

Received September 4, 2019, accepted September 30, 2019, date of publication October 11, 2019, date of current version October 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946962

A Heterogeneous Cloud Storage Platform With Uniform Data Distribution by Software-Defined Storage Technologies

CHAO-TUNG YANG¹, SHUO-TSUNG CHEN², WEI-HSUN CHENG¹,
YU-WEI CHAN³, AND ENDAH KRISTIANI^{4,5}

¹Department of Computer Science, Tunghai University, Taichung City 40704, Taiwan

²Department of Information Management, College of Management, Fu Jen Catholic University, New Taipei City 24205, Taiwan

³College of Computing and Informatics, Providence University, Taichung City 43301, Taiwan

⁴Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung City 40704, Taiwan

⁵Faculty of Engineering and Computer Science, Krida Wacana Christian University, Jakarta 11470, Indonesia

Corresponding author: Yu-Wei Chan (ywchan@gm.pu.edu.tw)


This work was supported by the Ministry of Science and Technology (MOST), Taiwan, under Grant 107-2218-E-029 -004, Grant 108-2221-E-029-010, and Grant 108-2622-E-029-007-CC3.

ABSTRACT Recently, the variety of cloud technologies and applications has been increasing significantly. With the popularity of cloud applications, the number of business and personal needs is also increasing rapidly. Meanwhile, the industrial development process with cloud technology costs so much that performance will be essential in employing the cloud infrastructure in a large-scale environment. In this paper, OpenStack is first implemented as a heterogeneous cloud storage environment to increase the availability and efficiency of the cloud system. Secondly, Ceph, HDFS, and Swift as a storage service are integrated into the environment based on a proportion-based file distribution mechanism. In this case, the sub-files are distributed to different storage services. Additionally, a user-friendly web interface is developed for the administrator and regular users. Finally, the performance analysis is presented to evaluate the proposed system.

INDEX TERMS Cloud system, cloud storage service, software-defined storage, data distribution, heterogeneous storage.

I. INTRODUCTION

In the past decade, one of the main focuses in the development of information technology has been the cloud service [12], [21], in which users can upload their computing or storage needs to the cloud environment via the Internet and then receive responses after their services have been met [6], [9], [17]. Many studies on cloud storage have mainly focused on the data preprocessing method on the client side [27] and the data storage method on the server side [29]. For instance, cloud storage service users may be concerned about their data security and storage availability. With some data encryption technologies, users' data security is improved. At the same time, the storage availability is also improved by using multiple cloud storage systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Minh Jo .

In recent years, Software-Defined Storage (SDS) has emerged as a storage virtualization technology [10], [19], [20]. Inspired by the Software-Defined Network (SDN), it divides the complexity of the router into two parts: the data plane and the control plane. The data plane is responsible for the packet forwarding based on the flow entries, which are given in the control plane by the controller server. Similarly, in SDS, the storage service is also divided into two parts: the pure raw storage service and the controller, which is software that resides in a server. The controller software is mainly responsible for managing the set of raw storage devices. With SDS, different resources can be integrated and resource management can be provided, such as replication, thin provisioning, snapshot and backup functions, such that the availability and usability of storage services are improved. Therefore, SDS is a better choice for users to build cloud storage services considering cost and security.

Cloud storage services have more convenience and provide data redundancy [14], [16]. These are important indicators when the user selects this kind of service. However, due to the data being placed on third-party storage platforms, the data security in cloud storage is not fully reliable for the user. To solve this issue, most research has used the encrypt algorithms or erasure-code technologies to improve data security [15], [18]. In this way, more than one cloud storage service might be needed to distribute data. Thus, in our previous work [35], we have proposed a proportion-based file distribution mechanism by integrating Ceph, HDFS and Swift open-source storage services. With this mechanism, the sub-files could be distributed to different storage systems successfully. However, some detailed functions and analyses were not considered and implemented. In this work, we implement a more complete open-source cloud storage platform to provide private heterogeneous cloud storage services. In this work, we simulate the complex environment of cloud storage services and evaluate their effectiveness. In addition, a graph user interface is provided to support file sharing. The user and administrator can operate the storage services through the web interface to access cloud services anytime and anywhere with any network devices.

The main goal of the system is to build a cloud platform contains a variety of storage technologies and achieve a uniform distribution of data stored thereon. The main contributions of this paper are summarized as follows.

- 1) We design an architecture of SDS technology for uniform data distribution on various cloud storage services.
- 2) We present an easy-to-use and friendly user interface for the file sharing of users and administrators.
- 3) Performance analyses of the proposed system architecture are presented.

The rest of the paper is organized as follows. Section II introduces the literature review and related works. In Section III, the system architecture and implementations are described. Experimental results are shown in Section IV. Finally, concluding remarks are given in Section V.

II. LITERATURE REVIEW AND RELATED WORKS

In this section, background and related works are reviewed to let users realize the system design and implementation specifically.

A. LITERATURE REVIEW

1) VIRTUALIZATION

By virtualizing the physical resources of a computer (e.g., server, network, memory, and storage) presented after conversion, the user can configure a better use of these resources. The virtualized resources typically included data storage and power, herein, especially server virtualization. The server virtualization refers to the use of one or more host hardware settings and has virtual configuration flexibility. The server virtualization consists of three categories,

which are the full virtualization, the para-virtualization and the hardware-assisted virtualization, respectively.

2) SDS

This computer data storage technology term [10], [19], [20] distinguishes storage hardware from the software that arranges the storage infrastructure which provides functions of policy management, such as duplication, replication, thin provisioning, snapshots, and backup.

3) SWIFT

The Swift is a part of OpenStack components and has a powerful redundant storage system [7]. OpenStack is responsible for ensuring the integrity of data replication across the clusters. Storage clusters scale horizontally simply by adding new servers. OpenStack will replicate its content from other active nodes to new locations in the cluster when a server or hard drive fails. In this case, inexpensive hard drives and servers can be used because of the ability of OpenStack to replicate across different devices. Figure 1 shows the main components of Swift (the icons with colors).

Each component is independent with the current object-replicator. The origin logic of the object-replicator is split into four parts with different colors. The components in cyan are responsible for calculating the hash in real time. The components in pink are used to index the hash of the suffix and to partition directories, to receive and send requests for comparing the hash of the partition or suffix, and to generate jobs for replicating suffix directories to the replication-queue. The partition-monitor is responsible for checking whether the partition has moved at the interval. The suffix-transporter is responsible for monitoring the replication-queue and invoking rsync to sync suffix directories.

