

東海大學

資訊工程研究所

碩士論文

指導教授：楊朝棟博士

整合共享儲存系統透過 Ceph 分散式檔案系統和
Rados 閘道用於巨量資料存取

The Integration of Shared Storages with the CephFS and
Rados Gateway for Big Data Accessing

研究生：翁嘉佑

中華民國一零六年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 翁嘉佑 所提之論文

整合 Ceph 分散式檔案系統和 Rados 閘道應用於
巨量資料存取

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召集人

江輔政

簽章

委員

員

江輔政
鈕朝欽

詹毓偉

指導教授

楊朝棟

簽章

中華民國 106 年 6 月 30 日

摘要

最近幾年，高可用性共享存儲將成為受歡迎的資訊產業發展方向。現在資訊產業降低高風險數據需求並提高存儲性能，讀寫，能越來越重要。Ceph 是一種分散式檔案系統提供高效能、可靠性及擴充性。因此，本文的研究主要應用 Ceph 存儲與大數據性能測試，以求最佳的讀寫速度性能和數據備份。本系統從 Hadoop 操作啟動。將資料儲存在 Hadoop 分散式檔案系統 (HDFS) 並複製到 Alluxio 記憶體空間。通過 Map Reduce 處理的數據得到結果，輸出將儲存到 Alluxio 記憶體空間。對於第一個實驗，我們使用 Ceph API 組件的 S3 API 和 Rados Gateway 作為 Alluxio 和物件儲存裝置 (OSD) 之間的橋樑。第二個實驗與第一個環境相同，但是 Map Reduce 的輸出將使用 Ceph File System(CephFS) 直接連接到物件儲存裝置 (OSD)。數據在 Ceph 中比在 Alluxio 中更安全，因為 OSD 可以用對象存儲級別備份數據。我們還可以使用 S3 瀏覽器 (GUI) 來維護 OSD 的數據，例如：授權訪問，保存文件夾，創建用戶帳戶，移動數據位置等。最後我們使用 Inkscope 監控來所有系統，如果系統出現任何問題，系統將回復錯誤或向用戶發出警告提示。

關鍵字: 巨量數據，高可用性，共享儲存，Ceph 儲存系統

Abstract

In recent years, high availability shared storage will become a popular information technology industry development orientation. Currently, information technology industries emphasize to reduce high risk data requirements and improve read and write performance of data storage. Therefore, the main purpose of this work is to improve read and write performance with the best way on Ceph Storage Cluster. In this system, the data is stored on Hadoop Distributed File System (HDFS), and the data stored in-memory virtual distributed store system that mentioned as Alluxio automatically. Then, the data would be processed through Hadoop Map Reduce method and the output would be inserted into Hadoop Distributed File System and Alluxio environment. The first experiment is to use S3 as application program interface that will connect to RADOS Gateway stored data into Object Storage Daemon (OSD). The second experiment is based on the first out experiment would be through Ceph File System (CephFS) connected to Object Storage Daemon directly. The data is saved in Ceph environment more secure than in Alluxio as in-memory storage system because OSD can be used for data backup based on object storage level. We can use S3 browser (GUI) to maintain data like grant access, maintain folders maintenance, create user accounts, move data location etc. The last one, we used Inkscope monitors all system. If there is any problem, system will give warning or error responds to users automatically.

Keywords: Big Data, High Availability, Share Storage, Ceph Storage System

致謝詞

這兩年在東海大學的研究生生活讓我受益良多，在研究所的課程幫助下，讓我對於研究的領域更加精進，透過雲端技術與分散式檔案系統的深入研究，能實際將這些技術應用於生活當中，並完成一篇論文與系統。

能完成這篇論文必須感謝很多人，首先，非常感謝我的指導教授楊朝棟教授，從大學部跟著楊老師作專題，到研究所作研究，一路上不斷傳授雲端技術以及分散式檔案的知識，除了研究的方面的教導，楊老師教了我更多平常作人處事的態度，有了這些東西才能讓我完成這篇論文，謝謝老師這幾年的指導，雖然在研究上常常碰到瓶頸，但因為老師的鼓勵與幫忙，使我能找到解決的辦法繼續作下去。

特別感謝口試委員張玉山教授、詹毓偉教授、伍朝欽教授以及江輔政教授特地撥空前來參加我的論文口試，在論文口試時提出很多論文的盲點和非常多寶貴的意見，讓我能將論文修改得更加完整，學生衷心感謝。我也要感謝我實驗室的學長姐、學弟妹以及最重要的同學們，尤其是陳彩進學長，總是不厭其煩地鼓勵我，並且在技術方面指導我，這兩年的生活一路走來，如果不是大家互相幫忙，要完成這麼多事情是不太可能的，因為有大家的陪伴，讓我在這兩年的生活中，增添許多快樂。

最後要感謝我的家人，如果沒有他們的支持我沒辦法完研究所的學業，因為有你們對我的關心與幫助，讓我的研究得以成功，由衷感謝一路陪伴的所有人。

東海大學資訊工程學系 高效能計算實驗室 翁嘉佑 106 年 07 月

Table of Contents

摘要	i
Abstract	ii
致謝詞	iii
Table of Contents	vi
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	2
1.3 Thesis Organization	3
2 Background Review and Related Works	4
2.1 Background	4
2.1.1 Ceph Storage System	4
2.1.2 Alluxio	7
2.1.3 Hadoop	7
2.1.4 Map Reduce	9
2.1.5 HDFS	10
2.1.6 Gluster File System	12
2.2 Related Works	13
3 System Design and Implementation	15
3.1 System Architecture	15
3.2 System Implementation	17
3.2.1 Ceph Storage Service Deployment	17
3.2.2 HDFS Deployment	17
3.2.3 Alluxio Deployment	18
3.2.4 Rados Gateway Deployment	20
3.2.5 Ceph File System Deployment	21
3.2.6 User Services	25

4	Experimental Results	26
4.1	Experimental Environment	26
4.2	Experimental Results	29
4.2.1	MapReduce Input and Output	29
4.2.2	Inkscope Monitoring System	32
4.2.3	The Performance between Rados Gateway and CephFS Comparison	37
4.2.4	The Performance among Rados Gateway, CephFS and GlusterFS Comparison	42
5	Conclusions and Future Work	45
5.1	Concluding Remarks	45
5.2	Future Work	46
	References	47
	Appendix	50
A	Ceph Installation	50
B	Rados Gateway Installation	53
C	Hadoop Installation	55
D	Alluxio Installation	59
E	CephFS Installation	61
F	Inkscope Installation	63

List of Figures

2.1	General Ceph Architecture	5
2.2	Alluxio Architecture	7
2.3	Map Reduce Architecture	9
2.4	HDFS architecture	11
3.1	Hadoop, Alluxio and Ceph integration architecture	16
3.2	Ceph environment	18
3.3	Ceph Osd tree	18
3.4	Add a hard drive	19
3.5	HDFS environment	19
3.6	Hadoop jps	20
3.7	Alluxio environment	20
3.8	Alluxio instances	21
3.9	Output state before	21
3.10	Output state after	21
3.11	OSD space saved data	22
3.12	Rados Gateway environment	22
3.13	Rados Gateway instances	23
3.14	Ceph File System instances	23
3.15	CephFS spatial changes	23
3.16	Alluxio WebUI before upload	24
3.17	Alluxio WebUI after upload	24
3.18	Inkscope environment	25
4.1	Hardware environment	27
4.2	Data Source in Alluxio	29
4.3	Map Reduce Output 1	29
4.4	Map Reduce Output 2	30
4.5	Ceph OSD Output	31
4.6	Inkscope Ceph Login Page	32
4.7	Inkscope Ceph cluste hosts	32
4.8	Inkscope Ceph status	33
4.9	Inkscope Ceph Pools Management	34
4.10	Inkscope Ceph osd page	35
4.11	Inkscope Ceph osd map	36
4.12	Inkscope Ceph Pools Management2	36

4.13	Speed Performance Test with S.Read/Write for each OSD and CephFS	38
4.14	Speed Performance Test with Randomize Read/Write for each OSD and CephFS	38
4.15	Speed Performance Test with S.Read/Write(30%) for each OSD and CephFS	38
4.16	Speed Performance Test Randomize Read/Write(30%) for each OSD and CephFS	39
4.17	IOPS Performance Test with S.Read/Write for each OSD and CephFS	41
4.18	IOPS Performance Test with Randomize Read/Write for each OSD and CephFS	41
4.19	IOPS Performance Test with S.Read/Write(30%) for each OSD and CephFS	41
4.20	IOPS Performance Test with Randomize Read/Write(30%) for each OSD and CephFS	42
4.21	Performance Test with read for Rados Gateway, CephFS and GlusterFS	43
4.22	Performance Test with write for Rados Gateway, CephFS and GlusterFS	44

List of Tables

3.1	Software Specification	17
4.1	Experimental hardware description	28
4.2	Experimental hardware specifications	28
4.3	Performance Comparison Test with Read, Write, Randomize for each OSDs	37
4.4	Performance Comparison Test with Read, Write, Randomize for CephFS	37
4.5	IOPS Performance Test with Read, Write and Randomize for each OSDs	40
4.6	IOPS Performance Comparison Test with Read, Write, Randomize for CephFS	40

Chapter 1

Introduction

In recent years the importance of continuous delivery will continue to remove the development and needs, requirements, availability and scalability. The tool must be highly available so that engineers can provide new software. Many companies need an auto-expand the shared storage system [1] and its services remain available even when the component services failed. The data can be stored in a database and file system or any other content, but eventually will be stored in a storage device such as SSD [2] or hard drive. It requires high availability shared storage, reduces the risk of device disruption, file system corruption, and the system connected to the device may be interrupted. In order to avoid interruption, it would like to have all of the components of the multiple instances. If an error occurs, other components may reference the automatically takes over and copy and distribute all data. Users do not need to worry if any disruption occurs.

1.1 Motivation

As Big Data has become so widespread, global industry have invested much capital and research on it. They want to use Bid Data technology to analyze data that is to huge to be easily handled. By using big data technology such as data mining to obtain valuable information, it is desired to alleviate issues of expanding data

currently facing with us. The tool must be highly available so that engineers can provide new software. Many companies need to automatically extend the shared storage system to solve the problem. Therefore, we propose a high availability, an open source, scalable, software-defined monitoring storage system.

Ceph [3] is an open source, scalable, software-defined storage system that can run on commodity hardware. Ceph from the outset has been developed in a self-management and self-healing with no single point of failure on a single software platform provides object, block and file system storage [4,5]. Because of its highly scalable software defined storage architecture, Ceph of traditional storage systems is also ideal replacement for cloud computing environment object and block storage the powerful storage solutions.

To achieve these goals, We store the data in the Hadoop Distributed File System (HDFS) and use Alluxio as a bridge to copy to Alluxio memory space. The result is processed by Map Reduce, and the output is stored in the Alluxio memory space. Ceph itself has two ways of passing, and we will compare the performance between the two. The first is Ceph RADOS Gateway, we use the Ceph API component S3 API and Rados Gateway as a bridge between Alluxio and Object Storage (OSD). The second is the Ceph File System [6], which is the same as the RADOS Gateway [7] environment, but the output of Map Reduce will be directly connected to the object storage device (OSD) using Ceph File System (CephFS).

1.2 Contributions

In this paper, we Integration a highly scalable, highly available, distributed distributed storage system. We use Ceph Storage , Alluxio [8], and Apache Hadoop [9,10] integrate technology into a complete system. We have encountered many problems, such as the lack of OSD memory space. We try many different ways to link each other big data tools, and finally we use Alluxio, Amazon S3 and

Hadoop. The most important thing is that we overcome the version of the problem. Each big data tool we have tried three different versions to test whether it can be successfully linked. And we Integrate Hadoop Distributed File System (HDFS) to speed up processing of files, using Alluxio's memory-centric architecture allows data access several times faster than existing traditional solutions. We use Inkscope monitors all system. We can clearly see the space configuration, if there are any problems with the system will respond to error or warning to users.

1.3 Thesis Organization

In Chapter 2, we will introduce some background information, Ceph Storage System, Apache Hadoop, Alluxio and Gluster File System. Chapter 3 shows system architecture. Chapter 4 shows the experimental environment and experimental results. Finally, In Chapter 5 we discuss and summarize our study for future work.

Chapter 2

Background Review and Related Works

2.1 Background

2.1.1 Ceph Storage System

Ceph [11] is an extensible, open source, software-defined storage system that runs on commodity hardware. It is designed to provide decentralized object storage and archiving systems for performance, reliability and scale. Ceph [12] has developed from scratch to store objects, blocks and file systems in a single software platform that is self-managing, self-healing, and without single points of failure. Ceph was made possible by a global community of enthusiastic storage engineers and researchers. It is open source and freely-available. Ceph [13] 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. Figure 2.1 shows the architecture of Ceph.

It is RADOS (Reliable Autonomic Distributed Object Store) as the main design of the decentralized storage platform, so the ability to expand horizontally very powerful. He can expand thousands of nodes horizontally, and thus provide

PB, or even EB-level storage space. Because of its highly scalable software-defined storage architecture, Ceph is an ideal replacement for traditional storage [14] systems and a powerful storage solution for objects and block storage for cloud computing environments.

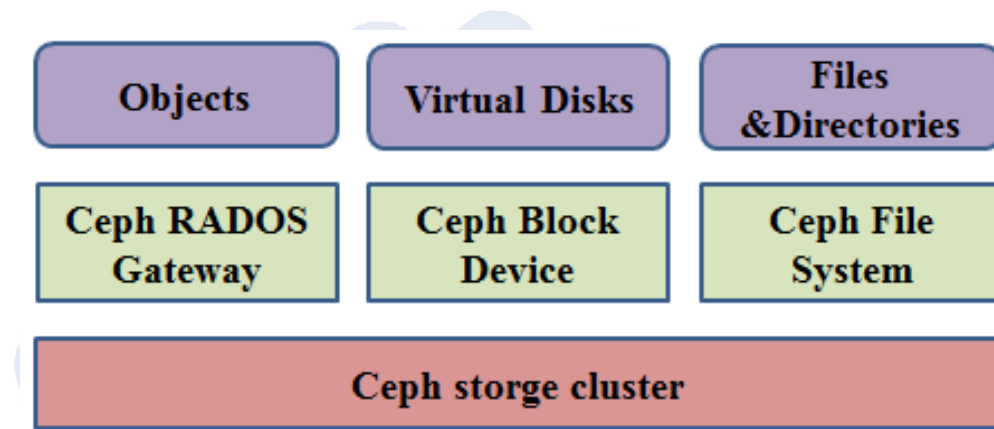


FIGURE 2.1: General Ceph Architecture

Ceph [4] consists of three component services, that are Object Storage, Block Device and CephFS.

