［Original Paper］

Development of Ring-type Dynamic Backup Server System not requiring Dedicated Management Server

Mitsuyoshi KITAMURA†, Toshikazu TAKESHITA‡, and Tatsuya OKAZAKI†

† Graduate School of Engineering, Tokyo Polytechnic University
‡ NEC Fielding, Ltd.

Abstract

Server management systems that adopt peer-to-peer methods have been proposed previously as management systems that do not require a dedicated management server. However, a problem with these methods is that the management program becomes complicated. In response, we propose a dynamic backup server system (DBSS) adopting a ring-type configuration that requires neither a dedicated management server nor a complex management program. Our DBSS program only manages one target server and has a straightforward configuration. To realize the ring-type configuration, DBSS adopts an executor system and a synchronization system. In addition, in consideration of the need to handle multiple failures simultaneously, we propose and adopt a dual monitoring method and an automatic connection selection method. Furthermore, by combining a dynamic management extension method with an automatic connection selection method, the server management system configuration can be changed efficiently. Herein, we report on the construction of an experimental system that adopts this method and show its configuration and specifications. We then report on experiments conducted to reproduce service provision and network problems, and the time measurements required to recover the target server function and target server. The obtained results show that the server function and the target server can be recovered efficiently by our proposed DBSS system even if various problems occur in multiple target servers.

1. Introduction

Digital technologies such as artificial intelligence (AI) and the Internet of Things (IoT) are rapidly developing, and digital transformation (DX) [1],[2], which aims to transform people's lives and businesses via the use of such technologies, is attracting significant attention. For example, DX currently plays an important role in helping researchers grasp the infection status of the novel coronavirus, which is rampant on a global scale. In DX, information and communication technology (ICT) infrastructure is very important, which means the roles of servers and networks are crucial. In terms of networks, the 5th generation mobile communication system (5G) [3] that began commercial service in Japan in the spring of 2020 is now bringing great technological innovations to society as a cutting-edge communication infrastructure. Hence, it is clear that we will be able to use more convenient services through the Internet in the future, and the role of the data centers that provide these services will become even more critical.

To that end, numerous important studies are now underway focused on data center networks. In one example, a new server-centric network for data centers, called RRect, has been proposed and has been demonstrated to be more power-efficient in interconnected networks [4]. Other examples include a proposal for a hierarchical modeling framework that can be used to evaluate data center network reliability and availability [5], a framework for automating the disassembly and placement of network functions according to traffic demands and network topology [6], studies pointing out problems with routing algorithms in data center networks, and proposed routing solutions that respond to high-speed and intensive traffic changes such as large-scale flow incast communication [7]. Furthermore, a network measurement and failure location examination system designed to facilitate long-term data center network maintenance [8] has been reported.

In addition, various studies on improving reliability in data centers are underway. These include proposed algorithms that optimize resources and improve load balance in terms of quality of service (QoS), task allocation and resource allocation [9], and consideration of workload impacts (node-level performance metrics and cell-level dynamic random-access memory (DRAM)

[Original Paper] Received 18 March 2022, revised 27 July 2022, accepted 23 August 2022.

access pattern) on DRAM failure [10]. Furthermore, as part of efforts to solve the data migration problems that result from storage performance mismatches in large-scale storage systems such as data centers, proposed countermeasures include the introduction of data transfer bypass nodes that improve migration performance levels [11], as well as a minority disk failure prediction model based on a transfer learning approach that, with the help of neural networks, to improve the predictive performance levels of minority disks in large data centers [12]. In other areas, measurements targeting the reliability of networking, storage, and computational resources based on their availability, outage, and downtime [13] have been reported.

The basic unit of a system that provides services in a data center is the server, and various ideas related to ensuring the high availability of server systems have been reported. Examples include the realization of a high-availability server system using Kubernetes and distributed storage [14], dynamic web server cluster designs using Nginx, php-fpm, MariaDB, GluterFS, and Keepalived [15], and the proposal of a distributed server allocation model that preventively determines server allocation under each failure pattern to minimize delays among all failure patterns for a single server [16]. Other studies are examining server hardware failure prediction technology using machine learning and deep learning while focusing on hard disk drive (HDD), random access memory (RAM), and central processing unit (CPU) issues [17]. Separately, high-availability distributed clusters that can dynamically change configuration while maintaining services based on scalability and traffic rates [18] and high-availability server clusters with prominently improved service reliability and availability levels achieved via the adoption of fencing technology [19] have been reported.

Failover cluster and load balancer cluster systems, which are commonly used to provide server system redundancy, can deal with server failures. However, a particular concern with these solutions is that costs become high because these cluster systems are basically applied to all servers providing the same service. Therefore, a multiple-server backup system (MSBS) [20] that can back up the server functions of multiple real servers with one real server has been proposed. The MSBS is based on multiple dynamic backup server systems (DBSSs), each of which manages only one target server. In addition, taking into consideration different server system operating environments, the adoption of a system interface that simplifies the control programs for the compound operations of power-saving, high-availability server systems [21], and a power-saving, high-availability server system with two different power-saving system configurations have been proposed [22].

Generally speaking, methods in which a dedicated management server monitors all target servers are widespread. Thus, it is necessary to design countermeasures for cases where a failure occurs in a management server, which normally results in complicated management program control methods. Such methods tend to increase the management load in relation to the number of managed servers. In response, a peer-to-peer (P2P) server management system [23] that does not require a dedicated management server has been proposed. In this method, all the servers that provide services to clients fulfill the roles of both management and target servers, and since each target server is managed by two management servers, management priorities can be set to prevent problems if a failure is detected. As a result, management loads do not increase when this method is used, even if the number of target servers increases.

However, since this method requires two management servers to manage one target server at the same time, it requires complicated control actions such as sharing the processing contents while considering management priorities between the management servers. As a result, the management program, including the control portion, becomes complicated.

Therefore, as an alternative server management system that also does not use a dedicated management server, we propose a ring-type DBSS configuration in which the DBSS is applied to each target server. In this configuration, the ring-type DBSS requires cooperation between each management server, so it is necessary to share management files. To accomplish this sharing, the synchronous editing method (SEM) [24], which is a configuration that does not use a file server for management, is adopted. In addition, the original server method [24] is adopted because a virtual server is used to recover failed server functions instead of a spare real server.

Because the virtual server is created on the target server providing client services, a virtual server creation method that considers the load status of the entire server group is adopted. To efficiently optimize the server management configuration in the event of a server failure, we propose and adopt an automatic connection selection method combined with a dynamic management extension method [23]. In addition, in consideration of the potential for simultaneous failure across multiple servers, we propose

and adopt a dual monitoring method. With those measures decided, we then constructed an experimental system that can operate both the proposed method and a P2P method and then conducted experiments to reproduce service provision program and network problems. Next, the server function and target server recovery times and the client access disconnection times were measured, and comparisons were made using both the proposed and P2P methods.

The remainder of this paper is organized as follows. Section 2 describes the outline of the DBSS operation, the ring-type DBSS configuration, and the management file contents. Next, Section 3 describes outlines of the SEM and original server methods, the operating procedure for the executor system that prevents the dispersion of the management program, the operating flow of the synchronization system for synchronizing management functions, and the virtual server creation method that considers the management server load status. In Section 4, we describe a dual monitoring method that considers the simultaneous failure of multiple servers, dynamic management extension and automatic connection selection methods for efficiently optimizing the server management configuration, and the additional DBSS functions required to realize the proposed system. Section 5 describes the configuration of the experimental system to which our proposed method was applied and the functions of each server. We then explain experiments conducted to reproduce various network or service problems on the target server and show that the proposed system can deal with them. Section 6 describes the characteristics of general server management systems and compares them with the proposed method in terms of operation and cost. Finally, we provide our conclusions in Section 7.

2. Overview of ring-type DBSS

2.1. DBSS operation outline

Figure 1 shows a configuration and operational overview of the DBSS [20]. As can be seen, the server management system consists of a management server, target servers (Mail, Web), a file server, and a switching hub. The management server can also create virtual servers (Mailb, Webb). A virtual Internet Protocol (IP) address is set for the target server, and services are provided to users using that IP address. User data and configuration files are saved in the file server using a network file system (NFS). The DBSS is executed on the management server, and one DBSS manages one target server. Therefore, when managing two target servers, as shown in the right side of the figure, two DBSSs are executed. As shown in the flowchart on the left, the DBSS is a very simple system and monitors the network and the service of the target server at regular intervals.

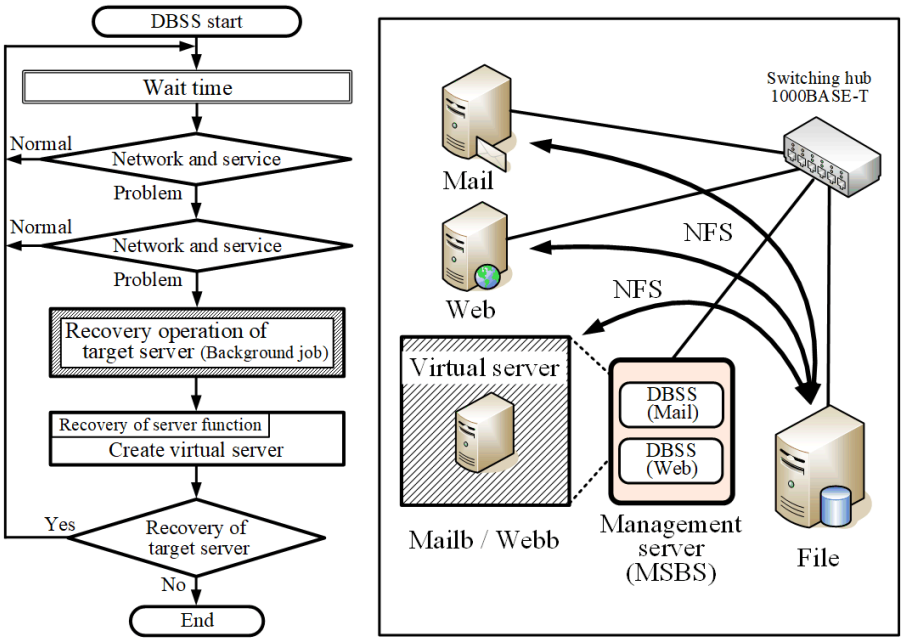


Fig. 1: DBSS configuration and operation overview.

