virtual servers on which the file-sharing system operates, the physical memory size is 8,192 MB, and a solid-state drive (SSD) is used for storage. The virtual server sets a single-core CPU and a memory size of 2,048 MB, and SSHFS and Secure Copy Protocol (SCP) are adopted as file transfer software to compare the file transfer speeds in the MFSS.

CentOS (Linux), which is often used as a server operating system (OS), was adopted for all real and virtual servers because it operates in a Controlled Unclassified Information (CUI) environment and facilitates efficient memory use. Figure 11(c) shows the operation flow for examining the synchronization time using the SSHFS as an example. In this experiment, the management file is edited on S1 and the MFSS detects file edits in order to create Diff.dat, which is then saved at each mount point connected to the S2 through S6 mount targets. After detecting Diff.dat in their mount targets, S2 through S6 edit the management file in the local area based on the contents (Modification/Creation/Deletion). Here, the time from when the processing for the management file is detected in S1 to the time when the processing is reflected in S2 through S6 is defined as the synchronization time.

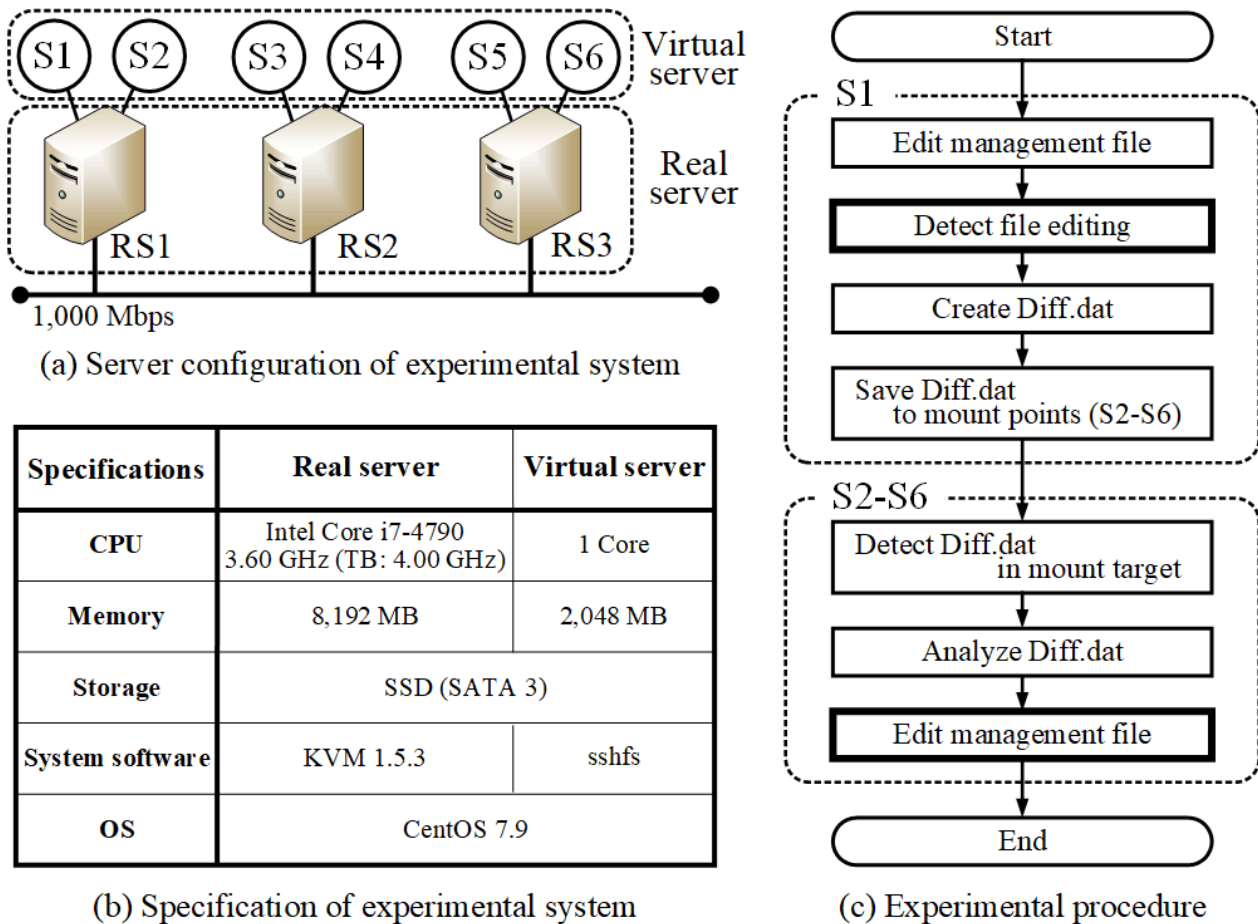


Fig. 11: File synchronization time measurement system and measurement operation flow.