- Object Storage: Ceph is a distributed object storage and file system designed to provide superior performance, reliability and scalability. Its software library to client applications provide direct access to reliable autonomous distributed object storage (RADOS) object storage system and some of the advanced features of the Ceph provides the basis, including RADOS block device (RBD), and the Ceph RADOS Gateway File System software library for the librados C, C, java, python and applications written in PHP. RADOS Gateway object storage will also be open to restful interface that can be used as a local Amazon S3 Display. Librados library provides advanced features, including:
 - Snapshots
 - Object level key-value mappings
 - Partial or complete reads and writes

-Atomic transactions with features like append, truncate and clone range

- **Block Storage:** Ceph's object storage system allows the user to use the Ceph installed as a low configuration of block device. The Ceph RADOS block device (RBD) can access the entire storage cluster in the striped and replicated block device image. When an application uses the block device to write data to the Ceph ceph automatically set the data in the striped and replication. The Ceph RADOS block device provides a block device and can be used as a system of block device is formatted and installed, can also offer QEMU and KVM virtual machines. Ceph RBD interfaces with the same Ceph object storage system that provides the librados interface and the CephFS file system, and it stores block device images as objects. Since RBD is built on top of librados, RBD inherits librados' capabilities, including read-only snapshots and revert to snapshot. Ceph's object storage system is not bounded to native binding or RESTful APIs. User can mount Ceph as a thinly provisioned block device. When write data to Ceph using a block device, Ceph automatically stripes and replicates the data across the cluster. By striping images across the cluster, Ceph increases read access performance for large block device images.
- **File System:** Ceph's file system (CephFS) to provide object storage and block device interface of the same object storage systems. Ceph provides a POSIX-compliant network file system that aims for high performance, large data storage, and maximum compatibility with legacy applications. Compared to many object storage systems available today Ceph's object storage system offers a significant feature: a traditional file system interface with POSIX semantics. Object Storage System is an important innovation, but they complement rather than replace traditional file system. The Ceph metadata server cluster provides a service that maps the directories and file names of the file system to objects stored within RADOS clusters. The metadata server cluster can expand, contract, and dynamically rebalance the file system to distribute data evenly among cluster hosts. As storage

requirements grow for legacy applications, organizations can configure their legacy applications to use the Ceph file system. This means user can run one storage cluster for object, block and file-based data storage. This ensures high performance and prevents heavy loads on specific hosts within the cluster.

2.1.2 Alluxio

Alluxio [15,16] (former known as Tachyon) is an Open Source Virtual Machine Allocated Memory speed storage system. Alluxio is the world's first memory-centric virtual distributed storage system. It unified data access and bridge calculation framework and the underlying storage system. The application only needs to connection Alluxio to access storage on the ground floor of the data storage system. In addition, Alluxio's memory-centric architecture allows data to be accessed several times faster than existing conventional solutions. Figure 2.2 is Alluxio Architecture.

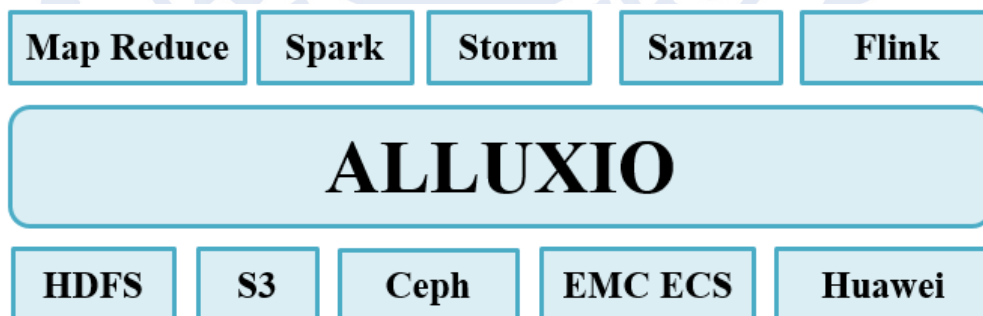


FIGURE 2.2: Alluxio Architecture

2.1.3 Hadoop

Apache Hadoop now is one of the most popular big data processing solution that is the apache software foundation open source frameworks. Hadoop implementation is constructed in accordance with published Google MapReduce and Google File System papers. The Hadoop framework transparently provides reliability and data for applications. The Apache Hadoop frame-work is built on top of the

Hadoop Distributed File System (HDFS), which supports a stable and automatic distributed processing system. HDFS integrates distributed storage resources into a fault-tolerant, efficient and large-capacity storage environment. Hadoop implements the map reduce programming framework, composed by the map and reduce the size of the input that allows the cluster the same as any of the nodes in the implementation of the data pieces. Hadoop From single server scales to thousands of machine and provides parallel computing and increase the number of computing.

The Apache Hadoop project consists of the following: The project includes these modules:

- Hadoop Common: The Hadoop common contains the libraries and modules of Hadoop.
- HDFS: HDFS is designed to provide high throughput access to very large datasets.
- Hadoop MapReduce: MapReduce is composed of the map and reduce, of which the input is divided into a plurality of blocks to be executed on each node.

Hadoop ecosystem has very diverse tools to make Hadoop useful in many applications.

Composed of two roles, Name node and Data nodes:

- The Name node is responsible for the management and storage of information (metadata, namespace) in each file attribute privilege in the file system.
- The Data node usually consists of hundreds of nodes, a data file will be cut into several smaller blocks (Block) stored in a different Data node, each block will also have several copies (Replica) stored in different nodes, so that when one of the nodes damaged, the file system data can be saved.

The Name node also needs to record the location of each file, when the need to access the file, the coordination Data node responsible for the response; When a node is damaged, the Name node also automatically relocates and copies the data.

2.1.4 Map Reduce

MapReduce [17,18] is a decentralized programming framework that allows service developers to write programs very easily, using a lot of computing resources to speed up the processing of large amounts of information. A MapReduce [19,20] operation can be divided into two parts Map and Reduce, a lot of information at the beginning of the operation, the system will be converted into a group (key, value) of the order and automatically cut into many parts. There are passed to different Mapper to deal with, Mapper processing is completed after the results of the results will be sorted into a group (key, value) of the order, and then passed to the Reducer integration of all Mapper results, and finally to the overall results. Figure 2.3 shows the architecture of Map Reduce.

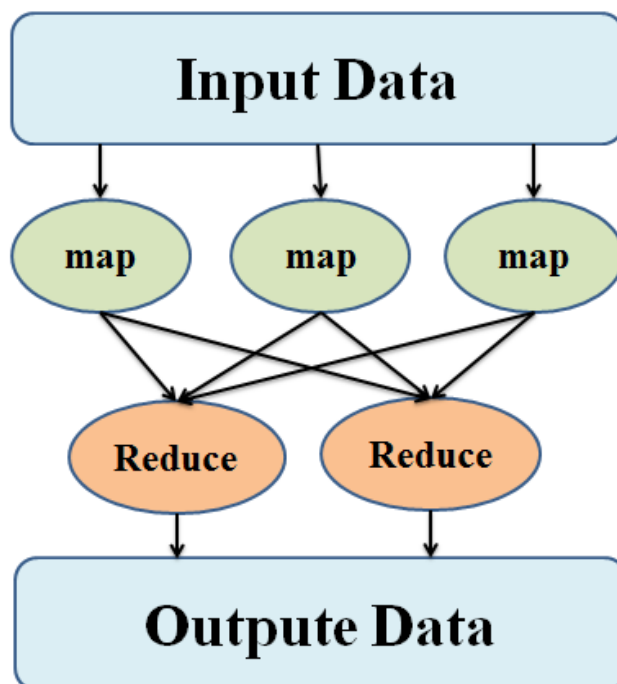


FIGURE 2.3: Map Reduce Architecture

2.1.5 HDFS

Hadoop Distributed File System(HDFS) is a distributed file system designed to run on commodity hardware. The detection of faults and automated recovery is an important architectural goal of HDFS. HDFS has master-slave architecture with a single Name Node as the master server to manage the file system. Besides, a number of DataNodes, usually one per node in the cluster, manage storage attached to the nodes. HDFS describes a file system namespace and allows user data stored in files. Internally, a file is split into one or more blocks that are stored in a set of Data Nodes. The Name Node executes file system namespace operations such as to open, close, and rename files and directories, and it controls the mapping of blocks to Data Nodes as well. The Data Nodes are responsible for responding read and write requests from clients of the file system. HDFS ensures input distribution and provides the user with an interface whose role is to provide chunks of data files to cluster nodes. Among its chief advantages, HDFS provides input locality by enabling nodes hosting input shards to apply their processing on such chunks, rather than on remotely stored data. Figure 2.4 shows the architecture of HDFS.

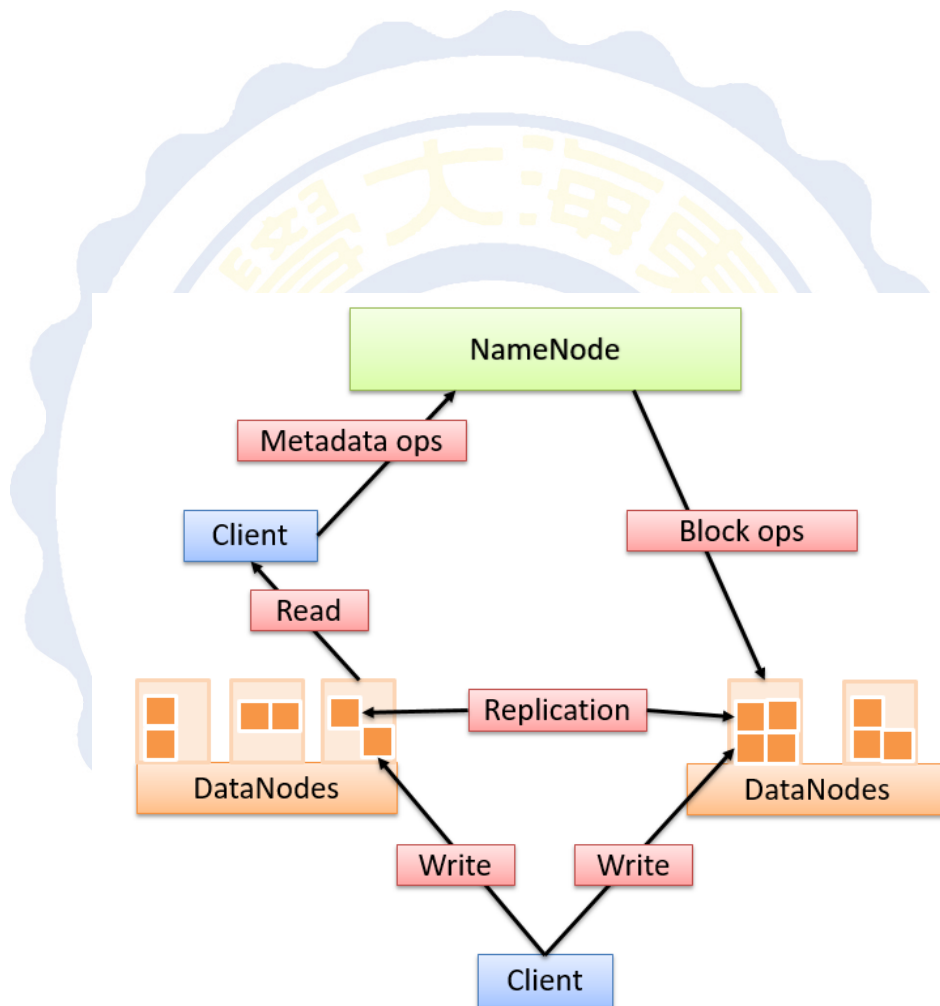


FIGURE 2.4: HDFS architecture

2.1.6 Gluster File System

GlusterFS is an open source distributed file system, it can be scattered storage space together to form a virtual storage pool. It supports scale-out, through an increased number of storage nodes to increase overall system capacity or performance, storage capacity can scale to petabytes. GlusterFS In addition to supporting distributed storage (different files on different storage nodes). It also supports the use of Replicated storage (the same file stored in more than two storage nodes) and Stripped storage (a file is divided into multiple fixed-length data, scattered in all storage nodes).

GlusterFS has the following advantages:

- GlusterFS supports TCP / IP and InfiniBand RDMA high-speed Internet interconnection.
- The client can access the data through the native GlusterFS protocol. Other terminals that are not running the GlusterFS client can access the data through the storage gateway through the NFS / CIFS standard protocol (the storage gateway provides flexible volume management and access agent functions).
- Storage server mainly provides basic data storage function, the client makes up the problem of no metadata server, take on more functions, including data volume management, I / O scheduling, file location, data cache and other functions, the use of FUSE (File System in User Space) module mounts GlusterFS on top of the local file system for POSIX-compatible access to system data.

2.2 Related Works

High Available share storage is very important issue for future development. X. Zhang et al, [21] 2016. Ceph is a distributed file system that provides high performance, reliability, and scalability. Maximize the Ceph intergovernmental data and metadata management replaces the configuration table to virtual random data distribution function designed for heterogeneous and dynamic clusters of unreliable OSDs. They use investigate the performance of Ceph on an Open Stack cloud using well-known benchmarks. They use lot of banchmarks for example Bonnie+, DD (Read and Write), RADOS Bench (Read/Write), Iperf Benchmark, Netcat Benchmark. They results show its good performance and scalability.

Moldoveanu Florica et al, [22] 2013. Cloud computing is becoming increasingly popular, and due to scalability to support rapid economic growth and productivity, its distributed data storage tier needs to be able to meet these requirements. The Ceph storage system contains a large number of cluster nodes and a large number of clients that interact with it. They proposed two acceleration mechanisms based on multi-core network SoCs to maximize the performance of each cluster node.

Chih-Fong Tsai et al, [23] 2016. Parallel and cloud computing platform is considered a very good solution for large data mining. There are two common methods to resolve the data problems. The first is based on data parallel paradigm distributed process in which a given large data sets can manually into n subset, and for the corresponding subset of the n n algorithms are implemented. The end result from n algorithm can be generated output combinations. The second is based on the cloud computing platform mapreduce the process. The process of mapping and the reduction process, of which the former perform filtering and sorting and later the executive summary operation to produce the final result. They are large-scale data-mining mapreduce method and accuracy and efficiency in the performance difference. Lab uses four large data for data classification. The result indicates that the program is based on the classification of mapreduce performance is very stable, regardless of the number of computer nodes are better

than baseline standalone and distributed. Mapreduce process requires the least cost to handle these large data sets.



Chapter 3

System Design and Implementation

This section describes the system architecture and implementation of decentralized storage systems. We store the data in the Hadoop Distributed File System (HDFS) and use Alluxio as a bridge to copy to Alluxio memory space. The result is processed by Map Reduce, and the output is storage in the Alluxio memory space.

3.1 System Architecture

The first experiment, the system uses the word data to load mapping to reduce environment. Data can be adjusted according to the user requirements. Here we set three sample data size: 5GB, 10GB and 15GB. The value key through the map, press the key sort [key, merge, [value-1, value-2, value-n]] algorithm is used to send data to the memory speed Alluxio virtual storage systems. We also RADOS through S3 and activate the Alluxio Gateway File Configuration API (S3 ceph components, data is also stored in the object store daemon (OSD).

For the second experiment, the system and the first system is the same as the amount of data in the system and data not through S3 API and RADOS gateway

to store data to the object store daemon (OSD), the OSD is directly connected to the Alluxio inserted. The second experiment reduces the S3 API and RADOS gateway level. These environmental inkscope by monitoring system monitoring. These environments have monitored by Inkscope monitoring system. Inkscope monitor system in all ceph. If there is any expiration, Inkscope will display the user's alert. Figure 3.1 shows the integration architecture.

