CEPH File System

Mina Rady – mina.mouawad@student.lut.fi

Computing Clusters, Computing Grids, Computing Clouds Course

Professor Andrey Shevel, ITMO University

This report is to provide a general overview of the Ceph file system (CephFS). The aim is to examine the general features of CephFS as a distributed Software Defined Storage (SDS) system. It is also interesting the limitations or strengths of CephFS with respect to deployment parameters.

Introduction: Proprietary vs Open Cluster Storage

SDS platforms, just as most categories of software, are offered in both proprietary and open source paradigms. However, for the case of SDS, it can be closely tight to underlying hardware. That is to say, an additional concern of choice between proprietary and open cluster storage, is that proprietary SDS might be offered exclusively with specific hardware configuration to "guarantee" minimal performance quality by the vendor. Meanwhile, Open Source SDS, by definition, offers the choice for deployment on any compatible hardware infrastructure (whether physical or virtual). At the same time, open source SDS can be solicited commercially with professional services to configure them or maintain them by professional experts. Figure 1 shows a basic outline of both paradigms in SDS.

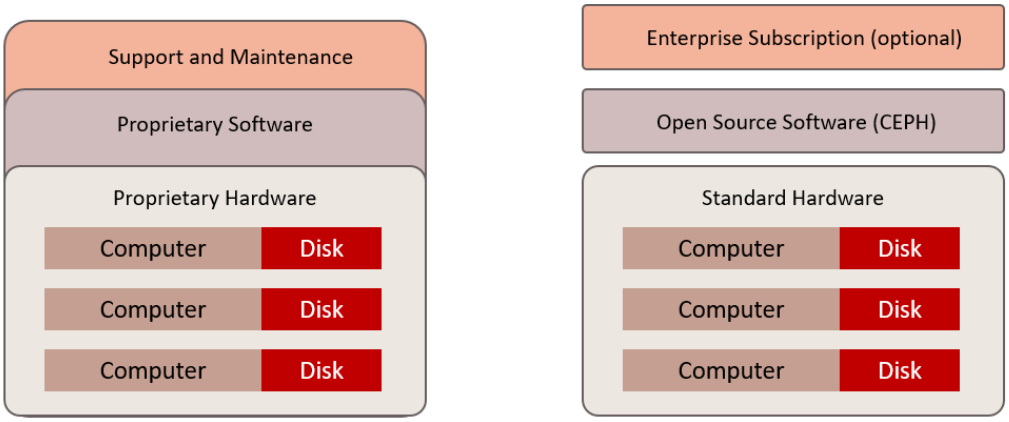


Figure 1 Proprietary (left) vs Open Source (right) paradigms in SDS

Ceph Overview

Ceph, as an SDS platform, is designed to handle distributed storage of large enterprises over distributed storage systems. It is meant to offer an abstraction of large-scale storage resources as one virtual storage file system. Meanwhile, it implements intelligent autonomous system behavior to achieve objectives such as: self-healing, self-management, and zero or minimal bottlenecks.

Essentially, Ceph offers POSIX interface which is attractive to maximize portability of applications using traditional file systems access methods to CephFS. At the same time, CephFS performs data replication operations in the backscene without any need of intervention from the original application.

Essentially, CEPH provides three access methods to the data store:

1- Data Object Access.
2- Block Device Access.
3- File System Access.

The main advantage of Ceph is that it alleviates the need for a file catalog which eliminates a possible critical performance bottleneck. For any typical distributed file system, there is a need to keep a record of existing files and their physical distribution across various storage disks/servers. To achieve this, classically, the record authority needs to be updated every time data is added or a data location is changed. Maintaing such mapping can be computationally intensive, especially for runtime lookup of data objects to respond to application requests. This can cause a bottleneck during heightened system loads. Ceph eliminates this bottleneck by relying on hashing mechanism for data indexing and lookup as illustrated in Figure 2.