If a problem is detected, the network and services are reexamined. In cases where the result is an abnormal state, after executing the target server recovery processing as a background job, a virtual server that can provide the same service as the target server is created. Thereafter, the function of the failed server is recovered by setting the virtual IP address, which offers services to clients, to the virtual server. In the background job, the DBSS examines whether the target server is in the restarted state and attempts to forcibly restart it by using the wake on LAN (WOL) function if it is not in the restart state. In the management program, the background job creates a file to report the result to the foreground job. If it is recovered, an up.file is created, and if it is not recovered, a down.file is created. Finally, depending on the recovery processing result for the target server executed in the background job, a decision is made as to whether to return to the normal monitoring state or terminate the DBSS.

2.2. Configuration overview of P2P method server management system and ring-type DBSS

Because the system shown in Fig. 1 uses a dedicated management server, management will not continue if that server fails. In addition, the load for management will increase if the number of target servers increases. A P2P method server management system that can solve these problems has been proposed [23]. Figure 2 shows an outline of the P2P method configuration and our proposed ring-type DBSS. The P2P method configuration can be seen in Fig. 2(a). In this method, all servers that provide services to clients fulfill the roles of both management and target servers. Because one target server is managed by two management servers, when a problem is detected, the server that manages preferentially is defined as management priority 1 (P1), and the other is defined as management priority 2 (P2). In this configuration, the management load does not increase even if the number of target servers increases. However, since two management servers manage one target server at the same time, complicated control between them is required in areas such as sharing the processing contents and considering management priorities. Therefore, the management program, including the control portion, is complicated.

Figure 2(b) shows the configuration outline of the proposed ring-type DBSS, which is a very simple system. Since the goal of this study is to design an effective server management system that does not require a dedicated management server, and since the DBSS is very simple, as shown in Fig. 1, we propose a method by which management is performed by adopting the DBSS in each target server. More specifically, one management server manages one target server, the mail server manages only the FTP server, and the FTP server only manages the Web 1 server. In this way, a ring-type management configuration is constructed. However, in order to realize the proposed method, it is necessary to add functions to the DBSS.

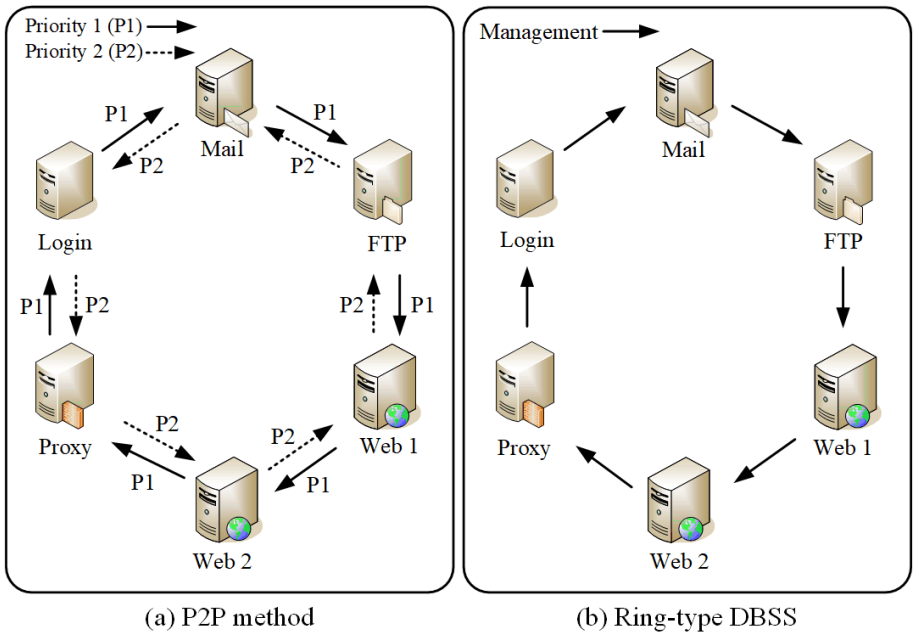


Fig. 2: Configuration overview of the P2P and ring-type DBSS server management systems.

2.3. P2P method operation flow

An operation flowchart of two server management priorities is shown in Fig. 3, where we explain the flow using only four servers. Servers S1, S3, and S4 are management servers, while S2 is the target server, and S2b is the virtual server for recovering the S2 function. Here, (P1) and (P2) indicate states in which a recovery operation is being performed with their respective priorities. Server S1 executes the management program with P1, and S3 executes the management program with P2. Both S1 and S3 regularly monitor the network and the service offer state of S2. In the P2P method, the synchronization processing is executed to synchronize the monitoring times of all the management servers, and the S.file, which is a synchronization file, is regularly broadcast from the management server. Thus, servers S1 with P1 and S3 with P2 synchronously monitor S2.

If trouble is detected in S2, S1 executes the background job that performs the S2 recovery operation. Server S1 executes a new program that manages S3 with P1. This program also starts a separate program that manages S1 with P2. Next, S1 creates S2b and sets its virtual IP address to permit the recovery of the S2 function. Thereafter, S1 enters a wait state in order to find the up.file or down.file, which was created by the background job. If S3 detects trouble in S2, it enters a wait state in order to receive the U.file or D.file, which is sent by S1. In this state, S3 regularly examines the network and the service offer state for S1. If S1 is determined to be in an abnormal state, then S3, rather than S1, performs an operation with P1. If S2 is determined to be recovered by the background job, the up.file is created. If S1 finds the up.file, the U.file is sent to S3 via inter-process communication and both S1 and S3 synchronously monitor S2 again after the U.file is received. If S2 is determined not to be recovered by the background job, the down.file is created. If S1 finds the down.file, the D.file is sent to S3 by inter-process communication. When S3 receives the D.file, the S2 management programs are terminated. The processes that must be performed in the example above show how the P2P method management program can become very complicated.

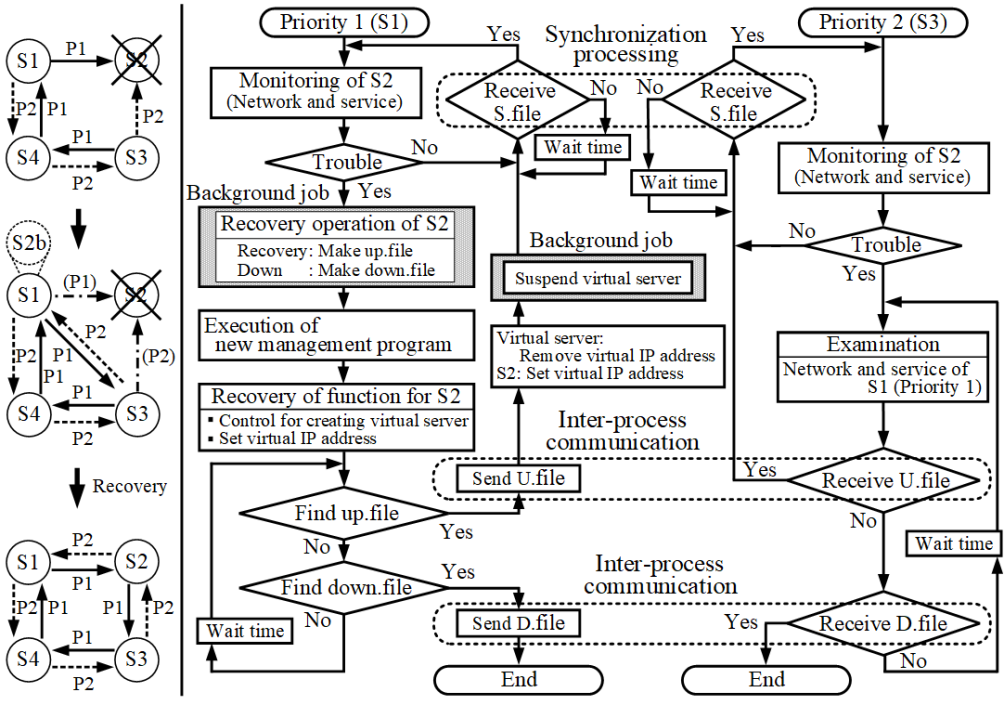
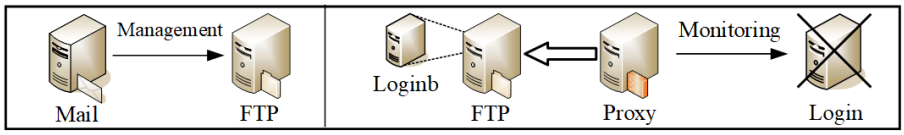


Fig. 3: Flow chart of each management priority in the P2P method.

2.4. Management file and management data

The DBSS shown in Fig. 1 does not require a management file because one management server manages one target server. However, to construct the ring-type DBSS, the management files (group, operation, separation, heartbeat, and problem files) shown in Fig. 4 are necessary because cooperation between each management server is required. The management data are the data required for the DBSS to manage the target server. These files and data are saved on the local memory disk on each management server.



