［Original Paper］

Development of Power-saving Server System not requiring a Specific Management Server

Mitsuyoshi KITAMURA†, Toshikazu TAKESHITA†, and Takumi YAMAGUCHI†

† Graduate School of Engineering, Tokyo Polytechnic University

Abstract

Power-saving server systems (PSSs) conserve power by using dedicated management servers to perform load monitoring and measurement processing for the virtual servers that provide services to clients, and by controlling both real and virtual servers to optimize server system configurations. However, because such systems must be capable of recovering successfully if a management server failure occurs, PSS control methods tend to become complicated, and management loads tend to increase according to the number of targeted servers. With those points in mind, the present study proposes a PSS that does not require a dedicated management server. Herein, the characteristics of the conventional and proposed methods used in PSSs are described and compared. To solve the problems caused by adopting the proposed method, executor and synchronization systems are introduced, and details of their operation methods are presented. In addition, power-saving states are set for two virtual server configurations (redundant and non-redundant) that can be selected based on actual server system operating states. Detailed operations governing load measurement and configuration changes in the PSS are also shown, and it is demonstrated that processing will continue even if the targeted virtual server fails during operation. Furthermore, an experimental system using the proposed method is constructed, and its architecture, specifications, and detailed operation procedures are described. In the power-saving condition, the proposed system operates at approximately 72% of the power consumption seen under normal conditions when the virtual servers are configured redundantly, and at approximately 40% of normal power consumption when a non-redundant configuration is used. Therefore, it is expected that the operating costs and operation loads of the management and target servers can be reduced by the power saving effects and failure avoidance countermeasures of our proposed system.

1. Introduction

The spread of the novel coronavirus in 2020 has resulted in a worldwide pandemic [1]. Thus, as part of efforts to reduce person-to-person contact, there are growing trends aimed at promoting teleworking and online education, which commonly utilize Internet-based information systems. Because these information systems consist primarily of networks and server systems, it is important to study optimal server system construction methods and to investigate network availability and speedup efforts [2].

A number of important studies related to reliability improvements in data centers have been conducted. These include a hierarchical modeling framework that can be used to evaluate data center network reliability and availability [3], a framework for automating the disassembly and placement of network functions according to traffic demands and network topology [4], and an integrated design that considers cloud service problems such as single link failures, service failures on data center networks, and data center network placement [5]. Additionally, in order to solve the data migration problems that result from storage performance mismatches in large-scale storage systems such as data centers, efforts have been made to introduce data transfer bypass nodes that improve migration performance levels [6], and a minority disk failure prediction model based on a transfer learning approach has been proposed to improve the predictive performance for minority disks in large data centers [7]. Furthermore, with the help of neural networks, efforts have been made to measure the reliability of networking, storage, and computational resources based on their availability, outage, and downtime [8].

Servers are the fundamental units of systems providing services in data centers, and various ideas aimed at ensuring that such systems have high availability have been reported. These include the failover and load balancer cluster systems that are commonly used to provide server system redundancy, in order to help deal with server failures. However, cluster systems that are used in real servers require two or more real servers to provide the services executed by one server. Therefore, a problem with cluster systems is their high costs. Thus, multiple server backup systems (MSBSs) that can back up the functions of several real

[Original Paper] Received 28 July 2020, revised 11 December 2020, accepted 11 February 2021.

servers using virtual servers started from one real server have been proposed [9].

Additionally, various reports have discussed high-availability server clusters with greatly improved service reliability have been achieved by the adoption of fencing technology [10], dynamic web server cluster design using Nginx, php-fpm, MariaDB, GluterFS, and keepalived [11], and highly available distributed clusters that can dynamically change configurations while maintaining services based on scalability and traffic rates [12].

In general, server management systems use dedicated management servers for monitoring the target servers. However, because such systems must be capable of recovering successfully if a management server failure occurs, management server control methods tend to become complicated, and management loads tend to increase according to the number of target servers. Therefore, we have previously proposed a peer-to-peer (P2P) server management system that does not require a dedicated management server as a way to manage a real server system [13], and a P2P method that is equipped with enhanced processing methods needed to recover virtual server functions [14]. When adopting this method in server management systems, even if the management server fails, other servers can assume its role and continue to perform management functions because each target server can also function as a management server.

When constructing a server system in a data center, it is necessary to pay particular attention to power saving in addition to high speed and security issues [15]. A report by the Ministry of Economy, Trade and Industry (METI) of Japan has estimated that, in comparison with 2006 levels, power consumption by information technology devices will increase nine-fold by 2025 [16]. Therefore, a number of important studies on power-saving network systems has been conducted [17-20]. Unfortunately, however, even though approximately half of all information technology devices now in use are network devices and servers, little research has been conducted on power saving in server systems.

To date, the reports that address power savings in server systems have focused on device power consumption in order to determine optimum server arrangements. This primarily involves using a server relocation service and ensuring optimal service performance levels while considering energy efficiency by controlling server allocation through the implementation of the Erlang-based approach [21-22]. Additionally, in order to simplify the power-saving and high-availability control programs employed when constructing a server system, we have proposed a method by which those control programs are operated alternately and independently, and a compound operation method for those control programs that adopts a system interface [23-25].

However, in any conventional server system control method aimed at power saving, a dedicated management server manages the target servers, which means the system must be capable of recovering successfully if a management server failure occurs. As a result, management server control methods tend to become complicated, and management loads tend to increase according to the number of target servers.

With those points in mind, the present study proposes a power-saving server system (PSS) that does not require a dedicated management server. Initially, the characteristics of the conventional method (which does require a dedicated management server) and the proposed method are shown and compared. Then, to solve the problems caused by adopting the proposed method, we introduce executor and synchronization systems and describe the details of their operation. In addition, power-saving states are set using redundant and non-redundant virtual server configurations while considering actual server system operating states. Details of governing load measurements and configuration changes in the PSS are shown, and it is demonstrated that processing will continue even if the targeted virtual server fails during operation. Furthermore, an experimental system using the proposed method is constructed, and its architecture, specifications, and details of its operation are described.

The remainder of the present paper is organized as follows. Comparisons between the conventional and proposed PSS methods are shown in Section 2, where we also review of power-saving server configuration types. Details of the executor and synchronization system that are necessary to solve the problems that result from adopting the proposed method are also provided. Section 3 describes details of the flow from load measurements to system configuration optimization for each server, along with the processing used in the event of a target server failure. Section 4 describes the construction and specifications of our experimental system, the power-saving efficiency of the PSS, and the time required for configuration changes. Finally, we provide our conclusions in Section 5.

2. PSS outline

2.1. Comparison of the proposed and conventional methods

In Fig. 1, we compare the proposed PSS method with the conventional method. Here, M, W, and F are virtual servers, and RS is the real server. The virtual and real servers are connected to the file server through a network file system (NFS) connection. We refer to a server group that provides each service to clients as a service group. M1 and M2 combine to create a mail service group, W1 and W2 combine to create a web service group, and F1 and F2 combine to create a File Transfer Protocol (FTP) service group. The virtual servers belonging to each service group are configured redundantly, and round-robin client access to redundant servers is performed via a load balancer.

The management server conducts target server load measurements, and the measured loads are aggregated by each service group. Power savings are realized by changing the server configuration based on the loads. Figure 1(a) shows the conventional method, which uses a dedicated management server to manage multiple target servers. In the proposed method, each of the virtual servers, which are the targets, is equipped with a management server function, as shown in Fig. 1(b). A synchronous system is adopted because all the virtual servers need to operate at the same time. Moreover, in consideration of management program dispersion related problems, an executor system is adopted as well.

