A Study of Building a Distributed System using Idle Resources

January 2022

Toshiya Kawato

Abstract

Computers have computing resources such as CPU and storage, and are used for a variety of purposes such as office work. There are computers that operate normally but are discarded due to equipment upgrades, etc. (hereafter referred to as "used-computers"). Their computers have their resources such as CPU and storage and so on, that is idle computing resources (hereinafter referred to as “idle resources”). However, we cannot make use of their computers anymore even though they have their idle resource. In order to make effective use of these idle resources of used-computers, we propose a distributed storage system using their resources as storage. We design and implement a system that automatically builds used-computers as distributed storage in order to use them with low labor. In addition, we operate the distributed storage system in a real environment as a backup storage for online storage and mail systems. However, if more used-computers are used to ensure capacity, power consumption will increase and the cost performance will become worse. In addition, the problem of securing a place for installation also arises. Therefore, we turn our attention to computers, which are used for office work and other purposes (hereinafter referred to as “ using-computers ”).

Even though using-computers are used for their purposes to operate stable and continuous works, they have surplus capacity over their intended use since recent computers have enough power for office works. These surplus resources are potentially idle resources, and computers in use also have idle resources. Therefore, we propose to combine the idle resources of used-computers and using-computers to form a distributed storage system. A using-computer has its original use such as office work. Therefore, when idle resources of the using-computer are used, they cannot be dedicated to distributed storage, which is a problem that cannot be conveniently controlled. They may not always be running. They may start and stop depending on the situation of their original use. In addition, their timing and frequency are not fixed. It is necessary to deal with the instability of this state. There are also problems such as the fluctuating amount of idle resources and ensuring security. Of these issues, we focus on the instability of state using-computers. Using-computers are not always available as a system because joining and leaving cannot be controlled. Therefore, the data to be stored into using-computers for load balancing and redundancy should be temporary. Thus, we suppose the original data exists in external of using-computers. For example, the original data can be placed on a used-computer. This will allow the system to continue in a situation where all using-computers leave the system. Here, a using-computer leaving can be divided into two types: one is that have enough time to process leaving operation such as a planned shutdown; and another one is that do not have an afford to process it, such as a normal network shutdown. If there is enough time, leaving node can declare its leaving to other nodes and move the files in its possession. However, if there is not enough time, other nodes need to detect leaving and process files held by leaving nodes. Therefore, we propose a process based on the declarations at the leaving node and the availability of moving files in its possession.

As a practical application of using idle resources, we propose an e-Learning system. This system is designed to distribute the load by storing and distributing content to the computers that use it. When the load on the system increases due to an increase in the number of client accesses, the system makes use of idle resources of using-computer for the load balancing. When the load decreases, using-computers are removed from the system. In this way, high scalability can be achieved inexpensively. We also propose a content delivery network (CDN) using using-computers as another use case. By building a cache server, which is a component of a CDN, on the internal network and delivering content from there, the amount of external data communication can be reduced. This will lead to a reduction in data communication charges on the shared network. Moreover, this can be achieved inexpensively by building the cache server using used and in-use computers.

Acknowledgments

I would like to thank my advisers, Prof. Takao Kawamura, Prof. Kazunori Sugahara, Dr. Kenichi Takahashi and Dr. Masayuki Higashino. Although I have many shortcomings, they kindly and politely taught me the basics of being a researcher with great persistence.

I would also like to thank Wataru Shintani and the students in both the Social Information Systems Laboratory and the Computer Science Laboratory. They have helped me in many ways.

I would like to express my gratitude to the people of Tottori University (my former affiliation) and National Institute of Technology, Yonago College (my current affiliation) for their understanding of my activities as a working graduate student.

Last but not least, I would like to thank my wife and daughter. Without their support, I would never have been able to complete my doctoral thesis and finish my graduate studies. Thank you for your continued support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computers have computing resources such as CPU and storage, and are used for a variety of purposes such as office work.

Even though we can use their computers which work normally until they break down, their computers may be abandoned or discarded due to inability to update an operating system due to insufficient performance or equipment renewal. In this thesis, such computer is referred to as a *used-computer*. Their computers have their computational resources such as CPU and storage and so on that are available but not being used. In this thesis, such resource is referred to as a *idle resource*.

On the other hand, the amount of data handled by an information system continues to increase, and storage for saving data is indispensable. If a storage capacity becomes insufficient, we can increase its capacity by adding new disks. It is, however, difficult to purchase a new disk when a budget is limited.

Therefore, we propose to utilize the idle resources of used-computers as a distributed storage system. In this case, it is necessary to discuss a storage system that considers characteristics of used-computers. In addition, in order to reduce works required for an installation and operation, it is necessary to automate processing necessary for construction. We organize the characteristics of used-computers, and consider a storage system based on the characteristics. We implement a system that automatically establishes an environment as a node of the distributed storage system on each used-computer.

Also, as a specific application, it is actually used for online storage and email archiving. However, when reusing used-computers, even if the installation cost is low, the operation cost may be high due to the increased power consumption. Also, it is necessary to secure a place for installation.

Therefore, we focus on computers that are used continuously, for example, for office work. In this thesis, such computer is referred to as a *using-computer*. Computers used for a certain purpose, *using-computers*, generally do not always use their own computational resources up to the upper limit, but have extra resources to operate them stably. Thus, using-computers have idle resources. In other words, their computers affords to compute something except a certain purpose. Therefore, we propose to build a distributed storage system by using idle resources of using-computers in addition to used-computers.

Since using-computers are existing and in use, power consumption can be reduced and there is no need to secure a new location for installation. However, using-computers have an original use. As a result, start-up and shutdown are uncontrollable and may not always be available. If the system is to be used as a distributed storage system, it is necessary to assume that participation and withdrawal from the system will occur frequently. There is also the issue of the amount of idle resources fluctuating according to the burden of the original use, and the issue of ensuring security. We first propose an algorithm for joining and leaving the system.

And we also propose an e-Learning system and a content delivery network

(*CDN*) as examples of the applied use of idle resources of each computer. In e-Learning system, high performance is required during peak hours when large amounts of content are used all at once. On the other hand, during normal times when the content is used in a distributed manner, not much performance is required. Here, if the system is installed to cope with peak periods, the cost will be high, and if it is installed to cope with normal times, it will not be able to cope with peak periods. Therefore, it is necessary to be able to flexibly adjust performance while keeping costs low. Therefore, we propose an e-learning system that utilizes using-computers and used-computers. By distributing the load by placing and delivering content to computers in use according to the system load, the processing performance of the system can be improved. CDN is a mechanism for efficiently distributing content. CDN is usually built an external network, but by building them on an internal network, the amount of network communication on external networks can be reduced. Here, by caching content that external networks have on using-computers in an internal network, the amount of communication in external networks can be reduced at a low cost.

This thesis is organized as follows. In chapter 2, we proposes the use of used-computers as a distributed storage. In chapter 3, we propose to use using-computers as distributed storage in addition to used-computers. In chapter 4, we propose an e-Learning system using idle resources of using-computers and used-computers. In chapter 5, we propose an content delivery network using idle resources of using-computers and used-computers. In chapter 6, we present the conclusion of this thesis.

Chapter 2

Distributed Storage using Idle Resources of Used-Computers

2.1 Introduction

The amount of data handled by an information system continues to increase, and storage for saving data is indispensable [1]. In order to cope with the increasing amount of data, capacity of HDD and SSD is growing, and online storage that can be used via a network has been widespread [2]. Moreover, organizations such as companies and universities use not only built-in storage such as personal computers used by individuals but also shared storage and storage of servers such as business systems.

Now, if the amount of data to be stored increases and a storage capacity becomes insufficient, we can increase its capacity by adding new disks. It is, however, difficult to purchase a new disk when a budget is limited. On the other hand, in modern times when information equipment floods, there are many unused idle resources despite those use value, and resources are not effectively utilized. For example, there are used-computers, which are left unused or being discarded due to renewals or other reasons.

In order to solve those problems, used-computers which are idle resources can be reused as storage. Since used-computers are existing devices, there is no initial cost in reusing themselves. We can, therefore, inexpensively introduce used-computers to reuse compared with introducing new storage. Moreover, in an environment such as a university where the total number of computers is large and computers are frequently replaced, there are many used-computers. These many unused-computers can be collected to be reused. It would be effective utilization of resources more and more to reuse used-computers for different purpose while they can still be available for generic computer usage. In order to use used-computers as storage, it is necessary to consider characteristics of used-computers. In addition, in order to reduce works of introduction and operation and make them easier, it is necessary to automate processing necessary for constructing a storage system with used-computers.

In this chapter, we organize the characteristics of used-computers and design a storage system based on its characteristics. Moreover, we focus on distributed storage, and then implement an auto-construction system that automatically constructs a distributed storage environment in used-computers which makes the used-computer join the distributed storage as a part of the storage. We will also confirm that used computers can be used as distributed storage by operating distributed storage in a real environment using used-computers.

2.2 Design

Before reusing used-computers as storage, it is necessary to discuss practical methods that have advantages and can be used for appropriate use. In this section, we organize the characteristics of used-computers. Moreover, we discuss a storage system considering its characteristics.

Small Capacity

Recent PCs are generally equipped with TB-class disks. However, capacity of recoverable used-computers may be a few hundred GB. For the reason, in order to have large capacity such as TB class, it is necessary to secure numerous used-computers. It is, however, inefficient and not realistic in terms of installation locations and power consumption in this state.

For this reason, if an existing disk of a used-computer cannot satisfy a required capacity, we replace the existing disk with a new large capacity disk in order to secure a capacity. Although costs are incurred on a disk to replace, disks for computers are less expensive than ones for storage products and servers, and storage can be constructed at low cost as a whole. Moreover, it is possible to reduce a risk of an occurrence of a failure by replacing a disk having a high failure rate with a brand-new one.

Low Performance

Compared with storage products and servers, computers are inferior in performance of CPU and NIC. Since their disks are also supposed to be used in general computers, it is difficult to request high-speed communication and high-load processing. Their disks are, thus, not suitable for a system in which reading and writing occur frequently. In addition, used-computers demonstrate even lower performance than current computers.

For this reason, a storage system needs to be a light-weight system that can demonstrate sufficient performance even for computers use. An appropriate utilization is supposed to be a backup system for storing data with low frequency of utilization or update.

Different Lifetime

Depending on how hard a computer is used until the computer becomes unused, performance may deteriorate compared to fresh condition, and it is difficult to predict a lifetime such as how long it can be used after collection and so on. Even if a computer can be used inexpensively, availability should not be reduced by its failure rate.

For this reason, it is necessary to sufficiently deal with failure prepared by monitoring a state of used-computers, e.g., estimation of a lifetime, automatic detachment on failure and automatic incorporation of a spare machine.

High Diversity

Since various manufacturers sell various models, computers are more diverse than storage products and servers. It would then take a lot of works to configure a used-computer as storage. The same model and performance are desirable when computers are used in a cluster configuration. It is, however, difficult when used-computers are reused because their performance and configurations vary among different models.

For this reason, for reducing construction works, a method that can automatically construct an environment for use as storage is necessary.

Available Numerous Units

In an environment such as a university where the total number of computers is large and replace is frequent, we can collect numerous used-computers. Moreover, since there is no initial cost for introduction to reuse used-computers themselves, it is possible to use numerous used-computers as long as there is a place for installation of used-computers. Disks are distributed and arranged in case of using numerous disks, and high availability can be realized at low cost if data can be arranged distributedly or redundantly. In addition, there are few restrictions on a place to install used-computers compared with rack-mounted servers. It is, therefore, possible to physically distribute used-computers and place them easily. Moreover, it is possible to flexibly install and use used-computers regardless of locations if there are power supply and network connectivity.

For this reason, we focus on distributed storage that is a configuration in which distributed disks on a network connecting to form one large storage. Data is distributed and stored in distributed storage because disks are distributed. Now, in addition to a distributed arrangement, availability in the case of a redundant arrangement of data will depend on the number of disks. used-computers are, hence, suitable since numerous used-computers are available. In addition, it is desirable that addition and deletion of used-computers in an operating system are easy, and distributed storage can flexibly deal with them.

We listed 5 points as characteristics of used-computers. As a storage system that takes into consideration the characteristics of used-computers, we, therefore, assume an inexpensive and highly available distributed storage system that uses numerous used-computers after securing capacity by replacing an existing disk of used-computers with a large-capacity disk as needed. Distributed storage is a mechanism to realize a data grid in which data is distributed to multiple storage. By distributing data across multiple storage devices, high fault tolerance can be achieved. In order to realize high availability, we distribute potential parts of failure, and we redundantly distribute data over numerous used-computers. In addition, it is assumed that an installation location is physically distributed in various places taking advantage of the flexibly selectable characteristic.

As the main use, we assume to handle data that can be processed even with low performance of used-computers, for example, data with low usage or update frequency. A storage system reusing used-computers cannot meet all requirements in various situations. It is, therefore, necessary to clearly separate roles and to use a storage system reusing used-computers in combination with other storage systems, e. g., we should leave data that is frequently utilized or updated to storage products and servers. In the next section, as a necessary measure for constructing the assumed distributed storage system, we discuss a system for automatically constructing a distributed storage environment.

Distributed storage is a mechanism to realize a data grid in which data is distributed to multiple storage. This mechanism has internal multiple storage and realizes high fault tolerance by distributing data. Here, when storing and getting data in distributed storage, it is not convenient if the user or application needs to specify internal storage. It is, therefore, required to hide its internal structure and to be used as one storage. Moreover, if only one piece of data is stored, it becomes impossible to get stored data when the storage with the data fails. It is, therefore, required to store copies of data for multiple storage, and make the data redundant.

2.3 Auto-Construction System

We discuss a system that automatically constructs a distributed system environment for used-computers.

2.3.1 Design

We design a system that automatically builds a distributed storage environment on used-computers. In using used-computers as storage, numerous used-computers are used from their characteristics. Moreover, many works to add used-computers to an existing distributed storage system are necessary. Such as replacement of used-computers when used-computers in failure and adding newly collected used-computers. It is, therefore, extremely inefficient to perform manually necessary works for each used-computer.

In this system, by automating additions required for using used-computers as distributed storage, it reduces works required for adding. In order to facilitate construction and management, this system executes each process for the used-computers via a network and the central server manages all at once. Moreover, manual works should be reduced as few as possible except for physical operation to connect used-computers to a network such as LAN cable connection.

In this system, there are two processes required for used-computers. One is processing for using used-computers as nodes on a network that constitute a distributed storage system. used-computers can be used as nodes even right after they are collected. There are, however, problems that OS is different and it is necessary to delete data before collection. We, therefore, delete an environment before collection by overwriting it with open source OS and use used-computers as nodes after unifying them in an easy-to-use environment for distributed storage.

The other is installing distributed storage software that constructing distributed storage in used-computers. Open source software is also used in distributed storage software as well as OS. By using open source software that can be used for free, used-computers can be reused as distributed storage with low cost.

2.3.2 Implementation

Automatic Installation of OS

We used Kickstart [3] and PXE boot [4] to install OS on used-computers. Kickstart is an automatic installation tool for Red Hat related OS. Moreover, PXE Boot is a network boot mechanism using Preboot eXecution Environment (*PXE*) and PXE boot can install and start OS via a network.

Now, it is necessary to assign IP addresses to used-computers in order to execute PXE boot. For administrative purposes, it is desirable to be able to identify each used-computer. We, therefore, created a script to automatically assign fixed IP addresses. A sent DHCP DISCOVER message is recorded in a log of a DHCP server when PXE boot performs. The script then always monitors the log and extracts a MAC address of a used-computer from a log of a DHCP DISCOVER message. If this MAC address is not found in the DHCP configuration file, the script adds new configuration and reloads the configuration.

