International Journal of Computer Applications (0975 – 8887)
Volume 102 – No.10, September 2014

Building an Object Cloud Storage Service System using OpenStack Swift

Sridevi Bonthu
Department of CSE
Vishnu Institute of Technology
Bhimavaram

Y S S R Murthy
Department of CSE
Sri Vasavi Engineering College
Tadepalligudem

M. Srilakshmi
Department of IT
Vishnu Institute of Technology
Bhimavaram

ABSTRACT

Cloud Storage Systems are increasingly noticed now-a-days as they are promising elastic capability and high reliability at low cost. In these services, the files are stored in an authenticated cloud storage service center. The most important feature is storage is adjusted dynamically, and there won't be any worry about space being inadequate or wasted. This paper presents a solution for deploying an Object Cloud Storage Service System based on the open-source cloud Operating System OpenStack and Swift.

General Terms

Cloud Computing, Object Storage, Open Source Software.

Keywords

Amazon S3, OpenStack, Swift, REST, Cloud Storage.

1. INTRODUCTION

Cloud Computing regards software service, platform service and Infrastructure services, which are made available as order based services to users in a pay-as-you-go model. These services are referred to as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) respectively. In a recent report of Berkeley, the importance of these services is highlighted as: "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service" [1]. Several research fields like utility computing, web computing, grid computing, distributed computing and virtual computing highly contributed to today's success of cloud computing.

Cloud Computing is increasingly noticed as it must go with cloud storage. Cloud storage success is considered economical and technology-driven. In the context of cloud computing, storage has not only been an important component of large-scale cloud services, but also been provided as a virtual storage infrastructure in a pay-as-you-go manner, such as Amazon Simple Storage Service (Amazon S3) [2]. Amazon S3 is regarded as a best reference of cloud storage service. Elastic and unlimited storage space is provided by it and this storage service can be accessed using a set of simple web interfaces. Users pay for the services each month for the amount of storage, get requests, put requests and data transfers. Moreover, the volume of data stored inside data centers has been observed to be growing even faster than Moore's Law [3, 4]. It has been reported that the storage space used for photo storage only in Face book has been over 20 PB in 2011, 160 PB in 2014 and is increasing by 60 TB every week [5].

With reference to Amazon S3 API [6], several features of it can be summarized as follows: Users create buckets first. These are analogous to folders which contain files. These buckets contain the arbitrary objects up to 5 terabytes in size. Each bucket can be accompanied by metadata up to 2 terabytes. In cloud storage services the buckets are referred to as containers and objects as basic unit of user data generally. Second, users can upload and download the entire objects into their respective containers. Each object in a bucket is identified by a unique key through REST and SOAP interfaces. HTTP is the primary protocol to access the objects. Third, a customized HTTP scheme which is based on Hash Message Authentication Code (HMAC) [7] used for accessing Amazon S3 buckets or objects. Both Access Key ID and Secret Access Key are obtained by registering with an account. Every request includes the Access Key ID, selected elements and their signatures. Amazon S3 service gets a Secret Access Key and computes a Signature for the message upon receiving an authenticated request. Fourth, users may not get their modified content of an object identifier very soon after overwriting, as it holds eventual consistency.

Usage of proprietary solutions locks us in with a single vendor or with a single programming language that do not comply with standard ways of exchanging data. In Open-Source, Eucalyptus [8, 9], OpenNebula [10, 11], Nimbus [12] and OpenStack [15] have been largely studied in the literature. The architecture and various components of these solutions were presented in these works. There are also comparative studies of different solutions [13, 14]. From a comparative study of existing Infrastructure-as-a-Service (IaaS)Papers [8-15], an open source cloud platform OpenStack is proposed, which is a modular and totally open for any use including research. From the OpenStack cloud platform, Swift software for building object storage cloud and keystone for providing authentication are selected.

OpenStack brings all the stuff through open service model like Amazon Web Services (AWS) and makes easier to get the power of the cloud on a dedicated platform. OpenStack [15] is one of the world's most deployed open cloud infrastructure platforms. It is deployed in a number of public and private clouds throughout the world. It started out as collaboration between NASA and Rackspace and has ended up in the open source community licensed under the Apache 2.0 license. Essentially, OpenStack is a cloud operating system that controls pools of resources. These resources can be in the form of computation, networking or storage.