Group file

No.	Target server	Management data
1	Mail	Mail.dat
2	FTP	FTP.dat
3	Web 1	Web1.dat
4	Web 2	Web2.dat

Operation file

Management server	Target server	Management data
Mail	FTP	FTP.dat
FTP	Web 1	Web1.dat
Web 1	Web 2	Web2.dat
Web 2	Proxy	Proxy.dat

Separation file

No.	Target server	Management data
6	Login	Login.dat

Heartbeat file

Management server	Heartbeat message (UNIX time)
Mail	1622708875
FTP	1622708875
Web 1	1622708875
Web 2	1622708875

Problem file

Virtual server (Backup server)	Trouble server	Server creating VM
Loginb	Login	FTP

Management data

	Mail.dat	Web1.dat
1st line : **Virtual IP address (VIP)**	172.21.14.201	172.21.14.203
2nd line: **Host name**	Mail	Web1
3rd line : **IP address (IPR)**	172.21.14.1	172.21.14.3
4th line : **Mac address**	00:00:00:00:00:01	00:00:00:00:00:03
5th line : **Service daemon**	postfix dovecot	httpd
6th line : **Server type**	NETWORK	CPU_NET

Fig. 4: Management files and management data.

The number, target server name, and name of the management data for managing the server are recorded in the group file. From this file, an operation file that contains the names of the management server, target server, and management data is created. In this figure, based on the information in the section surrounded by the thick line in the operation file, the mail server manages the FTP server. If the target server fails, a problem file containing the names of the virtual server that recovers the server function, the failed target server, and the management server that creates the virtual server is created. In the case of a login server failure, the proxy server controls the FTP server to recover its function by creating a virtual server loginb. This state is recorded in the section surrounded by the thick line in the problem file.

Each file is linked, and if it is determined that the server has failed, the data for that server are moved from the group file to the separation file. Then a new operation file is created from the group file. If the server is recovered, the corresponding data are deleted from the problem file, the data are returned from the separation file to the group file, and the operation file is reconstructed. The management configuration is dynamically changed based on this file. In the proposed system, each server internally monitors the network and service provision status. If it is determined to be normal, the current (UNIX) time is recorded in the heartbeat file.

The information necessary for management is recorded in the management data, and the management program is executed with this information as an argument. The virtual IP address set for the target server is recorded in the first line of the data, and the target server information is recorded in the second to sixth lines. Here, the service daemon name on the fifth line can be recorded using space separation, e.g., “postfix dovecot” because one server can provide multiple services to clients.

3. Basic technology for constructing ring-type DBSS

3.1. Management file sharing and recovery method for server functions

The proposed system adopts management file sharing and failed server recovery methods that do not require the use of a file server. Figure 5 shows our method for sharing management files on all management servers. Here, the SEM [24] is used as a

file sharing system for management files, while (M1) indicates the status of the target server recovery process.

Taking the group and problem files as an example, the method for sharing updated files if the target server FTP fails is shown. As shown in the figure, information on the running server is recorded in the group file, and the server determined to be abnormal is deleted from the file. However, it is still necessary to reflect this information on all management servers. In the SEM, the file editing control data (Diff.dat) corresponding to the edited portion is sent to each management server, which then receives it and edits the saving management file. This procedure allows the management files saved on each management server to be shared. The Diff.dat contents are “filename, Del, and deletion information” when deleting information; “filename, Add, and additional information” when adding information; and “filename, Mod, information before change, and information after change” when rewriting the contents.

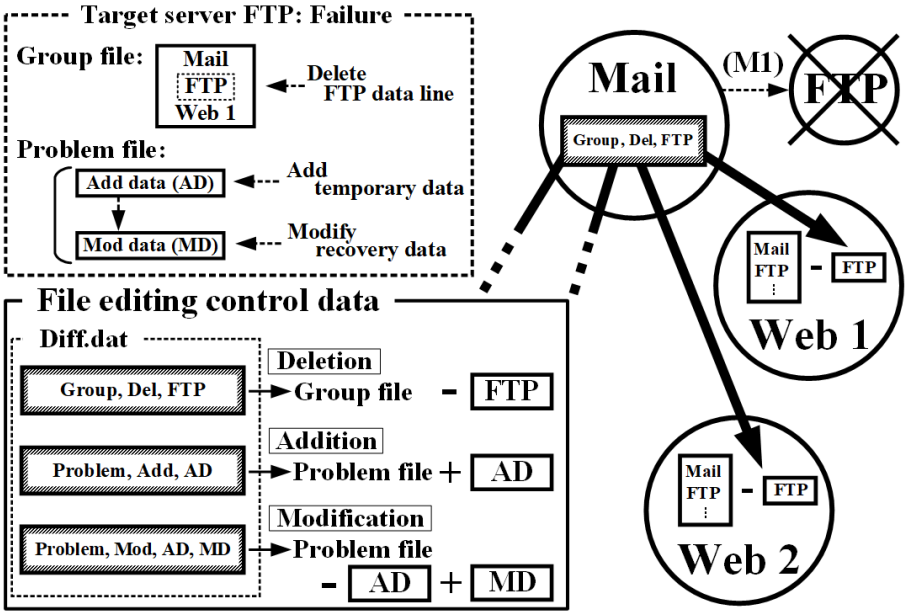


Fig. 5: Management file sharing.

Figure 6 shows an operation outline of the original server method [24], taking the case of recovering the function of a failed FTP server as an example.

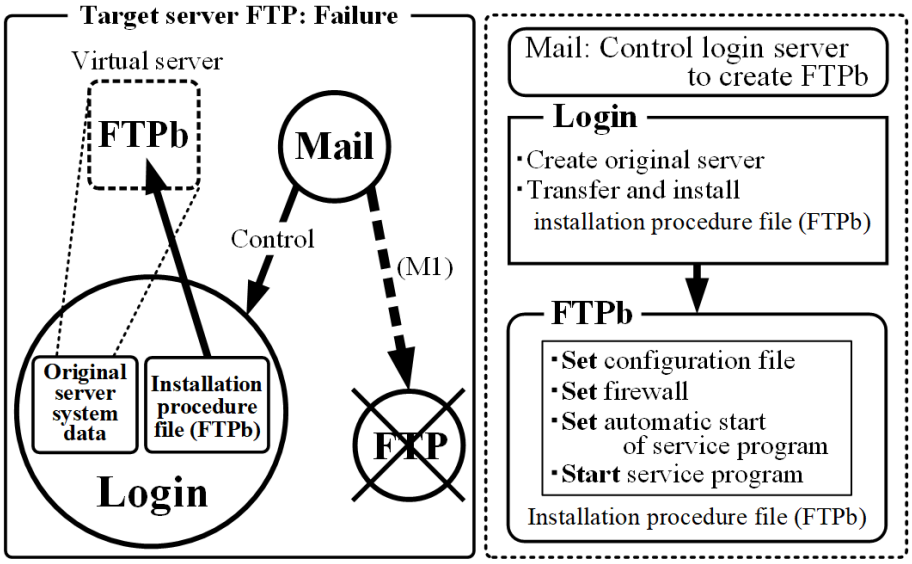


Fig. 6: Backup method for server functions.

The original server method consists of the original server on which the service programs of all target servers are installed and the installation procedure file that records information such as the startup settings of the service programs provided by the failed server. The original server is created using the system data for the original server. Thereafter, to use the original server as FTPb, the "Installation procedure file (FTPb)" is transferred to the server and installed. Using this procedure, the original server can operate as a backup server that recovers the FTP server function. Here, because the size of the installation procedure file is small, it is possible to save the installation procedure file for all target servers to all management servers.

3.2. System start and synchronization

Figure 7 shows the executor and synchronization systems used in the proposed method. In our method, the executor system starts and terminates the management program. Centralized management of management files in the initial state, the management data, and the management program is performed by the file management server. The management file, the management data, and the management program are sent from the file management server to the local memory disk on each management server at the management system start time. Therefore, when the management server is started, it is in the standby state, as shown in Fig. 7(a), because the management file is not saved.

When the management file, the management data, and the management program are acquired, the executor system is in the existence check state of the flag file. The file management server creates and deletes the flag file on each management server. Here, if the flag file is detected, the management program is started, and if it is deleted, the program is terminated. As shown in Fig. 2(b), the management file must be shared because management is performed by each management server in the proposed system, which means that management synchronization is required. Figure 7(b) shows the operational flow of the synchronization system. Each server uses the chronyd daemon to set the computer time accurately. Using this environment, we propose and adopt a synchronization method that uses time as the monitoring time of each management server. More specifically, an accurate monitoring interval time of 10 seconds is created by performing the operations using UNIX time. As a result, our proposed method does not require access to other servers to synchronize server management.