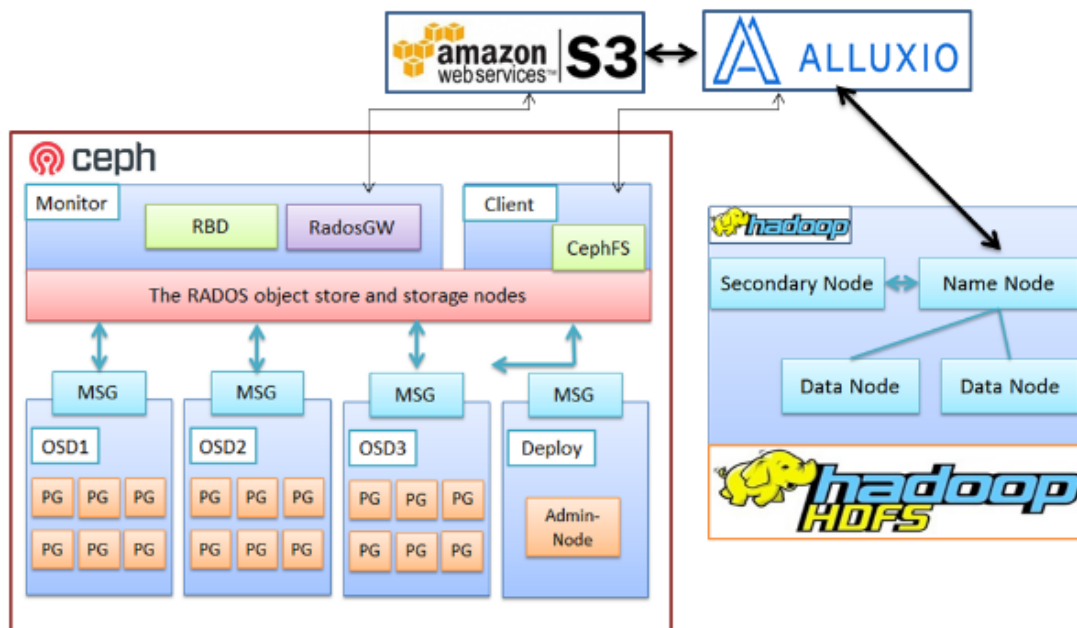


FIGURE 3.1: Hadoop, Alluxio and Ceph integration architecture

TABLE 3.1: Software Specification

No.	Description	Version
1	Apach Hadoop	2.7.3
2	Alluxio	1.4
3	Ceph	10.20(jewel)
4	Inkscope	1.1
5	Mongo DB	3.2
6	GlusterFS	3.6.9

3.2 System Implementation

In this work, we set up Apache Hadoop, Alluxio, Ceph, Inkscope, and GlusterFS. Table 3.1 shows the specifications of used software.

3.2.1 Ceph Storage Service Deployment

Ceph is a free-software storage platform, implements distributed object storage and file system, and provides interfaces for object-, block- and file-level storage. It has excellent performance, reliability and scalability. To achieve the functions above, Ceph has three kind of physical nodes: Object Storage Daemon (OSD), Monitors (MON) and Metadata (MDS) service. According to object storage deployment requirements. We only need to install OSDs and MONs. The overview of our Ceph architecture is shown in Figure 3.2. The Figure 3.3 shows the OSD work up. In the first time we installed ceph, we encountered insufficient hard disk space, so we have increased in the three OSD 20GB. As shown in Figure 3.4. In the ceph version we tried to use three different versions (Hammer, Infernalis, Jewel). We found that using Hammer and Infernalis versions can not be integrated with Hadoop and Alluxio. So we finally use the Jewel version.

3.2.2 HDFS Deployment

Hadoop has two kinds of nodes: master node and slave node. Master node uses NameNode service to control DataNode service which is running on slave nodes.

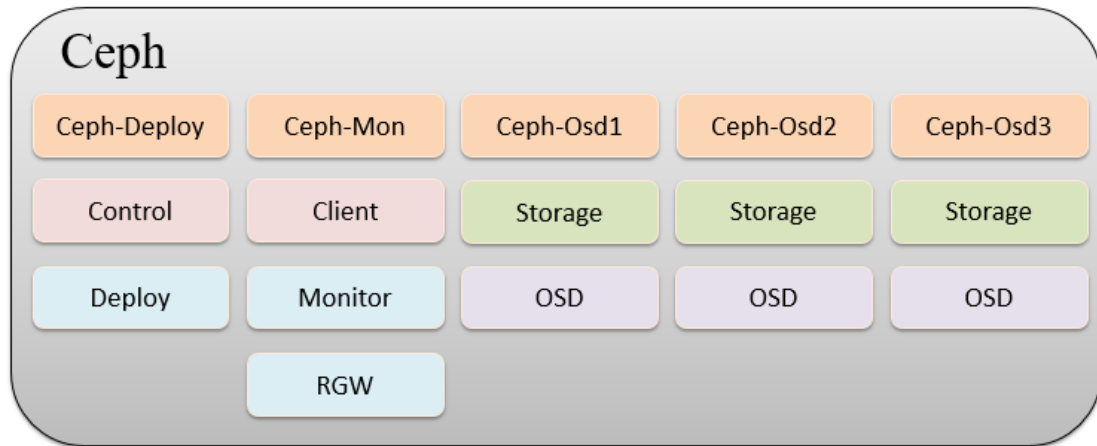


FIGURE 3.2: Ceph environment

```

userceph@deploy:~/ceph$ ceph osd tree
ID WEIGHT  TYPE NAME          UP/DOWN REWEIGHT  PRIMARY-AFFINITY
-1 0.05846  root default
-2 0.01949  host osd1
 0 0.01949  osd.0          up  1.00000  1.00000
-3 0.01949  host osd2
 1 0.01949  osd.1          up  1.00000  1.00000
-4 0.01949  host osd3
 2 0.01949  osd.2          up  1.00000  1.00000
userceph@deploy:~/ceph$ █

```

FIGURE 3.3: Ceph Osd tree

We built a HDFS architecture consisting of one master node and two slave nodes, as shown in Figure 3.5 and Figure 3.6. The NameNode executes file system namespace operations and also determines the mapping of blocks DataNodes. DataNodes are responsible for serving read and write requests from clients of file system. After installing Hadoop and HDFS, we must set the S3 API in the Hadoop environment and add the code in the core-site.xml.

3.2.3 Alluxio Deployment

Alluxio data storage and computing separation, two-part engine can be independently extended. The calculation engine (Hadoop) can access data from different data sources (Amazon S3). We deployment Alluxio in the Hadoop node, and then through Alluxio as a bridge to access the S3 API and Rados Gateway. As show in


```

root@osd1:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            477M  4.0K  477M   1% /dev
tmpfs           98M   868K   97M   1% /run
/dev/dm-0       19G   1.3G   17G   8% /
none            4.0K   0    4.0K   0% /sys/fs/cgroup
none            5.0M   0    5.0M   0% /run/lock
none            488M   0    488M   0% /run/shm
none            100M   0    100M   0% /run/user
/dev/sda1       236M   39M   185M  18% /boot
/dev/sdb        20G    33M   20G   1% /mnt/osd0
root@osd1:~#

```

FIGURE 3.4: Add a hard drive

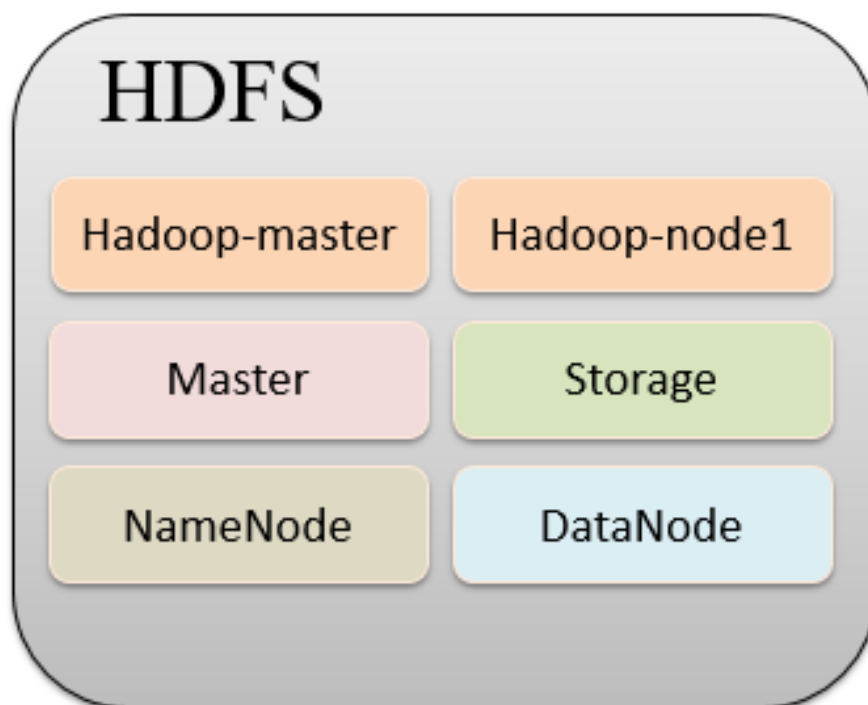


FIGURE 3.5: HDFS environment

Figure 3.7 and Figure 3.8. After integrating Hadoop and Alluxio, we use the Word Count example to test, we see the Output state is NOT-PERSISTED as show the Figure 3.9. After the data is transferred into the state becomes PERSISTED as show the Figure 3.10. We can see the OSD space saved data in the Figure 3.11.

```
userceph@hadoop:~/hadoop$ jps
45836 DataNode
46164 ResourceManager
45677 NameNode
46418 NodeManager
46626 Jps
46025 SecondaryNameNode
userceph@hadoop:~/hadoop$
```

FIGURE 3.6: Hadoop jps

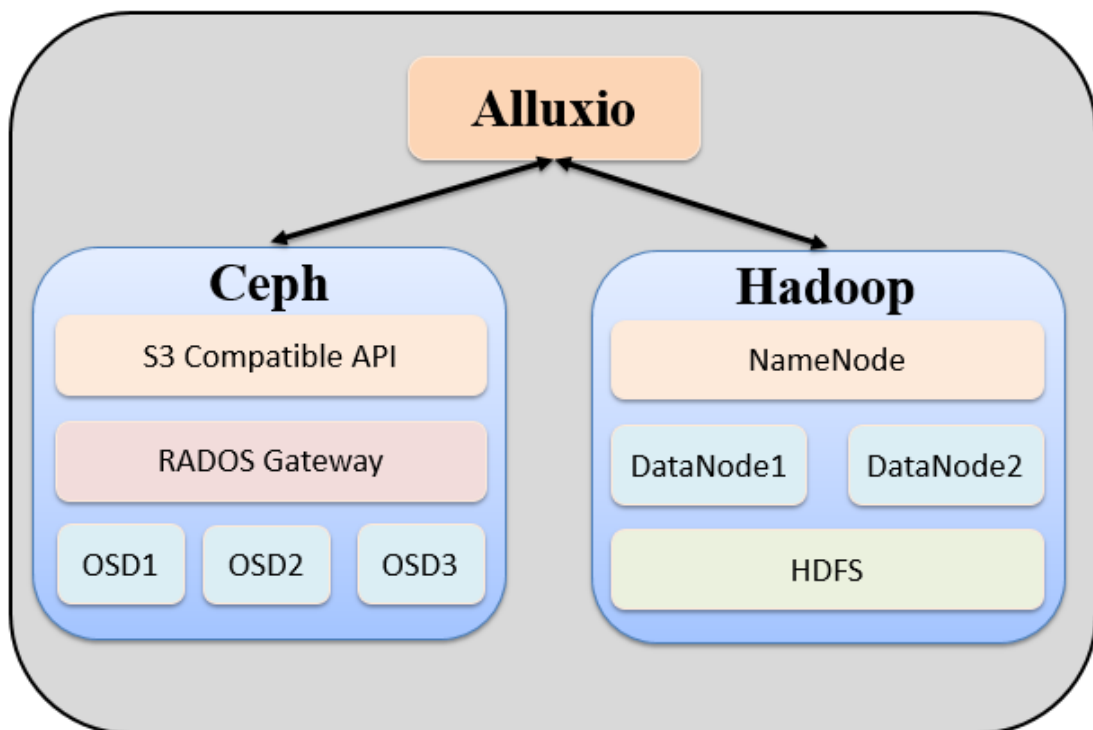


FIGURE 3.7: Alluxio environment

3.2.4 Rados Gateway Deployment

Also known as Ceph Object Storage. Provide RESTful API interface, compatible with Amazon S3 cloud storage services, and OpenStack object storage Swift. Through the Rados Gateway access to the Ceph storage cluster, mainly through the LIBRGW this library, you can achieve direct access to the effect. As show in 3.12 and Figure 3.13.

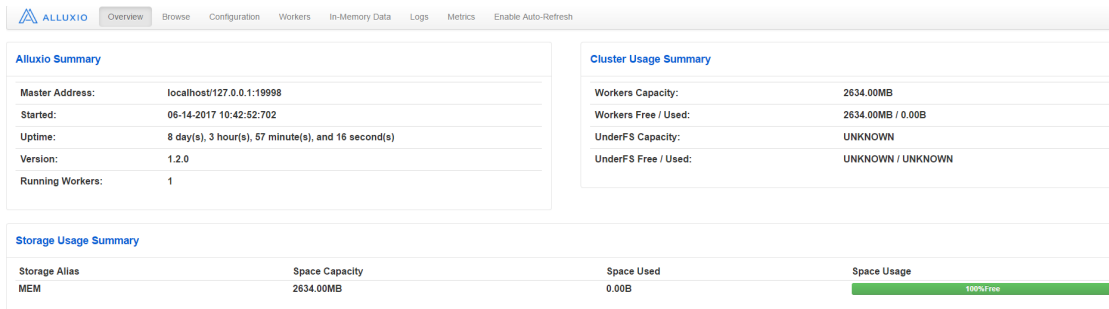


FIGURE 3.8: Alluxio instances

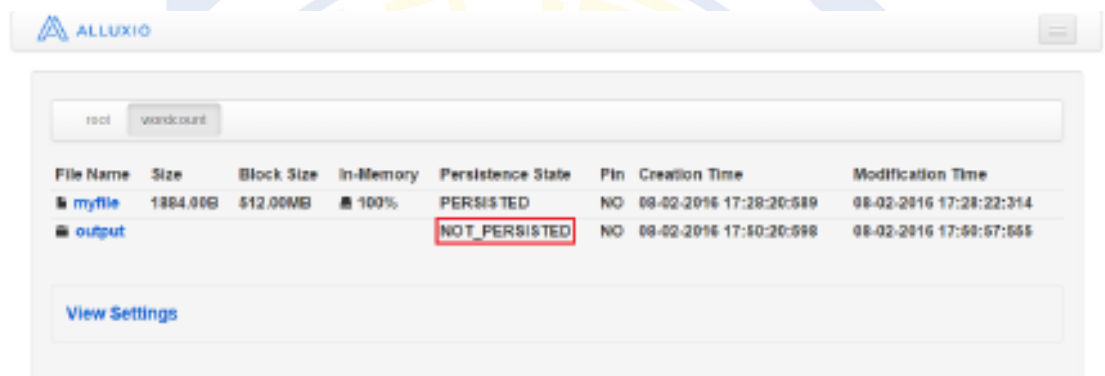


FIGURE 3.9: Output state before

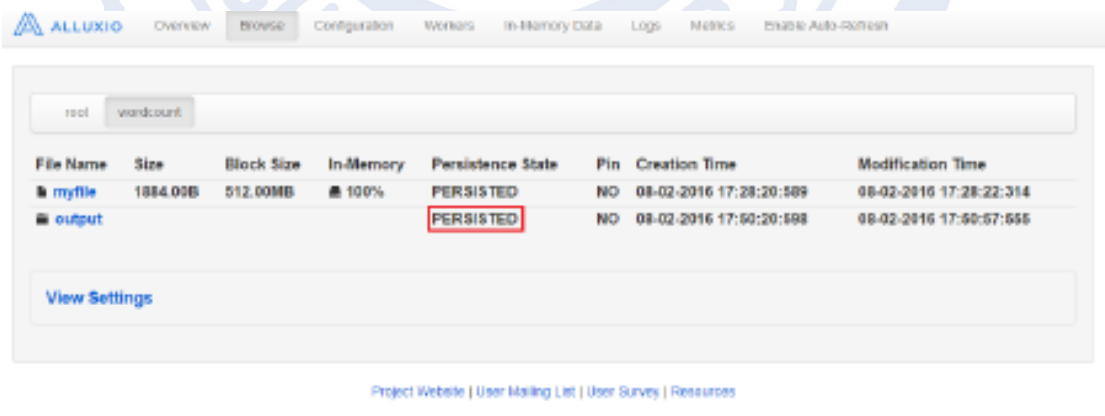


FIGURE 3.10: Output state after

3.2.5 Ceph File System Deployment

Ceph File System(CephFS) provides POSIX compatible file systems for users to mount files or folders in two ways: access to CephFS's core object, or the user space under the file system (Filesystem in User Space) and use. First we build CephFS mount up. Second we mount CephFS via Alluxio File System. As show in Figure 3.14. We will put 1.5G files into CephFS, and then observe the CephFS

```

userceph@mon1:~$ python s3ListObject.py
userceph@mon1:~$ python s3ListObject.py
test/wordcount/output/_SUCCESS test/wordcount/output/_SUCCESS 2016-08-02T10:22:39.121Z
test/wordcount/output/part-r-00000 test/wordcount/output/part-r-00000 2016-08-02T10:22:39.542Z
test/wordcount/output_$folder$ test/wordcount/output_$folder$ 2016-08-02T10:22:39.023Z
test/wordcount_$folder$ test/wordcount_$folder$ 2016-08-02T10:22:38.979Z
test_$folder$ test_$folder$ 2016-08-02T10:22:38.911Z
userceph@mon1:~$