Swift is proposed for building storage cloud based on the comparison between ceph, glusterFS, Sheepdog, walrus, cumulus [16]. Swift allows for a wide spectrum of uses, including support for web or mobile applications, backup, and active archive. Object Storage provides cost effective, scale-out and redundant storage, utilizing clusters of standardized hardware. Its goal is not to provide a file system or real-time data system, but rather provide a long-term storage system for large amount of static data. Data can be uploaded, downloaded, maintained and retrieved through an REST API. Authentication and authorization is optional but recommended [15]. With OpenStack Swift, three kinds of authentication can be provided, They are TempAuth, Swauth and keystone Identity services [17].

A quick summary of characteristics of OpenStack Swift are – Open source and freely available, it currently powers the largest object storage clouds, including Rackspace Cloud Files, the HP Cloud, IBM SoftLayer Cloud and countless private object storage clusters, it can be used as a stand-alone storage system or as part of a cloud compute environment. It runs on standard Linux distributions and on standard x86 server hardware, like Amazon S3 swift has an eventual consistency architecture, which make it ideal for building massive, highly distributed infrastructures with lots of unstructured data serving global sites, all objects stored in swift have an URL, applications store and retrieve data via industry standard RESTful http API, objects can have extensive metadata which can be indexed and searched, all objects are stored with multiple copies and are replicated in as-unique-as possible availability zones and/or regions, it can be scaled by adding additional nodes, which allows for a cost-effective linear storage expansion, failed nodes and drives can be swapped out while the cluster is running with no downtime.

The object storage also provides persistent block level storage service for use with OpenStack compute instances to meet storage needs from virtual instances. This also includes snapshot management for backing up data stored on block storage volumes, including the OpenStack Object storage in the full OpenStack cloud stack integration. One can use OpenStack API after getting authenticated through identity service to create and manage resources in an OpenStack cloud. Various methods for sending API requests to an OpenStack Swift Storage Cloud include a command line tool – curl, OpenStack command line for clients and REST clients. Object storage systems organizes the data in a hierarchy like Account, Container and Object. An Account is top-level of the hierarchy, Container like buckets and Object store data content, such as documents, images and so on. One can even use query parameters like marker, limit, end-marker to page through large list of containers or objects. OpenStack Swift Object Storage Cloud provides all the functionality to our local datacenters like an Amazon S3 Object Storage Cloud.

In this paper, a solution to build a cloud object storage service based on open source cloud framework is presented. The end users can access their data on storage cloud in several ways in which they are more familiar. The presented system supports more number of concurrent connections.

The remainder of the paper is organized as follows. In section 2, architecture of private Cloud Object Storage Service (COS3) is presented and discussion about the system components and prototype system based on the architecture in detail is included. Section 3 covers the related work done based on the prototype system. Section 3 summarizes the paper with future work.

2. SYSTEM MODEL

A high level system model has to be outlined to successfully design, deploy and evaluate a storage system. In this section a general architecture COS3 is presented. COS3 architecture is flexible, simple and modular with a hierarchical design. This architecture reflects common resource environments found in any academic settings. The system allows the users to access their own content by using REST based interfaces or command line tools like curl, OpenStack Swift clients. Of Course, for the systems better accessibility one can build an easy-to-interact web-front-end. Through REST based interfaces users can upload, download, list and look through the data using web browsers based on the Access Control Limits set to that particular user.