We constructed a PXE server that CentOS with Kickstart and PXE boot can be installed. The PXE server is composed of DHCP, TFTP, and HTTP servers. Here, we used [5] for the DHCP server, [6] for the TFTP server, and [7] for the HTTP server. figure 2.1 shows a flow of an automatic installation of OS. For the OS, we used CentOS [8], a free linux distribution from the Red Hat related OS. The DHCP server assigns a fixed IP address to a used-computer by an aforementioned script when a used-computer sends a PXE boot request. Next, the image for running PXE boot is delivered from the TFTP server to used-computers. Then, by executing PXE boot on used-computers, it will download OS image and Kickstart related files from the HTTP server and automatically install OS. automatically.

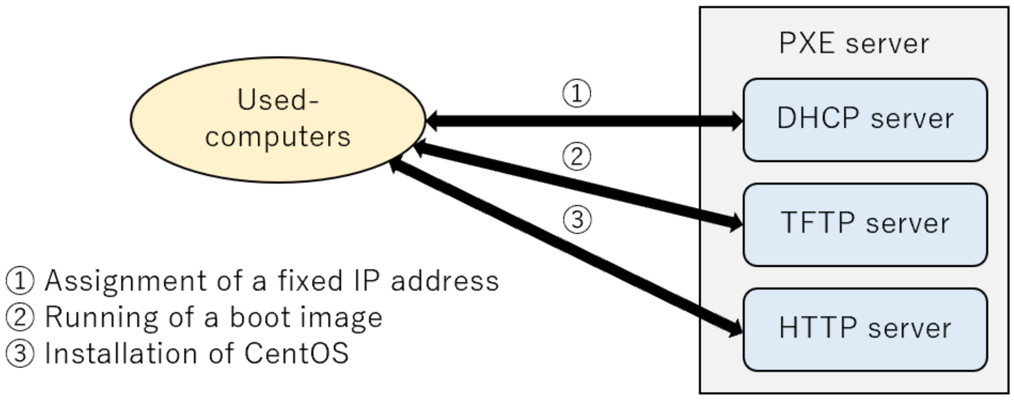


Figure 2.1: Installation of OS.

Automatic Installation of Distributed Storage Software

We focused on object storage [9] , [10] as a type of distributed storage. Object storage manages data in units of an object. Object storage is widely used in cloud storage [11] such as Amazon S3 [12], and they are used in on-premise environments [13].

Object storage uses an HTTP protocol conforming to Representational State Transfer (*REST*) [14] to access objects. Object storage can, therefore, be used like a web application and hide underlying actual file systems such as the Extended File System of Linux [15]. Software that tries to use object storage may, however, not support an HTTP access. We can solve this issue by introducing a gateway such as s3ql [16] that can access object storage and enable to mount object storage as if it were an ordinary physical disk.

Moreover, an object is stored in a flat space that is not a hierarchical structure after provided with a unique identifier indicating a storage location and metadata that can records various information. Since a unique identifier does not depend on a location of a disk where it is actually stored, there is a little restriction on arrangement of data. Object storage, therefore, is easy to move and distribute data, and easy to realize scaling out by storing copies to remote locations and adding disks.

OpenStack *Swift* [17] was used as software for constructing object storage. Swift is a component that realizes an object storage among OpenStack [18] which is open source software for cloud infrastructure. figure 2.2 shows a basic configuration of Swift. In Swift, objects are managed by containers, and containers are managed by accounts. An Object Server stores objects, a Container Server manages containers, and an Account Server manages accounts. All these servers are called a textitStorage Node. A Storage Node is grouped by a unit of a zone, and multiple zones can be created. The same objects, containers, and accounts are stored in each zone. Even if failure occurs in one zone, redundancy and improvement in fault tolerance can be, therefore, realized by the presence of other accessible zones. Moreover, communication between clients and Storage Nodes is authenticated by an Auth Server, and is relayed by a Proxy Server called a textitProxy Node.

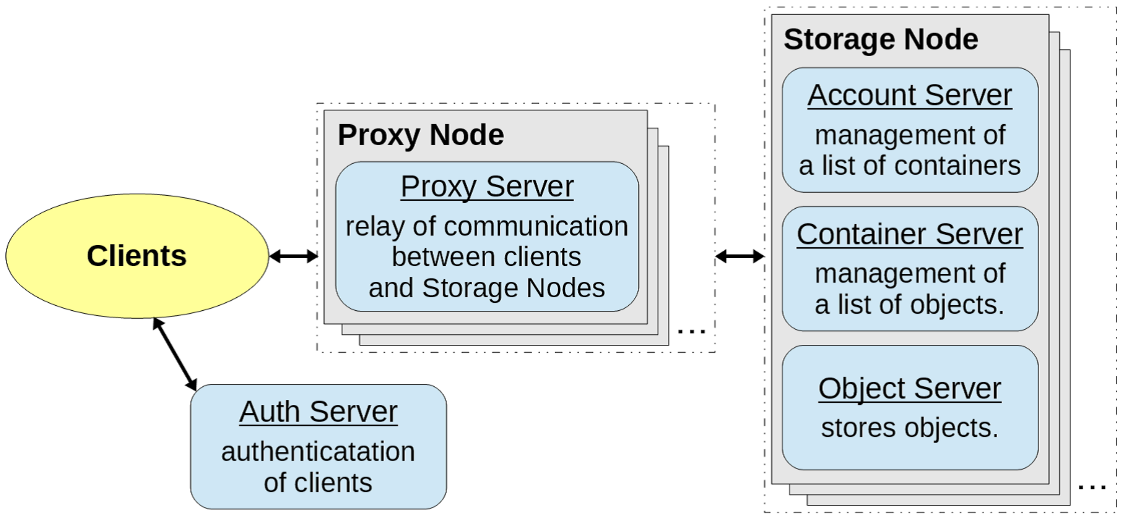

Clients
Proxy Node
Proxy Server
relay of communication between clients and Storage Nodes
...
Storage Node
Account Server
management of a list of containers
Container Server
management of a list of objects.
Object Server
stores objects.
...
Auth Server
authenticatation of clients

Figure 2.2: Basic configuration of Swift.

We used Ansible [19] for an automatic installation of Swift. Ansible can construct an environment for software to operate by writing codes like programming. Moreover, Ansible has idempotency that is the property that the same code always produces the same result. There are Chef [20] and Puppet [21] as similar software. Ansible, however, has the feature that it is unnecessary to install an agent on construction targets. If Python can operate on construction targets that can be connected by SSH, Ansible can be used. Ansible can be, therefore, easily introduced by simple configuration. Basic usage is to create and execute a file called *Playbook* that describes a configuration of an environment that you want to construct. Playbook is described in a general YAML [22] format. Playbook can, therefore, be simply described and never requires to master an especial notation.

First, we constructed an Ansible server on the PXE server. The Ansible server is a server on which Ansible itself is installed. Next, we created Playbooks and related files that describe configurations necessary for installing Swift. Object storage environment by Swift, therefore, constructs automatically on used-computers. Furthermore, since there is no need to create a proxy node and an auth server for each used-computer, and they were created manually and only storage nodes constructed. Moreover, since Python that is necessary for executing Ansible was included as standard, it was unnecessary to perform additional processing for used-computers. For a SSH connection, we used an arbitrary script that can be executed after installing OS by Kickstart. By the script, we automatically established a SSH connection using public key authentication.

2.3.3 Experiment

We actually constructed a distributed storage environment for used-computers by the auto-construction system. figure 2.3 shows the used-computer here. This was dc7900 by Hewlett Packard which became unnecessary by replacement at Tottori

University. Its capacity of HDD was 160 GB and its speed of NIC was 1 Gbps. Moreover, its dimension was as follows: height 66 mm, wide 251 mm, long 254 mm. We used 15 used-computers for this experiment, and we distributedly installed them on two remote campuses in Tottori University. figure 2.4 shows installation status on one campus. We used an aluminum rack to effectively utilize space.

Figure 2.3: The used-computer.

Each used-computer was aggregated and connected to one switching hub for each campus. Moreover, their switching hubs were connected to the experimental network. Speed of its ports was all 1 Gbps. After installation of used-computers, we executed PXE boot. Moreover, we confirmed that an installation of OS by PXE server and an installation of Swift by Ansible server were automatically completed. Total capacity was about 2,400 GB and by using 15 used-computers. Moreover, capacity that can actually be used as distributed storage excluding an area for OS was about 2,100 GB. In addition, the amount of space that can be used as actual data storage will decrease further as the number of zones of Swift is increased.

Through this experiment, we needed manual operation such as wiring and execution of PXE boot. We, however, confirmed that an auto-construction system could automate the most of a processing and utilized used-computers as distributed

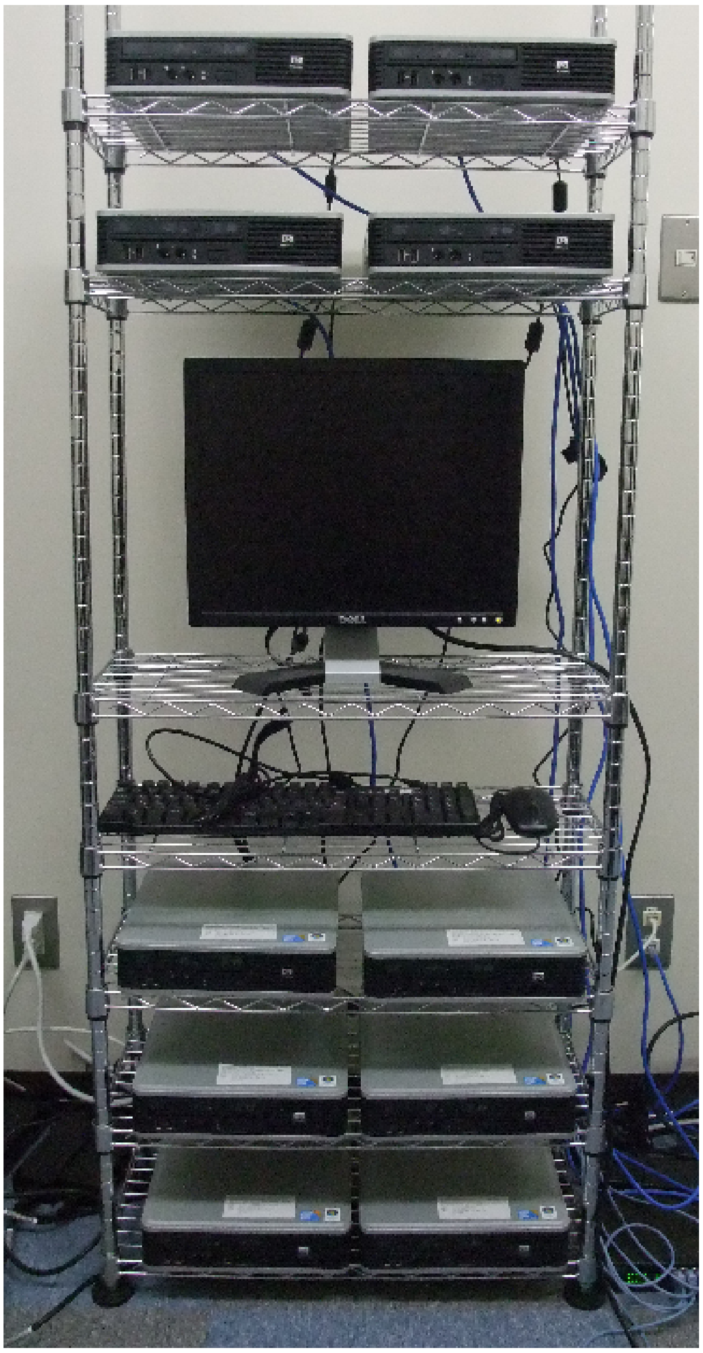


Figure 2.4: Installation of used-computers using an aluminum rack.

storage. We measured the required time to construct a distributed storage environment for one used-computer as a reference. The result was about 5 minutes and 40 seconds from turning on power until installation of OS was completed. Moreover, the result was about 2 minutes from installation of OS was completed until installation of Swift was completed. These can be processed in parallel. Therefore, operation time required for construction was much shorter than in the case where all operation of construction was manual.

2.4 Operation and Evaluation

In order to confirm that used-computers can actually be used as distributed storage, we incorporated the 15 used-computers that were constructed into the existing distributed storage system of Tottori University (hereinafter referred to as the *existing environment*) and operated it. Here, the Auth Server, Proxy Server, Account Server, and Container Server operating in the existing environment were used. Therefore, only the Object Server was built and operated in all used-computers.

The configuration of the Object Server for the entire system is shown in Figure 2.5. Figure 2.5 shows the configuration of the Object Server of the entire system after embedding. Swift's 1 to 5 are built on ProLiant from Hewlett Packard Enterprise [23]. Threr are general-purpose servers, and each of them has multiple disks. The total capacity is 72 TB, but for the sake of availability, three zones are set up and the same data is placed in all zones, so the actual capacity that can be stored is 24 TB. In addition, swift 5 originally has a capacity of 24 TB, but due to the addition of 2 TB from a used-computer, the configuration was changed so that only 22 TB is used. After the installation, we operated the system for more than half a year months as a backup storage area for Tottori University's online storage and mail server [24]. As a result, the problem was only storage failure on one used-computer. We were able to confirm that the used-computers can actually be used as distributed storage.

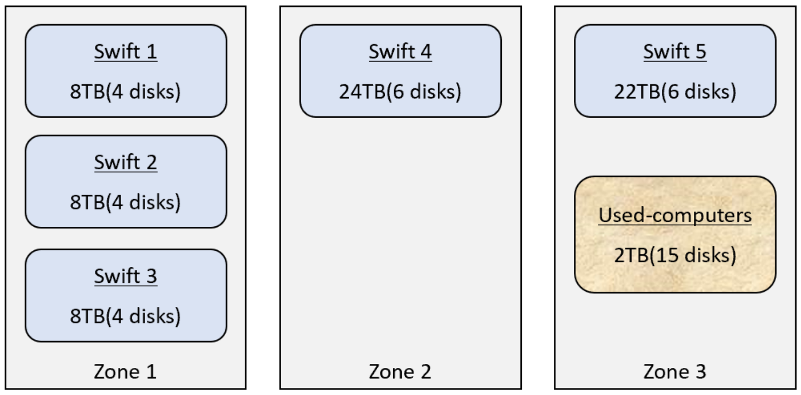


Figure 2.5: Configuration of Object Server.

In addition, in order to verify the difference in performance between the server in the existing environment and the used-computer, we investigated the time required for writing and reading when the size of the data was varied. The time required for writing is shown in Table 2.1, and the time required for reading is shown in Table 2.2. Zone 1 is a specific disk (2 TB, SATA2, 7200 RPM) with swift 3 (CPU: Xeon E3-1220 V2 3.10 GHz, memory: 4 GB, NIC: 1 Gbps), and zone 2 is a specific disk (2 TB, SATA2, 7200 RPM) with swift 4 (CPU: Xeon E3-1240 V2 3.40 GHz, memory: 8 NIC: 1 Gbps) with a specific disk (4 TB, SATA3, 7200 RPM). Zone 3 shows the results for a specific disk (160 GB, SATA2, 7200 RPM) in a specific used-computer. In addition, the disk write and read operations are performed using the swift command, which can be executed in the CLI, on a specific Proxy Server. The time required for writing was extracted from the Object Server log, and the time required for reading was extracted from the execution result of the swift command. The results are averaged over three runs.

Table 2.1: Time required to write.

Data (MB)	Zone 1 (s)	Zone 2 (s)	Zone 3 (s)
0.1	0.0200	0.0149	0.0235
1	0.0327	0.0350	0.0761
10	0.2338	0.2366	0.4044
100	3.8651	3.9683	5.8560
1000	50.8456	50.6713	56.6798

Table 2.2: Time required to read.