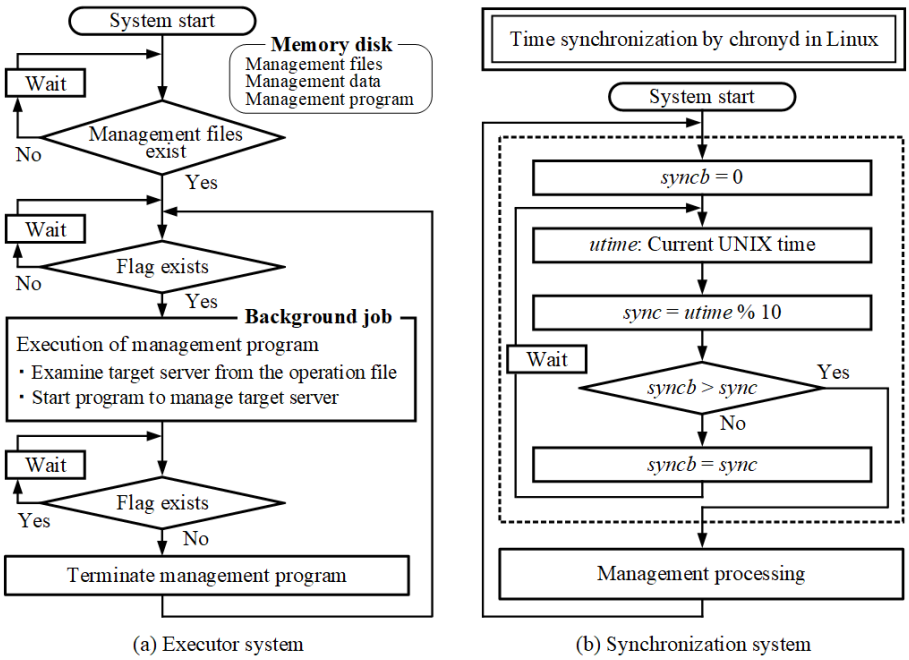


Fig. 7: Executor and synchronization systems.

3.3. Virtual server creation method considering load status

Figure 8 shows how to create a virtual server while considering the management server load status in order to recover the function of a failed server. Each management server periodically measures the server load, which consists of memory, CPU, and network usage. The coefficients and formulas required to calculate the predicted load are shown in this figure. Here, *i* is the number of measurements, *mi* is the server's available memory size, *ci* is the CPU idle rate, and *ni* is the total transfer bytes transmitted and received by the network. From the average load value acquired in two steps, the predicted load is calculated based on the difference. The predicted memory load Pm_n is defined as follows:

$$Pm_n\ (\mathrm{KB}) = 2 \times Am_n - Am_{n-1}. \tag{1}$$

The predicted CPU load Pc_n is given as follows:

$$Pc_n\ (\%) = 2 \times Ac_n - Ac_{n-1}. \tag{2}$$

The predicted load Pn_n of the network is defined as follows:

$$Pn_n\ (\mathrm{KB}) = 2 \times An_n - An_{n-1}. \tag{3}$$

Because the predicted loads obtained from equations (1), (2), and (3) may produce a negative value or exceed the upper limit due to fluctuations, the minimum value is set to zero, and the maximum value available for *P*m is the physical memory size of the management server, *P*c is the number of CPU cores × 100, and *P*n is the rated transfer rate × 2. The memory remaining capacity value Rm_n is defined as follows:

$$Rm_n\ (\%) = \frac{Pm_n}{Pm_{\max}} \times 100, \tag{4}$$

where Pm_{max} is the maximum value of *P*m. The CPU remaining capacity value Rc_n is given as follows:

$$Rc_n\ (\%) = \frac{Tc_n}{Tc_{\max}} \times 100, \tag{5}$$

where *T*c is the value obtained by adding the number of CPU cores to Pc_n, and Tc_{max} is the maximum value of *T*c. The network remaining capacity value Rn_n is defined as follows:

$$Rn_n\ (\%) = \frac{(N_{\mathrm{rat}} - Pn_n)}{N_{\mathrm{rat}}} \times 100, \tag{6}$$

where *N*rat is the total number of bytes transferred per second for the rated transmission and the network reception.

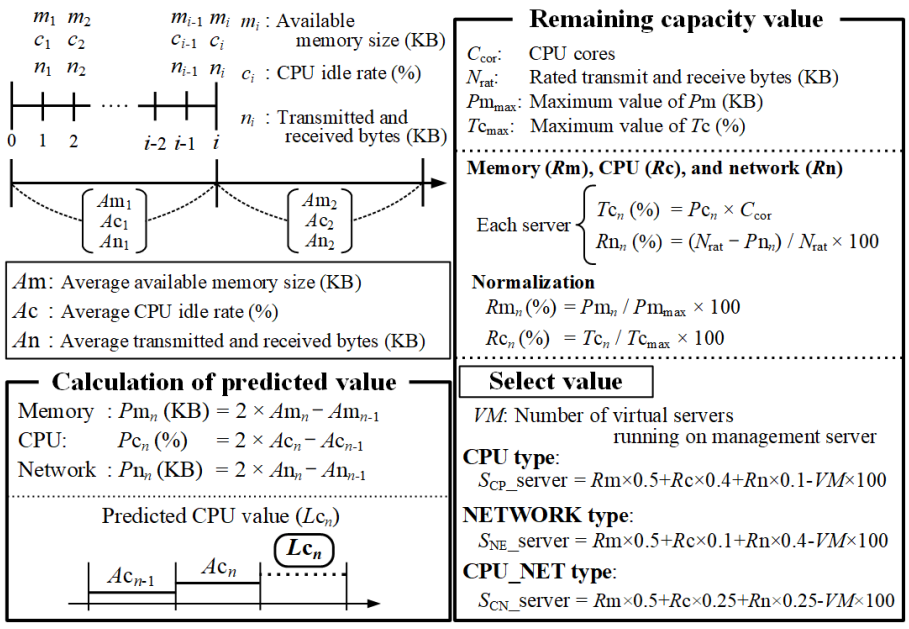


Fig. 8: Virtual server creation method considering load status.

In this system, the virtual server is used to recover the function of the failed target server. Therefore, it is necessary to select the management server for creating the virtual server. The selection value of each management server corresponding to the server type of the failed target server is calculated based on this remaining capacity value. The management server with the largest selection value is selected as the target for creating the virtual server. Three server types are defined: a server type that mainly uses a CPU, a NETWORK type server that mainly provides data, and a CPU_NET type server that uses a CPU and provides data. Here, the weight values of *R*m, *R*c, and *R*n are set according to the server type, and the total value obtained from these weights is set to a maximum of 100. Furthermore, *VM*, which is the number of virtual servers running on the management server, is 100 times weight. Because the memory load *R*m is important in every server type, *R*m is 0.5 times. For the CPU-type server, *R*c is 0.4 times, and *R*n is 0.1 times. For the NETWORK-type server, *R*c is 0.1 times, and *R*n is 0.4 times. In the CPU_NET-type server, *R*c is 0.25 times, and *R*n is 0.25 times. The weightings set here are based on a previous report [23]. However, in normal cases, weightings should be determined based on the server system operating environment during actual operations.

4. Key technologies required to configure ring-type DBSS

4.1. Dual monitoring method

This system adopts a ring-type configuration in which one management server manages one target server in order to simplify the management program. Therefore, if multiple target servers fail simultaneously, continued management may not be possible. To minimize this risk, we propose and adopt the dual monitoring method shown in Fig. 9.

Figure 9(a) shows the normal monitoring type, in which the management server monitors the target server, and which is called external monitoring because the network and the service-providing status of the target server are monitored by network access. In contrast, Fig. 9(b) shows the process we call internal monitoring, in which the management server accesses the default gateway address for examining the network and the local service port for examining the service-providing status. The result is recorded in the heartbeat file shown in Fig. 4, where it is shared by all management servers. In Fig. 9(c), which shows the management timing for simultaneous failures, internal monitoring is performed during the monitoring interval time, and it is possible to recognize the failed server by examining the heartbeat file before performing normal external monitoring. Therefore, if the FTP server and Web 1 server fail at the same time, the DBSS, which manages Web 1, is executed by the dynamic management extension method because the mail server that monitors the FTP server can recognize two server failures. As a result, it is possible to examine two servers at the same time by external monitoring.

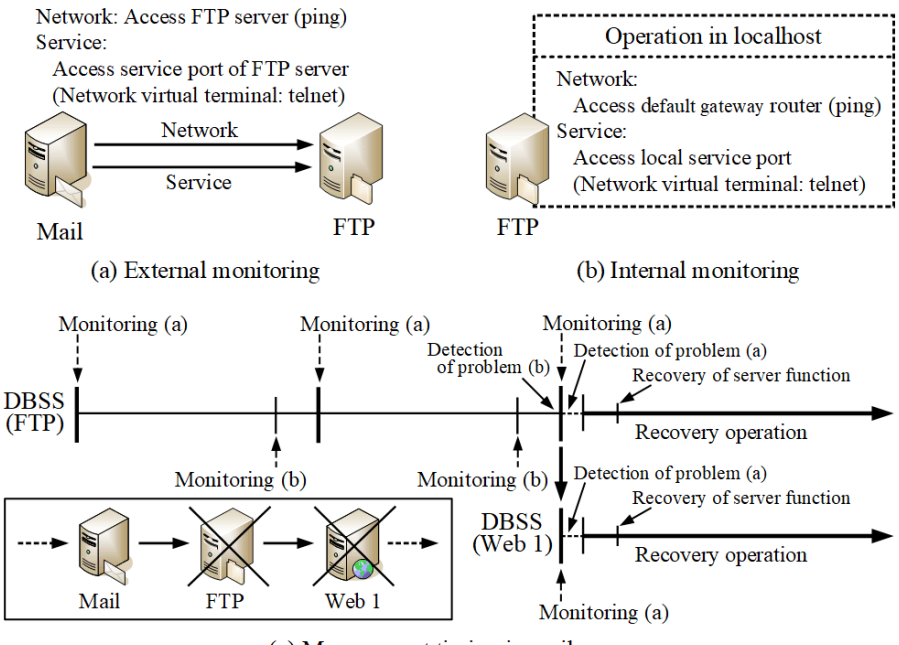


Fig. 9: Dual monitoring method.

