［Original Paper］

Development of File Management System for a Peer-to-Peer Method Server Management System

Mitsuyoshi KITAMURA† and Koki TANI‡

† Graduate School of Engineering, Tokyo Polytechnic University
‡ Quest Co., Ltd.

Abstract

Herein, a peer-to-peer (P2P) method server management system that does not require a dedicated management server is proposed as a way to construct a server system with high availability. In the P2P method, each management server uses a file server to share the management files required to manage their target servers and uses virtual server system data to recover server functions after failures. Hence, the P2P method is highly dependent on the file server when managing the server system, and the file server performance determines the number of target servers that can be managed. With these points in mind, we propose two systems (a synchronous editing method that provides a file-sharing system and an original server method that provides a virtual server startup method) that do not require the use of a file server. We also outline their configurations and performance levels and then discuss solutions to the problems that are likely to occur during the introduction of those systems. Furthermore, we report on the construction of an experimental server system that can operate using both our proposed method and a conventional method, with which the server function recovery times of both methods were measured and compared. The results of our experiments show that, using our proposed method, it is possible to construct a P2P method server management system that does not require a file server, and that the server performance is improved by the improved recovery processing of server functions after failures. Furthermore, customers can receive higher quality services because adopting our proposed method results in an increase in server system availability.

1. Introduction

In the Society 5.0 vision advocated by the Cabinet Office of Japan, it is expected that everyone and everything will soon become interconnected via the Internet of Things (IoT), which will allow a wide variety of knowledge and information to be analyzed by artificial intelligence (AI), after which the required information can be shared among appropriately authorized users at any time [1]. To realize these goals, networks with high speed, high capacity, and high reliability must be realized by fifth-generation (5G) mobile communications systems [2], [3]. This means that the ability to provide appropriately high levels of Internet services, the ability to analyze big data obtained from IoT devices or other sources, along with servers and their associated roles, will become more critical than ever before.

When constructing server systems, it is necessary to pay adequate consideration to the power-saving features in addition to high speeds and high availability rates [4], which means that stable operation, design optimization, construction, and operation management are fundamental issues [5]. Indeed, several relevant studies related to improving reliability in data centers have already been conducted. In one example, a hierarchical modeling framework that can be used to evaluate data center network reliability and availability was proposed [6]. In another, an integrated method that combines failure occurrence and failure location predictions was developed to enable both automated and operator response actions [7].

In addition, an integrated design that considers cloud service problems such as a single link failure, service failures on data center networks, and data center network placement was discussed in [8]. Meanwhile, in order to solve the data migration problems that result from storage performance mismatches in large-scale storage systems such as data centers, another study introduced data transfer bypass nodes that improve migration performance levels [9], while hardware problems and service abnormalities were evaluated in relation to the security of a server and the network resources of a virtual data center in [10], and a high-availability virtual infrastructure management framework that considers the problem rate of data center devices was reported in [11].

[Original Paper] Received 14 November 2019, revised 17 February 2020, accepted 16 March 2020.

Various other studies have been conducted on file-sharing data redundancy over networks. For example, throughput measurements in the case of sharing files of various sizes in wireless local area network (WLAN) environments [12], the adoption of remote synchronization (rsync) as a backup file synchronization method in a learning management system using Moodle [13], the consideration of secure cloud service file-sharing mechanisms [14], and user data file availability improvement produced by considering efficient resource placement and lookup for distributed file sharing systems [15] have all been investigated.

Since the central unit for providing data center services is its server system, numerous studies on high-availability server systems have been performed. In one example, a method that can significantly reduce service interruption time delays by considering both hardware and software server failures issues was proposed [16]. In another, a dynamic configuration change method for high availability server cluster systems that minimizes the disruptive effects of server additions and deletions on normal processing was discussed [17]. In a separate study [18], a configuration method for high-availability virtual data center server systems that permits the entire memory state of a failed server to be instantly copied and restarted on a hot spare node was reported. Additionally, a method for dynamically controlling data redundancy that uses network anomaly detection technology to increase the reliability of highly available distributed clusters was proposed in [19], while highly available server clusters with greatly improved service reliability and availability that were achieved by the adoption of fencing technology were discussed in [20], and highly available distributed clusters that can dynamically change configurations while maintaining services based on scalability and traffic rates were reported in [21].

Furthermore, as part of our research into high-availability server systems, we have developed and reported on additional issues related to various other systems. For example, failover cluster systems and load balancer cluster systems, which are commonly used to provide server system redundancy, can deal with server failures, but they often entail high costs because such cluster systems are basically applied to servers providing the same service. Therefore, to achieve both low cost and power saving features, a multi-server backup system was proposed in [22]. That study introduced a server management system using server virtualization technology that can back up the functions of several real servers in a single real server.

Other studies involving high-availability server systems focusing on power-saving server systems have also been reported. For example, in consideration of the need for simplifying the control programs used for configuring a high-availability server system with power-saving features, a hybrid server system operation method that can alternate between power-saving and high-availability modes was reported in [23]. Next, in [24] the authors reported on the adoption of a system interface that simplifies the compound control program operations required for switching between the power-saving and high-availability modes, and high availability improvements in the power saving state and performance levels of the system interface were performed and experimentally evaluated in [25].

In recent years, methods that use specific management servers to monitor server systems, including those discussed in [22] - [25], have become widespread. As a result, increasing management loads to match the number of target servers and the need for countermeasures when management server failures occur are ongoing concerns. Therefore, a peer-to-peer (P2P) method server management system that does not require a specific management server to manage a real server system [26], and a P2P method with the enhanced processing needed to recover server functions when managing a virtual server system have been reported previously [27]. In that method, since each target server also functions as a management server, even if the server providing management functions fails, other servers can assume the role and continue to perform management functions.

However, in the abovementioned P2P method [26], [27], a file server is still required because all of the management servers must constantly share management data. In addition, although this method requires the use of a virtual server to recover the functions of the failed server, the management server that will create the virtual server is not selected until recovery operations begin. Furthermore, a file server is also necessary because all management servers need access to the system data of all virtual servers. As a result, this method has the following three problems:

1) If the file server fails, server management cannot be continued.
2) The number of target servers is limited by the file server performance.
3) In cases where it is necessary to recover multiple server functions, access bottlenecks develop at the file server.

To resolve these problems, it is necessary to develop a P2P method server management system that does not use a file server.

With the above points in mind, the present study proposes a synchronous editing method (SEM) that can realize file sharing without using a file server and an original server method that is capable of starting a virtual server, and introduces them both into the P2P method. In the SEM, when the management file in the management server is edited, file editing control data (Diff.dat), which is data used for the file sharing, is simultaneously sent to all the other management servers and the management file is shared. In the original server method, the virtual server system data that can realize the basic functions of each target server, along with the installation procedure file needed to realize those functions, are distributed to each management server. The functions of the failed server are then recovered by combining them.

Herein, we show the operation outlines and performance levels of our methods, along with the changes required for introducing the SEM into the P2P server management system. Furthermore, we report on the construction of an experimental server system that can be operated using both our proposed method and a conventional file server-based method with which the server function recovery times of both methods were measured and compared. The experimental results show that server function recovery times using our proposed method are shorter than those of the conventional method.

