R&D Efforts in Storage Virtualization Technologies

Yoshifumi Fukumoto, Ichibee Naito, Toshio Hitaka, and Masahiro Shiraishi

Abstract

The NTT Software Innovation Center is active in the research and development (R&D) of storage virtualization technologies. This article introduces its R&D of Sheepdog, a distributed block storage system that can be used from any file system, and OpenStack Swift, a robust distributed object storage system featuring high operability.

Keywords: storage virtualization, distributed system, OSS

1. Introduction

Virtualization is finding widespread use as a technology to achieve flexibility and cost reductions in managing computer resources in a cloud infrastructure. Furthermore, *storage virtualization* technology can make multiple units of storage equipment appear as a single unit and make a single unit of storage appear as if it contains multiple units. It is an important technology that makes it easy to manage virtual machine images and to share application data.

This article introduces Sheepdog and OpenStack Swift, open-source storage virtualization technologies now under development at the NTT Software Innovation Center (SIC). The Sheepdog distributed block storage system is a type of storage that can be used as hard disk drives on personal computers (PCs) or servers via file systems. The OpenStack Swift distributed object storage system, meanwhile, is a type of storage that can read and write files using a REST (representational state transfer) API (application programming interface) and that enables large quantities of data to be stored and shared among applications.

2. Sheepdog distributed block storage system

PCs and servers have become a major part of our daily life. Files in a PC are read from and written to a hard disk via a file system. Here, the hard disk is treated as a type of block device that has the sole function of reading and writing data in block units of fixed size, and the file system has the task of writing and reading files by managing the locations of file data on the block device. Although there are various types of file systems, they all share this basic type of operation with respect to the block device. Block storage, meanwhile, is a type of storage that can provide block devices. It has the basic role of reading/writing and saving data and is considered to be the most versatile storage method.

Virtual block devices are essential in the operation of virtual machines. Therefore, in the construction of a virtual environment, the mainstream approach is to introduce shared storage appliances that can provide virtual block devices (virtual disks) of any size via the network. Shared storage can enhance the operability and reliability of a virtual environment through such virtualization functions as thin provisioning, storage snapshots, and live migration.

Sheepdog is open source software (OSS) that combines multiple commodity servers into a cluster (**Fig. 1**) in order to construct block storage that can be used in the same way as shared storage appliances. It can bundle the internal disks of the servers belonging to the cluster into a single storage pool, and it can provide virtual disks to users from the pool. Sheepdog

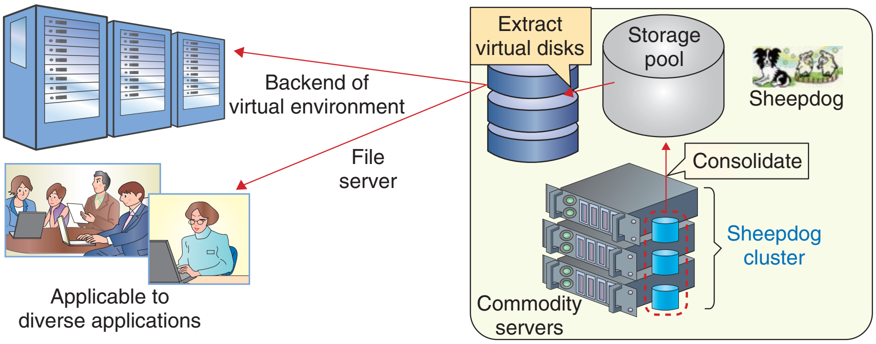


Fig. 1.   Sheepdog overview.

can be used in virtualization infrastructure software such as OpenStack and QEMU/KVM (Quick EMUlator/Kernel-based Virtual Machine), and it supports the iSCSI (Internet Small Computer System Interface) general storage interface.