4.2. Processing procedure at the time of server failure and recovery

Figure 10 shows how to recover the server function if the target server fails. Servers S1 through S6 are the management and target servers, while S1b through S4b indicate virtual servers for recovering server functions, and M1 through M4 indicate normal management states for monitoring the network and service-providing status. In addition, (M1), (M2), and (M3) indicate the status of the target server recovery process and whether it is possible to return to the normal state.

Figure 10(a) shows the server function recovery method if S2 fails. In this case, if the DBSS for managing S2 on S1 (DBSS_S2_S1) detects an S2 failure, DBSS_S2_S1 starts the DBSS for managing S3 (DBSS_S3_S1) using the dynamic management extension method. After that, DBSS_S2_S1 creates virtual server S2b on S6 using the largest selection value of all the management servers and sets the virtual IP address. As a result, the access from the client is switched to the virtual server. Here, DBSS_S2_S1, which executes (M1), performs the S2 recovery operation and then determines whether S2 returns to the normal state. If DBSS_S2_S1 determines that it would be too difficult to recover S2, DBSS_S2_S1 is terminated. If S2 is recovered, (M1) returns to M1, which is normal monitoring. Because S2b only performs its role during the S2 repair time in this system, S2b is not managed.

Figure 10(b) shows a case where two target servers fail at the same time. In this case, because this system considers simultaneous failures, the failure status of each target server is grasped by internal monitoring within the monitoring interval time. More specifically, DBSS_S2_S1 detects the S2 and S3 failures before performing external monitoring and can start DBSS_S3_S1 and DBSS_S4_S1 using the dynamic management extension method. Meanwhile, virtual servers S2b and S3b are created by DBSS_S2_S1 and DBSS_S3_S1, respectively, and the functions of the failed servers are recovered. If DBSS_S2_S1 and DBSS_S3_S1 determine that S2 and S3 would be too difficult to recover, DBSS_S2_S1 and DBSS_S3_S1 performing (M1) and (M2) terminate. If S2 and S3 are recovered, (M1) and (M2) return to M1 and M2, which corresponds to normal monitoring.

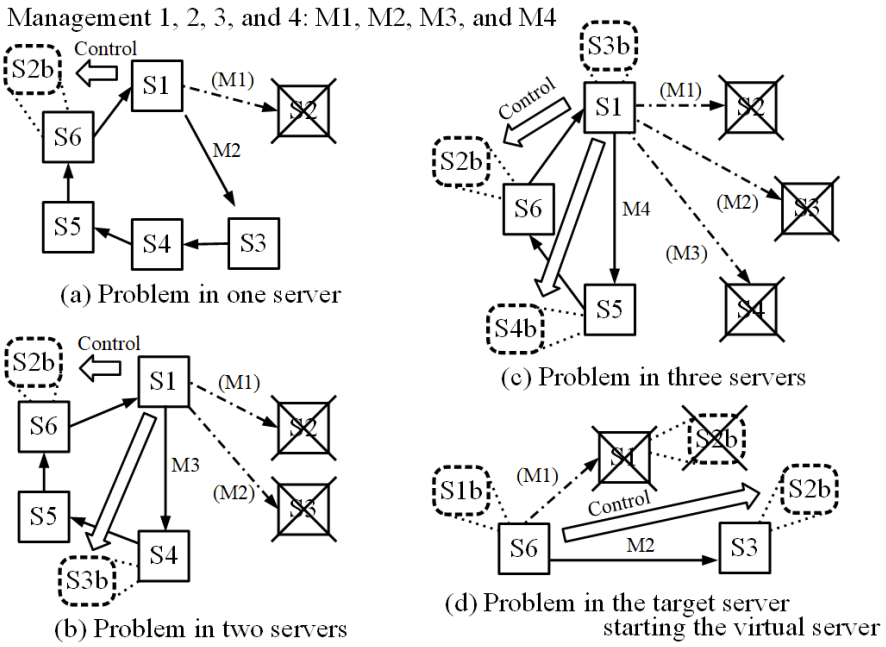


Fig. 10: Server function recovery method.

Figure 10(c) shows a case where S2, S3, and S4 fail at the same time. In this case, because DBSS_S2_S1 can detect S2, S3, and S4 failures before performing external monitoring, DBSS_S2_S1 can start DBSS_S3_S1, DBSS_S4_S1, and DBSS_S5_S1 using the dynamic management extension method. Meanwhile, virtual servers S2b, S3b, and S4b are created by DBSS_S2_S1, DBSS_S3_S1, and DBSS_S4_S1, respectively, and the functions of the failed servers are recovered. In this method, because the target server has the management server function, it is possible that the target server has already created the virtual server necessary to recover the failed server function. Figure 10(d) shows a case where S1, which creates S2b, fails. Based on the server load,

DBSS_S1_S6 creates S2b in S3 at the same time it creates S1b. Because this system uses the original server method, it is possible to create S2b on S3.

Figure 11 shows the processing procedure that takes place after the target server recovers. If the target server is recovered, the management server edits the management file, sends the management program and management file saved in local memory disk to the server, and then sends the flag file for the executor system. Here, to eliminate the program complexity related to server management configuration changes, we propose and adopt the automatic connection selection method. The method is executed after the monitoring interval time ends, and if the target server recording in the operation file shown in Fig. 4 is the same as the currently managed server, the management is continued. If it is different, the program is terminated. The processing procedure used when the target server is recovered from the state shown in Fig. 10(b) is shown below.

Figure 11(a) shows a case where only S2 is recovered after the simultaneous failure of S2 and S3. In this situation, DBSS_S2_S1 edits the management file, performs virtual IP address migration processing for S2b and S2, and shifts S2b to the suspended state. After that, DBSS_S2_S1 returns the management for S2 to the normal management state M1, and DBSS_S4_S1 performing M3 is terminated by the automatic connection selection method. DBSS_S4_S2 is started because S3 is in a failure state.

Figure 11(b) shows the processing procedure if S3 is recovered after S2 is recovered in Fig. 11(a). In this case, DBSS_S3_S1 edits the management file, performs virtual IP address migration processing for S3b and S3, and then shifts S3b to the suspended state. After that, DBSS_S3_S1 and DBSS_S4_S2 are terminated by the automatic connection selection method at the same time that DBSS_S3_S2 is started.

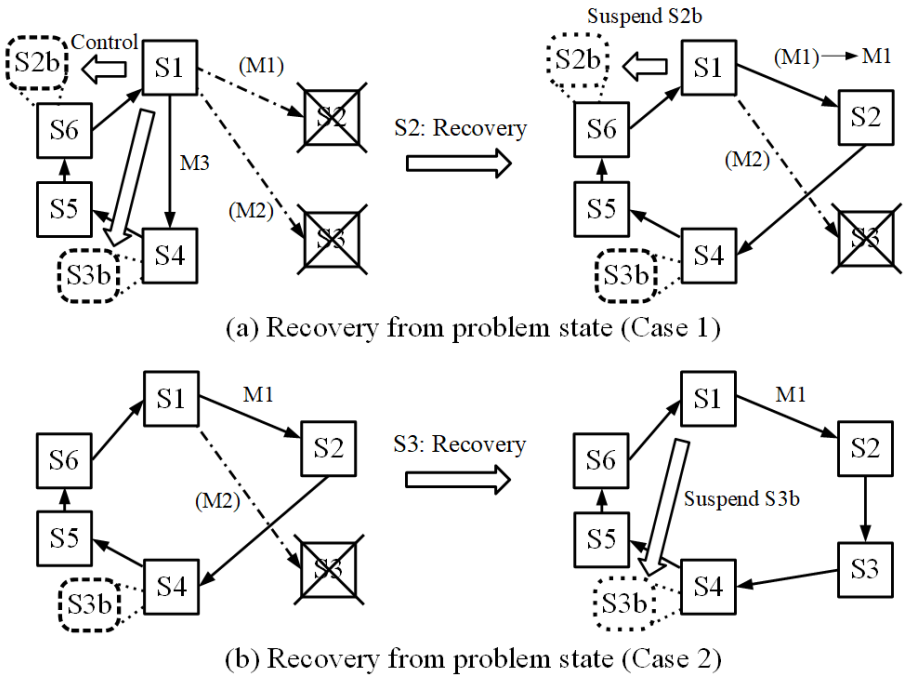


Fig. 11: Processing procedure after recovery.

4.3. Additional DBSS processing

Figure 12 shows additional items (A1 through A6) that are required to achieve a ring-type DBSS. The difference from the DBSS flowchart in Fig. 1 is that “Waiting time” is “Time synchronization processing”. Because all management servers are configured in a ring-type system, it is necessary to perform all monitoring timings simultaneously. In the portion indicated by A1, the load measurement at each management server, the predicted load calculation, and internal monitoring are performed within the monitoring interval time. In addition, the portion indicated by A2 is added before the management program executes external monitoring. Here, depending on the internal monitoring status, it determines whether to execute the DBSS that manages the new target server via the dynamic management extension method and whether to continue managing the current target server via the

automatic connection selection method.

If the target server failure is detected by external monitoring, the acquisition process for the predicted load shown in A3 is executed from each management server. External monitoring is executed again for reexamination, and if an abnormal status is detected, the management file shown in A4 is edited. After executing the recovery process for the target server as a background process, the management server decision process that creates the virtual server indicated by A5 is executed from the remaining capacity value. A virtual server is created on that server, and the server function of the failed server is recovered. If the target server is recovered, the processing shown in A6 is performed. Here, the management file is returned and sent to the target server, after which the flag file for starting the management program is sent.