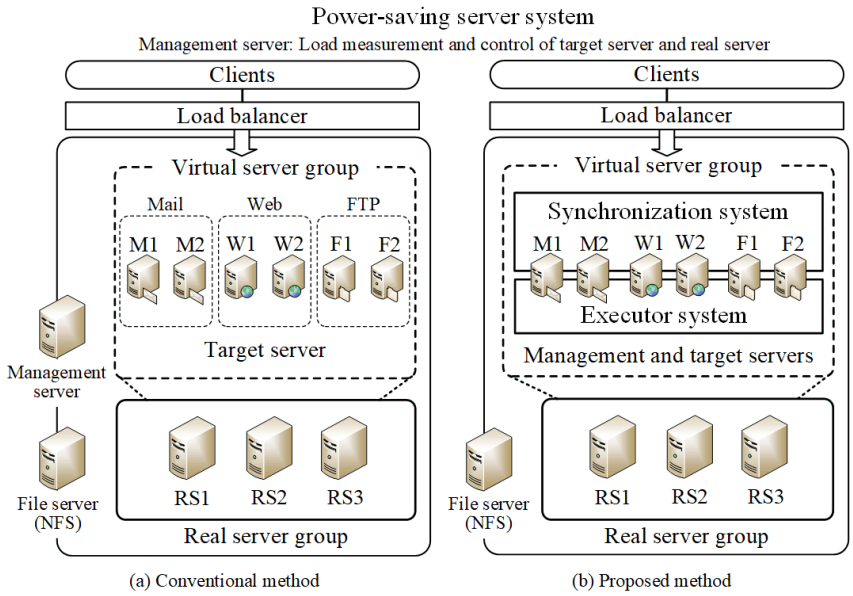


Fig. 1: Comparison of conventional and proposed methods for power-saving server systems.

2.2. Power saving by changing the server system configuration

In the proposed system, in which each of the targeted virtual servers has a management server function and performs load measurements, server system configuration changes are conducted based on the measured load. Figure 2 shows how power saving is realized by changing the server system configuration. In this figure, M, W, and F are virtual servers, and Ws and Fs are spare servers that supplement the service functions provided to clients on a base server. RS1, RS2, and RS3 are real servers, and VIP refers to a Virtual Internet Protocol (IP) address. There are two power-saving states: light load condition 1 (PS1) and light load condition 2 (PS2). RS1 and RS2 are used under the PS1 condition, but only RS1 is used under the PS2 condition.

At startup, the server system is operated under a moderate load condition in which all the real servers have been started, and the virtual servers belonging to each service group are redundantly configured, as shown in Fig. 2(a). If the load is determined to be the light in a case where the server is in the moderate load condition, the server system configuration shifts to the PS1 condition. However, if the number of accesses from clients to the server becomes low under the PS1 condition, the server system configuration shifts to the PS2 condition.

Figure 2(b) shows a case where the mail and FTP service group loads are determined to be light. In the case of the mail

service group, the state does not change because both M1 and M2 are created on the base server. In the case of the FTP service group, because RS3 is not set as the base server, Fs, which is the spare server, is created on the base server RS1. VIP6, which is set to F2, is released and set to Fs. Once VIP6 has been released from F2, F2 assumes a state that is not accessed from the load balancer. We refer to this state as an access stop state.

Figure 2(c) shows a case where all the service groups are determined to be under light load conditions. Here, Fs is created on RS1 and Ws is created on RS2. Thereafter, VIP5 and VIP6, which were created on RS3 and set to W2 and F2, respectively, are set to Ws and Fs, respectively. Because W2 and F2 are in the access stop state, RS3 is set to the suspended state, and power saving is realized. Here, the virtual servers belonging to each service group continue processing while maintaining the redundant state.

Figure 2(d) shows a case where client access to the mail and web service groups is low under the PS1 condition. VIP4 and VIP5, which were created on RS2 and set to M2 and Ws, respectively, are set to M1 and W1, respectively. As a result, M2 and Ws are in the access stop state.

Figure 2(e) shows a case where access to all service groups is low in the PS1 condition. VIP4, VIP5, and VIP3, which were created on RS2 and set to M2, Ws, and F1, respectively, are set to M1, W1, and Fs, respectively. Because M2, Ws, and F1 are in the access stop state, RS2 is set to the suspended state and power saving is realized. Here, the power consumption is approximately two-thirds of that seen in Fig. 2(c) and approximately one-third of that seen in Fig. 2(e) when compared with the moderate load condition shown in Fig. 2(a).

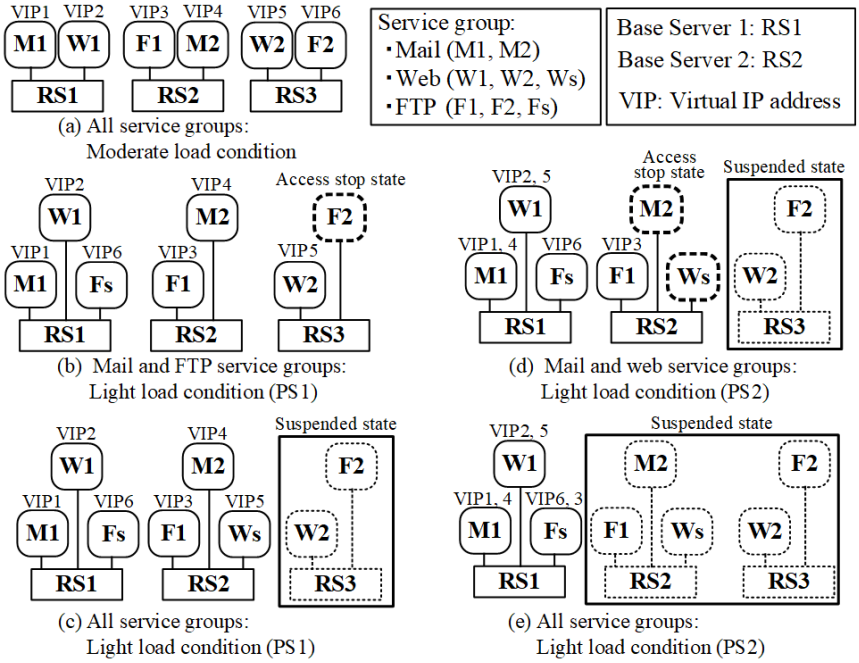


Fig. 2: Power saving realized by changing server system configuration.

2.3. Management file

The management file format for the PSS is listed in Table 1. These configuration files are edited by the PSS. In order to prevent malfunctions due to conflicted management file editing, a lock function, which rejects access from other programs during editing, is adopted. Detailed information on the virtual servers accessed from clients is recorded in the operation file, which contains the number, service group name, IP address of the real server that created the virtual server, virtual IP address, hostname of the virtual server, and IP address of the virtual server.

Detailed information on the spare servers is recorded in the spare server file, and detailed information on the backup servers, which recover functions for failed servers, is recorded in the backup server file. The spare and backup server files have the same

format as the operation file. In these files, the IP address of the real server and the virtual IP address are not recorded because they are determined during operation. When the spare or backup server is created on the real server, information of the created server is copied to the operation file, and then the real server IP address and the virtual IP address are added to the information in the file.

The access stop file and the failure file have the same format as the operation file. The information for the server in the access stop state shown in Fig. 2 is moved from the operation file to the access stop file, and is then returned to the operation file when the stopped server returns to the normal state. Virtual server information that is determined to be faulty is moved from the operation file to the failure file. The real server information used to create the virtual server is recorded in the real server file, which also contains the media access control (MAC) address and IP address of the real server. Here, the MAC address is used for reactivating a real server in a suspended state. The state of each service group (Moderate, PS1, or PS2) is recorded in the state file, which also contains the service group name, group state, and the IP address of the base server used to gather the virtual servers. Because multiple base servers are needed in the PS1 condition, multiple base server IP addresses are recorded.

Table 1: Management files.

Operation file

No.	Service group	Real server IP address	Virtual IP address	Virtual server	
				Hostname	IP address
1	Mail	172.21.14.201	172.21.14.101	M1	172.21.14.1
2	Web	172.21.14.201	172.21.14.102	W1	172.21.14.2
⋮					

Spare server file / Backup server file

Service group	Real server IP address	Virtual IP address	Virtual server	
			Hostname	IP address
FTP			Fs	172.21.14.53
⋮				

Access stop file / Failure file

No.	Service group	Real server IP address	Virtual IP address	Virtual server	
				Hostname	IP address
⋮					

Real server file

Real server IP address	MAC address
172.21.14.201	00:00:00:00:00:01
172.21.14.202	00:00:00:00:00:02
⋮	

State file

Service group	Load condition	Base server IP address	
		Base Server 1	Base Server 2
Mail	Moderate	172.21.14.201	172.21.14.202
Web	Light (PS1)	172.21.14.201	172.21.14.202
⋮			

2.4. Executor system

A particular problem related to the proposed method is the dispersion of the management program, because it requires the management program to be installed on all of the management servers. To resolve this problem, an executor system was developed and implemented. The construction and functions of the executor system are shown in Fig. 3. Each management server is connected to the file server through an NFS connection and has a shared area. With the executor program installed on each management server, each server can access the management program and Synchronization Programs 1 and 2 on the shared area via the NFS connection.