Various issues arise when using ordinary shared storage. These include scalability issues (prior design is needed for extending capacity and performance, degeneration is not possible in principle, and vendor lock-in can occur) and reliability issues (service interruptions, no access to some data because of hardware failures). Sheepdog has been designed to address these issues as a fully symmetric architecture in which the servers making up a cluster all have the same role. This gives Sheepdog three key features: (1) easy addition/removal of cluster servers, enabling flexible capacity scaling and load distribution in accordance with system scale beyond the capabilities of shared storage; (2) high reliability due to no single point of failure and the capability to avoid service interruptions and data loss even if some servers should fail; and (3) high manageability due to the automating of data rebalancing, redundancy restoration, and other processes when adding/removing servers, thereby reducing the number of necessary manual operations.

A virtual disk provided by Sheepdog is divided and multiplexed into objects of fixed size (initial size: 4 MB) that are then distributed among the servers making up the cluster, as shown in **Fig. 2(a)**. The consistent hashing algorithm that is used for determining where exactly to place these objects is depicted in **Fig. 2(b)**. In Sheepdog, a data structure called a *virtual node* is generated with respect to each server (physical node), and these virtual nodes are arranged along a ring in random order. In the process of writing data to a virtual disk, an object is generated or updated with respect to three physical nodes as the destination locations of that object. Specifically, based on the virtual node determined by the object ID, a second and third virtual node along the ring are selected, and the physical nodes corresponding to those virtual nodes are deemed to be that object's destination locations. In this way, Sheepdog can mathematically determine by consistent hashing where to place the data object. This enhances autonomy by eliminating the need for a centralized management server, thereby contributing to features (1) to (3) above.

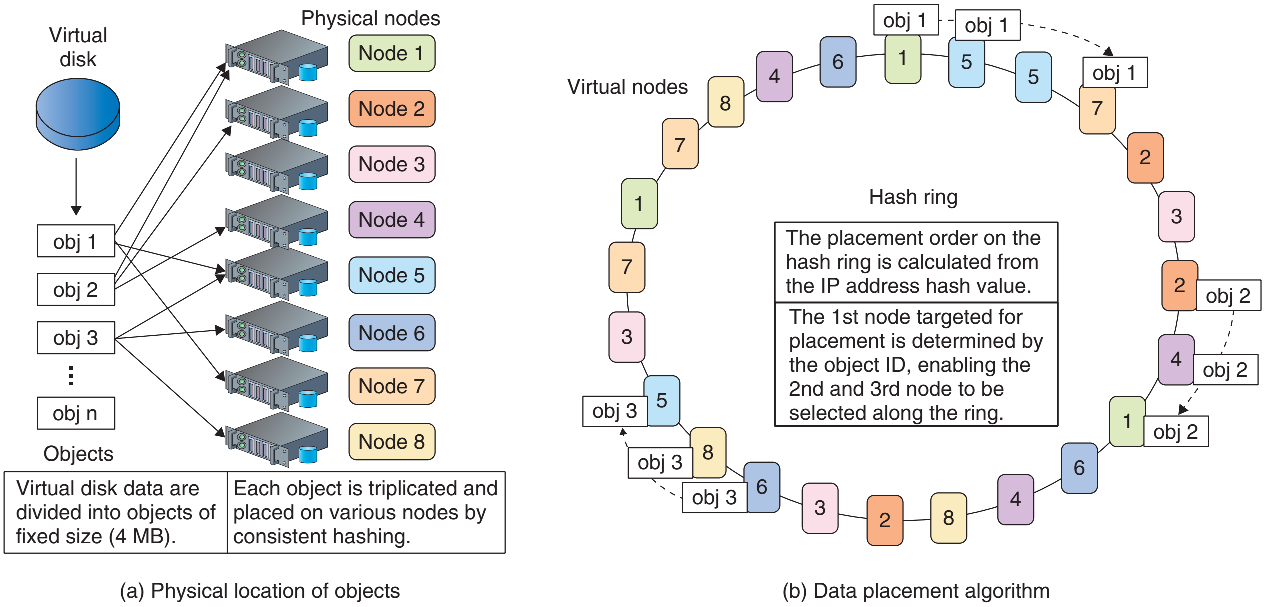
3. Recent activities

SIC is working to improve the operability and reliability of Sheepdog so that it can be used with confidence in commercial services.