4) CEPH

The Ceph which is a software storage platform intended to present block, object and file storage from a distributed computer cluster provides good performance, reliability, and scalability [31]. In addition, it is easy for users to manage the system due to its inherent self-management characteristics. Its architecture is presented in Figure 2.

5) HDFS

The Hadoop Distributed File System (HDFS) [3], [25] is a distributed file system designed to handle terabytes (or even petabytes) big data and to provide high-throughput access to the data. Files are replicated and stored across multiple physical machines to ensure high availability and reliability for big data processing applications.

In the HDFS architecture, there are mainly two kinds of nodes, which are the NameNode and the DataNode, respectively. The NameNode in a cluster is responsible for arranging the name space of file systems and granting the client's file read/write access. In addition, the NameNodes also serves as the administrator who is responsible for managing the data blocks creation, deletion and replication of DataNodes.

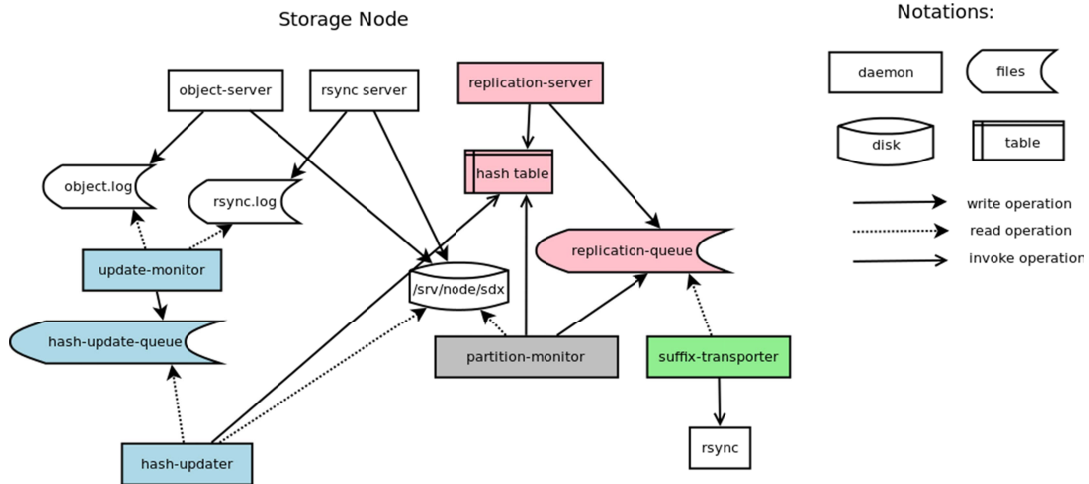


FIGURE 1. The swift system architecture.

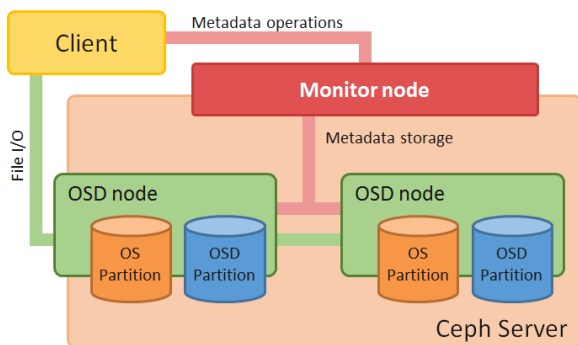


FIGURE 2. The ceph system architecture.

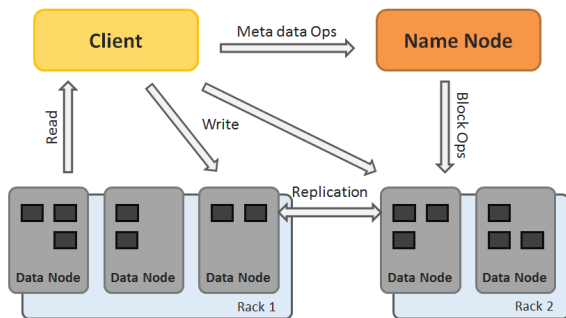


FIGURE 3. The HDFS architecture.

On the other hand, the DataNode is responsible for storing data in blocks within files and performing input/output operations of clients. The architecture of HDFS is presented in Figure 3.

6) OPENSTACK

OpenStack is a cloud computing project for public and private clouds that was released under the terms of the Apache License as an open-source platform [4], [23]. The project aims to simplify the implementation of all types

of clouds. Various interrelated project components of the cloud infrastructure are available in the OpenStack system. The technology monitors a large number of processing, storage, and networking resources throughout a data-center. In the platform, a dashboard is provided for administrators to monitor and control the allocation of users' resources via a web interface. Its architecture is presented in Figure 4 (the Kilo version).

B. RELATED WORKS

Spillner *et al.* [27] proposed integration platform compatibility with every cloud storage service to provide an uninterrupted storage system. However, the system is mainly for the user experience and is used to linking up every kind of cloud platform, not for processing files. Penga and Jiangb [17] proposed a cloud storage service system with a solution to deploy a cloud storage service system based on the open-source distributed database

In addition, Shen *et al.* [24] proposed a way for different file sizes to be uploaded to the most suitable system. However, their method will to all files being stored in the same system whether the user employs large or small files. Wu *et al.* [33] presented a method to store data in different cloud services after splitting the files and proposed a method for copying the file's requirements. However, they do not discuss the method of file distribution in their paper. Thus, this paper demonstrates a method for supporting distributed files, enabling the system to expand nodes, and providing load-balanced storage.

III. SYSTEM DESIGN AND IMPLEMENTATION

In this section, the overall system architecture is introduced. The proposed heterogeneous storage system is implemented based on the architecture and the proposed distribution mechanism. In addition, a graphical user interface is provided for users to access the whole system easily.

