Giacinto DONVITO
INFN-Bari

CEPH: overview e installazione

Agenda

- CEPH Highligth
- CEPH Features
- CEPH Architecture
- CEPH Installation

CEPH highlight

- 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”

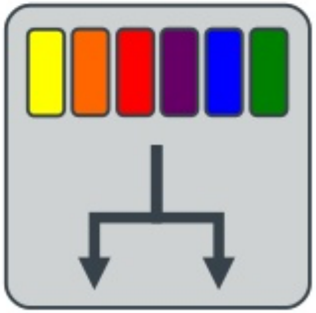
CEPH Features

- In CEPH tutto è un oggetto
- Non esiste il database per indicare la disposizione degli oggetti nel cluster
- http://ceph.com/papers/weil-crush-sco6.pdf
- Esiste una "regola" per scegliere dove memorizzare i vari oggetti:
  - ogni singolo nodo del cluster può calcolare la disposizione
  - NOSPOF

CEPH Features

- Why start with Object
  - more useful than (disk) blocks
    - names in a single flat namespace
    - variable size
    - simple API with rich semantics
  - more scalable than files
    - no hard-to-distribute hierarchy
    - update semantics do not span objects
    - workload is trivially parallel

CEPH Features



CRUSH

- Pseudo-random placement algorithm
- Fast calculation, **no lookup**
- Ensures even distribution
- Repeatable, deterministic
- Rule-based configuration
  - specifiable replication
  - infrastructure topology aware
  - allows weighting
- Stable mapping
  - Limited data migration

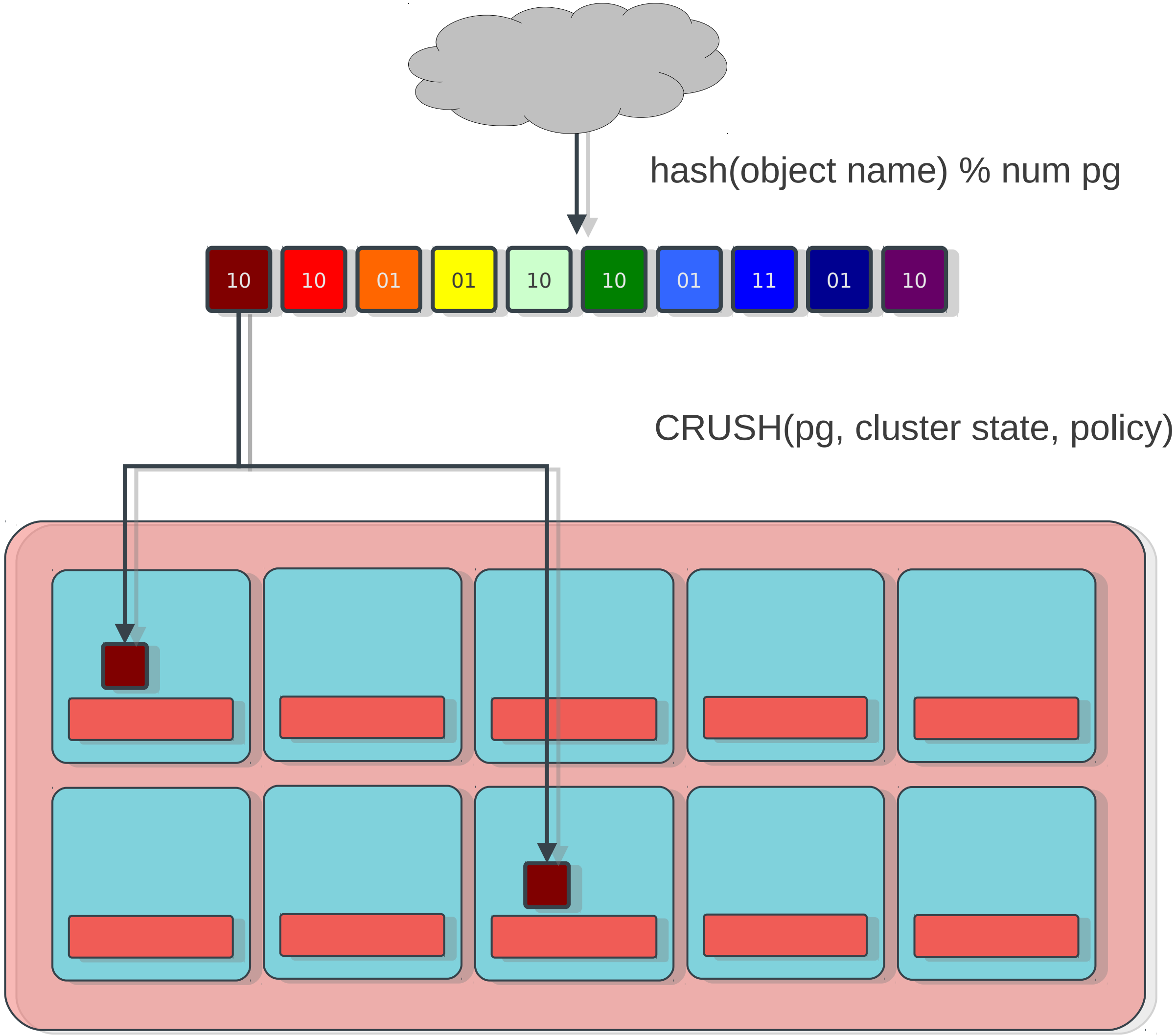
Distributed object storage

- CRUSH tells us where data should go
  - small “osd map” records cluster state at point in time
  - ceph-osd node status (up/down, weight, IP)
  - CRUSH function specifying desired data distribution
- object storage daemons (RADOS)
  - store it there
  - migrate it as the cluster changes
- decentralized, distributed approach allows
  - massive scales (10,000s of servers or more)
  - efficient data access
  - the illusion of a single copy with consistent behavior

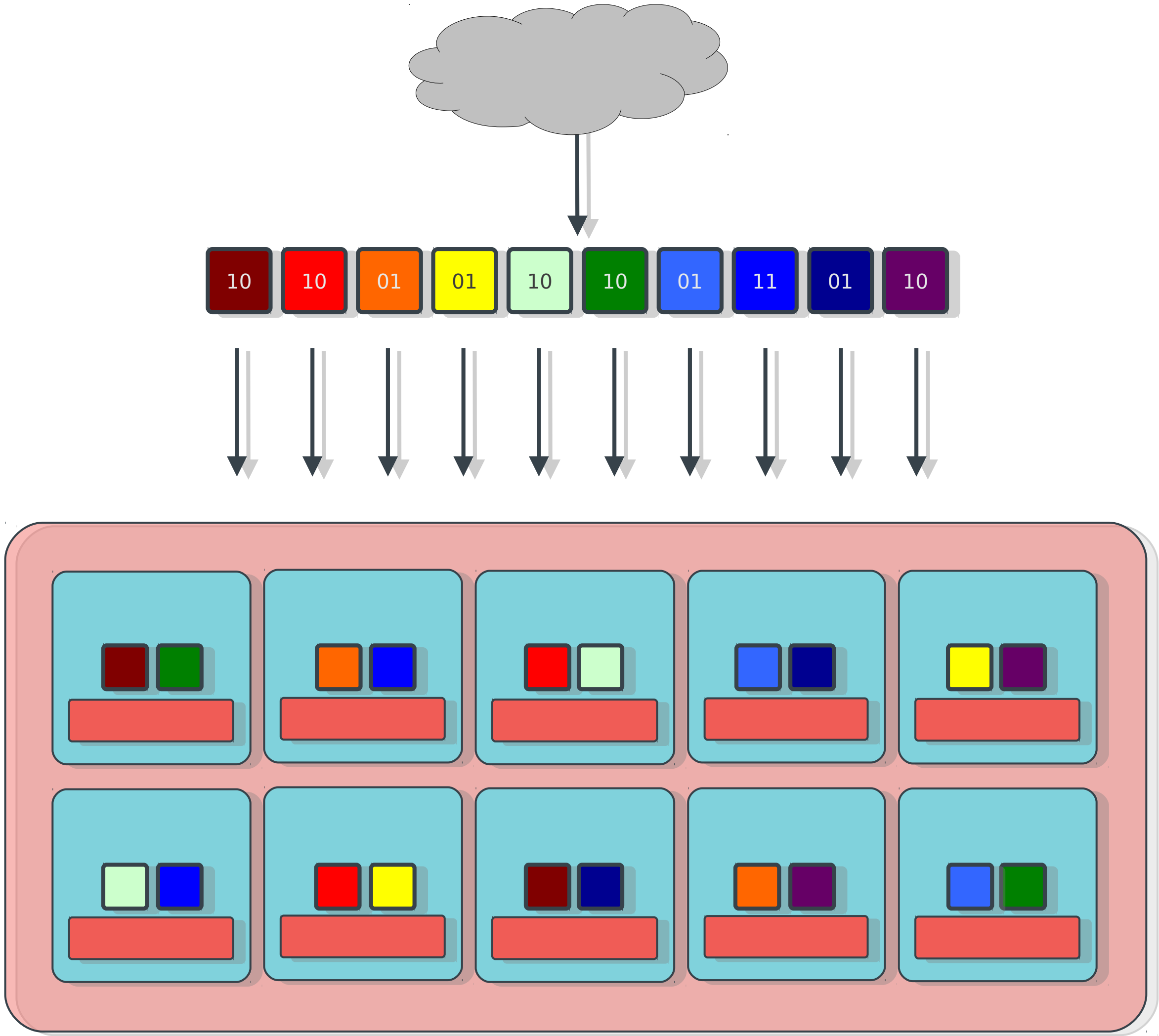
Distributed object storage

- dynamic cluster
  - nodes are added, removed; nodes reboot, fail, recover
  - recovery is the norm
- osd maps are versioned
  - shared via gossip
- any map update potentially triggers data migration
  - ceph-osds monitor peers for failure
  - new nodes register with monitor
  - administrator adjusts weights, mark out old hardware, etc.

