Ceph Overview

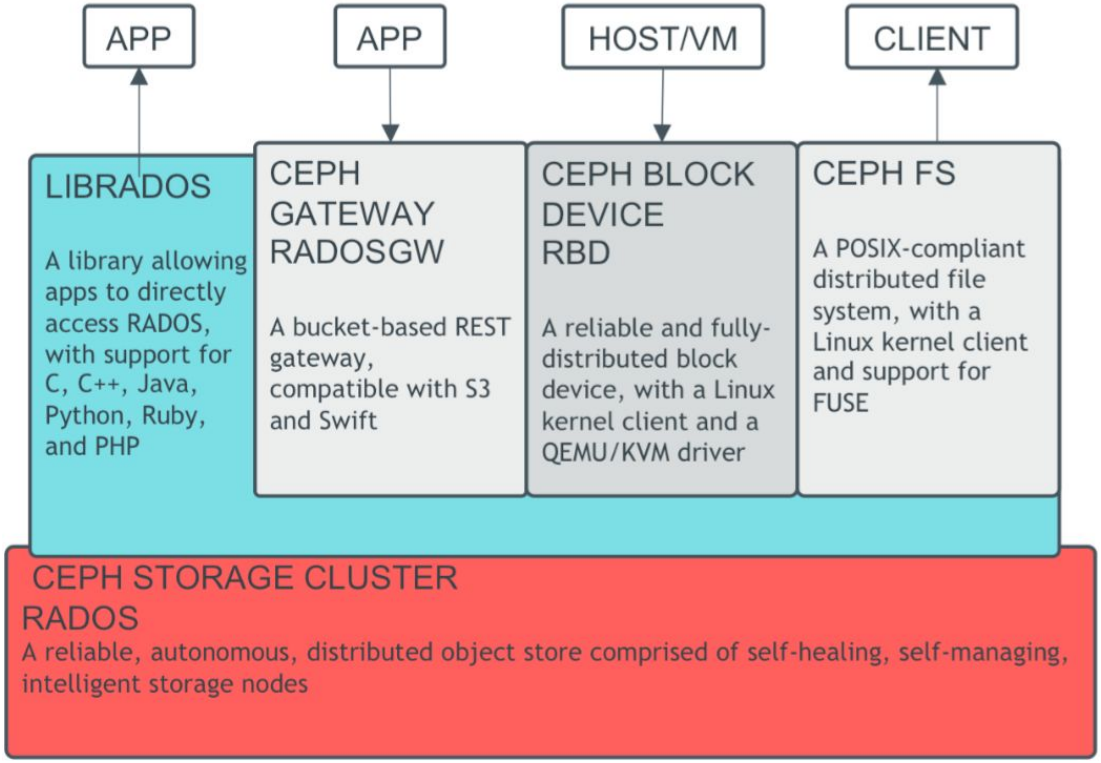
Steven Lee and Brenda Lapp

CAC Storage Challenge

- **Vendor Lock-In**
  - DDN storage is end-of-life
- **Fragmentation**
  - Fragmented storage systems: directly attached, Equallogic SAN
  - Too much stand by capacity
  - Unwanted storage cannot be easily re-deployed elsewhere
- **Scalable Object Storage**
  - We don't have one
- **Need A Solution Soon!**

Ceph: Unified File Storage

- **Object**
  - Native LIBRADOS (Reliable Autonomic Distributed Object Store)
  - RADOS Gateway provides S3/Swift REST API compatibility
- **Block:**
  - Linux kernel (krbd) and KVM (librbd) support
  - provides snapshotting and cloning capabilities
- **File:** CephFS
  - Not yet supported in production environment



Ceph Features

- **Hardware Agnostic**
  Runs on commodity servers with directly attached storage. Not locked in a specific hardware platform.

- **Flexible**
  Can define pools of storage with different redundancy rules (replication or erasure coding), disk types, geographic placement, depending on user requirements.