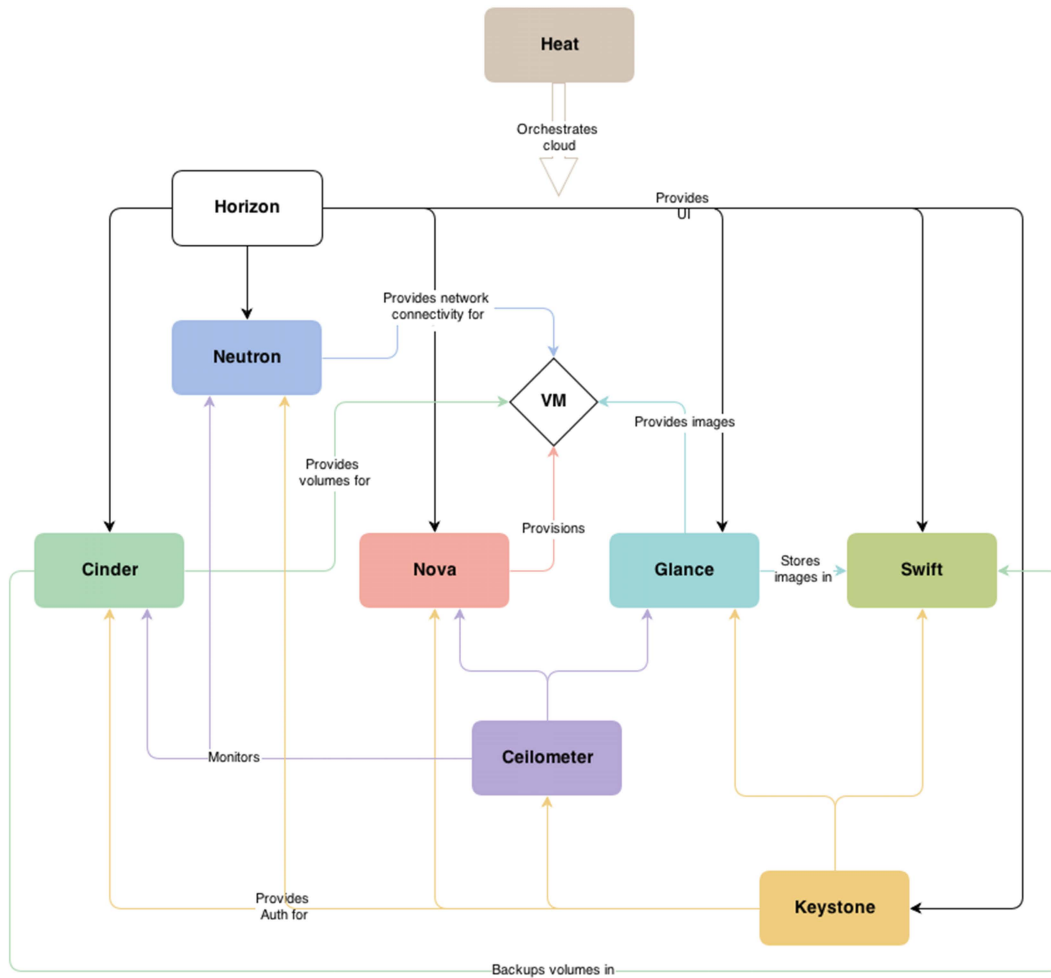


FIGURE 4. The openstack kilo architecture.

A. SYSTEM ARCHITECTURE

Figure 5 depicts the system architecture. OpenStack is implemented as the core of storage virtualization and unified management. The proposed system is based on OpenStack, a virtual machine is created to provide storage, control, and monitoring services. In the system, first, the storage services are based on heterogeneous storage platforms, which are the HDFS, Ceph, and Swift. Second, the controller manages the storage services and the heterogeneous storage. Third, the monitor service inspects the remaining capacity of each heterogeneous storage platform. Then, a distribution mechanism is proposed to the controller to allocate the files received from users. By using this mechanism, the files are automatically assigned to an appropriate storage service after users upload files. The detailed concept of the controller is shown in Figure 6.

B. SYSTEM IMPLEMENTATION

In the implementation of the proposed system, there are three main components, which are the storage service deployment, the file distribution mechanism and the

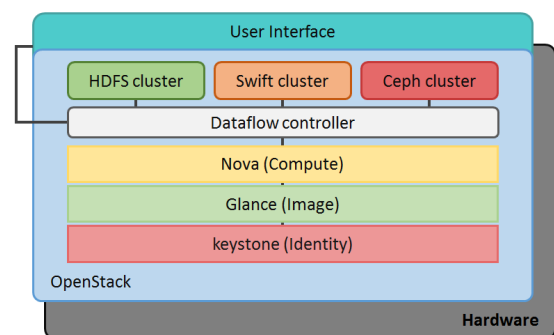


FIGURE 5. The proposed system architecture.

user service, respectively. In the following subsections, these components will be stated in detail.

1) THE STORAGE SERVICE DEPLOYMENT

The OpenStack is implemented to build and manage the proposed cloud system by using virtual machines with the Ubuntu Operating System (OS). The overall system is shown in Figure 7 and Figure 8.

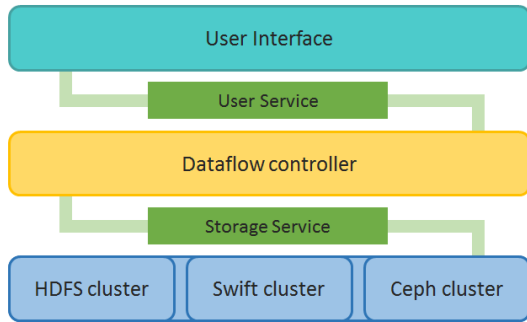


FIGURE 6. The controller architecture of our proposed system.

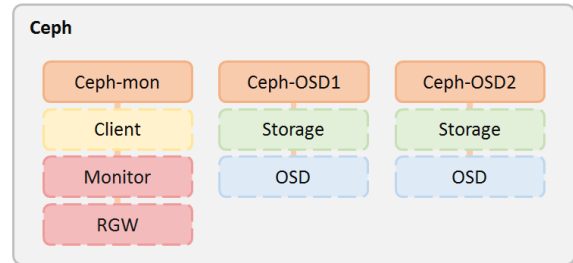


FIGURE 9. The overview of ceph environment.

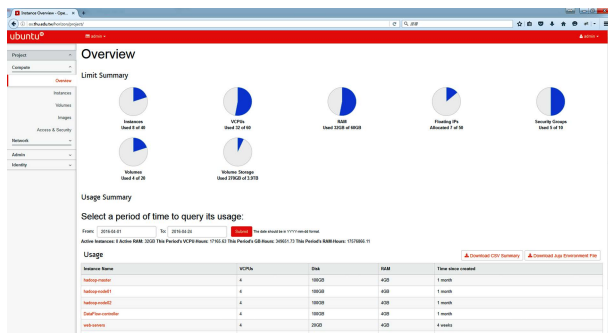


FIGURE 7. The overview page of openstack.

实例名称	映像名称	IP 地址	容量	密钥对	状态	可用区域	任务	电源状态	寿命	Actions
ceph-OSD2	ubuntu-20G	10.0.0.75	ubuntu_4.4.20	demo-key	使用中	compute02	无	正在执行	0分	新增即时存储
ceph-OSD1	ubuntu-20G	10.0.0.74	ubuntu_4.4.20	demo-key	使用中	compute01	无	正在执行	8分	新增即时存储
ceph-mon	ubuntu-20G	10.0.0.73	ubuntu_4.4.20	demo-key	使用中	compute01	无	正在执行	17分	新增即时存储

FIGURE 10. The ceph instance page.

