INTRO TO CEPH

OPEN SOURCE DISTRIBUTED STORAGE

Neha Ojha
Project Technical Lead for RADOS
2021.11.17

INTRO TO CEPH

OPEN SOURCE DISTRIBUTED STORAGE

- What is Ceph
  - and why do we care
- Ceph Architecture
  - RADOS
  - RGW - Object
  - RBD - Block
  - CephFS - File
- Management
- Community and Ecosystem

WHAT IS CEPH?

The buzzwords

- “Software defined storage”
- “Unified storage system”
- “Scalable distributed storage”
- “The future of storage”
- “The Linux of storage”

The substance

- Ceph is open source **software**
- Runs on commodity hardware
  - Commodity servers
  - IP networks
  - HDDs, SSDs, NVMe, NV-DIMMs, ...
- A single cluster can serve **object**, **block**, and **file** workloads

CEPH IS FREE AND OPEN SOURCE

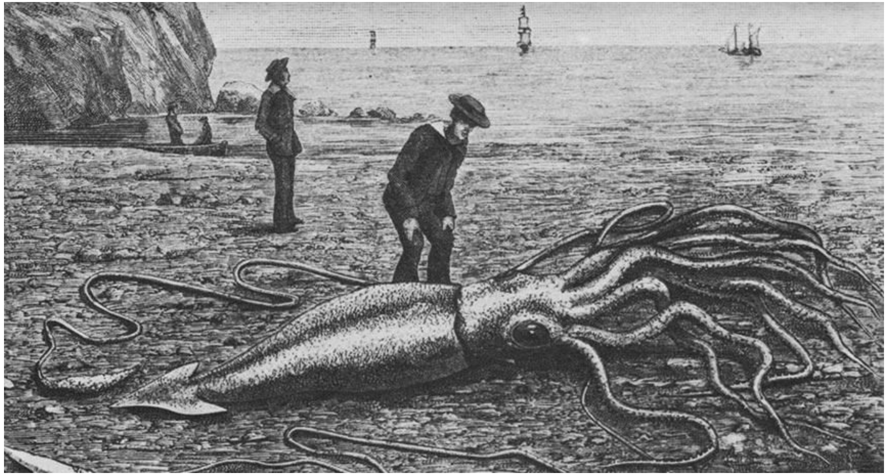
- Freedom to use (free as in beer)
- Freedom to introspect, modify, and share (free as in speech)
- Freedom from vendor lock-in
- Freedom to innovate

CEPH IS RELIABLE

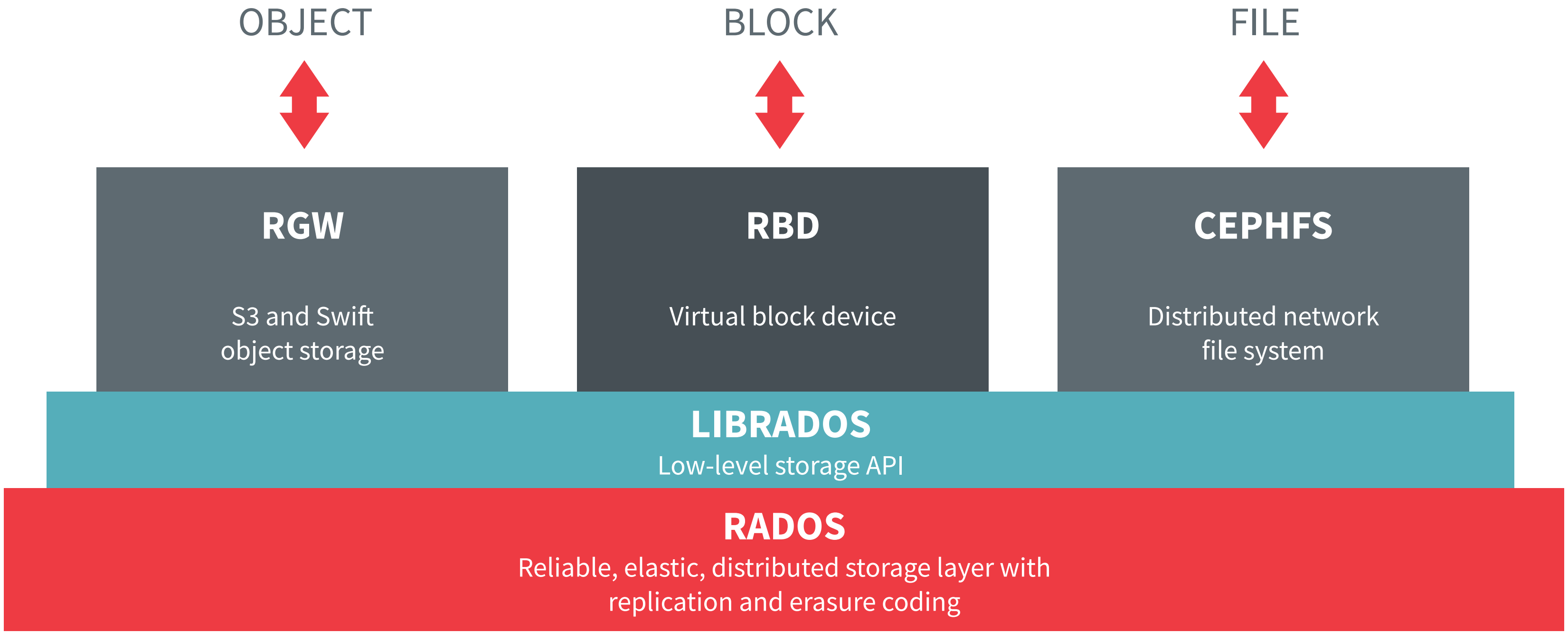
- **Reliable storage** service out of **unreliable components**
  - No single point of failure
  - Data durability via replication or erasure coding
  - No interruption of service from rolling upgrades, online expansion, etc.
- Favor consistency and correctness over performance

CEPH IS SCALABLE

- Ceph is elastic storage infrastructure
  - Storage cluster may grow or shrink
  - Add or remove hardware while system is online and under load
- Scale **up** with bigger, faster hardware
- Scale **out** within a single cluster for capacity and performance
- **Federate** multiple clusters across sites with asynchronous replication and disaster recovery capabilities



CEPH IS A UNIFIED STORAGE SYSTEM



RADOS

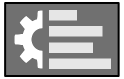
RADOS

- **R**eliable **A**utonomic **D**istributed **O**bject **S**torage
  - Common storage layer underpinning object, block, and file services
- Provides low-level data object storage service
  - Reliable and highly available
  - Scalable (on day 1 and day 1000)
  - Manages all replication and/or erasure coding, data placement, rebalancing, repair, etc.
- Strong consistency
  - CP, not AP
- Simplifies design and implementation of higher layers (file, block, object)

RADOS SOFTWARE COMPONENTS

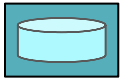
ceph-mon

Monitor
- Central authority for authentication, data placement, policy
- Coordination point for all other cluster components
- Protect critical cluster state with Paxos
- 3-7 per cluster



ceph-mgr

Manager
- Aggregates real-time metrics (throughput, disk usage, etc.)
- Host for pluggable management functions
- 1 active, 1+ standby per cluster