Data (MB)	Zone 1 (s)	Zone 2 (s)	Zone 3 (s)
0.1	0.0287	0.0253	0.0770
1	0.0353	0.0500	0.0680
10	0.1883	0.1543	0.3177
100	1.1447	1.1137	2.9703
1000	10.7987	10.5960	49.5640

In terms of the time required for writing and reading, it can be seen that the used-computers in Zone 3 take longer than the servers in Zone 1 and Zone 2. This is due to the difference in the performance of the disks and memory of the server and the used-computers. In practical use, if the data size is 10 MB or less, the difference is small and the impact is small. However, for data sizes of 100 MB or more, the difference in the time required is large, in seconds, and the impact is especially large for the 1000 MB case, where the difference is about 50 seconds. As a countermeasure, the size of the data to be saved should be limited to small ones, or the data should be divided into chunks before writing or reading.

2.5 Discussion

We have confirmed that used-computers can be used as distributed storage. Although the read/write speed is slower than when built on servers, it is sufficient for backup and other applications where speed is not important.

In this operation, the cost was 0 yen because we used existing virtual machines that built each server, as well as switching hubs, LAN cables, and power strips that were necessary in addition to used-computers. Each server can also be built on used-computers, and even if we had to purchase new switching hubs, etc., the cost would have been several tens of thousands of yen. Moreover, a 4TB 3.5-inch HDD or a 2TB 2.5-inch HDD can be purchased for about 10,000 yen, so even if you need a large capacity, you can install it cheaply.

Here, we compare the cost with the existing environment. Based on the actual cost of the existing environment, the installation cost was 2.4 million yen including maintenance for one year, and the maintenance cost was 400,000 yen per year. On the other hand, when using 18 used computers to secure the same capacity as the existing environment, the initial cost was 360,000 yen for 18 4TB HDDs, and the maintenance cost was only the replacement cost in case of HDD failure.

On the surface, the result is that used-computers are cheaper. However, this comparison fails to take into account labor costs and power consumption. When these factors are taken into account, it cannot be ruled out that used-computers are cheaper. Another problem is the need to secure an installation area for the used computer. In some cases, it may be more cost effective to install a new server or storage device instead of using the old computer. This is a difficult decision to make.

2.6 Related Works

Building a distributed storage system using general computers has already been tackled, mainly because it is effective in terms of cost effectiveness.

The Google File System [25] is a distributed file system that is designed to build a large-scale storage system using general computers. This system is designed to allow for the failure of the computer in which it is installed, and focuses on automatic recovery of data without losing it in the event of a failure. There are other distributed file systems such as Ceph [26] and GlusterFS [27]. In addition, a chained network RAID has been proposed to build a highly functional and inexpensive storage for online storage services using general computers [28]. This system is designed to provide high reliability for the data stored even when using unreliable computers, and is designed to be flexible enough to accommodate the addition or replacement of computers and to achieve scalability in the processing performance of the entire system. However, these existing studies mainly deal with the design and implementation of using computers as distributed storage, and do not describe in detail the processing methods required to actually use computers as distributed storage. It also assumes the use of new computers, and does not mention the use of used-computers.

There is [29] to build a large storage capacity by connecting a large number of general computers via the Internet. [29] is a distributed storage infrastructure developed to back up and share petabytes of data. The computer provided by the volunteer will be used as part of the giant storage. This is based on the assumption that existing computers will be used. However, it is an Internet-scale system and there is no mention of on-premise operation.

2.7 Conclusion

In this chapter, we have organized characteristics of used-computers and considered a storage system based on their characteristics. Results of considering, we have assumed an inexpensive and highly available distributed storage system using numerous used-computers after replacing existing disks of used-computers with large capacity as needed to secure capacity. Moreover, in order to reduce and facilitate labors required for construction and operation, we have implemented an auto-construction system of a distributed storage environment for used-computers and experimented. Through this experiment, we have confirmed that the most of the processing of construction were successfully automated and we could reuse used-computers as distributed storage. We also confirmed that the system can be used without any major problems after operating it in an actual environment for more than half a year.

Chapter 3

Distributed System using Idle Resources of Using-Computers

3.1 Introduction

The performance of computer components such as CPU, memory, and storage has been improved by technological advancement. On the other hand, the price according to these performance has downed. As an example, HDD is no longer rare to have the capacity of a terabyte class. Its price per gigabyte, however, is lower than when a gigabyte class was common. We, therefore, can use high-performance computers at a lower cost than before. Many computers are used for business in organizations such as companies and universities. When these computers are renewed, if a budget is the same as before, computers with higher performance than before can be introduced because of the price according to its performance has downed. Even if we try to introduce computers with low performance to reduce its cost, it is difficult to improve overall performance. There is, therefore, a limit in the introduction of low-performance computers, and we have to introduce computers with a certain extent of performance.

Here, the most common usage of computers in organizations is for office work such as document preparation and spreadsheet. Office work is a relatively light operation for modern computers. Thus, the computation performance of the computer is not utilized to the upper limit and is partially idle. Moreover, office work uses small size data relatively. In addition, for data sharing and security measures, organizations often store data in a shared storage prepared instead of built-in computer storage. Thus, built-in storage capacity is too much for office work, and unused space is left idle. For these reasons, computers in organizations have idle resources such as unused computing ability and storage capacity. The same can be said for using-computers that are over-performing for their intended use. Idle resources do not adversely affect the use of computers. These resources, however, are wasted despite use-value. We should utilize idle resources effectively for the spirit of "don't waste what is valuable". This spirit is expressed as MOTTAINAI [30] in a Japanese term.

On the other hand, the amount of data handled by an information system continues to increase, and storage for saving data is indispensable. Since the capacity of each storage is limited, when the amount of data to be stored increases, free storage space becomes insufficient. In this case, there are two ways to deal with this situation: reducing the amount of data and adding new storage. A method of reducing the amount of data depends on the property of data, such as type, content, and storing period. In contrast, a method of adding new storage is more versatile because it can accommodate any data by adding storage. In this method, the problem is costs. When a budget is limited, the costs of introducing and operating storage should be low to reduce its cost. Moreover, cloud storage is an option for introducing storage. This storage can flexibly set a usage form and capacity and is rapidly spreading. It, however, cannot be used when data cannot be stored outside due to regulations or security reasons. In this case, storage needs to be introduced in an on-premises environment.

In this chapter, as a means to introduce and operate storage at a low cost in an on-premises environment, we propose to utilize the idle resources of using-computers in an organization. We define a part of free capacity of using-computers as *surplus capacity* that is utilized as distributed storage. Distributed storage is a usage form in which multiple storage units are one virtual storage unit and data is distributed among it. Thus, even if surplus capacity of each computer is small, we can use it as a large capacity by using them as distributed storage. Moreover, the distributed data also improves availability and security measures. By utilizing surplus capacity as distributed storage, we can solve a problem of costs for adding storage and realize the MOTTAINAI spirit.

3.2 Design

3.2.1 Surplus Capacity

In order to utilize surplus capacity as distributed storage, we first describe requirements of distributed storage. Next, we organize characteristics of surplus storage and using-computers with surplus storage in an organization and consider appropriate utilization environment and method.

Surplus capacity is available capacity as distributed storage among total storage capacity and is a part of free capacity. Since the storage capacity of computers is different from each other, free space is also different. Moreover, how much free space is reserved and how much free space can be considered as surplus capacity depends on the usage of computers. It should be noted that if free capacity is all surplus capacity, surplus capacity must be smaller than free capacity since it cannot cope with the case where used capacity other than a distributed storage increases. Moreover, the usage state of computers always changes, and the capacity used for storage always fluctuates. If we define surplus capacity as a fixed capacity for each storage, it is unnecessary to consider fluctuation. We, however, want to make the best use of idle resources. Surplus capacity should fluctuate, and we use the largest surplus capacity for distributed storage. It is, therefore, necessary to have a mechanism to monitor a usage state of computers and each capacity of storage and to adjust surplus capacity dynamically.

In addition, storage has different performance other than capacities, such as read/write speed and degraded condition. At present, the general storage in the computer is HDD or SSD. The various performance of storage is greatly different from types of storage, and there are individual differences. The difference in read/write speed may be used as an index for which storage is used to store and get data. This enables efficient data management, such as storing frequently used data in high-performance storage. The degraded condition is greatly different by utilization history in addition to individual differences. In addition, using surplus capacity as distributed storage may cause the storage to degrade and become prone to failure. It is, therefore, necessary to take measures such as not using storage in bad condition by grasping the degraded condition of storage. In addition, it is necessary to ensure redundancy and have high availability for storage failure by copying and storing data.

We use some of the storage that using-computers have as surplus capacity. These computers have primary uses, such as office work. Its operation rate varies greatly depending on its application and usage form. For example, server computers to provide an information system all the time are always running in principle. On the other hand, office computers are turned on only when they are used and may be shut down when they are not used. Since the operating rate differs depending on computers, it is necessary to use corresponding to it. In computers with high operation rates, it is considered to operate essential functions in distributed storage such as the provision of a utilization interface and management of each surplus capacity and stored data. In computers with low operation rates, on the assumption that it may not be used due to shutdown, it is considered to store data of low utilization frequency and backup data.

3.2.2 System Structure

Figure 3.1 shows the structure of a proposed distributed storage system. This system is composed of two kinds of computers. One is a management computer which mainly manages systems and provides services. The other is a storage computer that mainly provides surplus capacity to the system These computers must be on a network that can communicate with each other.

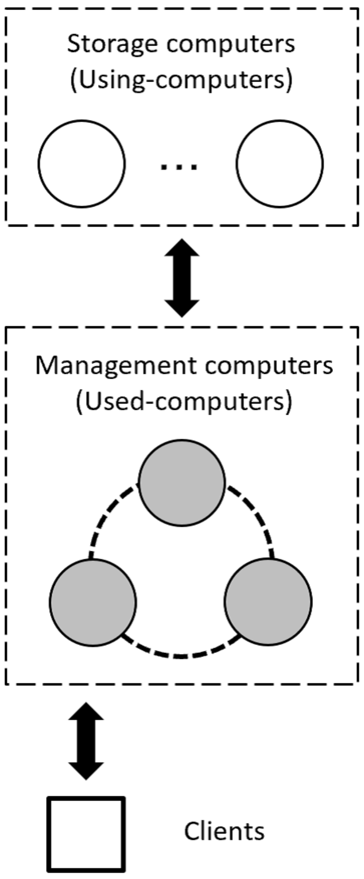


Figure 3.1: The structure of proposed distributed system.

Clients connect to one of the management computers when using the system. The connection destination is selected at random by a load balancing technique such as DNS round-robin. Moreover, a storage computer or a management computer may be a client.

3.2.3 Management Computers

A management computer provides an interface to clients. Clients, therefore, can use basic functions as distributed storage such as storing, getting, updating and deleting files. We use multiple used-computers that are available at all times to run at least one management computer. We, therefore, can realize high fault tolerance and load balancing. Moreover, they provide surplus capacity to the system and uses them to store and manage data. When the used capacity increases or decreases, surplus capacity is calculated dynamically, and the capacity provided to the system is changed. In this way, we can utilize idle resources to the maximum.

Though we use multiple computers as a management computer, there is a possibility of shutdown and failure, and they are not always available. It is necessary to have a configuration that can dynamically join and leave the system. For this reason, we apply a Peer to Peer (*P2P*) model as an architecture of management computers. Several technologies realize a P2P model. Here, we use a distributed hash table [31], [32]. With this technology, management computers are connected via a virtual network so that any management computer can communicate with all other management computers. Moreover, it can flexibly cope with the change of a virtual network composed of management computers such as joining and leaving of them. In addition, we can use a distributed database (*DDB*) [33] corresponding to a relational database in a distributed environment. By using a DDB, various information can be shared by each management computer.

3.2.4 Storage Computers

A storage computer is mainly provided surplus capacity to the system. We use multiple using-computers for storage computers. A P2P model is not applied like management computers. By mainly using it as a storage location for files, an impact on ordinary uses such as office work is reduced. The process of joining and leaving a storage computer the system is performed by requesting a management computer. Moreover, the process of increasing or decreasing surplus capacity of storage computers is the same as that of management computers. In addition, storage computers turn up only when necessary for office work, etc. Unlike management computers, they are not always running and may shut down.

3.2.5 File Encryption and Redundancy

We store files in the surplus space of management and storage computers. These computers are usually used for other purposes. There is, therefore, a sufficient possibility that stored files are got illegally. Thus, we encrypt files when they are stored. Encrypted files cannot be read without decryption. Encryption and decryption processes are performed only in response to file operations via management computers. Thus, even if files are got directly from storage, they cannot be read and an information leakage risk can be reduced.

When storing a file, if the file is stored in only one storage, the file cannot be got in the case of storage failure or file corruption. Moreover, when accesses are concentrated on the specific file, a response speed may decrease by a load of the storage and the computer in which its file is stored. Thus, when we store data in the system, we make redundant data by storing its copies in multiple storage that have surplus storage. Distributed copies of data improve availability and distribute access to data.

Encryption and redundancy are performed respectively. The first step is file encryption, and the next step is to make a copy of the encrypted file. By the way, these processes can be performed collectively by a secret sharing scheme (SSS). Shamir's SSS [34] is widely recognized as an SSS. In this method, the original file is divided into multiple data called shares, and the original data can be restored only when the necessary number of shares are collected. Original data will not be restored unless the necessary number of shares are leaked. Moreover, if a portion of shares is lost or corrupted, the original data can be restored if there are more shares than needed. It, therefore, can be used to improve confidentiality and availability. It, however, much time to process a SSS. For this reason, when using a SSS, it is necessary to evaluate whether it can be processed at a practical speed.

3.2.6 Storing and Getting of Files

When storing a file, it is assumed that encrypted files are first copied and distributed. An operation rate of management computers is high, and surplus capacity of their computer can be accessed. In contrast, the operation rate of storage computers is low, and surplus capacity of its computer may not be able to access. For this reason, if a file is stored only in a storage computer, the file may not be got due to the shutdown of the storage computer. We, therefore, basically store the original file on a management computer and copies on storage computers. Moreover, an importance level can be set as an option for each file. A file of a high importance level is regarded as a file that needs to be got all the time, and its copies are also stored on management computers. In contrast, a file of a low importance level is regarded as a file that cannot be got, and the original file is stored in storage computers as well as its copies. This importance level is specified when the file is saved. By the way, even if a storage computer is shut down, if the computer supports Wake-on-LAN, which allows the computer to turn on via a network, we can access stored files by forcibly starting the storage computer.

In a management computer group or a storage computer group, we select which computer's surplus capacity to store files according to the performance of its storage. For the performance of storage as an index, the size of its surplus capacity and its degradation state are used. This information is notified from a computer with its storage to management computers whenever there is a change. By default, files are stored preferentially in storage with large surplus capacity. We, however, do not store files to a storage that has a bad deterioration state, except in an emergency when it cannot be stored to another storage. We can use Self-Monitoring Analysis and Reporting Technology (*S.M.A.R.T.*) [35] as an indication for a deterioration state. By setting a threshold to the number of bad sectors that can be obtained by S.M.A.R.T., it is possible to determine whether a deterioration state is good or bad.

When getting a file, it is tried to get it from storage computers first. If it is not present in storage computers, it is got from management computers. In a management computer group or a storage computer group, we get a file from which computer's surplus capacity to store files according to the performance of its storage. In a management computer group or a storage computer group, it is accessed in the order in which the reading performance of storage is high. We can get the reading performance information of storage in S.M.A.R.T..

3.2.7 Relocation of Files

If each computer leaves the system, it notifies a management computer that it is leaving. If leaving of a computer is temporary, stored files it holds intact. If it is a permanent leaving, stored files in the storage of its computer are relocated to surplus capacity of another computer. The method for selecting a storage location for relocation is the same as that for selecting a storage location for files. Moreover, each computer may leave illegally due to a failure or failure without notice. Then, alive monitoring is carried out for each computer from management computers.

