2015 29th International Conference on Advanced Information Networking and Applications Workshops

Implementation of a Software-Defined Storage Service with Heterogeneous Storage Technologies

Chao-Tung Yang, Wei-Hsiang Lien, Yu-Chuan Shen, Fang-Yi Leu

Department of Computer Science, Tunghai University, Taichung City, 40704 Taiwan ROC
Email: ctyang@thu.edu.tw; andy07010@yahoo.com.tw; s1991829@gmail.com; leufy@thu.edu.tw;

***Abstract*—SDS becomes more and more popular, and several companies have announced their product. But the generic standard still has not appeared; most products are only appropriate for their devices and SDS just can integrate a few storages. In this thesis, we will use the OpenStack to build and manage the cloud service, and use software to integrate storage resources include Hadoop HDFS, Ceph and Swift on OpenStack to achieve the concept of SDS. The software used can integrate different storage devices to provide an integrated storage array and build a virtual storage pool; so that users do not feel restrained by the storage devices. Our software platform also provides a web interface for managers to arrange the storage space, administrate users and security settings. For allocation of the storage resources, we make a policy and assign the specific storage array to the machine that acquires the resource according to the policy.**

***Keywords*-Cloud service, Virtualization, Software-Defined Storage, Hadoop, HDFS, Ceph**

I. Introduction

Cloud computing, one of today's hottest topics, is being developed very fast. There are many large companies such as Google, Amazon, and Yahoo offering cloud services to millions of users at the same time. A lot of people including programmers in the enterprise, government, and research centers, or even some common people, want to build cloud computing environments. But the operating cost of a cloud may be very expensive; the cost includes acquirement and maintenance of hardware facilities, and salaries paid to operating engineers and so on. So how to reduce the cost is a big issue.

A. Motivation

Software-Defined Storage (SDS) is a kind of virtualization which integrates the storage resource and different storage device by software to increase the usability and activity; the users can adjust SDS according to their requirement to achieve the optimal performance. In recent years, SDS is an increasingly popular concept, and several companies have announced related products. But the generic standard still has not formed; hence, most of its products are only appropriate for some devices.

II. Background Review and Related Work

A. Swift

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used.

B. HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system designed to hold very large amounts of data (terabytes or even petabytes), and to provide high-throughput access to this information. Files are stored in a redundant fashion across multiple machines to ensure durability to failure and high availability to parallel applications [1]. HDFS has many similarities with other distributed file systems, but is different in several respects. One noticeable difference is HDFS's write-once-read-many model that relaxes concurrency control requirements, simplifies data coherency, and enables high-throughput access [2], [10]–[12].

C. Ceph

Ceph [7]–[9] is a software storage platform designed to present object, block, and file storage from a single distributed computer cluster. Ceph is a distributed storage designed to provide excellent performance, reliability and scalability. Ceph was made possible by a global community of enthusiastic storage engineers and researchers. It is open source and freely-available. Ceph software runs on commodity hardware. The system is designed to be both self-healing and self-managing and strives to cut both administrator and budget costs.

978-1-4799-1775-4/15 $31.00 © 2015 IEEE
DOI 10.1109/WAINA.2015.50

III. System Design and Implementation

A. System Architecture

The main goal of this thesis is to build a cloud service platform integrating with several heterogeneous storage technologies. Additionally, we will offer a simple GUI for users to manage the cloud service. In this chapter, we will overview the whole system.