Here, Synchronization Program 1 is used for load measurement, and Synchronization Program 2 is used to calculate the number of accesses to the server. The executor program, which is automatically executed when the management server is started, determines if a flag file has been saved to a local disk on the server executing the executor program. If a flag file exists, the executor program copies the management program and Synchronization Programs 1 and 2, which are saved in the shared area, to the local disk, and then executes both the management and synchronization programs. If the flag file has been deleted, the executor program stops all management and synchronization programs and then stops the management server. The executor system can install a function for dealing with updating and modifying the programs on each management server by editing the program on the file server. As a result, the executor system can resolve the management program dispersion problem.

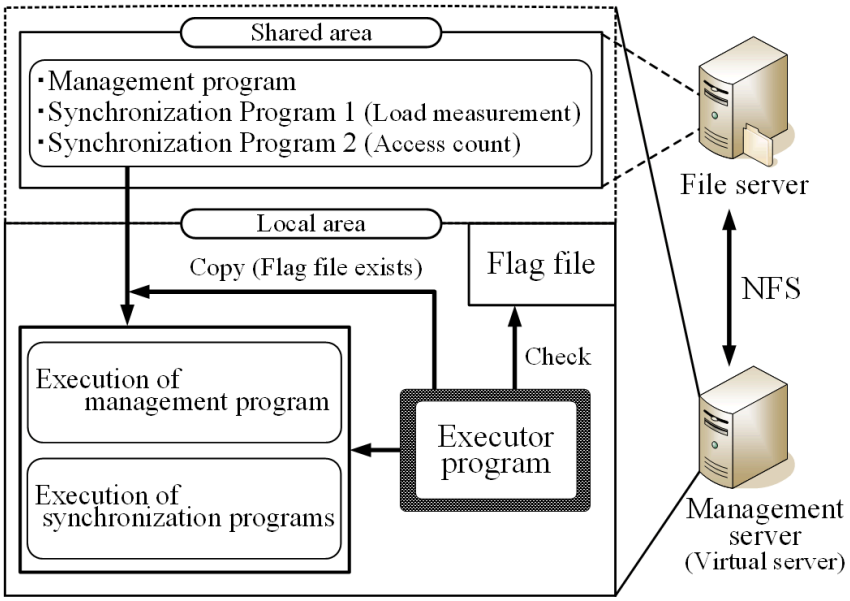


Fig. 3: Construction of executor system.

2.5. Synchronization system for load measurement

Figure 4 shows the details of Synchronization System 1 to be used for load measurements. This system first determines if a Stop.fil file is located in the shared area. If the Stop.fil file exists, the synchronization system enters a wait state. Although the proposed system does not have a dedicated management server, there are situations where one virtual server must control all of the real and virtual servers. In such cases, the server that performs this control is dynamically set based on the operation file.

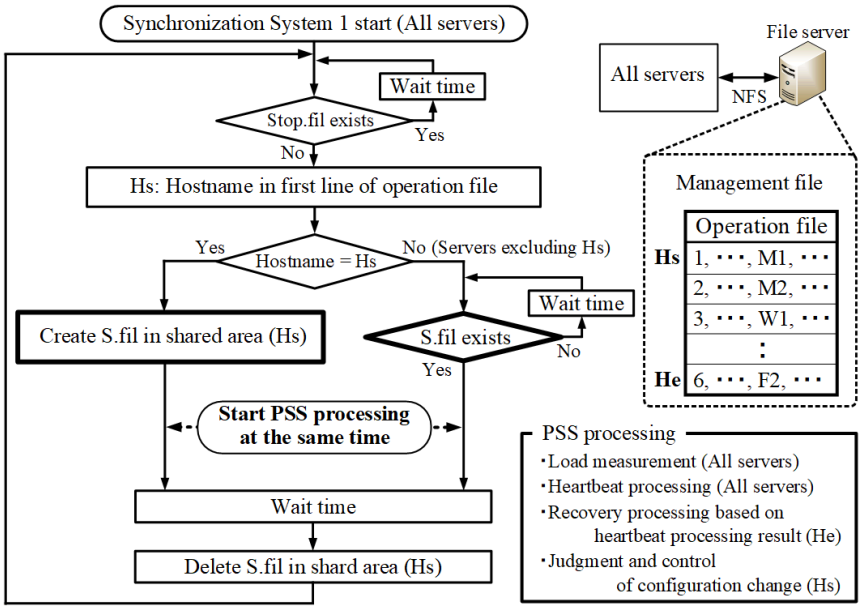


Fig. 4: Synchronization system for load measurement.

The name of the operating virtual server is recorded in the operation file. The server created on Base Server 1 and recorded at the beginning of the operation file is selected as Hs. In Fig. 4, virtual server M1 is selected as Hs. The servers (with the exception of M1) are in a wait state until an S.fil file, which is a synchronization file, is created. When M1 creates the S.fil file in the shared

area, all of the servers start PSS processing at the same time, after which Synchronization System 1 enters the wait state.

If the server system configuration is not changed, PSS processing is shorter than the waiting time. When the waiting time ends, M1 deletes the S.fil file. Moreover, heartbeat processing is also performed simultaneously with load measurement during the PSS processing. The server recorded at the end of the operation file is selected as He. In the figure, virtual server F2 is selected as He and performs recovery processing based on the heartbeat processing results.

3. Details of PSS processing and configuration change

3.1. Load and number of accesses

The load value of the virtual server belonging to each service group is used to shift from the moderate load condition to the PS1 condition or the reverse. Memory load measurement is performed in each virtual server using the UNIX "free" command. Central processing unit (CPU) and network load measurements are performed in each virtual server using the UNIX "sar" command. The proposed system performs load calculations based on the predicted load and the server type [13].

The flow calculation for the load on each server is shown in Fig. 5. In this system, all servers (instead of a dedicated server) perform measurements and calculations for the loads shown in the figure. Based on these measured values, the average available memory size (Am), the average CPU idle rate (Ac), and the average number of bytes transferred by the network (An) are calculated periodically. The difference in the average load value acquired at two different times is then used to calculate the predicted load. The predicted memory load L_{m} (in kilobytes) is defined as follows:

$$L_{\mathrm{m}}\ (\mathrm{KB}) = 2 \times A\mathrm{m}_n - A\mathrm{m}_{n\text{-}1}. \quad (1)$$

The predicted load L_{c} (in percent) for the CPU is given as follows:

$$L_{\mathrm{c}}\ (\%) = 2 \times A\mathrm{c}_n - A\mathrm{c}_{n\text{-}1}. \quad (2)$$

The predicted network load L_{n} (in kilobytes) is defined as

$$L_{\mathrm{n}}\ (\mathrm{KB}) = 2 \times A\mathrm{n}_n - A\mathrm{n}_{n\text{-}1}. \quad (3)$$

Because the predicted loads obtained from equations (1), (2), and (3) may have negative values and exceed the upper limits due to fluctuations, the minimum values are set to zero, and the maximum values are the set memory size of the virtual server for L_{m}, 100% for L_{c}, and the rated transfer speed × 2 for L_{n}. The mail service group is used as an example for calculating the loads in each service group. The memory load L_{ma} (in percent) is defined as follows:

$$L_{\mathrm{ma}}\ (\%) = \frac{L_{\mathrm{m\_M1}} + L_{\mathrm{m\_M2}}}{M_{\mathrm{max\_M1}} + M_{\mathrm{max\_M2}}} \times 100, \quad (4)$$

where M_{max} is the maximum memory size. The CPU load L_{ca} (in percent) is given by

$$L_{\mathrm{ca}}\ (\%) = \frac{L_{\mathrm{c\_M1}} \times C_{\mathrm{cor\_M1}} + L_{\mathrm{c\_M2}} \times C_{\mathrm{cor\_M2}}}{C_{\mathrm{cor\_M1}} + C_{\mathrm{cor\_M2}}}, \quad (5)$$

where C_{cor} is the number of CPU cores. The network load L_{na} (in percent) is calculated by

$$L_{\mathrm{na}}\ (\%) = \frac{L_{\mathrm{n\_M1}} + L_{\mathrm{n\_M2}}}{N_{\mathrm{rat\_M1}} + N_{\mathrm{rat\_M2}}} \times 100, \quad (6)$$

where N_{rat} is the total number of bytes transferred per second for the network transmission and reception rating. Here, if a router rated at 1,000 Mbps is taken as an example, because 128×10^3 KB can be transferred in 1 s, N_{rat} is 256×10^3 KB from the total value of transmission and reception.