The remainder of the present paper is organized as follows. Section 2 describes the operating state that exists when the target server fails while the conventional file server-based method is used as well as problems related to virtual server system data arrangements in situations where the file server is not used. Section 3 describes the SEM problems and the solutions required to realize management file sharing, as well as the file-sharing time measurement results. In addition, the structure and the virtual server startup time when the original server method is as the virtual server startup method are discussed. Section 4 describes the P2P method server management system improvements that are necessary to adopt our proposed system. Section 5 describes the construction and specifications of the experimental system that can operate both the proposed and conventional methods and shows that the proposed method improves the performance levels of server function recovery operations over those of the conventional method. We provide our conclusions in Section 6.

2. Outline of P2P method server management system and the problem of introducing a proposed system

Figure 1 shows the relationship between the system operation and the management file that results if a server fails in the P2P method server management system [26]. S1 through S6 are real servers that act as both management and target servers, and S2b denotes a virtual server used for recovering the server function if S2 breaks down. In the P2P method, since two management servers manage one target server, a server management priority is set for each server. In this figure, S2 is managed by S1 operating with Management Priority 1 (P1) and by S3 operating with Management Priority 2 (P2).

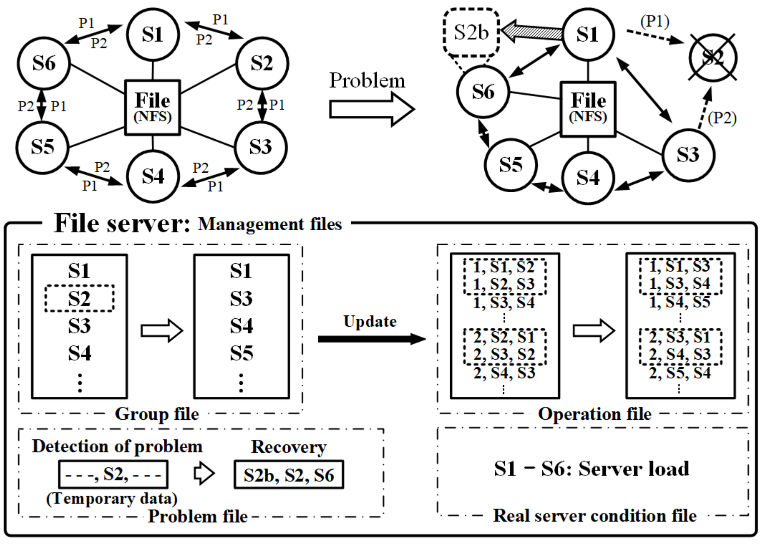


Fig. 1 Configuration of P2P method server management system and management file.

In the conventional system, management files are centrally managed on the file server, and each management server accesses them through a network file system (NFS) connection. Each management file includes the group, operation, problem, and real server condition files. Information on all management servers is recorded in the group file. The operation file, which contains the server management priority, the hostname of the management server, and the hostname of the target server, is automatically constructed based on the group file. If a problem is detected in the target server, status reports on the recovery operation are recorded in the problem file, and the load value of each management server is recorded in the real server condition file.

In this figure, S1 operating with P1 detects a problem in S2 and performs operations to create S2b on S6, which is the server with the lowest load among the management servers, based on the real server condition file. S1 edits the management file in the file server when a failure is detected and reflects the failure status on the system. Here, S1 detects a problem in S2, temporary data (problem server name S2) are recorded in the problem file, and the S2 information is deleted from the group file. Next, the operation file is automatically updated based on the group file. In order to recover the server function, S1 performs operations to create S2b on S6 and rewrites the problem file at the same time. These operations show that the file server plays an important role in the P2P method server management system.

As mentioned above, the P2P method uses a virtual server to recover the server function of the problem server. In that case, the load state of the management server is used to determine which management server will be used to create the virtual server. Figure 2 shows time charts of the server management (P2P) and load examination (Load), in which all management servers perform their processing synchronously. Each management server synchronously performs live monitoring on all target servers in which the operating states of the network and service are examined. If no problems are detected, the management servers remain in the waiting state (monitoring interval time). If a problem is detected in the target server, recovery processing is performed.

In the load examination process, the loads to be measured are memory usage, central processing unit (CPU) utilization, and network usage. Load measurement in the management server is performed twice at regular intervals, and the average value of each load is calculated from those measurements. Each predicted load is calculated based on the average values, which is then normalized to remove imbalances between management servers with different specifications, and the real server condition file is created. Server management and the load examination are performed at the timing shown in the figure. If recovery processing is performed, information from the latest real server condition file created by the load examination and the number of virtual servers created on the management server are used to create the virtual server on the appropriate management server.

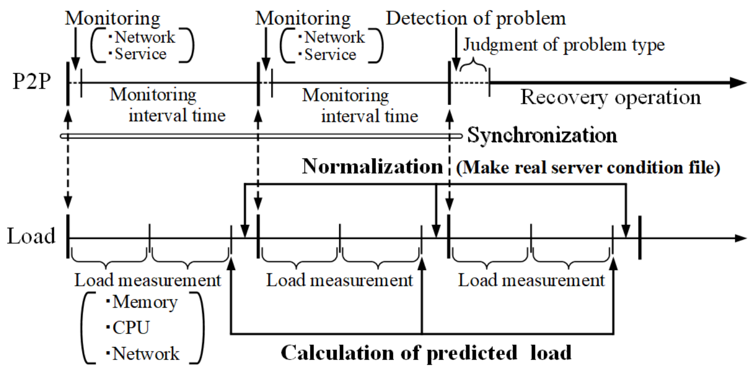


Fig. 2 Operation timing of monitoring for target server and load examination.

Figure 3 shows the problems with the virtual server startup method in a configuration that does not use a file server. In the P2P method server management system, if the target server fails, a virtual server is used to recover its function. The management server that creates the virtual server is determined by considering its load. Therefore, as shown in Fig. 3(a), the virtual server system data are stored on the file server. If the target server fails, the virtual server is created on the management server using the system data obtained through the NFS connection. This means that another problem occurs if the virtual server startup time is delayed in the case of simultaneously managed server failures. In the P2P method server management system, the management server used to create the virtual server is not predetermined, so if no file server is used, as shown in Fig. 3(b), all management servers must have virtual server system data corresponding to all managed servers. This is unrealistic from the viewpoint of the file capacities of each management server.