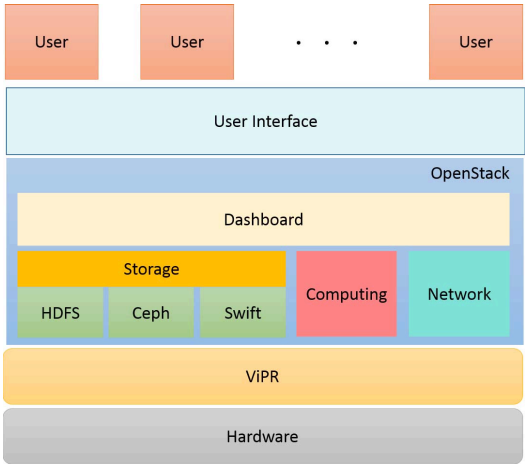

Figure 1. System Architecture

The system architecture consists of some components. The first part is a server node built by OpenStack. And we also used OpenStack to build some VMs to be the storage nodes. Second, we used three kinds of different storage technology to build the storage of OpenStack. And we used software ViPR to control the different storage devices and manage storage spaces for the storage of OpenStack. ViPR will display the storage with a simple appearance and it can be used to integrate different storage technologies. The system architecture is shown in Figure 1.

B. System Implementation

- Architecture of Ceph: there are three daemons in the Ceph side, including OSD, MDS and MON. The OSD is responsible for saving data as an object; it used an independent partition, and it will partition to a xfs type. It at least has one node, and the capacity of Ceph is defined by the Equation(3.1), where i means the number of osd from 0 to n, and n means the last number of OSD. The function shows the total capacity of OSD as the sum of all OSDs.

$$CephSize = \sum_{i=0}^{n} partitionsize(OSD)_i, n \geq 1 \quad (1)$$

  The second is MDS, responsible for storing the data structure of the data index, and writing it into the OSD. It at least has one node, and it uses a redundant mechanism. The last is MON which is responsible to monitor the running status of OSD and MDS, and according to the situation to adjust them. It at least has one node, and it uses a redundant mechanism too. Every OSD has an independent partition; and the three daemons will not conflict with one another. They can run on a same node.
- Architecture of HDFS: the HDFS is combined with the master node and the slave node which is called NameNode and DataNode. And we built a HDFS architecture consisting of one NameNode and two DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from clients of the file system. The DataNodes also performs block creation, deletion, and replication upon instruction from the NameNode.
- OpenStack: OpenStack was used to build our cloud; it is open source software to manager cloud. It was built on a VM with Ubuntu OS, and then some VMs were created to form a storage system cluster. The overview of the system is shown in Figure 2.

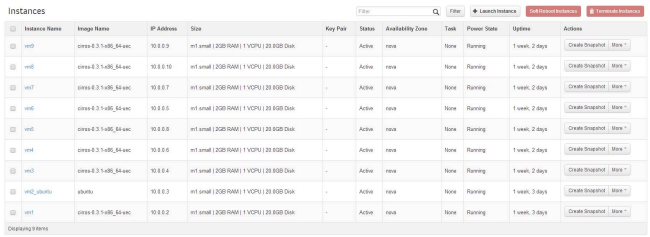

Figure 2. VM Instances

- EMC ViPR: traditional data center is just a set of some servers, storage, network and security and etc. In the traditional data center, to build a new application usually takes several weeks. But the process can be simplified by the data center driven by completely dynamic software, making it like building a new VM. If we can design a series of policies that can be implemented on a variety of new applications to automatically support infrastructure services included in a container, i.e., a virtual data center, then, in a few minutes or a few seconds, we can complete the deployment of a new application.
  EMC Virtualization Platform Reinvented (ViPR) is a logical storage system, not a physical storage. It can integrate EMC storage and third-party storage in a storage pool, and manage it as a single system, but still save the worth of original storage. ViPR can replicate data across several different place and data center with different store product, and it provides a unified block store, object store and file system and other services.

At the same time, ViPR provides a unified metadata service and self-service deployment, measurement and monitoring services. In addition to this, ViPR also applies to multi-tenant environments. EMC proposed a model about software-defined storage, and described the relation with OpenStack. The architecture is shown in Figure 3.