CEPH Features



CEPH Features



CEPH Features

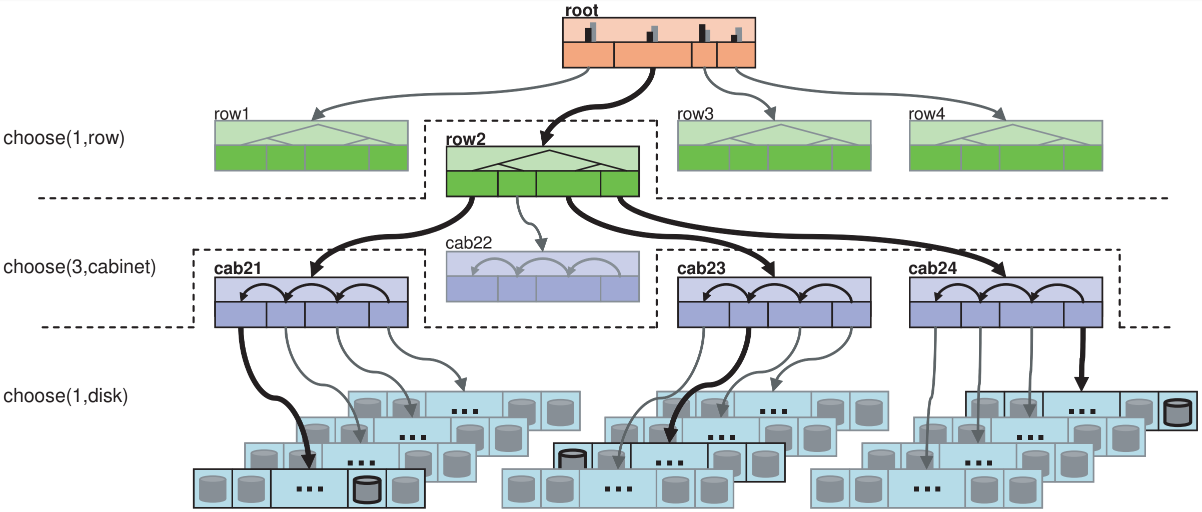
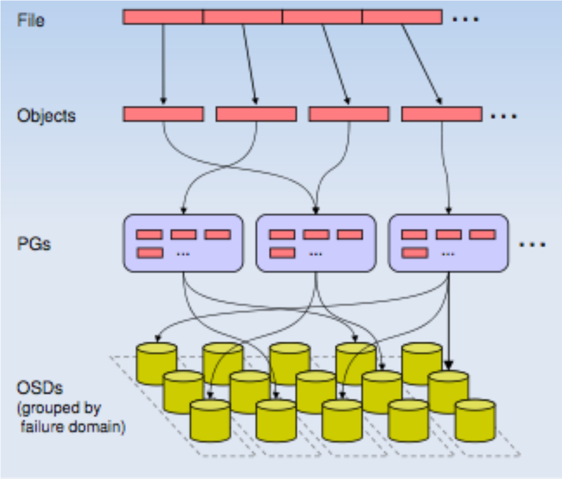


Figure 1: A partial view of a four-level cluster map hierarchy consisting of rows, cabinets, and shelves of disks. Bold lines illustrate items selected by each *select* operation in the placement rule and fictitious mapping described by Table 1.



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)