Figure 12 shows the file synchronization time in the MFSS. Here, the server executing the MFSS records the system operating status and time as a system log file. The synchronization time is measured from the log file. Figure 12(a) shows the contents of the log file recorded by the MFSS in S1 and S5. S1 detects the editing of the setting file at 14:28:52.301, creates Diff.dat, and then saves Diff.dat at the S2 through S6 mount points at 14:28:52.315. S5 detects Diff.dat at 14:28:52.327 and then, based on the contents, completes the editing of the setting file at 14:28:52.353. In this case, the synchronization time is 0.052 s.

Next, 10 experiments involving the editing, creation, and deletion of files using SSHFS or SCP as file transfer software in the MFSS are performed and the average file synchronization time for each server is shown in Fig. 12(b). The values in parentheses indicate the coefficient of variation in the 10 measurements, and the longest synchronization time of those measured from S2 through S6 is defined as the system synchronization time. The figure also shows the synchronization time when creating and deleting a 100 KB management file. In the case of file editing, SSHFS and SCP are synchronized in 0.052 and 0.276 s, respectively, while for file creation, SSHFS and SCP are synchronized in 0.067 s and 0.279 s, respectively. For file deletion, SSHFS and SCP are synchronized in 0.051 and 0.253 s, respectively. Because each coefficient of variation is low, the Diff.dat transfer time is stable, and it is thought that the SSHFS and SCP differences occurred because SSHFS is always connected between each server, while SCP connections are only made as needed.

Transmission server: S1	Reception server: S5
<p>⋮ ↙ Detect file editing</p> <p>14:28:52.301 Detection: Modification of setting file</p> <p>14:28:52.315 Transmission: Diff.dat (Modify)</p> <p>⋮ ↙ Send Diff.dat to S2-S6</p>	<p>⋮ ↙ Receive Diff.dat</p> <p>14:28:52.327 Detection: Receive Diff.dat</p> <p>14:28:52.353 Success: Modification of setting file</p> <p>⋮ ↙ Complete file editing</p>

(a) Experimental log file

S1: File editing	Average synchronization time (s) (Coefficient of variation)					
	File modification (1 line)		File creation (100 KB)		File deletion (100 KB)	
	SSHFS	SCP	SSHFS	SCP	SSHFS	SCP
S2	0.052 (0.103)	0.275 (0.103)	0.055 (0.195)	0.279 (0.195)	0.048 (0.219)	0.248 (0.219)
S3	0.048 (0.086)	0.265 (0.086)	0.043 (0.025)	0.269 (0.025)	0.038 (0.017)	0.231 (0.017)
S4	0.052 (0.143)	0.276 (0.143)	0.066 (0.238)	0.269 (0.238)	0.050 (0.275)	0.253 (0.275)
S5	0.052 (0.096)	0.275 (0.096)	0.055 (0.192)	0.277 (0.192)	0.048 (0.219)	0.248 (0.219)
S6	0.052 (0.120)	0.275 (0.120)	0.067 (0.233)	0.270 (0.233)	0.051 (0.263)	0.250 (0.263)
Synchronization for system	0.052	0.276	0.067	0.279	0.051	0.253

(b) Synchronization time

Fig. 12: MFSS log file and file sharing times.