```

FIGURE 3.11: OSD space saved data

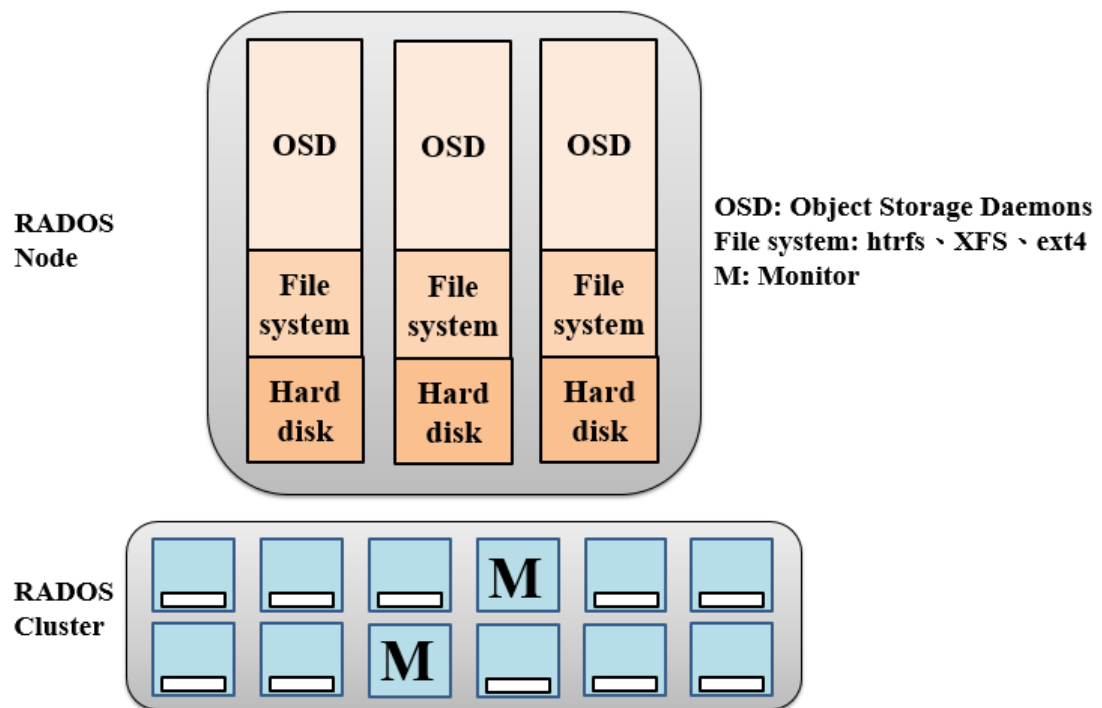


FIGURE 3.12: Rados Gateway environment

space changes, from the figure 3.15 you can see the CephFS space has indeed been eaten. In the Alluxio WebUI to see if the file is passed. As show the Figure 3.16 and 3.17

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>anonymous</ID>
    <DisplayName/>
  </Owner>
  <Buckets/>
</ListAllMyBucketsResult>

```

FIGURE 3.13: Rados Gateway instances

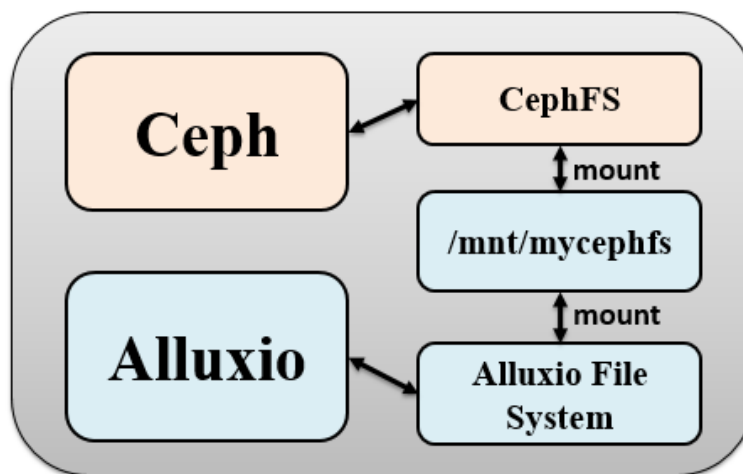


FIGURE 3.14: Ceph File System instances

```

hduser@namenode:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.7G  4.0K  7.7G   1% /dev
tmpfs           1.6G  904K  1.6G   1% /run
/dev/dm-0       902G  4.5G  851G   1% /
none            4.0K   0  4.0K   0% /sys/fs/cgroup
none            5.0M   0  5.0M   0% /run/lock
none            7.7G   0  7.7G   0% /run/shm
none            100M   0  100M   0% /run/user
/dev/sda1       236M   74M  151M  33% /boot
172.24.12.82:6789:/ 2.2T   16G  2.2T   1% /home/hduser/alluxio/mnt/mycephfs
hduser@namenode:~$ ls
1.5G.txt  alluxio  alluxio-1.3.0-hadoop2.7-bin.tar.gz
hduser@namenode:~$ sudo cp 1.5G.txt /home/hduser/alluxio/mnt/mycephfs
hduser@namenode:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.7G  4.0K  7.7G   1% /dev
tmpfs           1.6G  904K  1.6G   1% /run
/dev/dm-0       902G  4.5G  851G   1% /
none            4.0K   0  4.0K   0% /sys/fs/cgroup
none            5.0M   0  5.0M   0% /run/lock
none            7.7G   0  7.7G   0% /run/shm
none            100M   0  100M   0% /run/user
/dev/sda1       236M   74M  151M  33% /boot
172.24.12.82:6789:/ 2.2T   20G  2.2T   1% /home/hduser/alluxio/mnt/mycephfs
hduser@namenode:~$ █

```

FIGURE 3.15: CephFS spatial changes

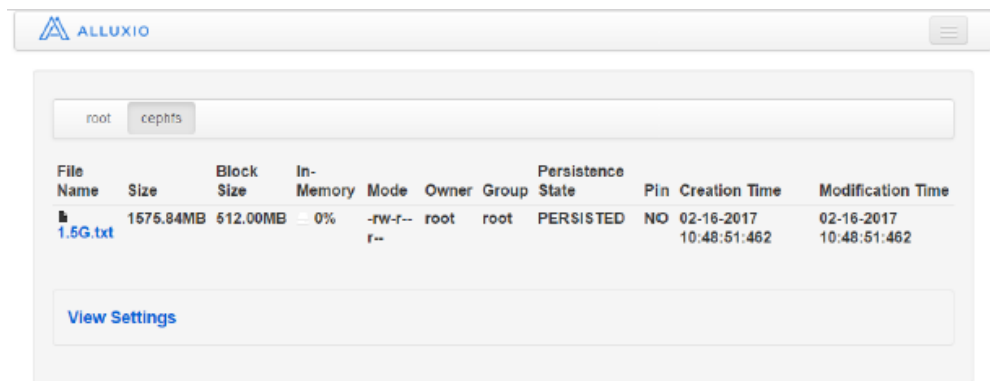


FIGURE 3.16: Alluxio WebUI before upload

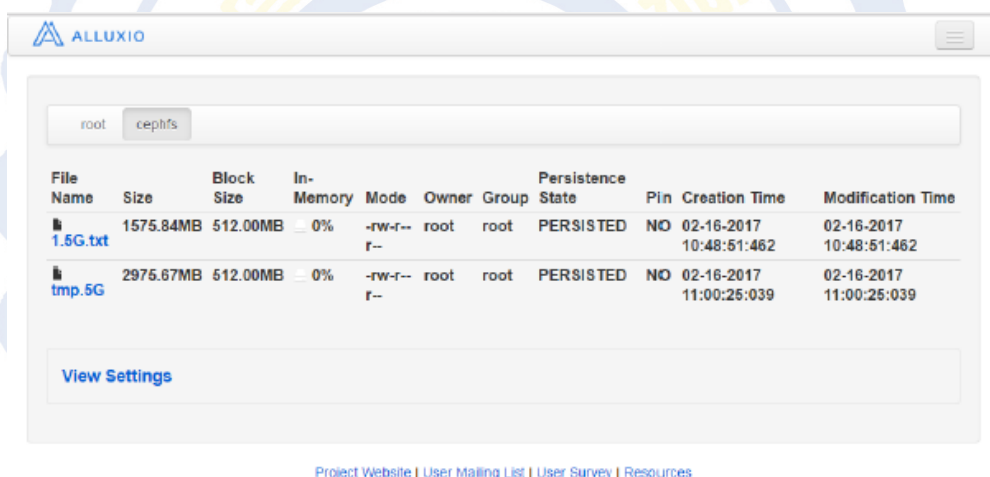


FIGURE 3.17: Alluxio WebUI after upload

3.2.6 User Services

We deploy the Inkscope system to monitor Ceph environment. Inkscope monitors all server hardware, networks, pools, and services. We also use MongoDB to store real-time metrics and historical metrics. As show in Figure 3.18.

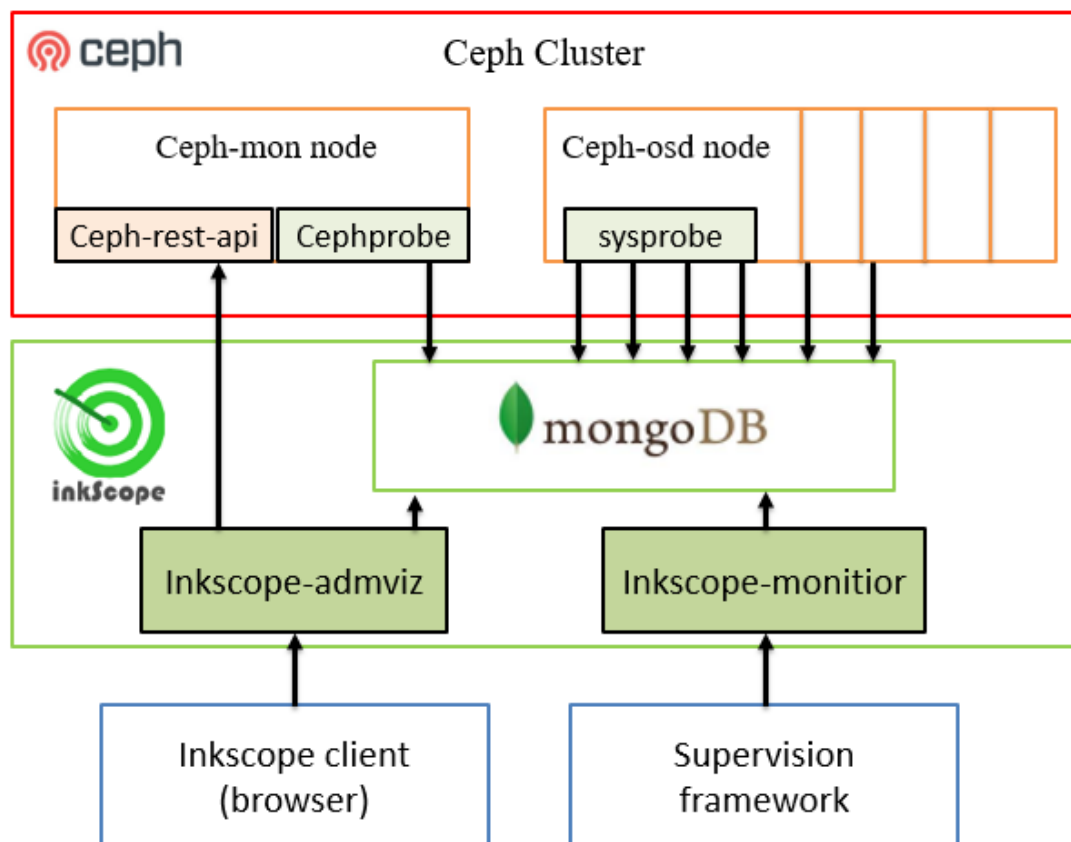


FIGURE 3.18: Inkscope environment

Chapter 4

Experimental Results

4.1 Experimental Environment

For the hardware specification of the computer that we use 6 servers for Ceph, 1 server for Inkscope, 1 server for Hadoop and Alluxio. These servers are the physical machine. We use 64-bit Ubuntu 14.04 as our operating system because this version is relatively stable relative to other versions. Figure 4.1 shows us the environment. Table 4.1 and Table 4.2 shows the hardware specification of the computer.



FIGURE 4.1: Hardware environment

TABLE 4.1: Experimental hardware description

Component	Sub-Component	Total server	server
Ceph	Deploy	1Unit	Deploy
	Monitior	1Unit	Mon1
	OSD	3Unit	Osd1 Osd2 Osd3
Hadoop Alluxio	Name Node Master Node	1Unit	namenode
	Data Node Worker Node	1Unit	datanode
Inkscope	Inkscope	1Unit	inkscope
Gluster	Gluster1	2Unit	glusterfs1
	Gluster2		glusterfs2

TABLE 4.2: Experimental hardware specifications

Hardware component	
CPU	Inter(R) Core(TM)i7-4770 CPU @3.40GHz 8Cores
RAM	16GB
HDD	1TB
OS	Ubuntu14.04 LTS

4.2 Experimental Results

4.2.1 MapReduce Input and Output

We show real-time Map Reduce in the Hadoop environment. The source data is stored in HDFS and then copied to the Alluxio memory space environment. The data is stored in the path: alluxio/wordcount/myfile

Figure 4.2 show the Data Source in Alluxio.

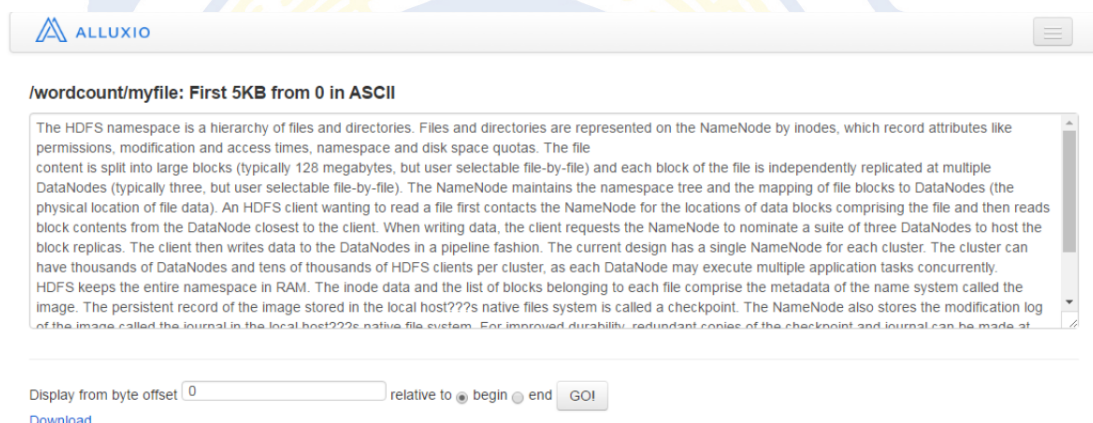


FIGURE 4.2: Data Source in Alluxio


Figure 4.3 and Figure 4.4 is the output result stored in the Alluxio memory.

The screenshot shows the Alluxio web interface. At the top, there is a header with the Alluxio logo and navigation tabs: Overview, Browse, Configuration, Workers, In-Memory Data, Logs, Metrics, and Enable Auto-Refresh. Below the header, there is a breadcrumb trail: root > wordcount > output. The main content area contains a table with the following data:

File Name	Size	Block Size	In-Memory	Persistence State	Pin	Creation Time	Modification Time
_SUCCESS	0.00B	512.00MB	100%	PERSISTED	NO	08-02-2016 17:50:57:555	08-02-2016 18:22:39:370
part-r-00000	1437.00B	512.00MB	100%	PERSISTED	NO	08-02-2016 17:50:54:639	08-02-2016 18:22:39:601

Below the table, there is a "View Settings" button. At the bottom of the page, there is a footer with links: Project Website | User Mailing List | User Survey | Resources.

FIGURE 4.3: Map Reduce Output 1



