Ceph Intro

All the contents in this doc are gathered from internet, just organized by me along with a few of my understandings, for the purpose of studying. Main resources include Ceph official doc, and blogs of developers.

–ddxgz

v1 - Sep, 2014

1. Brief Intro

Ceph is a unified, distributed storage system.

Unified means it provides 3 types of storage:

- object - native API librados, a REST interface
- block – like mount a network block devide
- file system - POSIX-compliant network file system

Reliability and scalability:

- 真正的无中心结构
- 没有理论上限的系统规模可扩展性

1.1. History

Ceph 是 Sage Weil 在加州大学攻读博士期间的研究课题，论文于 2006 年的 OSDI 学术会议上发表。Ceph 使用 C++语言开发，遵循 LGPL 协议。

一个存储系统会面临大致这么三种变化：

- 存储系统规模的变化：随着发展数据量越来越大，无法预料规模。

- 存储系统中设备的变化：一个由成千上万个节点构成的系统，时常会有节点的故障与替换。
- 存储系统中数据的变化：存储系统其中存储的数据的变化也很可能是高度频繁的。新的数据不断写入，已有数据被更新、移动乃至删除。

针对上述应用场景，Ceph 在设计之初的所要求几个技术特性是：

- 高可靠性。针对存储在系统中的数据而言，尽可能保证数据不会丢失。其次，也包括数据写入过程中的可靠性，也即，在用户将数据写入 Ceph 存储系统的过程中，不会因为意外情况的出现造成数据丢失。
- 高度自动化。具体包括了数据的自动 replication，自动 re-balancing，自动 failure detection 和自动 failure recovery。总体而言，这些自动化特性一方面保证了系统的高度可靠，一方面也保障了在系统规模扩大之后，其运维难度仍能保持在一个相对较低的水平。
- 高可扩展性。包括了系统规模和存储容量的可扩展，也包括了随着系统节点数增加的聚合数据访问带宽的线性扩展，还包括了基于功能丰富强大的底层 API 提供多种功能、支持多种应用的功能性可扩展。

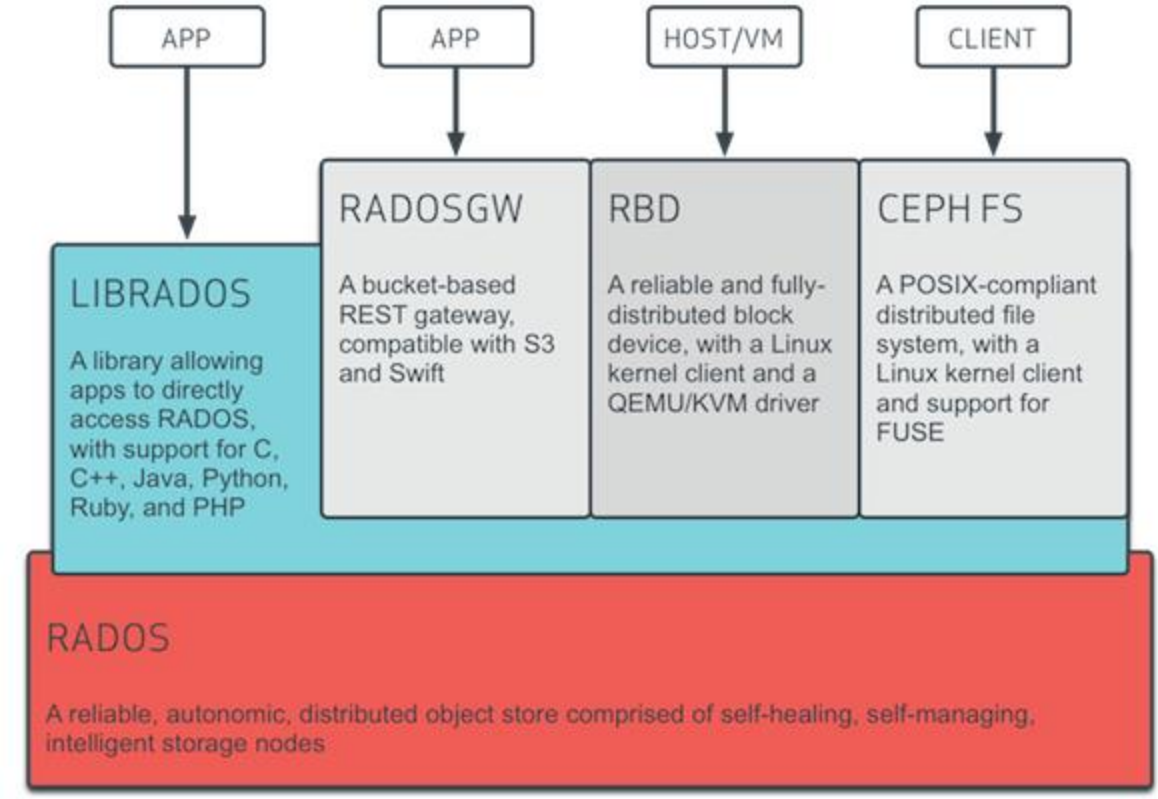
并且，也有必要充分发挥存储设备自身的计算能力。Ceph 与 Swift 类似的也是通过计算得到存储位置，其算法称为 CRUSH 。

