Ceph Mon: a technical overview

Paolo VIOTTI

26th March 2015

Abstract

Ceph is a free software storage platform designed to provide object, block and file storage using computer clusters running on commodity hardware. Ceph's main design goals include high scalability, fault tolerance and low maintenance requirements. This document provides an in-depth technical overview of the design of Ceph Monitor, i.e. the Ceph component in charge of maintaining a map of the cluster along with authorization information.

1 Ceph: an introduction

Ceph [1] is a free software storage platform that provides object, block and file storage using computer clusters running on commodity hardware. Since its origin as a research project around 2006, Ceph has undergone constant and substantial development, thus gaining popularity which is reflected by an ever increasing adoption as storage backend in state-of-the-art high-end computing systems [2].

As illustrated in Fig. 1, Ceph exposes to application clients multiple APIs:

- a POSIX-compatible distributed file system (**Ceph FS**), built as Linux kernel module or user-space FUSE client;
- a REST-based storage gateway (**RADOSGW**) compatible with OpenStack Swift and Amazon S3;
- a block storage device (**RDB**) suitable for virtualization platforms making use of kernel virtualization technologies such as QEMU or KVM.

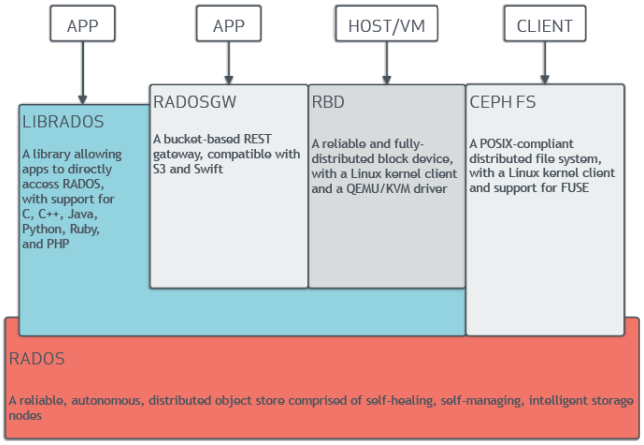

Figure 1: Ceph application stack

Beside, an application developer may even hook into the low level API exposed through **librados** and offered in several programming languages in order to directly connect with **RADOS** (*Reliable Autonomic Distributed Object Store*), i.e. the inner object storage layer that acts as foundation of the interfaces mentioned above.

The Ceph cluster consists of different components, running as distributed daemons:

- Cluster monitors (*ceph-mon*, or Mon) that keep track of active and failed cluster nodes;
- Metadata servers (*ceph-mds*) that store the metadata of inodes and directories;
- Object storage devices (*ceph-osd*) that actually store data on local filesystems;
- RESTful gateways (*ceph-rgw*) that expose the object storage layer as an interface compatible with Amazon S3 or OpenStack Swift APIs.

The rest of this document focuses on the monitor component, starting from its high-level architecture, deep down to the implementation details present, to date, in its C++ codebase [3]. To marry the need for this 1000 ft. view with the requirement of presenting the related low level features, in the following sections informal functional descriptions may be presented beside the name of their main corresponding entities in the code (e.g. `classes` or *data structures*). Most of the code required to run the monitor is contained in the `src/mon` directory of the repository [3] (∼34k LOC), although it shares with the rest of the codebase few functions and classes related to specific parts of the system (e.g. OSD or MDS daemons, respectively in `src/osd` and `src/mds`), or networking, logging and other basic facilities.

2 Ceph Mon

A Ceph Monitor maintains a set of structured information about the cluster state, including:

- the monitor map - `MonMap`
- the OSD map - `OSDMap`
- the Placement Group (PG) map - `PGMap`
- the MDS map - `MDSMap`.

Additionally, information about authorization and capabilities granted to users over the different components of the system is maintained. Besides, Ceph holds a history, as a sequence of *epochs*, of each state change in the maps. All these information are replicated across the ceph-mon instances - which are, for example, 3 or 5, deployed on different machines. Each monitor replica keeps its data strongly consistent using an implementation of the Paxos consensus algorithm (`Paxos`).