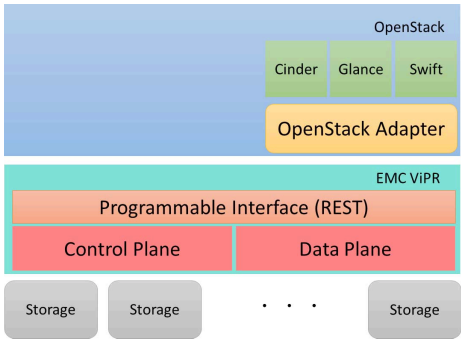

Figure 3. ViPR with OpenStack Architecture

ViPR run on three to five virtual server machines. It consists of the control plane and data plane; the first part realizes automatic store, and the second part provides data service by building on the first part. ViPR provides some APIs to let users to use services and manage storage, as shown in Figure 4.

One part of ViPR is the control plane. It can manage all applications of storage array, such as data mining, storage resource searching, and capacity counting and reporting. The controller controls the application of virtual storage array, and manages the storage resource. In order to achieve the goal, ViPR virtualizes the control path of physical storage, and runs the function with it. By the virtual plane, people can manage the storage pool and split it to different virtual storage arrays with policies, just like virtualization of the server. In addition to the controller and data service, ViPR provides the open RESTful API. The developers can develop new services without the limit of the hardware. Through these APIs, people can access data, and add, delete, modify, monitor and measure the logical storage resources.

ViPR is built by the scale-out architecture; if it is built with at least three clusters, the architecture will provide the ability of high availability, load balance and system upgrade and so on. It provides RESTfulAPI, GUI(console), CLI and SDK to let user control it with high flexibility as shown in Figure 5. And ViPR can provide the automatic of disaster recovery. Through the continuous remote replication technology, it can provide the ability of data replication continuously to successfully achieve the goal of disaster recovery.

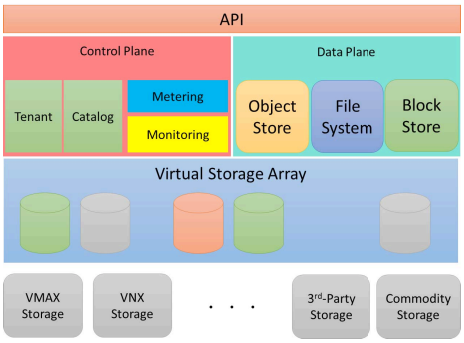

Figure 4. ViPR System Architecture

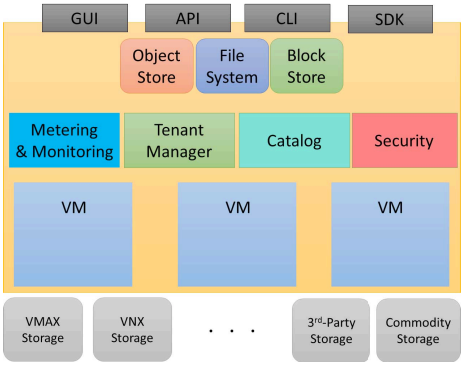

Figure 5. ViPR Cluster

IV. Experimental Results

A. Experimental Environment

We used OpenStack to build our cloud platform, which then was used to create and manage the distributed storage system. In the system we integrated three heterogeneous storage technologies. And we built the storage system by some VMs, in which Ceph was constructed by three VMs with specifications of 1 core CPU, 2GB memory, and a total of 60GB storage space. The first node was MON and OSD, the second was MDS and OSD, the third was OSD. These were components in the Ceph cluster. And the HDFS part was constructed by three virtual machines including one NameNode and two DataNodes with specifications of 1 core CPU, 2GB memory, and total of 40GB storage space.

Table I
Software Specification

Software	Version
OpenStack	Havana
Ceph	V0.72 EMPEROR
HDFS	2.2.0
Swift	
GlusterFS	3.5

B. Performance of Data Transport

In the first part of our experiment, we compared the three different kinds of storage technology, including Ceph, Hadoop HDFS and GlusterFS. The reason to choose them is that they are famous storage technology in the popular distributed storage systems today.

