


RESEARCH

Open Access



# On construction of a cloud storage system with heterogeneous software-defined storage technologies

Chao-Tung Yang<sup>1</sup>, Shuo-Tsung Chen<sup>2</sup>, Yu-Wei Chan<sup>3\*</sup>  and Yu-Chuan Shen<sup>1</sup>

\*Correspondence:

ywchan@gm.pu.edu.tw

<sup>3</sup> College of Computing and Informatics, Providence University, 200, Sec. 7, Taiwan Boulevard, Shalu Dist., Taichung 43301, Taiwan  
Full list of author information is available at the end of the article

## Abstract

With the rapid development of networks and Information technologies, cloud computing is not only becoming popular, the types of cloud services available are also increasing. Through cloud services, users can upload their requirements via the Internet to the cloud environment and receive responses following post-processing, for example, with cloud storage services. Software-Defined Storage (SDS) is a virtualization technology for cloud storage services. SDS uses software to integrate storage resources and to improve the accessibility and usability of storage services. Currently, there are many different open source projects available for SDS development. This work aims to utilize these open source projects to improve the efficiency of integration for hardware and software resources. In other words, in this work, we propose a cloud storage system that integrates various open source SDS software to make cloud storage services more compatible and user friendly. The cloud service systems can also be managed in a more convenient and flexible manner. The experimental results demonstrate the benefits of the proposed system.

**Keywords:** Cloud service, Storage service, Software-defined storage, Automatic distribution

## Introduction

In the last decade, cloud computing has attracted more and more attentions in both industry and academia [1–8]. It deeply changed people's lives due to its inherent advantages, such as on-demand self-service, resource pooling and rapid resource elasticity, etc.. With the services provided by cloud computing, users can upload their requirements via the Internet to a cloud environment and receive responses following post-processing in the cloud environment. Among these services, cloud storage service is one of the important and indispensable services [9–12]. Cloud storage makes data storage a service in which data is outsourced to a cloud server maintained by a cloud provider. With the service, data could be stored remotely into the cloud efficiently and safely. Thus, this service attracts many people, especially enterprises, due to that it brings appealing benefits, e.g., avoidance of capital expenditure on hardware and software, relief of the burden for storage management, etc. [13–15].

Nowadays, many large IT enterprises, such as Google, Microsoft, Amazon and Yahoo provide the service. Although the services could be provided by these large IT enterprises and the services have many advantages, some issues have to be concerned to be widely used by the government and users. For instance, in cloud storage, the data owner does not possess data physically after data is outsourced into the cloud service providers who are not fully trusted. Therefore, government and academic institutions choose to build their clouds by themselves. However, building cloud servers is very expensive due to the equipment cost and the corresponding maintenance cost. Thus, how to reduce the system construction cost and enhance the system's usability and accessibility is the main problem we concern. In this work, we have implemented a cloud system, in which various software-defined storage technologies and the cubic spline interpolation and distribution mechanisms are used together to provide a more easy-to-use, efficient, reliable and user-friendly cloud storage system. The main contributions of this work are summarized as follows:

- We implemented a cloud storage system to integrate various SDS technologies using cubic spline interpolation and distribution mechanisms. The proposed system consisted of three main components, they were the storage service, the file distribution mechanism and the user service, respectively. In addition, since the user's file size cannot be predicted and the received files were not the same with our measured results, we successfully solve this problem by integrating the cubic spline interpolation method.
- In the system architecture, we used open source software to make the system more compatible. In addition, a file was assigned automatically to an appropriate storage location after users uploaded files.
- We designed a user-friendly interface, users could easily upload their files and realized the usage percentages of storage as well as the status of their uploading jobs. Also, the parameters could be set freely to make the system more flexible by managers.

The rest of this paper was organized as follows. In the related work section, we introduced the literature review and related works. In the section of system design and implementation, we presented the system architecture and the corresponding methods. The experimental results were shown in the section of experimental results. Finally, concluding remarks were given.

### **Related work**

During the early development of cloud services, the exact meaning of software-defined service was inconclusive. The concept of "software-defined data center" was first proposed by VMware as software became more important. By employing the concept of virtualization in developing hardware resources as a resource pool, software could be employed to control the arrangement of hardware resources. When using programmable software to control the arrangement of hardware resources, there is no need to think about how to manipulate servers and security or allocate resources. In other words, all the resources function perfectly [16–18]. Cloud computing gave

rise to more possibilities, enabling software-defined services to be different concepts in hardware and software architectures. These concepts have in turn enabled the creation of custom functions and the automation of operations. Accordingly, many research papers and commercial products related to software-defined storage have been proposed.

Yang et al. [19] proposed an integrated storage service. They used the open source software—*OpenStack* [20] to build and manage cloud services, and also used software to integrate storage resources, including Hadoop HDFS, Ceph and Swift on Open Stack to achieve an SDS design. Software users can integrate different storage devices to provide an integrated storage array and to build a virtual storage pool, such that the services provided for users are not limited by the storage devices. Our work primarily follows the concepts in [19], but we improve the system architecture and propose a mechanism to store data efficiently. In addition, we provide a new and more friendly user interface.

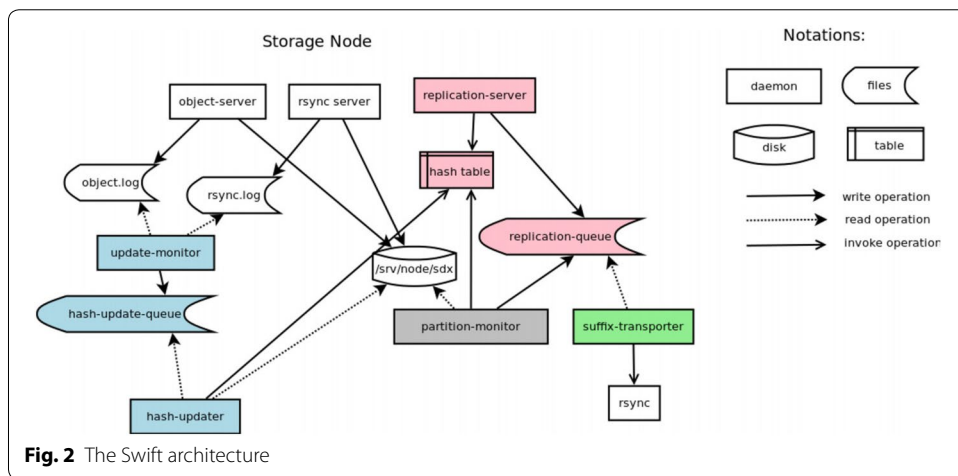
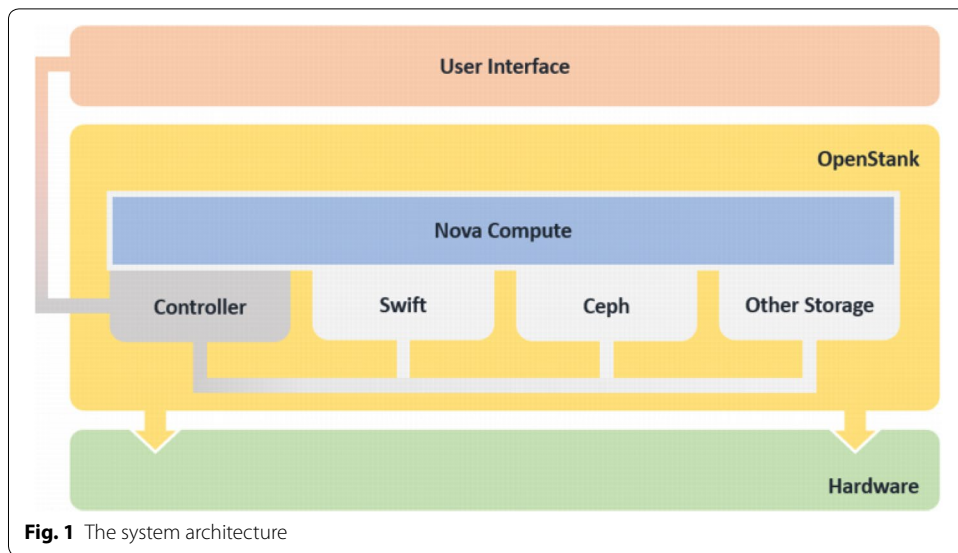
The EMC Virtualization Platform Reinvented (ViPR) [21] is a logical storage system, not a physical storage system. It can integrate EMC storage and third-party storage in a storage pool, and manage them as a single system while retaining the value of the original storage. ViPR can replicate data across different locations and data centers with different storage products, and provides a unified block store, object store, file system and other services. ViPR also provides a unified metadata service and self-service deployment, as well as measurement and monitoring services.