4. Operation experiment in two types of server management systems

4.1. Experimental system configuration

Figure 13 shows the server configuration of the experimental system combining RDBSS and PSS that was developed for operational verification of the MFSS. The system consists of three real servers (RS1, RS2, and RS3), a client, a load balancer, a 1000BASE-T switching hub, File Server 1 (which is required for performing the Network File System (NFS) connection with real and virtual servers), and File Server 2 (which is required to construct a server redundancy configuration). This experimental system, which can perform two types of our proposed MFSS and conventional system using File Server 1, is equipped with the RDBSS and the PSS explained previously. In the conventional method, management files and each virtual server system data are stored on File Server 1, which the management server uses via NFS connections. Each real server can create two mail servers (M1 and M2), two web servers (W1 and W2), and two FTP servers (F1 and F2) as virtual servers to provide mail, web, and FTP services to clients. Furthermore, each virtual server is connected to File Server 2 via the NFS in order to construct a redundant configuration. In the experimental system, the operating state, along with the time, is recorded in the log file.

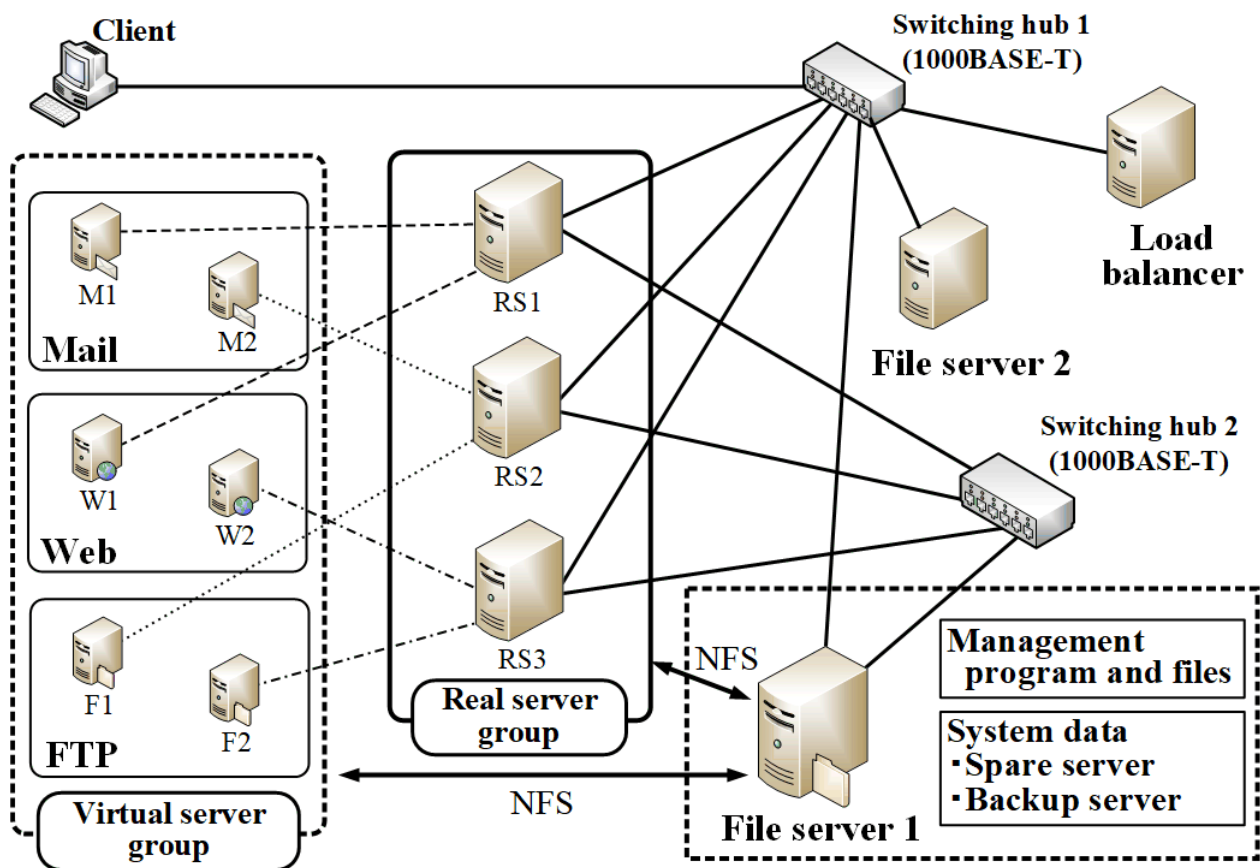


Fig. 13: Experimental system configuration.