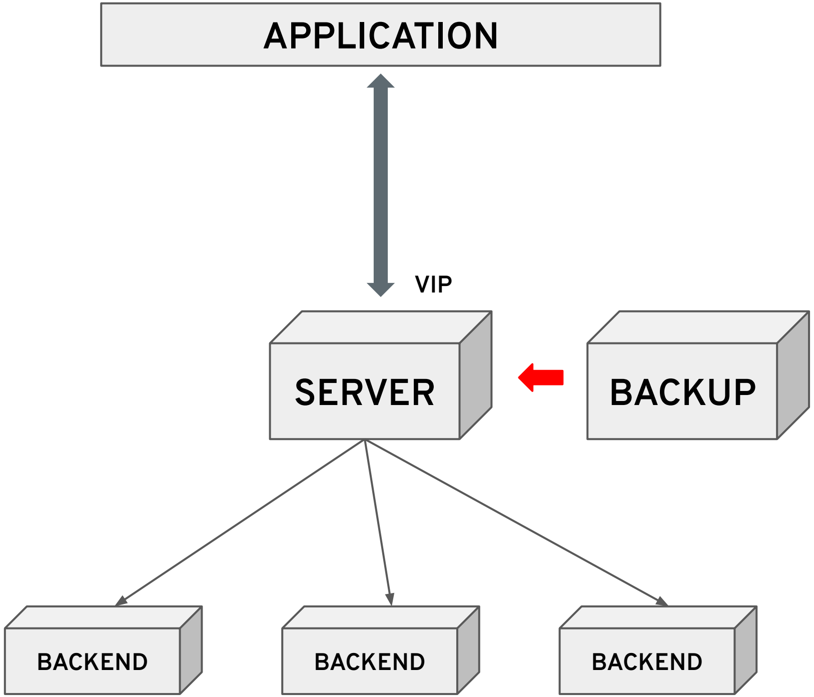


ceph-osd

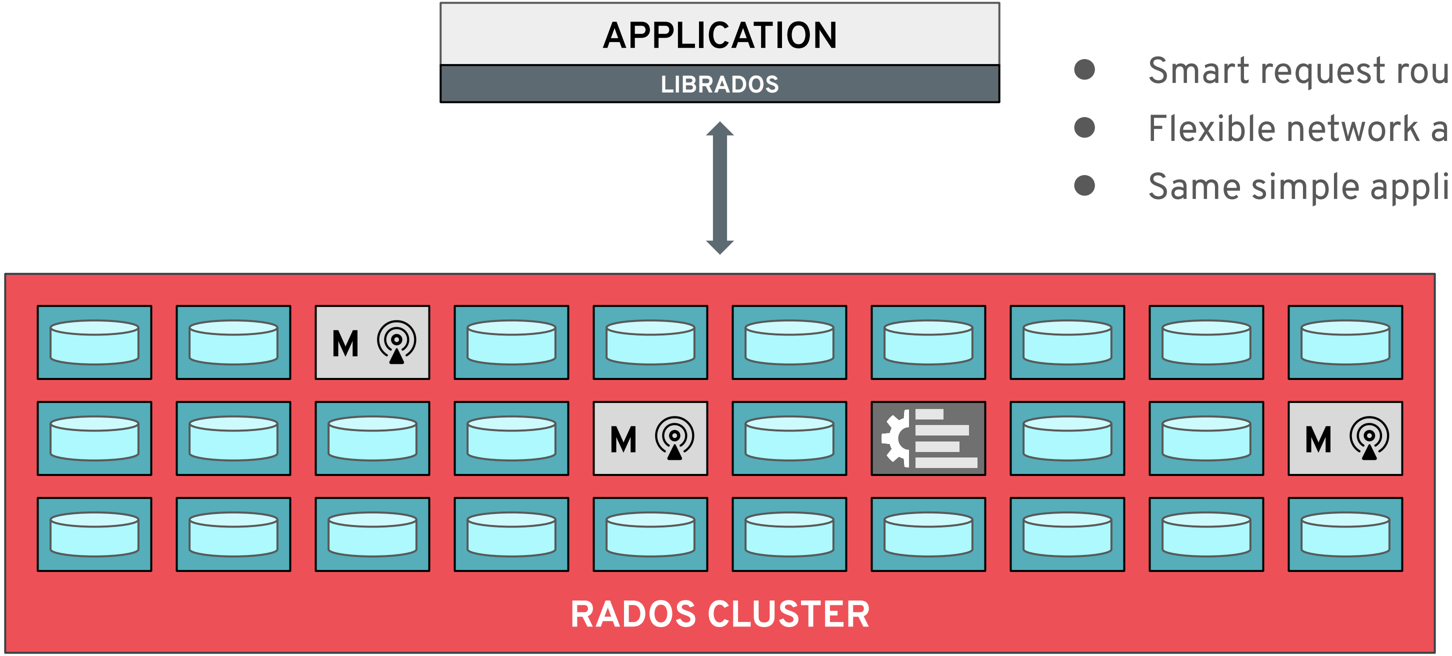
OSD (Object Storage Daemon)
- Stores data on an HDD or SSD
- Services client IO requests
- Cooperatively peers, replicates, rebalances data
- 10s-1000s per cluster

LEGACY CLIENT/SERVER ARCHITECTURE

- Virtual IPs
- Failover pairs
- Gateway nodes

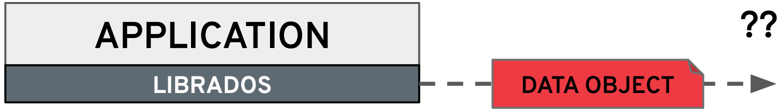


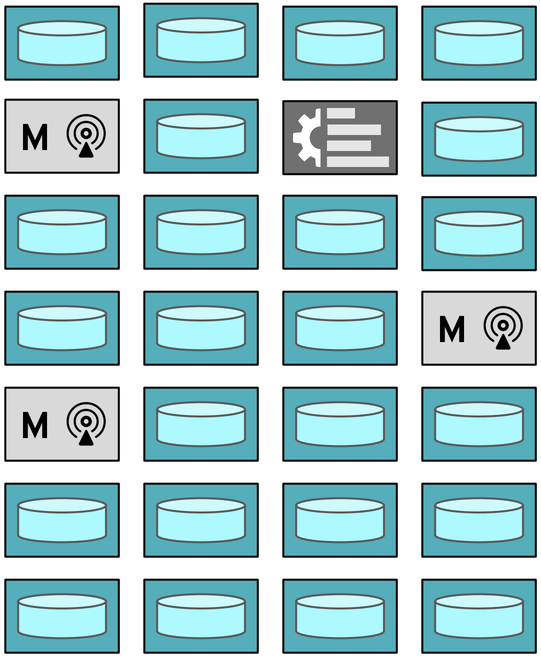
CLIENT/CLUSTER ARCHITECTURE



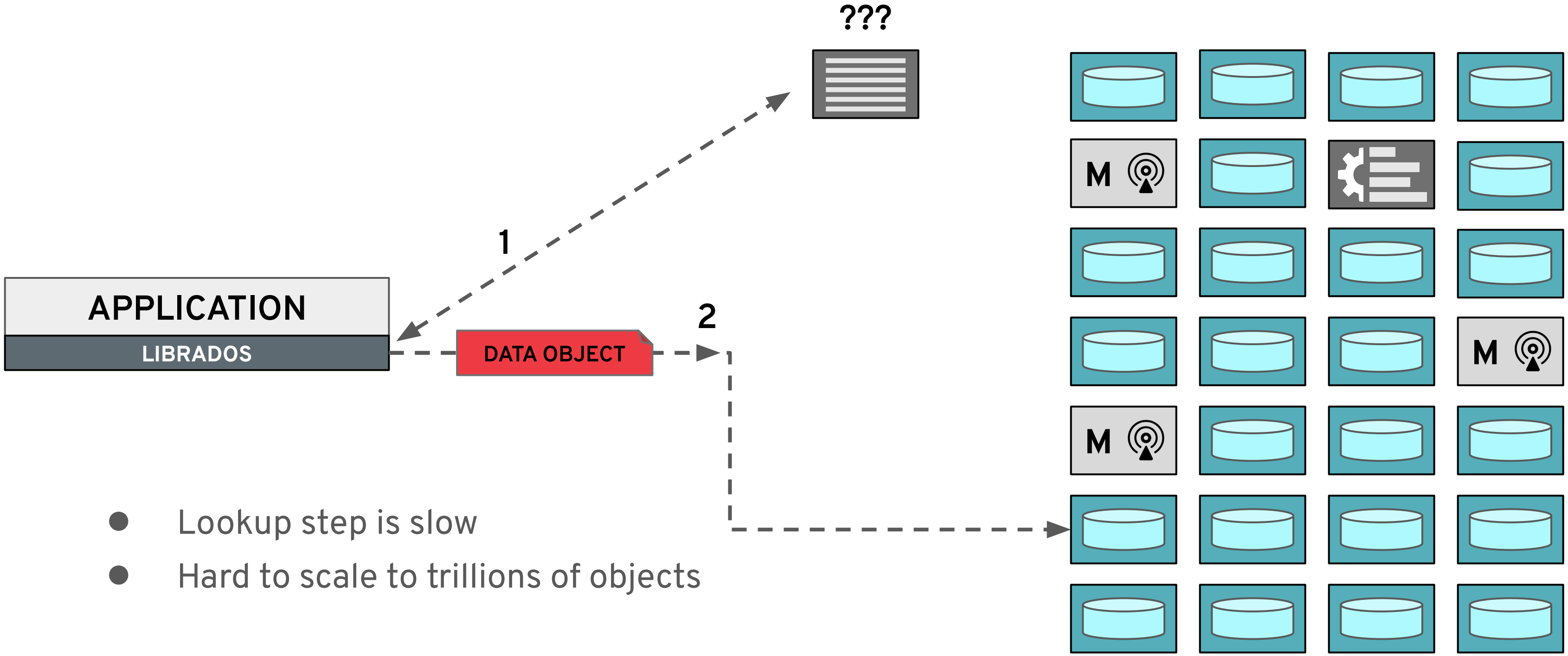
- Smart request routing
- Flexible network addressing
- Same simple application API