A file system architecture that efficiently organizes data and metadata and enables sharing in addition to exploiting the power of storage virtualization and maintaining simplicity in such a highly complex and virtualized environment was proposed by Ankur Agrawal et al. [22]. Tahani Hussain assessed the performance of an existing enterprise network before and after deploying distributed storage systems [23]. Additionally, simulation of an enterprise network with 680 clients and 54 servers followed by redesigning the system led to improvements in the storage system throughput by 13.9%, a reduction in average response time by 24.4% and a reduction in packet loss rate by 38.3%.

Chengzhang et al. [24] proposed a solution for building a cloud storage service system based on the open-source distributed database. Dejun Wang [25] proposed an efficient cloud storage mode for heterogeneous cloud infrastructures, and validated the model with numerical examples through extensive testing. He also highlighted the differences in a cloud storage system using traditional storage. For example, the demand from the performance point of view, data security, reliability, efficiency and other indicators need to be taken into consideration for cloud storage services, which are services in a wide range of complex network environments designed to meet the demands of large-scale users.

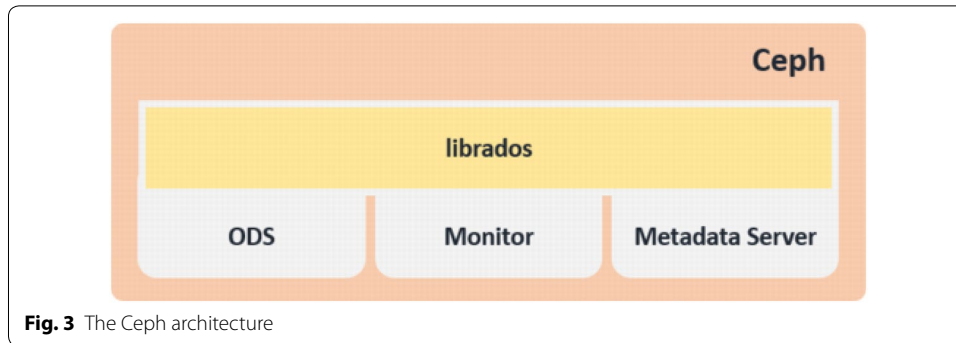
### **System design and implementation**

In this section, we introduce the system architecture and the implementation, which adopts open-source software for better development and maintenance in the future. The integrated heterogeneous storage technologies employed in the system are useful and complete object storage systems. In addition, a graphical user interface is provided so that an administrator can change the parameters to make the system more flexible.



**System design**

The proposed system architecture, as shown in Fig. 1, is divided into three layers. The first layer is the hardware layer, which consists of many computer hardware and network devices. The second layer is the virtualization layer designed with OpenStack, with several components including the compute portion, the network portion and the storage portion. Through the virtualization technology provided by the OpenStack platform, the hardware resources, including the compute, network and storage resources can be fully utilized by the integrated virtual machines (VMs) to constitute our services, including the storage and control services. The storage service consists of many storage systems, including Swift [26], Ceph [27] and other storage systems. In addition, Nova Compute is a component within the OpenStack platform developed to provide on-demand, scalable and self-service access to compute resources, such as VMs, containers and bare metal servers. The architectures of the Swift and Ceph systems are presented in Figs. 2 and 3, respectively.



Swift is a scalable redundant storage system, in which objects and files are written to multiple disks spread throughout servers in the data center. As shown in Fig. 2, the colored icons are the main components of the system, and are divided into four parts:

1. The cyan colored components are in charge of calculating hash in real time.
2. The pink colored components are in charge of indexing the hash of suffix and partition directories, receiving and sending requests to compare the hash of a partition or suffix and generating jobs replicating suffix directories to the replication queue.
3. The gray colored component, which is called the partition-monitor, is in charge of checking whether to move the partition at various intervals.
4. The green colored component, which is called the suffix-transporter, is in charge of monitoring the replication-queue and invoking rsync to sync the suffix directories.

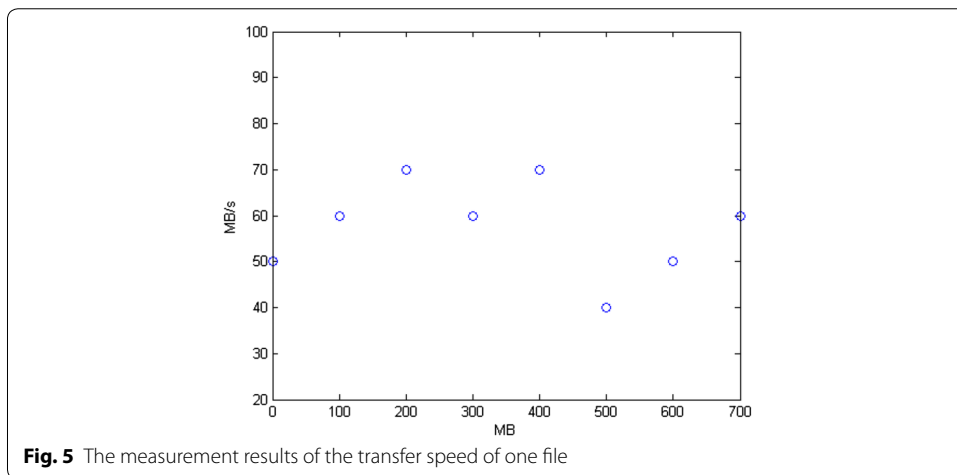
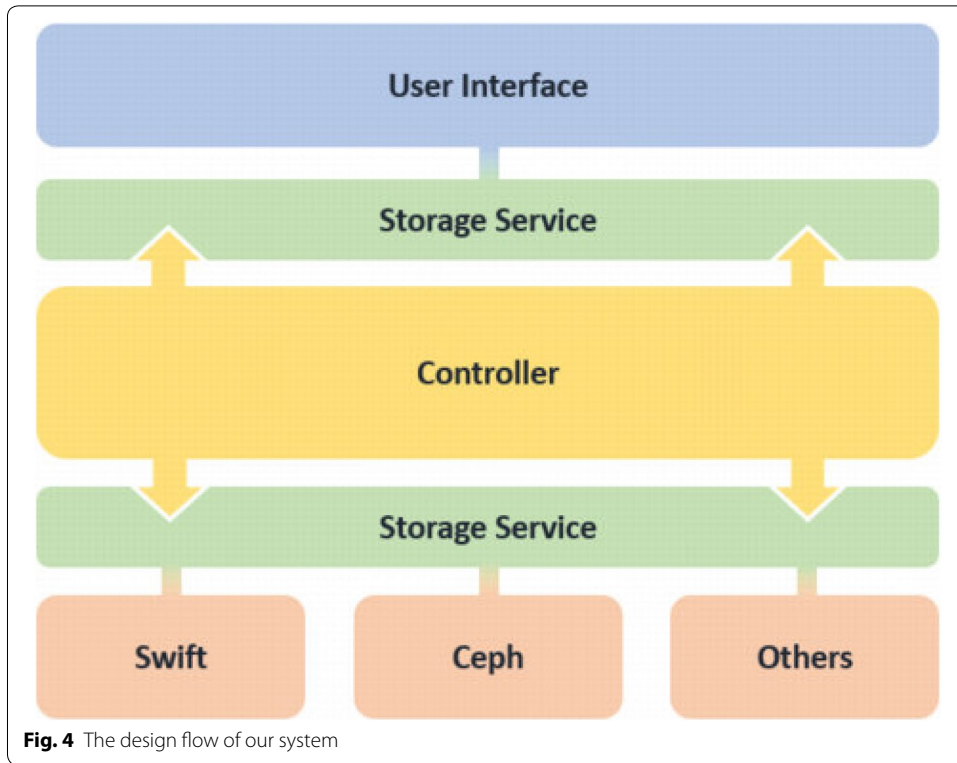
On the other hand, the control service, which is built into the controller node, is responsible for managing the storage services, which are constructed using storage functions. Through the control service and the storage functions, the controller can control the storage devices and resources indirectly. In addition, the controller node has its own distribution mechanism. The mechanism can automatically assign files to the appropriate storage functions after users upload their files. The third layer of the system provides a graphical user interface via a web browser to present our system functions, such that users can easily access the proposed cloud system services. Figure 4 shows the design flow of our system based on the controller architecture.