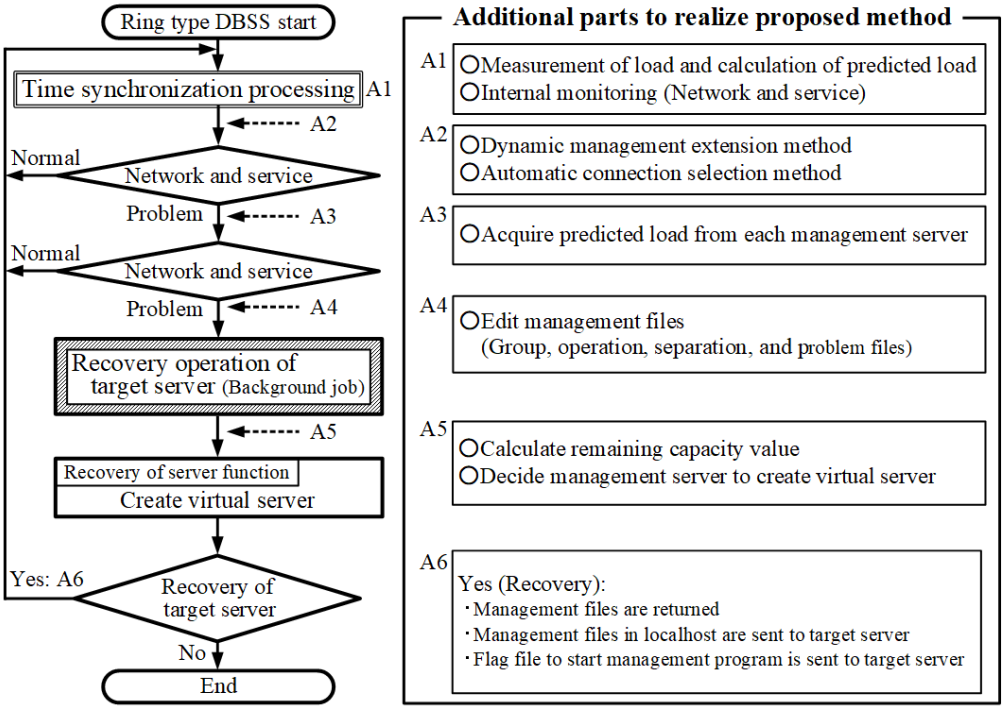


Fig. 12: Additional processing to realize ring-type DBSS.

5. Ring-type DBSS operation experiment

5.1. Experimental system

Figure 13 shows the server system configuration for the ring-type DBSS operation experiments. The experimental system consists of six servers (Mail, login, FTP, proxy, Web 1, and Web 2), a file server, a client, and two 1000BASE-T switching hubs. The six servers have management server functions and the target server that provides services to clients. One of the two switching hubs is for providing services to clients and the other is for the SEM.

When constructing a server redundancy system, file sharing storage is generally used. Here, storage for the target server providing services is provided by the file server, and each server is connected via NFS. The file server also has a file management server function that sends the management file and the management program at the system start.

The executor and synchronization systems shown in Fig. 7 are installed on each management server. The management program, management file, system log file, and system data of the original server are saved on the local disk of each management server. The SEM and original server methods shown in Figs. 5 and 6 are used for the management file sharing and virtual server creation methods. The operating status of the management program on each management server is recorded in the system log file.

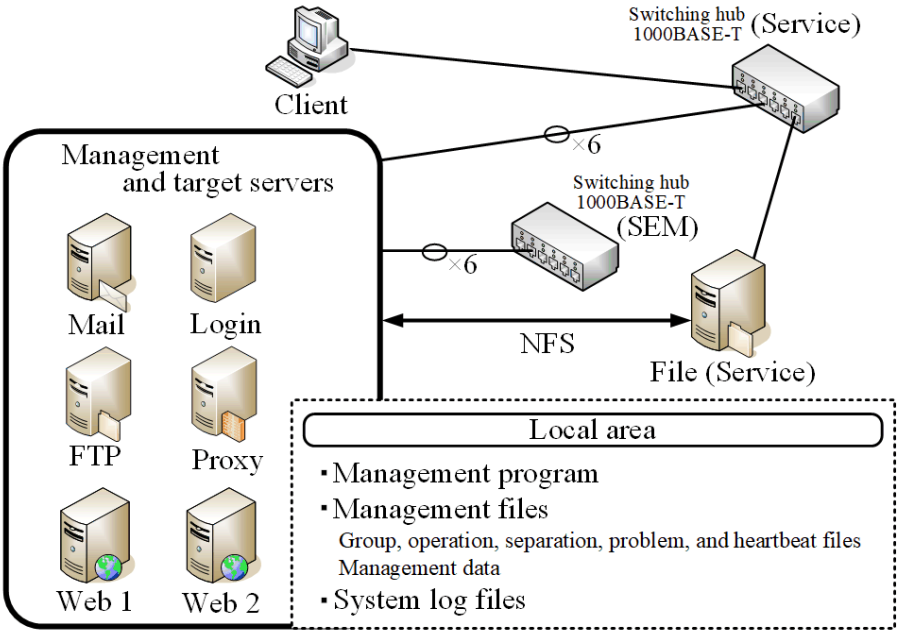


Fig. 13: Experimental server system for ring-type DBSS.

The experimental system specifications, characteristics of each server, and setting parameters during system operation are listed in Table 1. The specifications of the six management servers are the same, and the memory size is set to 8 GB in consideration of the need to create the virtual server. The Community Enterprise Operating System (CentOS), which is often used as the OS for servers, is used on the management and virtual servers. Virtualization is realized by a kernel-based virtual machine (KVM) running on the management server. Each server operates with a character-based user interface to ensure effective memory usage. The characteristics of each server are shown as fundamental data. These data are the average values of 10 experimental results. The average times in Table 1 are measured by the "time" command in UNIX.

Table 1: Specification, characteristics, and settings parameters for the experimental system.

Specifications	Management server (Target server)
CPU	Intel Core i7-3770 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, Apache, vsftpd, squid
OS	CentOS 7.9

Specifications	Virtual server
CPU	1
Memory	2,048 MB
OS	CentOS 7.9

Characteristics	Management server (Target server)
Start time (s)	31.07
Restart time (s)	33.33

Characteristics	Virtual server
Start time (s)	14.13
Time to activate suspended server (s)	1.52
Service examination time (s)	0.02~1.10

Setting parameters	
Monitoring interval time (s)	10.00
Network examination time (Disconnection) (s)	2.00
Maximum waiting time for service examination (s)	3.00
Maximum wait time for real server restart (s)	50.00

In the virtual server, the startup time from the suspended state is much shorter than the startup time from the normal state. Therefore, the virtual server is always in a waiting state as a suspended condition in this system. The service examination time varies depending on the type of service provided, and is 0.02 s for the FTP server, 0.23 s for the login server, 0.36 s for the mail server, 0.36 s for the proxy server, and 1.10 s for the web server. In the settings parameters, the monitoring interval time is set to 10.00 s, and the network examination time is set to 2.00 s if the target server network is in an abnormal state. In addition, the maximum waiting time for service examination is 3.00 s, and the maximum waiting time required to determine whether the target server is in the restart state is 50.00 s. This time is approximately 1.5 times the average restart time of the target server.

5.2. Log files to record the operating status

If a problem occurs in the target server, the proposed system records the operating status, such as the type of problem detected and the operating state of the virtual server, along with the time in the system log file. In addition, the operating state of the background job used to perform the recovery job for the target server, which is shown in Fig. 12, is also recorded in the log file. In the proposed system, the log data recorded by the foreground job and those recorded by the background job are saved in the same log file. For that reason, in the log data recorded by the background job, the string "_ _" is added to the content top in order to characterize the log data.

Figure 14 shows the log file contents used for measurement in the experiment, including the server function recovery time, the target server recovery time, and the server access disconnection time from the client. The system log file records the details of failure detection and recovery processing. The connection status related to server access is recorded in the client access log file. Because this system log file is used for experiments, numerical values or the word "Wait" is respectively recorded at one-second intervals during the monitoring interval or waiting times. The status "OK", which represents the condition in which services are offered on the target server, and the status "XX", which represents the condition in which services cannot be offered, are recorded in the client access log file together with the time.

The target server, which is a real server, is an FTP server. The IP address is denoted as IPR, and the virtual IP address is denoted as VIP. The virtual server IP address is denoted as IPV. The monitoring interval time is 10.00 s in the experimental system, and the problem is generated at the midpoint (5.00 s after startup) of the monitoring interval. The problem related to the network is reproduced by shutting down the FTP server.