As used capacity increases, surplus capacity decreases. If this results in a shortage of surplus capacity, we free up space in preparation for further reductions in surplus capacity. In this case, surplus capacity is made available by relocating stored files.

3.2.8 Management of Various Information

Various information is necessary for the processing of this system. It is good that each computer can always share the latest information instead of getting the information sequentially. For this reason, tables for managing various information are created by a DDB.

First, a computer table for managing various information about computers is created. Table 3.1 shows the definition of this table. An identifier of a computer also distinguishes a management computer from a storage computer. An IP address of a computer is used to communicate with the computer. The operating status of a computer is used to judge whether or not the computer is in a temporary leaving state.

Table 3.1: Definition of a computer table.

Items
An identifier of a computer
An IP address of a computer
Operating status of a computer

Next, a storage table for managing various information about storage is created. Since one computer may have multiple storage, it is managed separately from a computer table. Table 3.2 shows the definition of this table. Total capacity, used capacity, and free capacity of storage is obtained from each computer to management computers in the same way as surplus capacity.

Finally, a file table for managing various information about stored files is created. Table 3.3 shows the definition of this table. A file table only stores information

Table 3.2: Definition of a storage table.

Items
An identifier of storage
Total capacity of storage
Used capacity of storage
Free capacity of storage
Surplus capacity of storage
Reading performance of storage
A deterioration state of storage
An identifier of a computer with storage

about files such as an identifier and capacity and is used as an index for stored files. An entity of each file is stored directly in each storage separately from a distributed database. A path when storing files is fixed, and it is not necessary to store information about a path in this table.

Table 3.3: Definition of a file table.

Items
An identifier of a file
Name of a file
Size of a file
An importance level of a file
An identifier of a copy

We can access files by referring to a file table, a storage table, and a computer table in that order.

3.3 Dealing with Leaving of Using-Computers

In order to implement proposed system in its entirety, we first consider leaving of using-computers.

Using-Computers have an inherent purpose, and at unspecified times, the operating system may shut down, suspend, or disconnect from network communication. Due to these causes, when a node leaves the system, the number of file replicas to ensure redundancy is reduced.

Therefore, in order to maintain or recover the number of file replicas, it is necessary to have a function to detect when a node leaves the system and to recover the file replicas. However, since the appropriate or possible detection and recovery processes differ depending on the cause of the leaving, we categorize the leaving methods based on their processing differences and propose appropriate detection and recovery methods for each type of leaving.

3.3.1 Classification of Leaving

Node leaving here refers to the state in which one node becomes inaccessible to other nodes. Reasons for leaving a node include OS shutdown, suspend, and disconnection of the network connection. In these cases, the files held by the leaving node may or may not be able to be saved to other nodes. There are also cases where the leaving node may or may not be able to declare its intention to leave to a group of other nodes. For example, there are cases where the leaving node does not have enough time to wait for the completion of file transfer, or where there is no room to communicate with other nodes, such as in the case of forced shutdown of the OS. In order to maintain the number of file replicas efficiently, it is considered that appropriate processing is necessary depending on the combination of these possibilities.

Here, when a node leaves, it notifies other nodes of its intention to leave, which is called *leaving declaration*, and when a node leaves, it saves its files to other nodes, which is called *self-replication*.

If the node can wait for the shutdown to complete, it can afford to run leaving declaration and self-replication. However, in the case of a forced shutdown, these actions are not possible, and other nodes need to be able to detect this. Therefore, the nodes joining in the system maintain a ring network in the lexical order of IP addresses, and any node that cannot communicate on the ring is considered to have left the system without being able to make a declaration. The IP addresses of a group of nodes are managed in a distributed database, and each node can repair the network between nodes by referring to the distributed database, although this involves a reference cost.

The possible operations of a leaving node can be classified into three stages according to the reason for its departure.

The first stage is when a node can wait for both the leaving declaration and self-replication to be completed.

The second stage is when a node can wait for only the leaving declaration to be completed. Self-replication requires a longer transfer time depending on the data size of the file held by the node, but only the leaving declaration requires a shorter time. In addition, when other nodes receive the leaving declaration, they can quickly cooperate to recover the number of file replicas.

The third stage is when a leaving node neither leaving declaration nor self-replicate. In this case, a leaving node is detected from the disconnection of the ring network, and immediately after the detection, the group of nodes cooperates to recover the number of file replicas. This process is called *event-driven type*. However, there is a possibility that the number of replicas will not recover normally due to further disconnection during the recovery. Therefore, the system periodically checks the number of file replicas held by each node and recovers the number of file replicas that are less than the default number. This process is called *polling type*.

Although the first step is desirable for administrators, the second and third steps are likely to be more common since they cannot control using-computers.

3.3.2 Implementation

We implemented a way to deal with leaving and rejoining of using-computers. The development language was Java, and the project was managed by Apache Maven. The project was managed by Apache Maven [36]. Apache *Cassandra* [37] was adopted as the distributed database, and the DataStax Java Driver for Cassandra [38] was used to control Cassandra using the Java language. Cassandra manages and shares information about the nodes joining the system and the files held by each node.

First, we describe the implementation of the case with leaving declaration. In the case of self-replication, it moves its own files to the other node with the largest surplus capacity, and then leaves. In the case without self-replication, the node first notifies an other node that it is leaving. The node that receives the notification will then create a copy of files owned by the leaving node. If the node already has files, it further instructs the other node to replicate the file.

Next, we describe the implementation for the case without leaving declaration. In the case of event-driven type, the system detects that a node has left the system by not being able to acquire a file. The node that detects the departure makes a copy of the file owned by the leaving node. If the node already has the file, it further instructs other nodes to replicate the file. In the case of the polling type, the number of replicas is periodically checked, and when the number is less than the specified number, the node is considered to have left. Since the node that detects the leaving already has a copy of the file owned by the node that left, it instructs other nodes to copy the file.

3.3.3 Experiment and Discussion

We experimented with processing during classified node leaving. The experimental environment consisted of four computers with the following specifications: CPU: Intel Core i5-6500, RAM: 16 GB, SSD: 240 GB, OS: Debian 9.11. The number of file replicas maintained by the system was set to three.

First, we conducted an experiment to test how the presence or absence of self-replication affects the case where leaving declaration is present. In the experiment, we measured the processing time of one leaving node by varying the size and number of files it has. For the size change, we varied one file by 50 MiB. For the number change, we varied 10 MiB file the number of files by 5. In the case of no self-replication, we measure the processing time when the node that is declared to leave and the node that leaves do not possess the same file, which requires the most processing steps to recover the replication.

Figure 3.2 and figure 3.3 shows the results of the experiment. We confirmed that there was no difference in the processing time when the file size was changed. We confirmed that the self-replication process is more efficient than the non-replication process when the number of files is changed. However, considering that the difference in processing time is small and that the users of using-computers need to wait until the self-replication is completed, the necessity of self-replication when using-computers leave the network is considered to be low.

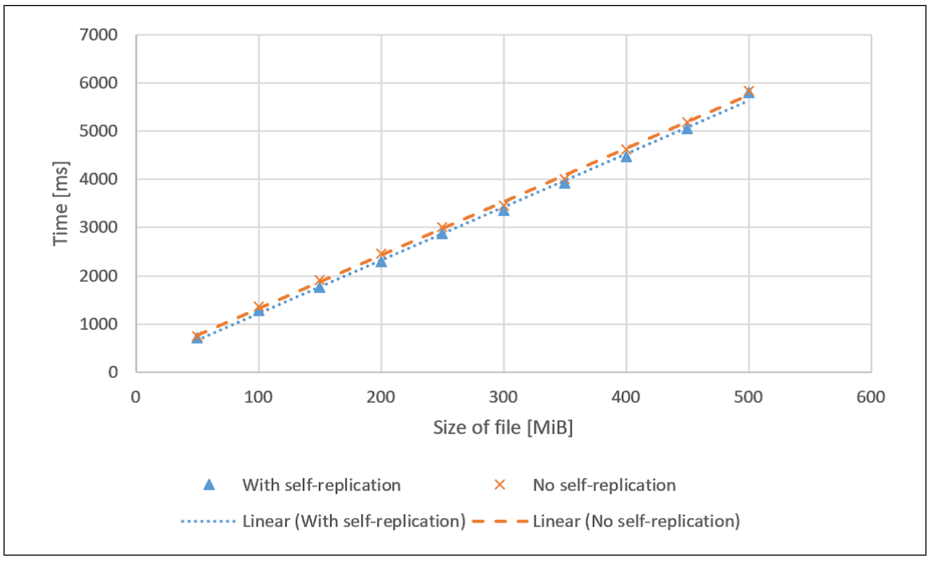


Figure 3.2: Comparison of the presence or absence of self-replication when the file size is varied.

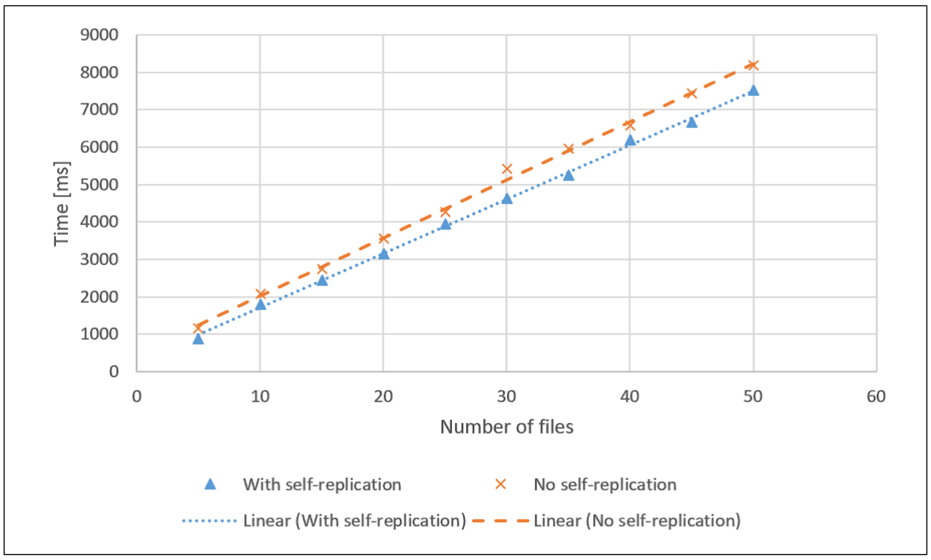


Figure 3.3: Comparison of the presence or absence of self-replication when the number of files is varied.

Next, we conducted an experiment to test the properties of event-driven type and polling type in maintaining the number of file replicas. The experimental conditions were the same as in the previous experiment. In the case of the event-driven type, the processing time required to recover a replication varies depending on the file held by the node that detected the leaving. Therefore, we measure the processing time for the case where the processing time is the longest, when the detected node does not have any replicas to recover.

The experimental results are shown in Figure 3.5 and Figure 3.5. In the polling type, the number of files has a larger effect on the processing time than in the event-driven type. The effect of the file size on the processing time is the same for the event-driven type and the polling type. On the other hand, the effect of the number of files on the processing time is much larger for event-driven type than for polling type. In addition, in the event-driven type, since multiple replicas are recovered together when a leaving is detected, the load of recovering the replicas may be concentrated on some nodes for a short time. However, this becomes a problem when the size or number of files exceeds a certain level. This suggests that the event-driven type recovers a large number of replicas of files with small sizes, while the polling type recovers a small number of replicas of files with large sizes, thereby compensating for the shortcomings of both types.

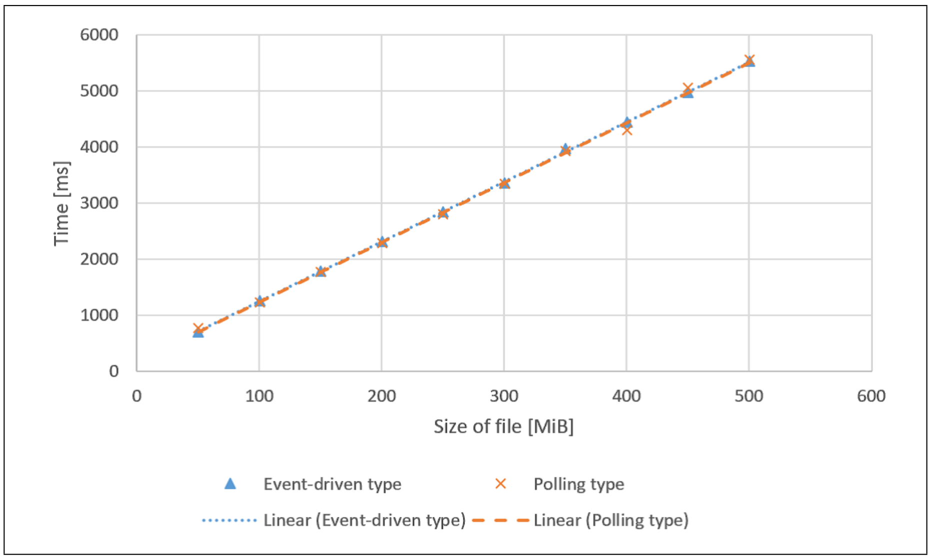


Figure 3.4: Comparison of each type when the file is varied.

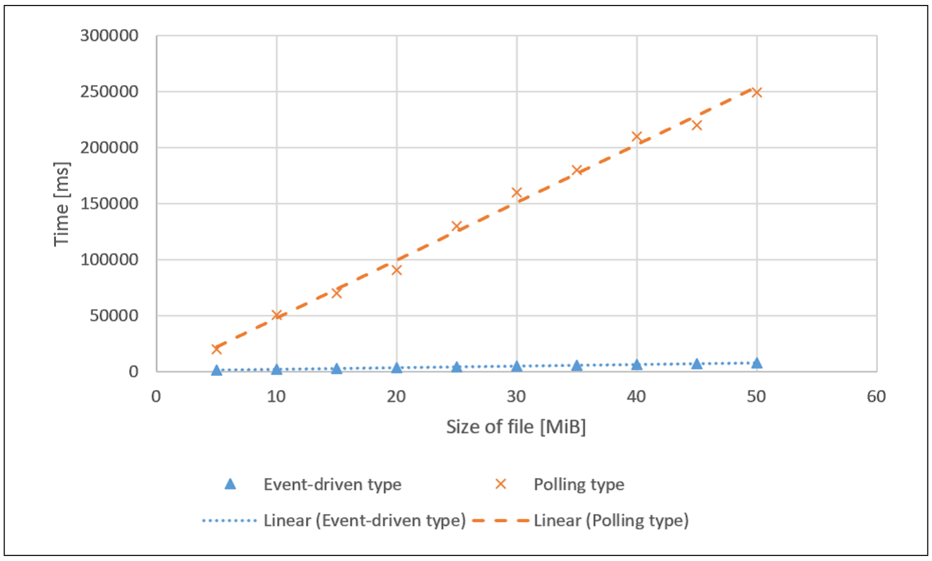


Figure 3.5: Comparison of each type when the number of files is varied.

3.4 Related Works

As an example of utilizing idle resources of using-computer, volunteer computing [39] is a worldwide effort. This is an effort to utilize computing resources provided by volunteers to perform tasks that are difficult to perform with a small number of computing resources and require a large number of computing resources. The donated computing resources are connected through the Internet based on the concept of grid computing to form a large computer system with computing resources comparable to a supercomputer. As an example of volunteer computing, there are many projects using BOINC [40], which is open to the public for free, and is used for various researches such as astronomy and biology. However, BOINC is mainly intended for large-scale computational processing. There is an [29] about building storage with volunteer computing, but it is an Internet-scale system and does not mention on-premise operation.