Figure 1 depicts the proposed architecture of COS3 to offer the services. This system consists three layers. They are Data Storage Layer, Data Transformation and Routing Layer and Data Access Layer. The key challenge include the design techniques followed for each of these layers and making these components work together in a coherent system

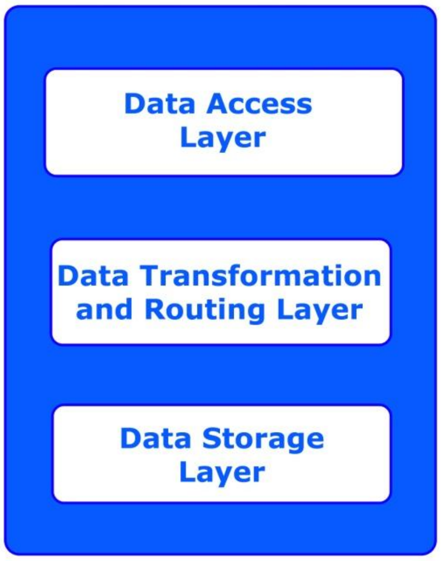

Fig1: Architecture of Cloud Object Storage Service System.

Whatever may be the storage system, the final goal is that the storage system must be highly scalable and highly available. New storage nodes could be easily added and removed from the storage components to achieve dynamic scalability. Data Transformation and Routing Layer mainly changes different type of web requests (eg. SOAP or REST) into bottom level data storage requests. If the end users want to share the cloud storage service more and more, then several different interfaces must be provided.

A prototype system for COS3 is based on the proposed architecture. COS3 includes six nodes on a hypervisor. Among the six nodes one node is proxy node and the remaining five nodes are storage nodes.

3. RELATED WORK

VMware vSphere Hypervisor is based on VMware ESXi, the hypervisor architecture that sets the industry standard for reliability, performance and cross-platform support. Using VMWare vSphere, a hypervisor is installed on the IBM X3650M4, 6-Core Server. To create cloud storage virtualization of server and storage are needed. Sever Virtualization enables different Operating Systems to share the same hardware and make it easy to move Operating Systems between different hardware, all while the applications are running. Storage Virtualization creates the abstraction layer between the applications running on the server, and the storage they use to store the data. To achieve Server Virtualization, six Virtual Machines are created on the host machine. Ubuntu 14.04 is adopted as the guest operating system on the VMs. Therefore a storage cluster has been configured using six servers of the SVES data center and its networking infrastructure. All the nodes have the same hardware configuration and are connected over a dedicated storage VLAN through Gigabit Ethernet links.

Among the six nodes, one node is proxy nodes and remaining are storage nodes. The structure of the storage nodes of COS3 is shown in figure 2. Proxy server exposes the public API and serves requests to storage entities. For each request, the proxy server looks up the location of the account, container and the object using the ring. The web interfaces or the command line tools interact with the proxy server only. Proxy server includes Authentication servers. Additional proxy servers can be added for providing high availability and scalability. The Storage nodes include Account, Container and Object Servers. Additional storage nodes can be added in a particular zone for scalability and capacity. Additional zones can also be added for even higher high availability.

The storage servers hard drives are formatted with XFS file system. Since swift requires extended attributes (XATTRS) to store the meta-data and XFS is the only file system thoroughly tested by Rackspace, it is the recommended one for using on the storage disks. The proxy server is exposed to public IP address and the storage servers are all in a private subnet connected to the proxy server through its private IP address on the same subnet. The entire network is connected through a GigE switch. It is advisable to use multiple proxy servers with 10 GigE network cards in a protection system.

Ubuntu14.04 Edition with Kernel 3.13.0-24-generic operating system is installed on all nodes. Swift version 1.13.1 and all the necessary stuff is also installed on the nodes. One of the *advantages* of Swift is that it can run on a wide range of hardware configurations, from commodity hardware to commercial hardware solutions. After the division of nodes, roughly 100 partitions are set per disk and the ring partition power has been set to 9. It accounts to a total of 512 partitions. The number of workers on proxy and storage server components has been configured as proxy server:8, Account: 4, Container Server: 4 and Object server: 4. Default values are left to the auditor, replicator, reaper and updater settings. After deploying a storage cloud, it can be accessed via browser, cyberduck or gladient. Those who are familiar with command line can use swift client python program or curl.