### System implementation

The implementation of the proposed system consists of three main components, the storage service deployment, the file distribution mechanism and the user services. In the following subsections, each component will be introduced in detail.

#### *The deployment of storage services*

In the first part, we introduce the storage services. We create VMs that form a storage cluster. Then, we use the open source software OpenStack to build and manage the cloud system.



***The mechanism of file distribution***

In the second part, we introduce the mechanism of file distribution. We first use the Cloud he user’s status is summarized Object Storage Benchmark (COSBench) [28] to measure the file transfer speed. COSBench is a benchmark tool for measuring the performance of cloud object storage services. The measured results of our testing are marked on the coordinate diagram, as shown in Fig. 5. In this work, considering that the user’s file size cannot be predicted and the received files will not be the same as our measured results, we need a mechanism to coordinate the interpolation into a linear equation. Based on the promising features studied in the reference works

[29–31], we therefore choose to use the cubic spline interpolation method to solve this problem.

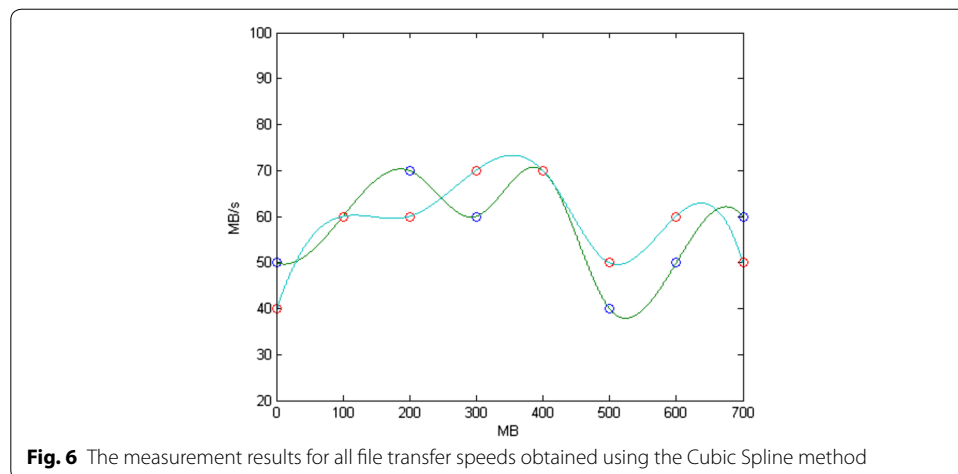
Interpolations using cubic splines have been well studied in [29–31]. In [29], the basis of cubic spline interpolation was introduced. Miao et al. [30] employed the cubic spline method to predict the storage volume of a data center by interpolating the storage volume time series such that an entire time series with the same number as the former series can be reconstructed. In addition, Mastorakis [31] showed that the cubic spline method is well suited for application to the problem of anomaly detection in cloud environments. A cubic spline is a spline constructed of piecewise third-order polynomials that pass through a set of  $m$  control points. The second derivative of each polynomial is commonly set to zero at the endpoints, since this provides a boundary condition that completes the system of  $m-2$  equations. This produces a so-called “natural” cubic spline and leads to a simple tridiagonal system that can be solved easily to give the coefficients of the polynomials.

By using the Cubic Spline, we obtain a new coordinates diagram and plot the interpolation figures for Swift and Ceph, as shown in Fig. 6. This can be used as the decision criteria when processing files. Certainly, this will not be the only method in our mechanism. We also consider the use of storage capacity for the environmental effect. Similar to the previous method of measurement, we perform measurements for storage environments with different capacities.

In addition, we propose Eq. (1) to obtain the transfer speed of the storage service, which is used to determine which of the storage services is better.

$$f_K(S) = \alpha f_i(S) + \beta f_c(S). \tag{1}$$

- $f_i(S)$  represents the transfer speed obtained in the transfer speed experiment when the file size is  $S$ .
- $f_c(S)$  represents the transfer speed obtained in the storage capacity experiment when the file size is  $S$ .
- $\alpha$  and  $\beta$  are the weights, with default values of 0.5. The sum of these two weights equals one.



- $f_K(S)$  represents the resulting transfer speed of the storage service, which is used to compare the performance of the storage services.

For example, we perform an experiment to determine the transfer speed for Swift and Ceph, and consequently obtain two functions,  $f_{ts}(S)$  and  $f_{tc}(S)$ . Another experiment is performed to test the storage capacity of Swift and Ceph to obtain two functions,  $f_{cs}(S)$  and  $f_{cc}(S)$ . The resulting functions  $f_{Swift}(S)$  and  $f_{Ceph}(S)$  are listed in Eqs. (2) and (3), respectively.

$$f_{Swift}(S) = \alpha f_{ts}(S) + \beta f_{cs}(S). \tag{2}$$

$$f_{Ceph}(S) = \alpha f_{tc}(S) + \beta f_{cc}(S). \tag{3}$$

After calculation, we obtain two values  $f_{Swift}(S)$  and  $f_{Ceph}(S)$ . The following mechanism compares these two values to determine which storage technology is better. If these two values are equal, we add a condition that depends on storage usage. The mechanism will choose the system with lower usage.

$$f = \begin{cases} f_{Swift}(S), & (f_{Swift}(S) > f_{Ceph}(S)) \text{ or} \\ & (f_{Swift}(S) = f_{Ceph}(S) \ \& \\ & Usage_{Swift} > Usage_{Ceph}) \\ f_{Ceph}(S), & (f_{Swift}(S) < f_{Ceph}(S)) \text{ or} \\ & (f_{Swift}(S) = f_{Ceph}(S) \ \& \\ & Usage_{Swift} < Usage_{Ceph}) \end{cases}$$

Our mechanism is scalable. We can add any condition that may affect the transfer speed. For example, as shown in Eq. 4, the function  $f_n(S)$  is another consideration for time consumption with a weight of  $\gamma$  and the sum of the three weights  $\alpha$ ,  $\beta$ , and  $\gamma$  must be one.

$$f_K(S) = \alpha f_t(S) + \beta f_c(S) + \gamma f_n(S). \tag{4}$$

### The experimental results

In this section, we show the experimental results and the system implementation performance. We first perform efficacy experiments to demonstrate the benefits of our system infrastructure. Next, we measure the speed of each storage object. This measurement is the basis of the file distribution mechanism. Finally, we show the user interface for our system.