The server type method is introduced to calculate the load of each service group. This method defines a CPU type server as one that mainly uses a CPU, a NETWORK type server as one that mainly provides data, and a CPU_NET type server as one that both uses a CPU and provides data. When a general server is taken as an example, the login server is defined as a CPU type, and the FTP and mail servers are defined as NETWORK types. A web server is defined as a CPU_NET type server because it provides Hypertext Markup Language (HTML) data, image data, and Common Gateway Interface (CGI) data.

Here, L_{ma}, L_{ca}, and L_{na} are weighted according to the server type, and the total obtained from these values is set to 100, at maximum. Because the memory load L_{ma} is important for every server type, L_{ma} has a weight of 0.5. The load in the service group varies depending on its type. The load for a CPU type server is given by

$$L_{SG} = L_{ma} \times 0.5 + L_{ca} \times 0.4 + L_{na} \times 0.1. \quad (7)$$

The load for a NETWORK type server is defined as follows:

$$L_{SG} = L_{ma} \times 0.5 + L_{ca} \times 0.1 + L_{na} \times 0.4. \quad (8)$$

The load for a CPU_NET type server is calculated by

$$L_{SG} = L_{ma} \times 0.5 + L_{ca} \times 0.25 + L_{na} \times 0.25. \quad (9)$$

Based on a previous report [13], we have adopted these weightings here. However, in normal cases, weightings should be determined based on the server system operating environment during actual operations.

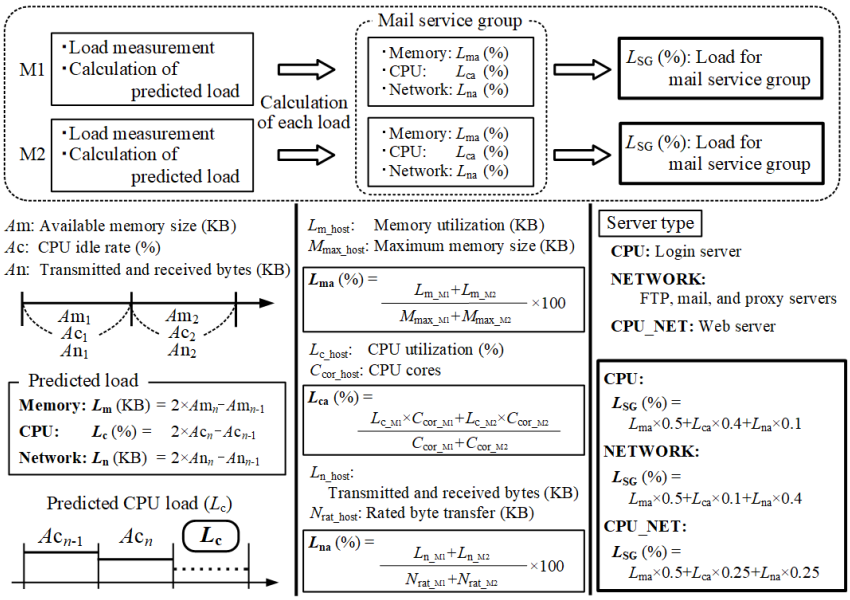


Fig. 5: Load calculation flow for each server.

The proposed system adopts the number of accesses from clients to the server as the condition for shifting from the PS1 to PS2 condition or the reverse. Figure 6 shows the calculation method for the number of accesses from clients. Here, the PS2 condition is designed to consider a server system that is frequently accessed by clients during the day but never accessed by clients at night. The number of accesses is calculated for each virtual server using the "tcpdump" UNIX command.

Figure 6(a) shows the method for calculating the number of client accesses. With a load balancing program and a monitoring program running in the load balancer, accesses to M2 are performed by the client and the monitoring program. The monitoring program performs k accesses per second to M2. The access numbers from clients for y seconds can be calculated by subtracting the number of accesses by the monitoring program ($k \times y$) from the total number of accesses z.

Figure 6(b) shows the method used for calculating the number of accesses that considers server system configuration changes. Here, one virtual IP address is set for one server under the PS1 condition. In the PS2 condition, multiple virtual IP addresses are set for one server, which means that the number of accesses from the load balancer monitoring program and the client will change depending on the load condition because the number of virtual IP address settings is different. Therefore, the y second measurement time is divided into n segments, and the number of accesses z is calculated from the number of accesses to all virtual IP addresses. The number of accesses in the PS1 condition for y seconds is given by

$$z = \frac{z_1}{1} + \frac{z_2}{1} + \cdots + \frac{z_{n-1}}{1} + \frac{z_n}{1}. \quad (10)$$

The number of accesses if the configuration is changed from the PS1 to PS2 condition in y seconds is given by

$$z = \frac{z_1}{1} + \frac{z_2}{1} + \frac{z_3}{1} + \frac{z_5}{2} \cdots + \frac{z_n}{2} + \frac{z_{n+1}}{2}. \quad (11)$$

In the figure, because the configuration change occurs when the number of accesses z_4 is calculated, the number of virtual IP addresses set for the server is different between the z_3 measurement and the z_5 measurement. Therefore, the number of accesses z is calculated by deleting the measured value of z_4 and adding z_{n+1}. Here, the number of accesses in each virtual server is calculated simultaneously by Synchronization System 2.

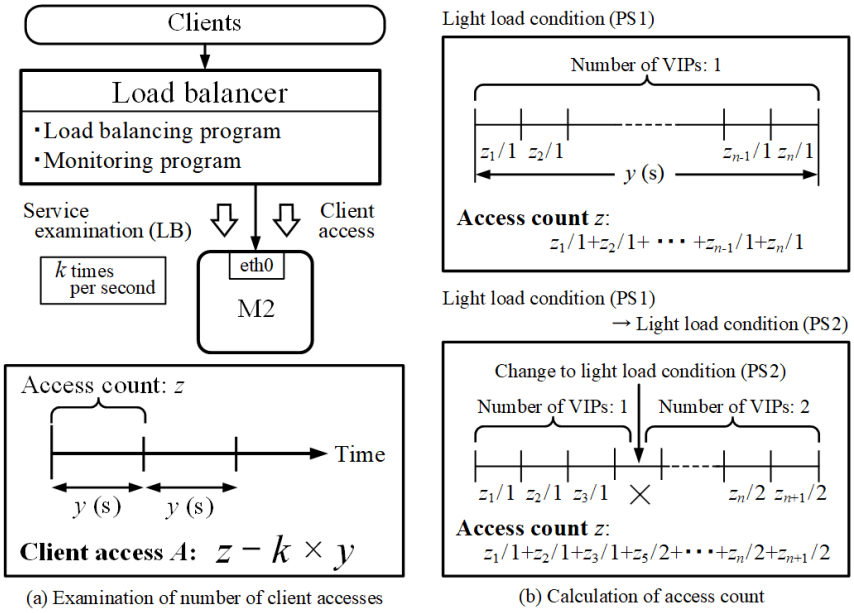


Fig. 6: Measuring number of accesses from clients.

3.2. Detailed processing prior to changing server system configuration

Figure 7 shows the classification of the load class, the access class, and the configuration change determined based on these classes. In this system, the server load is used to determine the configuration change in the moderate load condition and PS1 condition, and client accesses are used to determine the configuration change in the PS1 and PS2 conditions.

Figure 7(a) shows how to classify load classes based on the server load and how to determine configuration changes. The load values calculated for each service group are classified into two types: moderate and light load classes. As shown in the figure, two thresholds, which are for moderate and light loads, are set for classification. Loads that exceed the moderate load threshold are represented by area M, and loads that fall between the moderate and light load thresholds are represented by area ML. Finally, loads that are lower than the light load threshold are represented by area L.