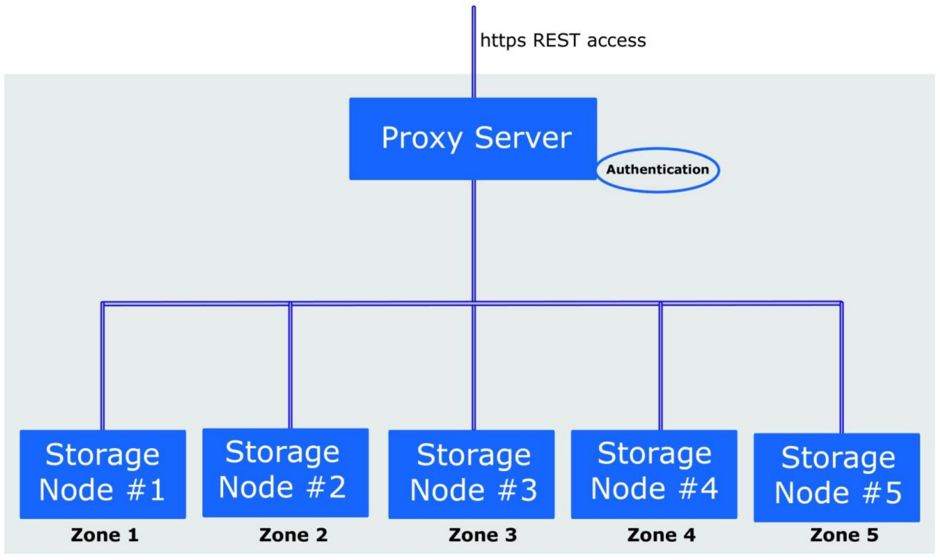

Fig 2: six node swift cluster - COS3

4. MEASUREMENT

Benchmarking a Swift cluster implementation is essential before the cluster is deployed for production use as an integral part of deploying a cloud storage platform based OpenStack Swift. Swift-bench is a command-line benchmark tool that is shipped along with Swift distribution [17]. In this work, Swift-bench is used as the workload generator to benchmark the Swift cloud in terms of #PUT operations per second, #GET operations per second and #DELETE operations per second.

The benchmarking results shown in figure 3 show the operation rate (operations per second on the Y-axis) of the PUT operation for the implemented swift cluster. Figures 4 and 5 shows the results of GET and DELETE operations respectively.

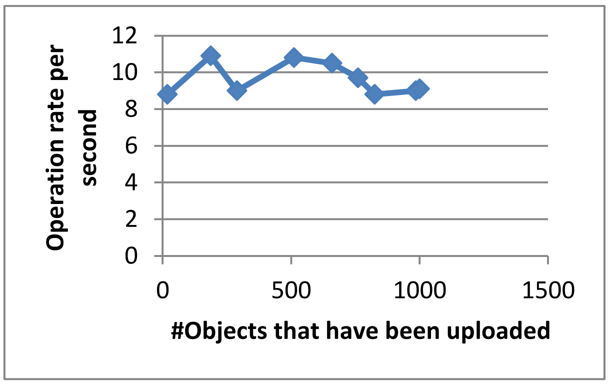

Fig 3: Workload of PUT requests

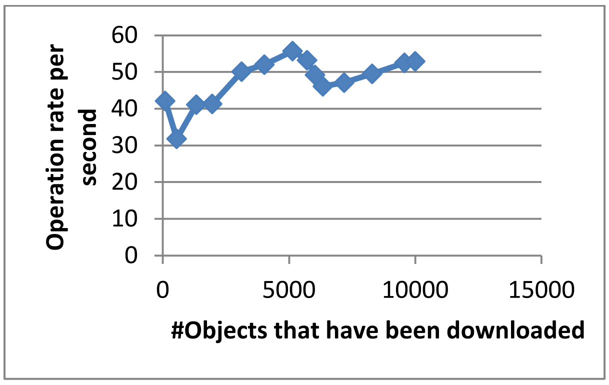

Fig 4: Workload of GET requests

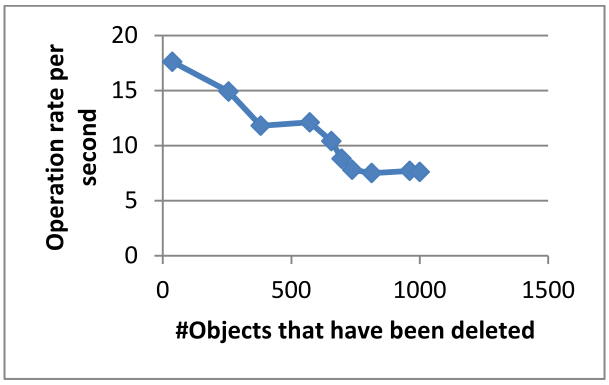

Fig 5: Workload of DELETE requests

Swift-bench allows to tune the workloads based on the following parameters:

object_size: defines the size of an object (or a file) that will be uploaded.