Zookeeper, a de facto standard coordination kernel, can be used with Sheepdog to manage the addition and removal of servers belonging to the cluster. SIC has performed exhaustive tests and long-term stability tests on Sheepdog clusters combined with Zookeeper to uncover problems, and has proposed revisions to the Sheepdog community to solve any problems found and improve its quality.

SIC is also working to implement a multipath function that would enable the connection between a client and Sheepdog to be made with more than one server within the cluster to establish redundant paths for reading/writing. This function would enable reading/writing to continue with another server in the event that an existing connection between the client and server within the Sheepdog cluster were severed.

Furthermore, to prevent service disruption and data loss, SIC is developing a function for using a remote site in the event that an entire base fails due to a


Virtual disk
Physical nodes
Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
Node 7
Node 8
obj 1
obj 2
obj 3
obj n
Objects
Virtual disk data are divided into objects of fixed size (4 MB).
Each object is triplicated and placed on various nodes by consistent hashing.
(a) Physical location of objects
Virtual nodes
Hash ring
The placement order on the hash ring is calculated from the IP address hash value.
The 1st node targeted for placement is determined by the object ID, enabling the 2nd and 3rd node to be selected along the ring.
obj 1
obj 2
obj 3
(b) Data placement algorithm

ID: identification
IP: Internet protocol

Fig. 2. Consistent hashing.

severe disaster or power outage.

The Sheepdog open source community has also implemented a function called *erasure coding*. Rather than simply replicating objects to prevent data loss, erasure coding is a technique that stores both divided data and parity data in the manner of RAID 5 (redundant array of independent disks, level 5). This function can reduce the consumption of disk space and minimize hardware costs.

4. OpenStack Swift, a distributed object storage system

It is now common for photographs taken with a particular smartphone and stored on the cloud to be made available for viewing by other terminals. As a result, the amount of data stored on the cloud has become massive, and the demand for low-cost, high-reliability cloud storage has been growing. To meet this need, the OpenStack community has developed object storage software called OpenStack Swift (referred to below as “Swift”). The NTT Group, Rackspace, and other enterprises have had commercial success with Swift.

Swift has three key features, as summarized below (**Fig. 3**).

(1) File operations by HTTP (REST API)

Data on Swift can be managed by any terminals including smartphones, tablets, and PCs through the use of HTTP (Hypertext Transfer Protocol). Swift is suitable for unstructured data such as backup, photos, and videos.

(2) High reliability

Losing data stored on a storage system is unacceptable. Swift generally creates three replicas of data in a cluster to achieve high reliability. Furthermore, a process called *replicator* regularly runs on each object-server node in the cluster to check whether the data saved on that disk are also stored on two other disks in the cluster. If it is determined that a disk has failed and has been unmounted, a new replica of data will be automatically reproduced.

(3) Scale out

Being a distributed autonomous system, Swift has no single point of failure and is capable of scaling out from a small cluster. A typical example of a Swift cluster configuration is shown in Fig. 3. In this example, the system consists of proxy nodes that receive requests from clients and storage nodes that actually store data. This results in highly extendible cluster architecture since proxy nodes can be added if