#### Setup of the experimental environment

In the setup for the experimental environment, we use OpenStack to build our cloud platform, which is then used to create and manage the distributed storage system. In the system, we adopt two heterogeneous storage technologies, namely Ceph and Swift. We use Ceph to build a storage system that consists of four VMs with dual core CPUs, 4 GB of memory and a total of 160 GB of storage space. The VM named ceph01 is MON and OSD, and the others are OSD . These VMs form a Ceph cluster. On the other hand, we use Swift to build a storage system consisting of four VMs, which include one proxy server and four storage nodes, with the same specifications of dual core CPUs, 4 GB of memory, and a total of 160 GB of storage space. Tables 1, 2, and 3 sequentially present the specifications for the software, hardware, and storage environments.



**Table 1 Hardware specifications**

Host name	CPU	Memory (GB)	Disk (GB)	OS
Controller	16 cores	48	100	Ubuntu 14.04
Compute01	24 cores	48	800	Ubuntu 14.04
Compute02	24 cores	48	800	Ubuntu 14.04

**Table 2 Storage environment specifications**

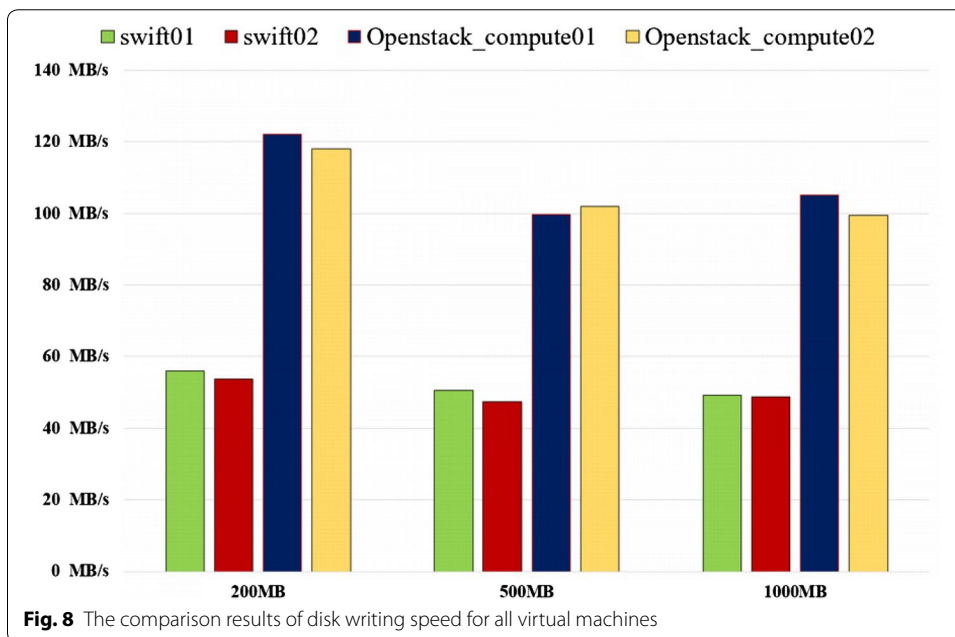
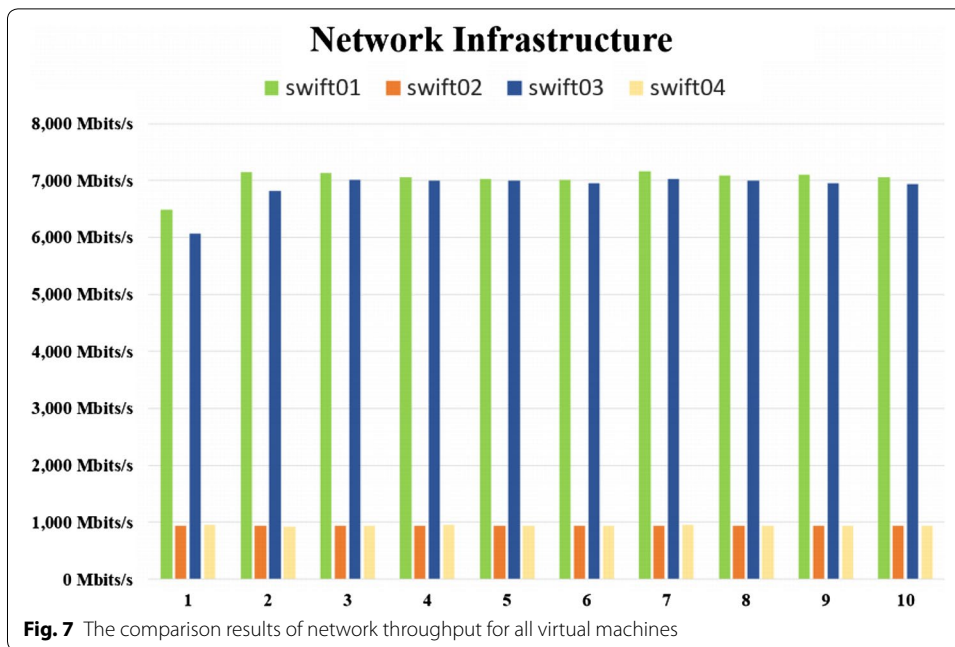
Host name	CPU	Memory (GB)	Disk (GB)	OS
Controller	4 cores	8	40	Ubuntu 14.04
Ceph01	2 cores	8	40	Ubuntu 14.04
Ceph02	2 cores	4	40	Ubuntu 14.04
Ceph03	2 cores	4	40	Ubuntu 14.04
Ceph04	2 cores	4	40	Ubuntu 14.04
Swift01	2 cores	4	40	Ubuntu 14.04
Swift02	2 cores	4	40	Ubuntu 14.04
Swift03	2 cores	4	40	Ubuntu 14.04
Swift04	2 cores	4	40	Ubuntu 14.04

**Table 3 Software specifications**

Software	Version
OpenStack	Juno
Ceph	Hammer v0.94
Swift	2.1.0

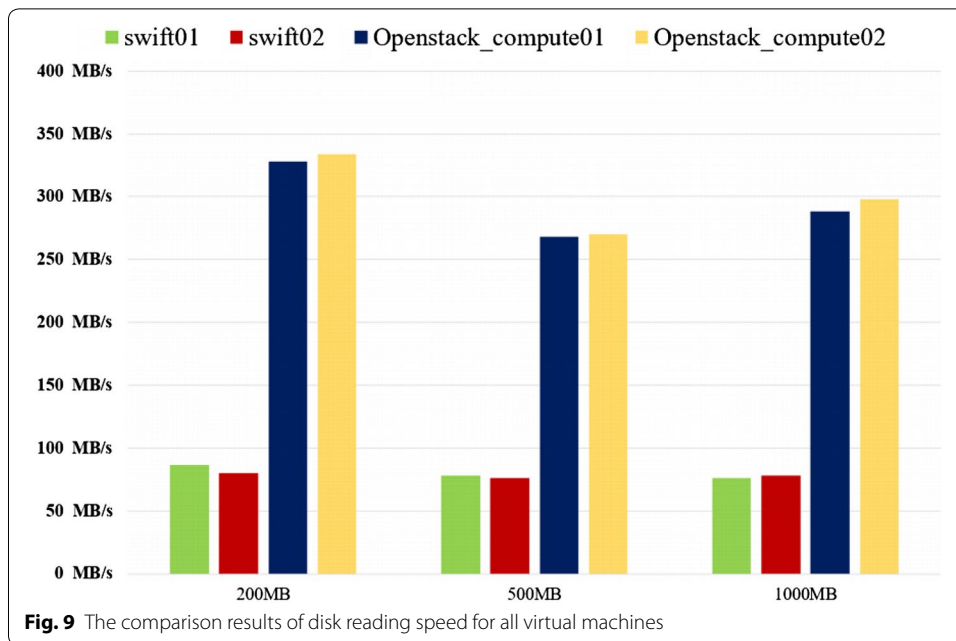
**Performance evaluations of our system**