DATA PLACEMENT



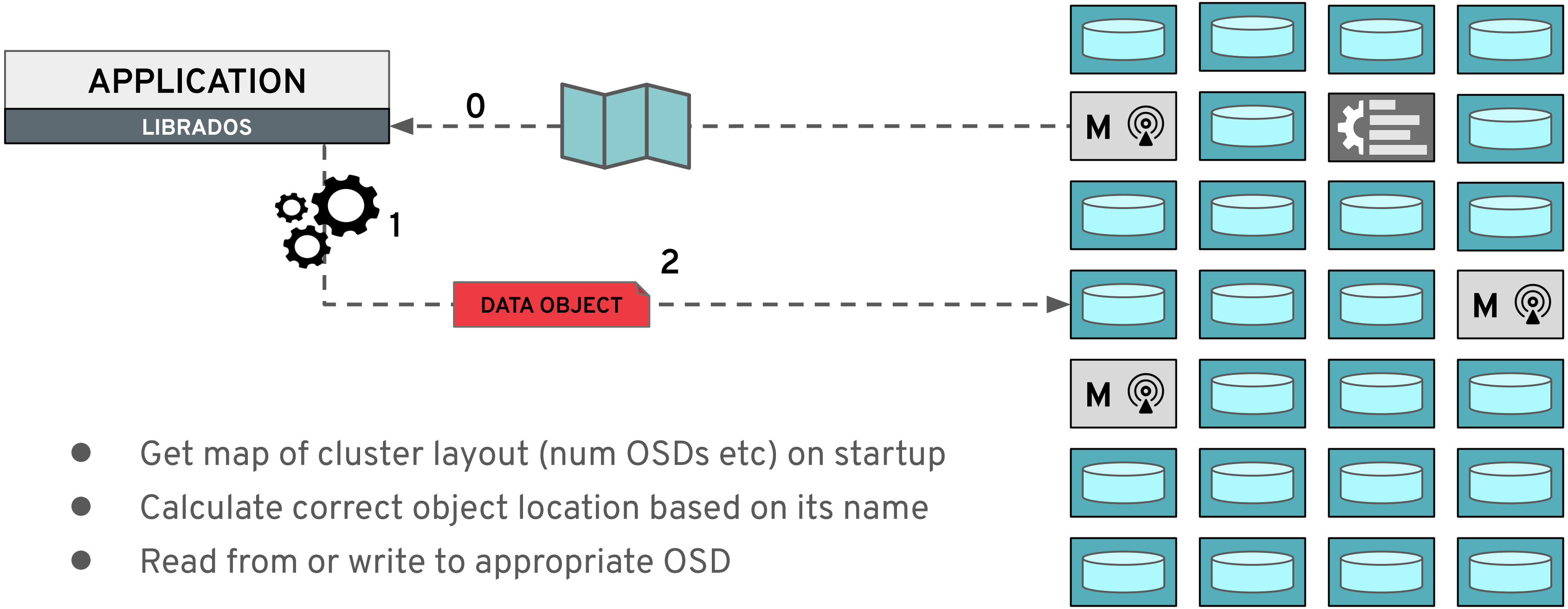


LOOKUP VIA A METADATA SERVER?



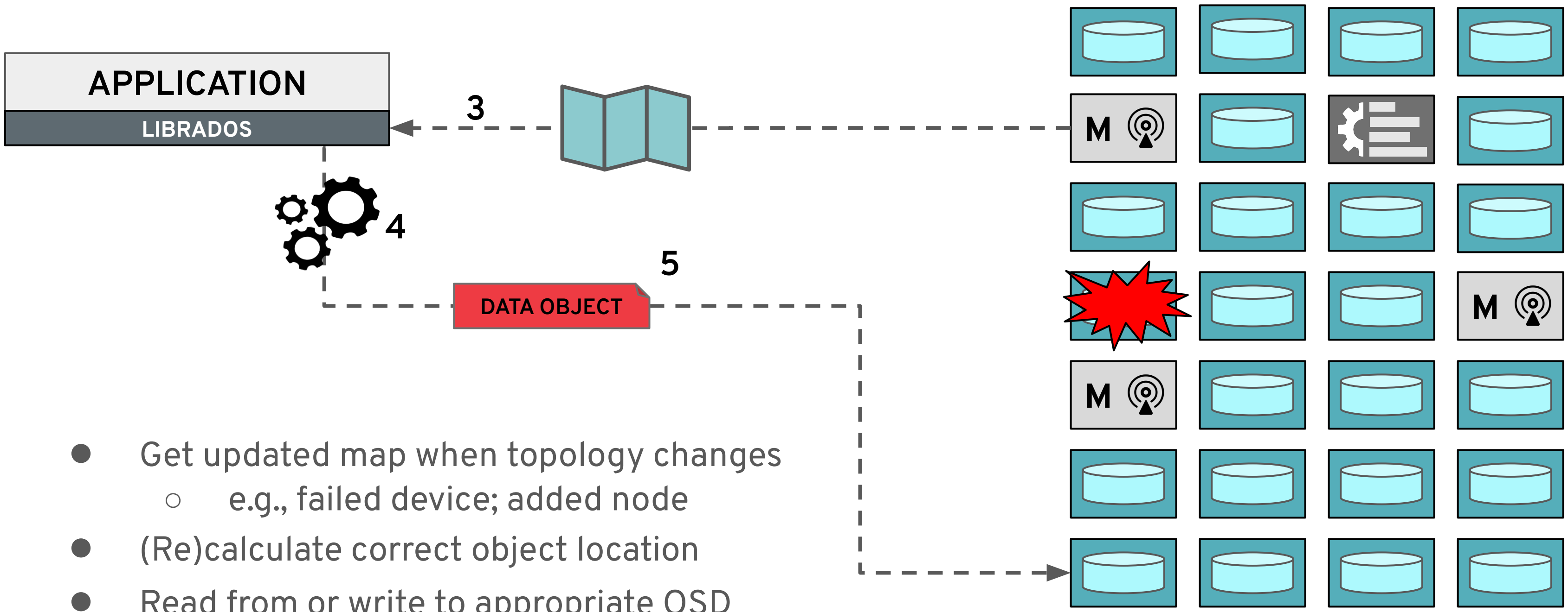
- Lookup step is slow
- Hard to scale to trillions of objects

CALCULATED PLACEMENT



- Get map of cluster layout (num OSDs etc) on startup
- Calculate correct object location based on its name
- Read from or write to appropriate OSD

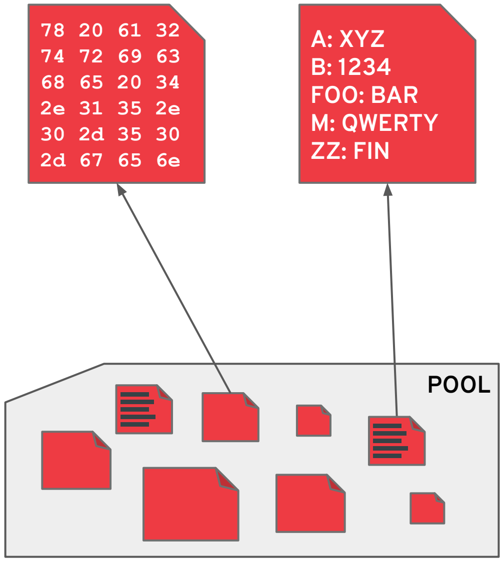
MAP UPDATES WHEN TOPOLOGY CHANGES



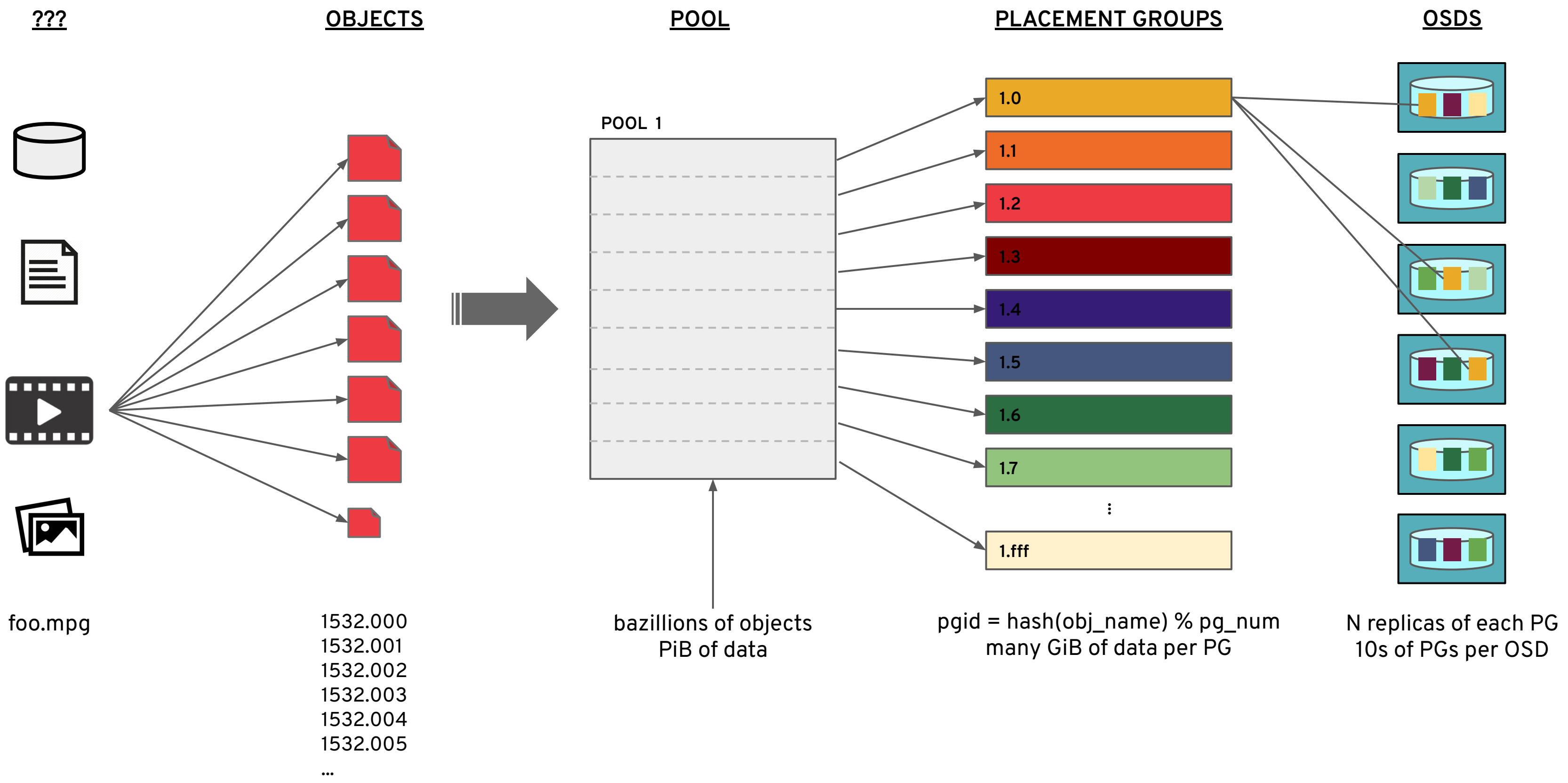
- Get updated map when topology changes
  - e.g., failed device; added node
- (Re)calculate correct object location
- Read from or write to appropriate OSD