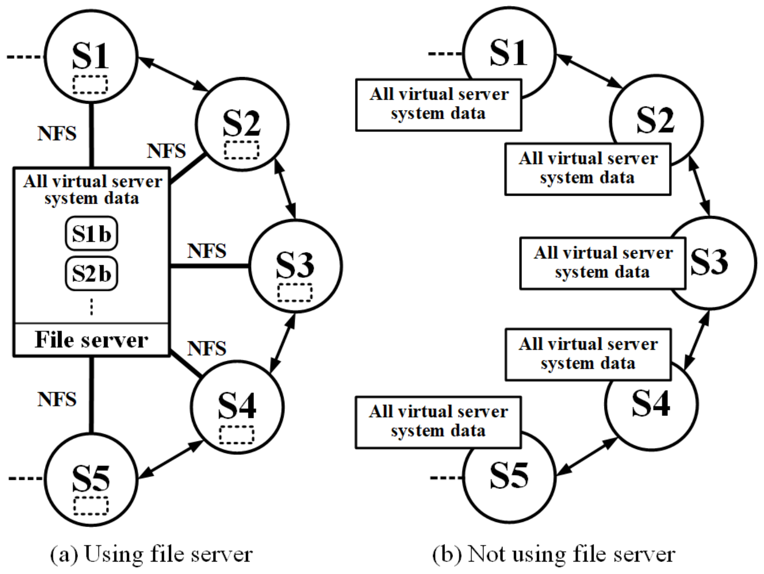

S1
S2
S3
S4
S5
NFS
NFS
NFS
NFS
NFS
All virtual server system data
S1b
S2b
File server
(a) Using file server
All virtual server system data
(b) Not using file server

Fig. 3 Storage locations of virtual server system data.

3. Synchronous editing method of management file and original server method

In this section, we provide details on our proposed SEM, which can achieve file sharing without using a file server, and on our original server-based virtual server startup method. In the SEM, when the management file in the management server is edited, it sends Diff.dat (which is the file sharing data) to other management servers and simultaneously shares the management file. In the original method, the virtual server system data needed for the basic functions of each target server and their installation files are distributed to each management server. When a server fails, the functions are recovered by combining them.

3.1. Synchronous editing method of management file

In order to mirror the management file contents on all management servers without using a file server, the management files are saved on each management server, and it is necessary to reflect updates that occur during system operation synchronously. Although rsync is often used for file sharing, problems occur when the same file is edited simultaneously from multiple computers. Therefore, we propose the use of SEM as a file sharing system.

Figure 4 shows a method for sharing updated files (in this case, the group and problem files) if the target server fails. Information on the running management server is recorded in the group file, and (as shown in the figure), the server determined to be faulty is deleted from the list in that file. This information must be reflected on all management servers. In the SEM, Diff.dat corresponding to the edited part are transmitted and received by each management server, which then edits its stored management file. The Diff.dat contents are “filename, Del, deletion information” when deleting information; “filename, Add, addition information” when adding information; and “filename, Mod, information before change, information after change” when rewriting the contents.

In the figure, S1 operating with P1 and S3 operating with P2 detect the S2 failure. At that point, S1 operating with P1 creates Diff.dat (Group, Del, S2) for the group file and Diff.dat (Problem, Add, AD) for the problem file, and sends them simultaneously to the other management servers. Next, Diff.dat (Problem, Mod, AD, MD) is created for the problem file at the same time as the failed server recovery processing and is sent simultaneously to the other management servers. After receiving Diff.dat, each management server edits its management file. The management files saved on each management server are then shared the same way.

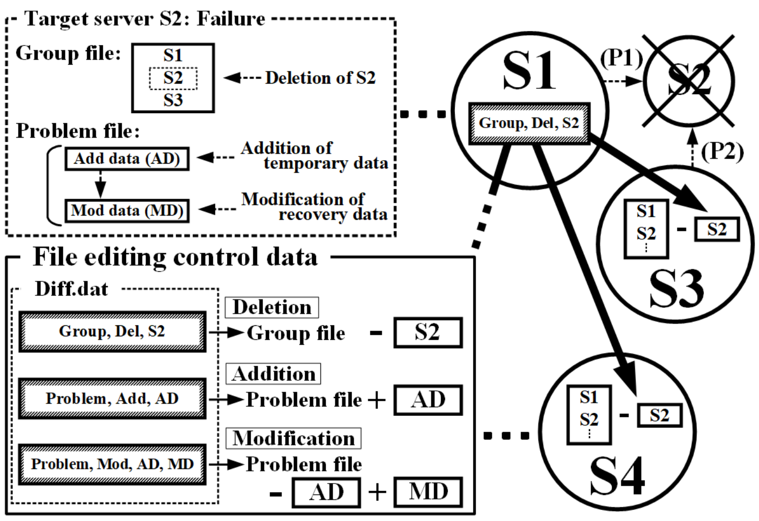


Fig. 4 SEM for management file.

Figure 5 shows the details of the processing in the SEM using the group file as an example. The P2P management program refers to the control program for the P2P method server management system. The SEM monitors the management file editing process. If editing is performed, interrupt processing is executed, and Diff.dat is sent to the other management servers. As shown in Fig. 5(a), there are two types of management file editing processing: processing by the P2P management program and processing by the SEM generated from the Diff.dat reception. Here, since the SEM edits the management file if the Diff.dat is received, a Diff.dat that duplicates the received content is created and sent to the other management servers. As a result, as shown in Fig. 5(b), a group_backup file that mirrors the group file is prepared.

In the initial state of the system, the contents of the group and group_backup files are the same. Here, the P2P management program edits the group file as usual. However, the SEM first edits the group_backup file and then copies its contents to the group file. As a result, in the case of edit by the P2P management program, the contents of the two files are different, but they will remain the same when SEM is used. If the two files are not the same, Diff.dat is created and sent to all the other management servers simultaneously, after which the group file is copied to the group_backup file.

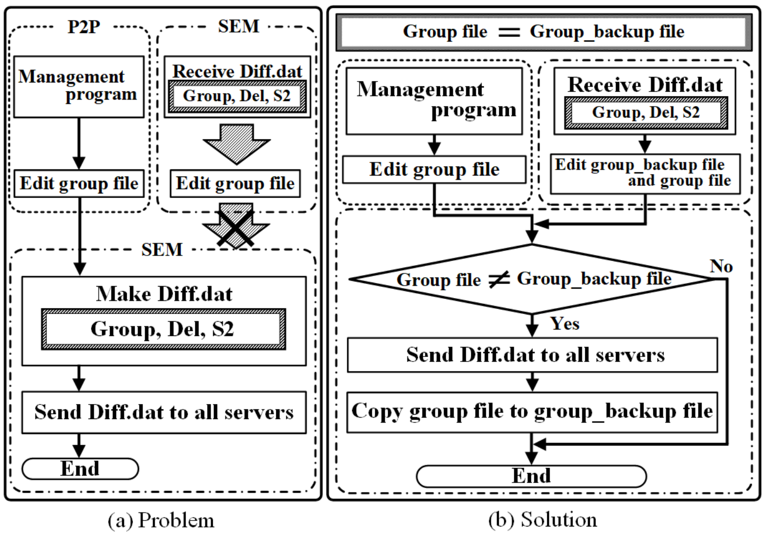
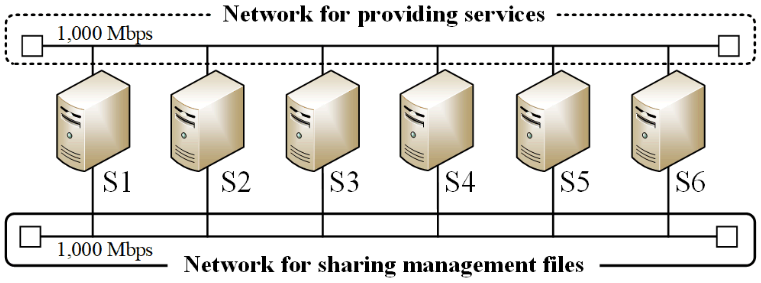


Fig. 5 Details of SEM processing.