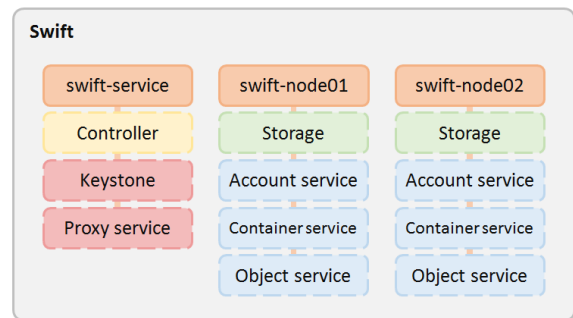


FIGURE 11. The swift environment overview.

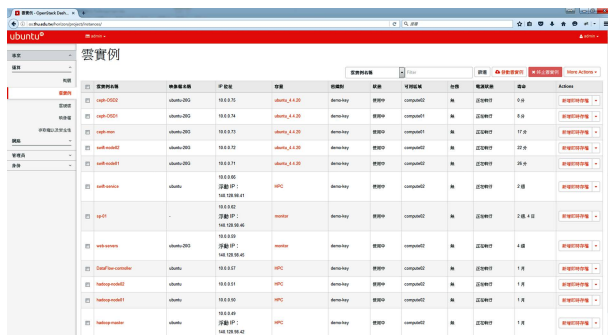


FIGURE 8. The VM instance page in openstack.

实例名称	映像名称	IP 地址	容量	密钥对	状态	可用区域	任务	电源状态	寿命	Actions
swift-node02	ubuntu-20G	10.0.0.72	ubuntu_4.4.20	demo-key	使用中	compute02	无	正在执行	22分	新增即时存储
swift-node01	ubuntu-20G	10.0.0.71	ubuntu_4.4.20	demo-key	使用中	compute02	无	正在执行	26分	新增即时存储
swift-service	ubuntu	10.0.0.66 浮动 IP : 140.128.98.41	HPC	demo-key	使用中	compute02	无	正在执行	2速	新增即时存储

FIGURE 12. The swift instances page.

a: CEPH DEPLOYMENT

Ceph is applied for a distributed object storage and file system and to provide the interface for object-, block-, and file-level storage. To achieve the required functions, Ceph has three kind of physical nodes: Monitors (MON), an Object Storage Daemon (OSD), and a Metadata (MDS) service. According to the object storage deployment requirements, as shown in Figure 2, the proposed system only needs to install OSDs and MONs. The overview of the Ceph architecture is shown in Figure 9 and Figure 10.

b: THE SWIFT DEPLOYMENT

Swift is one of the components in OpenStack. Figure 1 shows the Swift architecture. The Swift service includes container, proxy, object, and account servers. An authentication and authorization mechanism, such as the identity service

is provided in the proxy server. It also provides an internal mechanism which permits operations without OpenStack services. Based on Swift deployment requirements, the proposed system needs to include the following components: an identity service and account, proxy, object, and container servers. The Swift environment in the system is shown in Figure 11 and Figure 12.

c: THE HDFS DEPLOYMENT

In the proposed system, a HDFS architecture is employed which consists of one master node and two slave nodes. Figure 13 and Figure 14 show the details of the HDFS system. The NameNode executes the file system namespace and determines the mapping blocks of DataNodes. DataNodes have the task of serving read and write requests from clients of the file system.

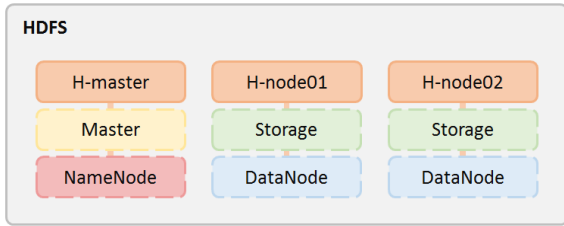


FIGURE 13. The HDFS environment overview.

实例名称	映像名称	IP 地址	容量	密钥	状态	可用区域	任务	电源策略	寿命	Actions
hadoop-node02	ubuntu	10.0.0.51	HPC	demo-key	使用中	compute02	无	正在执行	1月	新增即时存储
hadoop-node01	ubuntu	10.0.0.50	HPC	demo-key	使用中	compute02	无	正在执行	1月	新增即时存储
hadoop-master	ubuntu	10.0.0.49 浮动 IP : 140.128.98.42	HPC	demo-key	使用中	compute02	无	正在执行	1月	新增即时存储

FIGURE 14. The HDFS instances page.

2) THE FILE DISTRIBUTION MECHANISM

In this subsection, we propose a mechanism for file distribution and appropriate file allocation according to the environments and specifications of the Swift, HDFS and Ceph platforms. In the first phase of the mechanism, the remaining capacity information of each storage platform can be observed through the monitor node. In the second phase, each file is split into several sub-files with a similar file size to the proportion of the remaining capacity of all storage platforms. Based on the above information, each sub-file is allocated to a proper storage platform. The details of proposed mechanism is presented by the following equations.

$$X = \sum_{i=1}^n S_i. \tag{1}$$

$$S_i = X * R_i, \quad i = 1 \text{ to } n. \tag{2}$$

- X is the user-uploaded file size. We use it as file segmentation data.
- S is the result of the split file size. It can also be uploaded to the storage system's actual size.
- R is the split ratio. We use it to find the value S .
- n is the number of storage system selection. In our case, we have three storage system so that n is 3.

As shown above, we know that R is an important parameter that influences the platform capacity convergence speed. We show how to determine this value in the following:

$$R_j = \frac{C_j - \gamma * \min(C)}{\sum_{i=1}^n C_i - \gamma * \min(C)}, \quad j = 1 \text{ to } n \tag{3}$$

- C is the system storage capacity ratio satisfying $\sum_{i=1}^n C_i = 1$.
- γ is a filtering value satisfying $0 < \gamma \leq 1$.

As a result, we can obtain R with equation 3. After the process described above, we start to split our file. Finally, all the split files will be uploaded to the specified platform. A flow chart of the proposed file distribution mechanism is shown in Figure 15.