```
userceph@hadoop:~/alluxio-1.2.0$ bin/alluxio fs cat /wordcount/output/part-r-00000
(the      1
(typically  2
128      1
An       1
DataNode  2
DataNodes 5
During   1
Files    1
For      1
HDFS     4
NameNode  7
RAM.     1
The      10
When     1
a        6
access   1
also     1
and      12
application 1
are      2
as       1
at       2
attributes 1
be       1
```

FIGURE 4.4: Map Reduce Output 2

All the MapReduce data have been store in to the Ceph OSDs.Used S3 API and Rados Gateway as the first way and directly CephFS for the second way. Figure 4.5 show the Ceph OSD Output.



File Name	Size	Block Size	In-Memory	Persistence State	Pin	Creation Time	Modification Time
BasicNonByteBuffer_CACHE_ASYNC_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:15:948	05-25-2016 02:16:17:261
BasicNonByteBuffer_CACHE_CACHE_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:13:881	05-25-2016 02:16:14:353
BasicNonByteBuffer_CACHE_MUST_CACHE	84.00B	512.00MB	100%	NOT_PERSISTED	NO	05-25-2016 02:16:13:243	05-25-2016 02:16:13:261
BasicNonByteBuffer_CACHE_PROMOTE_ASYNC_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:13:072	05-25-2016 02:16:14:262
BasicNonByteBuffer_CACHE_PROMOTE_CACHE_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:10:046	05-25-2016 02:16:10:854
BasicNonByteBuffer_CACHE_PROMOTE_MUST_CACHE	84.00B	512.00MB	100%	NOT_PERSISTED	NO	05-25-2016 02:16:08:512	05-25-2016 02:16:08:517
BasicNonByteBuffer_CACHE_PROMOTE_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:11:955	05-25-2016 02:16:12:472
BasicNonByteBuffer_CACHE_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:15:187	05-25-2016 02:16:15:701
BasicNonByteBuffer_NO_CACHE_ASYNC_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:19:661	05-25-2016 02:16:21:264
BasicNonByteBuffer_NO_CACHE_CACHE_THROUGH	84.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:17:390	05-25-2016 02:16:17:920
BasicNonByteBuffer_NO_CACHE_MUST_CACHE	84.00B	512.00MB	100%	NOT_PERSISTED	NO	05-25-2016 02:16:16:047	05-25-2016 02:16:16:060
BasicNonByteBuffer_NO_CACHE_THROUGH	84.00B	512.00MB	0%	PERSISTED	NO	05-25-2016 02:16:18:747	05-25-2016 02:16:19:410
Basic_CACHE_ASYNC_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:15:868	05-25-2016 02:16:17:261
Basic_CACHE_CACHE_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:13:310	05-25-2016 02:16:13:791
Basic_CACHE_MUST_CACHE	80.00B	512.00MB	100%	NOT_PERSISTED	NO	05-25-2016 02:16:13:116	05-25-2016 02:16:13:132
Basic_CACHE_PROMOTE_ASYNC_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:12:687	05-25-2016 02:16:14:268
Basic_CACHE_PROMOTE_CACHE_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:08:555	05-25-2016 02:16:09:949
Basic_CACHE_PROMOTE_MUST_CACHE	80.00B	512.00MB	100%	NOT_PERSISTED	NO	05-25-2016 02:16:08:261	05-25-2016 02:16:08:388
Basic_CACHE_PROMOTE_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:10:863	05-25-2016 02:16:11:400
Basic_CACHE_THROUGH	80.00B	512.00MB	100%	PERSISTED	NO	05-25-2016 02:16:14:379	05-25-2016 02:16:14:829

FIGURE 4.5: Ceph OSD Output

4.2.2 Inkscope Monitoring System

The following is the Inkscope monitoring system for the Ceph environment. Inkscope monitors all server hardware, networks, pools, and services as follows: This page you can see that are two hard drives as object storage daemon 10 pool, 136 configuration group (placement group). When the user data will be saved to the cluster, each object must be mapped to a configuration group placement group and every one configuration group placement group will be mapped to a OSD, OSD is one of the other primary, is backup(replica). Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9 show the Inkscope Ceph status.



FIGURE 4.6: Inkscope Ceph Login Page

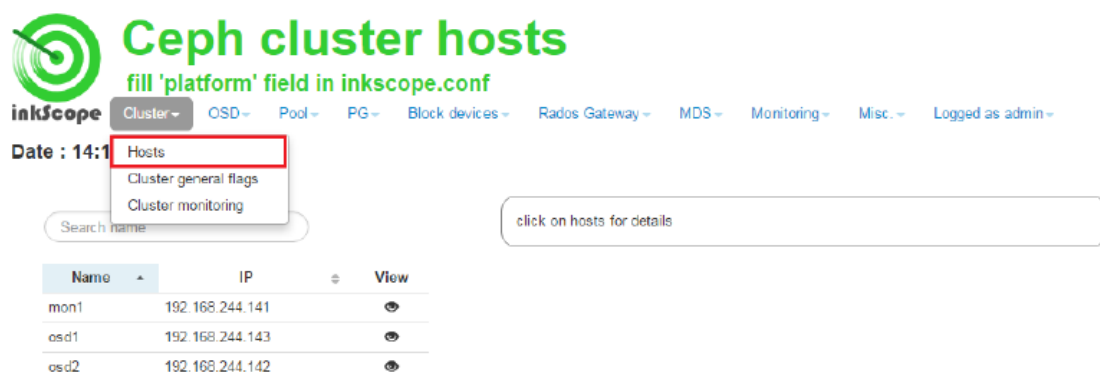


FIGURE 4.7: Inkscope Ceph cluste hosts

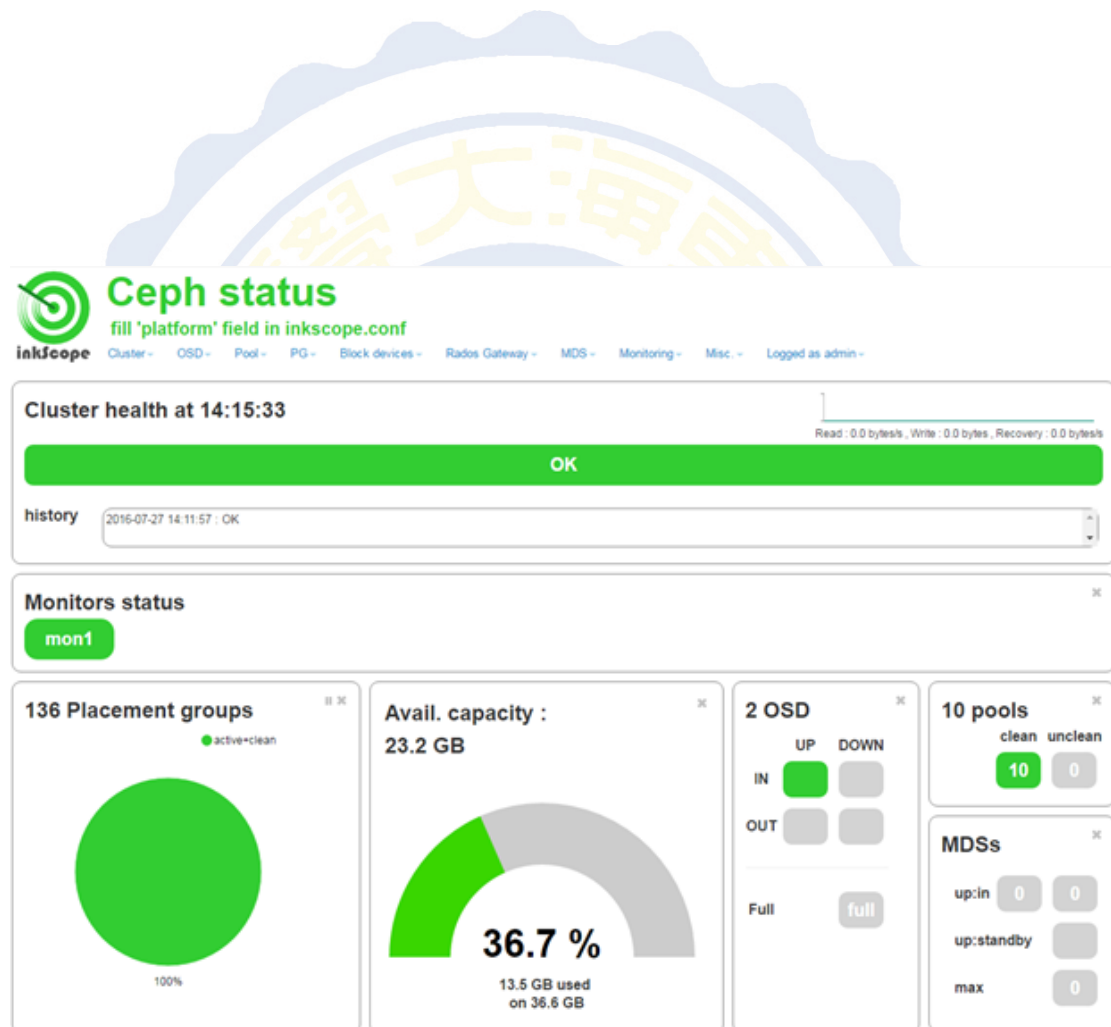


FIGURE 4.8: Inkscope Ceph status

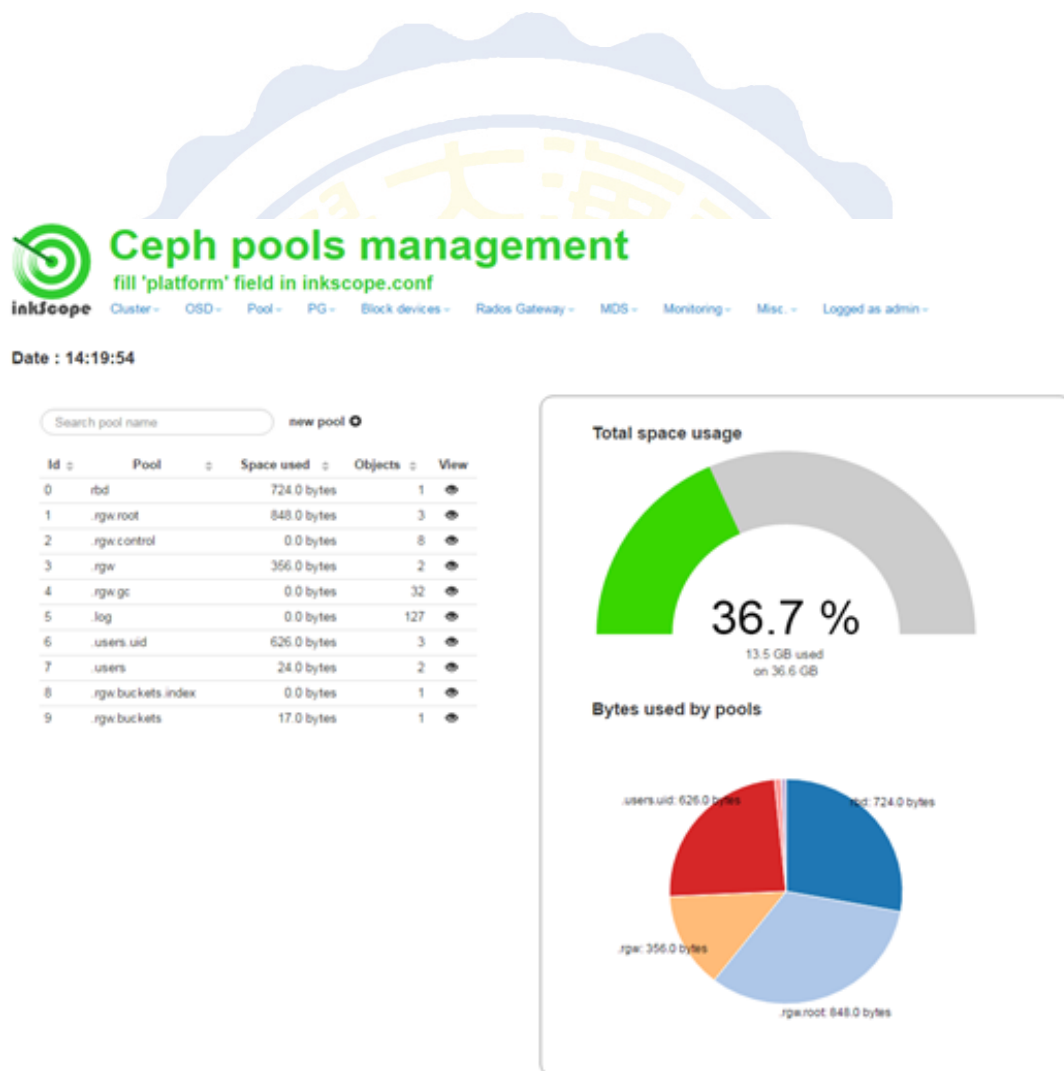


FIGURE 4.9: Inkscope Ceph Pools Management

If the OSD server in all systems has insufficient space or any problems, the Inkscope system will notify you. Figure 4.10, Figure 4.11 shows the osd status. Figure 4.12 shows the notification interface.



FIGURE 4.10: Inkscope Ceph osd page

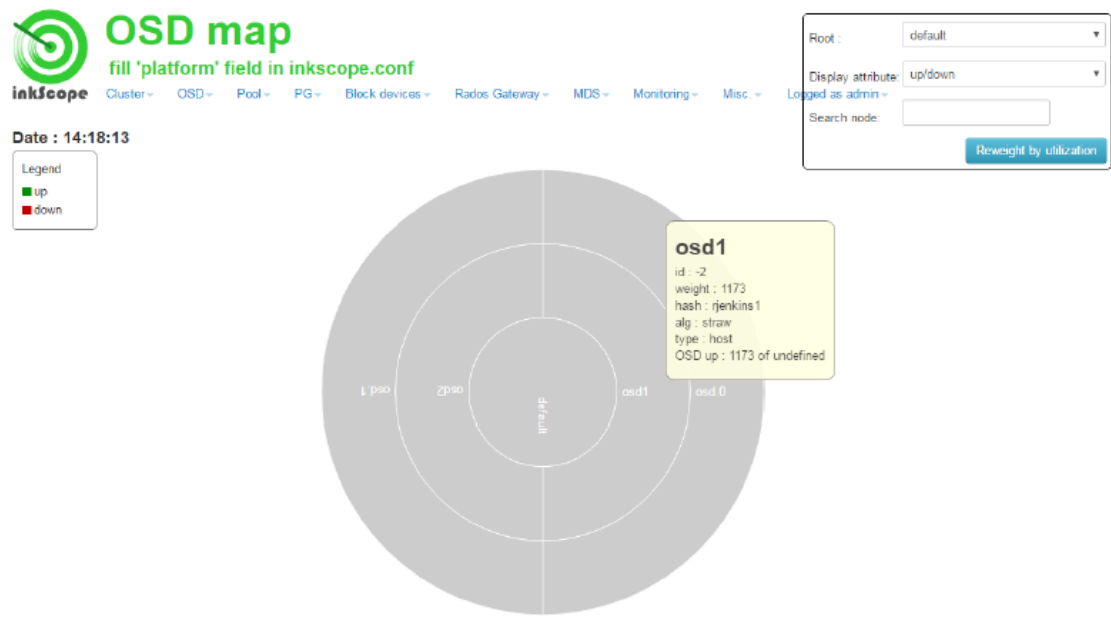


FIGURE 4.11: Inkscope Ceph osd map

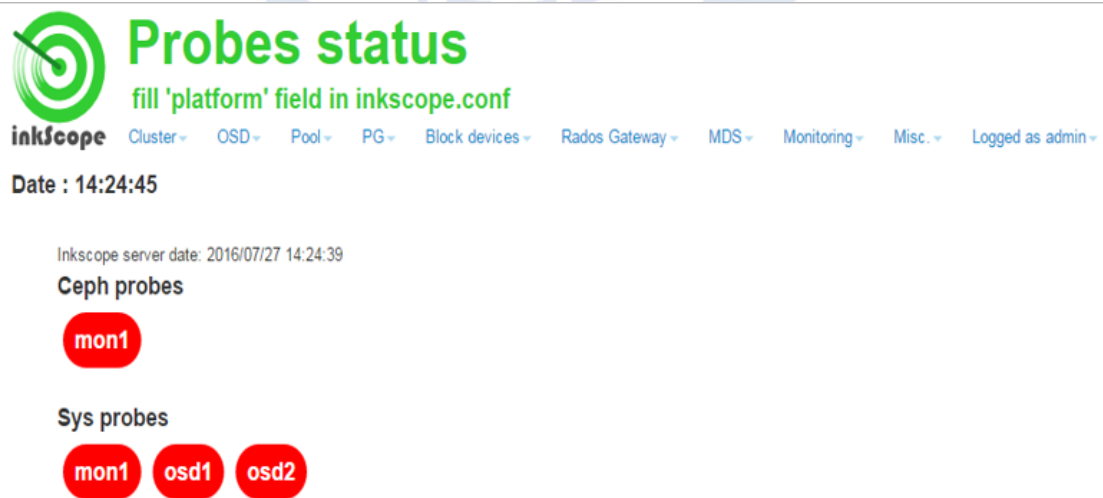


FIGURE 4.12: Inkscope Ceph Pools Management2

TABLE 4.3: Performance Comparison Test with Read, Write, Randomize for each OSDs

	Osd1			Osd2			Osd3		
	5G	10G	15G	5G	10G	15G	5G	10G	15G
S.Read	1.8	1.9	1.9	2.2	2.0	2.1	2.0	2.1	1.9
S.Write	24.0	25.7	24.8	26.6	24.0	24.0	18.1	20.5	19.9
Rand.Read	1.3	1.1	1.0	1.2	1.2	1.1	1.3	1.1	1.1
Rand.Write	2.6	2.5	2.4	2.6	2.5	2.3	2.3	2.3	1.9
S.Read(30%)	2.0	2.1	2.1	2.1	2.0	3.4	6.4	6.7	5.8
S.Write(30%)	0.8	0.9	1.5	0.9	0.8	1.5	2.8	2.8	2.5
Rand.Read(30%)	1.1	1.0	0.9	1.1	1.0	0.9	1.1	0.9	0.9
Rand.Write(30%)	0.4	0.4	0.3	0.4	0.4	0.3	0.4	0.3	0.3

TABLE 4.4: Performance Comparison Test with Read, Write, Randomize for CephFS

	CephFS		
	5G	10G	15G
S.Read	19.8	1.9	1.9
S.Write	0.7	25.7	24.8
Rand.Read	16.1	16.9	16.9
Rand.Write	0.5	0.5	0.5
S.Read(30%)	1.2	1.3	1.3
S.Write(30%)	0.5	0.5	0.6
Rand.Read(30%)	1.1	0.9	0.7
Rand.Write(30%)	0.4	0.4	0.3

4.2.3 The Performance between Rados Gateway and CephFS Comparison

We use FIO tool to benchmark. Using Rados Gateway and CephFS. Table 4.3 and Table 4.4 is our experimental result. The first way, three OSDs is using Rados Gateway and S3 API way to store the data in to OSDs. The second way is through CephFS stored the data to OSDs. According to experiment.