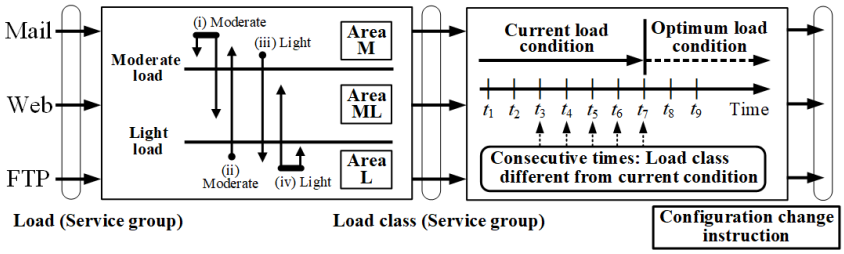
Area ML is adopted to reduce the influence of fluctuations near the threshold. The vectors indicating loads (i) through (iv) show the changes in the loads. The black dots indicate the initial state, and the arrowhead indicates the final state. If the load state shifts to area ML, the load is classified as belonging to the initial state. If the load shifts from area L to area M, it is determined as belonging to the moderate load class, and if it shifts from area M to area L, it is determined as belonging to the light load class. All virtual servers belonging to each service group operate from the load measurement process to the load class determination process.

The virtual server, which is created on Base Server 1 and recorded at the beginning of the operation file, determines the configuration change based on the load classes. It is desirable to decide the configuration change while considering the server system usage environment. In order to prevent excessive configuration changes due to load class fluctuations, the maximum number of consecutive condition detection operations is set for each load condition. As shown in the figure, the server system configuration is changed from the current load condition to the optimum load condition after a load class that is different from the current condition has been detected five times in a row.

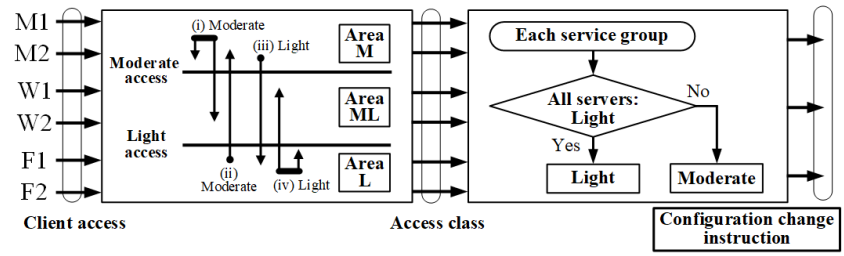
Figure 7(b) shows how to classify access classes based on client access levels and how to determine configuration changes. The example shown here considers a system in which clients frequently access the server during the day but never access the

server at night. The number of accesses calculated for each service group is classified into either the moderate or light access classes. As shown in the figure, two thresholds, moderate and light access, are set for classification. Loads that exceed the moderate access threshold are represented by area M. Loads that fall between the moderate and light access thresholds are represented by area ML. Finally, loads that are lower than the light access threshold are represented by area L.

The vectors show the changes in the number of accesses. The access class is determined by the same method shown in Fig. 7(a). The virtual server, which is created on Base Server 1 and recorded at the beginning of the operation file, determines the configuration change based on the access class. The configuration change is performed in cases where the service group access class is determined to be light.



(a) Flow until configuration change instruction based on load of service group



(b) Flow until configuration change instruction based on client access

Fig. 7: Determination process for configuration change in server system.

The operational flow for determining the configuration change is shown in Fig. 8. Here, the number of consecutive times for changing the server configuration is set to one. For the sake of efficiency, the controls for determining and performing configuration changes are executed on one server. The virtual server, which is created on Base Server 1 and recorded at the beginning of the operation file, determines the configuration changes in the mail, web, and FTP service groups. After the optimum configuration state for each service group has been determined, the server system configuration is optimized. Using the mail service group as an example, the operation used to determine the configuration change is shown below.

The proposed system examines the mail service group load class and then examines the current server configuration. If the load class is determined to be moderate, and the current server configuration is the moderate or PS1 load condition, the moderate load condition is determined to be the optimum server configuration. If the current server configuration is in the PS2 condition, the optimum server configuration is determined to be the PS1 condition. In our current system design, the PS2 condition cannot be shifted directly to the moderate load condition. If the load class is light and the moderate load condition is set for the current server configuration, the optimum server configuration is determined to be PS1. If the current server configuration is either PS1 or PS2, the access class is examined. If the access class is moderate, the optimal server configuration is determined to be PS1, and if it is light, the optimal server configuration is determined to be PS2. These operations are performed for each service group.

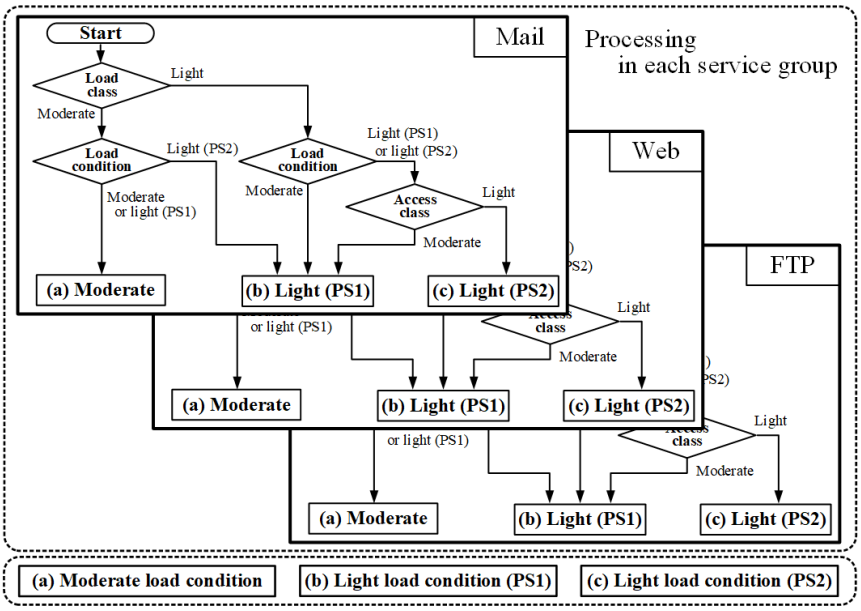


Fig. 8: Configuration change flow path for each service group in server system.

3.3. Details of process timing in PSS

Figure 9 shows the process timing, which is performed by the PSS for regular processing and configuration changes. In this system, load measurements for each virtual server start simultaneously when instructed by Synchronization System 1. During the load measurements, the memory size, the CPU idle rate, and the network transfer rate are used, as shown in Fig. 5. Moreover, heartbeat processing is performed immediately before the end of the load measurement process to detect virtual server failures. During heartbeat processing, the service offer state and the network are examined. If a failure is detected, a backup server is created to recover the server function.

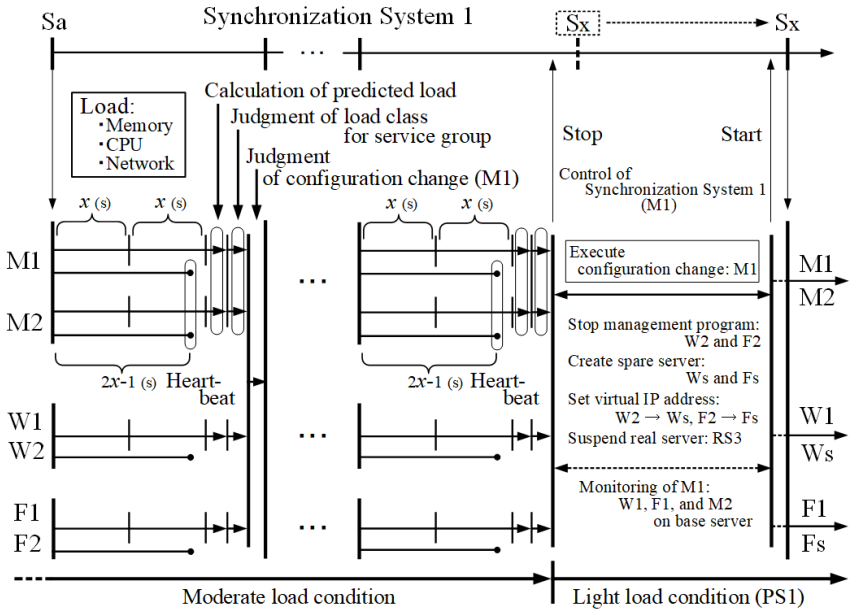


Fig. 9: Details of processing timing in PSS.