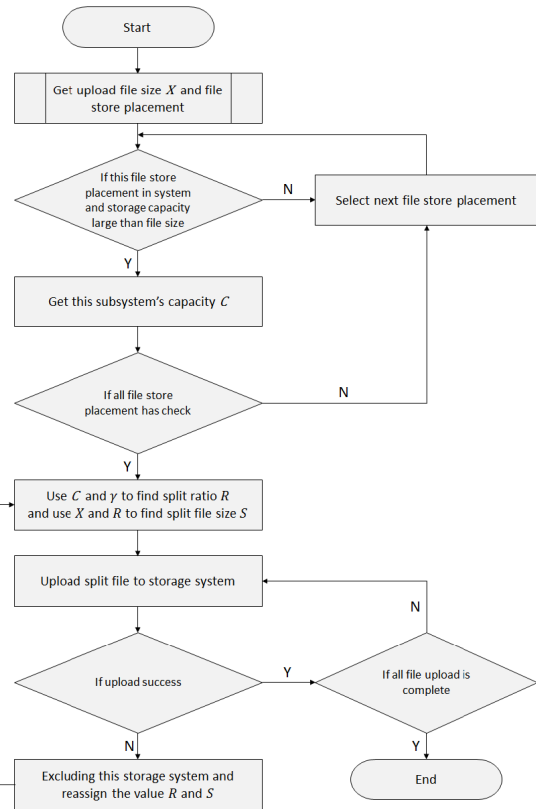


FIGURE 15. Flow chart of the proposed data distribution method.

3) USER SERVICE

In the system, a website is developed by using the JQuery and PHP language as an user interface to solve the platform compatibility problems. The web implements the web services due to the compatibility of various platforms, such as PC, mobile, and tablet.

In addition, the Responsive Web Design (RWD) approach is used to design the web interface. The RWD approach provides users with the best visual andis used interactive experience. In the system, the Bootstrap framework is used to develop the front-end component of our website. The Bootstrap provides an easy-to-use and powerful front-end development environment, in which HTML- and CSS-based design templates for common user interface components are provided.

The PC screen is shown in Figure 16, and the mobile screen is shown in Figure 17. We can see different kinds of web layout but in the same style because there is only one html page on the server. The webpage will show a different layout according to the user's screen.

IV. EXPERIMENTAL RESULTS

In this section, the experimental results are presented. First, the speeds of each storage platform are measured based on the file distribution mechanism. Second, the time spent on file splitting is measured. Third, the time gaps of the file upload time are compared. Finally, the user interface is described.

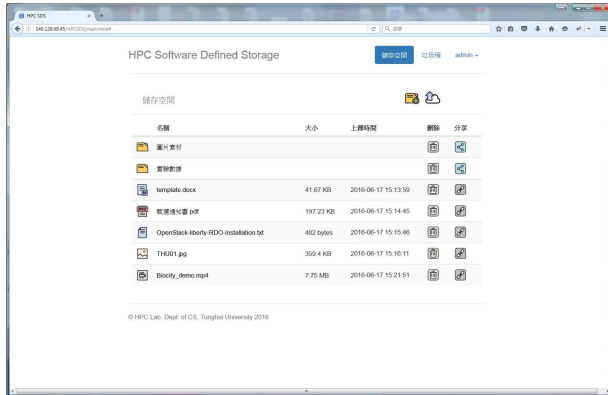


FIGURE 16. The responsive web design.



FIGURE 17. The responsive mobile design.

A. THE EXPERIMENTAL ENVIRONMENT

In this section, the experimental environment is presented in detail. HDFS was constructed by three VMs with the following specifications: 4-core CPU, 4-GB memory, and a total of 200-GB storage space. Tables 1, 2 and 3 show the experimental environment specifications.

B. PERFORMANCE

1) MEASUREMENTS OF THE UPLOAD SPEED

Figure 18 shows the upload speed results. The red line represents HDFS, the blue line represents Ceph, and the gray line

TABLE 1. Hardware specification.

Host name	CPU	Memory	Disk	OS
Openstack Controller	12 cores	64GB	2TB	Ubuntu 14.04.02
Openstack Network	24 cores	64GB	2TB	Ubuntu 14.04.02
Openstack Compute01	64 cores	96GB	2TB	Ubuntu 14.04.02
Openstack Compute02	64 cores	96GB	2TB	Ubuntu 14.04.02
Openstack Block Storage	64 cores	48GB	8TB	Ubuntu 14.04.02

TABLE 2. Virtual machine specification.

Host name	CPU	Memory	Disk	OS
client	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Ceph mon	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Ceph OSD1	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Ceph OSD2	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Swift controller	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Swift node01	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
Swift node02	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
HDFS master	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
HDFS node01	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02
HDFS node02	4 cores vCPU	4GB	20GB	Ubuntu 14.04.02

TABLE 3. Software specification.

Software	Version
OpenStack	Kilo
Ceph	10.2.1 Jewel
Swift	2.1.0
Hdfs	2.7.1

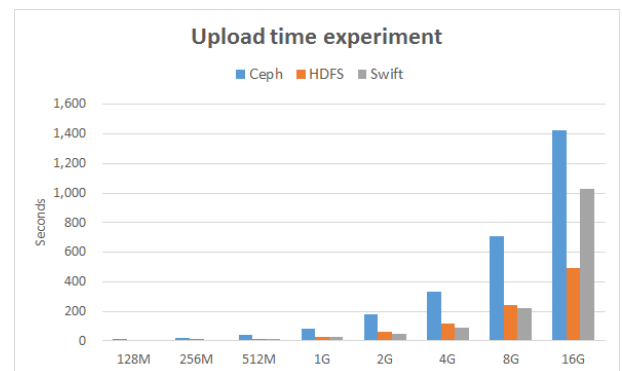


FIGURE 18. Measurements of the uploading speed.

represents Swift. It can be seen in the configuration that Ceph and Swift have poor upload performance for large files.

2) MEASUREMENTS OF THE NETWORK INFRASTRUCTURE SPEED

Network throughput is a critical factor that affects cluster performance. To determine the network performance, iPerf was chosen, which uses a client-server connection to measure TCP and UDP bandwidth as the testing tool. The results are shown in Figure 19. In the histogram, the vertical axis is the transmission bandwidth, while the horizontal axis is the number of tests. Ceph's bandwidth was almost 560 Mbits/s.

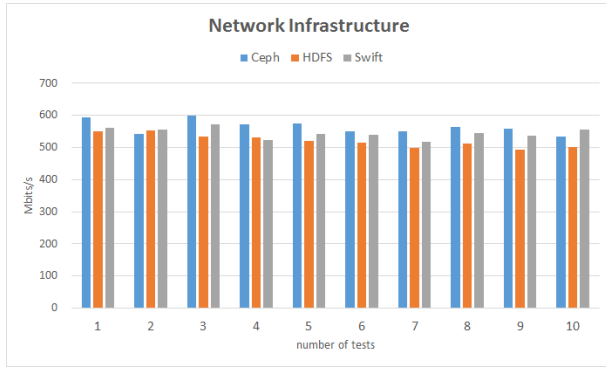


FIGURE 19. Measurements of the network infrastructure speed.