1.2. Architecture

A Ceph Storage Cluster requires

- at least one Ceph Monitor
- at least two Ceph OSD (Object Storage Device) Daemons

- Ceph Metadata Server is essential when running Ceph Filesystem clients

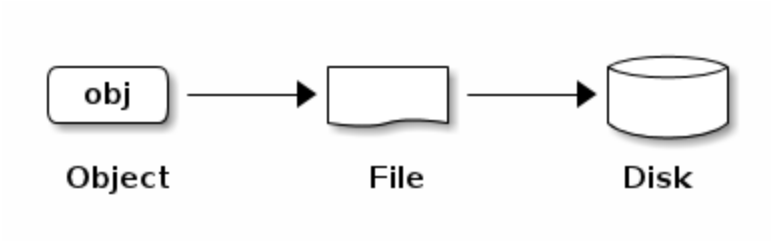


RADOS

The actual data put onto Ceph is stored on top of a cluster storage engine called RADOS(Reliable, Autonomic, Distributed Object Store), deployed on a set of storage nodes. Ceph OSD is the storage daemon that runs on every storage node (object server) in the Ceph cluster.

RADOS 这一层是一个完整的对象存储系统，所有存储在 Ceph 系统中的数据最终都是由这一层来存储的。Ceph 的高可靠、高可扩展、高性能、高自动化等等特性本质上也是由这一层所提供的。

Each object corresponds to a file in a filesystem, which is stored on an OSD. Ceph OSD Daemons handle the read/write operations on the storage disks.



Ceph OSD Daemons store all data as objects in a flat namespace (e.g., no hierarchy of directories).

An object has

- an identifier,
- binary data,
- metadata consisting of a set of name/value pairs. The semantics are completely up to Ceph Clients. For example, CephFS uses metadata to store file attributes such as the file owner, created date, last modified date, and so forth.

ID	Binary Data	Metadata
1234	01010101010101001101010010 01011000010101001101010010 01011000010101001101010010	name1 value1 name2 value2 nameN valueN

Note An object ID is unique across the entire cluster, not just the local filesystem.

librados

The Ceph Storage Cluster has a messaging layer protocol that enables clients to interact with a Ceph Monitor and a Ceph OSD Daemon. librados provides this functionality to Ceph Clients in the form of a library.

All Ceph Clients either use librados or the same functionality encapsulated in librados to interact with the object store. For example, librbd and libcephfs leverage this functionality. You may use librados to interact with Ceph directly.

You must install librados and any required packages to write applications that use librados. The librados API is written in C++, with additional bindings for C, Python and Java.

Rados Gateway (RADOSGW)

radosgw is a FastCGI service that provides a RESTful HTTP API to store objects and metadata on the Ceph cluster. The RADOS Gateway uses a unified namespace, which means you can use either the OpenStack Swift-compatible API or the Amazon S3-compatible API.

RBD

In virtual machine scenarios, people typically deploy a Ceph Block Device with the rbd network storage driver in Qemu/KVM, where the host machine uses librbd to provide a block device service to the guest. Many cloud computing stacks use libvirt to integrate with hypervisors. You can use thin-provisioned Ceph Block Devices with Qemu and libvirt to support OpenStack and CloudStack among other solutions.