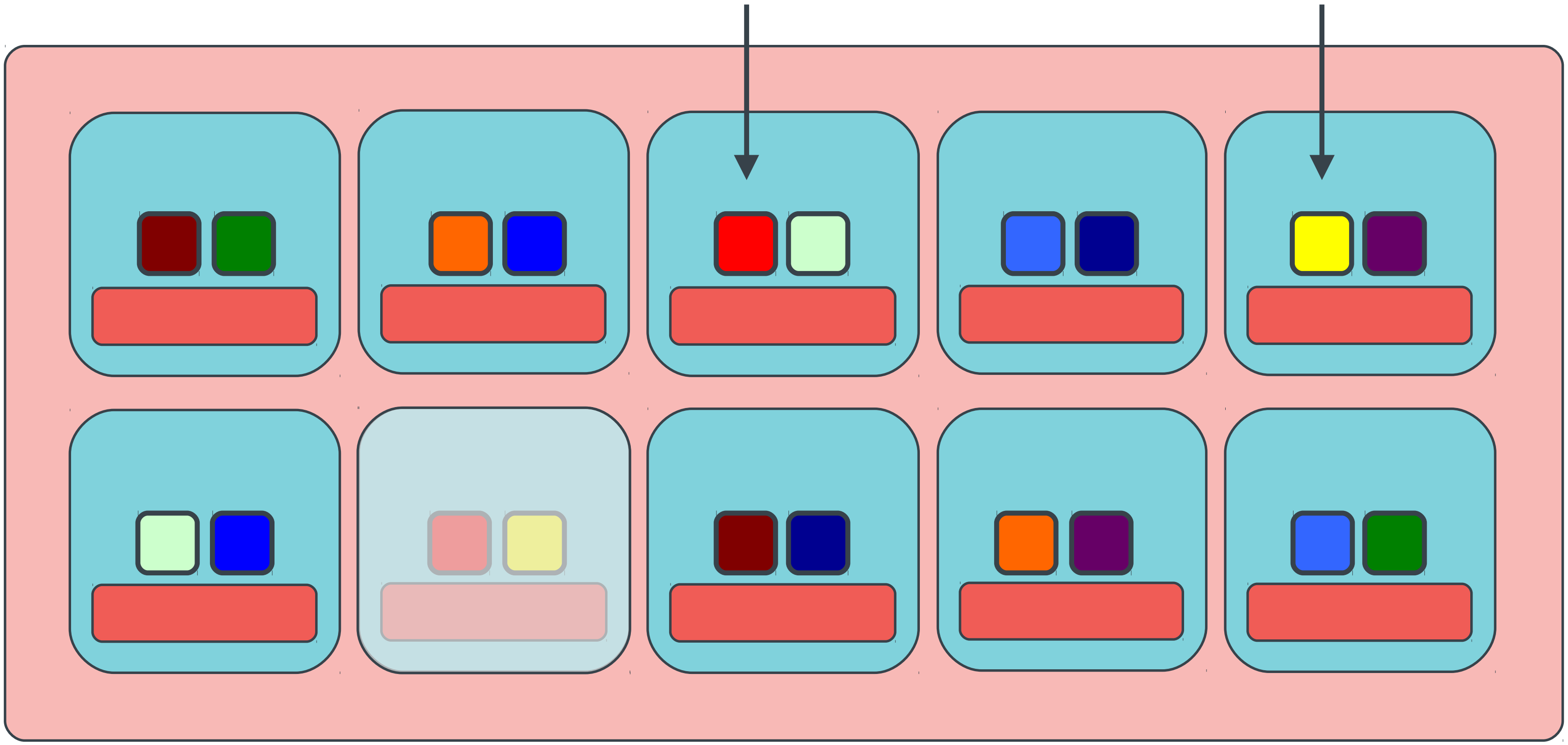
CEPH Features

recursive accounting

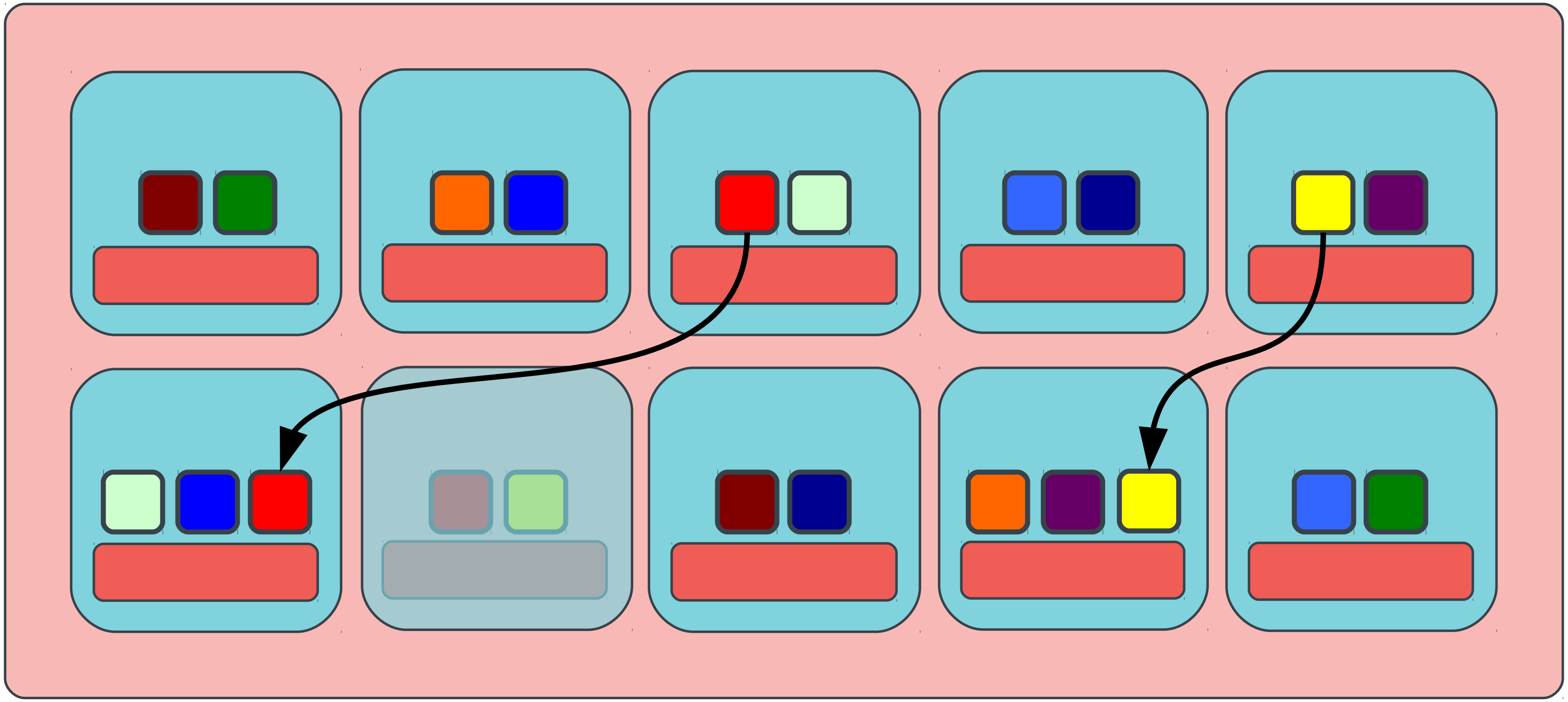
- ceph-mds tracks recursive directory stats
  - file sizes
  - file and directory counts
  - modification time
- virtual xattrs present full stats
- efficient

```
$ ls -alSh | head
total 0
drwxr-xr-x 1 root            root      9.7T 2011-02-04 15:51 .
drwxr-xr-x 1 root            root      9.7T 2010-12-16 15:06 ..
drwxr-xr-x 1 pomceph         pg4194980 9.6T 2011-02-24 08:25 pomceph
drwxr-xr-x 1 mcg_test1       pg2419992  23G 2011-02-02 08:57 mcg_test1
drwx--x--- 1 luko            adm        19G 2011-01-21 12:17 luko
drwx--x--- 1 eest            adm        14G 2011-02-04 16:29 eest
drwxr-xr-x 1 mcg_test2       pg2419992 3.0G 2011-02-02 09:34 mcg_test2
drwx--x--- 1 fuzyceph        adm       1.5G 2011-01-18 10:46 fuzyceph
drwxr-xr-x 1 dallasceph      pg275     596M 2011-01-14 10:06 dallasceph
```