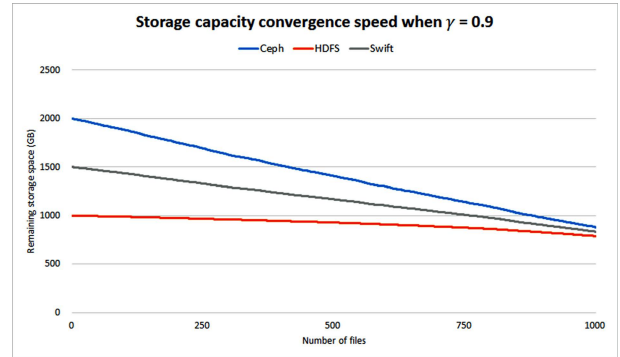


FIGURE 22. The convergence speed of storage capacity when $\gamma = 0.9$.

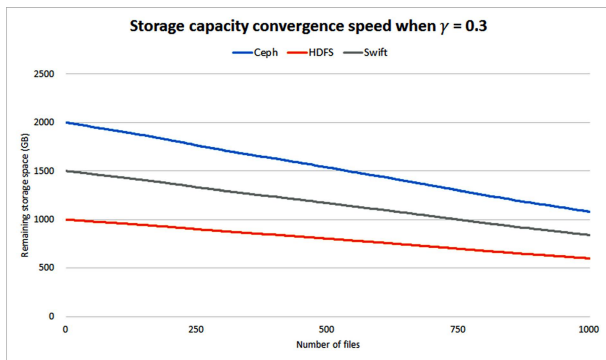


FIGURE 20. The convergence speed of storage capacity when $\gamma = 0.3$.

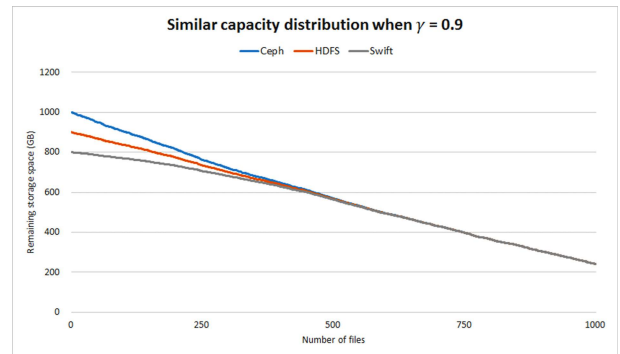


FIGURE 23. The similar capacity distribution with $\gamma = 0.9$.

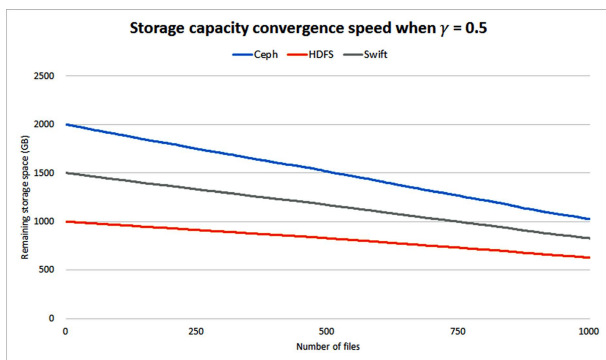


FIGURE 21. The convergence speed of storage capacity when $\gamma = 0.5$.

The HDFS bandwidth was approximately 520 Mbits/s. Swift’s bandwidth was almost 545 Mbits/s. The results were not significantly different.

3) DATA DISTRIBUTION EXPERIMENT

The methods allow the user to set the γ value to control the storage capacity convergence speed. To evaluate the distribution effectiveness for different γ values, this work examined Ceph 2000-Gigabyte, HDFS 1000-Gigabyte, and Swift 1500-Gigabyte capacities. Then, it compared the three cases with γ values equal to 0.3, 0.5 and 0.9, respectively. The results are shown in Figures 20, 21 and 22.

In these three experiments, 1000 random-sized files of around 1 Mb to 4 Gb were uploaded to our system. It can be seen that the large γ value might have caused a more obvious convergence rate. This means if the user sets a large γ value, more upload files will be split into the large-capacity storage system.

Moreover, to determine if the storage system capacity is similar, the method shows the kind of distribution. In the next experiment, the system examined Ceph 1000-Gigabyte, HDFS 900-Gigabyte, and Swift 800-Gigabyte capacities, and $\gamma = 0.9$ and 1000 random-sized files of around 1 Mb to 4 Gb were uploaded to our system. The results are shown in Figure 23.

As shown in the results above, it can be seen that the storage platform’s capacity became the same. After that, the γ value was not important for the file splitting. This means the split data became equal in distribution.

In a cloud storage system, adding new storage nodes is a common situation. A case that expands the storage space was designed during the experiment. The results are shown in Figure 24.

As shown in the above experiment, an additional 500 Gigabytes of storage space was added to HDFS when its storage space was close to 400 Gigabytes. It was found that when the storage space increased, the reduction ratio changed. This proves that the proposed system was based on the current storage capacity of the system to split uploaded files. Finally,

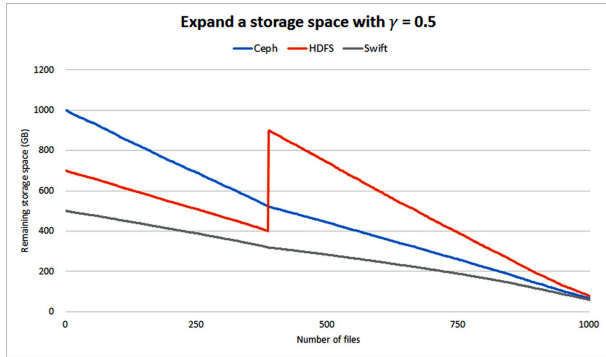


FIGURE 24. Expand a storage space with $\gamma = 0.5$.