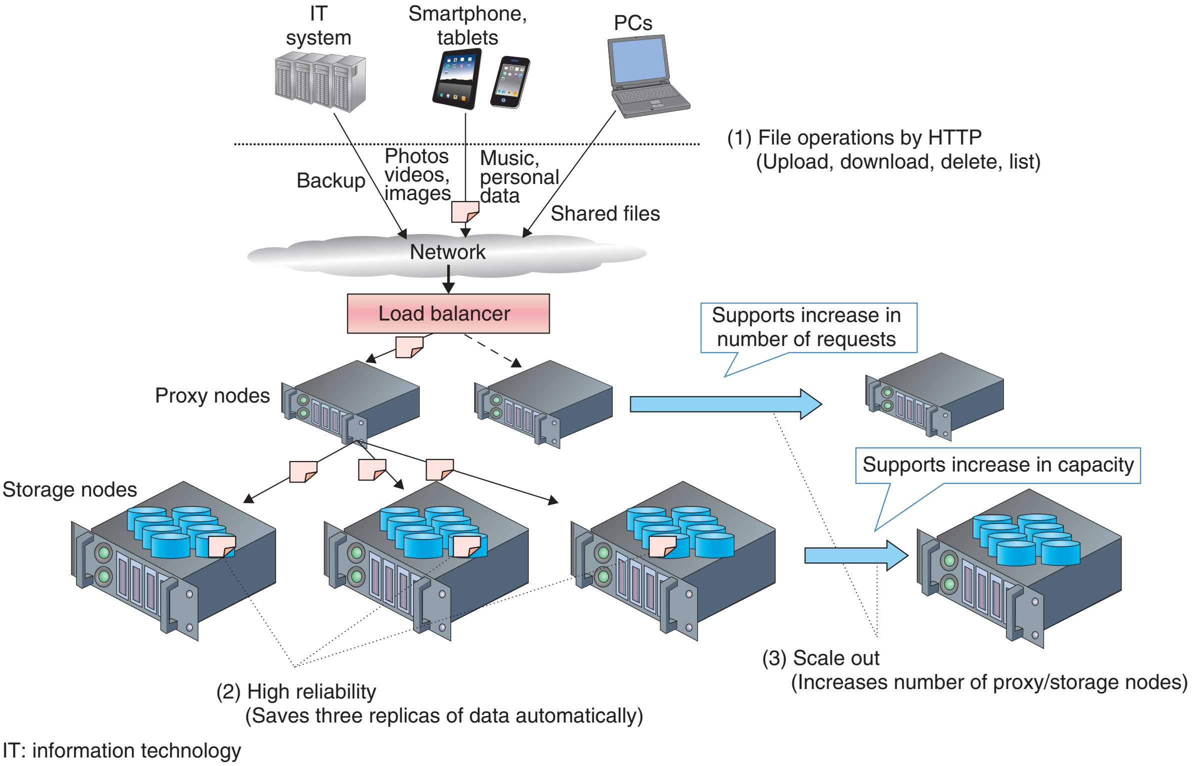

IT system
Smartphone, tablets
PCs
(1) File operations by HTTP (Upload, download, delete, list)
Backup
Photos videos, images
Music, personal data
Shared files
Network
Load balancer
Supports increase in number of requests
Proxy nodes
Supports increase in capacity
Storage nodes
(3) Scale out (Increases number of proxy/storage nodes)
(2) High reliability (Saves three replicas of data automatically)
IT: information technology

Fig. 3. Features of Swift.

requests become excessive, while storage nodes can be added if storage capacity becomes insufficient.

5. Improvement of Swift operability

SIC seeks to make Swift operation more efficient in order to facilitate the commercial cluster composed of distributed autonomous nodes and to provide services at low cost. To analyze the total operation time for a cluster with a total capacity of one petabyte, researchers at SIC constructed an actual PoC (proof of concept) environment and performed a quantitative evaluation of the time taken up by system configuring, system monitoring, equipment expansion (scale out), troubleshooting and recovery, and software updates. It was found that the time taken up by node addition for scale-out purposes as well as the time spent recovering from a disk failure made up a high percentage of the total operating time, so measures for improving in this regard were investigated.

To reduce the time needed to add nodes, it was decided to automatically install the OS (operating system) and applications using a Preboot Execution Environment (PXE) boot, to automate configuration settings using Chef, a configuration management tool, and to use Tempest, a tool for automating API testing in a pre-release operation check. Adopting these measures made a portion of the node scale-out procedure more efficient, reducing the time by about two-thirds compared to current values (**Fig. 4**). Tempest is a testing tool developed by the OpenStack community, but at SIC, researchers expanded the test items for Swift, which enabled efficient as well as complete testing.

The time from a disk failure to recovery must be minimized to ensure high data reliability. This study at SIC found that the S.M.A.R.T. (Self-Monitoring Analysis and Reporting Technology) system built into hard disk drives could be used to create a tool for automating the detection of a failed disk and for unmounting that disk (**Fig. 5**). This tool was estimated to reduce the time to recovery to one-fifth that of the manual procedure.