All of the virtual servers measure these loads twice and then use the difference in the load values to calculate the predicted load. The load class for each service group is determined based on the predicted load. M1, which is created on Base Server 1 and recorded at the beginning of the operation file, determines the server system configuration change based on the load class for each service group. If it determines that no configuration change is necessary, the load measurement process starts at the instruction timing for Synchronization System 1. If it determines that a configuration change is necessary, M1 creates a configuration change instruction file and executes the configuration change process based on that file.

In Fig. 9, Synchronization System 1 is set to the stop state by M1 until the configuration change finishes. In the figure, it can be seen that the configuration change for all service groups is performed from the moderate to PS1 conditions. M1 creates the spare servers (Ws and Fs), sets the virtual IP address of W2 and F2 to Ws and Fs, respectively, and then stops the PSS management program for W2 and F2, which are set to the access stop state. Thereafter, RS3 is suspended to save power. In this processing series, W1, F1, and M2, which are created on Base Servers 1 and 2, monitor the M1 processing status. After the configuration change, the server system operates in the state shown in Fig. 2(c).

Figure 10 shows how to deal with a server failure during the load measurement or configuration change processes. During load measurement, each virtual server performs heartbeat processing, as shown in Fig. 9. If a failure is detected, information regarding the failed server is moved from the operation file to the failure file. To recover the failed server functions, a backup server is created, and the backup server information recorded in the backup server file is added to the operation file. Once the server has recovered, the information is returned from the failure file to the operation file, the backup server is suspended, and the backup server information is deleted from the operation file.

In Fig. 10(a), because M2 failed at time T1 during the load measurement, the M2 information in the operation file is moved to the failure file, backup server Mb is created, and the Mb information is added to the operation file. Figure 10(b) shows how to deal with a failure in the server that controls the configuration change. In this system, in order to simplify the control process, one server processes the configuration change. The instruction file for the configuration change is divided at each stage, as shown in the figure, and the configuration change finishes by executing each instruction. This means that another server can assume the process even if the server performing the processing fails.

In the figure, the configuration change for all service groups is performed from the moderate load condition to the PS1 condition. As shown in Fig. 9, M1 performs the configuration change process. The instruction file for the configuration change comprises four stages, numbered (i) through (iv). In this scenario, M1 encounters a problem after processing of (i) finishes. W1, which is monitoring M1, detects the problem, and then performs recovery processing. After the Mb creation instruction, the remaining stages (ii) through (iv) are performed. Note that F1 and M2 monitor W1 as it is executing the configuration change.

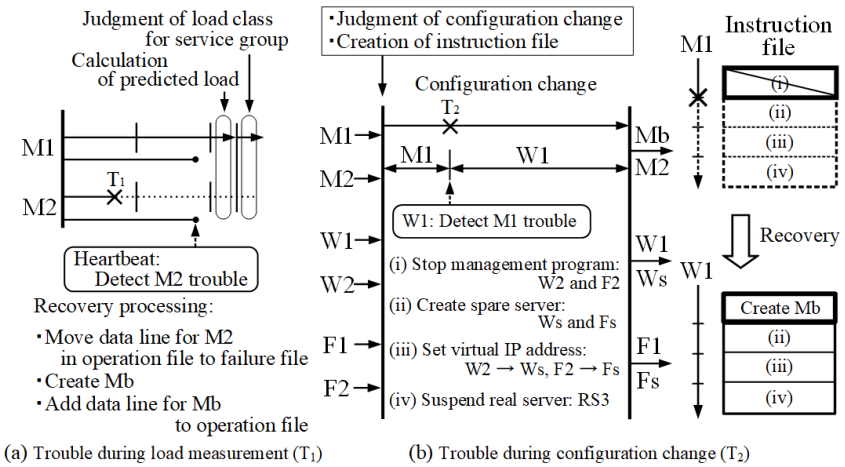


Fig. 10: Dealing with server failure.

4. System operation experiment

4.1. Experimental system

Figure 11 shows the server configuration of our proposed PSS experimental system, which does not require a dedicated management server. It consists of a file server, three real servers (RS1, RS2, and RS3) used for creating virtual servers, a load balancer, a client, and two 1000BASE-T switching hubs. The virtual server group provides services to clients and the real server group creates the virtual server group. The virtual servers offer mail, web, and FTP services to the clients. Two virtual mail servers (M1 and M2) belong to the mail service group, two virtual web servers (W1 and W2) belong to the web service group, and two virtual FTP servers (F1 and F2) belong to the FTP service group. Each service group is created by redundantly configuring virtual servers.

The two types of system data used for creating the virtual servers are stored in each real server. The three types of data (Ms, Ws, and Fs) used for the spare server and the three types of data (Mb, Wb, and Fb) used for the backup server are stored in the file server. The executor system shown in Fig. 3 is installed on each virtual server. The management program, management files, spare server system data, and backup server system data, which are saved in the file server, can be used via the NFS connection with the file server.

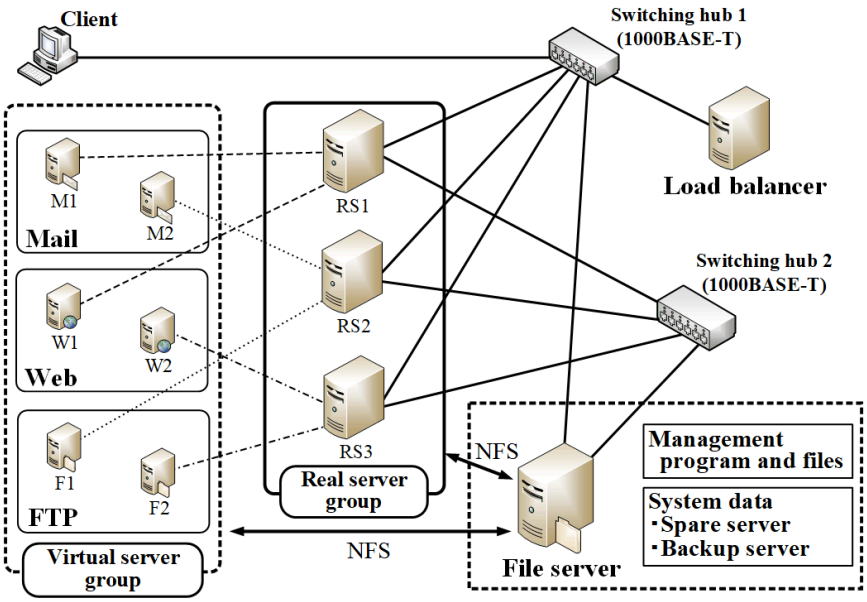


Fig. 11: Experimental system configuration.

The specifications of the file server, the real server, virtual server, spare server, backup server, and load balancer used in the experimental system are listed in Table 2. In the experimental system, a postfix server program that handles the simple mail transfer protocol (SMTP) is installed on the virtual mail server, a HTTP Daemon (httpd) program is installed on the virtual web server to handle HTTP, and a very secure FTP daemon (vsftpd) server program is installed on the virtual FTP server to handle FTP. The 64-bit version of the Community Enterprise Operating System (CentOS), which is often adopted as the server operating system, is installed on the file server, the real servers, virtual servers, spare servers, backup servers, and the load balancer. Virtualization is accomplished with Kernel-based Virtual Machine (KVM) software running on the real servers. The load balancer uses a software processing method, and UltraMonkey-L7 is installed. Each real server operates with a character-based user interface in order to utilize memory effectively.

Table 2: Experimental system specifications.