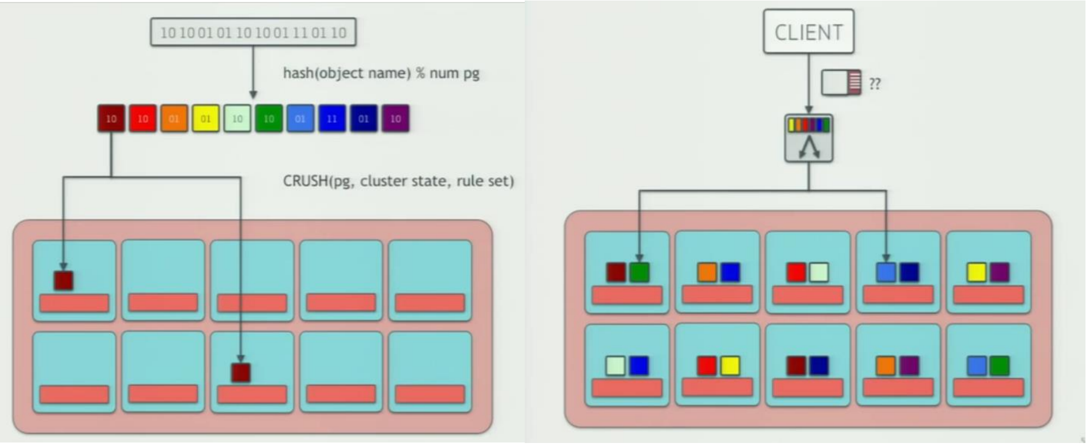

10 10 01 01 10 10 01 11 01 10
hash(object name) % num pg
10
10
01
01
10
10
01
11
01
10
CRUSH(pg, cluster state, rule set)
CLIENT
??

Figure 2 CRUSH Algorithm for hash indexing (left) and Lookup (right)

Ceph Architecture:

The core of Ceph Architecture (Figure 2), is RADOS which translates to a Reliable Autonomous Distributed Object Store. RADOS provides the core data management functionalities of Ceph while maintain POSIX compatible access.

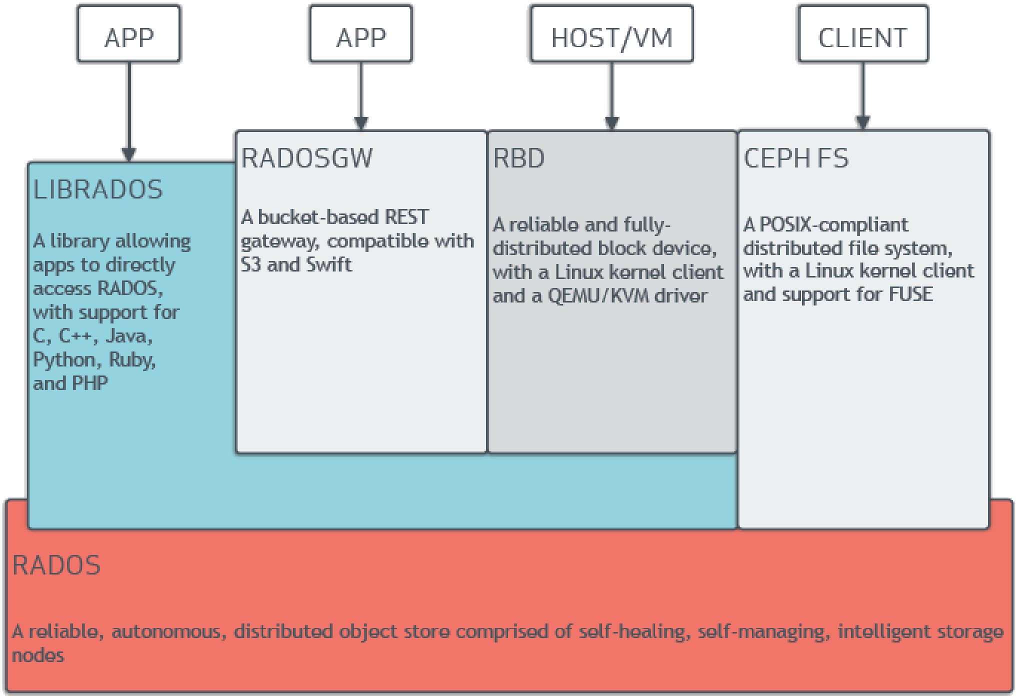

APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Figure 3 RADOS in Ceph Architecture

RADOS is accessible via 4 different venues as follows:

1- LBRADOS API: Ceph offers an API to allow applications to directly access RADOS with support for C, C++, Java, Puthong, Ruby, and PHP.
2- RADOSGW: REST gateway compatible with S3 and Swift. Compatibility with S3 access method is particularly significant with the rise of Amazon Web Services.
3- RBD: Reliable Block Device access is available for Host/VM usage in Linux Kernal client.
4- CephFS: A POSIX-Compliant fistributed file system, that is accessible from a Linux kernel client.

Main Components and Concepts:

Any Ceph deployment requires at least the following components (Figure 3):

1- OSDs.

2- Monitor Nodes.
3- Metadata Servers (only in case of shared files system).