To evaluate the performance of our system, two metrics are used, specifically network throughput and disk writing speed. In this experiment, we first install four VMs as the experimental nodes in the OpenStack environment. The four VMs are called swift01, swift02, swift03 and swift04, respectively. Since network throughput is a key factor for measuring cluster performance, we use a client-server connection to measure the TCP and UDP bandwidths. The results are illustrated in Fig. 7. In the resulting histogram, the horizontal axis represents the number of tests and the vertical axis represents the transmission bandwidth. As depicted in Fig. 7, the VMs are divided into group A and group B. Group A contains Swift01 and Swift03 VMs while group B consists of Swift02 and Swift04 VMs. The experimental results show that the bandwidth for group A is almost 7000 Mbits/s, while the bandwidth for group B is only about 900 Mbits/s. The large difference in the achieved bandwidth between the two groups is because they are deployed on different physical machines. The VMs in group A are used in the compute01 machine while those in group B are used in the compute02 machine. The results indicate that when the VMs communicate between the two physical machines, they communicate through the physical network. On the contrary, when the VMs communicate with each other in the same physical node, they communicate through the virtual network.



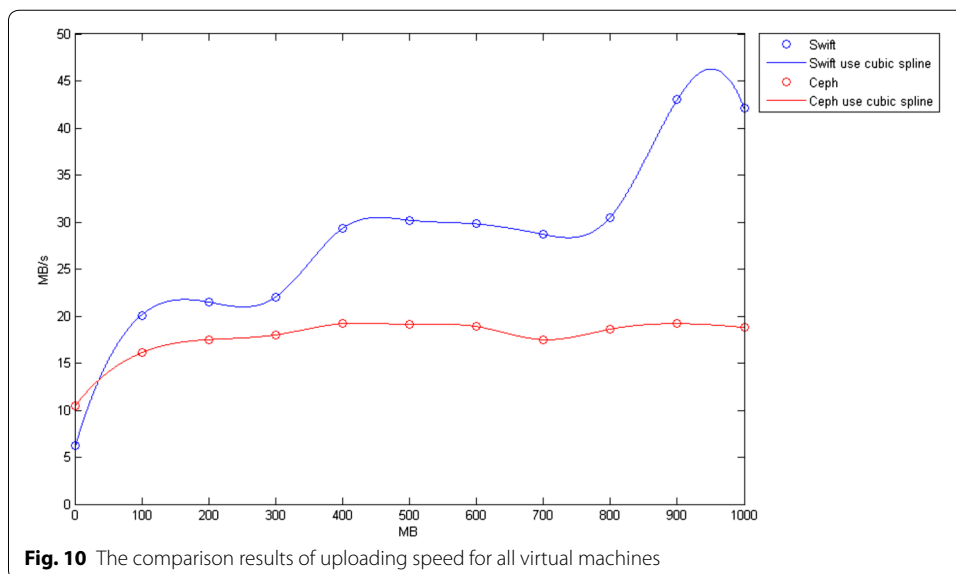
Next, we will discuss the comparison results with respect to the metric of the disk writing speed, which is a key factor for system performance. In this experiment, we use the Linux command `dd`, which is mainly employed to convert and copy files and to measure the disk writing speed. The results are illustrated in Figs. 8 and 9.

According to the previous results from measuring the network bandwidth, if VMs are deployed on the same host, their bandwidths are almost the same. Thus, we select `swift01`, `swift02`, `OpenStack compute01` and `OpenStack compute02` for comparison of



their disk writing and reading speeds. The results show that the VMs cannot take full advantage of the reading and writing resources and therefore require deployment of the storage system. These I/O tests can be used to debug and improve bottlenecks when problems are encountered. In addition, the experimental results for disk reading and writing speed help us decide on the number of VMs deployed on the physical machine and understand how best to deploy the storage cluster.

Figure 10 shows the comparison results for the upload speed in the Ceph and Swift storage clusters. In the figure, the blue hollow circle represents the upload measurements in the Swift storage cluster while the red hollow circle represents the corresponding

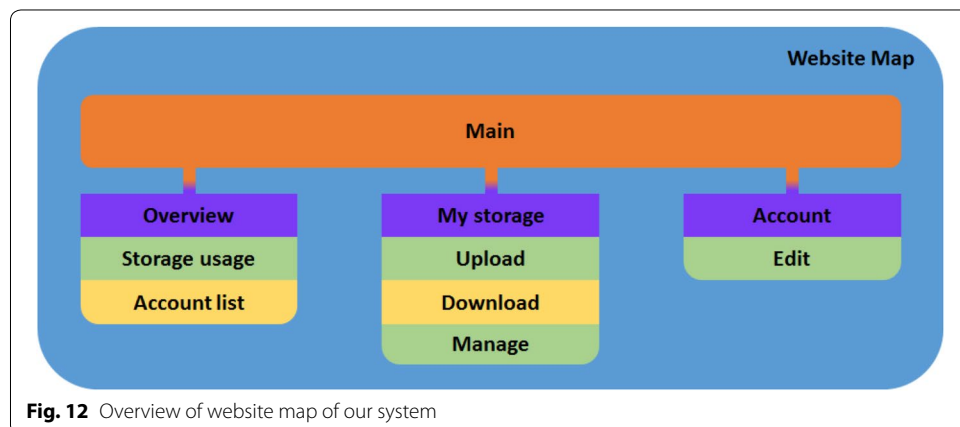
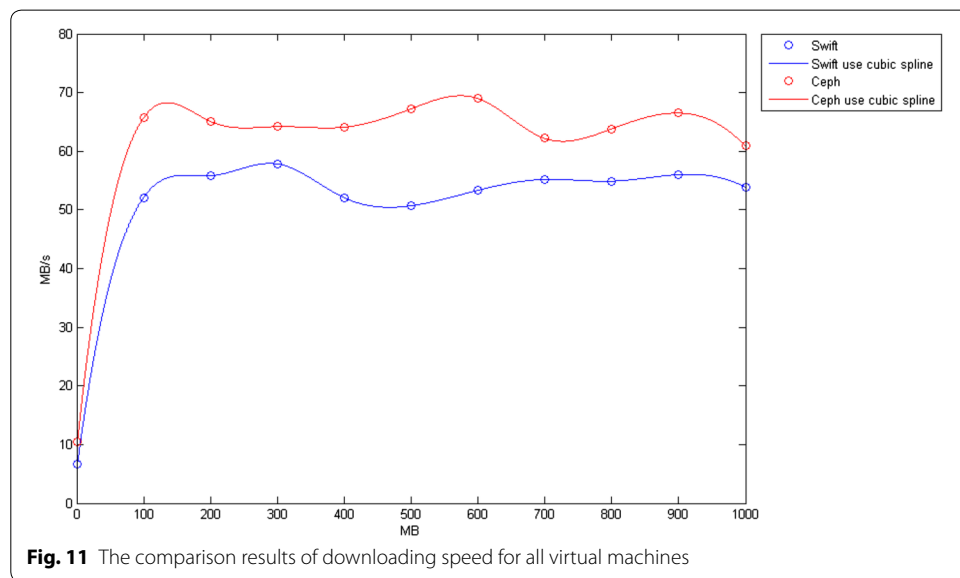