RADOS DATA OBJECTS

- Name
  - 10s of characters
  - e.g., “rbd_header.10171e72d03d”
- Attributes
  - 0 to 10s of attributes
  - 0 to 100s of bytes each
  - e.g., “version=12”
- Byte data
  - 0 to 10s of megabytes
- Key/value data (“omap”)
  - 0 to 10,000s of items
  - 0 to 10,000s of bytes each
- Objects live in named “pools”

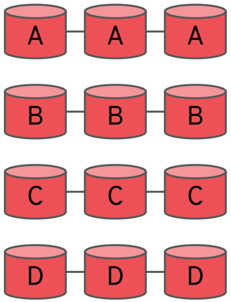


? → OBJECTS → POOLS → PGs → OSDs



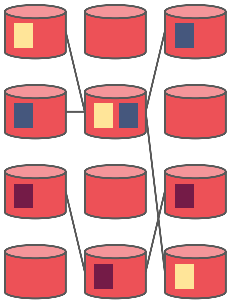
WHY PLACEMENT GROUPS?

REPLICATE DISKS



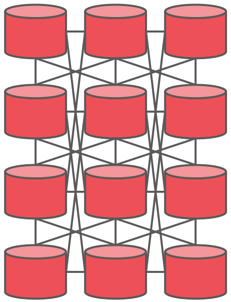
- Each device is mirrored
- Device sizes must match

REPLICATE PGS



- Each PG is mirrored
- PG placement is random

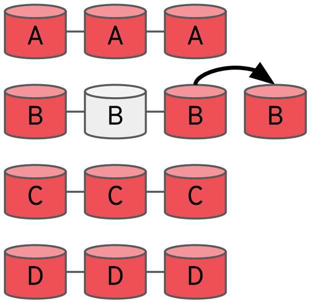
REPLICATE OBJECTS



- Each object is mirrored
- Object placement is random

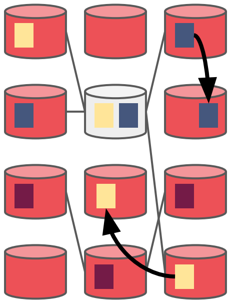
WHY PLACEMENT GROUPS?

REPLICATE DISKS



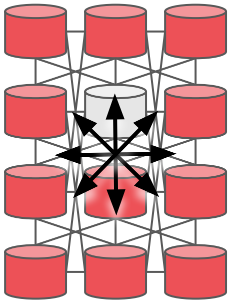
- Need an empty spare device to recover
- Recovery bottlenecked by single disk throughput

REPLICATE PGS



- New PG replicas placed on surviving devices
- Recovery proceeds in parallel, leverages many devices, and completes sooner
- No spare needed

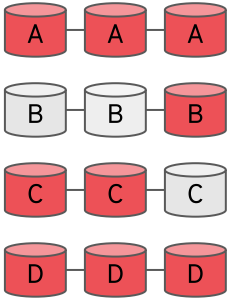
REPLICATE OBJECTS



- *Every* device participates in recovery

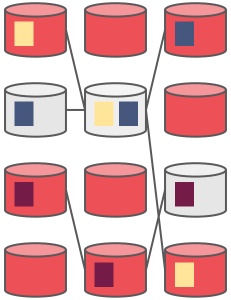
WHY PLACEMENT GROUPS?

REPLICATE DISKS



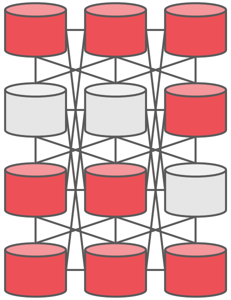
- Very few triple failures cause data loss (of an entire disk)

REPLICATE PGS



- Some triple failures cause data loss (of an entire PG)

REPLICATE OBJECTS



- **Every** triple failure causes data loss (of some objects)

PGs balance competing extremes

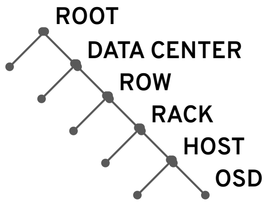
KEEPING DATA SAFE

“Declustered replica placement”

- More clusters
  - Faster recovery
  - More even data distribution
- Fewer clusters
  - Lower risk of concurrent failures affecting all replicas
- Placement groups a happy medium
  - No need for spare devices
  - Adjustable balance between durability (in the face of concurrent failures) and recovery time

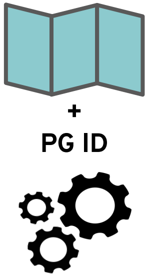
Avoiding concurrent failures

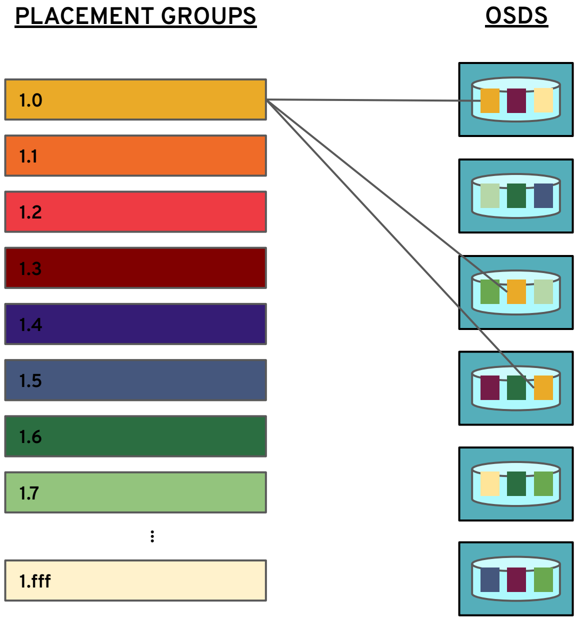
- Separate replicas across failure domains
  - Host, rack, row, datacenter
- Create a hierarchy of storage devices
  - Align hierarchy to physical infrastructure
- Express placement policy in terms hierarchy



PLACING PGs WITH CRUSH

- Pseudo-random placement algorithm
  - Repeatable, deterministic, calculation
  - Similar to "consistent hashing"
- Inputs:
  - Cluster topology (i.e., the OSD hierarchy)
  - Pool parameters (e.g., replication factor)
  - PG id
- Output: ordered list of OSDs
- Rule-based policy
  - "3 replicas, different racks, only SSDs"
  - "6+2 erasure code shards, 2 per rack, different hosts, only HDDs"
- Stable mapping
  - Limited data migration on change
- Support for varying device sizes
  - OSDs get PGs proportional to their weight