In this experiment, we built three storage systems: Hadoop, Ceph and GlusterFS and recorded the data of the performance of uploading and downloading files on the three systems. We used three cases to upload and download a single file. The first case was for files with the size of 1MB to 16GB. The second is tests of small sized files of 1MB to 100MB, with 16GB total file size. And the last one is files100MB to 1000MB, with 16GB total file size.

The result for the experiment to upload a single file from 1MB to 16GB from the client node to the server node is shown in Figure 6. We observe that when the file size is smaller than 512MB, the performance of the three storage systems are similar. But when the file size is larger than 512MB, the performance of Hadoop is better than GlusterFS and Ceph. According to this result, we can say the ability of Hadoop is better than GlusterFS and Ceph when uploading large files.

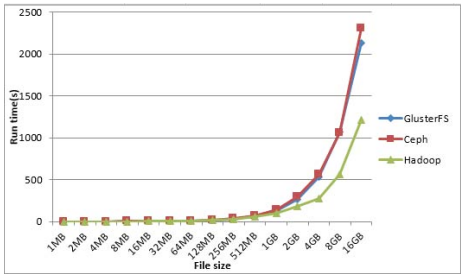

Figure 6. Write single file with different file size

The result for the experiment to download a single file from 1MB to 16GB from the client node to the server node is shown in Figure 7. From the result, we observe that when the file is smaller than 512MB, the performance of the three storage systems are similar. But when the file size is larger than 512MB, the performance of GlusterFS is better than the other two. However, when the file size is larger than 4GB, the performance of Hadoop is better than Ceph.

In this part, we uploaded 16GB file in total, and split the file into smaller files of 10MB to 100MB. We will use it to test the effect of performance when transporting files of different sizes. In Figure 10, we can see the curve is almost stable; however when the file size is just larger than 60MB, the run time of Hadoop increase fast. And when the file size is between 30MB to 60MB, its performance is the best. The performance of GlusterFS and Ceph are similar to each other. And Hadoop performs better than they.

In this part, we downloaded 16GB file in total, and split

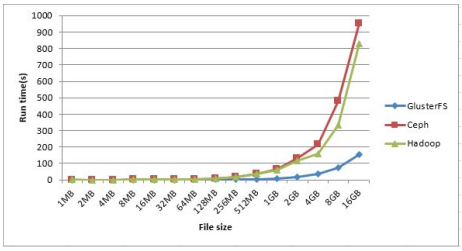

Figure 7. Read single file with different file size

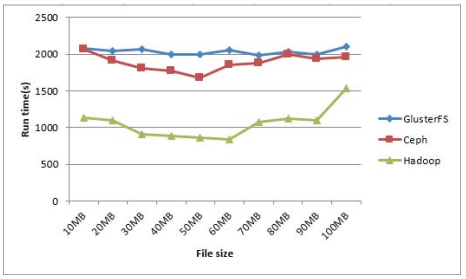

Figure 8. Write small file

the file to be 10MB to 100MB. We will use it to test the effect of performance of downloading different file sizes, as shown in Figure 11. In this experiment, we find the results have not consistent trends; they are close. The performance of Hadoop is better than Ceph most of the time, but when the file size between 50MB and 80MB, the performance of Ceph and Hadoop is close.

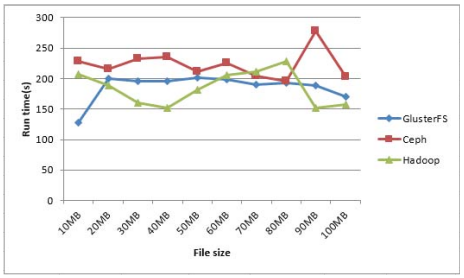

Figure 9. Read small file

In this case, we uploaded 16GB file in total, and split the file from 100MB to 1000MB; we wanted to test the effect of performance of data transport between larger file and small file. From the results, we can observe that the performance of Hadoop is better than the other two. Moreover, when the file size is smaller than 500MB, Ceph is better than GlusterFS, but if the file size is larger than 500MB, Ceph is either similar to GlusterFS or even worse.

