International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:12, No:2, 2018

Attempt to Reuse Used-PCs as Distributed Storage

Toshiya Kawato, Shin-ichi Motomura, Masayuki Higashino, Takao Kawamura

***Abstract*—**Storage for storing data is indispensable. If a storage capacity becomes insufficient, we can increase its capacity by adding new disks. It is, however, difficult to add a new disk when a budget is not enough. On the other hand, there are many unused idle resources such as used personal computers despite those use value. In order to solve those problems, used personal computers can be reused as storage. In this paper, we attempt to reuse used-PCs as a distributed storage. First, we list up the characteristics of used-PCs and design a storage system that utilizes its characteristics. Next, we experimentally implement an auto-construction system that automatically constructs a distributed storage environment in used-PCs.

***Keywords*—**Distributed storage, used personal computer, idle resource, auto construction.

I. Introduction

RECENTLY, the amount of data handled by an information system continues to increase, and storage for saving data is indispensable. 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. 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, it is possible increase capacity by adding 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 personal computers, *used-PCs*, which are left unused or being discarded due to renewals or other reasons.

In order to solve those problems, used-PCs can be reused as storage. Since used-PCs are existing devices, there is no cost in reusing themselves. We can, therefore, inexpensively introduce used-PCs to reuse compared with introducing new storage. Moreover, in an environment such as a university where the total number of PCs is large and PCs are frequently replaced, there are many used-PCs. These many unused-PCs can be collected to be reused. It is widely common to disassemble and recycle used-PCs. There are few attempts to reuse used-PCs for another use application which is different from one when they were installed. It would be effective utilization of resources more and more to reuse used-PCs for different purpose while they can still be available for generic PC usage.

Toshiya Kawato, Shin-ichi Motomura, Masayuki Higashino and Takao Kawamura are with the Tottori University, Tottori, 680-8550, Japan (e-mail: t.kawato@tottori-u.ac.jp, motomura@tottori-u.ac.jp, higashino@tottori-u.ac.jp, kawamura@eecs.tottori-u.ac.jp).

In this paper, we organize the characteristics of used-PCs and design a storage system based on its characteristics. Moreover, we experimentally implement an *auto-construction system* that automatically constructs a *distributed storage environment* in used-PCs which makes the used-PC join the distributed storage as a part of the storage.

II. Related Works

It has already been attempted to construct a storage system with a cluster configuration of personal computers. Google File System [1] is a distributed file system on the premise that a large scale storage system is constructed using servers based on PCs. Google File System is designed to allow built-in PCs to fail, and focuses on automatic recovery without losing data even in failure. In addition to Google File System, there are distributed file systems such as Ceph [2] and GlusterFS [3]. It is possible to construct a storage system using PCs by using these systems.

These existing methods are, however, based on the premise that servers have been dedicated for a storage system since they are installed. They are not supposed to reuse *used-PCs*, which are used for a different usage other than storage, and research that reuses used-PCs as storage has not been conducted. In order to use used-PCs as storage, it is necessary to consider characteristics of used-PCs. 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-PCs.

III. Characteristics of Used-PCs

Before reusing used-PCs 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-PCs. Moreover, we discuss a storage system considering its characteristics.

A. Small Capacity

Although recent PCs with TB class disks are common, capacity of a used-PC that can be collected at present may be about several hundred GB. For the reason, in order to have large capacity such as TB class, it is necessary to secure numerous used-PCs. 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-PC 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 PCs are inexpensive compared with disks for storage products and servers, and a 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.

B. Low Performance

Compared with storage products and servers, PCs are inferior in performance of CPU and NIC. Since their disks are also supposed to be used in general PCs, 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.

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

C. Different Lifetime

Depending on how hard a PC is used until the PC 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 PC 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-PCs, e.g., estimation of a lifetime, automatic detachment on failure and automatic incorporation of a spare machine.

D. High Diversity

Used-PCs have models of various performance and configurations. Since various manufacturers sell various models, PCs are more diverse than storage products and servers. It would then take a lot of works to configure a used-PC as storage.