Among idle resources, surplus storage capacity is mainly used as distributed storage. As examples of the use of surplus capacity as distributed storage, there are several storage services such as Storj [41] and Sia [42] that use blockchain technology. These are cloud storage services via virtual currency, and the user must pay for using the storage. On the other hand, they can receive compensation for providing storage. The data is not stored on the storage managed by a specific service provider as is the case with cloud storage, but on the storage of each user's computer. This is a decentralized system in which data is encrypted and stored in a distributed manner on each user's computer. This system has strong security. However, the use of virtual currency is required to use the system, and it cannot be used only within a specific network such as an organization.

There are cases where the surplus capacity of each computer is connected via iSCSI to form a single high-capacity storage unit. In [43], the surplus capacity of each computer is connected by iSCSI and used as a single large storage. By adding information such as storage capacity and read/write performance to the iSCSI communication packets, the information is mutually shared among the storage devices, and data is dynamically allocated, such that data that is frequently read/written is stored or moved to high-performance storage, and data that is infrequently read/written is stored or moved to low-performance storage. In [44], the data is encrypted by public key and hysteresis signature is added to the surplus capacity. In [45], the extra capacity is used to reduce the cost of backing up the data. In [45], extra capacity is used to share and back up data securely, easily, and at low cost. In [47], the authors propose a decentralized storage service that uses secret sharing techniques to store data in a distributed manner, and uses surplus capacity to achieve confidentiality, availability, and low cost. In these studies, there is no detailed description of what happens when an using-computer with surplus capacity stops. In addition, there are various methods of using surplus capacity, but the surplus capacity used is a fixed capacity, and it is not assumed to change dynamically.

3.5 Conclusion

In this chapter, we proposed to utilize the surplus capacity of using-computers as distributed storage. First, we designed the system based on the fact that the using-computers have their original purpose. Next, we studied the process to maintain the number of file replicas when a node leaves the system as a partial implementation of the system. As a result, we defined the processes of leaving declaration and self-replication, and implemented and evaluated them.

Chapter 4

E-Learning System using Idle Resources of Each Computer

4.1 Introduction

E-Learning systems that enable learners to study by themselves without a teacher have widely deployed in universities, high schools and companies. Students can study anytime and anywhere by themselves. For example, seventy six percent of national universities in Japan introduce an e-Learning system [48]. Moreover, eighty percent of companies introduce the system in a survey of 360 Japanese companies [49]. The market size of e-Learning system increases year by year, and it will reaches to 312.6 billion in 2021 in Japan [50].

It is important that an e-Learning system can be used continuously without delay in education and training. The system, therefore, must be able to manage a lot of e-learning contents and handle a lot of access from students There are various e-Learning systems, most of which are implemented as a client-server (CS) model. For example, Moodle [51], one of the famous e-Learning system, is a client-server model. In such a CS based e-Learning system, the concentration of loads on a server causes low response time, and the server becomes a single point of failure. We, therefore, have to prepare multiple servers to distribute computation/storage. Moreover, additional equipment such as a load balancer is also indispensable.

In order to solve these problems, we have developed a distributed e-learning system integrated a P2P model with a CS model. In a P2P model, a system is constructed from multiple machines called nodes. By applying a P2P model to the server part of the CS model, this system distributes functions of a server to multiple computers. A system with this structure is called a hybrid P2P-based system [52]. This system, therefore, has overcome the problem of a single point of failure and concentration of loads in a CS model.

Here, if we can use many computers as nodes, higher fault tolerance and more load distribution can be realized. It, however, requires additional costs to prepare computers to be used as nodes. It, however, requires additional costs to prepare computers to be used as nodes. Considering the performance required for an e-Learning system, high performance is needed during peak times when large amounts of content are used simultaneously, but not as much performance is needed during normal times. Preparing nodes for peak times will increase the cost, but if they are introduced for normal times, they will not be able to handle peak times. For this reason, it is desirable to be able to flexibly adjust performance as needed while keeping costs low.

Here, there are many computers in an organization used for office works. We discuss to use such a computers as a node. In recent years, computers have become inexpensive but have high performance. Thus office computers have also become high performance. However, their performance is not fully utilized. In other words, their office computer affords to computer something except office works. Moreover, cloud services become popular and we often store data on a shared storage for data sharing and security. This increases, free storage space inside the office computers. These computational resources and storage are idle resource, and should be utilized for other purposes.

In this chapter, we propose to use idle resources of using-computers for a distributed e-Learning system. If we can use such an idle resources, we do not need to prepare new computer to be used as a node. No additional cost is required. Moreover, computers with idle resources are usually used for other purposes such as office work. It, therefore, may not be available due to turn off. For this reason, a dedicated server with all the functions of a server and running at all times is prepared by using used-computers. We use using-computers by temporarily incorporating them into the system only when it is necessary to perform load balancing or replacement in the event of a failure.

4.2 Overview of a Distributed e-Learning System

A web-based system is the most popular for an e-Learning system [53], [54]. Figure 4.1 shows the overview of a web-based e-Learning system. There are teachers, learners, and managers. Users study by themselves in a web browser through a computer or a smartphone they have.

A distributed e-Learning system is also a web-based system. Main functions of our system are to provide learning contents and online test. A teacher uploads learning contents and creates online tests. A learner studies learning contents in a web browser, and take an online test. A manager manages those data. Our system manages three types of data, learning contents, online test data, and users' information. Learning contents and online test data are composed of various kinds of data

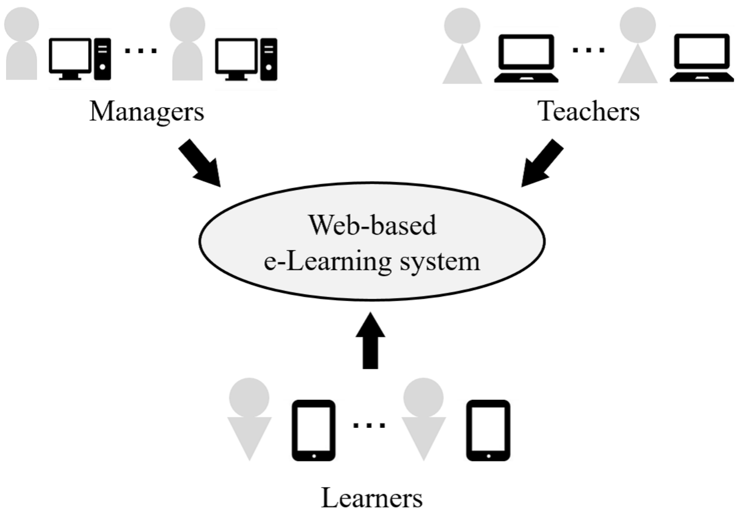


Figure 4.1: The overview of a web-based e-Learning system.

such as images and texts. Users ' information is text data such as authentication information and the learning history of a user.

In order to implement our system, first, we design a system structure that integrates a P2P model and a client-server model. Then, we discuss how our system manages data.

4.2.1 System Structure

In an e-Learning system, comfortability and continuously are important factors. In a client-server model, the loads of computation and storage concentrate on a server. It causes low response time. Moreover, the server becomes the single point of failure. As the countermeasure to these problems, preparing multiple servers to distribute computation/storage is effective, but it requires additional costs. Rich universities/companies have enough money to purchase such multiple servers, but poor universities/companies cannot pay such a money. A P2P model, therefore, has been mentioned.

When we apply a pure P2P model to an e-Learning system, the copyright issue of learning contents arises. In a pure P2P model, users ' machines have to manage learning contents. Learning contents, however, may have a copyright notice. It is, therefore, inappropriate the learning contents to be stored in users' machines. User information is also inappropriate to be store in other users ' machines Thus, users' machines have not manage any data. We, therefore, integrate a P2P model with a client-server model.

Figure 4.2 shows the structure of a proposed e-Learning system. A client in figure 2 is a user ' s machine. A user ' s machine has only a function as a client and does not join in a P2P network. Any data, therefore, is not stored in a user ' s machine. On the other hand, we prepare multiple machines for the distribution of server ' s functions. For the distribution, we make a P2P network constructed from multiple machines. By dividing server's functions to these multiple machines, we realize load balancing and high fault tolerance. Moreover, we assume inexpensive computers to be used as these multiple machines. For example, they are old computers that are left unused or being discarded due to renewals or other reasons in an organization. Since server's functions are distributed among multiple machines, we do not need to purchase an expensive machine. We, therefore, can construct a proposed e-learning system at low lost.

4.2.2 Node Management

Machines that construct a P2P network are called nodes. Nodes on a P2P network have to communicate with each other. Each node, therefore, have to know where other nodes are. In order to manage nodes, we introduce a *node management table.*

The node management table consists of identifiers, IP addresses and free storage space of each node. By getting an IP address of other node from a node management

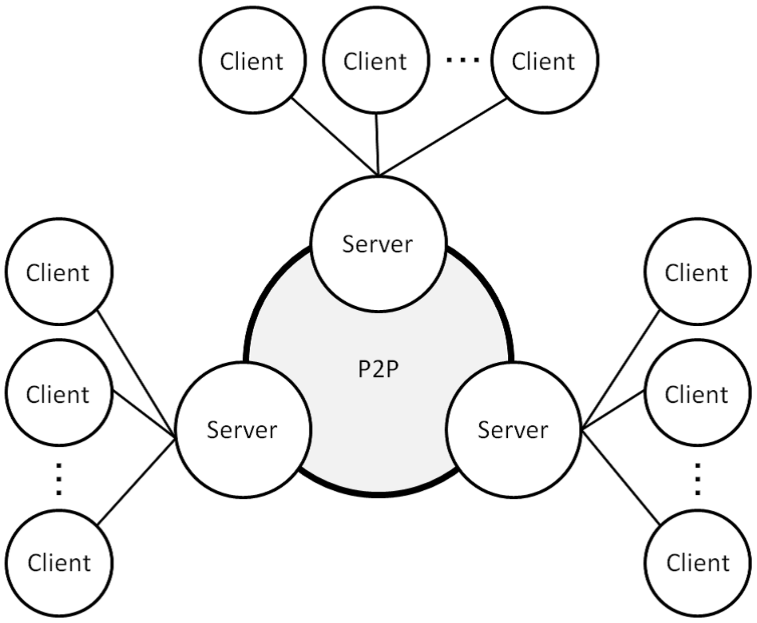


Figure 4.2: The structure of a distributed e-Learning system.

table, each node can communicate with other nodes. When storing data, sufficient free storage space is required. Thus, the node management table has free storage space of each node for the decision of a node where data is stored. By getting free storage space of each node from a node management table, we can select a node who has sufficient storage space for storing data. Each node adds own information to a node management table when joining a P2P network. Each node deletes own information from a node management table when leaving from a P2P network. In addition, each node rewrites own information in a node management table when an IP address or free storage space changes.

4.2.3 Data Management

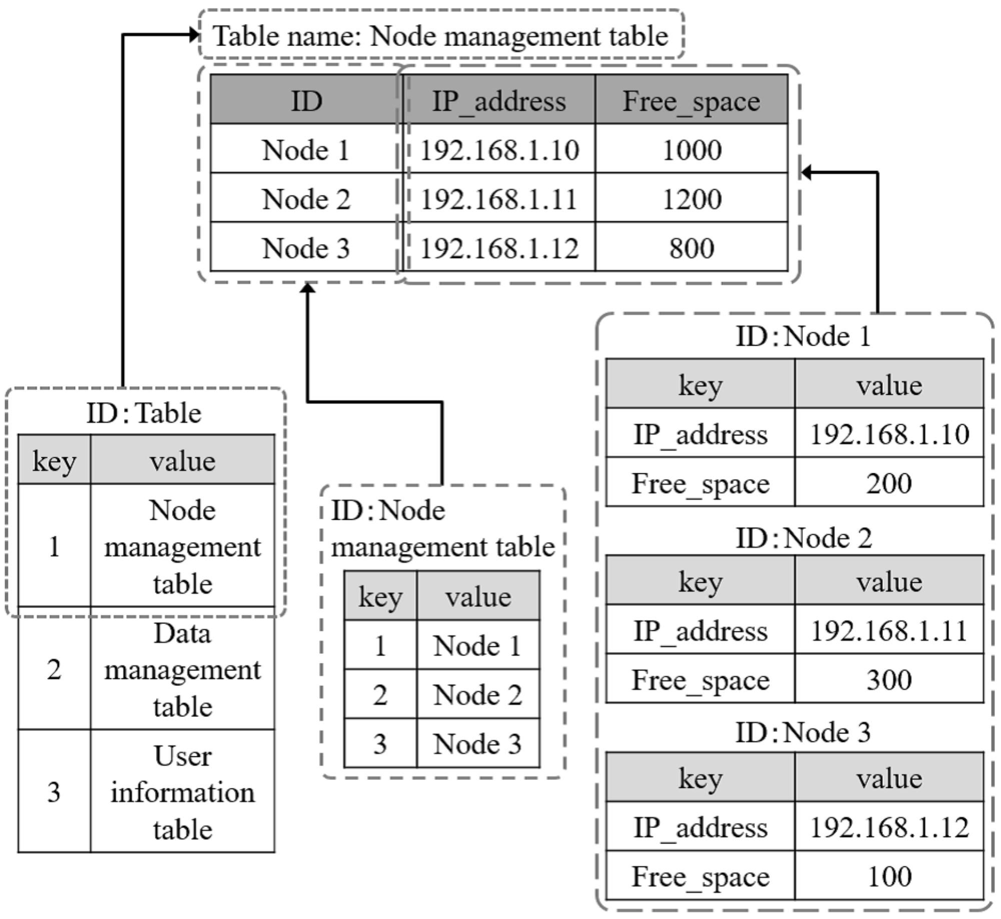
This system provides services for learners to study learning contents and take online test. The system, therefore, has to manage learning contents and online test data. In addition to these data, the system has to manage user information such as learning history, the results of online test, an ID and a password for user authentication. Each user information is written in each text file. Leaning contents and online test data are composed of various kinds of data such as images and texts. It is inefficient to manage those data independently, thus, those data of same learning contents and online test is associated with each other. Then, the system manages the associated data in a *data management table*.

The data management table consists of the path and the IP address of a node an associated data is stored. The data management table stores only a path and an IP address, not store associated data. Since the data management table indicates the location where learning contents and online test data exist, anyone can find the location of any learning contents and online test data by referring the data management table. Learners can download any learning contents and online test data to study by themselves; a teacher can update learning contents and online test data. Moreover, when the rest of storage space in a node is small, we can move data to other a node by rewriting a path and an IP address in the data management table. It is effective to balance storage space in each node.

4.2.4 Distributed Hash Table

We introduced a node management table and a data management table. Since the system manages learning contents, online test data, and user information in each node, we can overcome the problem of a single point of failure on those data. Here, we consider who will manage the node management table and the data management table and how. If a specific node manages their table, the specific node becomes a single point of failure. Thus, we design a database based on a Distributed Hash Table (DHT). A database is the mechanism for managing tables collectively. DHT is widely used as a method to a construct system based on a P2P model. DHT has high scalability. DHT is a technology that distributes the management of a Hash Table to multiple nodes. DHT is a simple data structure of a key and value pairs. DHT, therefore, cannot represent a database. To construct a database based on DHT, we prepare three DHT and combine them. This mechanism is used as a distributed database (DDB).

A table of the database consists of rows and columns. To represent rows and columns by DHT, we prepare three DHT for the management of table names, the representation of rows, columns of each tables. Figure 4.3 shows the structure of DDB. The lower part of figure 4.3 shows three DHT constructing a table shown in the upper part. DHT at the left one of the lower part is for the management of table names. We store the identifier of a table as a key and the table name as a value. The example in figure 4.3 shows there are three tables, named Node management table, Data management table and User information table. DHT located at the center of the lower part manages rows in a table. In figure 4.3, DHT named Node management table manages three rows, Node 1, Node 2, and Node 3. The rest DHT at the right of the lower part is for representation of columns. We store a column name as a key and data body as a value. By combining those three DHT, we can generate tables and realize a database.