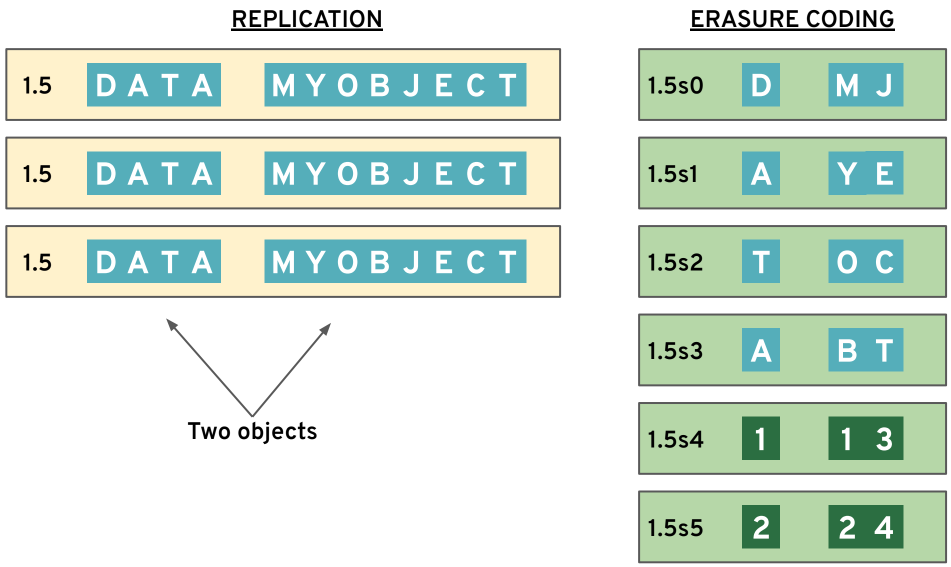


pgid = hash(obj_name) % pg_num
many GiB of data per PG

N replicas of each PG
10s of PGs per OSD

REPLICATION AND ERASURE CODING

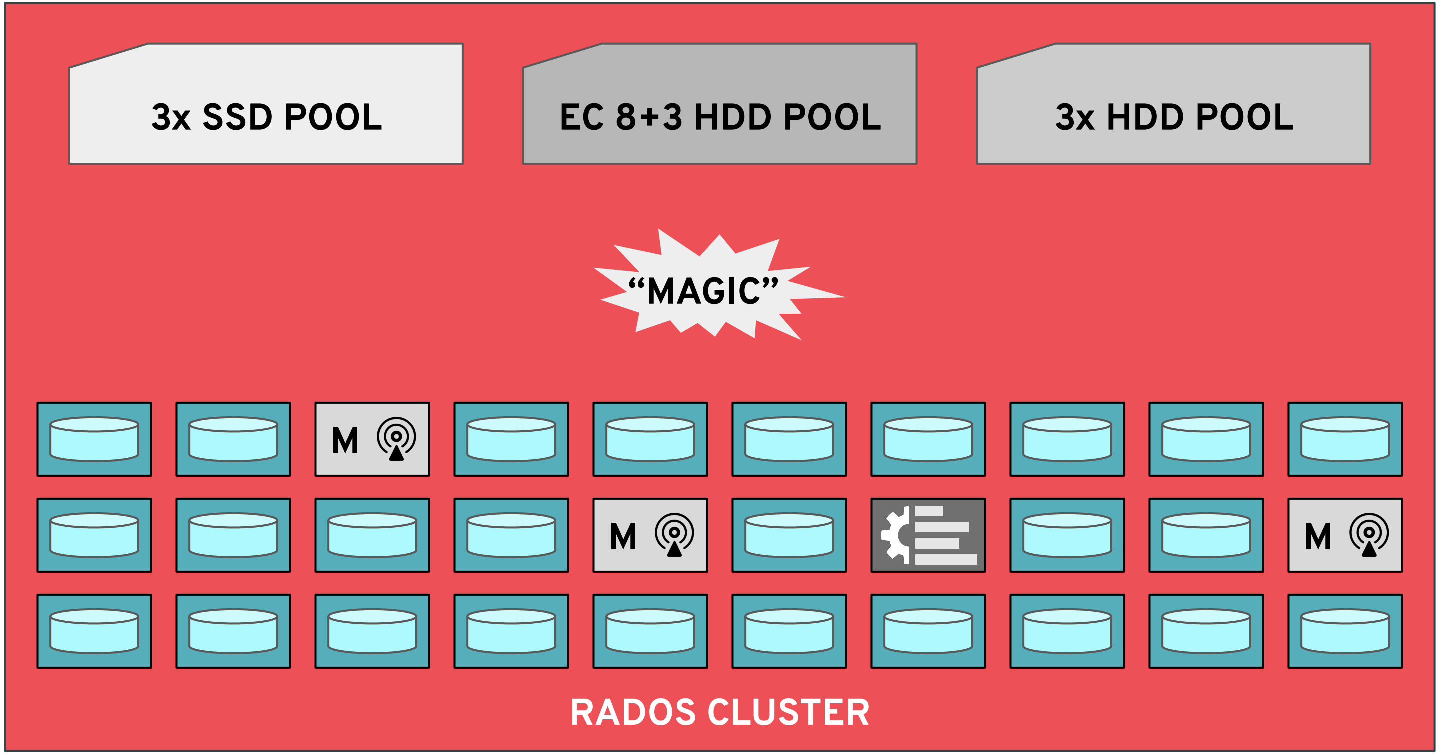
- Each RADOS pool must be durable
- Each PG must be durable
- Replication
  - Identical copies of each PG
  - Usually 3x (200% overhead)
  - Fast recovery--read any surviving copy
  - Can vary replication factor at any time
- Erasure coding
  - Each PG “shard” has different slice of data
  - Stripe object across **k** PG shards
  - Keep addition **m** shards with per-object parity/redundancy
  - Usually more like 1.5x (50% overhead)
  - Erasure code algorithm and **k+m** parameters set when pool is created
  - Better for large objects that rarely change



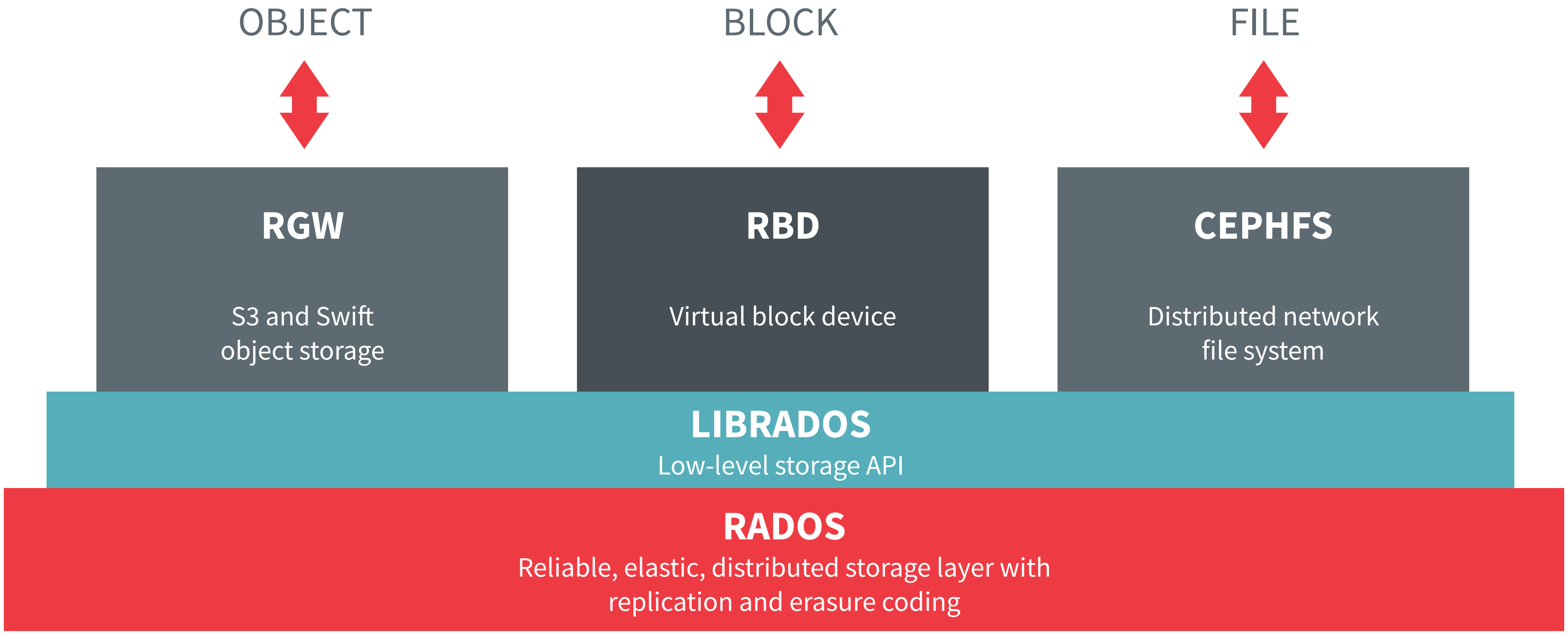
SPECIALIZED POOLS

- Pools usually share devices
  - Unless a pool's CRUSH placement policy specifies a specific class of device
- Elastic, scalable provisioning
  - Deploy hardware to keep up with demand
- Uniform management of devices
  - Common "day 2" workflows to add, remove, replace devices
  - Common management of storage hardware resources