2.1 Architecture

As illustrated in Fig. 2, the Ceph monitor is composed by several sub-monitors which oversee different aspects of the cluster. All these monitors refer to a unique instance of Paxos[1] in order to establish a total order between possibly concurrent operations and achieve strong consistency. Thus, Paxos is agnostic of any semantic associated with the updates it helps ordering, as in fact it just delivers blobs (i.e. *bufferlist* as payload of `Message`).

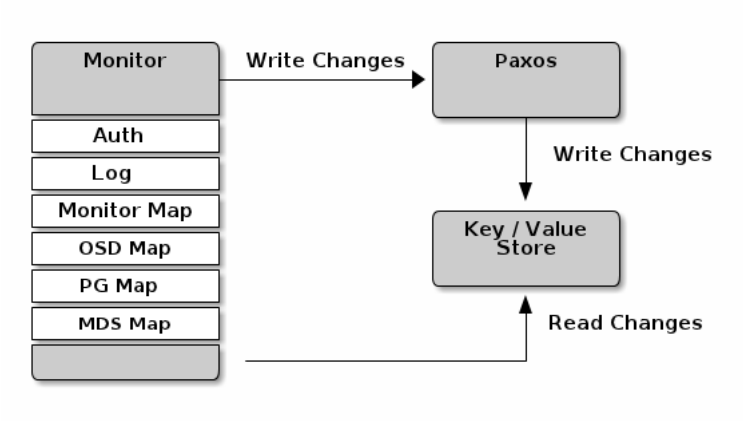

Figure 2: Ceph Monitor high-level architecture

[1] This was not true before v0.58, see: `http://ceph.com/dev-notes/cephs-new-monitor-changes/`

2.2 Code

Table 1 shows an incomplete list of classes involved in the functioning of the monitor. To each class is

Entity	Purpose	Data and attributes
`ceph_mon.cc`	main entry point for monitor daemon and batch commands	-
`Monitor`	the main Monitor class • holds and start `Paxos`, `MonitorDB`, `Messenger` and all the sub-monitors (`*Monitor`) • dispatch messages coming from `Messenger` to the sub-monitors	`MonMap`, `KeyServer`, `KeyRing`, `Paxos`, `MonitorDBStore`

Table 1: Main classes used by Ceph Monitor

monitor states

PROBING: return "probing"; SYNCHRONIZING: return "synchronizing"; ELECTING: return "electing"; LEADER: return "leader"; PEON: return "peon"; SHUTDOWN: return "shutdown";

Paxos states:

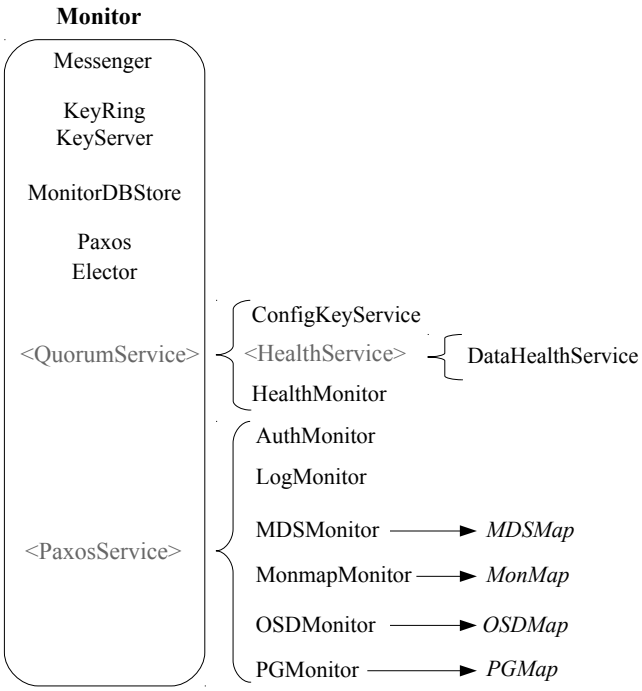

Figure 3: Ceph Monitor classes

2.3 Paxos

2.4 Additional notes

```
To ask / understand:

 - MonitorStore.{h,cc} never used
 - difference between MonCommands and DumplingMonCommands?
```

References

[1] Ceph storage platform. `http://ceph.com/`

[2] OpenStack User Survey Insights: November 2014.
`http://perma.cc/367D-G5YN`

[3] Ceph source code. `https://github.com/ceph/ceph`