6. Future developments

Sheepdog is a fully symmetric distributed block storage system that provides high extendibility,

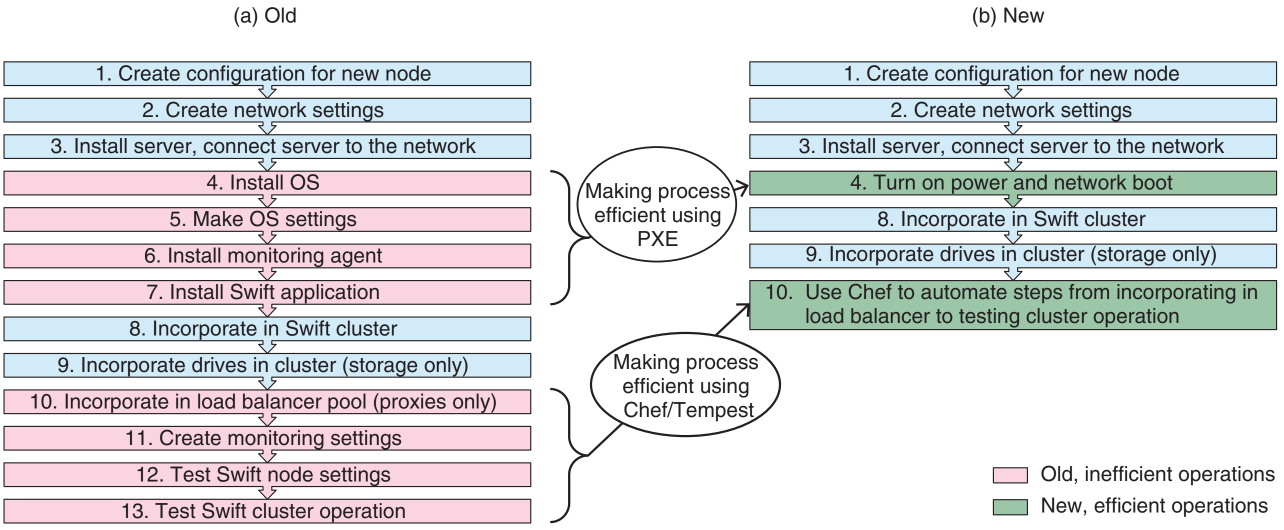

(a) Old
1. Create configuration for new node
2. Create network settings
3. Install server, connect server to the network
4. Install OS
5. Make OS settings
6. Install monitoring agent
7. Install Swift application
8. Incorporate in Swift cluster
9. Incorporate drives in cluster (storage only)
10. Incorporate in load balancer pool (proxies only)
11. Create monitoring settings
12. Test Swift node settings
13. Test Swift cluster operation
Making process efficient using PXE
Making process efficient using Chef/Tempest
(b) New
1. Create configuration for new node
2. Create network settings
3. Install server, connect server to the network
4. Turn on power and network boot
8. Incorporate in Swift cluster
9. Incorporate drives in cluster (storage only)
10. Use Chef to automate steps from incorporating in load balancer to testing cluster operation
🟥 Old, inefficient operations
🟩 New, efficient operations

Fig. 4. Raising efficiency at time of node scale out.