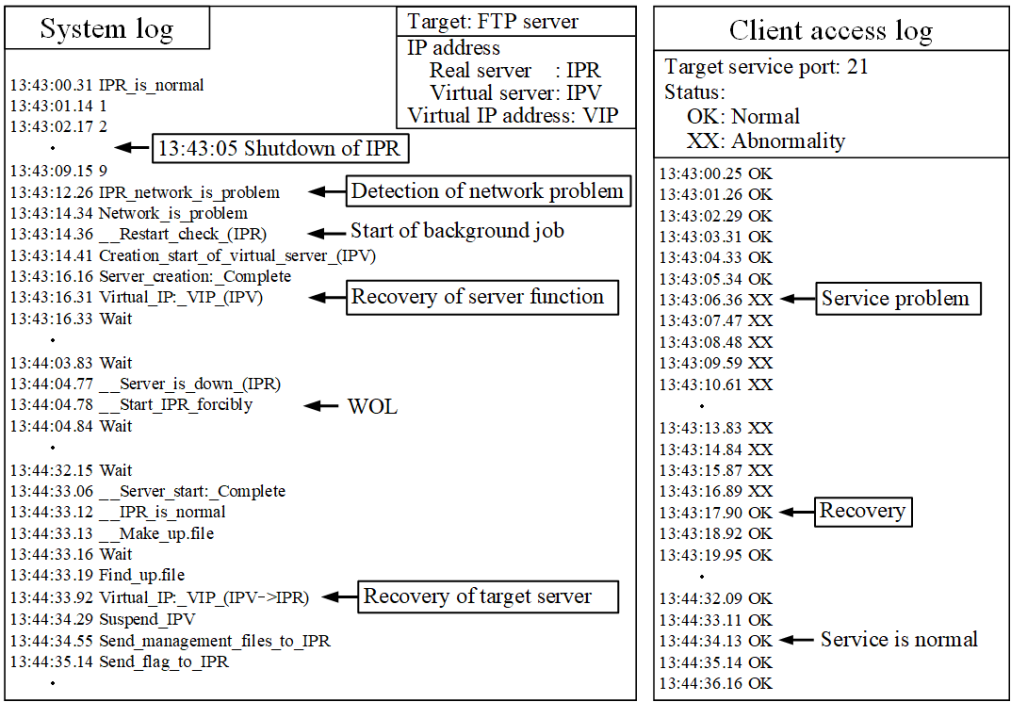


Fig. 14: Log files used to record the operating status.

In the system log file, the problem occurs at 13:43:05 in the inter-monitoring time, and the network problem is detected at 13:43:12.26. Virtual server creation begins at 13:43:14.41 after reexamining the network problem at 13:43:14.34, and creation is completed at 13:43:16.16. The server function is recovered at 13:43:16.31 after setting the virtual IP address. The difference between the time at which the server function is recovered and the time at which the network problem is detected is the server function recovery time. In the client access log, the problem in the server is detected at 13:43:06.36, and the service is recovered at 13:43:17.90. This difference is the server access disconnection time.

In the system log file, the target server recovery job is started at 13:43:14.36 by the background job. The target server is determined to be in a stop condition at 13:44:04.77, and a forcible start is attempted by the WOL function. Thereafter, the target server is determined to be started normally at 13:44:33.12. In order to inform the management program running in the foreground, an up.file is created, and the background job is terminated. When the up.file is detected in the foreground job, the virtual IP address of the virtual server is set in the target server. As a result, the target server is recovered at 13:44:33.92. The difference between the time at which the target server is recovered and the time at which the network problem is detected is the target server recovery time. Although the target server recovery job is performed at 13:44:33.92, the abnormality of the service offer state is not found at 13:44:34.13 in the client access log file. Finally, the flag file is sent to the target server at 13:44:35.14 after the management files are sent to the target server at 13:44:34.55.

5.3. Experimental results in the case of target server failure

Next, experiments to reproduce the problem on the network or the service on the target server FTP are performed in the experimental system shown in Fig. 13. The mail server, which is the management server, manages the FTP server. The monitoring interval time is 10.00 s, and a network or service problem is reproduced at the midpoint (5.00 s after startup) of the monitoring interval. The recovery time required to deal with the problem(s) is measured, and the results are listed in Table 2. Each measured time is the average value of 10 experimental results. As shown in the table, two problem types are defined. St represents the condition in which the service function is recovered by restarting the target server, while Nt represents the condition in which the target server is shut down.

Table 2: Recovery times in the case of problems in one or two servers.

St: Service program problem (Recovery by server restart)
Nt: Shutdown of target server

Problem in one server

Recovery time (s) / Problem list		Server function (Target server)		Access disconnection time (Client)	
		Proposed method	P2P method	Proposed method	P2P method
Service program	St	5.07 (33.23)	5.24 (33.48)	10.61	10.65
Network	Nt	4.05 (81.74)	4.13 (82.04)	11.64	11.66

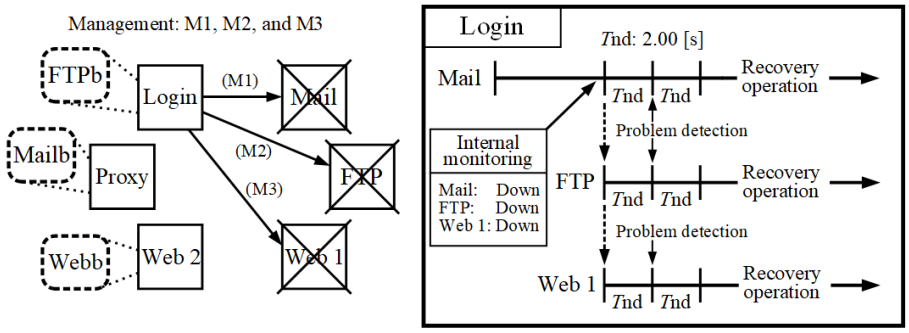
Problem in two servers

Recovery time of server function (s)		Value: Proposed method / (Value): P2P method Target server: Web 1	
		St	Nt
Target server: FTP	St	5.39 (5.46) 5.18 (5.19)	4.06 (4.20) 5.12 (5.21)
	Nt	5.19 (5.31) 4.06 (4.11)	4.21 (4.54) 4.07 (4.14)

"Server function" in the upper table (Problem in one server) indicates the time until the server function is recovered by the virtual server after the problem has occurred. The client accesses the server in one-second intervals, and the "Access disconnection time (Client)" in the table indicates the time until the access by the client is recovered from the server access disconnection state. The table shows the measurement time results for both the proposed and P2P methods, and the numerical values in parentheses indicate the target server recovery time. The target server recovery time is the difference time between the time at which the problem is detected and the time at which the target server is recovered. The proposed method has the same server function recovery time as the P2P method in any problem, and the access disconnection time at the client is also the same. The target server recovery time is also almost the same in both methods.

The recovery time for the case in which the problem occurs simultaneously on two servers is listed in the lower table (Problem in two servers). In the proposed method, the mail server, which is the management server, manages the FTP and Web 1 servers. In the P2P method, the mail server, which is the (P1) management server, manages the FTP server and the Web 2 server, which is the (P2) management server, manages the Web 1 server. Here, the monitoring interval time is 10.00 s, and the problem (which is St or Nt) is reproduced in the two servers at the midpoint (5.00 s after startup) of the monitoring interval. Two types of problems are combined for the two target servers, and four types of experiments are performed. The numerical values not enclosed in parentheses indicate the proposed method results, while the numerical values enclosed in parentheses indicate the P2P method results. There is a slight delay in the server function recovery time compared to the one server failure, which is due to the time required to deal with simultaneous failures such as the management server selection processing necessary to create the virtual server.

Figure 15 shows the recovery time in the case of simultaneous failure of the mail, FTP, and Web 1 servers. Here, Nt (shown in Table 2) is reproduced simultaneously for three servers. The server function recovery time is the time from the failure detection to the recovery. The monitoring interval time is 10.00 s, and Nt is reproduced simultaneously to three servers at the midpoint (5.00 s after startup) of the monitoring interval. Additionally, *T*nd, which is "Network examination time (Disconnection)" shown in Table 1, is the network examination time when the target server network is abnormal.



Problem: Nt (Shutdown of target server)

Target server	Recovery time of server function (s)		Access disconnection time (s)	
	Proposed method	P2P method	Proposed method	P2P method
Mail	4.44	4.30	11.73	11.44
FTP	4.06	4.23	11.30	15.77
Web 1	4.26	4.15	11.52	11.26

Fig. 15: Recovery times for cases in which a problem occurs simultaneously in three servers.

In the case of the proposed method, as shown in the figure, the login server, which is the management server, performs external monitoring of the FTP, Web 1, and mail servers based on the internal monitoring results for each server. A problem in three servers is detected, and the server functions are recovered. From the experimental results, we can confirm that the server function recovery time for the FTP server is the shortest, followed by Web 1 and mail. The difference is considered to have

occurred in the internal monitoring result because the login server first executes the management program for FTP, then Web 1, and finally the mail server. A similar tendency can be seen in the access disconnection time at the client.

In the P2P method, for the case in which network problem Nt occurs simultaneously in three servers, if the login server detects a network problem in the mail server, the management program is started to manage the FTP server after a lapse of *T*nd. Thereafter, the program detects network trouble in the FTP server after a lapse of *T*nd. Because the FTP server that manages the Web 1 server is in a failed state, the Web 2 server manages Web 1. The server function recovery time is almost the same for all three servers. However, the FTP server has the longest access disconnection time at the client due to the time difference in failure detection.

6. Comparison of proposed method and general server management systems

High server system availability is important for the stable provision of Internet services. Figure 16 shows common system configurations used for realizing high server system availability along with the configuration of our proposed method. Specifically, Figs. 16(a) and 16(b) show the outline of failover cluster and load balancer cluster systems, respectively. In a cluster system, multiple servers are linked and operated as one system, so even if one server fails, other servers continue to operate. In Fig. 16(a), we can see that if Server 1 fails, Server 1b (which was in the standby state) continues to provide service. In the load balancer system shown in Fig. 16(b), we can see that all three servers are connected to the client via the load balancer, and if a server fails, it is automatically deselected. To operate both systems (a) and (b), it is necessary to execute management software for each cluster. This means both systems need a server that will efficiently collect and examine information on the operating status of each cluster. Although these methods are suitable for sustaining high availability, they also result in cost and power consumption issues because the basic configuration requires all servers to be capable of providing the same services.