In this case, we download 16GB file in total, and split the file from 100MB to 1000MB. From the result, we found that the performance of them is close, but Ceph rises suddenly when the file size is between 500MB and 600MB. And

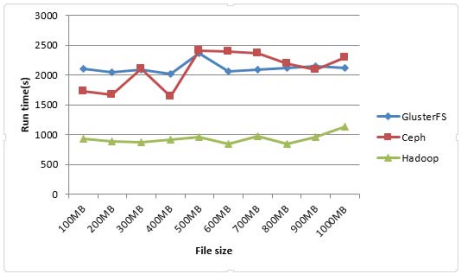

Figure 10. Write small file

Hadoop is the best in them.

Figure 11. Read small file

C. User Interface of ViPR

ViPR can integrate some different original storage architecture through a data service, and provide some automatic mechanism for the storage systems to simplify the management work and decrease the work complexity. It can transform the physical storage from different company or system to be a single, scalable and open virtual storage platform. It can manage all storage resource by a single interface, as shown in Figure 12.

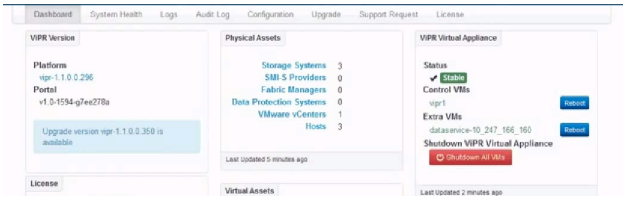

Figure 12. Dashboard

As shown in Figure 13, people can register their storage system to the physical assets part. And people can add some computers in the physical storage array to be a host.

ViPR transforms the physical storage to be a virtual storage plane, and it can create or manage the virtual storage pools automatically by the policy from the manager. When people create the policy, he can set how to create the virtual storage pool according to the storage performance which the service needs, as shown in Figure 14.

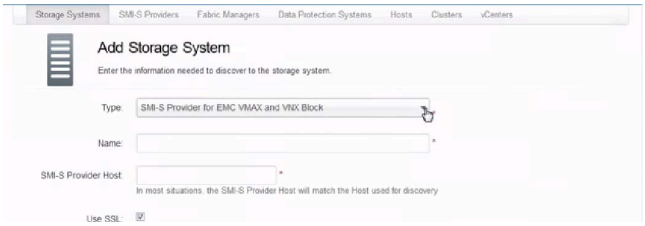

Figure 13. Add PhysicalAsset

Figure 14. Add VirtualAsset

There are some data service will run on the plane, and the virtual storage pool and virtual storage array can provide the application with the data service, as shown in Figure 15. People can split the virtual storage pool to be some virtual storage arrays, and manage them by the policy. There are three types in the data service:

- Object: the file systems can store, take a file and modify the unstructured data by the data service, and the origin data application system can read or write the data easily.
- HDFS: collocation the service, people can transform the origin storage device to be a big data storage place. And they can use it to build a big data environment quickly without building an analysis cluster.
- Block: in the block data service, people can take a data across several storage array, and it can support many kinds of applications, file systems, database and virtual platforms.