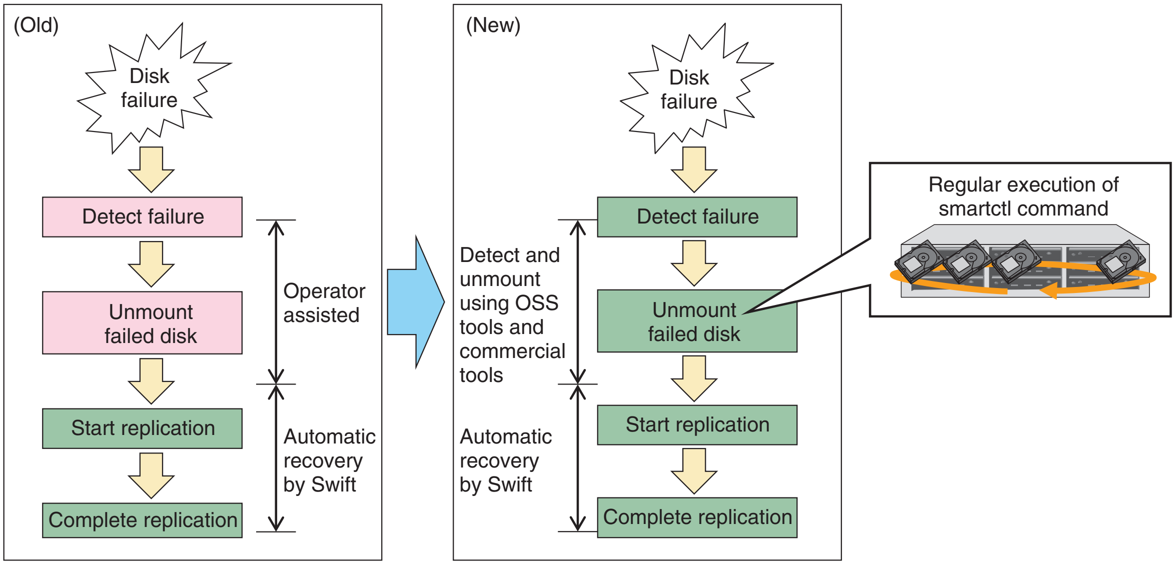
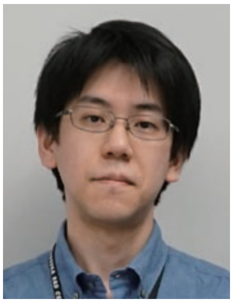


Fig. 5. Raising efficiency at time of disk failure.

reliability, and ease of operation. It is beginning to be introduced into actual services in Japan and in other countries. To help customers feel at ease about introducing Sheepdog in their operations, we plan to continue our efforts to improve quality and reliability while also sharing operating procedures and carrying out tests with users.

Swift is a highly reliable, scalable object storage system. We plan to further develop the operation automation with the operating efficiencies introduced here while also developing erasure coding (a function for raising disk usage efficiency while maintaining robustness), which is being studied as a new function in the Swift community.

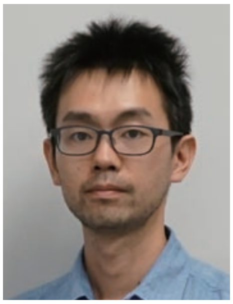
Going forward, we plan to pursue quality improvements and function extensions in both Sheepdog and Swift together with major developers and users in those communities with the aim of improving stability, performance, and operability.



Yoshifumi Fukumoto

Research Engineer, Distributed Computing Technology Project, NTT Software Innovation Center.

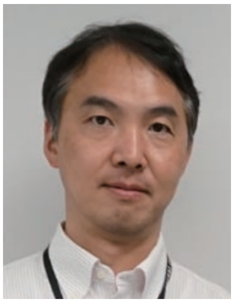
He received the B.E. in information engineering from Keio University, Tokyo, in 2009. He joined NTT Cyber Space Laboratories in 2009 and studied distributed machine learning platforms. He is currently studying distributed storage systems. He is a member of the Database Society of Japan.



Ichibee Naito

Research Engineer, NTT Software Innovation Center.

He received the M.E. in information engineering from Waseda University, Tokyo, in 2006. He joined NTT Cyber Space Laboratories (now, NTT Media Intelligence Laboratories) in 2006 and studied distributed autonomous computing platforms. He is currently studying reliability of distributed storage systems.



Toshio Hitaka

Senior Research Engineer, Supervisor, Distributed Computing Technology Project, NTT Software Innovation Center.

He received the B.E. in mathematics in 1992 and the M.E. in information engineering in 1994 from Hokkaido University. Since joining NTT in 1994, he has been engaged in R&D of database management system technology. As a result of organizational changes in July 2012, he is now with the NTT Software Innovation Center, and has been engaged in R&D of operating systems and virtualization technology.

Masahiro Shiraishi

Senior Research Engineer, Supervisor, NTT Software Innovation Center.

He received the M.E. in mathematics from Kagoshima University in 1991. He joined NTT in 1991 and studied and developed operating system platforms. He is currently studying distributed storage systems.