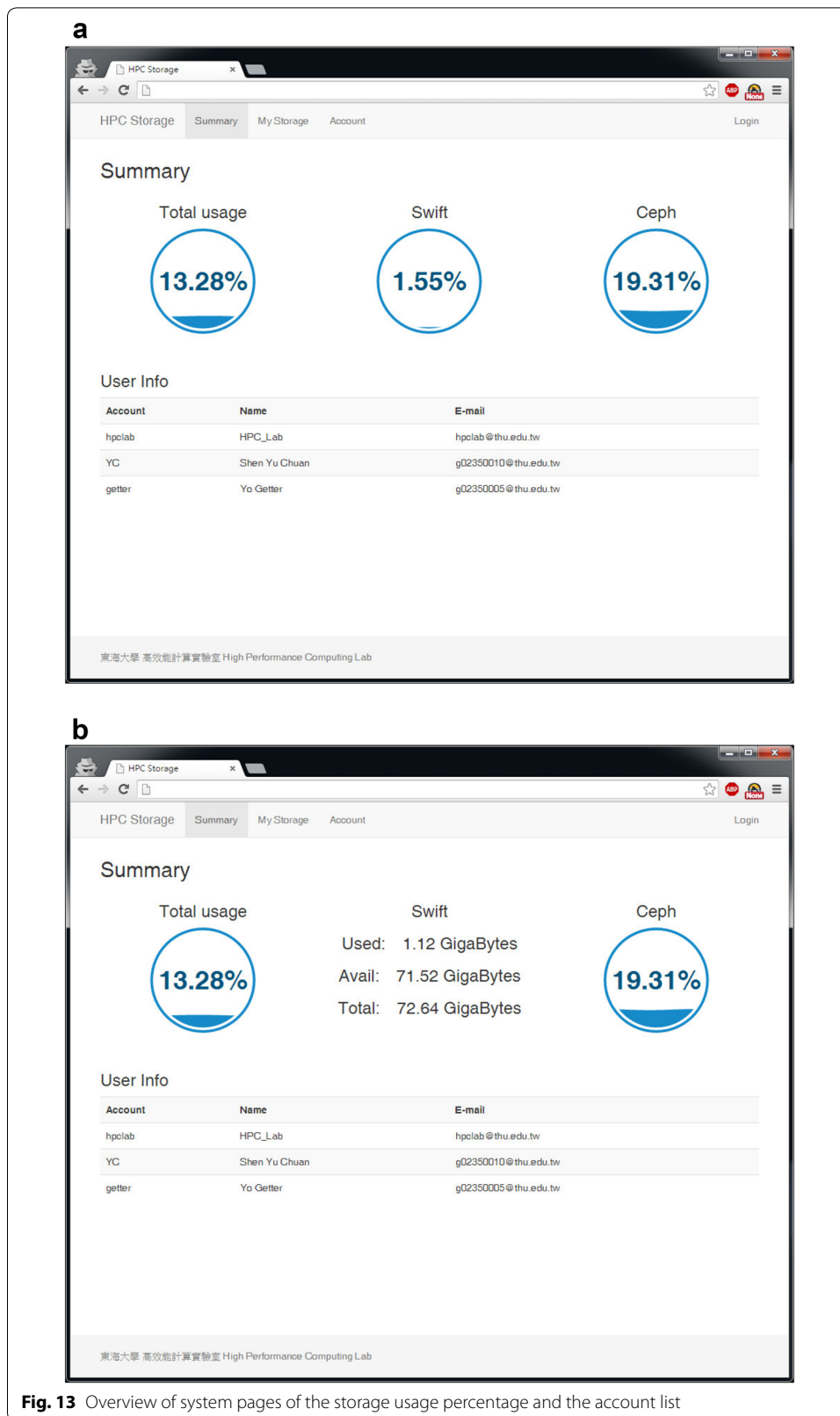
values in the Ceph storage cluster. In addition, we apply cubic spline to obtain continuous curves with respect to the Ceph and Swift clusters. From the figure, we see that the upload speed in the Swift cluster stabilizes at about 20-30 MB/s, with a significant increase when the file size is larger than 800 MB. On the contrary, in the Ceph cluster, the upload speed is almost 15 MB/s. These two curves intersect once when the file size is about 50 MB. Thus, the upload speed for Ceph is faster than that of Swift when the file size is less than 50 MB and is slower when the file size is larger than 50 MB.

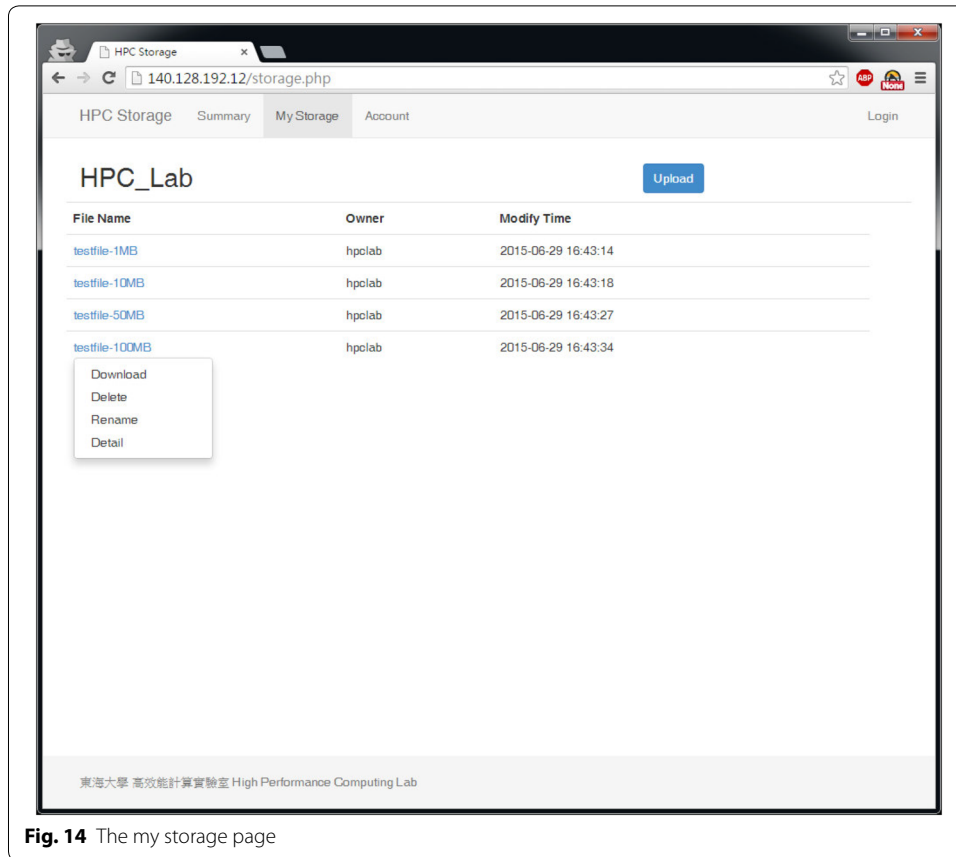
Figure 11 shows the experimental results with respect to the download speed in the Ceph and Swift storage clusters. The results show that the download speed for the Ceph cluster is faster than that of the Swift cluster.