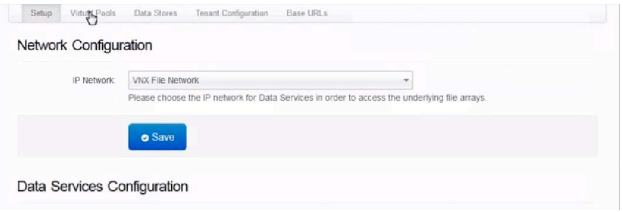

Figure 15. DataService Setup

D. Discussion

We used ViPR to provide the storage management to system; it can manage storage array simply through the data plane and control plane. The data plane is an advantage data service which runs on storage arrays, for example it can offer

ability of object storage on file system storage platforms or offer ability of block storage on normal storage platforms. The data service which can virtualize the storage system combines different data types (file, object, and block), protocol (iSCSI, NFS, and REST), security, usability. We used ViPR to offer a storage pool to our system, and we prove that the system will run normality by the experiments in this chapter. According to the experimental results, we can find that the performance of HDFS is better than the others, and thus, HDFS is more mature and has higher performance than the others. And ViPR can create virtual storage pool and assign the storage space to different kinds of storage.

V. Conclusions and Future Work

A. Conclusion Remarks

How to manage the cloud in a more humane way and make cloud deployment simpler are very important. In this thesis, we built a cloud platform to achieve the concept of Storage-Defined Storage by software, through distributed cloud architecture to build a high reliability and high scalability cloud service, which integrates heterogeneous storage technologies. Finally we also provide a high usability user friendly interface to let users use this system with ease.

B. Future Work

Our system still has many places to improve. The first problem is the storages which the software of our system can support has limit. It just can support some storage technologies. And the other is the environment of our system is small. In the future we hope we can deploy software that can support more storage technologies. And we hope to build a larger environment with our system architecture, and improve the performance of the data transport from the management node to the storages.

Acknowledgment

This work was sponsored by the U-Care ICT Integration Platform for the Elderly, No. 103GREEnS004-2, Aug. 2014, Tunghai University. This work was supported in part by the Ministry of Science and Technology, Taiwan ROC, under grant numbers MOST 103-2221-E-029-021.

References

[1] Chao-Tung Yang, Wen-Chung Shih, and Chih-Lin Huang. Implementation of a distributed data storage system with resource monitoring on cloud computing. In *GPC*, pages 64–73, 2012.

[2] Chao-Tung Yang, Wen-Chung Shih, Guan-Han Chen, and Shih-Chi Yu. Implementation of a cloud computing environment for hiding huge amounts of data. In *International Symposium on Parallel and Distributed Processing with Applications*, pages 1–7, 2010.

[3] Chengzhang Penga and Zejun Jiang. Building a cloud storage service system. *Procedia Environmental Sciences*, 10:691–696, 2011.

[4] Ankur Agrrawal, Rahul Shankar, Shagun Akarsh, and Piyush Madan. File system aware storage virtualization management. In *2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–11. IEEE, 2012.

[5] Tahani Hussain, Paulvanna Nayaki Marimuthu, and Sami J. Habib. Managing distributed storage system through network redesign. In *Asia-Pacific Network Operations and Management Symposium*, pages 1–6, 2013.

[6] Dejun Wang. An efficient cloud storage model for heterogeneous cloud infrastructures. *Procedia Engineering*, 23:510–515, 2011.

[7] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *OSDI*, pages 307–320, 2006.

[8] Brian N. Bershad and Jeffrey C. Mogul, editors. *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*. USENIX Association, 2006.

[9] Jonghyeon Yun, Yong Hun Park, Dong Min Seo, Seok Jae Lee, and Jae Soo Yoo. Design and implementation of a metadata management scheme for large distributed file systems. *IEICE Transactions*, 92-D(7):1475–1478, 2009.

[10] Grant Mackey, Saba Sehrish, and Jun Wang. Improving metadata management for small files in hdfs. In *CLUSTER*, pages 1–4, 2009.

[11] Jeffrey Shafer, Scott Rixner, and Alan L. Cox. The hadoop distributed filesystem: Balancing portability and performance. In *ISPASS*, pages 122–133, 2010.

[12] Hsiang-Ching Chao, Tzong-Jye Liu, Kuong-Ho Chen, and Chyi-Ren Dow. A seamless and reliable distributed network file system utilizing webspace. In *WSE*, pages 65–68, 2008.