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

E. Available Numerous Units

In an environment such as a university where the total number of PCs is large and replace is frequent, we can collect numerous used-PCs. Moreover, since there is no cost to reuse used-PCs themselves, it is possible to use numerous used-PCs as long as there is a place for installation of used-PCs. 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-PCs compared with rack-mounted servers. It is, therefore, possible to physically distribute used-PCs and place them easily. Moreover, it is possible to flexibly install and use used-PCs 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-PCs are, hence, suitable since numerous used-PCs are available. In addition, it is desirable that addition and deletion of used-PCs in an operating system are easy, and distributed storage can flexibly deal with them.

We listed 5 points as characteristics of used-PCs. As a storage system that takes into consideration the characteristics of used-PCs, we, therefore, assume an inexpensive and highly available distributed storage system that uses numerous used-PCs after securing capacity by replacing an existing disk of used-PCs with a large-capacity disk as needed. In order to realize high availability, we distribute potential parts of failure, and we redundantly distribute data over numerous used-PCs. 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-PCs, for example, data with low usage or update frequency. A storage system reusing used-PCs cannot meet all requirements in various situations. It is, therefore, necessary to clearly separate roles and to use a storage system reusing used-PCs 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.

IV. Experimentally Implementation of Auto-Construction System

In using used-PCs as storage, numerous used-PCs are used from their characteristics. Moreover, many works to add used-PCs to an existing distributed storage system are necessary. Such as replacement of used-PCs when used-PCs in failure and adding newly collected used-PCs. It is, therefore, extremely inefficient to perform manually necessary works for each used-PC.

In this system, by automating additions required for using used-PCs as distributed storage, it reduces works required for adding. In order to facilitate construction and management, this system executes each process for the used-PCs 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-PCs to a network such as LAN cable connection.

In this system, there are two processes required for used-PCs. One is processing for using used-PCs as nodes on a network. Used-PCs 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-PCs 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-PCs. Open source software is also used in distributed storage software as well as OS. Used-PCs can, therefore, be used distributed storage.

A. Automatic Installation of OS

We used Kickstart [4] and PXE boot [5] to install OS on used-PCs. 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-PCs in order to execute PXE boot. For administrative purposes, it is desirable to be able to identify each used-PC. We, therefore, created a script to automatically assign fixed IP addresses in this paper. 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-PC 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. Fig. 1 shows a flow of an automatic installation of OS. The DHCP server assigns a fixed IP address to a used-PC by the script when a used-PC sends a PXE boot request. Moreover, OS is automatically installed by downloading a boot image from the TFTP server, and an OS image and Kickstart related files from the HTTP server. It is, therefore, possible to automatically complete an installation of OS, initial setting and network setting only by connecting used-PCs to the network and executing PXE boot.

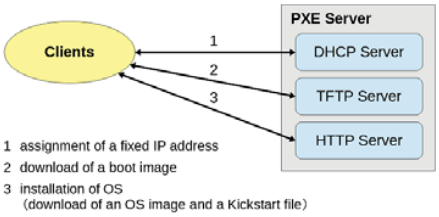

Fig. 1 Installation of OS

B. Automatic Installation of Distributed Storage Software

We focused on object storage [6], [7] as a type of distributed storage. Object storage manages data in units of an object. Object storage can be implemented using open source software as well as products, and they are used in educational and research information systems at a university [8].

Object storage uses an HTTP protocol conforming to Representational State Transfer (REST) [9] 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. 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 [10] 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* [11] was used as software for constructing object storage. Fig. 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 *Storage 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 *Proxy Node*.

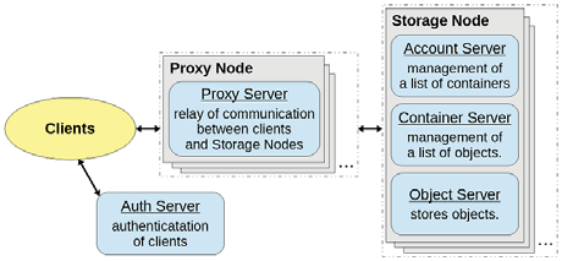

Fig. 2 Basic configuration of Swift

We used Chef [12] for an automatic installation of Swift. Chef can construct an environment for software to operate by writing codes like programming. Moreover, Chef has idempotency that is the property that the same code always produces the same result. Basic usage is to create and execute a file called a *Recipe* that describes a configuration of an environment that you want to construct in a folder called a *Cookbook*. First, we constructed a Chef server on the PXE server. The Chef server is a server on which Chef itself and knife-solo are installed. Knife-solo here is a tool that can executes Recipes for Chef on used-PCs. Next, we created a Cookbook for Swift, and created Recipes and related files that describe configurations necessary for installing Swift. Object storage environment by Swift, therefore, constructs automatically on used-PCs. Furthermore, since there is no need to create a proxy node for each used-PC, and a proxy node was created manually and only storage nodes constructed in this paper.