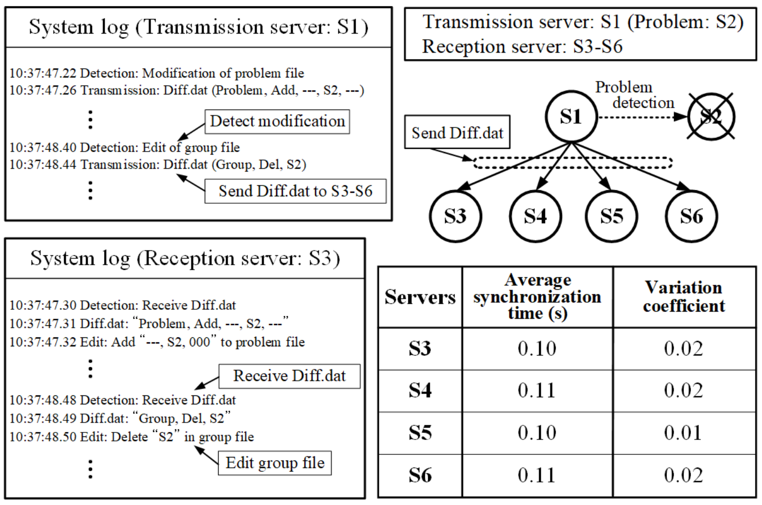
The experimental system shown in Fig. 6 was constructed to examine the management file synchronization time when SEM is used. This system consists of six servers that act as both management and target servers, a network for providing services to clients, and a network for sharing management files. The memory size of the six management servers was set as 8,192 MB, and CentOS (64-bit), which is a popular server operating system (OS), was used. To ensure efficient memory use, each server operates in a character-based user interface (CUI) environment. In this experiment, the management file on a specific management server is edited, and the time passage from that point to the time when the contents are updated in the management file on the other management servers is measured. The measurement time is extracted from the log file recorded by the SEM.


Network for providing services
1,000 Mbps
S1
S2
S3
S4
S5
S6
1,000 Mbps
Network for sharing management files

Specifications	S1	S2	S3	S4	S5	S6
CPU	Intel Core i7-3770 3.40 GHz (TB: 3.90 GHz)					
Memory	8,192 MB					
Storage	SSD (SATA 3)					
System software	postfix, sshd, httpd, vsftpd, squid					
OS	CentOS 7.7 (64-bit)					

Fig. 6 File sharing time measurement system and specifications.

Figure 7 shows the management file synchronization times for each management server. Here, a problem occurs on S2 and is detected by S1. S1 edits the management file and then sends Diff.dat from S3 to S6. This experiment targets the group file synchronization time. In the S1 system log, a group file edit is detected at 10:37:48.40, and Diff.dat is sent at 10:37:48.44. As the receiving server, the S3 system log indicates that Diff.dat was received at 10:37:48.48 and that a group file edit was completed at 10:37:48.50. The synchronization time here is the difference between 10:37:48.40 and 10:37:48.50, which is 0.10 s.



System log (Transmission server: S1)

```
10:37:47.22 Detection: Modification of problem file
10:37:47.26 Transmission: Diff.dat (Problem, Add, ---, S2, ---)
        ⋮
10:37:48.40 Detection: Edit of group file
10:37:48.44 Transmission: Diff.dat (Group, Del, S2)
        ⋮
```

System log (Reception server: S3)

```
10:37:47.30 Detection: Receive Diff.dat
10:37:47.31 Diff.dat: "Problem, Add, ---, S2, ---"
10:37:47.32 Edit: Add "---, S2, 000" to problem file
        ⋮
10:37:48.48 Detection: Receive Diff.dat
10:37:48.49 Diff.dat: "Group, Del, S2"
10:37:48.50 Edit: Delete "S2" in group file
        ⋮
```

Servers	Average synchronization time (s)	Variation coefficient
S3	0.10	0.02
S4	0.11	0.02
S5	0.10	0.01
S6	0.11	0.02

Fig. 7 Log file of SEM and sharing time.

In the figure, these measurement times are the average values of ten experimental results. Examining the results from S3 to S6, we can see that the synchronization time for the system is 0.11 s. In addition, from each variation coefficient, it can be seen that there are almost no fluctuations in the synchronization time of each management server.

From these results, it is clear that SEM file sharing is sufficiently stable and that its dependability and usefulness are high.

3.2. Our original server method

As shown in Fig. 3, if the file server is not used, it is necessary to store virtual server system data corresponding to all target servers on all of the management servers. However, we propose an original server method that can solve this problem. In the P2P method, only one virtual server is created on one management server, even if half the system servers fail [26]. Therefore, in our proposed method, two original servers, which install all the service programs, are set on each management server. In addition, the installation procedure file that is needed to enable the original server to recover the failed server function is prepared. This procedure file contains the configuration file installation program. In addition, the installation procedure of the configuration file for the service program provided by the failed server, along with the program startup settings, are also recorded in this procedure file.

Figure 8 shows an operational outline of our original server method when recovering the failed S2 function. First, S1 detects a problem in S2 and takes control of S6. At that point, the original server is created using the original server system data stored in S6. Next, the "Installation procedure file (S2b)" is transferred and installed in order to allow the original server to be operated as S2b. Using this procedure, the original server can operate as the virtual server that recovers the S2 server function. Because the installation procedure file is small in size, the installation procedure files for all target servers can be stored on all management servers.

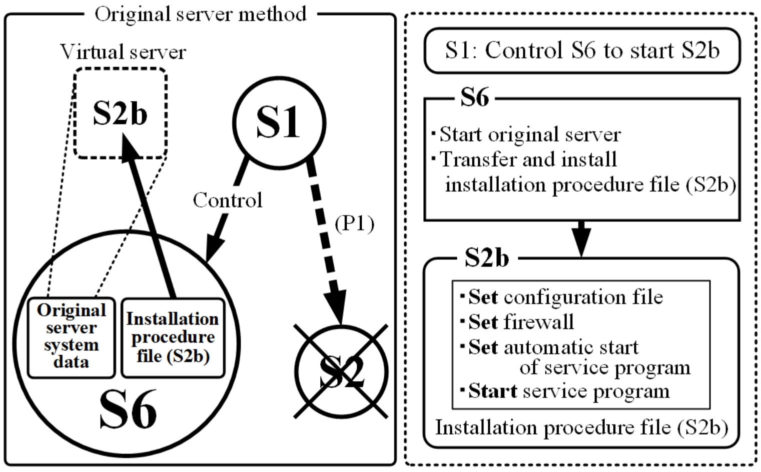
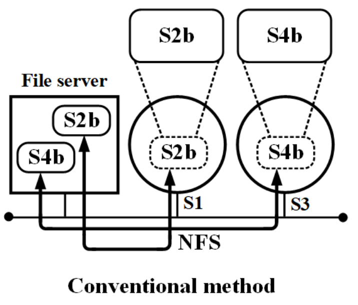

Original server method
Virtual server
S2b
S1
Control
(P1)
Original server system data
Installation procedure file (S2b)
S6
S2
S1: Control S6 to start S2b
S6
・Start original server
・Transfer and install installation procedure file (S2b)
S2b
・Set configuration file
・Set firewall
・Set automatic start of service program
・Start service program
Installation procedure file (S2b)

Fig. 8 Outline of original server method.