ID	IP_address	Free_space
Node 1	192.168.1.10	1000
Node 2	192.168.1.11	1200
Node 3	192.168.1.12	800

key	value
1	Node management table
2	Data management table
3	User information table

key	value
1	Node 1
2	Node 2
3	Node 3

key	value
IP_address	192.168.1.10
Free_space	200

key	value
IP_address	192.168.1.11
Free_space	300

key	value
IP_address	192.168.1.12
Free_space	100

Figure 4.3: The structure of DDB.

4.3 A Distributed e-Learning System using Idle Resources

As mentioned above, we have proposed a distributed e-learning system integrated a P2P model with a CS model. In this distributed e-Learning system, we propose the use using-computer as nodes.

4.3.1 Design

Figure 4.4 shows the structure of a proposed distributed e-Learning system. A client in figure 4.4 is a user ’ s machine. This system is based on a CS model in which clients connect to servers and servers provide services to clients. We, however, prepare multiple machines for the distribution of server ’ s functions. By dividing server ’ s functions to these multiple machines, we realize load balancing and high fault tolerance.

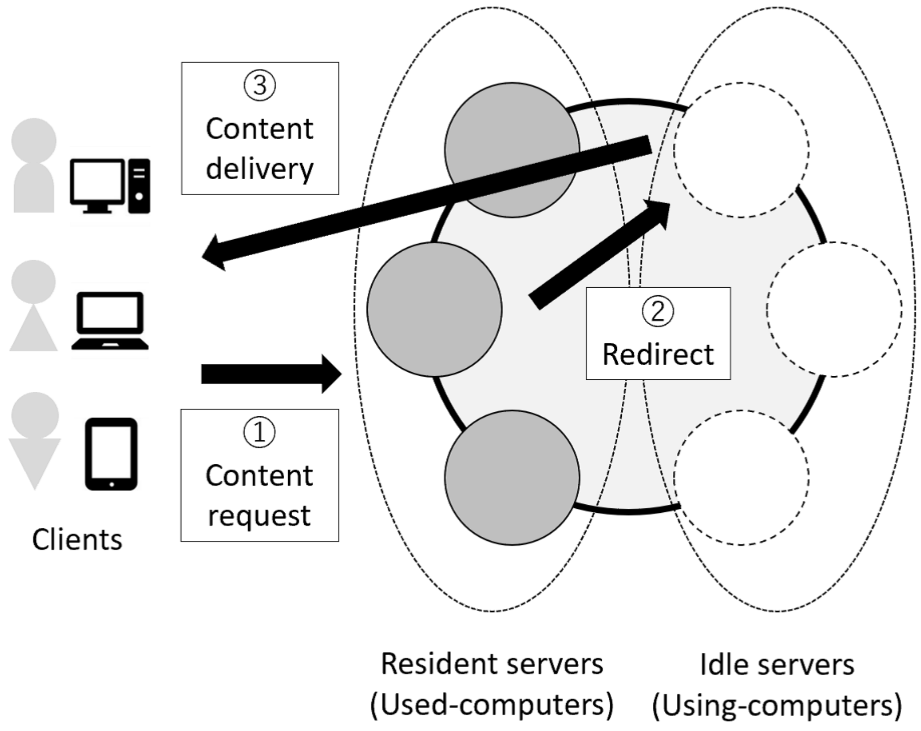

③
Content delivery
② Redirect
① Content request
Clients
Resident servers (Used-computers)
Idle servers (Using-computers)

Figure 4.4: The structure of a distributed e-Learning system using idle resources.

We construct this system with two types of servers. One is a resident server that resides exclusively for the system except in the case of maintenance or failure. We use used-computer as these servers. The other is an idle server that joins or leaves dynamically according to the load increase or decrease or a resident server fails. We use using-computer as these servers.

When a load of the system is low, it operates only on resident servers. Clients, therefore, first connect to any resident server when using the system. When accesses are concentrated on the resident server and the load increases, the idle server is merged with itself to distribute the accesses and improve the processing performance. Access distribution is achieved by placing contents on idle servers and redirecting access from clients. If a load decreases, idle servers are left. For this reason, this system has high scalability that can flexibly adjust its performance according to a load. Moreover, when a failure occurs in resident servers, this system improves fault tolerance by allowing an idle server to be used as a resident server if necessary.

4.3.2 Placement Method of Contents

If a file of learning content is stored only in a specific server, we cannot access the file when an unexpected loss of a file or server that have the file leaves. Moreover, when an access from clients concentrate on a specific learning content, a load of a server holding it increases. We, therefore, replicate a learning content and store them on multiple servers. Thus, the availability of learning contents is improved. Moreover, access to a learning content from clients is distributed among multiple servers.

Here, it is considered that learning contents have different utilization frequency. It is desirable to distribute a load by placing many replicas for learning contents with high utilization frequency. On the other hand, learning contents with low utilization frequency need only have enough replicas to guarantee minimum redundancy. We, therefore, want to keep a minimum number of replicas in normal times and change the number of replicas according to utilization frequency. Moreover, since idle servers join and leave according to increase and decrease of a load, frequency of joining and leaving is more than that of resident servers. It is, therefore, necessary to place replicas corresponding to the frequent occurrence of joining and leaving in idle servers. Thus, we define *strong-replica* and *weak-replica* as replicas of learning contents.

A Strong-Replica

A strong-replica is a replica that increases redundancy. When a learning content is newly stored, a certain number of its strong-replica is created and distributed in resident servers. Moreover, the system ensures that there is always a constant number of strong-replicas. For example, if a server with a strong-replica leaves, the number of strong-replicas decreases. In this case, a new strong-replica is created for an existing server. For this reason, even if some strong-replicas are unexpectedly inaccessible, we can use a learning content always because of the presence of other strong copies.

A Weak-Replica

On the other hand, a weak-replica is a replica for load balancing. This is newly created when utilization frequency becomes high due to access to a learning content is concentrated. It is created on an idle server. Thus, idle servers join the system when weak-replicas are created. It then distributes a load by redirecting client accesses to idle servers with weak-replicas. If it is used less frequently, a weak-replica is removed and an idle server is left from the system. By dividing a replica of a learning content into a strong and a weak-replica, we can realize its redundancy and load balancing efficiently.

Replica Management

In order to manage each information of learning contents based on a strong and a weak-replica, a file table has a file name of each learning content, an IP address of a stored server, a file path on a stored server, and a replica number of a file. A replica number is a serial number that can distinguish between a strong and a weak-replica. We simplify a process by managing strong and weak-replicas in the same way on a DDB. Moreover, we also simplify a process further by not distinguishing an original file from a strong-replica. Each server refers to a file table to provide clients with access to learning contents.

Process of Each Replica

We describe each process about a strong-replica and a weak-replica.

When a learning content is newly stored, a preset number of strong-replicas are created and stored on resident servers. Moreover, the system adds information about stored strong-replicas to a file table. When a learning content is deleted, the system refers to a file table and delete appropriate strong and weak-replicas from servers with them. Moreover, the system removes information about deleted replicas from a file table. In the case of updating learning contents, the system carries out its newly storing process after their deletion process.

When getting learning contents, the system refers to a file table and provides clients with access to servers with strong or weak-replicas. An access destination is determined at random for load balancing. The system records the number of accesses to each replica. The system keeps track of the number of accesses to each replica for a given period, and accesses the replicas in order of decreasing number of accesses for load balancing. Moreover, if more than a certain number of replicas is accessed, the system creates a weak-replica on an idle server and adds its information to a file table. Weak copies are deleted when the number of accesses falls below a certain number.

If a server to be left from the system has a strong-replica, the server creates a strong-replica to another server and rewrites its information in a file table before the server leaves. If a server has a weak-replica, the server deletes it and rewrites its information in a file table before the server leaves. In addition, servers that join the system may leave unexpectedly due to a failure. In this case, the system refers to a file table, and if a left server has a strong-replica, it creates a new strong-replica on another server and rewrites a file table. If a left server has a weak-replica, the system deletes its information from a file table.

4.3.3 Implementation

DDB implements in Apache Cassandra [37]. Cassandra are prepared for various programming languages. In this implementation, we use a Java driver [38].

Cassandra has *Keyspace* that corresponds to a database of a relational database. In Keyspace, we can create *ColumnFamily* that corresponds to a table of a relational database. In creating ColumnFamily, it is necessary to define its name, a column name and a data type, and a primary key column. Moreover, multiple ColumnFamily can be created in one Keyspace.

A server table is created by using one ColumnFamily. Table 4.1 shows the definition of a server table. Each server rewrites this table when joining or leaving a cluster. Moreover, it also refers to this table to get free storage space when storing learning contents.

A file table is created by using one ColumnFamily as well as a server table. Table 4.2 shows the definition of a file table. Each server rewrites this table when joining or leaving a cluster. Moreover, it also refers to this table to get free storage space

Table 4.1: Definition of a server table.

Row name	Explanation
id	An identifier of a server
ip	An IP address of a server
fs	Free storage space of a server

when storing learning contents. In addition, this table records a number of accesses to a file for a given period and uses it as a reference for making weak-replicas.

Table 4.2: Definition of a file table.

filename	A name of a file
location	An IP address of a server which has a file
filedir	A path where a file exists
repid	A replication number
count	A number of accesses to a file for a given period

We describe each process flows of an implemented system. Programs for each process are created by Java. Moreover, HTTP is used for client-to-server and server-to-server communication.

When a learning content is newly stored, a preset number of strong-replicas are created and stored on resident servers. The storing destination is determined in the order of an amount of free storage space by referring to a server table. Moreover, a path on a server is set uniformly in advance. After storing, the system adds this information to a file table. The system sets a replication number to an integer starting with 1 to 99. When a learning content is deleted, the system refer to a file table and deletes appropriate strong and weak-replicas from servers with them using the destination IP address and path. After deleting, the system removes information about deleted replicas from a file table. In the case of updating learning contents, the system carries out its newly storing process after its deletion process.

When a client gets learning contents, a server accessed by this client refers to a file table; and if a file is stored in its own server, this server provides the file to the client. If the file is not stored on the server, the server gets an IP address and a path of the server where the file is stored from a file table and redirected to its server. A priority of a redirected destination is in order of the least number of accesses to a file for a given period. Here, if a strong-replica of a learning content is accessed more than a number of accesses to a file for a given period, the system creates a weak-replica to an idle server. Moreover, the system adds its information to a file table and use it as an access destination. A replication number should be an integer starting with 100 to distinguish strong-replicas. In addition, we reset a number of accesses at a given period to detect a temporary load increase. The system deletes a weak-replica if its number of accesses is below a certain number.

If a server with learning contents leaves by a plan, the server processes corresponding to its replicas. In the case of a strong-replica, the server creates a new strong-replica to another server and rewrites its information in a file table. A server to be created is a resident server with the largest free storage capacity. In the case of a weak-replica, the server deletes it and rewrites its information in a file table. After processing replicas, the server leaves from the system. In the case of unexpected leaving, the system refers to a file table, and if a left server have a strong-replica, it creates a new strong copy on another server and rewrites a file table. If a left server has a weak-replica, the system deletes its information from a file table. Each server checks a server table for communication at regular intervals to detect unexpected disconnection or failure.

4.3.4 Experiment and Evaluation

We examined how the content acquisition time changes when there are only strong-replicas and when there are also weak-replicas.

As an experimental environment, we prepared 16 virtual machines (CPU: Intel Core i5-6500 (1 core), RAM: 512GB, HDD: 40GB, OS: Debian 9).

We placed 1KB of content on a single node and measured the average response completion time when simultaneous accesses to the content occurred. The number of simultaneous accesses was set to 600, and Apache JMeter [55] was used. In the case of strong-replica only, the number of replications was set to 3. In the case of weak-replica, weak-replicas were created on all nodes on the first attempt, and all weak-replicas were deleted every three attempts. The measurement results are shown in figure 4.5.

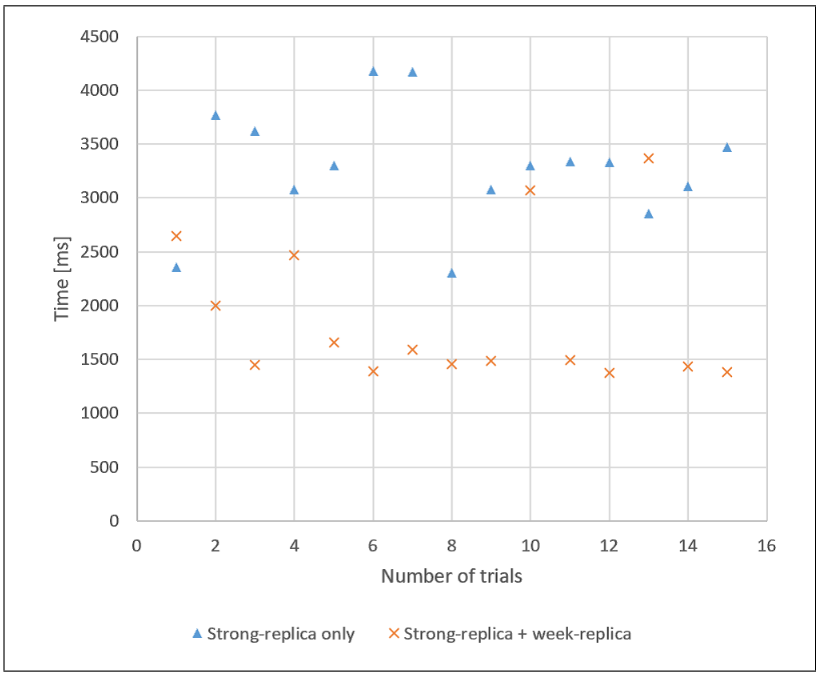


Figure 4.5: Comparison of average response time with and without weak-replica.

In the first, fourth, seventh, tenth, and thirteenth trials accessed without the weak-replica, the response time is shorter for the strong-replica only, the time is shorter for the strong-replicas only. This is because it takes time to create the weak-replicas and distribute the accesses. However, in trials where weak-replicas are present, the time is shorter when weak-replicas are present. From this, it was confirmed that processing performance could be improved by creating weak-replicas on the using computers and distributing the access.

4.4 Related Works

So far, e-Learning systems based on the P2P model have been developed [56], [57]. In these, all the nodes participating in the system communicate directly with each other in an equal relationship to improve fault tolerance and distribute the load. These can solve the problems of the CS model by distributing the processing. However, if all nodes leave the system, the system cannot continue to run.

In order to solve the problems of P2P model and CS model respectively, an e-Learning system based on hybrid P2P [58], [59] has been proposed. In [58], the bootstrap node is introduced to share preliminary configuration information with new entrants to the platform. In [59], responsiveness and availability are improved by giving different roles to nodes, such as only providing functions as a server. However, these do not mention the cost. In the proposed system, the cost can be reduced by using idle resources of using-computers and used-computers as nodes.

4.5 Conclusion

In this chapter, we have proposed to use idle resources of using-computers and used-computers for an e-Learning system. By using using-computers as nodes and having them join the system, the processing performance can be flexibly adjusted according to the load of the system. We also defined the strong-replica and the weak-replica and tested the effects. Therefore, we have achieved high fault tolerance and flexible load balancing as needed at a low cost.

Chapter 5

Content Delivery Network using Idle Resources of Each Computer

5.1 Introduction

As we enter the year 2020, the impact of COVID-19 is changing the way of life of human society [60]. There is a need to avoid situations where people are enclosed, crowded, and close together, and measures are being taken in a variety of settings. In various schools, which are educational institutions, face-to-face classes, which have been conducted as a rule, are being avoided as much as possible, and remote classes and dispersed school attendance are being implemented [61].