Ceph Filesystem (Ceph FS)

The Ceph FS provides a POSIX-compliant filesystem as a service that is layered on top of the object-based Ceph Storage Cluster. Ceph FS files get mapped to objects that Ceph stores in the Ceph Storage Cluster. Ceph Clients mount a CephFS filesystem as a kernel object or as a Filesystem in User Space (FUSE).

The Ceph Filesystem service includes the Ceph Metadata Server (MDS) deployed with the Ceph Storage cluster. The purpose of the MDS is to store all the filesystem metadata (directories, file ownership, access modes, etc) in high-availability Ceph Metadata Servers where the metadata resides in memory. Ceph FS separates the metadata from the data, storing the metadata in the MDS, and storing the file data in one or more objects in the Ceph Storage Cluster.

1.3. CRUSH

Similar to Swift, Ceph Clients and Ceph OSD Daemons both use the CRUSH algorithm to efficiently compute information about object location, instead of having to depend on a central lookup table.

CRUSH is an algorithm that can calculate the physical location of data in Ceph, given the object name, cluster map and CRUSH rules as input. CRUSH describes the storage cluster in a hierarchy that reflects its physical organization, and thus can also ensure

proper data replication on top of physical hardware. Also CRUSH allows data placement to be controlled by policy, which allows CRUSH to adapt to changes in the cluster membership.

2. Compare with Swift for storing video

Swift 最早起源于 2008 年，Rackspace 开发 Swift 以作为当时已经颇受欢迎的 Amazon S3 对应业务作为回应。因此，Swift 的设计目标十分纯粹，就是一个优秀的、可以和 S3 相媲美的对象存储系统。其他要求纯属多余，因此完全不在 Swift 开发者的考虑之列。由此可见，Swift 正是一个典型的起源于公司内部的、作为正式产品开发的开源项目。从这一点而言，Swift 和“学院范儿”的 Ceph 可谓截然不同。也正是因为这个原因，Swift 获得了一个得天独厚的优势：不缺启动用户，一开始就有生产环境下的大规模部署应用案例。事实上，相对成熟、Web 场景下应用案例多，是 Swift 社区目前依然反复强调的一个优势。

从技术上讲，Swift 的特点主要体现在设计目标明确，就是要做一个纯粹的对象存储系统，因此不会考虑 Ceph 所强调的统一存储特性。同时，为了便于和其他项目、应用集成，Swift 选择了 Python 语言进行开发。由此可见，Ceph 和 Swift 的区别，本质上是由其产生背景和应用目标所导致的。对这二者进行对比，并进行技术上的评判，并不非常公平。

作为开源分布式存储系统中的两个优秀代表，Ceph 和 Swift 的设计和特性之中，也有着不少的相通之处：

- 都强调良好的可扩展性，都采用了无中心点结构，通过多节点扩展的方式尽可能解决了其可靠性和性能顾虑。
- 都能提供可配置的高可靠性。在两者的集群中，数据的备份数都可以选择，在常见生产环境中，也都使用三备份的方式。
- 都强调自动化的集群管理。Swift 同样引入了自动化的集群维护能力。

2.1 Feature Comparison

	Swift	Ceph
Replication	Yes	Yes
Max. obj. size	5GB (bigger objects segmented)	Unlimited
Consistency	Eventually consistent	Strongly consistent
Multi DC installation	Yes	No (demands asynchronous evetual consistency replication, which Ceph does not yet support)
Writing algorithm	Synchronous	Synchronous
Amazon S3 compatible API	Yes	Yes
Data placement method	Ring (static mapping structure)	CRUSH (algorithm)
Extension	Middlewares plug into WSGI pipeline	Does not allow modularity, doesn't include all the middleware shipped with Swift

An advantage of Swift is that it is arguably more approachable with flexible middleware that can plug into the WSGI pipeline. It is also easy to have Swift plug in a whole

lot of different auth systems and have all sort of middleware modifying its behavior and integrating specific feature. This makes it a credible alternative to S3.

One final advantage for Swift is being proven in production at very large scale, with a lot of different public cloud running it already (like Rackspace/HP/Cloudwatt etc..) and are happy with it.