### User interface design

In this subsection, we will introduce the design of the user interface in our system. An overview of the website map is shown in Fig. 12. The user interface in our system mainly consists of three parts: the *system overview* page (as shown in Fig. 13), the *my storage* page (as shown in Fig. 14) and the *account* page (as shown in Fig. 16). In







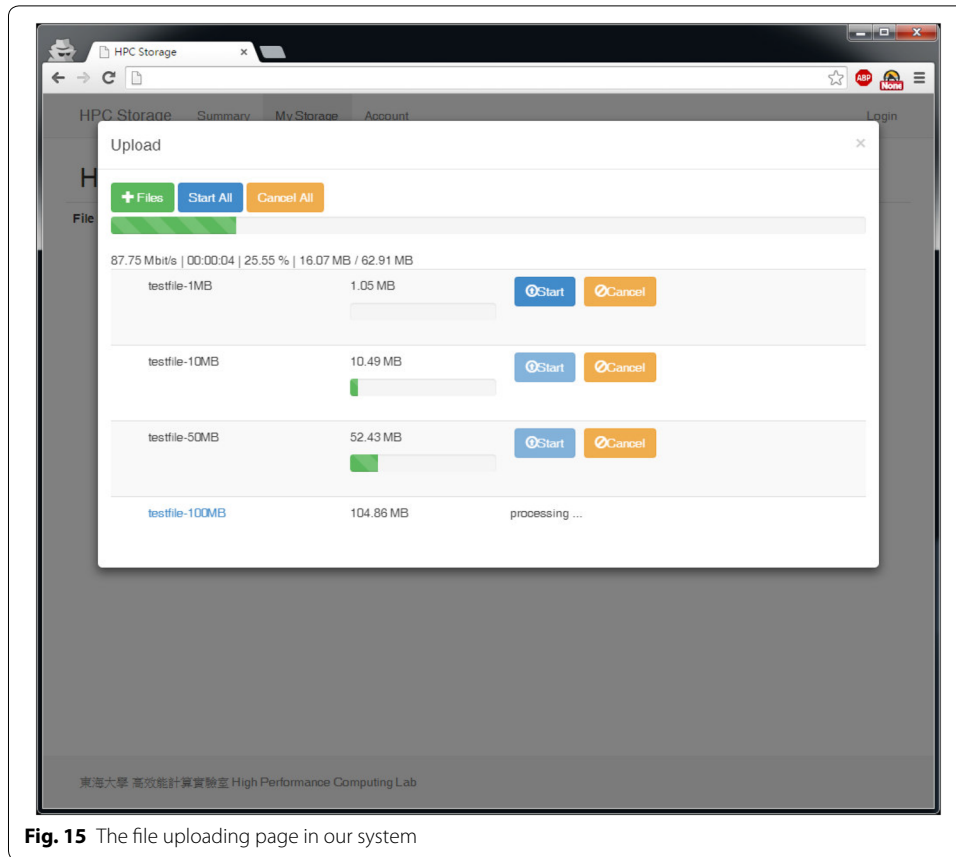
**Fig. 14** The my storage page

the *system overview* page, the user’s status is summarized, and users can review their storage usage and account information. The *my storage* page is the main part of the user interface in the system. It consists of basic operations, such as upload, download, remove and modify operations. The *account* page shows the user information. Users can modify their personal information via this page.

As shown in Fig. 13, there are two panels in the *system overview* page. The two panels are used to show the storage usage percentages and the account list. We use three small liquid fill gauges to display the percentages for the total usage, the Swift usage and the Ceph usage. More detailed information is shown when the mouse moves over the liquid fill gauge, as shown in Fig. 13. In addition, there is a table that shows information for all the accounts when the user logs into the administrator mode.

The *my storage* page is the major operating part of our system. When the page is loaded, a file list is shown in the middle of the page and a drop down menu pops up when the right mouse button clicks a file name, as shown in Fig. 14. The drop down menu has four functions: download, delete, rename and detailed information. All functions related to the storage operations are displayed in this page.

We use AJAX, JQuery and the bootstrap framework to implement the uploading process. The web page pops up a window upon left clicking the upload button, as shown in Fig. 15. The figures show four files in the list. One file is ready to upload, two are uploading and the last is being processed. The upload function allows multiple files to be uploaded at the

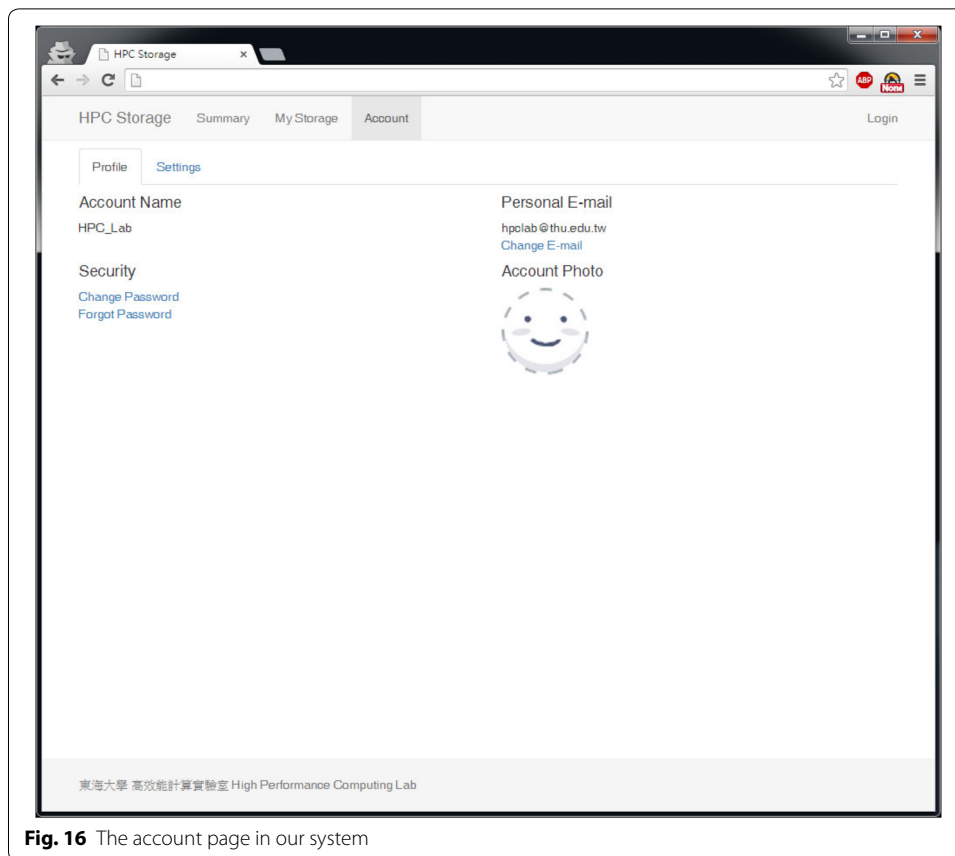


**Fig. 15** The file uploading page in our system

same time. The files have individual upload progress bars and the total upload progress bar is shown near the top of the page. The total progress bar shows detailed upload information including the transfer speed, the remaining time and the completed percentage. The upload functions have the following advantages:

- Friendly user interface: a visualization of the upload progress is provided. This makes it easy for users to monitor and control their uploading jobs.
- Supports the upload of multiple files: users can upload multiple files at the same time.
- Background processing: users can upload their files in the background while accessing other functions simultaneously in the *my storage* page.