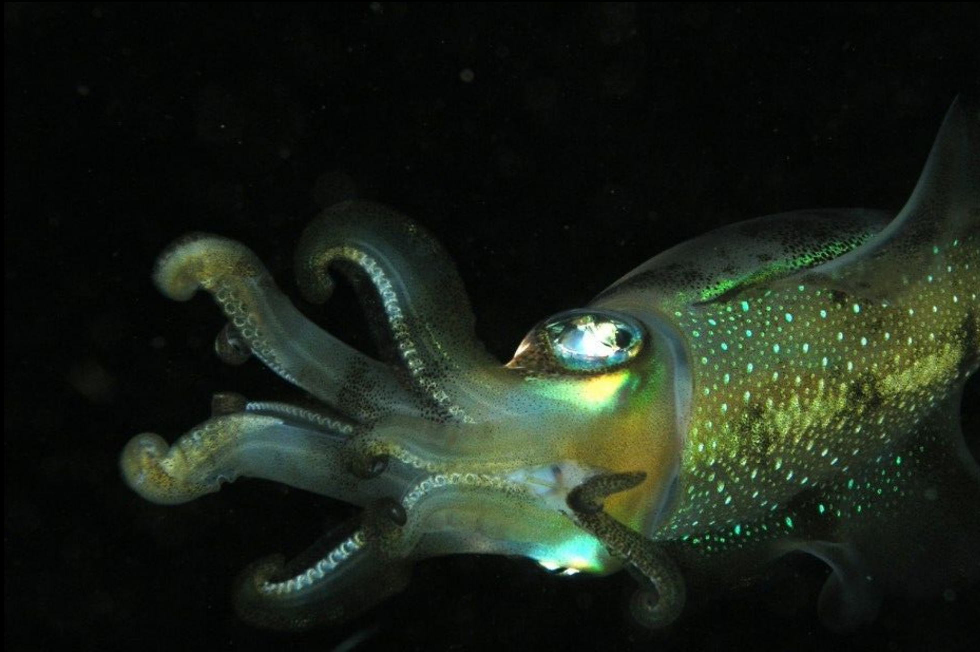
RADOS VIRTUALIZES STORAGE



PLATFORM FOR HIGH-LEVEL SERVICES

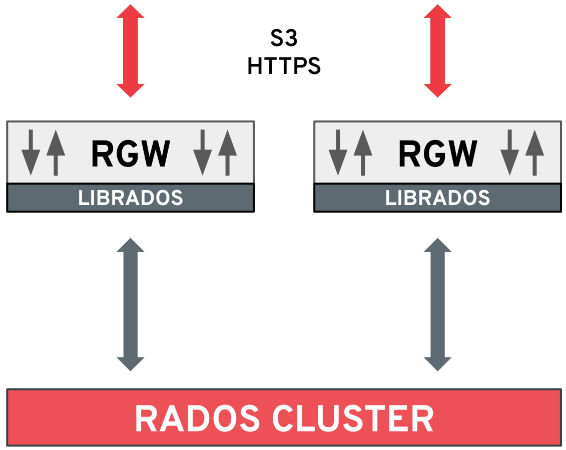


RGW: OBJECT STORAGE

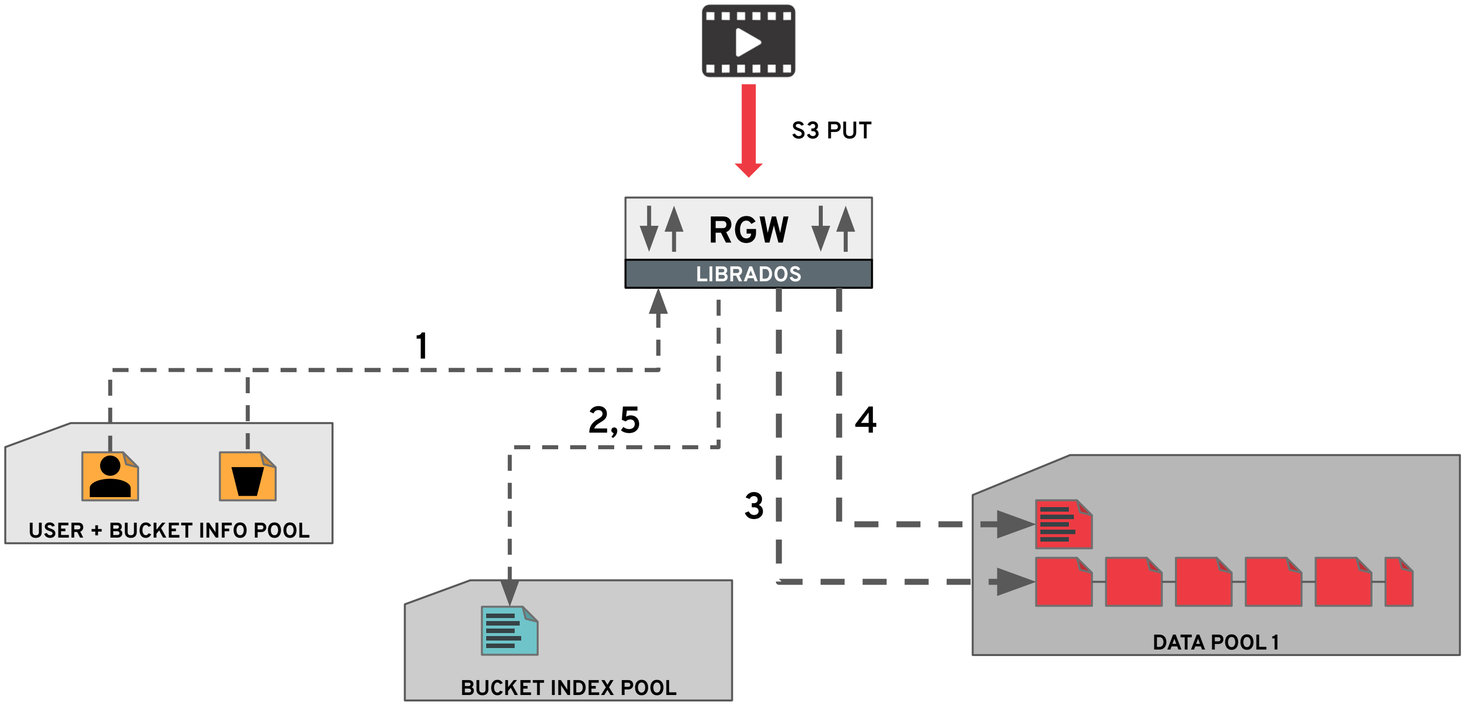


RGW: RADOS GATEWAY

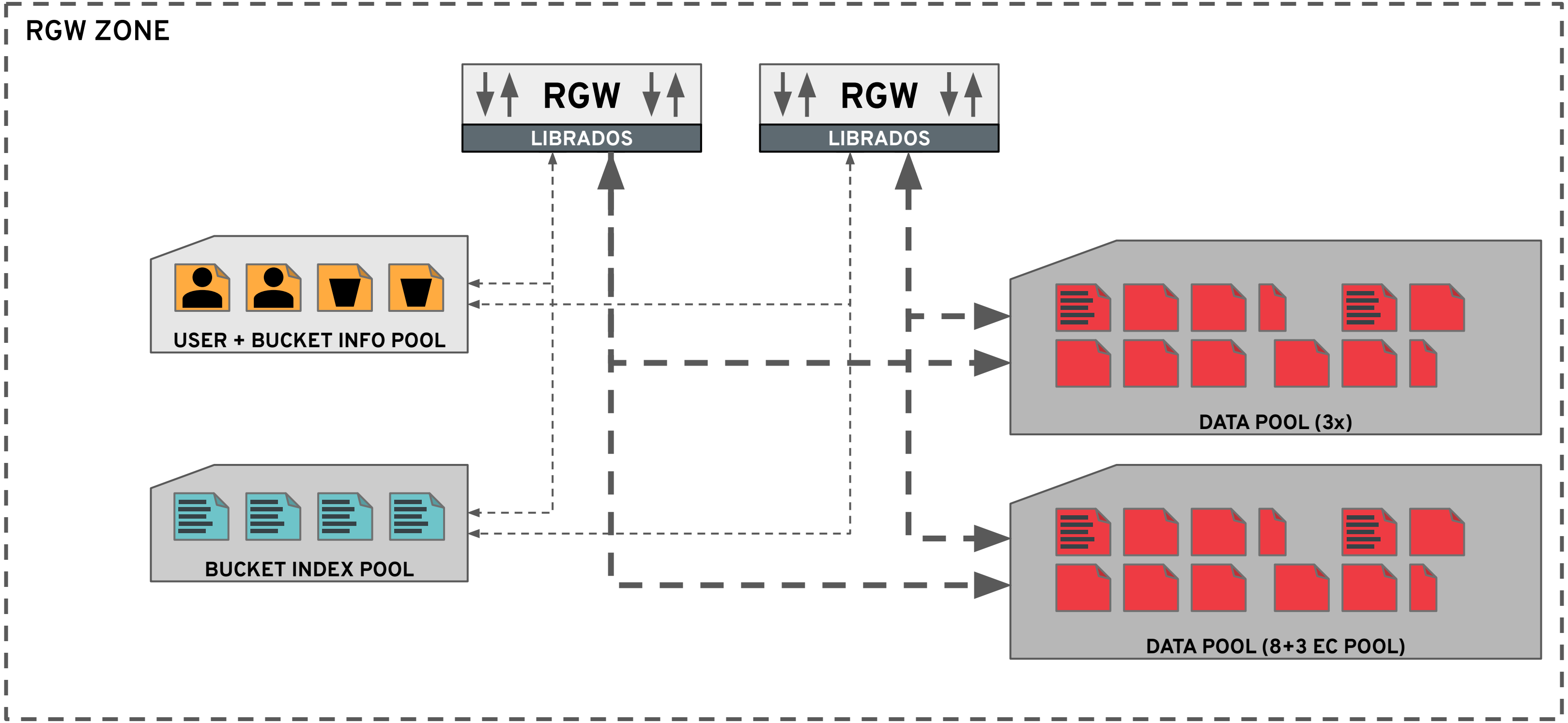
- S3 and Swift-compatible object storage
  - HTTPS/REST-based API
  - Often combined with load balancer to provide storage service to public internet
- Users, buckets, objects
  - Data and permissions model is based on a superset of S3 and Swift APIs
  - ACL-based permissions, enforced by RGW
- RGW objects not same as RADOS objects
  - S3 objects can be very big: GB to TB
  - RGW stripes data across RADOS objects