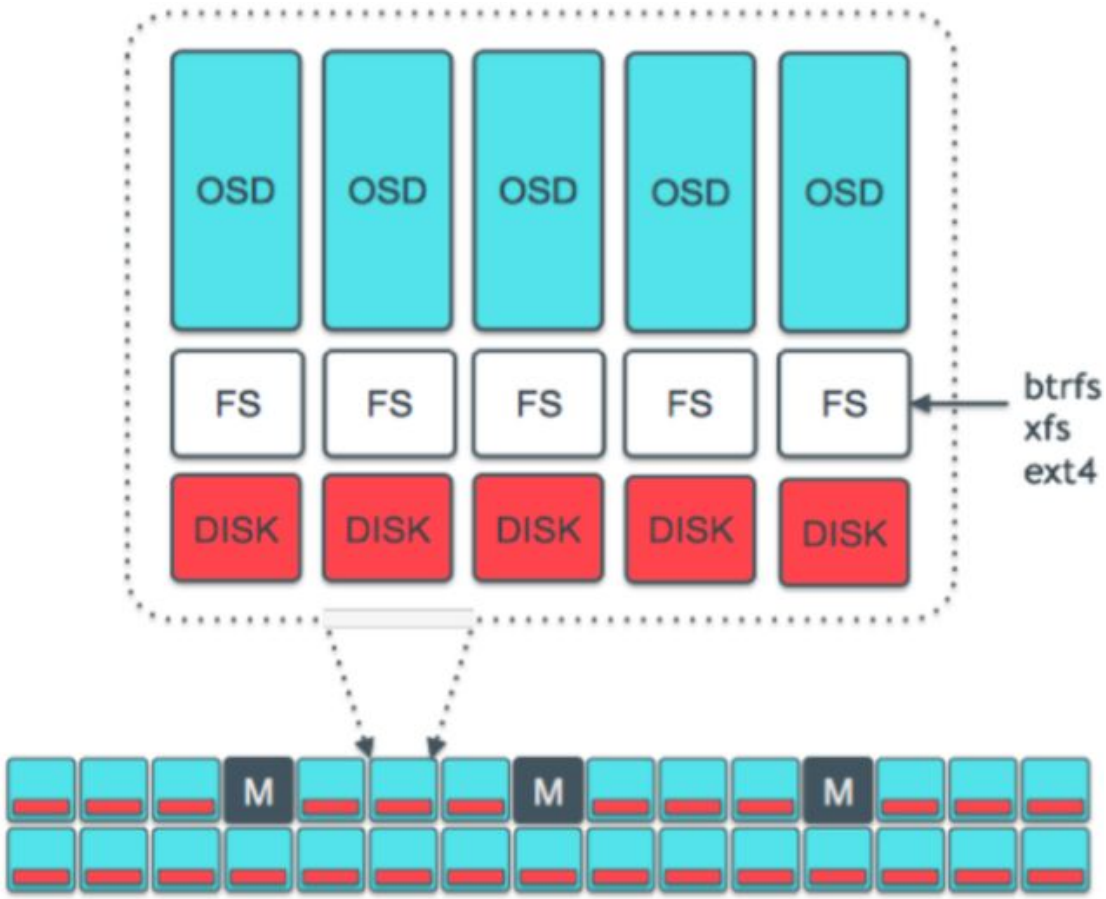
- **Scalable**
  In both bandwidth and capacity. No metadata servers. Silent clients do not generate network traffic/cluster load.

- **Self-Healing**
  Recovers automatically after disk or server failures.

Ceph Storage Cluster

A Ceph Cluster consists of:

- **Ceph Nodes**
  - Each node hosts multiple OSDs (object storage devices)
  - Each OSD has:
    - 1 hard disk
    - 1 xfs file system
    - 1 Ceph OSD daemon
    - (optional) journal hosted on a separate SSD
- **Monitors (min 3)**
- **Optional RADOSGW for providing S3/Swift compatibility**



Ceph Node Hardware Recommendations

- **CPU: 1 GHz (Hyperthreaded) modern CPU core per OSD**
- **RAM: 1 GB per each TB of hosted data**
- **No more than 16 OSDs per node in I/O intensive environments (e.g. hosting block devices)**
- **SSD: max 6 OSD journals per SATA or SAS connected SSD**

Ceph OSD Daemon

- **OSD is *primary* for some objects:**
  - Serves data to clients
  - Maps versions are reconciled with the clients as part of each transaction. Update clients with new maps when needed.
  - Responsible for replication
  - Responsible for coherency
  - Responsible for re-balancing
  - Responsible for recovery
- **OSD is *secondary* for some objects:**
  - Controlled by primary OSD
  - Can become primary
- **Transactions with clients are atomic**
- **Notifies monitors when disk or other OSD failures are detected**