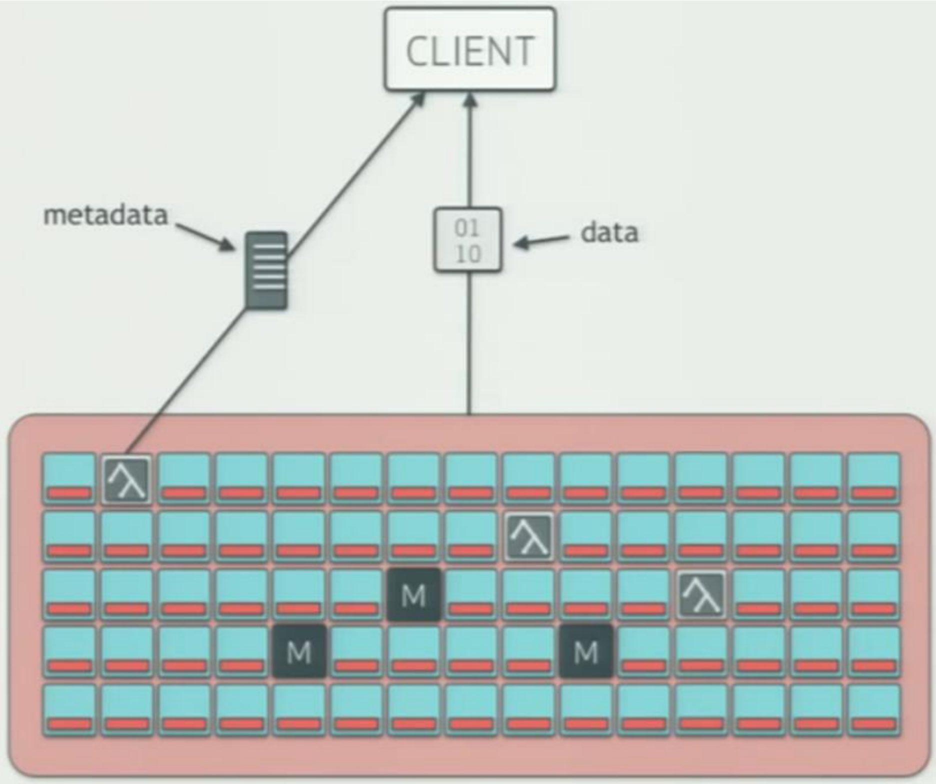


Figure 4 Typical Ceph Deployment

OSD stands for Object Storage Daemon to run on top of each storage node. The daemon points out a to a specific path in the storage disk and makes it a storage location for Ceph. OSDs are responsible for handling all data read/write operations in the storage devices they are assigned to. Theoretically, even the storage path of an OSD can point out to a RAIS infrastructure. OSDs are also responsible for peering, replication can recovery tasks.

Monitor Nods (or MONs) do not provide any data operations services, however, they are exclusively responsible for monitoring status of storage nodes such as their capacity and their status. Any Ceph deployment need to have an odd number of MONs since it is necessary for consensus making for distributed decision making.

Metadata Servers (MDs) store metadata on behalf of the Ceph Filesystem (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). They make it feasible for POSIX file system users to execute basic commands like ls, find, etc. without placing a huge burden on the Ceph Storage Cluster.

CephFS Main Concepts

The following concepts are essential to operation of CephFS:

- Pools: logical containers or partitions for storage objects.
- Placement Groups: Client does not talk to placement groups directly but it is important to be aware of how many there are as the they are the hash buckets. It is computationally intensive to map objects to OSDs, so you map them to placement groups and then you get which OSD a pool is on
- One placement group spans several OSDs, one OSD serves several placement groups
- Dynamic Subtree Partitioning is used by Metadata Servers to distribute metadata acroos deployed MDs to eliminate bottlenecks, especially during peak access to specific data resources.