Figure 16(c) shows a general server management system that uses a dedicated management server. Here, the management server grasps the operating status of the target server that provides services to clients and provides a recovery function when a target server fails. In this setup, each target server is configured with the ability to provide different services to clients, and the management server system software and operating status of each target server are managed on one management server. However, the most significant disadvantage of this system is that management cannot proceed if the management server fails. Furthermore, since the management server management load increases as the number of target servers rises, it is necessary to configure a redundant management server and create a high resource environment, which raises complicated control and cost concerns.

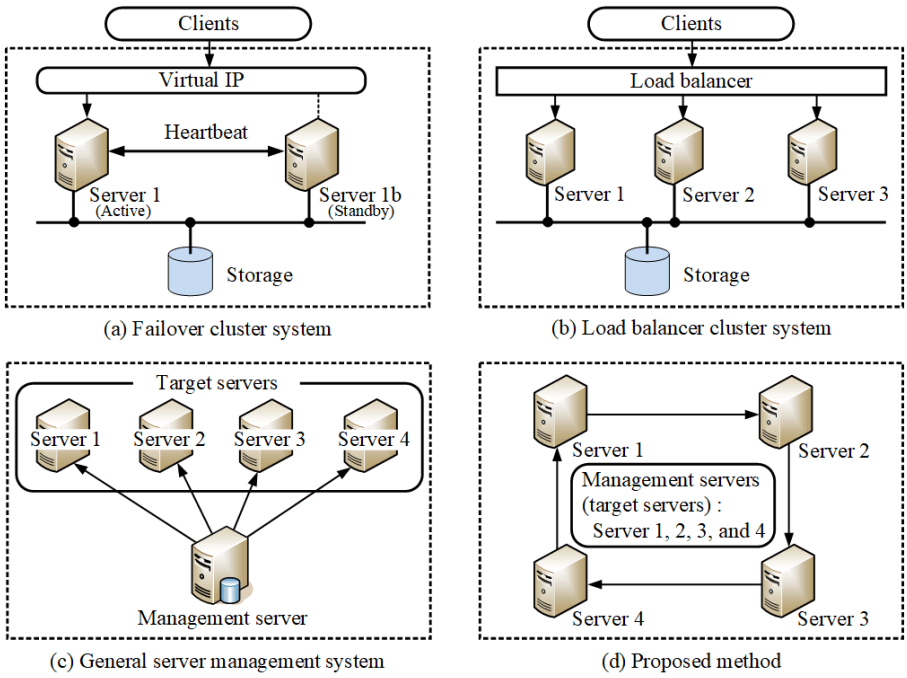


Fig. 16: Server system management method.

To solve the problems in systems (a) through (c), the P2P server management system has been reported as a system that does not require a dedicated management server. However, since the complexity of the required system software also poses a problem, we proposed the method shown in Fig. 16(d) to resolve it. As shown in the experimental results in Section 5, our proposed system has the same high-performance availability as the P2P method. Here, since each target server is configured to provide different services to clients, a server is required for providing centralized software and management server data management, as well as for collecting operation information from each management server. The proposed system achieves this by using a file server to provide service redundancy.

Systems (a) through (d) basically operate automatically, and the administrator is only required to monitor their statuses. Therefore, there are no significant differences between normal monitoring and high-availability (safety) operations. However, in situations where a problem occurs in the management system, countermeasures are easier in system (d) because the management program is simpler than those of systems (a), (b), and (c). Furthermore, our proposed system is superior to the others in terms of cost.

7. Conclusions

Herein, we proposed a dynamic backup server system (DBSS) that adopts a ring-type configuration that does not require a dedicated management server and does not complicate the management program. We then showed an overview of the proposed system, management file details, and the additional management program components necessary to construct the system. In addition, the operation flows for the adopted synchronization system and an executor system needed to prevent the dispersion of management programs were shown. Furthermore, we proposed and adopted a dual monitoring method that prevents delays in dealing with multiple simultaneous server failures and dynamic management extension and automatic connection selection methods that can efficiently select the server management configuration.

An experimental system that adopted this method and the P2P method was constructed, and the configuration and the specification were shown, after which experiments to reproduce the problems of the service-providing program and the network were conducted. In those experiments, the recovery times of the target server functions and the target server when various problems occurred in one or multiple target servers were measured.

From the obtained results, it was shown that a server management system that does not require a dedicated management server is effective for efficiently achieving a high-availability server system, and experiments have shown that our proposed method provides the same performance level as the previously reported P2P method [23],[24] in terms of recovery time when a server fails. Additionally, it was shown that the management program of our proposed method has a simpler construction than the P2P method and that compared to the most commonly adopted server management systems, our proposed system can keep server systems costs low and availability high. These results, in turn, indicate that our proposed system can provide stable Internet services at a lower cost, thereby facilitating the expansion of various Internet services and contributing to making our future lives more convenient.

References

[1] Ministry of Economy, Trade and Industry, https://www.meti.go.jp/press/2021/08/20210831005/20210831005-2.pdf, 2021.11.8.

[2] Information-technology Promotion Agency (JAPAN), https://www.ipa.go.jp/files/000093706.pdf, 2021.11.10.

[3] Ministry of Internal Affairs and Communications, https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/pdf/02honpen.pdf, 2021.10.7.

[4] Z. Li and Y. Yang, “RRect: A Novel Server-Centric Data Center Network with High Power Efficiency and Availability,” IEEE Transactions on Cloud Computing, Vol. 8, No. 3, 2022, pp. 914-927.

[5] 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.

[6] D. Moro, G. Verticale, and A. Capone, “A Framework for Network Function Decomposition and Deployment,” IEEE Conf. Proc., Vol. 2020, No. DRCN, 2022, pp. 1-6.

[7] N. Maksic, “Topology Independent Multipath Routing for Data Center Networks,” IEEE Access, Vol. 9, 2021, pp. 128590-128600.

[8] Q. Xun, W. Li, H. Guo, and Y. Wang, “Enabling Lightweight Network Performance Monitoring and Troubleshooting in Data Center,” IEEE Conf. Proc., Vol. 2021, No. INFOCOM WKSHPS, 2021, pp. 1-2.

[9] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, “A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications,” IEEE Access, Vol. 9, 2021, pp. 41731-41744.

[10] X. Wang, Y. Li, Y. Chen, S. Wang, Y. Du, C. He, Y. Zhang, P. Chen, X. Li, W. Song, Q. Xu, and L. Jiang, “On Workload-Aware DRAM Failure Prediction in Large-Scale Data Centers,” IEEE Conference Proc., Vol. 2021, No. VTS, 2021, pp. 1-6.

[11] C. Kari, S. Chen, A.-M. Sepehr, and V. Pallipuram, “Data Migration in Large Scale Heterogeneous Storage Systems with Nodes to Spare,” IEEE Conf. Proc., Vol.2019, No. ICNC, 2019, pp. 854-858.

[12] J. Zhang, K. Zhou, P. Huang, X. He, M. Xie, B. Cheng, Y. Ji, and Y. Wang, “Minority Disk Failure Prediction Based on Transfer Learning in Large Data Centers of Heterogeneous Disk Systems,” IEEE Trans. on Parallel and Distributed Systems, Vol. 31, No. 9, 2020, pp. 2155-2169.

[13] A. Nasir, T. Alyas, M. Asif, and M. N. Akhtar, “Reliability Management Framework and Recommender System for Hyper-converged Infrastructured Data Centers,” IEEE Conf. Proc., Vol. 2020, No. iCoMET, 2020, pp. 1-6.

[14] A. A. Khatami, Y. Purwanto, and M. F. Ruriawan, “High Availability Storage Server with Kubernetes,” IEEE Conf. Proc., Vol. 2020, No. ICITSI, 2020, pp. 74-78.

[15] M. Data, D. P. Kartikasari, and A. Bhawiyuga, “The Design of High Availability Dynamic Web Server Cluster,” IEEE Conf. Proc., Vol. 2019, No. SIET, 2019, pp. 181-186.

[16] S. Yanase, S. Masuda, F. He, A. Lawabata, and E. Oki, “Heuristic Approach to Distributed Server Allocation with Preventive Start-Time Optimization against,” IEICE Trans. on Commun. (Web), Vol. E104.B, No. 8, 2021, pp. 942-950.

[17] N. Georgoulopoulos, A. Hatzopoulos, K. Karamitsios, I. M. Tabakis, K. Kotrotsios, and A. I. Metsai, “A Survey on Hardware Failure Prediction of Servers Using Machine Learning and Deep Learning,” IEEE Conf. Proc., Vol. 2021, No. MOCAST, 2021, pp. 1-5.

[18] 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.

[19] M. Ljubojevic, A. Bajic, and D. Mijic, “Implementation of High-Availability Server Cluster by Using Fencing Concept,” IEEE Conf. Proc., Vol. 2019 No. INFOTEH, 2019, pp. 1-5.

[20] 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.

[21] 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.

[22] 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, pp. 34-54.

[23] 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.

[24] M. Kitamura and K. Tani, “Development of File Management System for a Peer-to-Peer Method Server Management System,” JISSJ, vol. 16, no. 1, 2020, pp. 1-16.

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 Associate Professor in 2022. His current interests are the optimum design and construction of server-client systems and the analysis of various characteristics of server systems.

Toshikazu TAKESHITA

He received his M.E. degree in 2022 from Tokyo Polytechnic University, where he is currently in charge of the design and construction for server systems at NEC Fielding, Ltd.

Tatsuya OKAZAKI

He received his B.E. degree in 2022 from Tokyo Polytechnic University, where he is currently working toward the M.S. degree. He is engaged in developing software for experimental systems and analyzing performance data.