Figure 9 shows a comparison of the virtual server startup times for the proposed and conventional methods. As shown in the figure, the conventional method creates virtual servers S2b and S4b on real servers S1 and S3 through an NFS connection with the file server. In contrast, the proposed method uses the original server system data stored on the local disk on the server to create a virtual server, as shown in Fig. 8. In the virtual server, the start time in the suspended condition is much shorter than that in the normal condition. For this reason, the virtual servers are in a suspended condition in both the proposed and conventional methods. With the exception of the CPU specifications, all other experimental system specifications are the same for the file server and the two servers. In the figure, "Start time of one server" indicates the startup time of one virtual server in the normal state, "Time to activate one suspended server" indicates the time to activate one virtual server in the suspended state, and "Time to activate two suspended servers" indicates the time required to activate two suspended virtual servers. The variation coefficient in each case is also shown. These measurement times are the average values of ten experimental results. The average times in this figure are given by the command "time" in UNIX. For all results, it can be seen that the startup times for the proposed method are shorter than those for the conventional method. In the simultaneous startup time of two servers, the startup time for the proposed method was almost the same as the startup time of one server, whereas the time required for the conventional method was significantly increased. This increase is caused by delays resulting from simultaneous NFS access to the file server.

From these results, it is clear that the dependability and usefulness of our proposed method are high because the virtual server startup is stable and the startup time is shorter than the conventional method.



Specifications	S1 / S3	File server
CPU	Intel Core i7-3770 3.40 GHz (TB: 3.90 GHz)	Intel Core i7-4770 3.40 GHz (TB: 3.90 GHz)
Memory	8,192 MB	
Virtualization software	KVM 1.5.3	
OS	CentOS 7.7 (64-bit)	

Experimental result		Start time of one server (s)	Time to activate one suspended server (s)	Time to activate two suspended servers (s)
Proposed method (Original server method)	Average (Variation coefficient)	13.17 (0.02)	1.61 (0.02)	1.67 (0.02)
Conventional method	Average (Variation coefficient)	20.33 (0.02)	4.62 (0.01)	8.50 (0.02)

Fig. 9 Comparison of virtual server startup time.

4. Required points of change for introducing the proposed system to the P2P method

As shown in Figure 2, in the P2P method, the load on the management server is measured at regular intervals, after which the predicted load is calculated, and the real server condition file is then edited based on that load prediction. In the conventional method, all management servers can share management files because the editing is performed on the file server. However, our proposed method requires more time for sharing than the conventional method. In the P2P method, the virtual server that recovers the failed server function is created on the management server with the lowest load at that point in time. Therefore, if failure strikes more than one server simultaneously, the virtual servers needed to recover each server function are created on the same management server because the real server condition file cannot be synchronized. Figure 10 shows the solution to this problem.

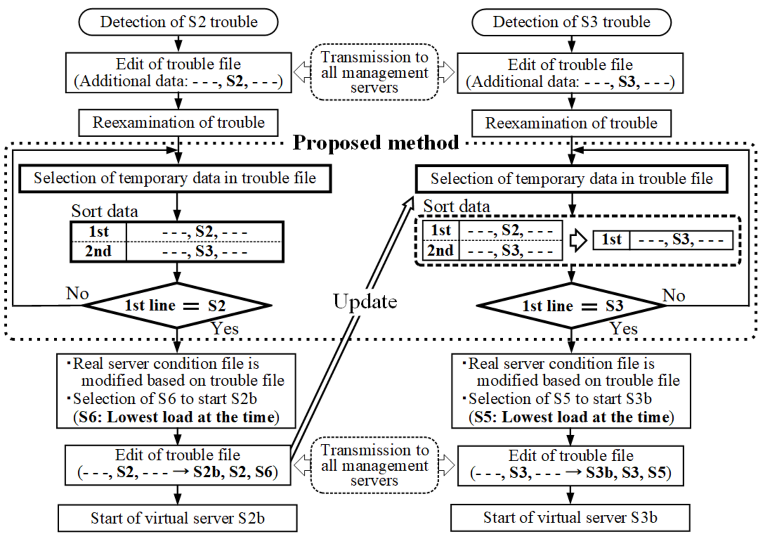

Detection of S2 trouble
Edit of trouble file (Additional data: - - -, S2, - - -)
Transmission to all management servers
Detection of S3 trouble
Edit of trouble file (Additional data: - - -, S3, - - -)
Reexamination of trouble
Reexamination of trouble
Proposed method
Selection of temporary data in trouble file
Selection of temporary data in trouble file
Sort data
1st - - -, S2, - - -
2nd - - -, S3, - - -
Sort data
1st - - -, S2, - - -
2nd - - -, S3, - - -
1st - - -, S3, - - -
No
1st line = S2
Yes
Update
1st line = S3
No
Yes
・Real server condition file is modified based on trouble file
・Selection of S6 to start S2b
(S6: Lowest load at the time)
・Real server condition file is modified based on trouble file
・Selection of S5 to start S3b
(S5: Lowest load at the time)
Edit of trouble file (- - -, S2, - - - → S2b, S2, S6)
Transmission to all management servers
Edit of trouble file (- - -, S3, - - - → S3b, S3, S5)
Start of virtual server S2b
Start of virtual server S3b

Fig. 10 Dealing with simultaneous target server failures.

As shown in Fig. 1, the P2P method records temporary data in the problem file if a target server failure is detected. This editing is immediately synchronized on all management servers by the SEM. The sort data are created by the following procedure. First, the temporary data in the problem file are selected and sorted in the order of the target servers. Next, recovery processing of the first line recorded in the sort data of the target server is performed, and the management server that manages the other target servers enters the waiting state. At that point, the problem server S2 recovery process is performed while the problem server S3 recovery process waits.

The number of virtual servers created on each management server based on the problem file is examined, and the results are recorded in the real server condition file. The load status of all management servers is then examined based on the real server condition file, and S6 is selected as the management server that will create S2b. Next, the temporary data "- - -, S2, - - -", which are related to S2 in the problem file, are edited into regular data "S2b, S2, S6". As a result, because the S3 temporary data are the first in the sort data, the S3 recovery process is started, and the problem of creating the virtual server on the same management server is resolved.

In the proposed method, as shown in Fig. 11, the P2P management program, a transmission program in the SEM, and a reception program in the SEM, all access the management file. Therefore, if the management file is edited at the same time, malfunctions can occur. To address that possibility, we introduced the prioritized file locking system shown in the figure. In terms of priority, the P2P management program is the highest, the transmission program is second, and the reception program is the lowest. The file lock function based on these priorities operates as the management file is edited. Each program also has a lock file to secure the management file. The P2P management program is lp2p, the transmission program is ltra, and the reception program is lrec.