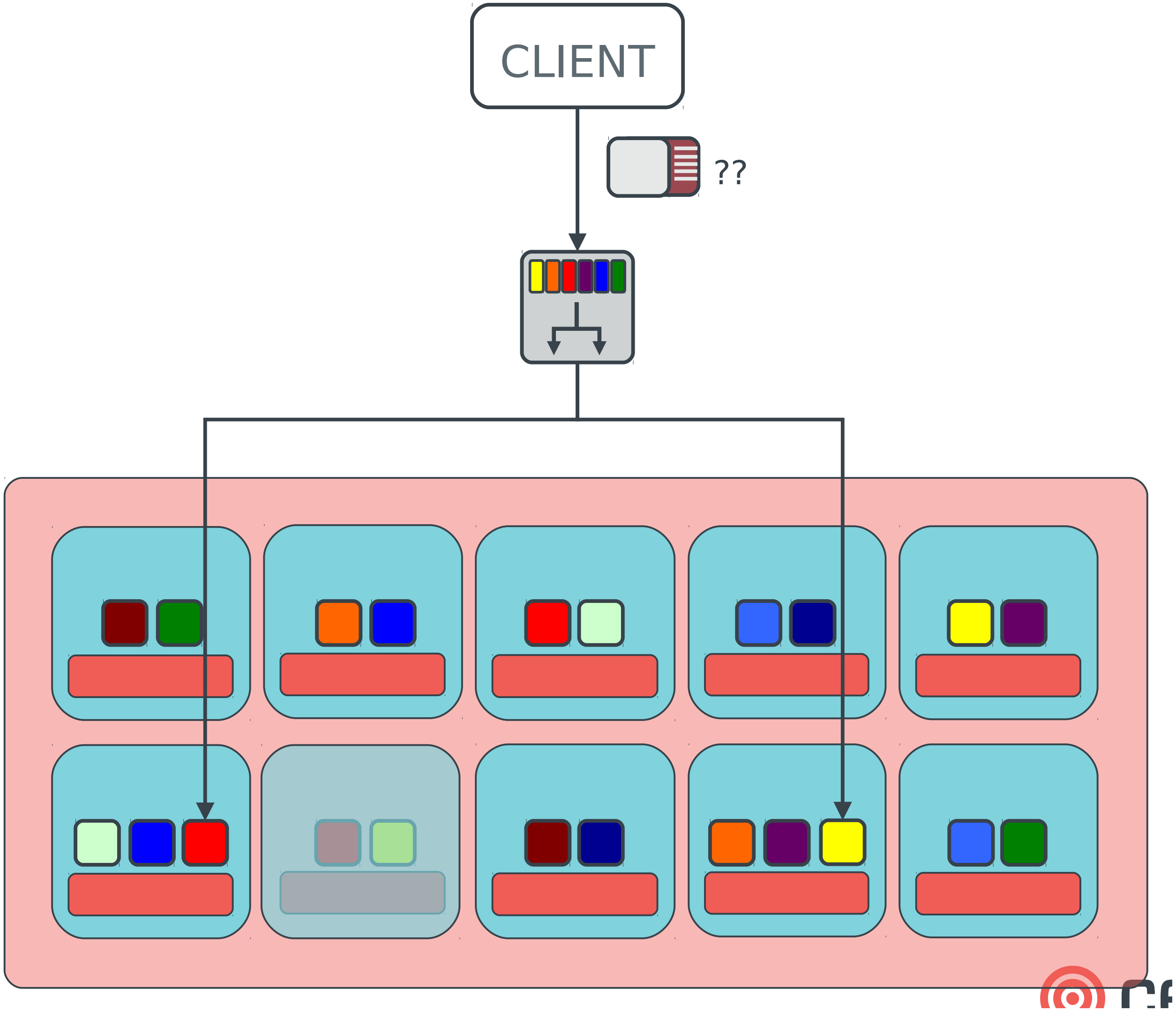
CEPH Features



CEPH Features



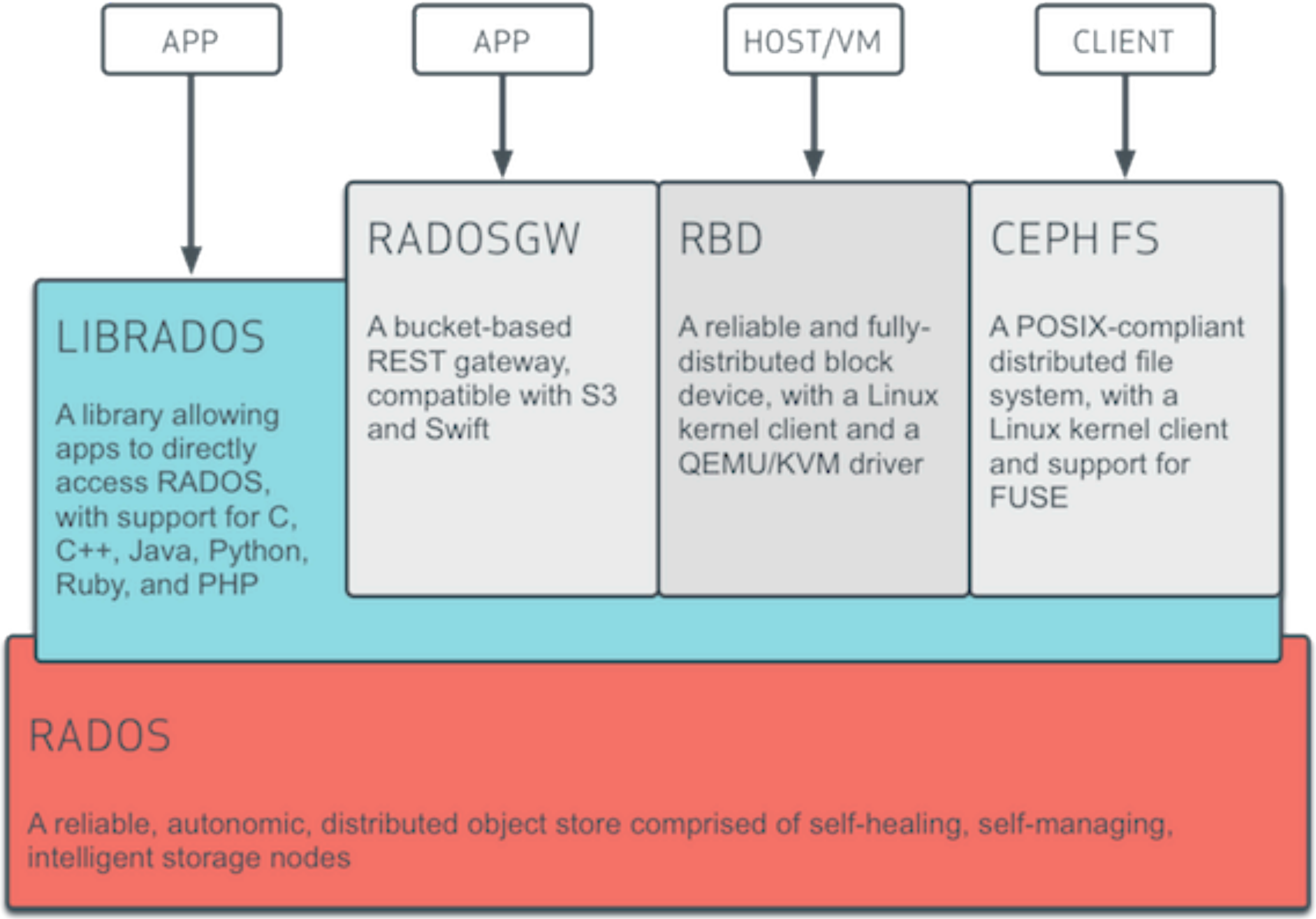
CEPH Features



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 aRESTful 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

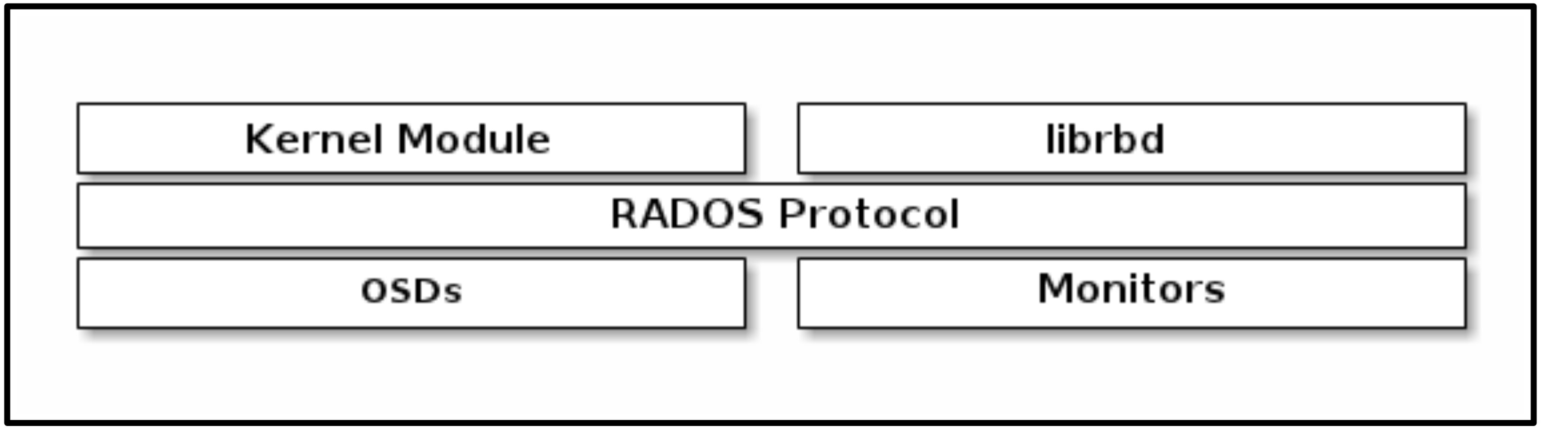
RBD: A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

CEPH FS: A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

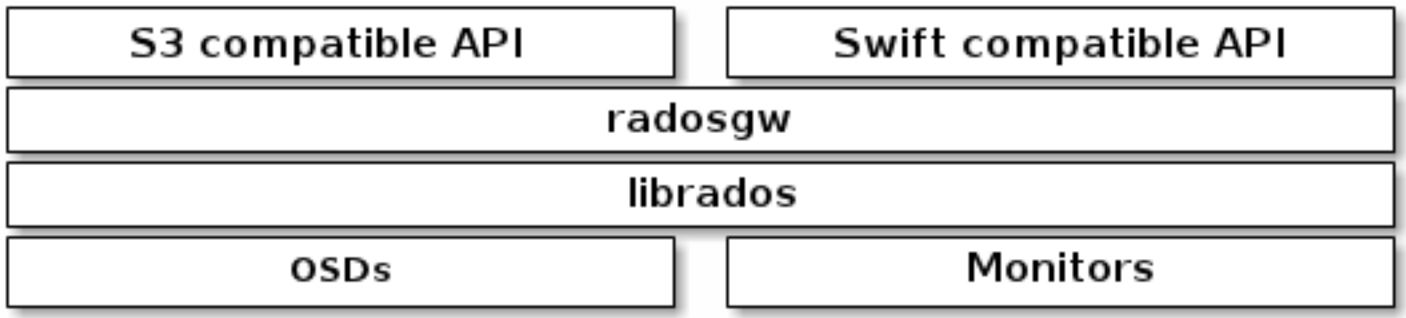
RADOS: A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

CEPH Architecture

Ceph block devices are thin-provisioned, resizable and store data striped over multiple OSDs in a Ceph cluster



CEPH Architecture



S3 compatible API

Swift compatible API

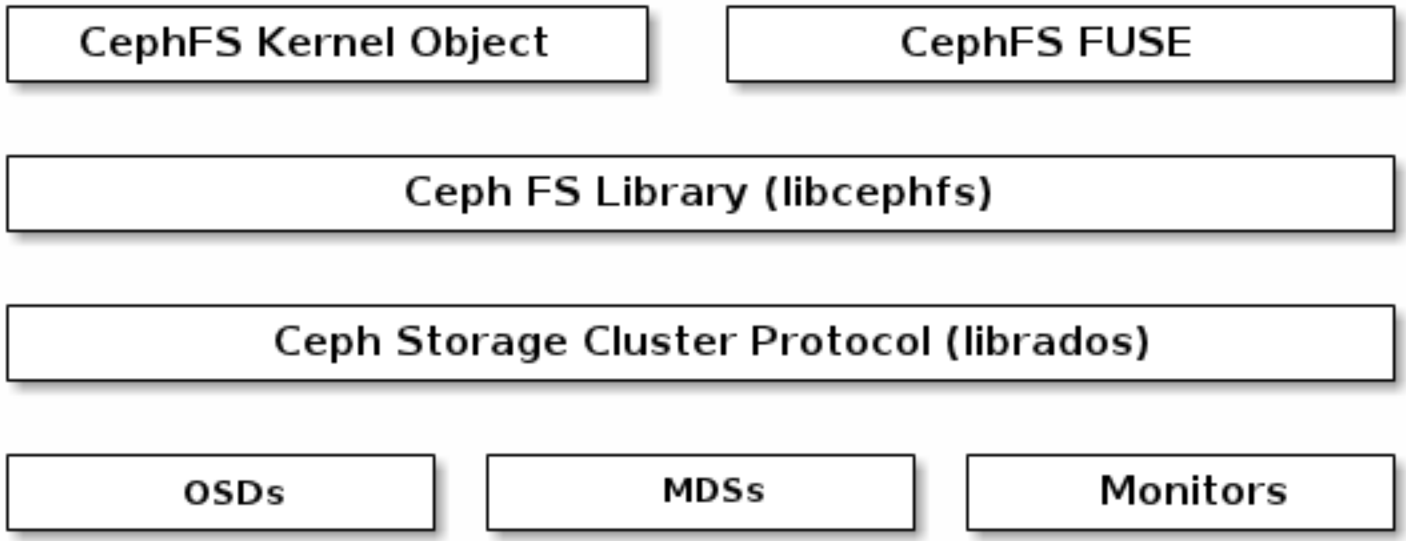
radosgw

librados

OSDs

Monitors

CEPH Architecture



CephFS Kernel Object

CephFS FUSE

Ceph FS Library (libcephfs)