4.2. Experimental system specifications and characteristics

Table 1 shows the specifications and characteristics of the real server, File Server 1, the virtual server, and the load balancer of the experimental system. Because the real server creates a virtual server, the memory is set to 8,192 MB and operates in a CUI environment in order to facilitate effective memory use. The postfix server program that processes the Simple Mail Transfer Protocol (SMTP) is installed in the virtual mail server, the Apache HyperText Transfer Protocol (HTTPd) server program is installed in the virtual web server, and the Very Secure FTP daemon (VSFTPD) server program that processes FTP is installed in the virtual FTP server. In addition, the specifications of File Server 1 are the same as the real server, and CentOS is installed on all servers and the load balancer. Virtualization is accomplished with

Kernel-based Virtual Machine (KVM) software running on the real servers. The load balancer uses a software processing method, and HAProxy is installed.

The characteristics of each server, which are provided as the average values of the results of 10 experiments, are given as basic data. The average times in Table 1 are given by the command “time” in UNIX. Note that, in the virtual server, the startup time from the suspended state is considerably shorter than the startup time from the normal state. Therefore, the virtual server always waits in the suspended state. In addition, in the startup time from the suspended state, there is a difference in the startup time between using the file server (NFS) and the MFSS, and the interval time for RDBSS monitoring or PSS examination is 10 seconds.

Table 1: Experimental system specifications and characteristics.

Specifications	Real server (RS1, RS2, RS3)	File server 1	Specifications	Load balancer	
			CPU	Intel Core 2 Quad Q9400 2.66 GHz	
CPU	Intel Core i7-4790 3.60 GHz (TB: 4.00 GHz)		Memory	4,096 MB	
Memory	8,192 MB		System software	HAProxy	
Storage	SSD (SATA 3)		OS	CentOS 7.9	
Virtualization software	KVM 1.5.3		Characteristics	Real server	
OS	CentOS 7.9		Start time (s)	29.41	
			Time to activate suspended server (s)	3.98	
Specifications	Virtual server		Characteristics	Virtual server	
CPU	1 Core			MFSS	NFS
Memory	2,048 MB		Start time (s)	11.69	18.25
System software	postfix, Apache, vsftpd, sshfs		Restart time (s)	14.56	14.52
OS	CentOS 7.9		Time to activate suspended server (s)	1.27	3.98

4.3. Operation comparison between proposed and conventional methods

Table 2 shows the results of operation experiments with the RDBSS and the PSS in the proposed system and the conventional system. In the RDBSS, a service problem (St) and a network problem (Nt) are reproduced for one target server, and the server function recovery times and the access disconnection time in the client are measured. Here, St is a problem caused by stopping the service program and is recovered by restarting the server, while Nt is a problem that occurs by shutting down the server. Each problem is reproduced at the midpoint (5.00 s after startup) of the interval time. The access disconnection time in the client indicates the time until the access by the client is recovered from the server access disconnection state. In the PSS, the time required to change each server configuration and the power consumption in each server configuration are measured. The measurement time in both systems is the average value of the results of 10 experiments, and the values in parentheses indicate the coefficient of variation. The SHW3A watt-hour meter manufactured by System Artware is used to measure the power consumption, and each time is measured from the system log file.

In the RDBSS, the proposed (MFSS) system shortens the server function recovery time by approximately 2 s compared to the conventional (NFS) system in the St and Nt problems. As shown in Table 1, the server function recovery time is affected by the difference in startup time of the virtual server to recover the functions of the problem server. The access disconnection time in the client shows a similar trend. In the PSS, there is an approximately 5 s difference in shift time from the moderate to light load conditions between the MFSS and the NFS, which is caused by the difference between the local server and NFS connection creation processes. This difference occurs because the creation process for two spare servers is executed as shown in Fig. 2. When returning from the light to moderate load condition, the results are almost the same. Based on the above results, the MFSS can be adopted in the RDBSS and the PSS.

Table 2: Operation comparison between proposed (MFSS) and conventional (file server) methods.