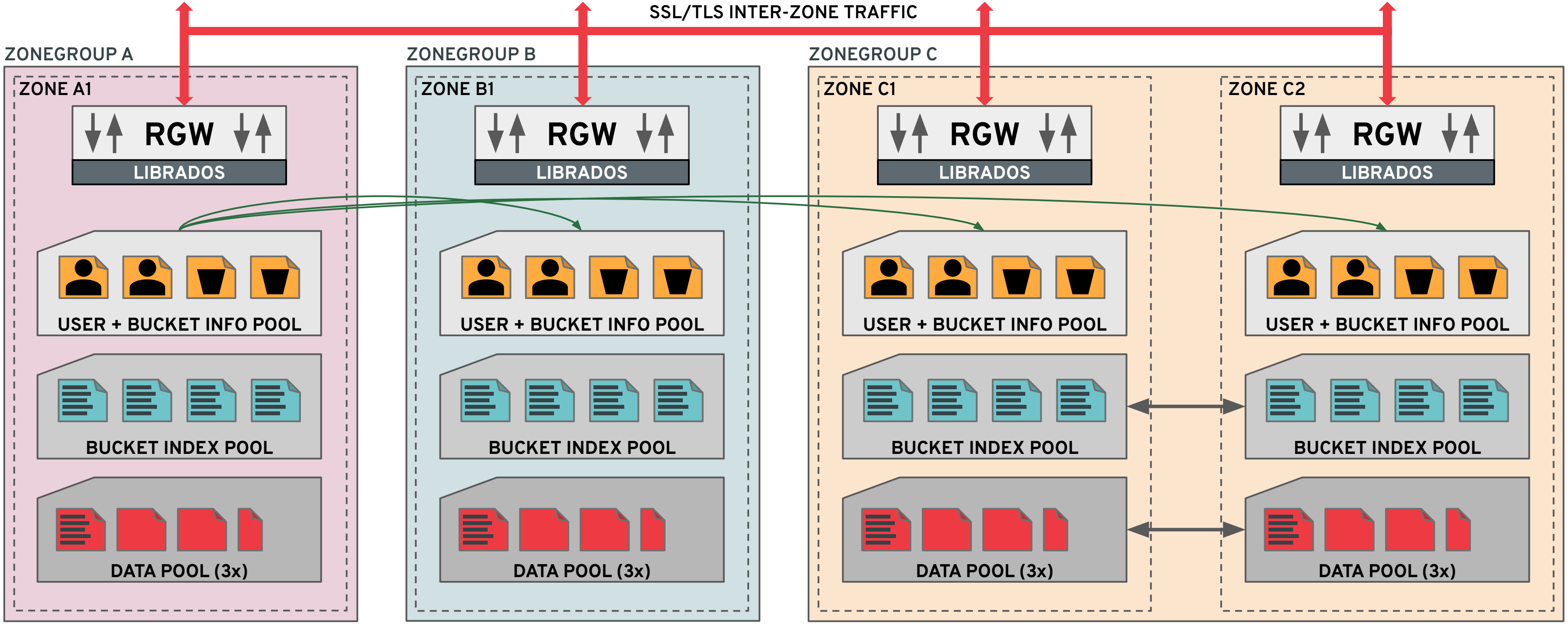
RGW STORES ITS DATA IN RADOS



RGW ZONE: POOLS + RGW DAEMONS



RGW FEDERATION AND GEO-REP



- Zones may be different clusters and/or sites
- Global view of users and buckets
- Each bucket placed in a ZoneGroup
- Data replicated between all Zones in a ZoneGroup

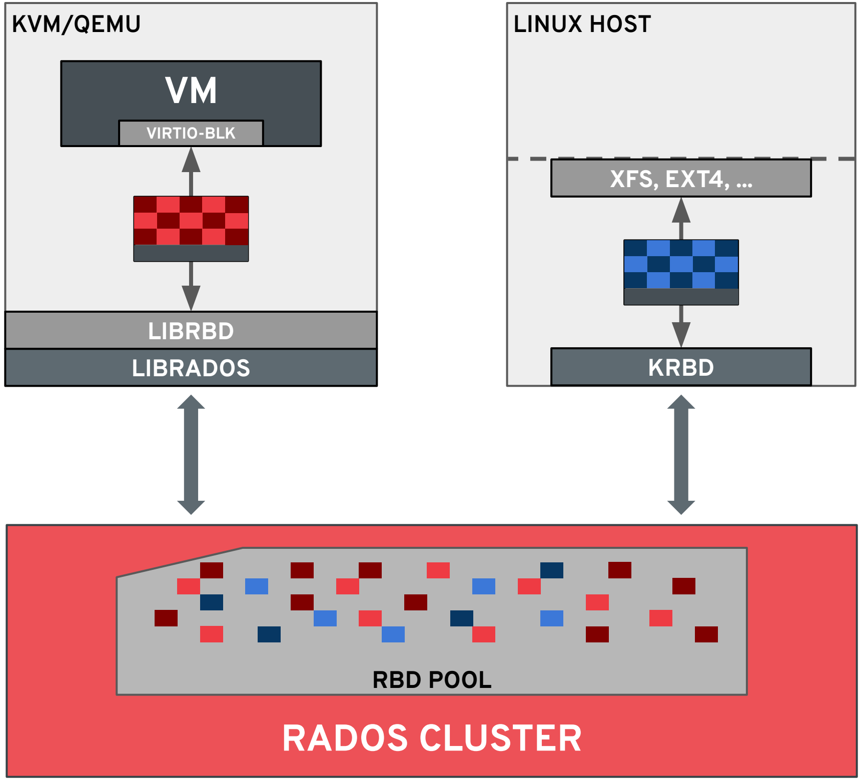
OTHER RGW FEATURES

- Very strong S3 API compatibility
  - https://github.com/ceph/s3-tests functional test suite
- STS: Security Token Service
  - Framework for interoperating with other authentication/authorization systems
- Encryption (various flavors of API)
- Compression
- CORS and static website hosting
- Metadata search with ElasticSearch
- Pub/sub event stream
  - Integration with knative serverless
  - Kafka
- Multiple storage classes
  - Map classes to RADOS pools
  - Choose storage for individual objects or set a bucket policy
- Lifecycle management
  - Bucket policy to automatically move objects between storage tiers and/or expire
  - Time-based
- Archive zone
  - Archive and preserve full storage history

RBD: BLOCK STORAGE

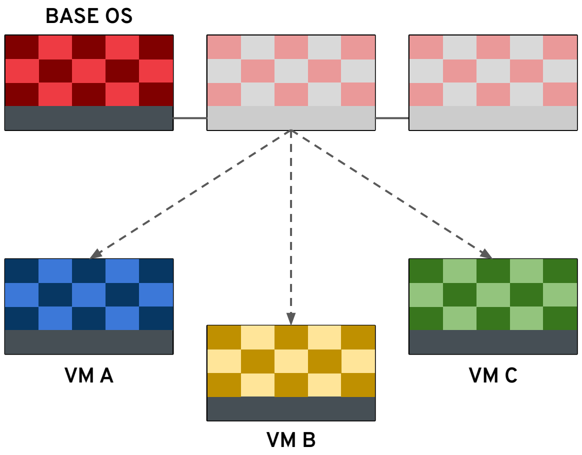
RBD: RADOS BLOCK DEVICE

- Virtual block device
  - Store disk images in RADOS
  - Stripe data across many objects in a pool
- Storage decoupled from host, hypervisor
  - Analogous to AWS's EBS
- Client implemented in KVM and Linux
- Integrated with
  - Libvirt
  - OpenStack (Cinder, Nova, Glace)
  - Kubernetes
  - Proxmox, CloudStack, Nebula, …