In the case of distance learning, the main form of learning is for students to study at home instead of commuting to school. In this case, there are more opportunities to use computers for reading materials, watching videos, and writing reports than in the past. With the implementation of distance learning, the use of computers in classes and self-study has been rapidly promoted and practiced, and this is expected to continue in the future even when the location of classes and self-study is moved to the school. The GIGA School Initiative [62] led by the Japanese Ministry of Education, Culture, Sports, Science and Technology (*MEXT*) is also working to improve education using ICT by providing one computer per student and enhancing the network, and the use of computers in class and for self-study in schools is expected to spread. It is expected that the use of computers in classes and for self-study will become more widespread.

When teaching or self-studying on a computer, content such as video images and documents may be acquired and used via a network. The type of content to be used depends on the content of the class or self-study, but if the subject matter of the class or self-study is fixed, a situation where many people use the same content is possible. In this situation, if each person tries to obtain the content, the server that holds the content and the network to the server will be burdened with data communication. In particular, when a particular content is used in real time, such as during a class, that amount of communication will occur at the same time, and the load on the server and network will increase rapidly in a short period of time, which may cause delays and failures. In addition, if the target content exists on a network outside of the school, data communication related to the acquisition of the content will occur on the shared network, and this will occupy the limited bandwidth. The amount of data traffic on shared networks is increasing due to COVID-19 [63], and educational institutions are being urged to minimize the amount of data traffic in remote classes [64]. For this reason, it is desirable to reduce the amount of external data communication from the school network to the shared network outside the school. If the amount of external data communication is reduced, the amount of internal data communication is also reduced, and in general, the amount of data communication in the shared network can be reduced.

Content Delivery Network (*CDN*) [65] is the infrastructure for delivering content over a network, and it is widely used as a method for delivering mainly large-volume content such as OS updater and videos. A basic CDN consists of a server that holds the original of the content (hereinafter referred to as the "origin server") and a server that holds a copy of the content as a cache (hereinafter referred to as the "cache server"). The duplicate of the content held by the origin server is placed in the cache server. The content is then delivered to the end user through the delivery of the duplicate from the cache server, and not directly from the origin server. By distributing multiple cache servers both geographically and network-wise, the cache server that is most suitable for the entire CDN and the end user's situation can be selected for delivery. CDN is generally used with services provided by providers such as Akamai [66] but since they are built on the cloud, they cannot be used within the school network. However, since it is built on the cloud, it cannot be used within the school network. Therefore, it is not possible to reduce the amount of data communication on the shared network line. In order to reduce the amount of data traffic, it is possible to build a CDN on the school network and exchange copies of the contents on the school network, but in this case, the cost of preparing each server becomes an issue.

Therefore, we propose to reduce the amount of external data communication by using using-computers with idle resources on the internal network to build our own CDN on-premise. The users of the CDN will be user using computers connected to the internal network where the CDN will be built, and not from external network. There is no origin server because the original content is not managed, and the CDN is composed of a management server that manages the CDN and a cache server that places and distributes copies of the content. When using content on the external network, the content is acquired through external data communication only for the first time and placed in the cache server as a duplicate. After that, the duplicate content will be delivered, eliminating the need for external data communication and reducing the amount of external data communication. In addition, the path to acquire the relevant content can be shortened. By using existing using-computers, it is possible to reduce the amount of new equipment that needs to be installed as servers, etc., and to build a unique CDN at a low cost.

In this chapter, we design an on-premise CDN that utilizes using-computers to reduce the amount of data communication in a shared network by reducing the amount of external data communication.

5.2 Design

For the proposed CDN, we show the configuration and processing flow after organizing the assumed environment and using-computers to be targeted.

5.2.1 Assumed Environment

The proposed CDN is built on the internal network, and it is assumed that there are using-computers on the network as described below. The target contents are video images and documents on the Internet. The computers used by the users to obtain and use the contents are connected to the internal network. We assume a school as the specific environment, but it can be replaced by other organizations, such as companies, as long as the environment satisfies the same conditions.

5.2.2 Target Computers

The using-computers to be used in the proposed CDN are only those that are connected to the internal network, and are classified into three types: clients, *resident servers*, and *idle servers*. Clients are computers that are used to access content.

This includes computers, tablets, and smartphones. The pure purpose of a client is to acquire content for use in classes and self-study, but it is difficult to imagine using the maximum amount of resources of a computer for this purpose alone. Therefore, it is thought that idle resources also exist in clients, and we would like to make effective use of them by preparing another use for them, just like other using-computers. In addition, prior preparation such as software installation is necessary. For this reason, it is necessary for clients to have their own personal computers. For this reason, it is necessary to be able to distinguish between the use of a computer for pure content use when the client is owned by an individual.

Resident servers are computers that are always running, and are assumed to be servers used to run information systems. Since it is always running, it can be used in principle to provide services to clients at all times, except when a failure occurs. We assume that you will be using used-computer for this server.

On the other hand, idle servers are computers are started up only when necessary, and stopped when they are no longer needed. We assume that you will be using using-computer for this server. The frequency and timing of starting and stopping varies depending on the user and purpose of its computer, and cannot be controlled. Therefore, in idle servers, it is necessary to assume that it will start and stop at different frequencies and timings. In addition, idle servers may become clients, and these situations need to be considered. There is also a possibility that resident servers will become clients as well, but this situation is not assumed here.

5.2.3 Structure of Proposed CDN

The configuration of the proposed CDN is shown in Figure 5.1. It consists of a management server, a cache server, and clients. Since there are using-computers that serve as both cache servers and clients, they are represented as belonging to both groups in the figure.

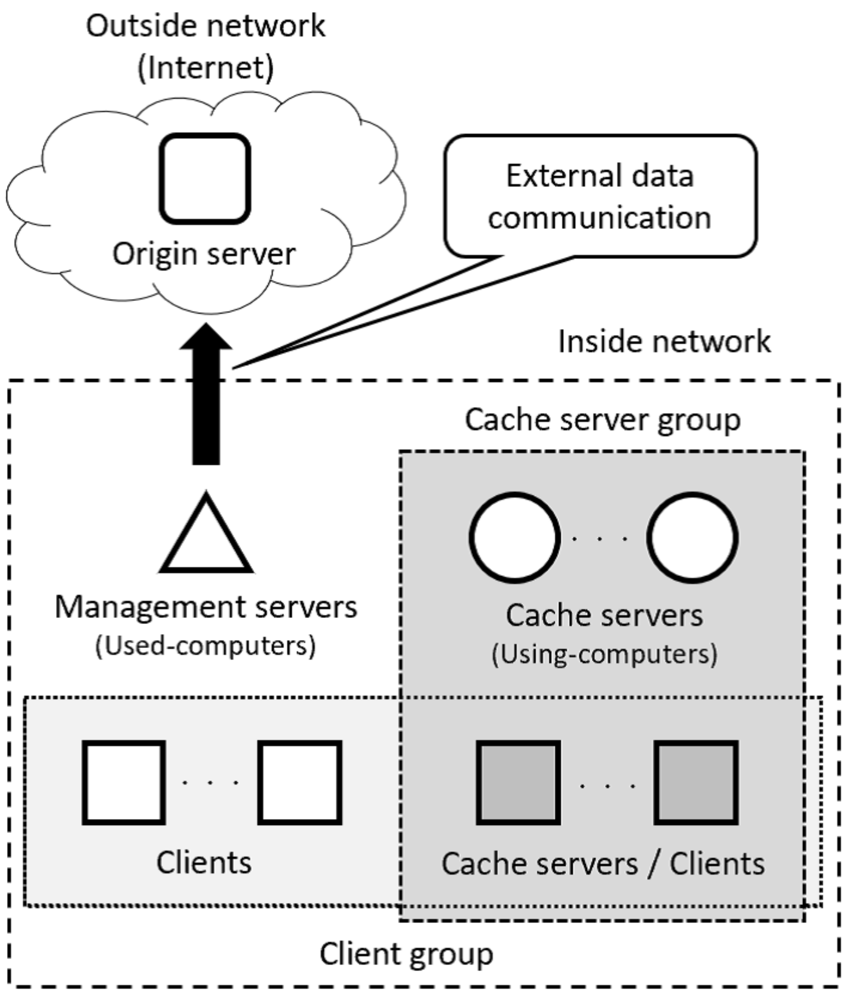


Figure 5.1: The structure of the proposed CDN.

The management server functions as a proxy server that relays communications to acquire contents from clients, and is configured so that clients always relay to the management server first when acquiring contents. In addition, it maintains a list of cache servers and their contents. The management server is responsible for the processing of the replicas. The process is split by the management server. In this way, the system is easy to install and use. However, in the configuration, if there is only one management server, it becomes a single point of failure.Therefore, it is necessary to improve load balancing and fault tolerance by preparing multiple servers.

The cache server keeps copies of the content under the direction of the management server and delivers the copies to the clients. For this reason, the cache server is mainly used as a storage to hold the replicas of the contents. The cache server uses two types of computers: idle computers and clients that may be used as cache servers. The start-up and shutdown of these computers cannot be controlled, and their availability depends on the situation at the time. For this reason Therefore, it is not possible to always use a specific business computer or client as a cache server. Therefore, instead of always using a specific business computer or client as a cache server, all business computers and clients that can be used as cache servers should be registered with the management server in advance. In addition, it is possible that there is no computer that can be used as a cache server at all. However, even in this case, the original content is maintained by the origin server, so external communication will occur, but it will not result in a situation where the client cannot use the content.

In the proposed CDN, if the target contents are located on the external network, the origin server is also located on the network outside the school. If there are a large number of clients using the content, it is possible to reduce the amount of communication with the outside world by obtaining the content from the origin server only the first time and then distributing a copy of the content. It is also possible that the contents exist in the internal network. In this case, although it does not reduce the amount of data communication in the external network, it avoids concentrating the load on the origin server that has the content and distributes the load. There is no need to change the configuration of the CDN, since the only difference between the origin server in the internal network and the external network is the URI used to obtain the content.

5.2.4 Content Placement and Delivery

The management server identifies the content that the client wants to retrieve by obtaining the URI included in the communication when the client retrieves the content. By comparing this information with the list of available cache servers and the contents held by the cache servers, the process is split.

Figure 5.2 shows the basic processing flow when there is no copy of the content to be retrieved on an available cache server. When a client attempts to acquire content, the communication is relayed by the management server (1). The management server branches off the process, retrieves the content from the origin server on behalf of the client (2 and 3), and places a copy in an available cache server (4). At the same time, it updates the list of cache servers and contents. The client then requests a redirect to the cache server that delivers the replicas (5), and the client connects to the cache server to retrieve the contents (6 and 7). If the content to be used is available on an available cache server, processes 2 through 4 are skipped, and the next process after 1 is to request a redirect (5).

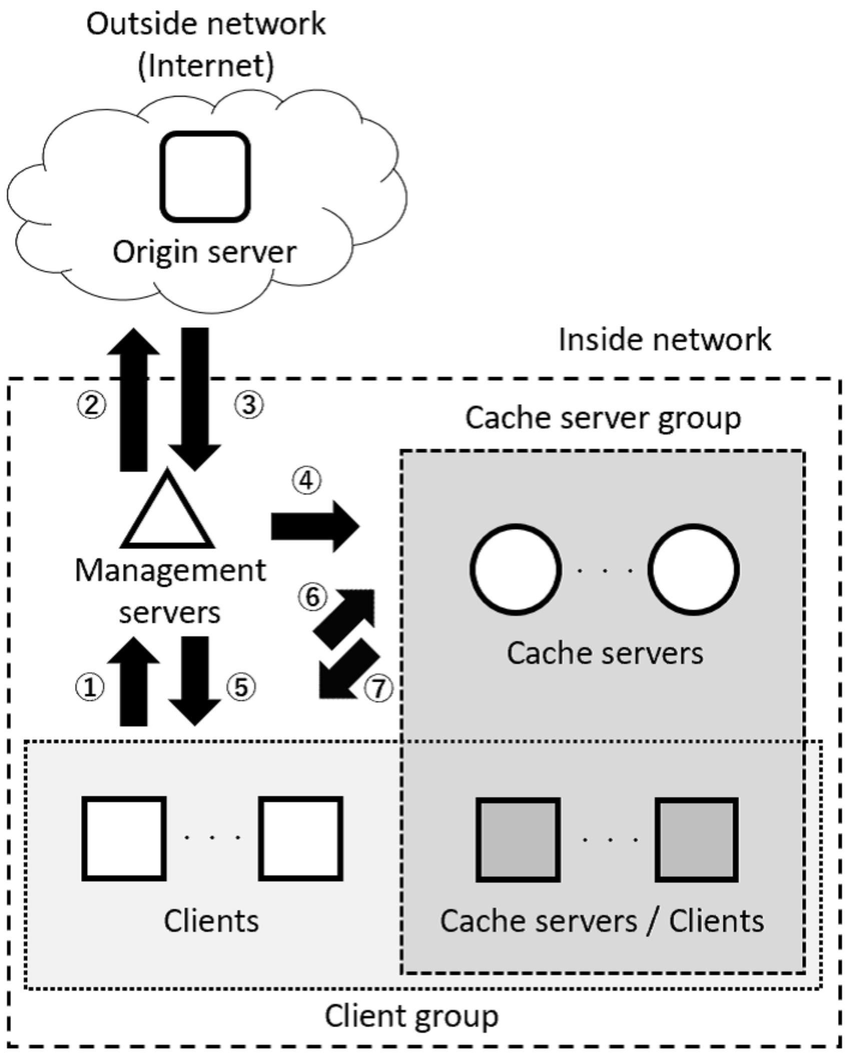


Figure 5.2: The process flow of the proposed CDN.

The replicas are placed by selecting multiple replicas from the available cache servers in the order of the amount of available storage space. The distribution source is randomly selected from the available cache servers that have the content. It should be noted that there is room for consideration in selecting the appropriate distribution destination and number of replicas, taking into account the impact on the original use of the cache server, the restrictions imposed by the original use, and the location relationship with the client.

If a large number of accesses occur simultaneously to content that does not exist on an available cache server, it is possible that the placement of the content on the cache server and its delivery from the cache server will not be completed in time to meet the client's request. If this is not possible, the priority is to reduce the amount of external data communication, and the client will wait to acquire the content until the placement in the cache server is completed. However, in the case of a large number of simultaneous accesses when it is known in advance that the content will be used, the client can avoid waiting for the content to be placed in the cache server by performing the operation to acquire the content in advance. This is the current policy, but there is room for consideration on how to shorten the latency and how to balance the amount of external data communication with the latency.

5.2.5 Cache Servers and Content Management

Information about cache servers and the content is managed by the management server. These information is used to select the location of the content and the source of distribution.

Information of cache servers is registered in advance by connecting to the management server from using-computers and clients side to be used as the cache server. The cache server manages information such as the IP address required for communication, the startup status, and the free storage space required for placing content replicas, and updates this information through regular communication between the management server and the cache server, as well as communication with the cache server when it starts and stops. Periodic communication is used to respond to failures such as unexpected stoppage of the cache server.

5.3 Related Works

In CDN, there have been many studies on algorithms for content placement and delivery, and various methods have been proposed and evaluated to reduce the amount of data transmission and achieve high-speed delivery to clients [67], [68]. In addition, there is [69] that assumes the construction of a CDN within an organization, rather than using a CDN provided by a CDN provider or ISP. This assumes that the bases that make up the organization are physically dispersed, and proposes an effective design based on the differences between CDNs operated by CDN providers and ISPs and CDNs built within the organization. However, the computers that make up the CDN are under the control of the CDN administrator. In addition, it is assumed that the users of the content are not from within the organization but from outside.

Another method for sharing content is based on Peer to Peer (P2P) technology, such as BitTorrent [70]. In contrast to basic CDN and other methods in which a server holding the content distributes it to clients who request it, this method allows the computers participating in the system to mutually request and distribute content by having equal roles and functions. There are also CDNs that use P2P technology [71]. Although P2P technology can be applied as a part of the proposed CDN, it is not used in this study for simplifying the configuration and operation.