The Figure 4.13 shows the Osd and CephFS in sequential read and write access comparison. We observe from the figure in terms of reading CephFS is better than Osd. In terms of writing is different.

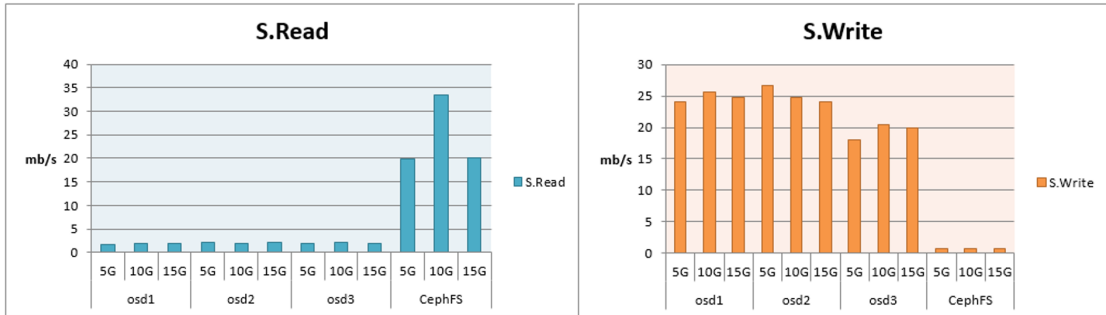


FIGURE 4.13: Speed Performance Test with S.Read/Write for each OSD and CephFS

The Figure 4.14 shows the Osd and CephFS in random read and write access comparison. We observe from the figure in terms of reading CephFS is apparently excellent than Osd. In terms of writing is different.

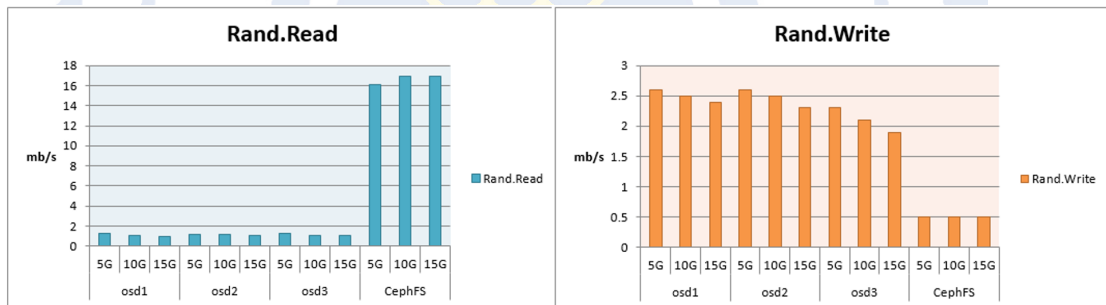


FIGURE 4.14: Speed Performance Test with Randomize Read/Write for each OSD and CephFS

The Figure 4.15 shows the Osd and CephFS in sequential read(30%) and write(30%) access comparison. We observe from the figure that the osd Slightly better than CephFS.

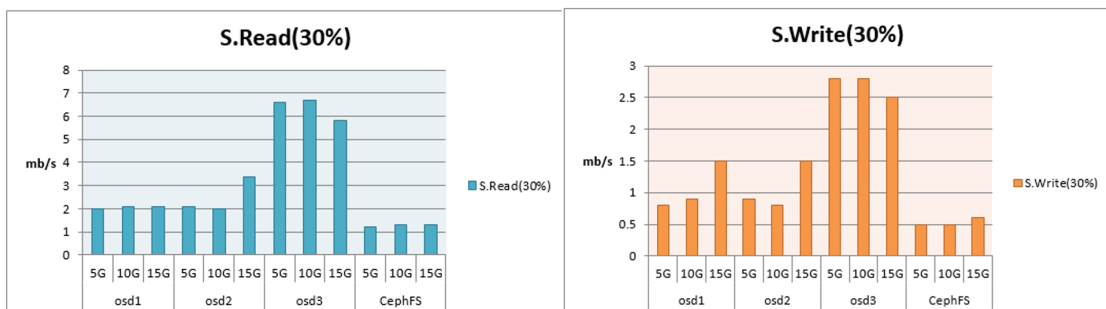


FIGURE 4.15: Speed Performance Test with S.Read/Write(30%) for each OSD and CephFS

The Figure 4.16 shows the Osd and CephFS in random read(30%) and write(30%) access comparison. We observe from the figure that no significant difference.

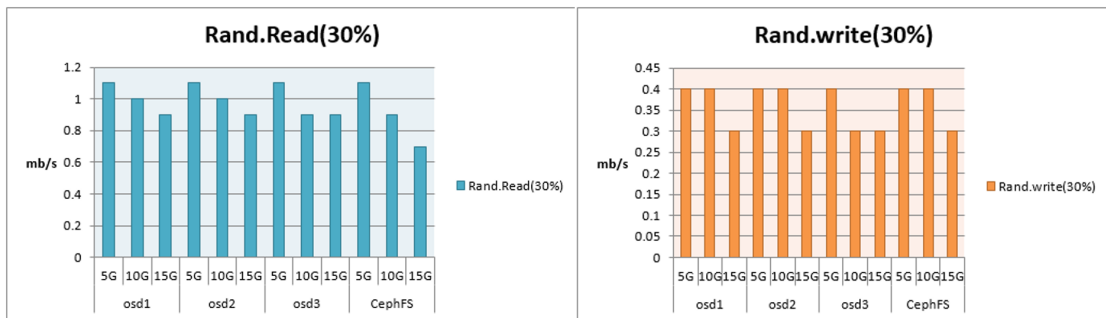


FIGURE 4.16: Speed Performance Test Randomize Read/Write(30%) for each OSD and CephFS

TABLE 4.5: IOPS Performance Test with Read, Write and Randomize for each OSDs

	Osd1			Osd2			Osd3		
	5G	10G	15G	5G	10G	15G	5G	10G	15G
S.Read	118	112	121	137	127	134	125	132	120
S.Write	1504	1609	1552	1664	1545	1505	1132	1281	1246
Rand.Read	82	74	67	78	75	71	85	72	71
Rand.Write	166	158	150	163	157	149	144	132	124
S.Read(30%)	130	133	131	132	126	217	416	424	368
S.Write(30%)	54	56	94	55	53	94	176	179	157
Rand.Read(30%)	71	62	60	73	64	59	68	58	58
Rand.Write(30%)	28	25	24	29	25	24	27	21	23

TABLE 4.6: IOPS Performance Comparison Test with Read, Write, Randomize for CephFS

	CephFS		
	5G	10G	15G
S.Read	1238	2084	1260
S.Write	40	40	40
Rand.Read	1006	1058	1056
Rand.Write	31	29	27
S.Read(30%)	74	79	83
S.Write(30%)	30	32	34
Rand.Read(30%)	65	56	43
Rand.Write(30%)	26	22	17

As shown in the above Table 4.5 and Table 4.6. The first way, three OSDs is using Rados Gateway and S3 API way to store the data in to OSDs. We measured each OSD speed and got above result. The second way is through CephFS stored the data to OSDs. According to experiment, we can conclude that the performance of the CephFS than Rados Gateway and S3 API. The measurement results in IOPS is the same. When the value, higher the better read and write performance.

The Figure 4.17 shows the Osd and CephFS in sequential read and write access comparison. We observe from the figure as same as speed test, in terms of reading CephFS is better than Osd. In terms of writing is different.

The Figure 4.18 shows the Osd and CephFS in random read and write access comparison. We observe from the figure in terms of reading CephFS is significant better than Osd.

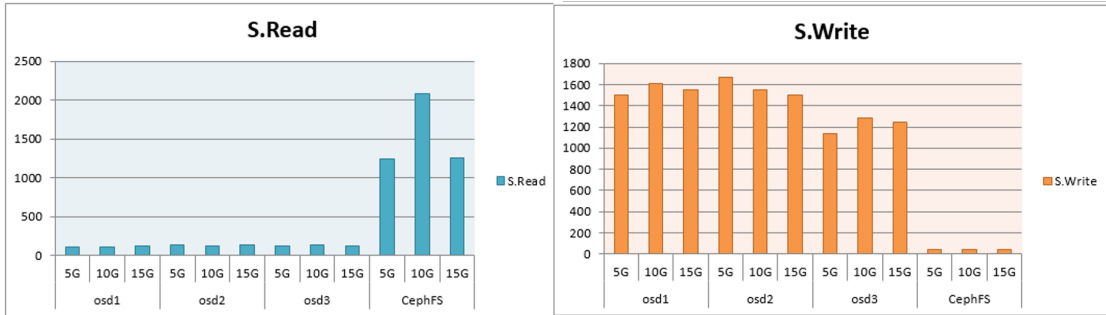


FIGURE 4.17: IOPS Performance Test with S.Read/Write for each OSD and CephFS

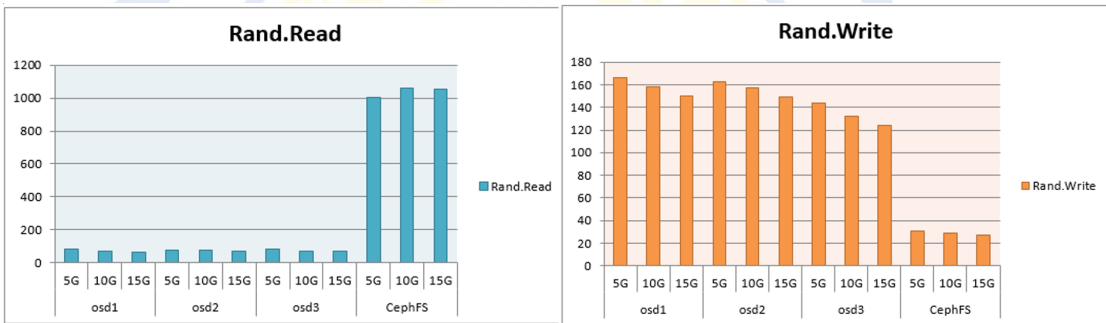


FIGURE 4.18: IOPS Performance Test with Randomize Read/Write for each OSD and CephFS

The Figure 4.19 shows the Osd and CephFS in sequential read(30%) and write(30%) access comparison. We observe from the figure in terms of reading Osd is Slightly better than CephFS.

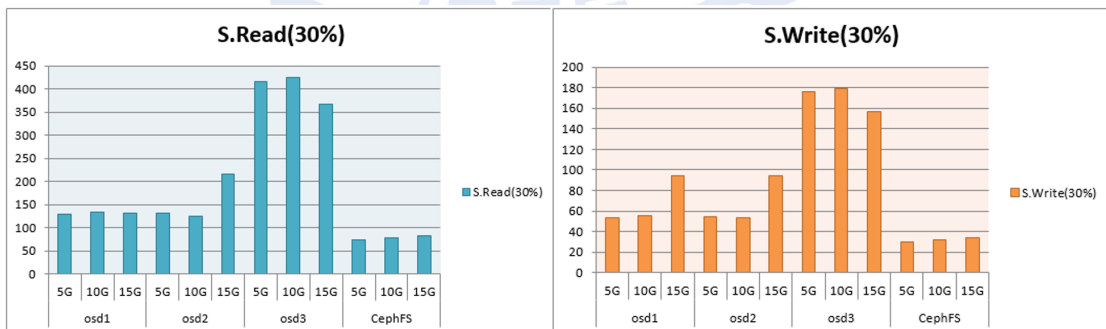


FIGURE 4.19: IOPS Performance Test with S.Read/Write(30%) for each OSD and CephFS

The Figure 4.20 shows the Osd and CephFS in random read(30%) and write(30%) access comparison. We observe from the figure in terms of reading and writing are not significant difference.

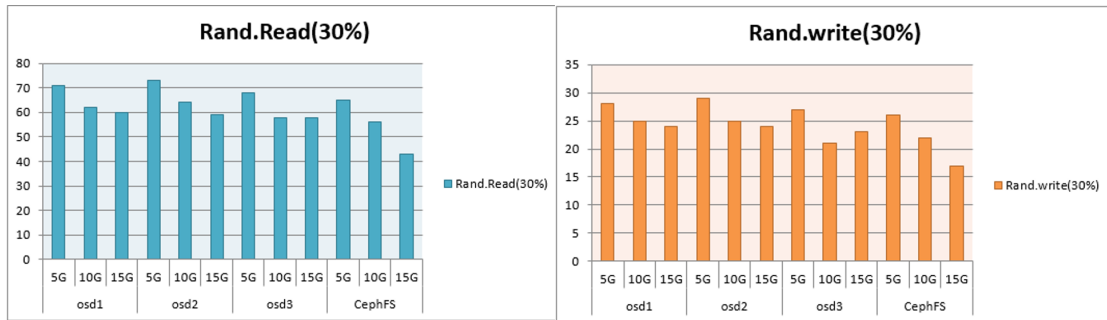


FIGURE 4.20: IOPS Performance Test with Randomize Read/Write(30%) for each OSD and CephFS

4.2.4 The Performance among Rados Gateway, CephFS and GlusterFS Comparison

This experiment compares GlusterFS, Rados Gateway and CephFS three file systems to record the time required for file access. In the client to create a single file on the file size of 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB, 4GB, 8GB, 16GB, 32GB of access performance test.

The Figure 4.21 above is the time it takes for different sizes of files to be uploaded from the client to the distributed file system. We observe from the figure be clearly. When the file is less than 512MB, the time is no significant difference. Starting from 512MB, it is significant that GlusterFS and OSD upload time spent almost twice as much as CephFS, So this figure shows that CephFS is superior to GlusterFS and OSD in file storage.

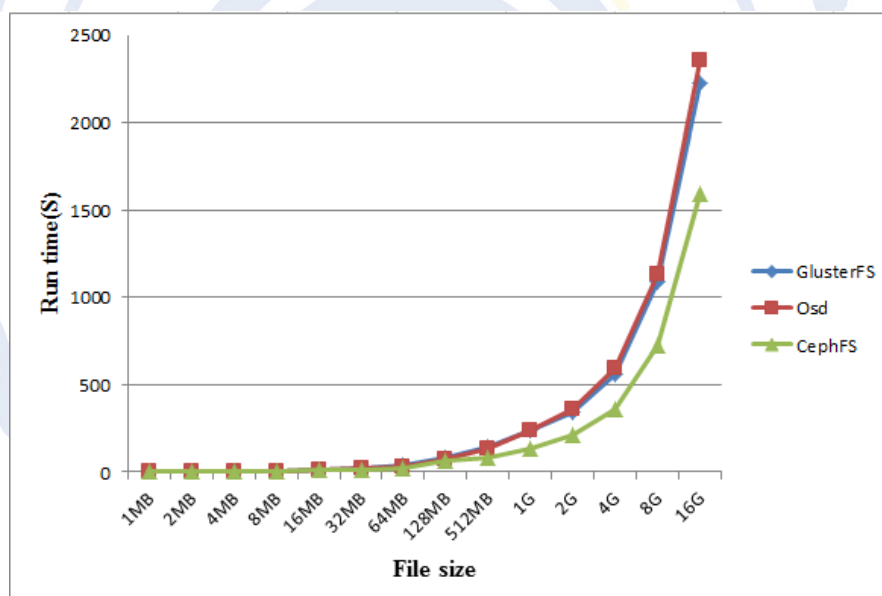


FIGURE 4.21: Performance Test with read for Rados Gateway, CephFS and GlusterFS

The experiment in Figure 4.22 shows the time it takes for different sizes of files to be downloaded from the distributed file system to the client. This figure suggest at 2GB had a Slight difference. Starting from 4GB, CephFS had better performance than Gluster and Osd.

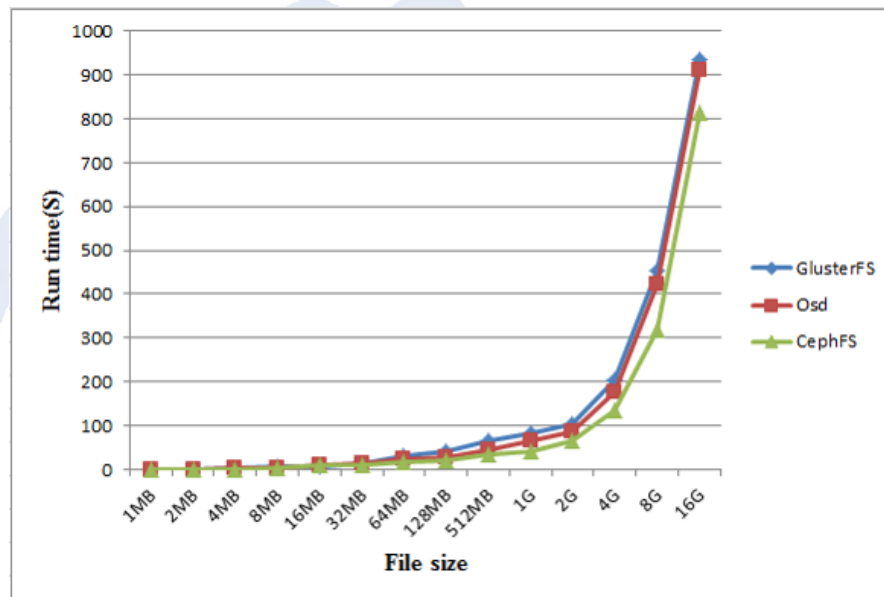


FIGURE 4.22: Performance Test with write for Rados Gateway, CephFS and GlusterFS

Chapter 5

Conclusions and Future Work

5.1 Concluding Remarks

Ceph is a high-performance, scalable storage, safe and stable management platform management mode with simple and low cost and built in addition to significantly reduce the build time and reduce system risk. Stable system architecture and flexible virtual machine settings. Let the platform can provide high reliability. High scalability and meet user demand of the user storage system. We also use Inkscope to monitor the operation of the machine and the space usage. If the machine has any problems we can clearly know through the web interface.

