CEPH: overview e installazione
Agenda

- CEPH Highligth
- CEPH Features
- CEPH Architecture
- CEPH Installation
Ceph was initially created by Sage Weil for his doctoral dissertation.

On March 19, 2010, Linus Torvalds merged the Ceph client into Linux kernel version 2.6.34.

In 2012, Weil created Inktank Storage for professional services and support for Ceph.

In April of 2014 Red Hat purchased Inktank bringing the majority of Ceph development in-house.
CEPH highlight

- Project started in 2007
- An object based parallel file-system
- Open source project (LGPL licensed)
- Written in C++ and C
- Kernel level
- Posix compliant
- No SPOF
- Both data and metadata could be replicated dynamically
- Configuration is config file based
- Flexible striping strategies and object sizes
  - Could be configured “per file”
In CEPH tutto è un oggetto
Non esiste il database per indicare la disposizione degli oggetti nel cluster


Esiste una “regola” per scegliere dove memorizzare i vari oggetti:

- ogni singolo nodo del cluster può calcolare la disposizione
- NOSPOF
Figure 1: A partial view of a four-level cluster map hierarchy consisting of rows, cabinets, and shelves of disks. Bold lines illustrate items selected by each select operation in the placement rule and fictitious mapping described by Table 1.
CEPH Features

- È in grado di fornire Block/Object/Posix storage
- File system supportati come back-end
  - Non-Production
    - btrfs
    - ZFS (On Linux)
  - Production
    - ext4 (small scale)
    - xfs (enterprise deployments)
CEPH Features

- Intelligent server: replicate data, migrate object, detect node failures
  - this could happen because everyone know where object belongs
- inodes are stored together with the directory object: you can load complete directory and inodes with a single I/O ("find" or "du" are greatly faster)
CEPH Features

- SAN (shared) disk is not needed to achieve HA
- Support snapshots
- Support quotas (per directory sub-tree)
- The RADOS Gateway also exposes the object store as a RESTful interface which can present as both native Amazon S3 and OpenStack Swift APIs.
- Ceph RBD interfaces with object storage system that provides the librados interface and the CephFS file system
- stores block device images as objects. Since RBD is built on top of librados, RBD inherits librados’s capabilities, including read-only snapshots and revert to snapshot
CEPH Architecture

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, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes
Ceph block devices are thin-provisioned, resizable and store data striped over multiple OSDs in a Ceph cluster.
CEPH Architecture

S3 compatible API
radosgw
librados
OSDs

Swift compatible API

Monitors
CEPH Architecture

- CephFS Kernel Object
- CephFS FUSE
- Ceph FS Library (libcephfs)
- Ceph Storage Cluster Protocol (librados)
- OSDs
- MDSs
- Monitors
CEPH Architecture
CEPH Architecture

- `fd=open("/foo/bar", O_RDONLY)`
  - Client: requests open from MDS
  - MDS: reads directory /foo from object store
  - MDS: issues capability for file content
- `read(fd, buf, 1024)`
  - Client: reads data from object store
- `close(fd)`
  - Client: relinquishes capability to MDS

- MDS out of I/O path
- Object locations are well known—calculated from object name
If OSDs use Btrfs as their local file system, data is written asynchronously using copy-on-write, so that unsuccessful write operations can be fully rolled back.
CEPH Architecture

The figure on the left shows the per-MDS throughput (ops/sec) as a function of MDS cluster size (nodes) for different operations: makedirs, makefiles, openshared, openssh=include, and openssh=lib. The graph illustrates how the throughput changes with the increase in the number of nodes in the cluster.

The right figure displays the throughput (ops/sec) over time (seconds) for different MDSs labeled mds0, mds1, mds2, and mds3. The labels mds3 fails, mds1 & 2 fail, and mds0 fails indicate key points in the dataset.

The bottom part of the image visualizes a busy directory fragmented across many MDSs, with different sections colored to represent different MDSs.
**Region:** A region represents a *logical* geographic area and contains one or more zones. A cluster with multiple regions must specify a master region.

**Zone:** A zone is a *logical* grouping of one or more Ceph Object Gateway instance(s). A region has a master zone that processes client requests.

**Important** Only write objects to the master zone in a region. You may read objects from secondary zones. Currently, the Gateway does not prevent you from writing to a secondary zone, but **DON’T DO IT.**
- https://ceph.com/docs/master/architecture/
- http://ceph.com/docs/master/start/intro/
- http://ceph.com/docs/master/release-notes/