Specifications	File server	Real server (RS1, RS2, RS3)
CPU	Intel Core i7-4790 3.60 GHz (TB: 4.00 GHz)	
Memory	8,192 MB	
Storage	SSD (SATA 3)	
System software	nfsd KVM 1.5.3	KVM 1.5.3
OS	CentOS 7.7 (64-bit)	

Specifications	Virtual server/ Spare server/ Backup server
CPU	1
Memory	2,048 MB
System software	postfix, httpd, vsftpd
OS	CentOS 7.7 (64-bit)

Specifications	Load balancer
CPU	Intel Core 2 Quad Q9400 2.66 GHz
Memory	4,096 MB
System software	UltraMonkey-L7
OS	CentOS 6.10 (64-bit)

4.2. Log file indicating operation status

This system records the operating status and time in the system log file. Figure 12 shows the system log in a case involving a configuration change. The comments are used to present important content. In this figure, the M1 system log that processes the configuration change and the W1 system log that monitors M1 are shown as examples.

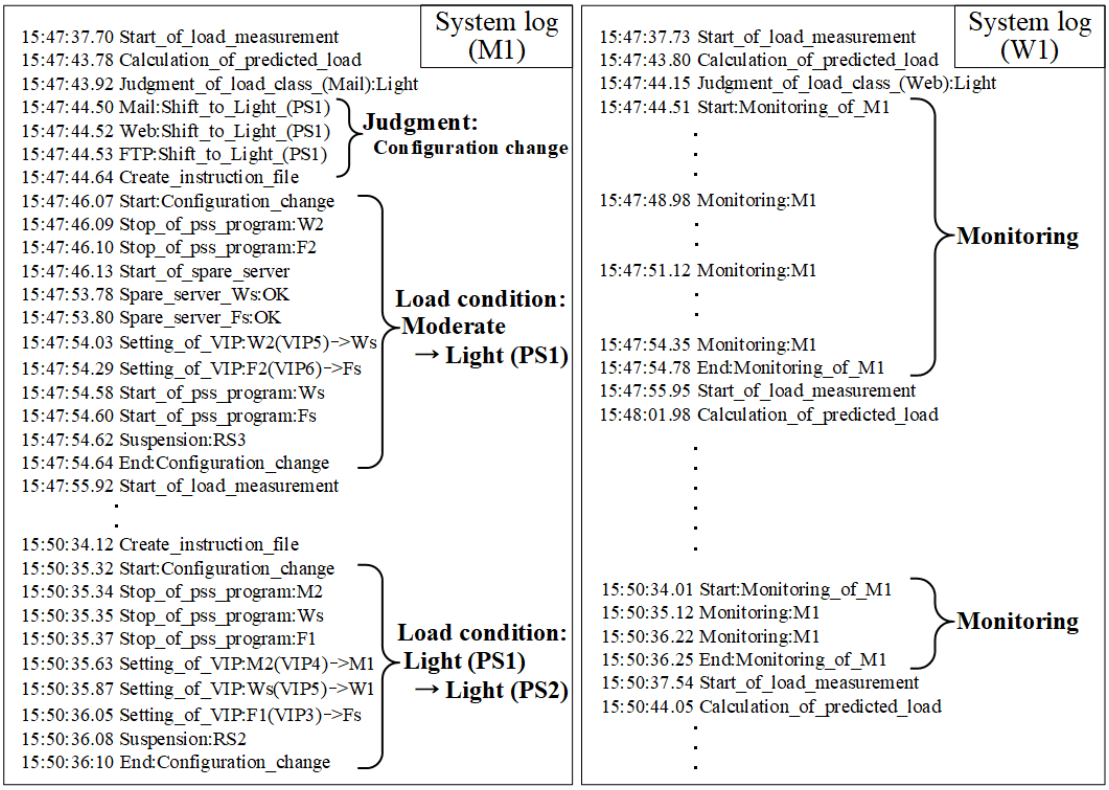
In Fig. 12(a), the server configuration shifts from the moderate load condition to the PS1 condition, and then shifts from the PS1 to PS2 condition. In the M1 system log, after starting the load measurement at 15:47:37.70, the mail service group is determined to be in the light load class. Because the web and FTP service groups are also the same state, M1 creates an instruction file for the configuration change at 15:47:44.64, and then starts the configuration change at 15:47:46.07.

After stopping the PSS management program for W2 and F2 at 15:47:46, creation instructions for spare servers Ws and Fs are performed at 15:47:46.13, and creation finishes at 15:47:53. Thereafter, the VIP5 and VIP6 virtual IP addresses are set to Ws and Fs at 15:47:54, and the suspension instruction for RS3 is performed at 15:47:54.62. At 15:47:55.92, the operation state under the PS1 condition starts. In the W1 system log, M1 monitoring starts at 15:47:44.51 and ends at 15:47:54.78. In the M1 system log, the configuration change from the PS1 to PS2 condition starts at 15:50:35.32. After stopping the PSS management programs for M2, Ws, and F1 at 15:50:35, the VIP4, VIP5, and VIP3 virtual IP addresses are set to M1, W1, and Fs, respectively. Thereafter, the suspension instruction for RS2 is performed, and the configuration change finishes. In the W1 system log, M1 monitoring starts at 15:50:34.01 and ends at 15:50:36.25.

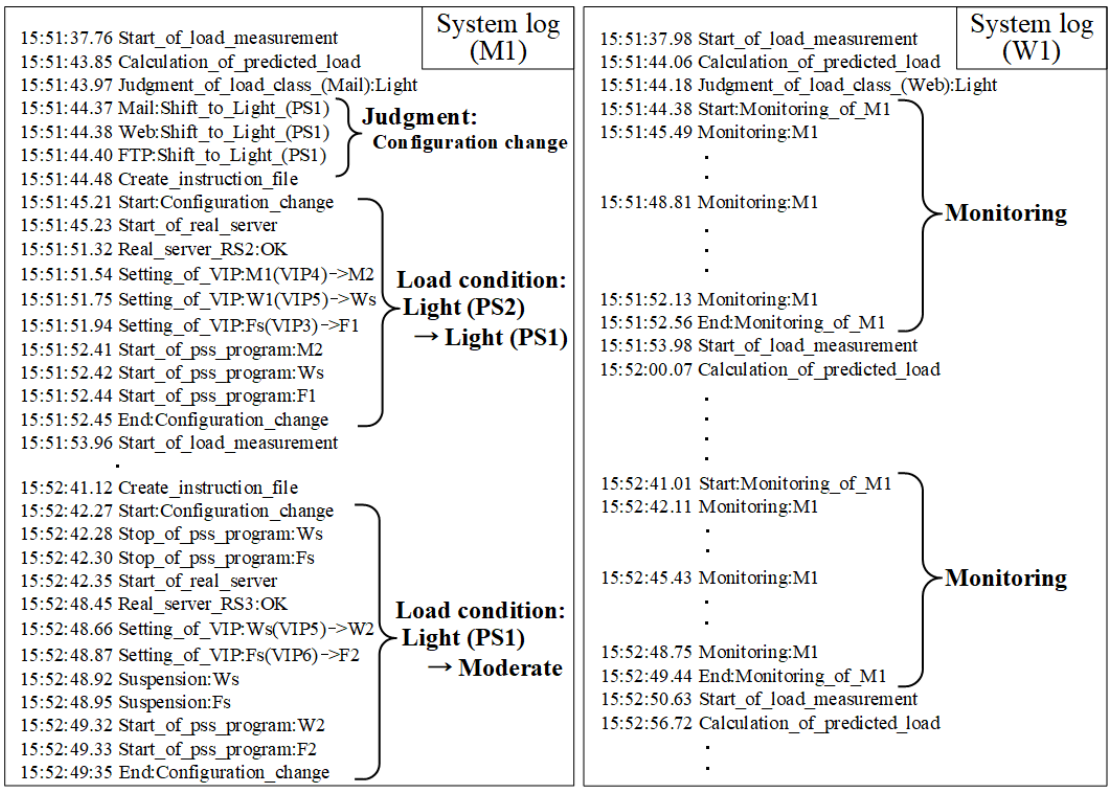
In Fig. 12(b), the server configuration shifts from the PS2 to PS1 condition, and then shifts from the PS1 condition to the moderate load condition. In the M1 system log, the mail service group is determined to be in the light load class at 15:51:43.97. Because the web and FTP service groups are also in the same state, M1 creates an instruction file for the configuration change at 15:51:44.48, and starts the configuration change at 15:51:45.21.