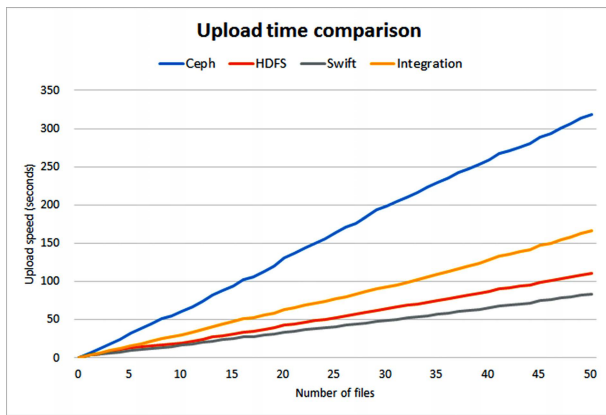


FIGURE 25. Comparisons of the uploading time.

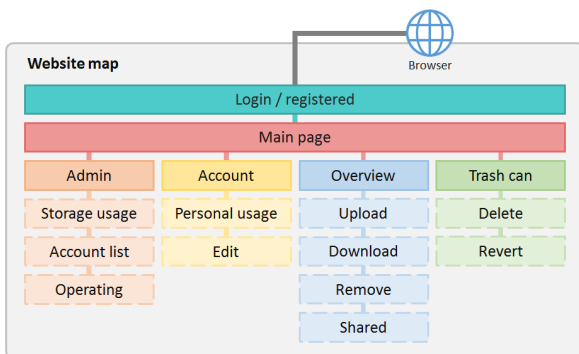


FIGURE 26. Overview of the user interface page.

a comparison of the upload time with the single-storage system is presented. The results are shown in Figure 25.

This diagram data is from Figure 22. The data size was sorted, and 50 pieces of data with 20 data gaps were chosen. This experiment clearly shows that the complex upload speed was homogenized by the sub-system’s upload speed.

C. USER INTERFACE

In this subsection, a user interface is stated. Figure 26 demonstrates an overview of the user interface. The system mainly provides three different kinds of user interfaces, which are the *Overview*, the *Trash – Can* and the *Accounting*, respectively. In each interface, several functions are listed in the



FIGURE 27. The overview interface.

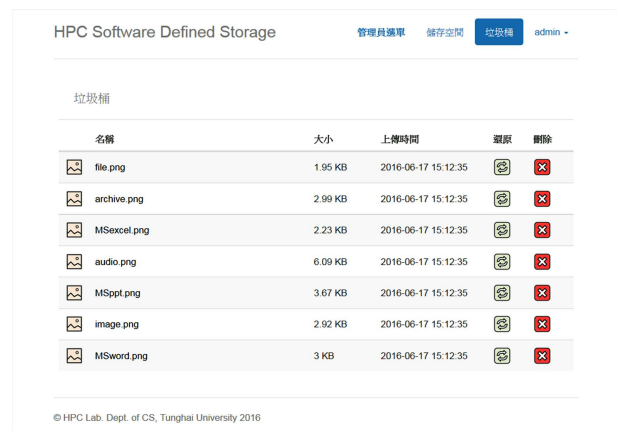


FIGURE 28. The trash-can interface.

main page. In the following, each kind of interfaces will be introduced specifically.

1) THE OVERVIEW INTERFACE

The *Overview* interface is an important part of this system, as shown in Figure 27. It mainly contains the following standard operations: download, upload, share and remove folder. Additionally, a file link is provided for the user to download a file without logging in. To implement the functions of file upload, AJAX, JQuery, and Bootstrap were used to show the uploading progress by the progress bar. This can enhance the user experience when uploading large files. The interface also provides a multi-file upload feature.

2) THE TRASH-CAN INTERFACE

The interface enables the user to recover the deleted files, as shown in Figure 28. With the interface, users can easily delete files manually or automatically.

3) THE ACCOUNTING INTERFACE

With this interface, users can edit their user account, such as their username, password and E-mail address. The interface is shown in Figure 29.

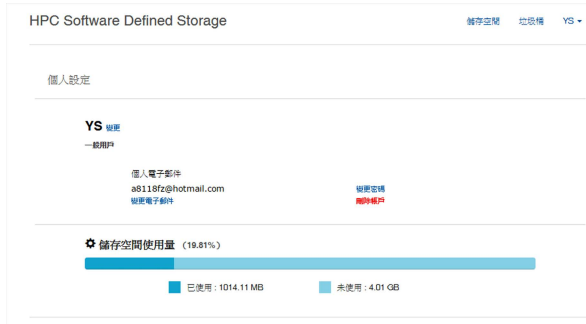


FIGURE 29. The accounting interface.

V. CONCLUSION AND FUTURE WORK

In this work, a heterogeneous cloud storage platform was built based on software-defined storage technology. In the system, for the file storage, a method which provided uniform data distribution to achieve storage resource load balance was presented. A heterogeneous storage system that integrates three different kinds of SDS open-source software was established. In addition to evaluating each of the storage software, the system could also stimulate the local system convergence of various public cloud storage conditions using this environment.

For the file storage mechanism, a method supporting uniform data distribution was proposed. The Gamma was a user-defined value that influenced the storage space convergence speed. It could be found that the more substantial the gamma values, the faster the storage space convergence speed. This method allowed the user to add different sizes of storage space, and the final storage resources could achieve storage load balancing. Moreover, a high-usability user interface was provided. This interface was designed as a web application and based on the RESTful architecture. To enhance the user experience, asynchronous JavaScript and XML technology was applied, thereby enabling the web application to update the content without redirecting and reducing resource load by refreshing all pages.

Due to the lack of hardware resource quantity, the storage cluster was built with virtualization technology. In the future, it can be applied by using a physical machine environment. The currently used back-end storage system was limited to open-source. Thus, further development is needed, including the public and private cloud as a hybrid cloud storage environment. For the file storage function, an improvement of security and availability of the system and providing more useful features for file operations in the user interface are required.

REFERENCES

- [1] (2016). *Cubic Spline*. [Online]. Available: <http://mathworld.wolfram.com/CubicSpline.html>
- [2] (2016). *EMC ViPR*. [Online]. Available: <http://www.emc.com/vipr>
- [3] (2016). *Hadoop HDFS*. [Online]. Available: <https://hadoop.apache.org/>
- [4] (2016). *OpenStack*. [Online]. Available: <https://www.openstack.org/>