RDBSS					
Problem list		Measured time (s)			
		MFSS		NFS	
		Server function (Coefficient of variation)	Access disconnection (Coefficient of variation)	Server function (Coefficient of variation)	Access disconnection (Coefficient of variation)
Service program	St	5.58 (0.006)	11.28 (0.002)	7.67 (0.018)	13.33 (0.001)
Network	Nt	4.21 (0.007)	11.38 (0.027)	6.40 (0.006)	14.13 (0.029)

PSS			
Load condition (All service groups)	Moderate	Light (PS1)	Light (PS2)
Power consumption (W) (Target: 3 real servers)	89.02	63.84	35.65
Load condition	Required time (s) (Coefficient of variation)		
	MFSS		NFS
Moderate → Light (PS1) (All service groups)	3.71 (0.044)		9.26 (0.017)
Light (PS1) → Moderate (All service groups)	8.75 (0.028)		8.98 (0.174)

5. Conclusions

Herein, we discussed our proposed management file sharing system (MFSS), which can share management files without using a file server and is specialized for server management systems. The detailed construction and operation outline of the MFSS and its operation at the time of server failure and recovery, along with its performance, were also shown. To verify the operation of our proposed MFSS, a ring-type dynamic backup server system (RDBSS) and a power-saving server management system (PSS), which do not require a dedicated management server and utilize different file-sharing timings, were adopted. To facilitate comparisons of the management file-sharing processes, we adopted a conventional method using a file server.

We then constructed an experimental server system that can operate with both our proposed and conventional methods and showed its configuration and specifications. In the RDBSS, the recovery time of the server function was measured, while in the PSS, the time required for shifting the server configuration required to realize power saving was measured. Next, the results of the proposed and conventional methods were compared. In the RDBSS and the PSS, it was shown that the proposed method provided higher performance than the conventional method. These results indicate that the MFSS can be adopted for multiple types of server management systems with different management file-sharing timings.

The MFSS realizes file sharing by distributing the differences of edited management files to each server. Therefore, for systems inside a data center, the MFSS can be applied to hundreds of servers.

<Reference>

- [1] A. Alnoman, "Delay-aware Scheduling Scheme for Ubiquitous IoT Applications in Edge Computing," IEEE Conference Proc., Vol. 2021, No. ISNCC, pp. 1-4, Nov. 2021.
- [2] J. Cui, B. Li, H. Zhong, G. Min, and L. Liu, "A Practical and Efficient Bidirectional Access Control Scheme for Cloud-Edge Data Sharing," IEEE Trans. on Parallel and Distributed Systems, Vol. 33, No. 2, pp. 476-488, Feb. 2022.
- [3] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, and J. Zhao, "Maximizing User Service Satisfaction for Delay-Sensitive IoT Applications in Edge Computing," IEEE Trans. on Parallel and Distributed Systems, Vol. 33, No. 5, pp. 1199-1212, May 2022.
- [4] S. Yue, J. Ren, N. Qiao, Y. Zhang, H. Jiang, Y. Zhang and Y. Yang, "TODG: Distributed Task Offloading With Delay Guarantees for Edge Computing," IEEE Trans. on Parallel and Distributed Systems, Vol. 33, No. 7, pp. 1650-1665, July 2022.
- [5] P. Malathi and S. Suganthidevi, "Comparative Study and Secure Data Deduplication techniques for Cloud Computing storage," IEEE Conference Proc., Vol. 2021, No. ICSSES, pp. 1-5, Sep. 2021.
- [6] A. Yehezkel, E. Elyashiv, and S. Barkai, "Using CFN for Uniform Sampling of Cloud-Native Datacenters," IEEE Conference Proc., Vol. 2022, No. CCNC, pp. 967-968, Jan. 2022.
- [7] O. Ivanchenko, V. Kharchenko, B. Moroz, Y. Ponochoynyi, and L. Degtyareva, "Availability Assessment of a Cloud Server System: Comparing Markov and Semi-Markov Models," IEEE Conference Proc., Vol. 2021, No. IDAACS, pp. 1-6, Sep. 2021.
- [8] A. Zhao, Z. Liu, J. Pan, and M. Liang, "A Novel Addressing and Routing Architecture for Cloud-Service Datacenter Networks," IEEE Trans. on Services Computing, Vol. 15, No. 1, pp. 414-428, Jan. 2022.
- [9] J. Masoudi, B. Barzegar, and H. Motameni, "Energy-Aware Virtual Machine Allocation in DVFS-Enabled Cloud Data Centers," IEEE Access, Vol. 10, pp. 3617-3630, Jan. 2022.
- [10] Z. Chen, J. Hu, G. Min, C. Luo, T. El-Ghazawi, "Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement," IEEE Trans. on Parallel and Distributed Systems, Vol. 33, No. 8, pp. 1911-1923, Aug. 2022.
- [11] S. Zhang, S. Han, B. Zheng, K. Han, and E. Pang, "Group Key Management Protocol for File Sharing on Cloud Storage," IEEE Access, Vol. 8, pp. 123614-123622, Jan. 2020.
- [12] B. S. Rawal and G. Manogaran, "Implementation of a secure multi-cloud storage framework with next-generation cryptosystems and split-protocol," IEEE Conference Proc., Vol. 2021, No. ISNCC, pp. 1-6, Oct. 2021.
- [13] S. Lin, J. Yin, Q. Pei, L. Wang, and Z. Wang, "A Nested Incentive Scheme for Distributed File Sharing Systems," IEEE Conference Proc., Vol. 2021, No. SmartIoT, pp. 60-65, Aug. 2021.
- [14] H. S. Nguyen, D. N. Nguyen, and S. Sugawara, "A Dynamic-Clustering Backup Scheme for High-Availability