The operation of the prioritized file locking system in the transmission program is shown below. In the management file edit, if all three lock files cannot be located, ltra is created. After that, the existence of lp2p, which is the highest priority lock file, is determined. If lp2p does not exist, the management file is edited by the transmission program. If lp2p exists, ltra is deleted, and the control state of the transmission program returns to the lock file examination state. As a result, the programs are prevented from editing the management file at the same time, so malfunctions caused by simultaneous editing are avoided.

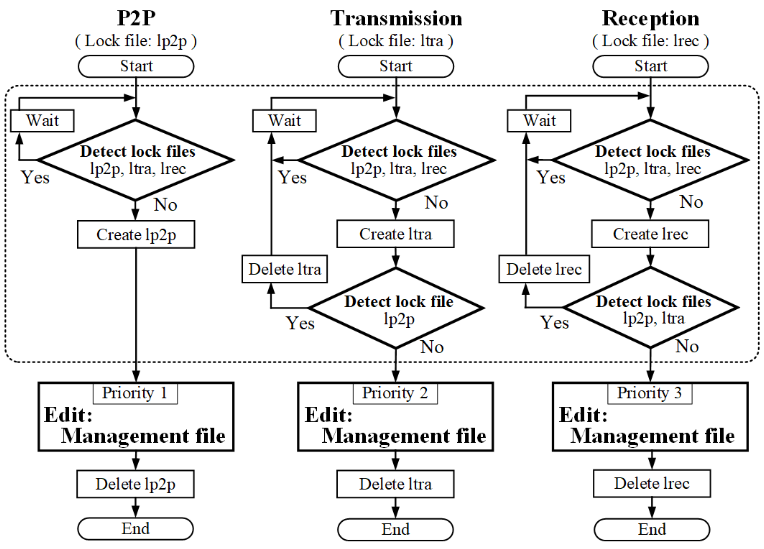


Fig. 11 Prioritized file locking system.

Figure 12 shows the startup procedure and recovery process for the management server in the proposed method. As can be seen in the figure, the management program is stored in each management server, and an executer system performs P2P management program start and end operations. To speed up access to the management file, it is stored on a memory disk. As a result, because the management file is not saved when the management server is started, it enters a standby state, as shown in Fig. 12(a). If the management file is acquired, the executer system enters the operational state. If the flag file is detected, the management program corresponding to each management priority is started. If the flag file is deleted, each management program is terminated.

Figure 12(b) shows the procedure that is followed up to the point where S2 recovers after the management server S1 detects a problem with S2. If S1 detects a network problem in S2, it attempts to restart it. If the network and service offering state of S2 are recovered, the latest management file owned by S1 is sent to S2, and then the flag file is transmitted. This allows S2 to operate as the management server. In contrast, in the conventional method, because the management file exists on the file server, S1 only transmits the flag file to S2.

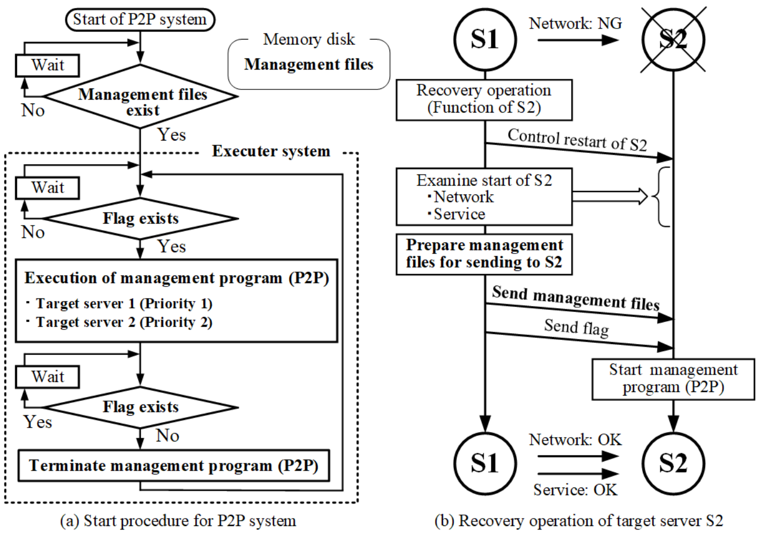

Start of P2P system
Memory disk
Management files
Wait
Management files exist
No
Yes
Executer system
Wait
Flag exists
No
Yes
Execution of management program (P2P)
・Target server 1 (Priority 1)
・Target server 2 (Priority 2)
Wait
Flag exists
Yes
No
Terminate management program (P2P)
(a) Start procedure for P2P system
S1
Network: NG
S2
Recovery operation (Function of S2)
Control restart of S2
Examine start of S2
・Network
・Service
Prepare management files for sending to S2
Send management files
Send flag
Start management program (P2P)
S1
Network: OK
Service: OK
S2
(b) Recovery operation of target server S2

Fig. 12 Startup procedure and recovery process for the management server.

5. Experiment of P2P method server management system incorporating the proposed method

5.1. Experimental system

The construction of the experimental server management system incorporating the P2P method is shown in Fig. 13. The experimental system consists of six servers (Mail, Login, File Transfer Protocol (FTP), Proxy, Web 1, and Web 2), File server 1, File server 2, a client, a router with a 1000BASE-T function, and a 1000BASE-T switching hub. All six servers are equipped with the management server function, and the target server functions required for offering services to the client.

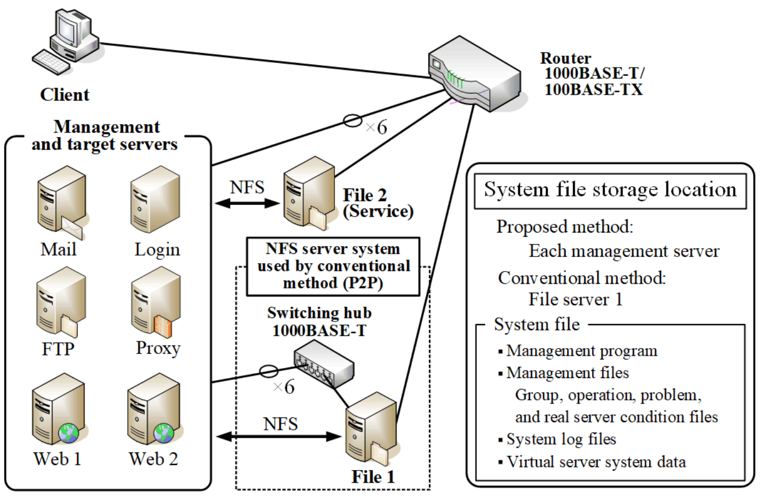


Fig. 13 Configuration of the experimental system.

File server 1 and the 1000BASE-T switching hub are used in the conventional method. Additionally, file sharing storage is generally used in the case of constructing a redundant system for a server. Here, File server 2 is used as the storage. The executer system shown in Fig. 12(a) is installed on each management server.

The system file using the P2P method contains the management program, management files, system log files, and virtual server system data. The proposed and conventional methods store this system file in different locations. Specifically, it is stored in the local area on each management server in the proposed method and in the shared area of File server 1 in the conventional method. In the conventional method, each management server uses the system file through the NFS connection, and the operating state of the management program on each management server is recorded in the system log file.