Monitors (MON)

- **Maintain the cluster maps**
  - MON Map
  - OSD Map
  - MDS Map
  - PG Map
  - CRUSH Map
- **Provide consensus for distributed decision making**
  - Each MON knows about all the other MONs in the cluster
  - First establish a quorum of more than half of MONs so an odd number of MONs is needed in the cluster
  - MONs in a quorum can distribute maps to OSDs and clients. Clients request cluster maps from a MON only when the primary OSD fails.
- **Monitors are not in data path**

Data Placement

Pools

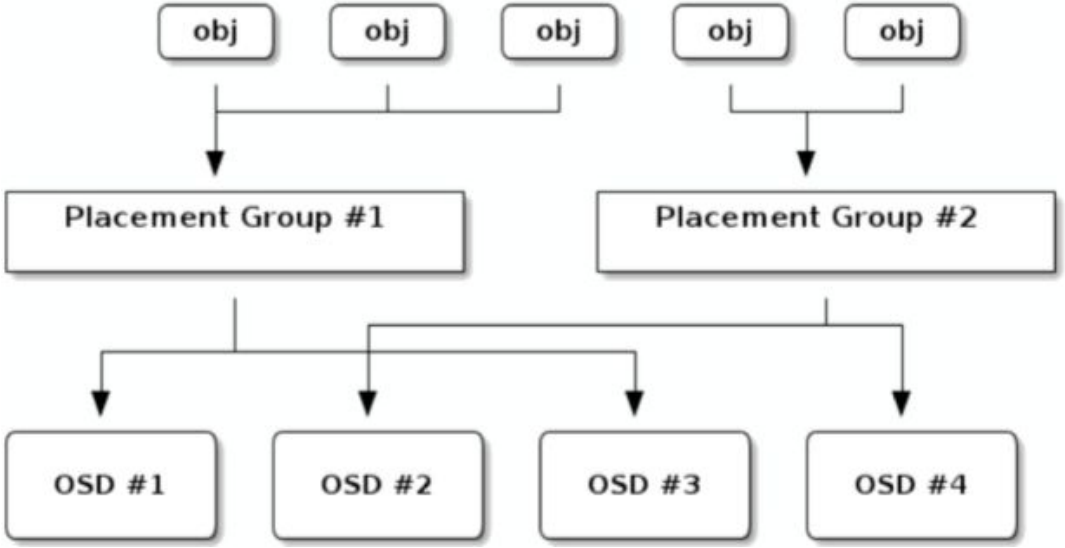
Pools are logical partitions in the cluster with the following attributes:

- **Ownership / access**
- **Protection type: replicated or erasure coding**
- **Number of placement groups**
- **CRUSH placement rules**

Data Placement (Cont'd)

Placement Group (PG)

- **The cluster is split into Placement Groups (PGs)**
- **A PG contains a collection of objects**
- **An object's PG is determined by CRUSH hashing the object name against the number of PGs in the pool.**
- **A PG's location is determined by CRUSH according to desired protection and placement strategies.**



Data Placement (Cont'd)