concurrency: defines how many concurrent threads will be launched for a benchmark test.

num_objects: the number of objects in total that will be uploaded to Swift.

num_gets: the number of objects in total that will be downloaded from Swift.

num_containers: the number of containers that will be created for storing the objects. The objects are uniformly distributed across all containers.

5. FUTURE WORK

As future work a quantitative comparison of the presented solution through performance evaluation measurements can be done. Utility of dynamically migrating service executions across handheld and server platforms can also be developed. A good web-front-end for accessing the cluster can be developed using open source as the storage cloud can be accessed via REST, curl or swift client only. The open source front-end-design options include PHP-Open Cloud SDK, Web Dav Claimito etc.

6. CONCLUSION

In this paper, a solution for building a storage cloud based on open source tools is proposed. Based on this architecture any educational institution can build their own storage cloud to provide storage accounts to the students. This work will help enterprises / organizations to setup their own private, public or hybrid cloud. This work completely leverages existing open source solutions to expedite the development effort and builds a local prototype personal cloud consisting of low end PC and mobile devices. By benchmarking, it is also found that PUT and DELETE are dominant operations on Swift cluster.

7. REFERENCES

[1] Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, and Zaharia M. Above the clouds: A berkeley view of cloud computing. Technical report; 2009.

[2] Amazon simple storage service (Amazon S3), http://aws.amazon.com/s3/, 2013.

[3] World’s data more than doubling every two years — Driving big data opportunity, new IT roles, http://www.emc.com/about/news/press/2011/20110628-01.htm, 2013.

[4] IDC says world’s storage is breaking Moore’s law, more than doubling every two years, http://enterprise.media.seagate.com/2011/06/insideit-storage/idc-says-worlds-storage-is-breaking-mooreslaw-more-than-doubling-every-two-years/, 2012.

[5] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel, Finding a needle in Haystack: Facebook’s photo storage, in Proc. 9th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2010.

[6] Amazon Simple Storage Service, aws documentation, API reference (API version 2006-03-01) Amazon Web Services LLC; 2014.

[7] Krawczyk H, Bellare M, Canetti R. HMAC: Keyed-hashing for message authentication; 1997.

[8] Eucalyptus URL: http://www.eucalyptus.com/
Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G.,Soman, S., Youseff, L. et al. (2009) The Eucalyptus Open-Source Cloud Computing System. In: 9th IEEE/ACM International Symposium on Cl uster Computing and the Grid .Shanghai, China 2009.

[9] B. Sotomayor, R.S. Montero, I.M. Llorente, I. Foster, Virtual infrastructure management in private and hybrid clouds IEEE Internet Comput. 13 (2009) 1422. doi:10.1109/MIC.2009.119.

[10] OpenNebula URL: http://opennebula.org/

[11] Nimbus. URL: http://www.nimbusproject.org/.

[12] Peter Sempolinski and Douglas Thain,A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus, University of Notre Dame.

[13] Z.Lei, B. Zhang, W. Zhang, Q. Li, X. Zhang, and J. Peng Comparison of Several Cloud Computing Platforms. Second International Symposium on Information Science and Engineering, pages 23-27,2009

[14] OpenStack URL: http://www.openstack.org/

[15] Prakashan Korambath, Narcis Madern Investigating Private Cloud Storage Deployment using Cumulus, Walrus, and OpenStack/Swift.

[16] OpenStack Swift Authentication System URL: http://docs.openstack.org/developer/swift/overview_auth.html

[17] Benchmarking – SwiftStack Documentation URL: https://www.swiftstack.com/docs/faqs/benchmarking.html