- [5] A. Agrawal, R. Shankar, S. Akarsh, and P. Madan, "File system aware storage virtualization management," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Oct. 2012, pp. 1–11.
- [6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [7] J. Arnold, *OpenStack Swift: Using, Administering, and Developing for Swift Object Storage*. Newton, MA, USA: O'Reilly Media, Inc., 2014.
- [8] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Burlington, MA, USA: Morgan Kaufmann, 1995, pp. 9–17.
- [9] A. Botta, W. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generat. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.
- [10] Coraid. (2013). *The Fundamentals of Software-Defined Storage*. [Online]. Available: <http://san.coraid.com/>
- [11] J. Fan and Q. Yan, *Nonlinear Time Series: Nonparametric and Parametric Methods*. New York, NY, USA: Springer, 2005.
- [12] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360° compared," in *Proc. Grid Comput. Environ. Workshop*, Nov. 2008, pp. 1–10.
- [13] T. H. Hussain, P. N. Marimuth, and S. J. Habib, "Managing distributed storage system through network redesign," in *Proc. 15th Asia-Pacific Netw. Oper. Manage. Symp.*, Sep. 2013, pp. 1–6.
- [14] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 2, p. 12, May 2017.
- [15] S. K. Majhi and S. K. Dhal, "Placement of security devices in cloud data centre network: Analysis and implementation," *Procedia Comput. Sci.*, vol. 78, pp. 33–39, Apr. 2016.
- [16] H. X. Mao, X. L. Shu, K. Huang, and L. Zhang, "Research of data reliability technology based on erasure code redundancy technology in cloud storage," *Adv. Mater. Res.*, vols. 912–914, pp. 1345–1348, Apr. 2014.
- [17] C. Peng and Z. Jiang, "Building a cloud storage service system," *Procedia Environ. Sci.*, vol. 10, pp. 691–696, Dec. 2011.
- [18] M. M. Potey, C. A. Dhote, and H. D. Sharma, "Homomorphic encryption for security of cloud data," *Procedia Comput. Sci.*, vol. 79, pp. 175–181, Apr. 2016.
- [19] S. Robinson. (2013). *Software-Defined Storage: The Reality Beneath the Hype*. [Online]. Available: <http://www.computerweekly.com/opinion/Software-defined-storage-The-reality-beneath-the-hype>
- [20] M. Rouse. (2013). *Software-Defined Storage*. [Online]. Available: <http://searchsdn.techtarget.com/definition/software-defined-storage>
- [21] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2263–2268, 2013.
- [22] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Proc. 2nd Int. Conf. Comput. Netw. Technol.*, Apr. 2010, pp. 222–226.
- [23] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, pp. 38–42, 2012.
- [24] Y.-C. Shen, C.-T. Yang, S.-T. Chen, and W.-H. Cheng, "Implementation of software-defined storage service with heterogeneous object storage technologies," in *Proc. ASE BigData Social Inform.*, 2015, vol. 29, no. 4, Art. no. 3.
- [25] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol.*, May 2010, pp. 1–10.
- [26] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors," in *Proc. 2nd ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst.*, 2007, vol. 41, no. 3, pp. 275–287.
- [27] J. Spillner, J. Müller, and A. Schill, "Creating optimal cloud storage systems," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1062–1072, 2013.
- [28] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. M. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith, "Intel virtualization technology," *Computer*, vol. 38, no. 5, pp. 48–56, May 2005.
- [29] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.
- [30] D. Wang, "An efficient cloud storage model for heterogeneous cloud infrastructures," *Procedia Eng.*, vol. 23, pp. 510–515, Dec. 2011.

- [31] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Oper. Syst. Design Implement.*, 2006, pp. 307–320.
- [32] S. A. Weil, A. W. Leung, S. A. Brandt, and C. Maltzahn, "RADOS: A scalable, reliable storage service for petabyte-scale storage clusters," in *Proc. 2nd Int. Workshop Petascale Data Storage, Held Conjoint. Supercomput.*, 2007, pp. 35–44.
- [33] S. Wu, K.-C. Li, B. Mao, and M. Liao, "DAC: Improving storage availability with deduplication-assisted cloud-of-clouds," *Future Generat. Comput. Syst.*, vol. 74, pp. 190–198, Sep. 2017.
- [34] Q. Zheng, H. Chen, Y. Wang, J. Zhang, and J. Duan, "COSBench: Cloud object storage benchmark," in *Proc. 4th ACM/SPEC Int. Conf. Perform. Eng. (ICPE)*, S. Seelam, Ed. New York, NY, USA, 2013, pp. 199–210.
- [35] W.-H. Cheng, C.-I. Chiang, C.-T. Yang, S.-T. Chen, and J.-C. Liu, "The implementation of supporting uniform data distribution with software-defined storage service on heterogeneous cloud storage," in *Proc. 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2016, pp. 610–615.



CHAO-TUNG YANG received the Ph.D. degree in computer science from National Chiao Tung University, in July 1996. He is a Distinguished Professor of computer science with Tunghai University, Taiwan, where he joined the Faculty of the Department of Computer Science, in August 2001. He has published more than 280 articles in journals, book chapters, and conference proceedings. His current research interests include cloud computing and service, big data, parallel computing, and multicore programming. He is a member of the IEEE Computer Society and the ACM. He is serving on a number of journal editorial boards, including *Future Generation Computer Systems*, the *International Journal of Communication Systems*, *KSII Transactions on Internet and Information Systems*, and the *Journal of Cloud Computing*.

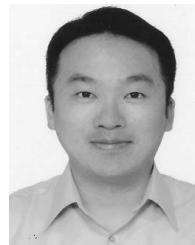


SHUO-TSUNG CHEN received the B.S. degree in mathematics from National Cheng Kung University, Tainan, in 1996, the M.S. degree in applied mathematics from Tunghai University, Taichung, Taiwan, in 2003, and the Ph.D. degree in electrical engineering from National Chinan University, Nantou, Taiwan, in 2010. He is currently an Assistant Professor with the Department of Information Management, College of Management, Fu Jen Catholic University, Taiwan. He is also an Adjunct

Assistant Professor with the Department of Electronic Engineering, National Formosa University, Taiwan.



WEI-HSUN CHENG received the M.S. degree in computer science from Tunghai University, Taichung, Taiwan, in 2015. He is currently a computer engineer.



YU-WEI CHAN received the B.S. and M.S. degrees in information engineering from Tamkang University, Taiwan, in 1997 and 2001, respectively, and the Ph.D. degree in computer science from National Tsing Hua University (NTHU), in 2013. He joined the College of Computing and Informatics of Providence University (PU), Taichung, in February 2016, as an Assistant Professor. His current research interests include game theoretic wireless networking, cognitive radio networks, cloud computing, and big data analysis.



ENDAH KRISTIANI received the M.S. degree in electrical engineering (information technology) from Universitas Gadjah Mada, Yogyakarta, Indonesia, in 2007. She is currently pursuing the Ph.D. degree with the High Performance Computing Laboratory, Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung, Taiwan. In August 2007, she joined the Department of Informatics Engineering, Faculty of Engineering and Computer Science, Krida Wacana Christian University (UKRIDA) Jakarta.

...