The specifications of the experimental system, the characteristics of each server, and the setting parameters when the system is in operation are listed in Table 1. The specifications of the six management servers are similar. The memory size is 8,192 MB in consideration of the virtual server startup requirements. CentOS (64-bit) was installed on the management server, File server 1, and the virtual servers. Virtualization was accomplished with Kernel-based Virtual Machine (KVM) software running on the management server and File server 1. Each server operated in a CUI environment to ensure effective memory use.

The characteristics of each server are shown as fundamental data. These data are the average values of ten experimental results. The average times in Table 1 are given by the command “time” in UNIX. In “Virtual server” shown in the figure, “Characteristics” indicates results for a proposed server configuration, and “Characteristics (NFS)” indicates results for a conventional server configuration. In the virtual server, the start time in the suspended condition is much shorter than that in the normal condition. As a result, the virtual server is always in a suspended condition in this system. In “Setting parameters”, “Network examination time (Disconnection)” indicates the network examination time when the target server network is abnormal. “Maximum waiting time for service examination” indicates the maximum waiting time for the service examination.

Table 1 Experimental system specifications, characteristics, and setting parameters.

Specifications	**Management server (Target server)**	**File server 1**
CPU	Intel Core i7-3770 3.40 GHz (TB: 3.90 GHz)	Intel Core i7-4770 3.40 GHz (TB: 3.90 GHz)
Memory	8,192 MB	
Storage	SSD (SATA 3)	
Virtualization software	KVM 1.5.3	
System software	postfix, sshd, httpd, vsftpd, squid	nfsd
OS	CentOS 7.7 (64-bit)	

Specifications	**Virtual server**
CPU	1
Memory	2,048 MB
OS	CentOS 7.7 (64-bit)

Characteristics	**Management server (Target server)**
Start time (s)	26.76
Restart time (s)	29.26

Characteristics	**Virtual server**
Start time (s)	13.17
Time to activate suspended server (s)	1.61

Characteristics (NFS)	**Virtual server**
Start time (s)	20.33
Time to activate suspended server (s)	4.62

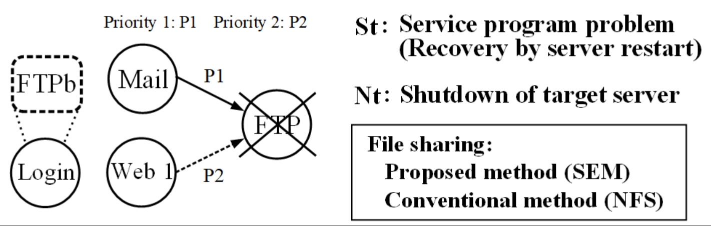
Setting parameters	
Monitoring interval time (s)	10.00
Network examination time (Disconnection) (s)	2.00
Maximum waiting time for service examination (s)	3.00

5.2. Experimental results in the case of target server problem

In the system shown in Fig. 13, an experiment conducted to reproduce the FTP problem managed by Mail and Web 1 is performed. The monitoring interval time is set to 10 s, and the problem on the service program (or the network on FTP) is reproduced at the monitoring interval midpoint (5 s after startup). The recovery times required to deal with these problems are then measured, and the results are shown in Fig. 14.

Each measurement time is the average value of ten experimental results. The service problem is reproduced by stopping the service that provides the FTP program. The network problem is reproduced by shutting down FTP. The two problem types are shown in Fig. 14, where St represents a service problem that can be recovered by restarting the target server, and Nt represents a network problem in a case where the target server is shut down. “Server function” in Fig. 14 indicates the time passage from the point of problem detection until the time the server function is recovered by the virtual server.

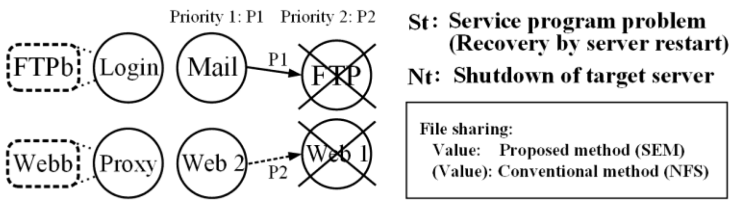
The client accesses the server at one-second intervals, “Access disconnection time (Client)” indicates the time that passes before the access by the client is recovered from the server access disconnection state. Here, the figure shows the measurement time results for both the proposed and conventional methods. The numerical value in parentheses represents the recovery time for the target server, which is the time from the point of problem detection until the target server has recovered to a normal state. For all problems, the proposed method has a shorter server function recovery time than the conventional method, and the same tendency is seen in the client access disconnection time. As for target server recovery time, the results of both methods are approximately the same.


Priority 1: P1
Priority 2: P2
FTPb
Mail
P1
FTP
Login
Web 1
P2
St: Service program problem (Recovery by server restart)
Nt: Shutdown of target server
File sharing:
Proposed method (SEM)
Conventional method (NFS)

Problem list \ Recovery time		Measured time (s)			
		Proposed method	Conventional method	Access disconnection time (Client)	
		Server function (Target server)	Server function (Target server)	Proposed method	Conventional method
Service program	St	5.15 (32.77)	8.43 (32.12)	11.76	15.22
Network	Nt	4.08 (118.22)	7.38 (117.90)	12.53	15.54

Fig. 14 Comparison of server function recovery time in the case of a single server failure.

The recovery time for cases in which problems occur simultaneously on both FTP and Web 1 is shown in Fig. 15. Here, the experimental conditions are the same as in the case of a problem in one server shown in Fig. 14. St and Nt (shown in Fig. 14) are reproduced simultaneously for both servers. The server function recovery time measured by management server Mail operating with P1 and management server Web 2 operating with P2 is indicated. The monitoring interval time is set to 10 s, and problems on the service program or network of each target server are reproduced at the monitoring interval midpoint (5 s after startup).


Priority 1: P1
Priority 2: P2
FTPb
Login
Mail
P1
FTP
Webb
Proxy
Web 2
P2
Web 1
St: Service program problem (Recovery by server restart)
Nt: Shutdown of target server
File sharing:
Value: Proposed method (SEM)
(Value): Conventional method (NFS)

Recovery time of server function		Target server: Web 1 [Measured time (s)]	
		St	Nt
Target server: FTP [Measured time (s)]	St	5.55 (11.65) / 5.25 (11.31)	4.11 (10.37) / 5.17 (10.47)
	Nt	6.42 (12.25) / 4.12 (10.27)	4.30 (10.37) / 4.08 (10.28)

Fig. 15 Comparison of server function recovery time in the case of two server failures.

We performed four kinds of experiments by combining the problems of two managed servers. Note that numerical values that are not enclosed in parentheses indicate the results of the proposed method, while the numerical values in parentheses indicate the results of the conventional method. The server function recovery times in the proposed method are almost the same as the results shown in Fig. 14, whereas the server function recovery times for the conventional method are significantly delayed in all cases. In the conventional method, as shown in Fig. 9, the delay is caused by the simultaneous startup of the virtual servers used to recover the server functions.