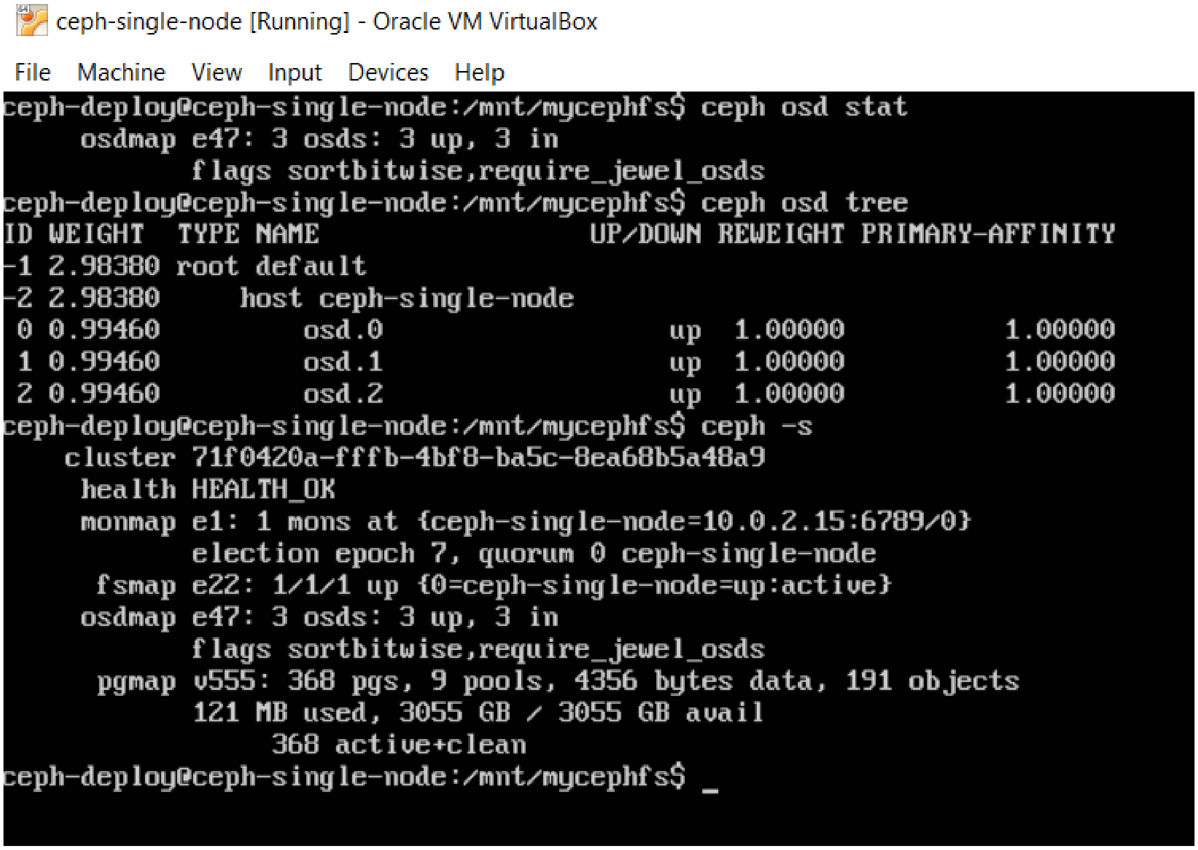
Deployment Considerations:

- Pricing: Ceph is open source, therefore source code is free of charge.
- Professional support: By Kernel Software Inc, Wisconson, USA: $3700 storage architecture. $1700 over existing Open Stack Deployment.
- Intended for large enterprise operations. Not recommended for physically small storage deployments
- It automatically balances storage spaces. Therefore note recommended if special operations are required in the storage clusters.
- Hard Disk Parameters:
  - Higher RPM, more energy cost but higher IOPS and throughput
  - Higher Density contribute positively to performance but negatively to maintainability
- Redundancy: increases read performances, but write performance does not change.
- Erasure Coding: CephFS allows customized data and parity bit distribution. This would cause higher CPU load but lower probability for failure of recovery.
- Scalability: Adding more nodes, Ceph will make sure data are evenly distributed across nodes until you hit network topology limitations.

Demonstration

- For the report, a deployment was created on a virtual machine with one main storage disk of 8 GB (dynamic) and 3 storage disks of 1 TB each (dynamic). Three OSDs were installed in addition to 1 Monitor and 1 Metadata Server.
- One networking installation challenge was faced that was not mentioned in the installation guidelines (but has several reports on online forums) was that only real internal IP addresses need to be used to refer to hosts for Ceph installation to be successful. Otherwise, it will report unresolvable addresses to the host machine.

- The following screenshot shows result of execution of general check commands:
  - Health check: ceph –s
  - Stats: ceph osd stat
  - OSD List: ceph osd tree –f json



```
ceph-deploy@ceph-single-node:/mnt/mycephfs$ ceph osd stat
     osdmap e47: 3 osds: 3 up, 3 in
            flags sortbitwise,require_jewel_osds
ceph-deploy@ceph-single-node:/mnt/mycephfs$ ceph osd tree
ID WEIGHT  TYPE NAME                 UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 2.98380 root default
-2 2.98380     host ceph-single-node
 0 0.99460         osd.0                  up  1.00000          1.00000
 1 0.99460         osd.1                  up  1.00000          1.00000
 2 0.99460         osd.2                  up  1.00000          1.00000
ceph-deploy@ceph-single-node:/mnt/mycephfs$ ceph -s
    cluster 71f0420a-fffb-4bf8-ba5c-8ea68b5a48a9
     health HEALTH_OK
     monmap e1: 1 mons at {ceph-single-node=10.0.2.15:6789/0}
            election epoch 7, quorum 0 ceph-single-node
      fsmap e22: 1/1/1 up {0=ceph-single-node=up:active}
     osdmap e47: 3 osds: 3 up, 3 in
            flags sortbitwise,require_jewel_osds
      pgmap v555: 368 pgs, 9 pools, 4356 bytes data, 191 objects
            121 MB used, 3055 GB / 3055 GB avail
                 368 active+clean
ceph-deploy@ceph-single-node:/mnt/mycephfs$ _
```