5.4 Conclusion

We proposed and showed the design of an on-premise CDN that utilizes idle resources. By constructing our own CDN on the internal network, we can reduce the amount of external data communication compared to the case where users acquires contents on the external network. In addition, by using using-computers on the internal network as the computers that make up the CDN, an original CDN can be constructed at a low cost.

Chapter 6

Conclusion

In this thesis, we discussed building of a distributed system using idle resources of computers.

In chapter 2, we have organized characteristics of used-computers and considered a storage system based on their characteristics. Results of considering, we have assumed an inexpensive and highly available distributed storage system using numerous used-computers after replacing existing disks of used-computers with large capacity as needed to secure capacity. Moreover, in order to reduce and facilitate labors required for construction and operation, we have implemented an auto-construction system of a distributed storage environment for used-computers and experimented. Through this experiment, we have confirmed that the most of the processing of construction were successfully automated and we could reuse used-computers as distributed storage.

In chapter 3, we proposed to build an inexpensive storage system by utilizing the surplus capacity of using-computers as distributed storage. The design of the system was based on the assumption that the computers in use have their original purpose. As a partial implementation of the system, we also considered a process to maintain the number of file replicas when a node leaves the system.

In chapter 4, we have proposed to use idle resources of using-computers and used-computers for an e-Learning system. By using using-computers as nodes and having them join the system, the processing performance can be flexibly adjusted according to the load of the system. We, therefore, have been realized load balancing and high fault tolerant of the system at low cost.

In chapter 5, we considered content delivery network using idle resources of using-computers and used-computers. By using idle resources to build a CDN on-premises, data traffic to and from the outside world can be reduced, and data traffic on the shared network can be reduced inexpensively.

It is hoped that the above results will lead to the effective use of idle resources in the future.

References

[1] Seagate. RETHINK DATA: PUT MORE OF YOUR BUSINESS DATA TO WORK—FROM EDGE TO CLOUD. 2020.

[2] Jiyi Wu, Lingdi Ping, Xiaoping Ge, Ya Wang, and Jianqing Fu. Cloud Storage as the Infrastructure of Cloud Computing. *Proceedings of the 2010 International Conference on Intelligent Computing and Cognitive Informatics*, pp. 380-383, 2010.

[3] Chapter 27. Kickstart Installations Red Hat Enterprise Linux 7 Red Hat Customer Portal. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations.

[4] Preboot Execution Environment (PXE) Specification. http://www.pix.net/software/pxeboot/archive/pxespec.pdf.

[5] ISC Open Source Projects / dhcp · GitLab. https://gitlab.isc.org/isc-projects/dhcp.

[6] Tftp-server Download (DEB, RPM). https://pkgs.org/download/tftp-server.

[7] Welcome! - The Apache HTTP Server Project. https://httpd.apache.org/.

[8] The CentOS Project. https://www.centos.org/.

[9] Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, and Julian Satran. Object storage: the future building block for storage systems. *Local to Global Data Interoperability - Challenges and Technologies*, pp. 119-123, 2005.

[10] Mike Mesnier, Greg Ganger, and Erik Riede. Object-based storage. *IEEE Communications Magazine*, Vol. 41, pp. 84-90, 2003.

[11] Jiyi Wu, Lingdi Ping, Xiaoping Ge, Ya Wang, and Jianqing Fu. Cloud storage as the infrastructure of cloud computing. *Proceedings of the 2010 International Conference on Intelligent Computing and Cognitive Informatics*, pp. 380-383, 2010.

[12] Cloud Object Storage – Amazon S3 – Amazon Web Services. https://aws.amazon.com/s3/.

[13] Shin-ichi Motomura, Toshiya Kawato, and Masaya Kimoto. An evaluation of the use an object storage using an object storage gateway. *Journal for Academic Computing and Networking*, No. 17, pp. 3-9, 2013 (in Japanese).

[14] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. *Ph.D Thesis, University of California, Irvine*, 2000.

[15] Cao, Mingming, Suparna Bhattacharya, and Ted Ts'o. Ext4: The Next Generation of Ext2/3 Filesystem. *LSF*, 2007.

[16] GitHub - s3ql_s3ql a full featured file system for online data storage. https://github.com/s3ql/s3ql.

[17] Welcome to Swift's documentation!. https://docs.openstack.org/swift/latest/.

[18] Open Source Cloud Computing Infrastructure - OpenStack. https://www.openstack.org/.

[19] Ansible is Simple IT Automation. https://www.ansible.com/.

[20] Chef Software DevOps Automation Solutions Chef. https://www.chef.io/.

[21] Powerful infrastructure automation and delivery Puppet. https://puppet.com/.

[22] The Official YAML Web Site. https://yaml.org/.

[23] HPE ProLiant Servers: intelligent hybrid cloud infrastructure foundation — HPE. https://www.hpe.com/us/en/servers/proliant-servers.html.

[24] Shin-ichi Motomura, Toshiya Kawato, and Masaya Kimoto. Usecase of object storage for education and research computer systems. *Journal for Academic Computing and Networking*, No. 19, pp. 26-34, 2015 (in Japanese).

[25] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. *19th ACM Symposium on Operating Systems Principles*, 2003.

[26] Ceph.io — Home. https://ceph.com.

[27] Gluster.org. https://www.gluster.org/.

[28] Toshikazu Ichikawa, Machiko Toyoda, and Katsumi Takahashi. Design and Implementation of Highly Reliable and Highly Available Block-level Storage Using PC Cluster. *IPSJ Journal*, Vol.49, No. 6, pp. 2106-2117, 2008.

[29] Adam L. Beberg and Vijay S. Pande. Storage@home: Petascale Distributed Storage. *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, 2007.

[30] Eiko Maruko Siniawer. Affluence of the Heart': Wastefulness and the Search for Meaning in Millennial Japan. *The Journal of Asian Studies*, Vol. 73, No. 1, pp. 165-186, 2014.

[31] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 161-172, 2001.

[32] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *ACM SIGCOMM Computer Communication Review*, Vol. 31, No. 4, pp. 149-160, 2001.

[33] Himanshu Joshi and G. R. Bamnote. Distributed Database: A Survey. *International Journal of Computer Science and Applications*, Vol. 6, No. 2, pp. 289-292, 2013.

[34] Adi Shamir. How to Share a Secret. *Communications of the ACM*, Vol. 22, No. 11, pp. 612-613, 1979.

[35] How Hard Disk S.M.A.R.T. Works?. https://www.hdsentinel.com/smart/index.php.

[36] Maven – Welcome to Apache Maven. https://maven.apache.org/.

[37] Apache Cassandra - The Apache Software Foundation!. http://cassandra.apache.org/.

[38] DataStax Java Driver for Apache Cassandra. https://docs.datastax.com/en/developer/java-driver/index.html.

[39] Muhammad Nouman Durrani and Jawwad A.Shamsi. Volunteer computing: requirements, challenges, and solutions. *Journal of Network and Computer Applications*, Vol. 39, pp. 369-380, 2014.

[40] BOINC. https://boinc.berkeley.edu/.

[41] Storj - Decentralized Cloud Storage. https://www.storj.io/.

[42] Sia. https://sia.tech/.

[43] Shingo Shimano, Atsushi Nunome, Yuta Yokoi, Kiyoshi Shibayama, and Hiroaki Hirata. A Dynamic Configuration Scheme of Storage Tiers for an Autonomous Distributed Storage System. *Information Engineering Express*, Vol. 3, No. 4, pp. 91-104, 2017.

[44] Shin Tezuka, Hiroko Tomori, Ryuya Uda, and Ken-ichi Okada. Distributed Secure Virtual File System Using Hysteresis Signatures. *Proceedings of the 23rd International Conference on Advanced Information Networking and Applications*, pp. 90-97, 2009.

[45] Jun Nishimura, Hayato Ishibashi, Kota Abe, and Toshio Matsuura. A Distributed Backup System Utilizing Unused Disk Area of Multiple Network Nodes. *Proceedings of the 67th National Convention of IPSJ*, pp.711-712, 2005 (in Japanese).

[46] Hiroko Tomori, Shin Tezuka, Ryuya Uda, Masahito Ito, Satoshi Ichimura, Kazuya Tago, and Toru Hoshi. An Implementation and Evaluation of a Distributed File Backup System for Digital Forensics. *IPSJ Journal*, Vol. 50, No. 6, pp. 1561-1574, 2009 (in Japanese).

[47] Iifan Tyou, Toru Ogata, and Yukio Koike. A Study of Decentralized Storage Service with Secret Sharing Scheme. *Proceedings of the Multimedia, Distributed, Cooperative, and Mobile Symposium*, Vol. 2016, pp. 359-365, 2016.

[48] The Open University of Japan Foundation. Research on promotion of ICT utilization education. 2011 (in Japanese).

[49] JMA Management Center Inc.. Implementation status survey on e-learning for 360 Japanese companies. 2016 (in Japanese).

[50] Yano Research Institute Ltd.. e-Learning Business Performance Analysis 2021. 2021 (in Japanese).

[51] Moodle - Open-source learning platform — Moodle.org. https://moodle.org.

[52] Beverly Yang and Hector Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. *Proceedings of the 27th International Conference on Very Large Data Bases*. 2001.

[53] Margaret Driscoll. Web-Based Training: Creating e-Learning Experiences. *John Wiley & Sons*, 2002.

[54] Anil K Aggarwal. Web-based education: learning from experience. *IGI*, 2003.

[55] Apache JMeter - Apache JMeter$^{\text{TM}}$. https://jmeter.apache.org/.

[56] Shinichi Motomura, Ryosuke Nakatani, Takao Kawamura, and Kazunori Sugahara. Distributed e-Learning System Using P2P Technology. *Proceedings of the 2nd International Conference on Web Information Systems and Technologies.* pp. 250-255, 2006.

[57] Masayuki Higashino, Tadafumi Hayakawa, Kenichi Takahashi, Takao Kawamura, and Kazunori Sugahara. Management of Streaming Multimedia Content using Mobile Agent Technology on Pure P2P-based Distributed e-Learning System. *International Journal of Grid and Utility Computing*, Vol. 5, No. 3, pp. 198-204, 2014.

[58] H.A.C. Priyankara, K.A.K.D.D.B. Jayasuriya, and U.D.N.D. Gunasekara. Vyapthi: A Leveraged P2P Content Sharing Platform for Distributed e-Learning Systems. *Proceedings of the 18th International Conference on Advances in ICT for Emerging Regions.* pp. 126-132, 2018.

[59] Kazunari Meguro, Shinichi Motomura, Takao Kawamura, and Kazunori Sugahara. Distributed e-Learning System with Client-Server and P2P Hybrid Architecture. *International Journal of Computer and Information Engineering.* Vol. 3, No. 7, pp. 1731-1735, 2009.

[60] Chanjuan Sun and Zhiqiang Zhai (John). The efficacy of social distance and ventilation effectiveness in preventing COVID-19 transmission. *Sustainable Cities and Society*, Vol. 62, No. 102390, 2020.

[61] Shazia Rashid and Sunishtha Singh Yadav. Impact of Covid-19 pandemic on higher education and research. *Indian Journal of Human Development*, Vol. 14, No. 2, pp. 340–343, 2020.

[62] Ministry of Education, Culture, Sports, Science and Technology. Realization of the GIGA school concept. https://www.mext.go.jp/a_menu/other/index_00001.htm. (in Japanese).

[63] National Institute of Informatics. Trends related to Internet traffic. https://www.nii.ac.jp/news/upload/20200417-4_Mic.pdf. (in Japanese).

[64] National Institute of Informatics. Request for cooperation in data diet: To teachers who teach remote classes. https://www.nii.ac.jp/event/other/decs/tips.html. (in Japanese).

[65] Vijay K. Adhikari, Yang Guo, Fang Hao, Volker Hilt, Zhi-Li Zhang, Matteo Varvello, and Moritz Steiner. Measurement Study of Netflix, Hulu, and a Tale of Three CDNs. *IEEE/ACM Transactions on Networking*, Vol. 23, No. 6, pp. 1984–1997, 2015.

[66] Akamai CDN - Content Delivery Network Solutions — Akamai https://www.akamai.com/solutions/content-delivery-network.

[67] Takuma Nakajima, Masato Yoshimi, Celimuge Wu, and Tsutomu Yoshinaga. Color-based Cooperative Cache and its Routing Scheme for Telco-CDNs. *IEICE Transactions on Information and Systems*, Vol. E100-D, No. 12, pp. 2847–2856, 2017.

[68] M. Zubair Shafiq, Amir R. Khakpour, and Alex X. Liu. Characterizing caching workload of a large commercial Content Delivery Network. *Proceedings of the 35th Annual IEEE International Conference on Computer Communications*, pp. 1-9, 2016.

[69] Riccardo Lancellotti, Claudia Canali, and Andrea Corbelli. On Private CDNs with Off-Sourced Network Infrastructures: a Model and a Case Study. *Journal of Communications Software and Systems*, Vol. 14, No. 4, pp. 376-185, 2018.

[70] BitTorrent.org. https://www.bittorrent.org/.

[71] Satoshi Nishimura, So Tanaka, and Yosuke Endo. Development of a Hybrid P2P-CDN Live Streaming System. *Proceedings of the Institute of Image Information and Television Engineers nter Annual Convention 2013*, p. 12-9, 2013.

Publications

Journals

1. Toshiya Kawato, Shinichi Motomura, Masayuki Higashino, and Takao Kawamura. Auto-Construction for Distributed Storage System reusing Used Personal Computers. *Journal of Computers*, 13(10):1156-1163, 2018.

International Conferences and Workshops

1. Toshiya Kawato, Masayuki Higashino, Kenichi Takahashi, and Takao Kawamura. Attempt to Reuse Used-PCs as Distributed Storage. *Proceedings of the 20th International Conference on Computer Networks and Communications (International Journal of Computer and Information Engineering)*, 12(2):107-110, 2018.

2. Toshiya Kawato, Masayuki Higashino, Kenichi Takahashi, and Takao Kawamura. Proposal of e-Learning System integrated P2P Model with Client-Server Model. *Proceedings of the 18th International Conference on Electronics, Information, and Communication*, pp.279-284, 2019.

3. Toshiya Kawato, Masayuki Higashino, Kenichi Takahashi, and Takao Kawa-

mura. Attempt to Utilize Surplus Storage Capacity as Distributed Storage. *Proceedings of the 3rd International Conference on Information and Computer Technologies*, pp.351-355, 2020.

4. Toshiya Kawato, Masayuki Higashino, Kenichi Takahashi, and Takao Kawamura. Proposal of Distributed e-Learning System Using Idle Resources. *Proceedings of the 5th International Conference on Computer and Communication Systems*, pp.557-561, 2020.

Domestic Conferences and Workshops

1. Toshiya Kawato, Shinichi Motomura, Masayuki Higashino, and Takao Kawamura. Distributed Storage System reusing Used-PCs. *Proceedings of the 80th National Convention of IPSJ*, pp.23-24, 2018 (in Japanese).

2. Toshiya Kawato, Masayuki Higashino, Kenichi Takahashi, and Takao Kawamura. Attempt to Use Surplus Storage Capacity of Computers as Distributed Storage. *IPSJ SIG Technical Reports*, 2019-DPS-180(19):1-6, 2019 (in Japanese).

3. Toshiya Kawato, Shinichi Motomura, Masayuki Higashino, and Takao Kawamura. Proposal for Reduction of the Amount of External Data Traffic by On-Premise CDN using Idle Resources. *IPSJ SIG Technical Reports*, 2020-DPS-184(17):1-6, 2020 (in Japanese).

4. Wataru Shintani, Masayuki Higashino, Toshiya Kawato, Kenichi Takahashi, and Takao Kawamura. A Study of the Method of Keeping the Number of Files' Replicas when a Node Leaving on a Distributed e-Learning System. *Proceedings of the 71th Chugoku-Section Joint Convention of the Institutes of Electrical and Information Engineers*, 2020 (in Japanese).