In the integration system often encounter version update and version does not support each other's problems, the occurrence of such problems will make us all must be new, such as Ceph version we tried three Hammer, Infernalis, Jewel. We found that the Jewel version is compatible with Hadoop and Alluxio. We also encountered insufficient hardware space, lack of integration authority and OSD tree can not start normally.

This paper use open source software components are Ceph, Alluxio and Hadoop environment. Through this experiment, we get a few results as follows:

- High speed read/write storage and flexible expansion storage space with NFS using the OSD other format.
- Ceph data is safer in Rados Gateway because of intermediate security.
- Hadoop operations can be saved directly to the OSD via the CephFS plugin. In this way, access to data is more efficient than Rados Gateway.
- CephFS is significantly better than Rados Gateway at writing.
- CephFS, Rados Gateway and GlusterFS in the case of small data comparison are no significant differences at writing.
- When the data starts from GB, CephFS is significantly better than the others.

5.2 Future Work

In our system, due to the lack of hardware resources quantity. We hope to have the opportunity to increase the number of OSD nodes. We hope to have the opportunity to use more physical machine environment. We can store the entity data on this system, such as the campus WIFI Log data. For the file storage function, we hope to use erasure code to improve security and availability.

References

- [1] P. Shu, R. Gu, Q. Dong, C. Yuan, and Y. Huang. Accelerating big data applications on tiered storage system with various eviction policies. pages 1350–1357, 2016.
- [2] H. Wu, L. Zhu, K. Lu, G. Li, and D. Wu. Stagefs: A parallel file system optimizing metadata performance for ssd based clusters. pages 2147–2152, 2016.
- [3] K. Zhan and A.H. Piao. Optimization of ceph reads/writes based on multi-threaded algorithms. pages 719–725, 2016.
- [4] D. Manini, M. Gribaudo, and M. Iacono. Modeling replication and erasure coding in large scale distributed storage systems based on ceph. volume 18, pages 273–284, 2016.
- [5] S. Meyer and J.P. Morrison. Supporting heterogeneous pools in a single ceph storage cluster. pages 352–359, 2015.
- [6] M.A. Sevilla, N. Watkins, C. Maltzahn, I. Nassi, S.A. Brandt, S.A. Weil, G. Farnum, and S. Fineberg. Mantle: A programmable metadata load balancer for the ceph file system. volume 15-20-November-2015, 2015.
- [7] Z. Farkas, P. Kacsuk, and A. Hajnal. Connecting workflow-oriented science gateways to multi-cloud systems. pages 40–46, 2015.
- [8] Y. Huang, Y. Yesha, M. Halem, Y. Yesha, and S. Zhou. Yinmem: A distributed parallel indexed in-memory computation system for large scale data analytics. pages 214–222, 2016.

- [9] I. Mavridis and H. Karatza. Performance evaluation of cloud-based log file analysis with apache hadoop and apache spark. *Journal of Systems and Software*, 125:133–151, 2017.
- [10] J. Pinto, P. Jain, and T. Kumar. Hadoop distributed computing clusters for fault prediction. 2016.
- [11] W. Kong and Y. Luo. Multi-level image software assembly technology based on openstack and ceph. pages 307–310, 2016.
- [12] M.D. Poat and J. Lauret. Performance and advanced data placement techniques with ceph’s distributed storage system. *Journal of Physics: Conference Series*, 762(1), 2016.
- [13] L. Wang and Y. Wen. Design and implementation of ceph block device in userspace for container scenarios. pages 383–386, 2016.
- [14] P. Li and W. Xu. Optimizing hash-based distributed storage using client choices. 2016.
- [15] A. Ahmad, A. Paul, S. Din, M.M. Rathore, G.S. Choi, and G. Jeon. Multilevel data processing using parallel algorithms for analyzing big data in high-performance computing. *International Journal of Parallel Programming*, pages 1–20, 2017.
- [16] D. Shankar, X. Lu, and D.K.D.K. Panda. Boldio: A hybrid and resilient burst-buffer over lustre for accelerating big data i/o. pages 404–409, 2016.
- [17] E.N.C. Wai, P.-W. Tsai, and J.-S. Pan. Hierarchical pso clustering on mapreduce for scalable privacy preservation in big data. *Advances in Intelligent Systems and Computing*, 536:36–44, 2017.
- [18] R.J. Commons, K. Thriemer, G. Humphreys, I. Suay, C.H. Sibley, P.J. Guerin, and R.N. Price. The vivax surveyor: Online mapping database for plasmodium vivax clinical trials. *International Journal for Parasitology: Drugs and Drug Resistance*, 7(2):181–190, 2017.

- [19] K. Matsuzaki. Functional models of hadoop mapreduce with application to scan. *International Journal of Parallel Programming*, 45(2):362–381, 2017.
- [20] B. Jena, M.K. Gourisaria, S.S. Rautaray, and M. Pandey. A survey work on optimization techniques utilizing map reduce framework in hadoop cluster. *International Journal of Intelligent Systems and Applications*, 9(4):61–68, 2017.
- [21] X. Zhang, S. Gaddam, and A.T. Chronopoulos. Ceph distributed file system benchmarks on an openstack cloud. pages 113–120, 2015.
- [22] P.S. Andrei, M. Florica, M. Alin, A. Victor, and C.M. Claudiu. Hardware acceleration in ceph distributed file system. pages 209–215, 2013.
- [23] C.-F. Tsai, W.-C. Lin, and S.-W. Ke. Big data mining with parallel computing: A comparison of distributed and mapreduce methodologies. *Journal of Systems and Software*, 122:83–92, 2016.

Appendix A

Ceph Installation

I. Modify hosts