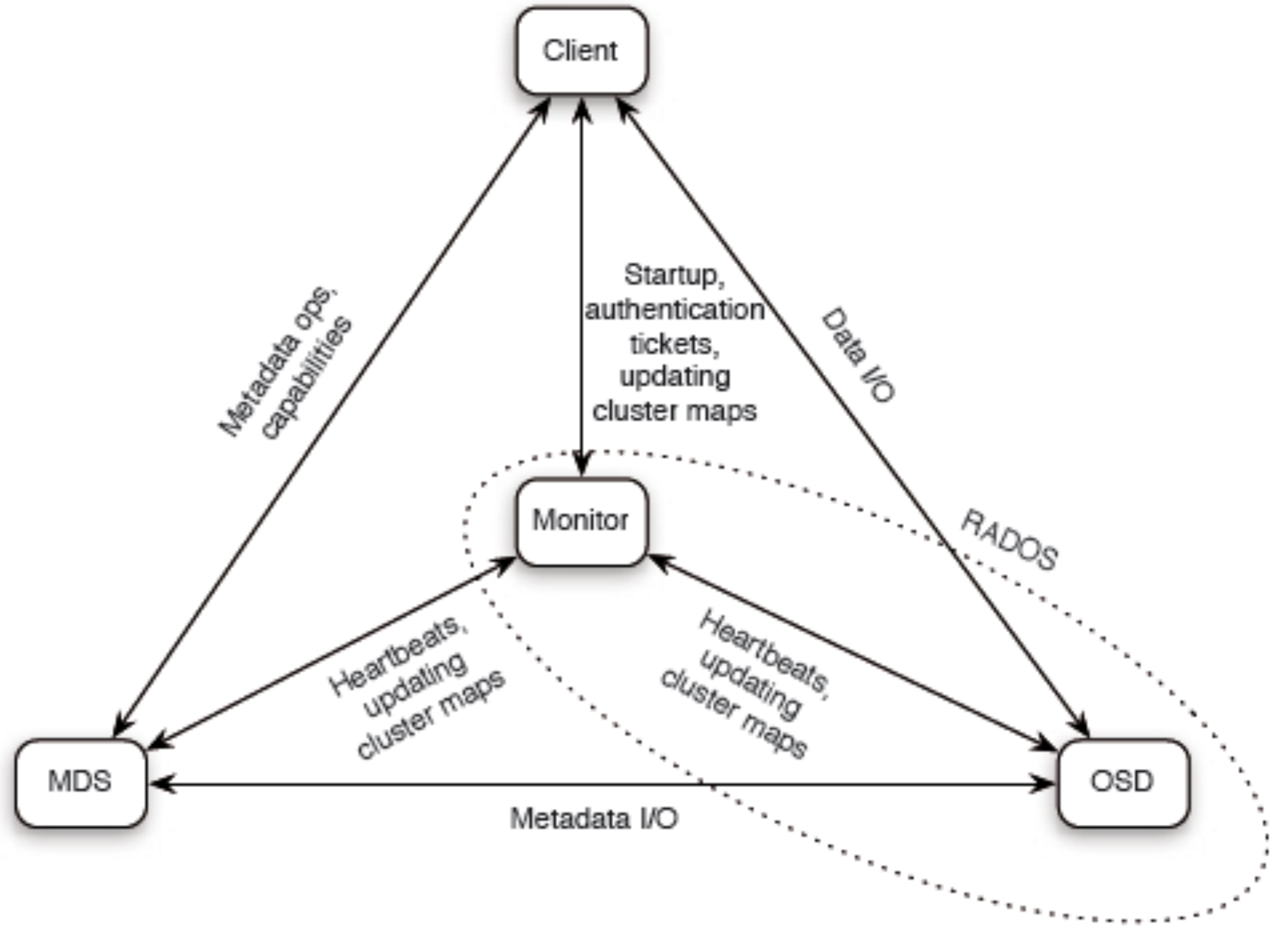
Ceph Storage Cluster Protocol (librados)