The start instruction for RS2 is performed at 15:51:45.23, and RS2 startup finishes at 15:51:51.32. After the virtual IP addresses VIP4, VIP5, and VIP3 are set to M2, Ws, and F1, respectively, the PSS management programs are started on those servers at 15:51:52. At 15:51:53.96, the operation state under the PS1 condition starts. In the W1 system log M1, monitoring starts at 15:51:44.38 and ends at 15:51:52.56. In the M1 system log, the configuration change from the PS1 condition to the moderate load condition starts at 15:52:42.27. After the Ws and Fs PSS management programs stop at 15:52:42, the start instruction RS3 is performed at 15:52:42.35, and startup finishes at 15:52:48.45.

After virtual IP addresses VIP5 and VIP6 are set to W2 and F2, respectively, at 15:52:48, the PSS management program starts on those servers, and the configuration change finishes. In the W1 system log, M1 monitoring starts at 15:52:41.01 and ends at 15:52:49.44.



(a) Log of configuration change from the moderate to light load condition



(b) Log of configuration change from the light to moderate load condition

Fig. 12: Log file indicating operation status.

4.3. Operational characteristics of experimental system

The operational characteristics of our experimental system are listed in Table 3. The times for the real, virtual, spare, and backup servers are the average times given by the UNIX "time" command. "Virtual server (Local)" represents the characteristics of the virtual server created by the real server. Note that the startup time for the first virtual server is 11.99 s, and the startup time for the second is 11.56 s. This time lag is assumed to be caused by disk cache processing in the real computer system because the virtual server is software. "Spare and backup servers (NFS)" presents the characteristics of the spare and backup servers created on the real server by using the spare and backup server system data stored on the file server via the NFS connection. The normal startup time is 18.61 s, whereas the startup time from the suspended state is 4.07 s. Therefore, in this system, the spare server and the backup server are kept in the suspended state.

In the PSS, the power consumption is 89.02 W because all three real servers are operating in the case of the moderate load condition. In the case of the PS1 condition, the power consumption is 63.84 W (72% of the moderate load condition) because only two real servers are operating (RS3 is in suspended state). In the case of the PS2 condition, the power consumption is 35.65 W (40% of the moderate load condition) because only one real server is operating (RS2 and RS3 are suspended).

The time required to change the system configuration from the moderate load condition to the PS1 condition is 9.71 s, and the spare server activation time has the most effect on this value. The time required to change the system configuration from the PS1 condition to the moderate load condition is 8.07 s, and the time required to activate a suspended real server has the most effect on this value. Furthermore, because virtual servers offering services to clients are configured redundantly (except under the PS2 condition), there is no effect on clients if the server fails. In the PS2 condition, because the number of accesses from clients is almost zero and the startup time for the backup server is 4.07 s, there is very little effect on clients even if a server failure occurs in this state.

Table 3: Experimental system operational characteristics.

Performance	Real server	Virtual server (Local)		Spare and backup servers (NFS)	
		First	Second	First	Second
Start time (s)	30.13	11.99	11.56	18.61	18.02
Restart time (s)	32.28	13.43		13.50	
Time to activate suspended server (s)	3.94			4.07	

PSS		
Load condition (All service groups)	**Active real servers (Number of active virtual servers)**	**Power consumption (W) (Total of 3 real servers)**
Moderate	RS1, 2, 3 (6)	89.02
Light (PS1)	RS1, 2 (6)	63.84
Light (PS2)	RS1 (3)	35.65
Load condition	**Operation for configuration change**	**Required time (s)**
Moderate → Light (PS1) (All service groups)	Create two spare servers → Set two virtual IP addresses → Suspend one active real server	9.71
Light (PS1) → Moderate (All service groups)	Activate one suspended real server → Set two virtual IP addresses → Suspend two spare servers	8.07

5. Conclusions

Herein, we reported on the development of a power-saving server system (PSS) that does not require a dedicated management server. A specific PSS problem encountered by the conventional method was shown, and the specifications and characteristics of the proposed method that can solve the problem while enabling the same power-saving level were demonstrated. In addition, the management program dispersion problem that could result from the use of the proposed method is explained, and the solutions in the form of the executor and synchronization systems were explained.

In our proposed system, the following method was adopted. Each server providing services to clients performs load measurements and calculates the predicted load. Thereafter, one server, which is selected based on the operation file, creates an instruction file for changing to the optimal server configuration and executes the necessary configuration changes based on those instructions.

The power-saving states involve two virtual server configurations (redundant and non-redundant) that are based on the actual operating state of the server system. In addition, an experimental system using the proposed method was constructed, and its architecture, specifications, and details of its operation were described. In the power-saving condition, the proposed system operates at approximately 72% of the power consumption required during normal conditions when virtual servers are configured redundantly, and at approximately 40% of the power consumption required during normal conditions when a non-redundant configuration is used.

Server system operators constantly face issues such as reducing operating costs, increasing operating loads to meet increasing numbers of target servers, and dealing with server failures. In particular, dealing with problems that arise with the management server, which is the key component of the system, is critical for handling such problems. By adopting our proposed system, operation costs and operating loads can be reduced, power savings can be accomplished based on the server system operating environment, and the server system management functions including failure countermeasures for the management and target servers can be automated. This will allow the burdens and time constraints imposed on the server system operator to be reduced significantly.

References

[1] Ministry of Health, Labor and Welfare https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/newpage_00032.html, 2020.6.

[2] T. Hussain, A. Haider, M. A. Rafique, R. Ali, M. U. Rehman, and K. R. Ullah, “A High Performance Edge Server System,” IEEE Conf. Proc., Vol. 2019, No. iCoMET, 2019, pp. 1-4.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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,” IEEE Trans. on Parallel and Distributed Systems, Vol. 31, No. 9, 2020, pp. 2155-2169.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] W. Shengquan, L. Jun, C. Jian-Jia, and 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, 2011, pp. 289-298.

[16] Ministry of Economy, Trade and Industry, http://www.meti.go.jp/committee/materials/downloadfiles/g80520c03j.pdf, 2008.5.

[17] M. Hirono, T. Sato, J. Matsumoto, S. Okamoto, and N. Yamanaka, "HOLST: Architecture Design of Energy-efficient Data Center Network based on Ultra High-speed Optical Switch," IEEE Conference Proc., Vol. 2017, No. LANMAN, 2017, pp. 1-6.

[18] K. K. Thangadorai, R. K. Saranappa, A. S. Ahmed, K. Murugesan, S. M. Singh, R. Mundra, M. N. Sataraddi, S. Sriram, V. Singh, B S. Kumar, M. M. Patil, and D. Das, "Intelligent and Adaptive Machine Learning-based Algorithm for Power Saving in Mobile Hotspot," IEEE Conf. Proc., Vol. 2020, No. CCNC, 2020, pp. 1-6.

[19] J. K. Perin, A. Shastri, and J. M. Kahn, "Design of Low-Power DSP-Free Coherent Receivers for Data Center Links," Journal of Lightwave Tec., Vol. 35, No. 21, 2017, pp. 4650-4662.

[20] Y. Lee, K. Miyanabe, H. Nishiyama, N. Kato, and T. Yamada, "Threshold-Based RRH Switching Scheme Considering Baseband Unit Aggregation for Power Saving in a Cloud Radio Access Network," IEEE Systems Journal, Vol. 13, No. 3, 2019, pp. 2676-2687.

[21] Y. Fukushima, T. Murase, and T. Yokohira, "Energy-Aware Optimal Server Location Decision in Server Migration Service," IEICE Technical Report, Vol. 116, No. 428, 2017, pp. 53-58.

[22] A. Husen, I. Raza, and H. S. Asad, "Erlang Based Server Selection Scheme Using Software Defined Networking in Datacenter for Energy Conservation," IEEE Conf. Proc., Vol. 2019, No. FIT, 2019, pp. 36-41.

[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, pp. 34-54.

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.

Toshikazu TAKESHITA

He received his B.E. degree in 2020 from Tokyo Polytechnic University, where he is currently working toward the M.S. degree. He is in charge of implementation of experimental hardware systems and analyses of power-saving data.

Takumi YAMAGUCHI

He received his B.E. degree in 2020 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.