V. Experiment of Reuseing Used-PCs as Distributed Storage

We actually constructed a distributed storage environment for used-PCs by the auto-construction system. Table I shows the number and specification of used-PCs here.

TABLE I
Utilized Used-PCs

Type x Number	NIC	Disk x Number
SFF* Desktop x 9	1Gbps	HDD160GB x 1
Laptop x 1	100Mbps	HDD320GB x 1
Laptop x 1	1Gbps	HDD640GB x 1
Desktop x 1	1Gbps	HDD160GB x 2

*Small Form Factor.

First, each used-PC connected to the same network and executed PXE boot. In many cases, it was possible to execute PXE boot by pressing a specific key at startup. There were, however, cases where standard setting could not execute PXE boot, and we had to change BIOS or UEFI setting depending on a configuration of used-PCs. After executing PXE boot, an installation and setting of OS by PXE server were automatically completed, and an installation and setting of Swift by Chef server were automatically completed. Total capacity was about 2,700 GB by using 12 used-PCs. Moreover, actual available capacity for a data storage was about 900 GB because we set the number of zones of Swift to three in this paper. 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 utilize used-PCs as distributed storage.

VI. Discussion

In the experiment, there was difference in the time required until a processing completed depending on performance of a NIC of used-PCs. In particular, required time for a laptop PC equipped with a 100 Mbps NIC was too long. Since a construction process was automated, it did not matter if it took time or not. Heavy traffic, however, is generated when used-PCs are used as storage. At least 1 Gbps is required for a NIC of used-PCs to use as storage in order to avoid a bottleneck.

In addition, disk replacement is required in case of increasing capacity. Since a 2.5 inch disk is expensive compared to a 3.5 inch disk, desktop PCs, which are capable of mounting 3.5 inch disks, are suitable for realizing large capacity inexpensively.

VII. Concluding Remarks

In this paper, we have organized characteristics of used-PCs and considered a storage system based on their characteristics. Moreover, we have experimentally implemented an auto-construction system of a distributed storage environment for used-PCs. In addition, We have confirmed by experiment that used-PCs can utilize as distributed storage.

In future works, an implementation automatically handles failure such as disconnecting used-PCs that experience failure. We, therefore, automate a series of tasks from an installation to failure handling which are necessary for operation and reduce further labors. Moreover, there is practical operation of a distributed storage system reusing used-PCs. We will, therefore, evaluate performance such as speed and a failure rate, and investigate concretely what kind of data is suitable for reading and writing. In addition, we will calculate the advantage in cost-effectiveness by verifying a running cost such as power consumption. By implementing these, we will show effectiveness of reusing used-PCs as storage.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 16K00477.

References

[1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System,” 19th ACM Symposium on Operating Systems Principles, 2003.
[2] “Ceph Homepage - Ceph,” from http://ceph.com/, Retrieved November 10, 2017.
[3] “Gluster,” from https://www.gluster.org/, Retrieved November 10, 2017.
[4] “CHAPTER 26. KICKSTART INSTALLATIONS,” from https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations, Retrieved November 10, 2017.
[5] “Preboot Execution Environment (PXE) Specification,” from http://www.pix.net/software/pxeboot/archive/pxespec.pdf, Retrieved November 10, 2017.
[6] 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.
[7] Mike Mesnier, Greg Ganger, and Erik Riede, “Object-based storage,” IEEE Communications Magazine, Vol. 41, pp.84-90, 2005.
[8] 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 (published in japanese).
[9] Roy Thomas Fielding,“Architectural Styles and the Design of Network-based Software Architectures,” Ph.D Thesis, University of California, Irvine, 2000.
[10] “nikratio / S3QL - Bitbucket,” from https://bitbucket.org/nikratio/s3ql/, Retrieved November 10, 2017.
[11] “Welcome to Swift’s documentation!,” from https://docs.openstack.org/swift/latest/, Retrieved November 10, 2017.
[12] “Chef - Automate Your Infrastructure,” from http://ceph.com/, Retrieved November 10, 2017.