2.2 Performance

Here is a performance evaluation of Ceph and Swift for object storage. But as pointed out in the comments, there may exist issues in this evaluation:

- Used different levels of API to test each. It used a high level HTTP/REST based API to access Swift, and a low-level C++ based librados API to access Ceph.
- Swift cluster may wasn't properly configured.
- It optimized the Ceph cluster with an extra disk for journals.

Another performance benchmark of Ceph Block and Swift by intel.

2.3 Use cases

在实际使用中，还是需要根据实际需求进行方案选择：

- 如果需要一个纯粹的对象存储系统，则 Swift；
- 如果需要一个纯粹的块存储系统，则 Ceph；
- 如果是一个小规模的、希望控制系统复杂度的 OpenStack 部署方案，则 Ceph；
- 如果是一个规模较大的系统，块存储和对象存储分别有较大的业务需求，则可以考虑将二者分离，分别采用 Ceph 和 Swift。

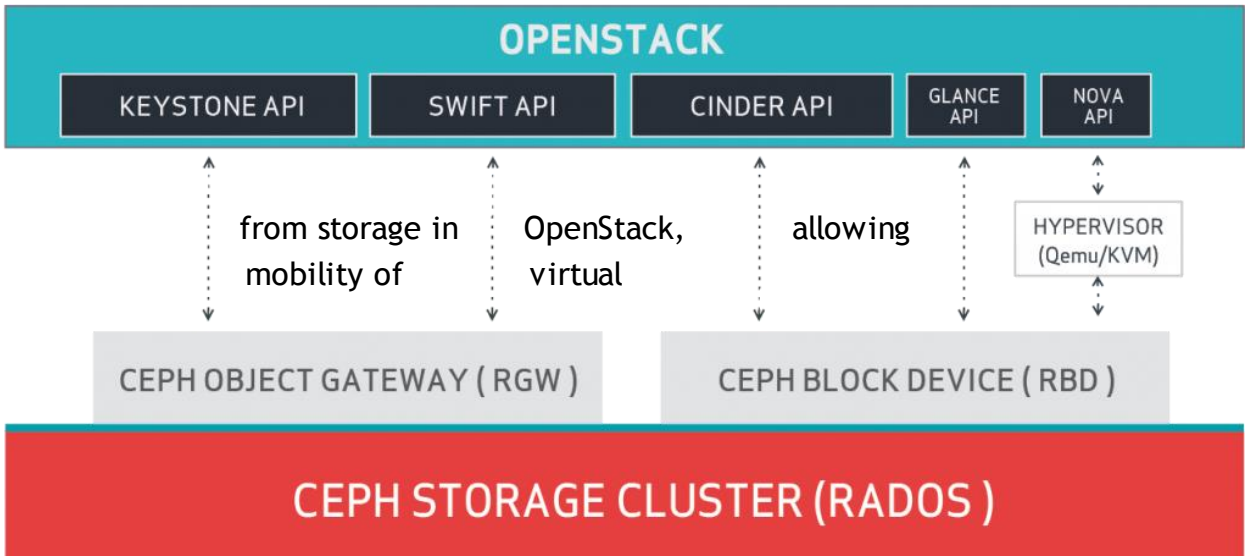
故，若单纯只为存储视频，Swift 因其相对简单的结构，是为更好的选择。

3. Integrating with OpenStack

Ceph is in the Linux kernel, and has been integrated with OpenStack since the Folsom release.

3.1 Unified Storage for OpenStack

- Ceph uniquely combines object and block into one complete storage. A total replacement for Swift, along with a fully integrated network block device for Cinder.
- Ceph's fully distributed storage cluster and block device decouple compute