OSDs

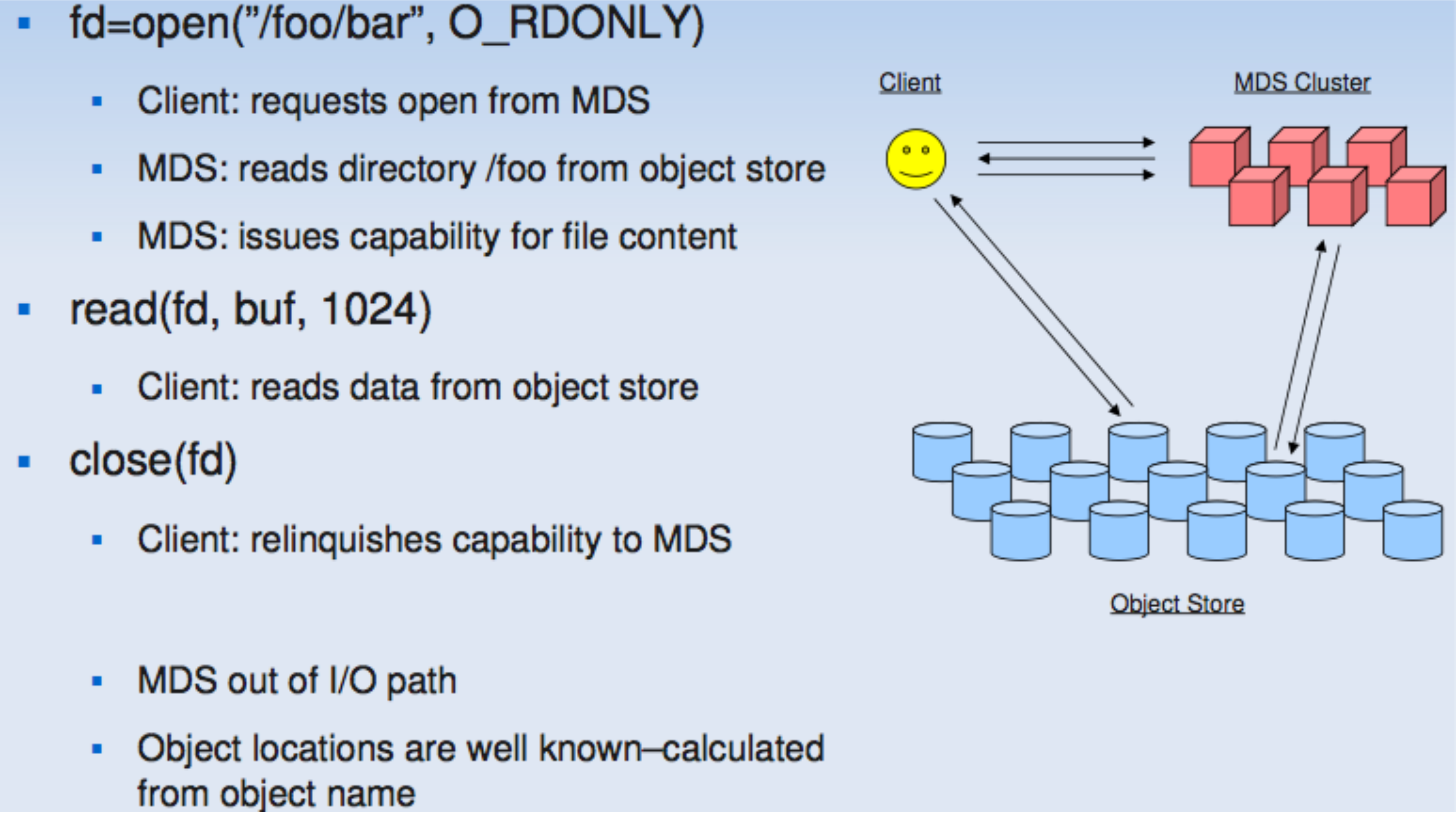
MDSs

Monitors

CEPH Architecture

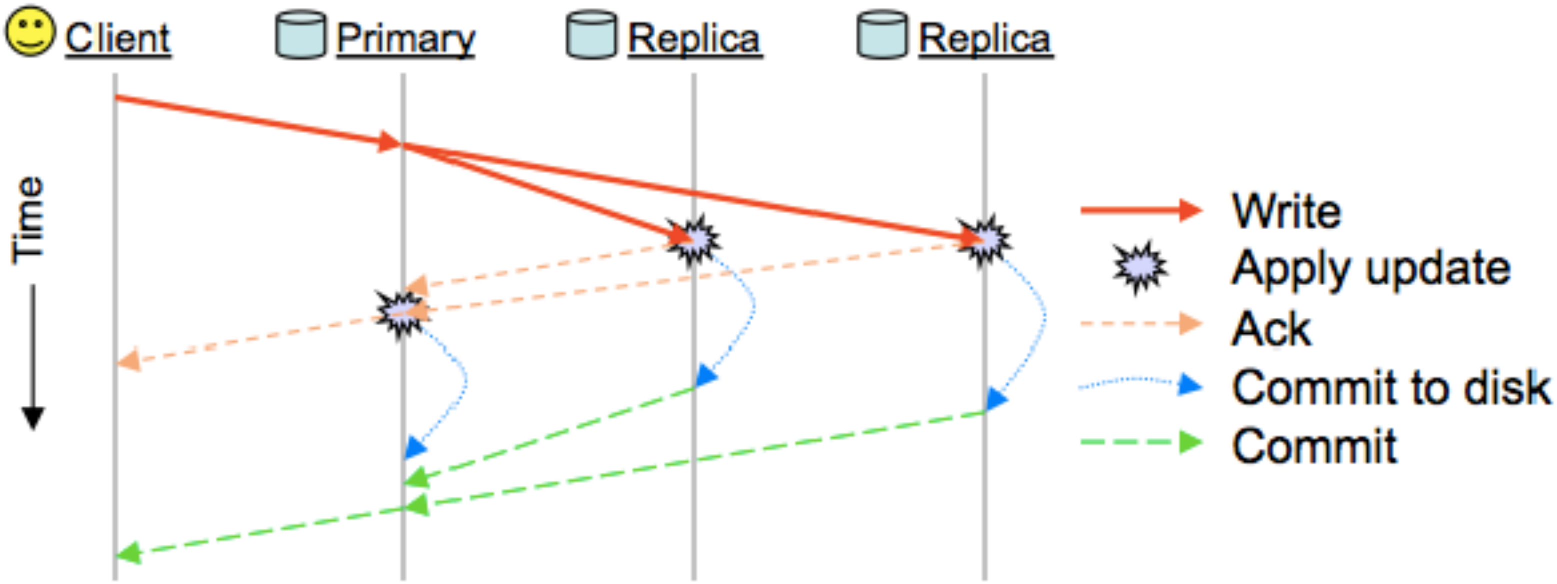

Client
Metadata ops, capabilities
Startup, authentication tickets, updating cluster maps
Data I/O
Monitor
RADOS
Heartbeats, updating cluster maps
Heartbeats, updating cluster maps
MDS
OSD
Metadata I/O

CEPH Architecture



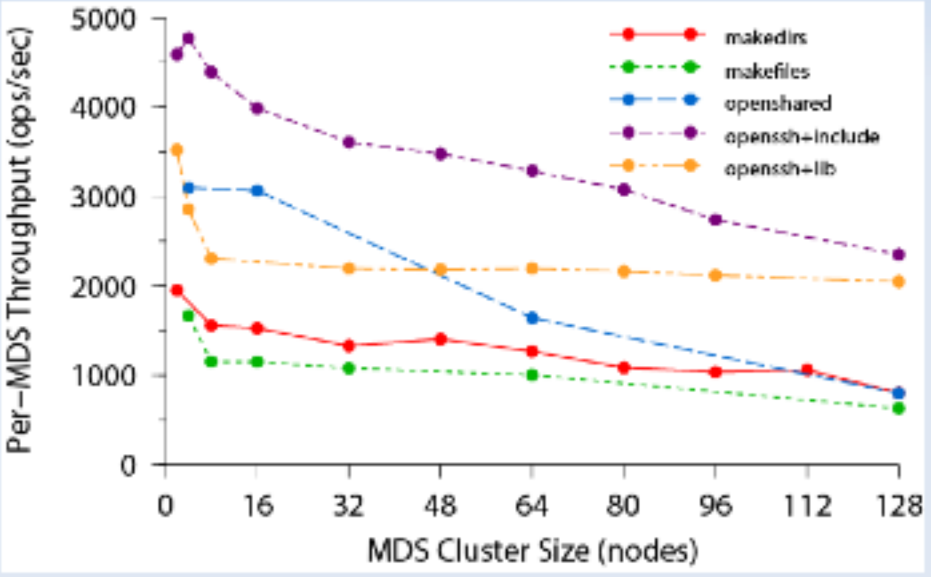
- fd=open("/foo/bar", O_RDONLY)
  - Client: requests open from MDS
  - MDS: reads directory /foo from object store
  - MDS: issues capability for file content
- read(fd, buf, 1024)
  - Client: reads data from object store
- close(fd)
  - Client: relinquishes capability to MDS
  - MDS out of I/O path
  - Object locations are well known–calculated from object name

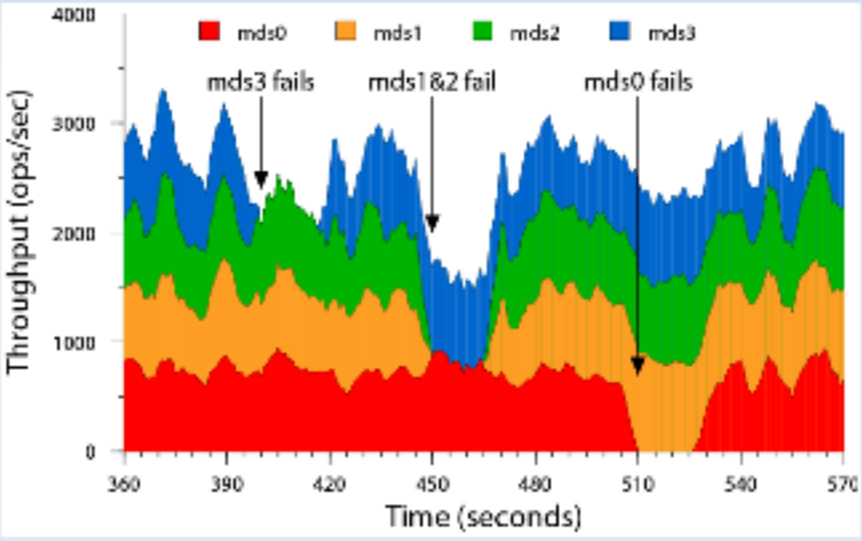
CEPH Architecture

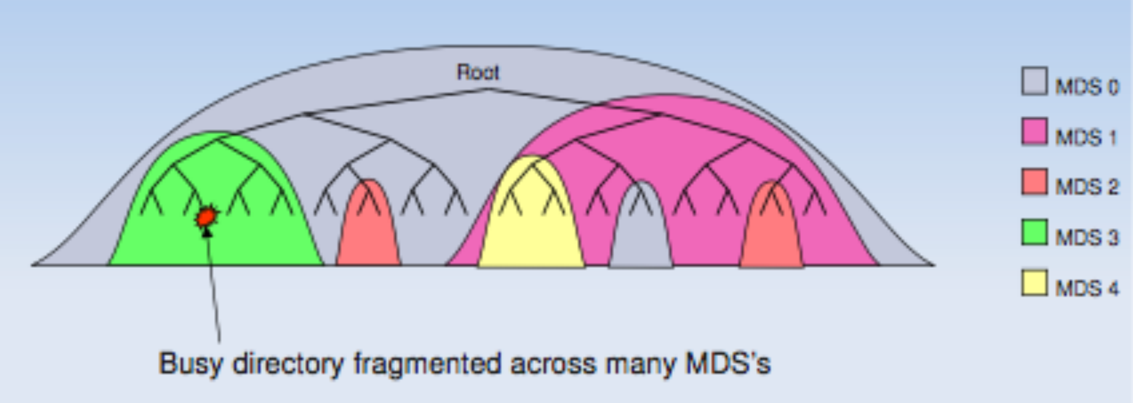


If, OSDs use Btrfs as their local file system, data is written asynchronously using copy-on-write, so that unsuccessful write operations can be fully rolled back.