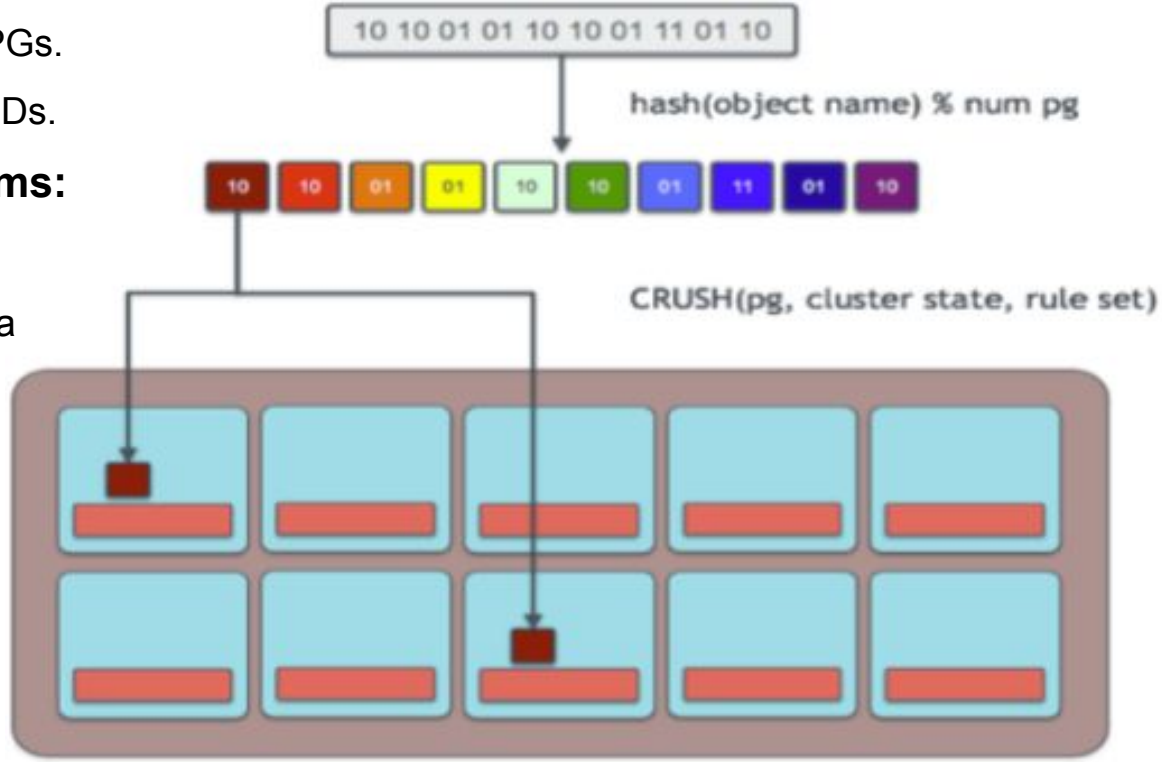
OSG States

- **In or out: an OSD is IN if PGs are mapped to it.**
- **Up or Down: an OSD is Up if it can serve data.**
- **Example:**
  - **An OSD is In and Up if it serves its data normally.**
  - **An OSD is In and Down if it contains objects but cannot serve the data (temporary failure).**

Data Placement (Cont'd)

CRUSH (Controlled Replication Under Scalable Hashing)

- **CRUSH is Ceph's data distribution mechanism:**
  - Distributes objects into PGs.
  - Distributes PGs onto OSDs.
- **Psuedo-random algorithms:**
  - Deterministic
  - Clients can compute data location from cluster maps. No metadata server required.



Data Placement (Cont'd)

CRUSH (Cont'd)

- **Rule-based configuration**
  - replica / erasure coding
  - OSD weight: how much data should go to an OSD.
  - OSD primary affinity: which OSD will be selected as primary
  - CRUSH Map:
    - List of all Ceph nodes and OSDs
    - Buckets: definition of existing infrastructure (sites, rows, racks, servers)
    - Placement rules