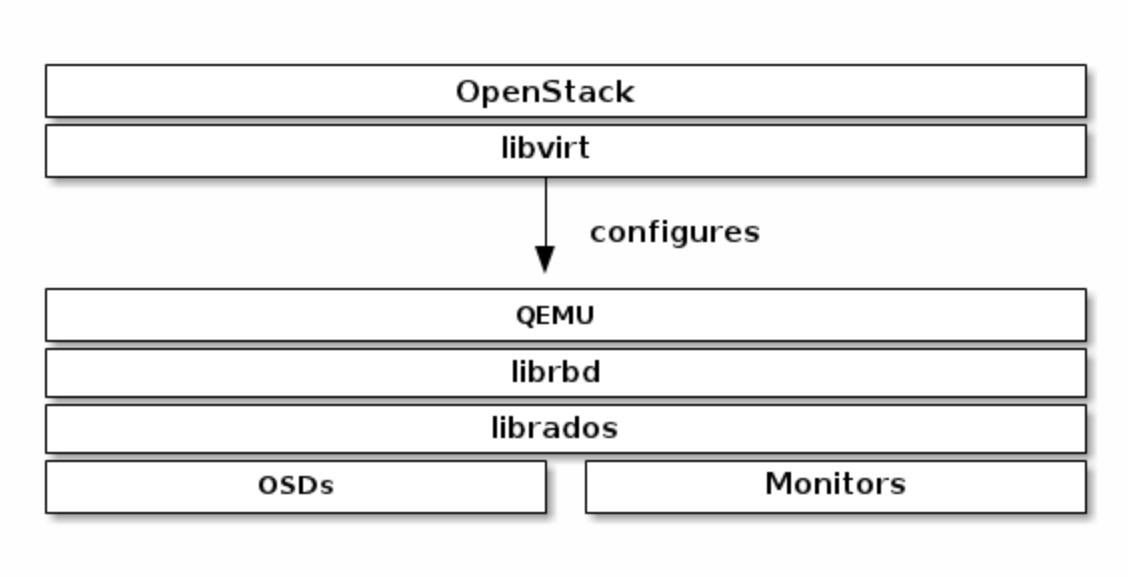


machines across your entire cluster.

- Ceph's block device also provides copy on write cloning that enables you to quickly create a thousand VMs from a single master image, requiring only enough space to store their subsequent changes.

3.2 Block Devices and OpenStack

We can use Ceph Block Device with openstack through libvirt, which configures the QEMU interface to librbd.



Ceph stripes block device images as objects across the cluster, which means that large Ceph Block Device images have better performance than a standalone server.

3.3 Three parts of OpenStack integrate with Ceph's block devices:

- Images: OpenStack Glance manages images for VMs. Images are immutable. OpenStack treats images as binary blobs and downloads them accordingly.
- Volumes: Volumes are block devices. OpenStack uses volumes to boot VMs, or to attach volumes to running VMs. OpenStack manages volumes using Cinder services.
- Guest Disks: Guest disks are guest operating system disks. By default, when you boot a virtual machine, its disk appears as a file on the filesystem of the hypervisor. Prior to OpenStack Havana, the only way to boot a VM in Ceph was to use the boot-from-volume functionality of Cinder. However, now it is possible to boot every virtual machine inside Ceph directly without using Cinder, which is advantageous because:
    - it allows you to perform maintenance operations easily with the live-migration process.
    - and if the hypervisor dies it is also convenient to trigger nova evacuate and run the virtual machine elsewhere almost seamlessly.

You can use OpenStack Glance to store images in a Ceph Block Device, and you can use Cinder to boot a VM using a copy-on-write clone of an image.

3.4 Integrating Ceph Object Gateway with OpenStack Keystone

It is possible to integrate the Ceph Object Gateway with Keystone, the OpenStack identity service. This sets up the gateway to accept Keystone as the users authority. A user that Keystone authorizes to access the gateway will also be automatically created on the Ceph Object Gateway (if didn't exist beforehand). A token that Keystone validates will be considered as valid by the gateway.

A Ceph Object Gateway user is mapped into a Keystone tenant. A Keystone user has different roles assigned to it on possibly more than a single tenant. When the Ceph Object Gateway gets the ticket, it looks at the tenant, and the user roles that are assigned to that ticket, and accepts/rejects the request according to the RGW keystone accepted roles configurable.

Keystone itself needs to be configured to point to the Ceph Object Gateway as an object-storage endpoint.

The Ceph Object Gateway will query Keystone periodically for a list of revoked tokens. Configuring the Ceph Object Gateway to work with Keystone also requires converting the OpenSSL certificates that Keystone uses for creating the requests to the nss db format.