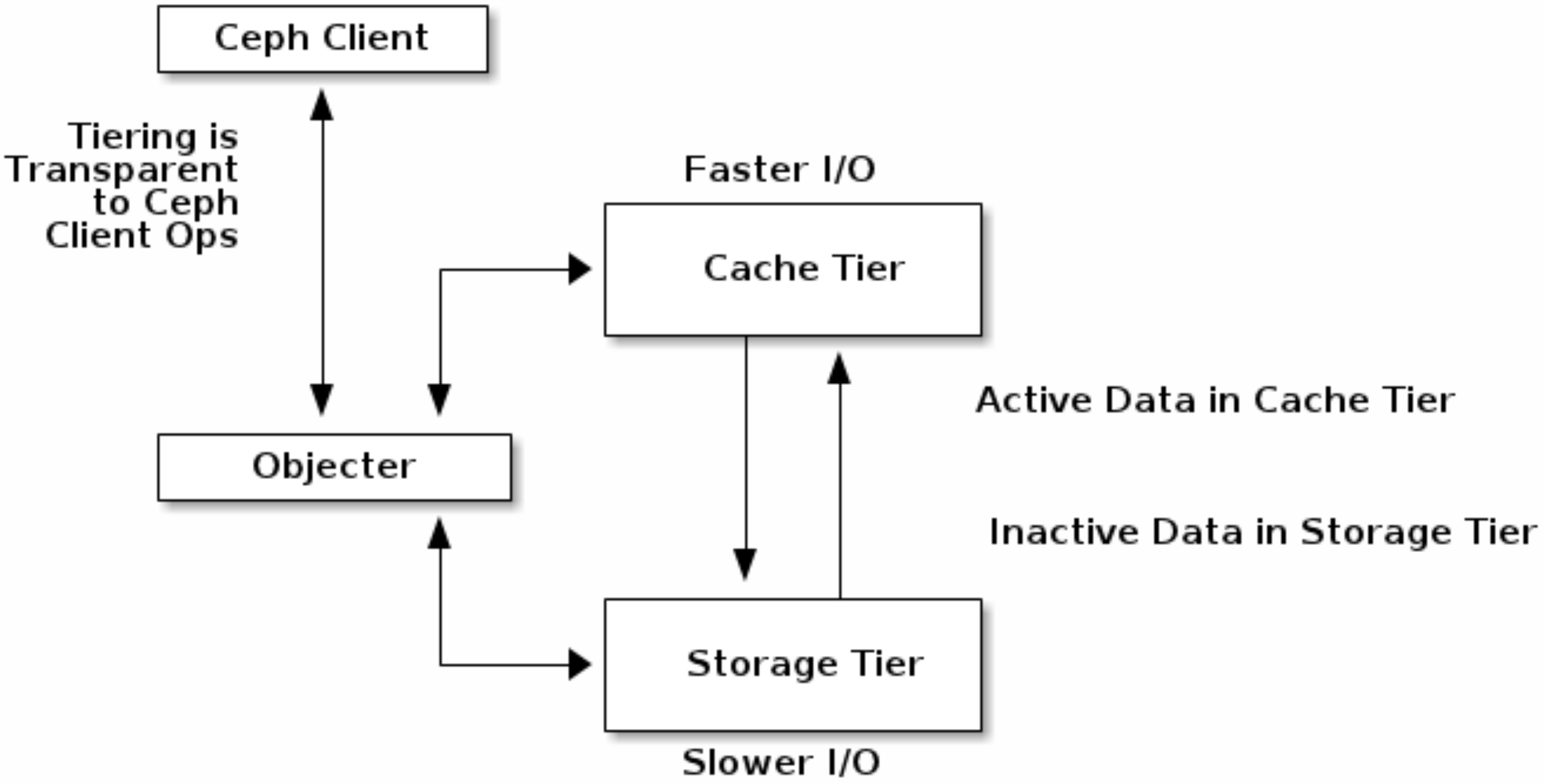
CEPH Architecture







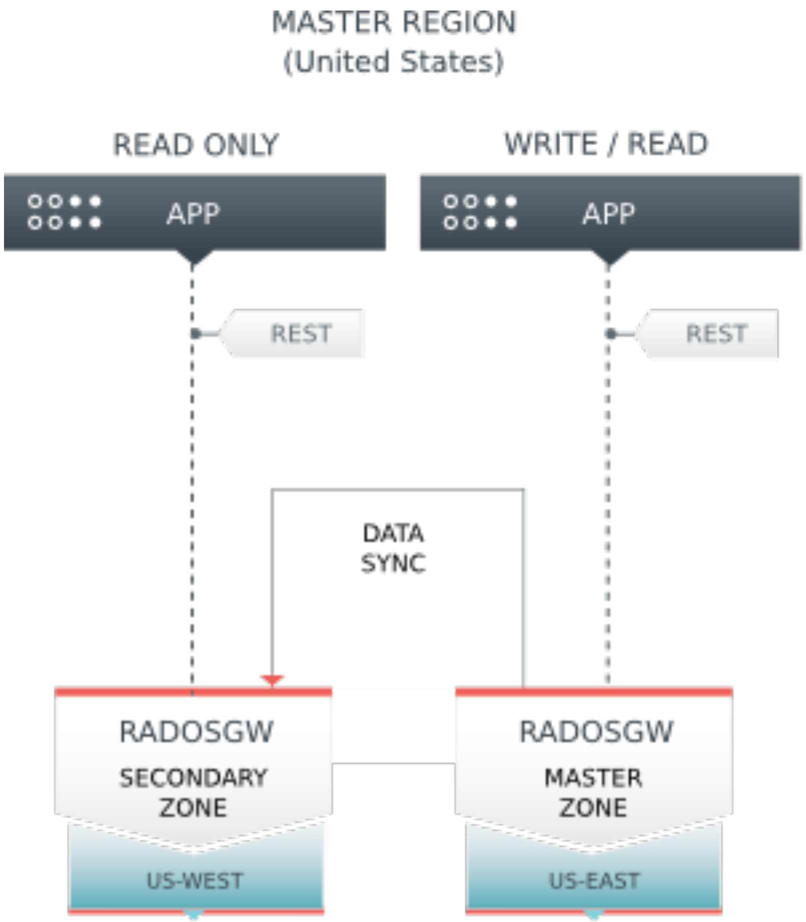
CEPH Architecture


Ceph Client
Tiering is Transparent to Ceph Client Ops
Faster I/O
Cache Tier
Active Data in Cache Tier
Objecter
Inactive Data in Storage Tier
Storage Tier
Slower I/O

CEPH Architecture

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**.



Architectural considerations – Redundancy and replication considerations

- Tradeoff between Cost vs. Reliability (use-case dependent)
- Use the Crush configs to map out your failures domains and performance pools
- Failure domains
  - Disk (OSD and OS)
  - SSD journals
  - Node
  - Rack
  - Site (replication at the RADOS level, Block replication, consider latencies)
- Storage pools
  - SSD pool for higher performance
  - Capacity pool
- Plan for failure domains of the monitor nodes
- Consider failure replacement scenarios, lowered redundancies, and performance impacts

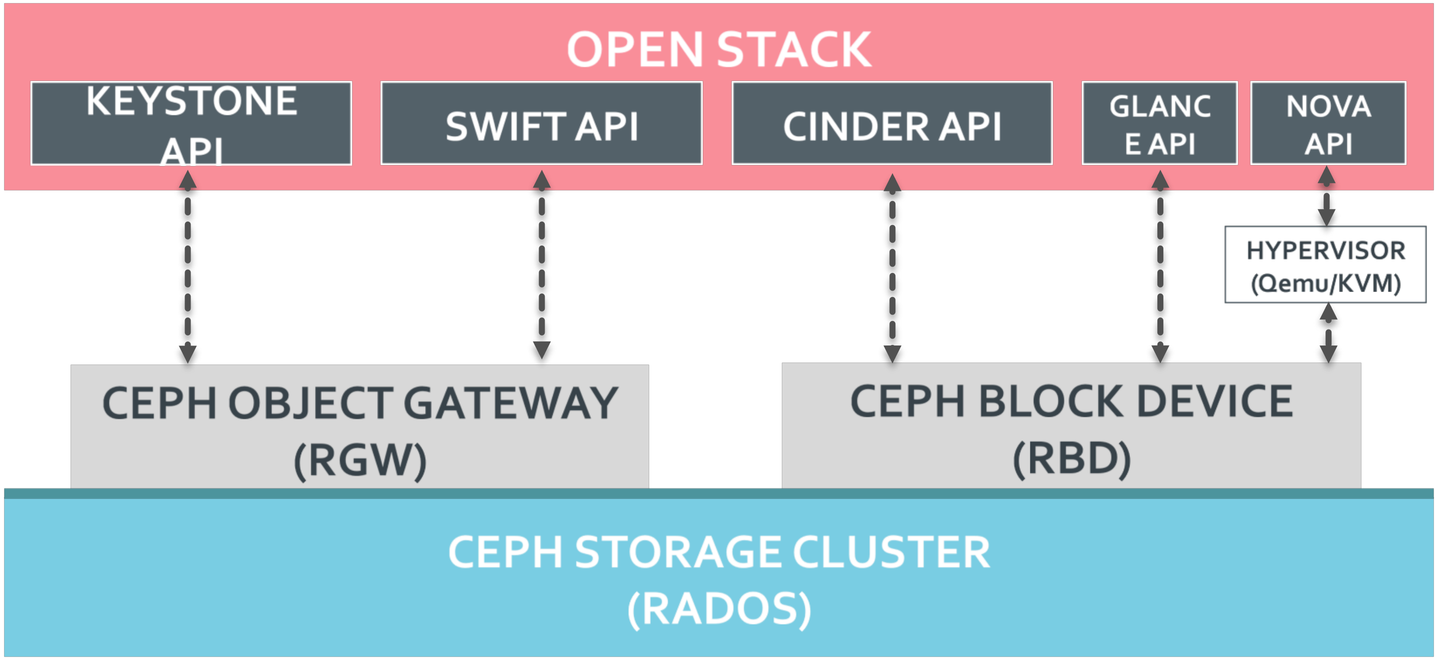
Server Considerations

- **Storage Node:**
  - one OSD per HDD, 1 – 2 GB ram, and 1 Gz/core/OSD,
  - SSD's for journaling and for using the tiering feature in Firefly
  - Erasure coding will increase useable capacity at the expense of additional compute load
  - SAS JBOD expanders for extra capacity (beware of extra latency and oversubscribed SAS lanes)
- **Monitor nodes (MON):** odd number for quorum, services can be hosted on the storage node for smaller deployments, but will need dedicated nodes larger installations
- **Dedicated RADOS Gateway nodes** for large object store deployments and for federated gateways for multi-site