Failure Is the Norm

Ceph OSD Failure

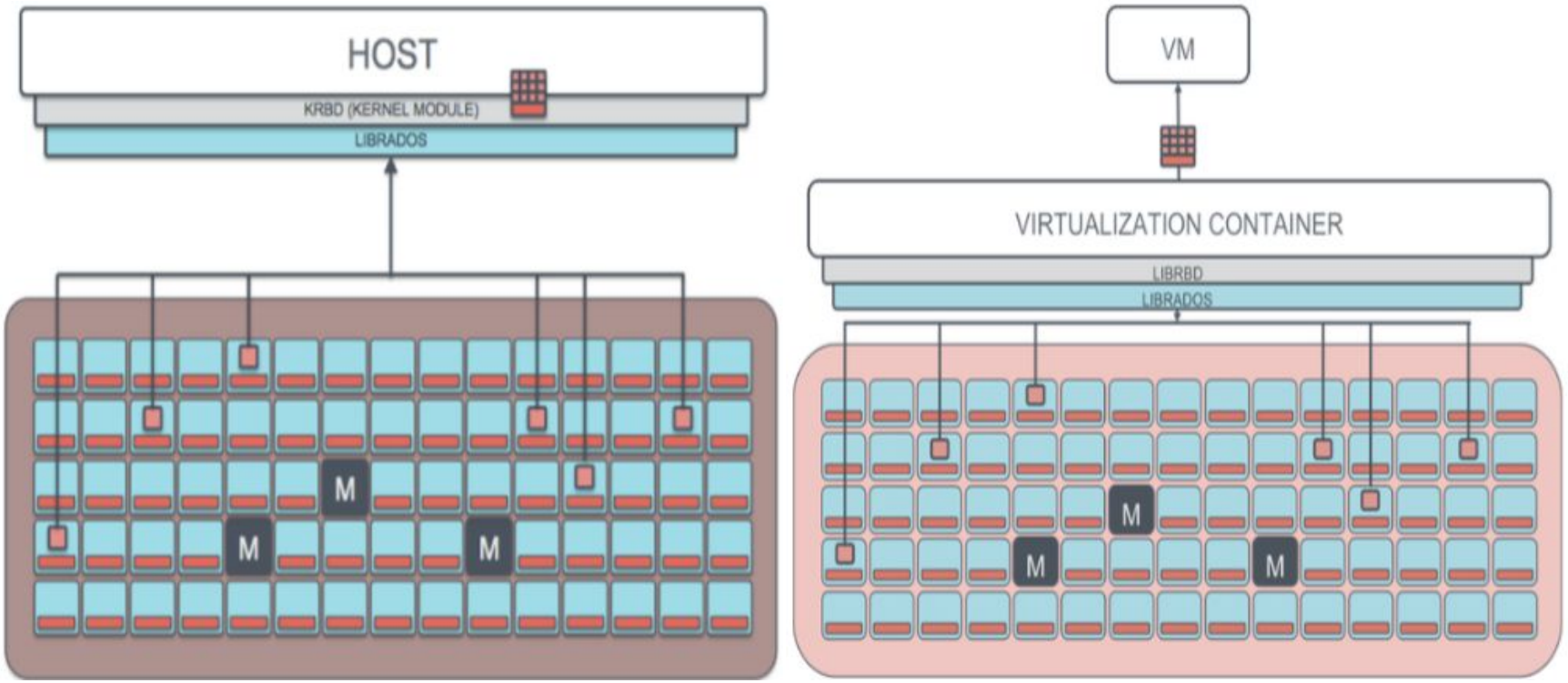
- **ceph-osd daemon dies**
  - Peers heartbeats fail; peers inform monitor
  - New OSD map published with osd.123 as ‘down’
- **PG maps to fewer replicas**
  - If osd.123 was primary in a PG, a secondary OSD takes over
    - What happens if a client tries to contact osd.123 to access objects stored there?
  - PG is “degraded” (N-1 replicas)
  - Data redistribution is not triggered
- **Monitor marks OSD “out” after 5 minutes (configurable)**
  - PG now maps to N OSDs again
  - PG re-peers, activates
  - Primary replicates to the “new” OSD

Ceph Journal

- **Each OSD has a journal implemented by a ring buffer**
  - By default the journal is store on a partition on the OSD's disk.
  - Optionally the journal can be on a separate shared SSD on the same node. A SATA or SAS connected SSD can host journals for up to 6 OSDs; a NVMe SSD can host up to 12.
- **Ceph OSDs**
  - Record each write operation to the journal before reporting the operation as completed
  - Commit the operation to the file system whenever possible
  - Replay the journal upon OSD restart.
  - Serve read request from the file system and never the journal.
- **The primary OSD acknowledges to the client only after all secondaries report their write operations as completed (i.e. record the write operation in journal)**

Ceph Block Device

- **Native support for both physical servers (krbd) and KVM VMs (librbd)**



Ceph Block Device Features

- **Need to host RBDs in replicated pools; erasure coded pools are not supported.**
- **Snapshots**
  - Instantly created
  - Read-only
  - Do not take up space… until original data changes (copy-on-write)
- **Clones**
  - Are copies of snapshots
  - Writable
  - Do not take up space… until original data changes or clones are written to (copy-on-write)

Integration with Cloud Stacks

- **Eucalyptus**
  - Currently supported for block storage (EBS volumes)
  - Currently not supported for object storage. Work is being done on using Ceph for object storage, either using native librados or RADOSGW.
- **OpenStack**
  - Supported for both object and block storage.