```
# sudo vim /etc/hosts
```

II. Modify hostname

```
# sudo vim /etc/hostname  
# sudo service hostname start
```

III. Update and upgrade

```
# sudo apt-get update && sudo apt-get upgrade
```

IV. Install NTP

```
# sudo apt-get install ntp openssh-server
```

V. Add userceph

```
# sudo useradd -d /home/{username} -m {username}  
# sudo passwd {username}  
# echo "{username} ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/{username}
```

```
# sudo chmod 0440 /etc/sudoers.d/{username}
```

VI. Create SSH authentication login

```
# ssh-keygen
# ssh-copy-id userceph@mon1
# ssh-copy-id userceph@osd1
# ssh-copy-id userceph@osd2
# ssh-copy-id userceph@osd3
```

VII. Modify config

```
# sudo vim ~/.ssh/config
Host mon1
  Hostname mon1
  User userceph
Host osd1
  Hostname osd1
  User userceph
Host osd2
  Hostname osd2
  User userceph
Host osd3
  Hostname osd3
  User userceph
```

VIII. Download release key

```
# wget -q -O- 'https://download.ceph.com/keys/release.asc' | sudo apt-key add -
# echo deb http://download.ceph.com/debian-jewel/ $(lsb_release -sc) main | sudo tee /etc/apt/sources.
# sudo apt-get update && sudo apt-get install ceph-deploy
```

IX. Create cluster

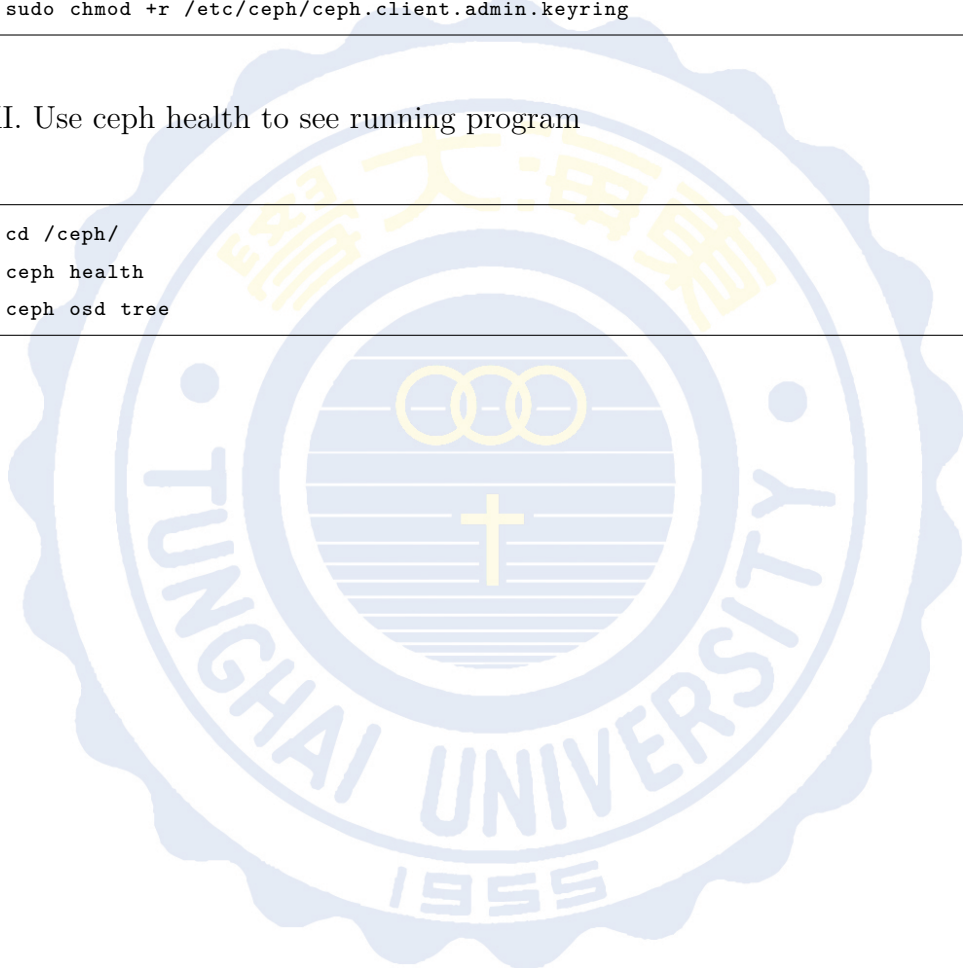
```
# mkdir ~/ceph && cd ~/ceph
# ceph-deploy new {mon-nodes}
# sed -i '$a osd pool default size = 3' ceph.conf
```

X. Install Ceph

```
# ceph-deploy install {deploy-node} {mon-nodes} {osd-nodes}
# ceph-deploy mon create-initial
# ceph-deploy osd prepare osd1:/mnt/{osd0; osd1; osd2}
# ceph-deploy osd activate osd1:/mnt/{osd0; osd1; osd2}
# ceph-deploy admin {all nodes}
# sudo chmod +r /etc/ceph/ceph.client.admin.keyring
```

XI. Use ceph health to see running program

```
# cd /ceph/
# ceph health
# ceph osd tree
```



Appendix B

Rados Gateway Installation

I. Install Ceph Object Gateway

```
# ceph-deploy install --rgw mon1
```

II. Create Rados Gateway

```
# ceph-deploy rgw create mon1
```

III. Rados Gateway website

```
# mon1:7480
```

IV. Create Gateway Rados user account for S3 access

```
# sudo radosgw-admin user create --uid="testuser" --display-name="First User"
```

V. Install python boto

```
# sudo apt-get install -y python-boto
```

V. Create and modify py

```
# sudo vim s3test.py
import boto
import boto.s3.connection
access_key = 'I5WPD6S2BX6JKE2I4TU8'
secret_key = '3WtojizD5CIGNsx161subabQAPu0fHw7BnGhlrdg'
conn = boto.connect_s3(
aws_access_key_id = access_key,
aws_secret_access_key = secret_key,
host = '{hostname}', port = {port},
is_secure=False, calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)
bucket = conn.create_bucket('my-new-bucket')
for bucket in conn.get_all_buckets():
print "{name} {created}".format(
name = bucket.name,
created = bucket.creation_date,
)
# python s3test.py
# scp -r tmp userceph@mon1:~
# sudo vim s3ListObject.py
# python s3ListObject.py
# sudo vim s3ListObject.py
# python s3ListObject.py
# sudo vim s3generate.py
# python s3generate.py
```


Appendix C

Hadoop Installation

I. Modify hosts

```
# sudo vim /etc/hosts
```

II. Install JDK

```
# sudo apt-get -y install openjdk-7-jdk  
# sudo ln -s /usr/lib/jvm/java-7-openjdk-amd64 /usr/lib/jvm/jdk
```

III. Creat SSH authentication login

```
# ssh-keygen -t rsa -f ~/.ssh/id_rsa -P ''  
# cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

IV. Install hadoop

```
# wget http://ftp.twaren.net/Unix/Web/apache/hadoop/common/hadoop-2.6.0/  
hadoop-2.6.0.tar.gz  
# tar zxf hadoop-2.6.0.tar.gz  
# mv hadoop-2.6.0 hadoop
```

V. Modify .bashrc

```
# sudo vim .bashrc
export JAVA_HOME=/usr/lib/jvm/jdk/
export HADOOP_INSTALL=/home/userceph/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
# source ~/.bashrc
```

VI. Set hadoop config

```
# cd hadoop/etc/hadoop
# vim hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/jdk/

# vim core-site.xml

<property>
  <name>fs.default.name</name>
  <value>hdfs://hadoop-master:9000</value>
</property>

# vim yarn-site.xml

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>hduser</value>
</property>

# cp mapred-site.xml.template mapred-site.xml
# vim mapred-site.xml

<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>

# mkdir -p ~/mydata/hdfs/namenode
```

```
# mkdir -p ~/mydata/hdfs/datanode
# vim hdfs-site.xml

<property>
  <name>dfs.replication</name>
  <value>2</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/home/hduser/mydata/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>/home/hduser/mydata/hdfs/datanode</value>
</property>

# vim slaves
hadoop
```

VII. Format HDFS

```
# hdfs namenode -format
```

VIII. Start hadoop

```
# start-all.sh
```

IX. Use jps to see java running program

```
# jps
```

X. Hadoop environment set Ceph S3 API account

```
# vim core-site.xml
<property>
<name>fs.default.name</name>
<value>hdfs://namenode:9000</value>
</property>
<property>
<name>fs.s3n.awsAccessKeyId</name>
```

```
<value>BHA5B6APZUATJMMQ12UT</value>
</property>
<property>
<name>fs.s3n.awsSecretAccessKey</name>
<value>pXiRSF9fm3SSJxSGoYzNcHpC2Gm5N5G6UdeqdWhQ</value>
</property>
<property>
<name>fs.alluxio.impl</name>
<value>alluxio.hadoop.FileSystem</value>
<description>The Alluxio FileSystem (Hadoop 1.x and 2.x)</description>
</property>
<property>
<name>fs.alluxio-ft.impl</name>
<value>alluxio.hadoop.FaultTolerantFileSystem</value>
<description>The Alluxio FileSystem (Hadoop 1.x and 2.x) with fault tolerant
support</description>
</property>
<property>
<name>fs.AbstractFileSystem.alluxio.impl</name>
<value>alluxio.hadoop.AlluxioFileSystem</value>
<description>The Alluxio AbstractFileSystem (Hadoop 2.x)</description>
</property>
```

Appendix D

Alluxio Installation

I. Install Alluxio

```
# wget http://downloads.alluxio.org/downloads/files/1.2.0/alluxio-1.2.0
-hadoop2.6-bin.tar.gz
# tar xvfz alluxio-1.2.0-hadoop2.6-bin.tar.gz
# bin/alluxio bootstrap-conf localhost local
```

II. Modify Alluxio

```
# vim alluxio-env.sh
export ALLUXIO_UNDERFS_ADDRESS=s3n://my-new-bucket/test/
#ALLUXIO_UNDERFS_ADDRESS=${ALLUXIO_UNDERFS_ADDRESS:-"${ALLUXIO_HOME}/underFSStorage/"}
export ALLUXIO_WORKER_MEMORY_SIZE=${ALLUXIO_WORKER_MEMORY_SIZE:-1GB}
export ALLUXIO_JAVA_OPTS+="
-Dalluxio.user.file.writetype.default=CACHE_THROUGH
-Dlog4j.configuration=file:${CONF_DIR}/log4j.properties
-Dalluxio.worker.tieredstore.levels=1
-Dalluxio.worker.tieredstore.level0.alias=MEM
-Dalluxio.worker.tieredstore.level0.dirs.path=${ALLUXIO_RAM_FOLDER}
-Dalluxio.worker.tieredstore.level0.dirs.quota=${ALLUXIO_WORKER_MEMORY_SIZE}
-Dalluxio.underfs.address=${ALLUXIO_UNDERFS_ADDRESS}
-Dalluxio.worker.memory.size=${ALLUXIO_WORKER_MEMORY_SIZE}
-Dalluxio.master.hostname=${ALLUXIO_MASTER_ADDRESS}
-Dorg.apache.jasper.compiler.disablejsr199=true
-Djava.net.preferIPv4Stack=true
-Dfs.s3n.awsAccessKeyId=QINHSA7JD2MBEKN97F4A
-Dfs.s3n.awsSecretAccessKey=mbjd9H80AokLYFP2Yx44K64Jgy5JdXZIlrWRrcPn
```

```
-Dalluxio.underfs.s3.disable.dns.buckets=true
-Dalluxio.underfs.s3.endpoint=mon1
-Dalluxio.underfs.s3.proxy.https.only=false
-Dalluxio.underfs.s3.endpoint.http.port=7480
"
export ALLUXIO_MASTER_JAVA_OPTS="${ALLUXIO_JAVA_OPTS}"
export ALLUXIO_WORKER_JAVA_OPTS="${ALLUXIO_JAVA_OPTS}"
```

III. Format Alluxio

```
# ./bin/alluxio format
# ./bin/alluxio-start.sh local
```

IV. Use jps to see Alluxio running program

```
# jps
```


Appendix E

CephFS Installation

I. Create user

```
# sudo useradd -d /home/{username} -m {username}
# sudo passwd {username}
# echo "{username} ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/
{username}
# sudo chmod 0440 /etc/sudoers.d/{username}
```

II. Install NTP

```
# sudo apt-get install ntp openssh-server
```

III. Create SSH authentication login

```
# ssh-copy-id {username}@{nodeIP}
```

IV. Install Ceph kit

```
# ceph-deploy install client
# ceph-deploy admin client
```

V. Update kernal

```
# sudo apt-get install linux-generic-lts-xenial
# sudo reboot
```

VI. Create mount

```
# sudo mkdir -p /mnt/mycephfs
# sudo chmod 777 /mnt/mycephfs
```

VII. Check Ceph.client.admin.keyring

```
# cd /etc/ceph
# sudo cat ceph.client.admin.keyring
# sudo mount -t ceph monIP:6789:/ /mnt/mycephfs -o name=admin,secret=cat
# df -h
```

Appendix F

Inkscope Installation

I. Download InkScope-Admviz, InkScope-Common, InkScope-Cephrestapi and InkScope-Monitoring

```
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-admviz_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-common_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-monitoring_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-cephrestapi_1.4.0.2.deb
```

II. Download Common, Cephrestapi, Cephprobe and Sysprobe

```
Mon[node]
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-common_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-cephrestapi_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-cephprobe_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-sysprobe_1.4.0.2.deb
```

III. Download Common and Sysprobe

```
OSD[node]
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-common_1.4.0.2.deb/inkscope-sysprobe_1.4.0.2.deb
# wget https://raw.githubusercontent.com/inkscope/inkscope-packaging/master/
DEBS/inkscope-sysprobe_1.4.0.2.deb
```

IV. Download inkscope-admviz, inkscope-common and inkscope-monitoring

```
Inkscope
# sudo dpkg -i inkscope-admviz_1.4.0.2.deb
# sudo dpkg -i inkscope-common_1.4.0.2.deb
# sudo dpkg -i inkscope-cephrestapi_1.4.0.2.deb
# sudo dpkg -i inkscope-monitoring_1.4.0.2.deb
Mon[node]
# sudo dpkg -i inkscope-common_1.4.0.2.deb
# sudo dpkg -i inkscope-cephrestapi_1.4.0.2.deb
# sudo dpkg -i inkscope-cephprobe_1.4.0.2.deb
# sudo dpkg -i inkscope-sysprobe_1.4.0.2.deb
OSD[node]
# sudo dpkg -i inkscope-common_1.4.0.2.deb
# sudo dpkg -i inkscope-sysprobe_1.4.0.2.deb
```

V. Create ceph-rest-api user

```
MON[node]
# sudo chmod 777 /etc/ceph
# sudo ceph auth get-or-create client.restapi mds 'allow' osd 'allow *' mon
'allow *' > /etc/ceph/ceph.client.restapi.keyring
# cd /etc/ceph
# ls
```

VI. Added /etc/ceph/ceph.conf

```
# vim /etc/ceph/ceph.conf
[client.restapi]
log_file = /dev/null
keyring = /etc/ceph/ceph.client.restapi.keyring
# ceph-rest-api -i restapi
```

VII. Install kit

```
# sudo apt-get install python-pip apache2 libapache2-mod-wsgi mongodb python-ceph
# cd ceph
# radosgw-admin user create --uid=inkscope --display-name="Inkscope admin" --
access-key="inkscopeAccess" --secret="inkscopeSecret" --
caps="users=*;metadata=*;buckets=*"

```

VIII. Modify inkscope

```
# sudo vim /opt/inkscope/etc/inkscope.conf
"ceph_conf": "/etc/ceph/ceph.conf",
"ceph_rest_api": "inkscope_host:inkscope_port",
"ceph_rest_api_subfolder": "ceph_rest_api",
"mongodb_host" : "mpongo_host",
"mongodb_set" : "mongodb0:27017,mongodb1:27017,mongodb2:27017",
"user": "inkscope", "access_key": "inkscopeAccess", "secret_key": "inkscopeSecret"

```

IX. Modify ports.conf

```
# sudo vim /etc/apache2/ports.conf

```

X. Install flask-login simple-json

```
# sudo pip install flask-login simple-json

```

XI. Start inkScope

```
# sudo a2ensite inkScope\
# sudo a2enmod proxy
# sudo vim /etc/apache2/sites-enabled/inkScope.conf
ProxyPass /ceph-rest-api/ http://<inkscope_host>:<inkscope_port>/ceph_rest_api/
api/v0.1/
# sudo service apache2 restart
# \textbf{sudo vim /etc/mongodb.conf}
# sudo service mongodb restart

```

XII. Modify and Install kit OSD[node]

```
# sudo vim /opt/inkscope/etc/inkscope.conf
"ceph_conf": "/etc/ceph/ceph.conf",
"ceph_rest_api": "inkscope_host:inkscope_port",
"ceph_rest_api_subfolder": "ceph_rest_api",
"mongodb_host" : "mpongo_host",
"mongodb_set" : "mongodb0:27017,mongodb1:27017,mongodb2:27017",
"user": "inkscope", "access_key": "inkscopeAccess", "secret_key": "inkscopeSecret"
# sudo apt-get install lshw sysstat
# sudo apt-get install python-pip python-dev
# sudo pip install psutil==2.1.3
# sudo pip install pymongo
# sudo /etc/init.d/sysprobe start
# sudo /etc/init.d/sysprobe status
```

XIII. Install MON[node]

```
# sudo apt-get install python-dev python-pip
# sudo pip install psutil==2.1.3
# sudo vim /opt/inkscope/etc/inkscope.conf
"ceph_conf": "/etc/ceph/ceph.conf",
"ceph_rest_api": "inkscope_host:inkscope_port",
"ceph_rest_api_subfolder": "ceph_rest_api",
"mongodb_host" : "mpongo_host",
"mongodb_set" : "mongodb0:27017,mongodb1:27017,mongodb2:27017",
"user": "inkscope", "access_key": "inkscopeAccess", "secret_key": "inkscopeSecret"
# sudo pip install pymongo
# sudo /etc/init.d/sysprobe start
# sudo /etc/init.d/cephprobe status
```

XIV. Carried out inkscope

```
# sudo ceph-rest-api -i restapi
# sudo service apache2 restart
# cd /var/log
# sudo nohup ceph-rest-api -i admin
```