From the above results, it can be seen that the use of our proposed method facilitates high server system availability because server function recovery times are shorter and because the influence of multiple simultaneous failures is smaller than would result when the conventional method is used. Moreover, the results shown in Figs. 14 and 15 show that a P2P method server management system incorporating our proposed method is practical.

6. Conclusions

When constructing a high-availability server system, it is essential that the server management system be able to continue its management functions even if the managing server fails. In conventional P2P method server management systems, the file server plays a vital role in sharing management files and virtual server system data. Therefore, if the file server fails, server management cannot continue. In order to solve this problem, we have developed and proposed a P2P method server management system that does not use a file server. To realize this system, we developed a synchronous editing method for sharing management file data and an original server method for starting a virtual server, and then showed their structures and performance levels. In addition, an experimental system that can operate the proposed and conventional methods was constructed and used to measure and compare the server function recovery times of both methods. The results show that the server function recovery times of the proposed method are shorter than those of the conventional method.

Because our proposed method can maintain server management in a stable manner when failures occur, it can prevent loss of trust in the service provider during times of system trouble. Moreover, the operating rates of server systems using our proposed system are likely to improve because the server function recovery time is shorter than that possible with a conventional method, which means that users can receive higher quality services. In addition, for server system administrators, improvements in work productivity can be expected because the server management system has higher overall performance levels.

References

[1] Cabinet Office (Japan)
https://www8.cao.go.jp/cstp/english/society5_0/index.html, 2019.

[2] The Fifth Generation Mobile Communications Promotion Forum
https://5gmf.jp/en/, 2019.

[3] B. Meunier and J. Cosmas, "5G Internet of Radio Light Virtual Reality System," Proc. IEEE Conf., Vol. 2018, No. BMSB, 2018, pp. 1-5.

[4] W. Shengquan, L. Jun, C. Jian-Jia, L. Xue, "Power Sleep: A smart Power-Saving Scheme with Sleep for Servers under Response Time Constraint," IEEE J. Emerg. Sel. Top. Circuit. Syst., vol. 1, no. 3, pp. 289-298, Sep. 2011.

[5] M. Kitamura and C. Fujihashi, "Analysis of Response Time Characteristics of a Real Computer by Heterogeneous-Task Bursty-Arrival Models," IEICE Trans., vol. J81-B-I, no. 4, pp. 243-253, Apr. 1998.

[6] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, "Reliability and Availability Evaluation for Cloud Data Center Networks Using Hierarchical Models," IEEE Access, Vol.7, 2019, pp. 9273-9313.

[7] H. Otsuka, K. Joshi, M. Hiltunen, S. Daniels, and Y. Matsumoto, "Online Failure Prediction with Accurate Failure Localization in Cloud Infrastructures," IEICE Technical Report, Vol. 113, No. 496, 2014, pp. 7-12.

[8] J. Xiao, B. Wu, L. Zhang, H. Wen, X. Jiang, and P.-H. Ho, "Joint Design on DCN Placement and Survivable Cloud Service Provision over All-Optical Mesh Networks," IEEE Trans. Commun., Vol. 62, No. 1, 2014, pp. 235-245.

[9] C. Kari, S. Chen, A.-M. Sepehr, and V. Pallipuram, "Data Migration in Large Scale Heterogeneous Storage Systems with

Nodes to Spare," Proc. IEEE Conf., Vol.2019, No. ICNC, pp. 854-858.
[10] Q. Zhang, M.-F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable Virtual Data Center Embedding in Clouds," Proc. IEEE INFOCOM 2014, vol. 1, pp. 289-297.
[11] M.-G. Rabbani, M.-F. Zhani, and R. Boutaba, "On Achieving High Survivability in Virtualized Data Centers," IEICE Trans. Commun., vol. E97-B, no. 1, 2014, pp. 10-18.
[12] B. Kim, N. Funabikiy, and T. Nakanishi, "File Synchronization Performance using Bittorrent Sync in IEEE802.11ac Wireless Networks," IEICE Technical Report, Vol. 113, No. 463, 2014, pp. 29-34.
[13] F. Purnama, T. Usagawa, R. M Ijtihadie, and Linawati, "Rsync and Rdiff implementation on Moodle's backup and restore feature for course synchronization over the network," Proc. IEEE Conf., Vol.2016, No.TENSYMP, 2016, pp. 24-29.
[14] A. Bhagat and N. Rathee, "Addressing Techniques for Secure Data Sharing in Cloud," Proc. IEEE Conf., Vol. 2018, No. ICICCT, 2018, pp. 499-503.
[15] 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, 2019, pp. 545-556(J-STAGE).
[16] N. Kimura, A. Yamada, H. Seshake, and T. Nishizono, "High Availability Server Platform for IP Communication Services," IEICE Trans., Vol. J88-B, No. 1, 2005, pp. 224-233.
[17] E. Iwasa, M. Irir, M. Kaneko, T. Fukumoto, and K. Ueda, "Methods of Dynamic Scaling for High Availability Server Clusters," IEICE Trans., Vol. J97-B, No. 1, 2014, pp. 19-30.
[18] W. L. Yeow, C. Westphal, and U. C. Kozat, "Highly Available Virtual Machines with Network Coding," Proc. IEEE INFOCOM 2011, Vol. 1, 2011, pp. 386-390.
[19] T. Ono and K. Ueda, "Data Redundancy Dynamic Control Method for High Availability Distributed Clusters," IEICE, Vol. J102-B, No. 9, 2019, pp. 705-709.
[20] M. Ljubojevic, A. Bajic, and D. Mijic, "Implementation of High-Availability Server Cluster by Using Fencing Concept," Proc. IEEE Conf., Vol. 2019 No. INFOTEH, pp. 1-5.
[21] T. Komatsu and K. Ueda, "Data Redundancy Dynamic Control Method for High Availability Distributed Clusters," IEICE Technical Report, Vol.117, No.459, 2018, pp. 91-96.
[22] M. Kitamura, "Configuring a Low-cost, Power-saving Multiple Server Backup System: Experimental Results," IEICE Trans. Commun., Vol. E95-B, No. 1, 2012, pp. 189-197.
[23] M. Kitamura, "Configuration of a Power-saving High-availability Server System Incorporating a Hybrid Operation Method," JISSJ, Vol. 10, No. 2, 2015, pp. 1-17.
[24] 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, 2019, pp. 79-88.
[25] 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, 2020 (Acceptance).
[26] 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, 2018, pp. 14-40.
[27] M. Kitamura, H. Nakagome, Y. Shimizu, K. Tani, and Y. Udagawa, "Development of Peer-to-Peer Method Server Management System in Virtual Server System," IEICE Technical Report, Vol. 118, No. 47, 2018, pp. 15-20.

Authors Biography

Mitsuyoshi KITAMURA

He received his B.E. degree from Tokyo Polytechnic University in 1984. In 1990, he became a research assistant at Tokyo Polytechnic University, where he became an assistant professor in 2003. His current interests are the optimum design and construction of server-client systems and the analysis of various characteristics of server systems.

Koki TANI

He received his M.E. degree in 2020 from Tokyo Polytechnic University, where he is currently in system development and management in cloud computing at Quest Co., Ltd.