- Distributed File Sharing Systems,” IEICE Transactions on Communications (Web), Vol. E102.B, No. 3, pp. 545-556(J-STAGE), Mar. 2019.
- [15] M. Kitamura, “Configuring a Low-cost, Power-saving Multiple Server Backup System: Experimental Results,” IEICE Trans. Commun., Vol. E95-B, No. 1, pp. 189-197, Jan. 2012.
 - [16] A. A. Khatami, Y. Purwanto, and M. F. Ruriawan, “High Availability Storage Server with Kubernetes,” IEEE Conference Proc., Vol. 2020, No. ICITSI, pp. 74-78, Oct. 2020.
 - [17] N. Georgouloupoulos, A. Hatzopoulos, K. Karamitsios, I. M. Tabakis, K. Kotrotsios, and A. I. Metsai, “Investigation and Simulation of Hardware Errors in Kernel Logs of Linux-based Server Systems,” IEEE Conference Proc., Vol. 2021, No. SEEDA-CECNSM, pp. 1-7, Sep. 2021.
 - [18] T. Ono and K. Ueda, “Data Redundancy Dynamic Control Method for High Availability Distributed Clusters,” ACM Proceedings, No. SoICT 2018, pp. 185-191, Dec. 2018.
 - [19] M. Kitamura, Y. Shimizu, and K. Tani, “Development of a Power-saving, High-availability Server System by Compound Operation of a Multiple-server Backup System and Power-saving Server System,” JISSJ, Vol. 14, No. 2, pp. 79-88, Mar. 2019.
 - [20] M. Kitamura, Y. Shimizu, and K. Tani, “Development and Operation Experiment of a Power-saving, High-availability Server System by Compound Operation of a Power-saving Server System and a Multiple-server Backup System,” JISSJ, Vol. 15, No. 2, pp. 34-54, Mar. 2020.
 - [21] M. Kitamura, Y. Udagawa, H. Nakagome, and Y. Shimizu, “Development of a Server Management System Incorporating a Peer-to-Peer Method for Constructing a High-availability Server System,” JISSJ, Vol. 13, No. 2, pp. 14-40, Mar. 2018.
 - [22] M. Kitamura and K. Tani, “Development of File Management System for a Peer-to-Peer Method Server Management System,” JISSJ, Vol. 16, No. 1, pp. 1-16, Sep. 2020.
 - [23] M. Kitamura, T. Takeshita, and T. Okazaki, “Development of Ring-type Dynamic Backup Server System not requiring Dedicated Management Server,” JISSJ, Vol. 18, No. 1, pp. 22-42, Sep. 2022.
 - [24] M. Kitamura, K. Tani, T. Takeshita, and T. Yamaguchi, “Development of Server Function Recovery System for Peer-to-Peer Method Server Management System adopted for Virtual Server System,” JISSJ, Vol. 17, No. 1, pp. 27-34, Sep. 2021.