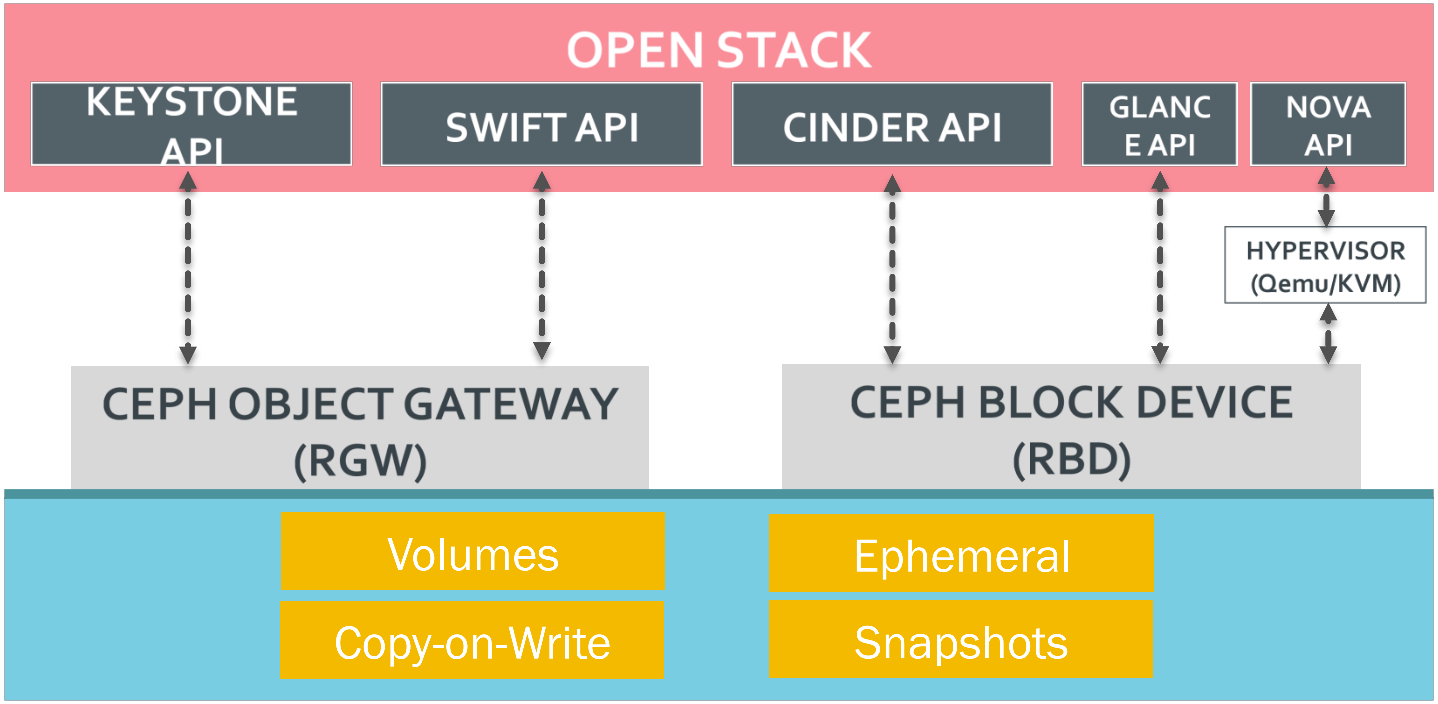
Networking Considerations

- **Dedicated or Shared network**
  - Be sure to involve the networking and security teams early when design your networking options
  - Network redundancy considerations
  - Dedicated client and OSD networks
  - VLAN's vs. Dedicated switches
  - 1 Gbs vs 10 Gbs vs 40 Gbs!
- **Networking design**
  - Spine and Leaf
  - Multi-rack
  - Core fabric connectivity
  - WAN connectivity and latency issues for multi-site deployments

CEPH & OpenStack



CEPH & OpenStack


OPEN STACK

KEYSTONE API

SWIFT API

CINDER API

GLANCE API

NOVA API

HYPERVISOR (Qemu/KVM)

CEPH OBJECT GATEWAY (RGW)

CEPH BLOCK DEVICE (RBD)

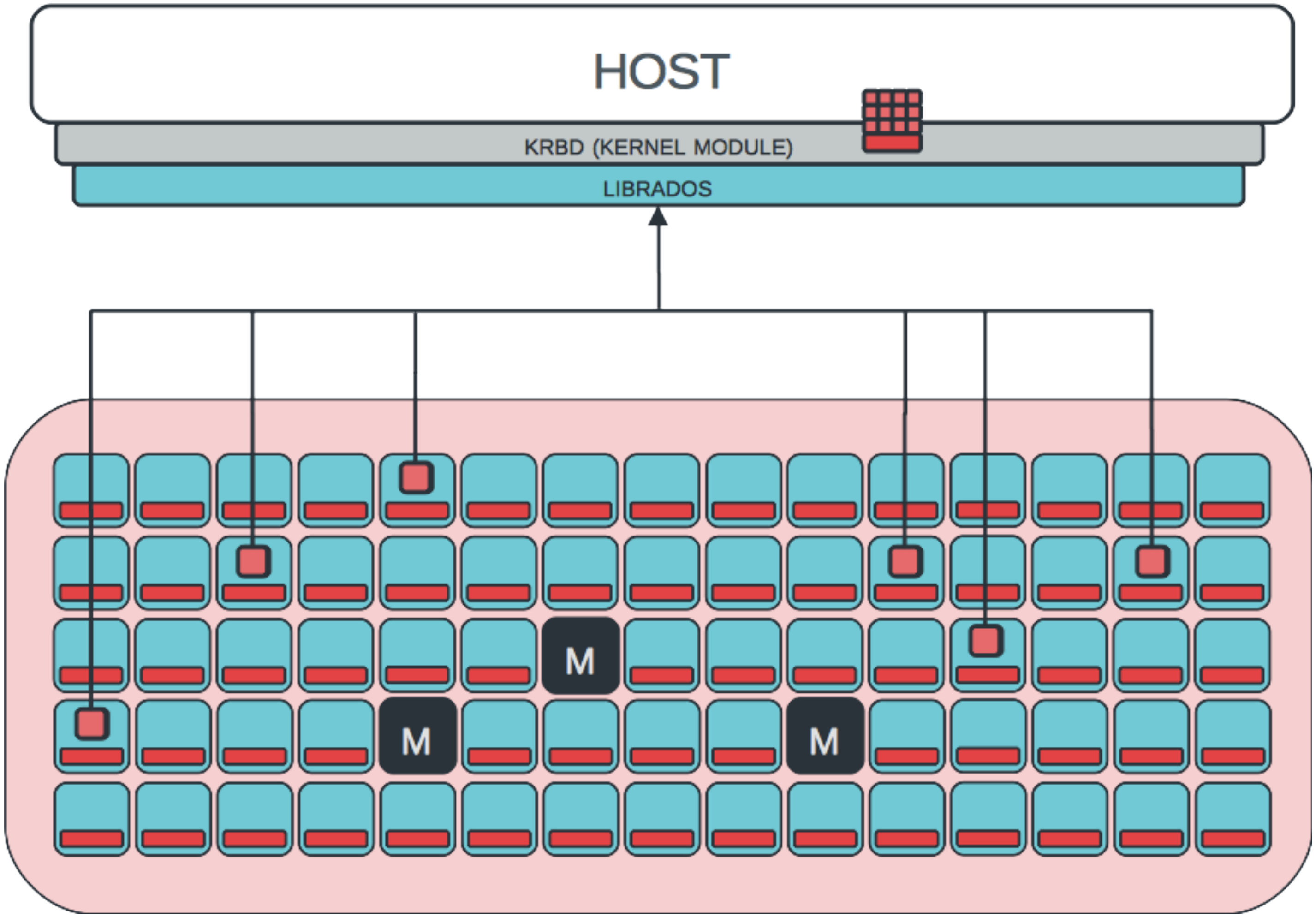
Volumes

Ephemeral

Copy-on-Write

Snapshots

CEPH & OpenStack



Link Utili

- https://ceph.com/docs/master/architecture/
- http://ceph.com/docs/master/start/intro/
- http://ceph.com/docs/master/release-notes/