The last part is the *accounting* page, as shown in Fig. 16. The *accounting* page has two main functions, which are the viewing and the editing. Through these functions, detailed accounting information can be viewed and edited. The design of all the pages in the system follows the design concept of RWD. Whatever the device used, the bootstrap framework displays the appropriate web layout according to the screen size.



**Fig. 16** The account page in our system

## Conclusion

In this work, we implemented a cloud storage system by integrating the open source storage software to provide a software-defined storage service. In the system, we used the distributed cloud architecture to provide high reliable and scalable cloud services which integrate several software storage technologies. In addition, we provided an user interface with high usability to make the proposed system more user friendly. In the future, we plan to build a larger system with more VMs and integrating more heterogeneous storage technologies.

### Authors' contributions

C-TY conceptualized the study and proposed the system design. S-TC implemented the system and wrote the manuscript. Y-WC wrote and revised the manuscript. Y-CS performed the experiments. All authors read and approved the final manuscript.

### Author details

<sup>1</sup> Department of Computer Science, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, 40704 Taichung, Taiwan. <sup>2</sup> College of Future, Bachelor Program in Interdisciplinary Studies, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan. <sup>3</sup> College of Computing and Informatics, Providence University, 200, Sec. 7, Taiwan Boulevard, Shalu Dist., Taichung 43301, Taiwan.

### Acknowledgements

This work was supported in part by the Ministry of Science and Technology, Taiwan ROC, under Grant Numbers 106-2622-E-029-002-CC3, 107-2221-E-029-008, and 107-2218-E-029-003.

### Competing interests

The authors declare that they have no competing interests.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Received: 13 October 2018 Accepted: 19 March 2019

Published online: 02 April 2019

**References**

1. Zhou Z, Ota K, Dong M, Xu C (2017) Energy-efficient matching for resource allocation in d2d enabled cellular networks. *IEEE Trans Vehicul Technol* 66(6):5256–5268
2. Xu C, Gao C, Zhou Z, Chang Z, Jia Y (2017) Social network-based content delivery in device-to-device underlay cellular networks using matching theory. *IEEE Access* 5:924–937
3. Mo Y, Peng M, Xiang H, Sun Y, Ji X (2017) Resource allocation in cloud radio access networks with device-to-device communications. *IEEE Access* 5:1250–1262
4. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: *Proceedings of the 2008 grid computing environments workshop: 2008*; Austin, USA, pp 1–10
5. Nurmi D, Wolski R, Grzegorzczak C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2009) The eucalyptus open-source cloud-computing system. In: *Proceedings of the 2009 9th IEEE/ACM international symposium on cluster computing and the grid: 2009*; Shanghai, China, pp 124–131
6. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Comput* 8:14–23
7. Buyya R, Yeo CS, Venugopal S (2008) Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In: *Proceedings of the 10th IEEE international conference on high performance computing and communications: 2008*; Dalian, China, pp 5–13
8. Kim H-W, Jeong Y-S (2018) Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain. *Human-centric Comput Inform Sci* 8(1):11
9. Vernik G, Shulman-Peleg A, Dippl S, Formisano C, Jaeger MC, Kolodner EK, Villari M (2013) Data on-boarding in federated storage clouds. In: *Proceedings of the 2013 IEEE sixth international conference on cloud computing: 2013*; Santa Clara, USA, pp 244–251
10. Kolodner EK, Tal S, Kyriazis D, Naor D, Allalouf M, Bonelli L, Brand P, Eckert A, Elmroth E, Gogouvitis SV, Harnik D, Hernandez F, Jaeger MC, Lakew EB, Lopez JM, Lorenz M, Messina A, Shulman-Peleg A, Talyansky R, Voulodimos A, Wolfsthal Y (2011) A cloud environment for data-intensive storage services. In: *Proceedings of the 2011 IEEE third international conference on cloud computing technology and science: 29 Nov.-1 Dec. 2011*; Athens, Greece, pp 357–366
11. Rhea S, Wells C, Eaton P, Geels D, Zhao B, Weatherspoon H, Kubiatowicz J (2001) Maintenance-free global data storage. *IEEE Internet Comput* 5:40–49
12. Mesnier M, Ganger GR, Riedel E (2003) Object-based storage. *IEEE Commun Mag* 41:84–90
13. Mesbahi MR, Rahmani AM, Hosseinzadeh M (2018) Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Comput Inform Sci* 8(1):20
14. Zhang Y, Xu C, Liang X, Li H, Mu Y, Zhang X (2017) Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. *IEEE Trans Inform Forensic Sec* 12(3):676–688
15. Ren Z, Wang L, Wang Q, Xu M (2018) Dynamic proofs of retrievability for coded cloud storage systems. *IEEE Trans Serv Comput* 11(4):685–698
16. Li Y, Feng D, Shi Z (2013) An effective cache algorithm for heterogeneous storage systems. *Sci World J* 2013:693845
17. Lin W, Wu W, Wang JZ (2016) A heuristic task scheduling algorithm for heterogeneous virtual clusters. *Sci Program* 2016:7040276
18. Callegati F, Cerroni W, Contoli C (2016) Virtual networking performance in openstack platform for network function virtualization. *J Elec Comput Eng* 2016:266–267
19. Yang C-T, Lien W-H, Shen Y-C, Leu F-Y (2015) Implementation of a software-defined storage service with heterogeneous storage technologies. In: *Proceedings of the 2015 IEEE 29th international conference on advanced information networking and applications workshops (WAINA): 24–27 March 2015*, pp 102–107
20. OpenStack. <https://www.openstack.org/> (2015)
21. EMC ViPR. <http://www.emc.com/vipr> (2015)
22. Agrrawa A, Shankar R, Akarsh S, Madan P (2012) File system aware storage virtualization management. In: *Proceedings of the 2012 IEEE international conference on cloud computing in emerging markets (CCEM): 11–12 Oct. 2012*; Bangalore, India, pp 1–11
23. Hussain T, Marimuthu PN, Habib SJ (2013) Managing distributed storage system through network redesign. In: *Proceedings of the 2013 15th Asia-Pacific network operations and management symposium (APNOMS): 25–27 Sept. 2013*; Hiroshima, Japan, pp 1–6
24. Peng C, Jiang Z (2011) Building a cloud storage service system. *Procedia Environ Sci* 10:691–696
25. Wang D (2011) An efficient cloud storage model for heterogeneous cloud infrastructures. *Procedia Eng* 23:510–515
26. OpenStack Swift. <https://wiki.openstack.org/wiki/Swift> (2015)
27. Weil SA, Brandt SA, Miller EL, Long DD, Maltzahn C (2006) Ceph: A scalable, high-performance distributed file system. In: *Proceedings of the 7th symposium on operating systems design and implementation: 6–8 November 2006*; Seattle, USA, pp 307–320
28. Zheng Q, Chen H, Wang Y, Zhang J, Duan J (2013) Cosbench: Cloud object storage benchmark. In: *Proceedings of the 4th ACM/SPEC international conference on performance engineering (ICPE 2013): 21–24 April 2013*; Prague, Czech Republic, pp 199–210
29. Knott GD (2012) *Interpolating Cubic Splines*. Springer, Berlin
30. Miao B, Dou C, Jin X (2016) Main trend extraction based on irregular sampling estimation and its application in storage volume of internet data center. *Comput Intell Neurosci* 2016:1–12
31. Mastorakis G (2015) *Resource management of mobile cloud computing networks and environments*. IGI Global, Hershey