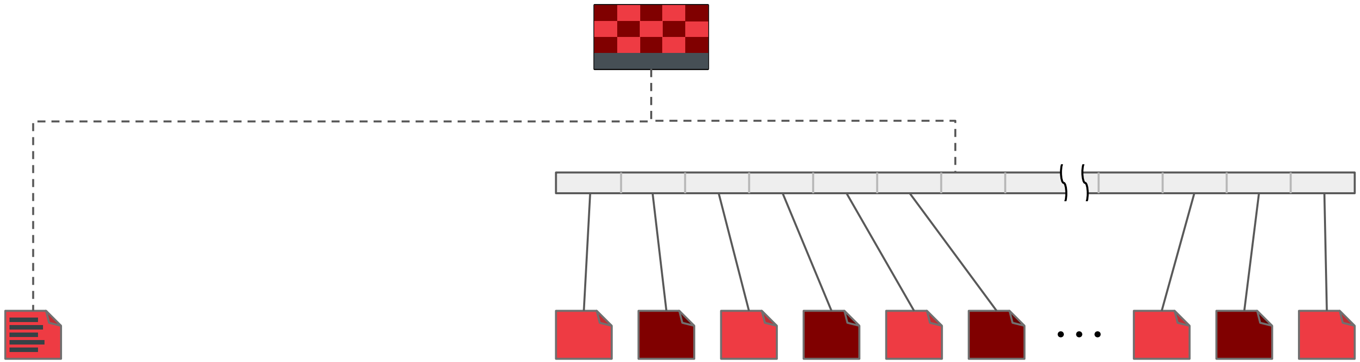


SNAPSHOTS AND CLONES

- Snapshots
  - Read-only
  - Associated with individual RBD image
  - Point-in-time consistency
- Clones
  - New, first-class image
  - Writeable overlay over an existing snapshot
  - Can be snapshotted, resized, renamed, etc.
- Efficient
  - O(1) creation time
  - Leverage copy-on-write support in RADOS
  - Only consume space when data is changed



RBD: DATA LAYOUT



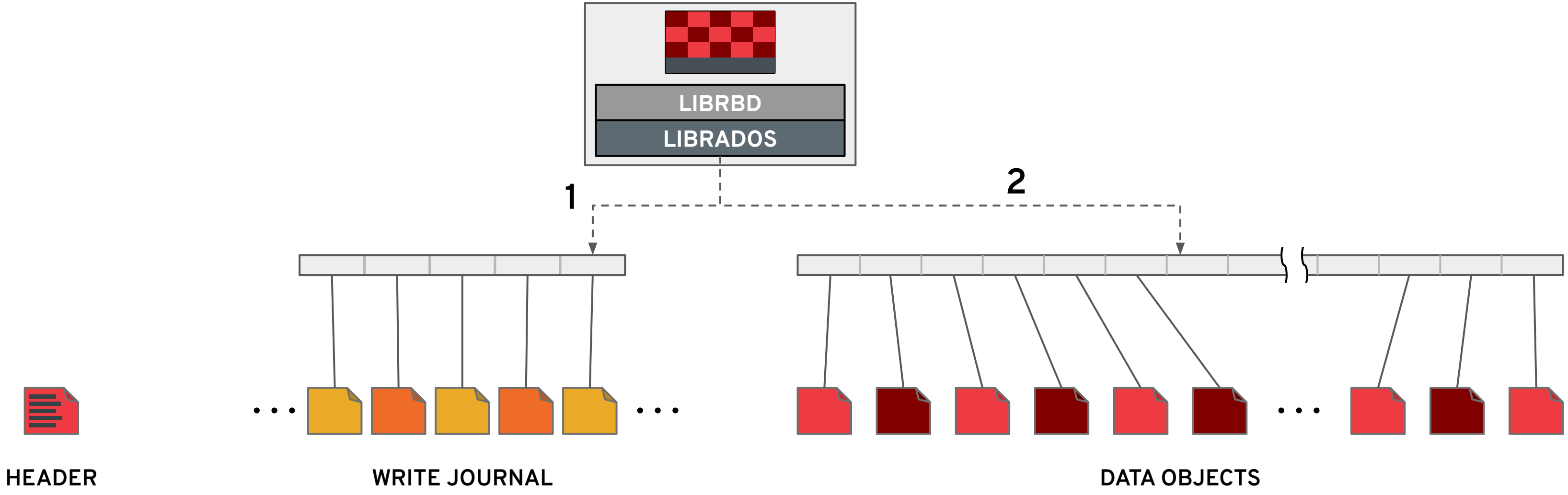
HEADER

- Image name
- Image size
- Striping parameters
- Snapshot metadata (names etc.)
- Options
- Lock owner
- ...

DATA OBJECTS

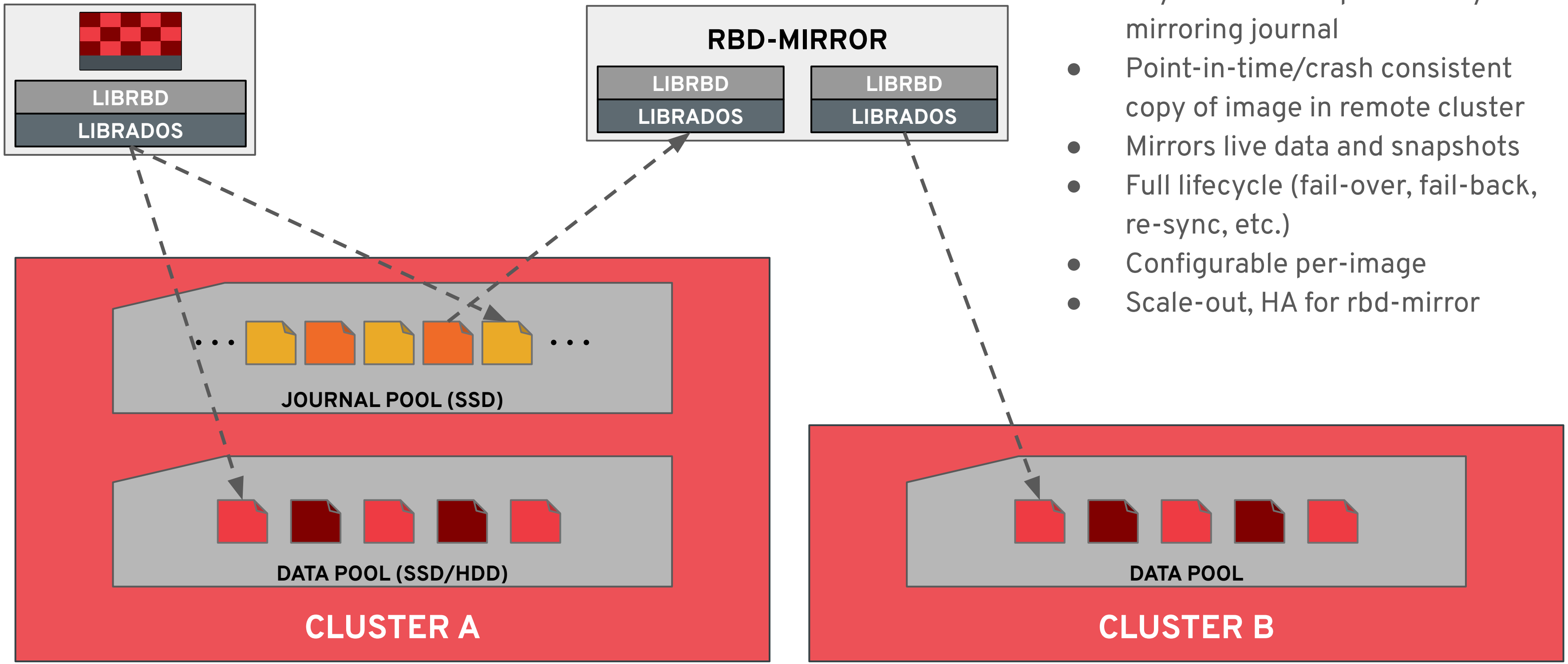
- Chunk of block device content
- 4 MB by default, but striping is configurable
- Sparse: objects only created if/when data is written
- Replicated or erasure coded, depending on the pool

RBD: JOURNALING MODE



- Recent writes
- Metadata changes

RBD MIRRORING



- Asynchronous replication by mirroring journal
- Point-in-time/crash consistent copy of image in remote cluster
- Mirrors live data and snapshots
- Full lifecycle (fail-over, fail-back, re-sync, etc.)
- Configurable per-image
- Scale-out, HA for rbd-mirror

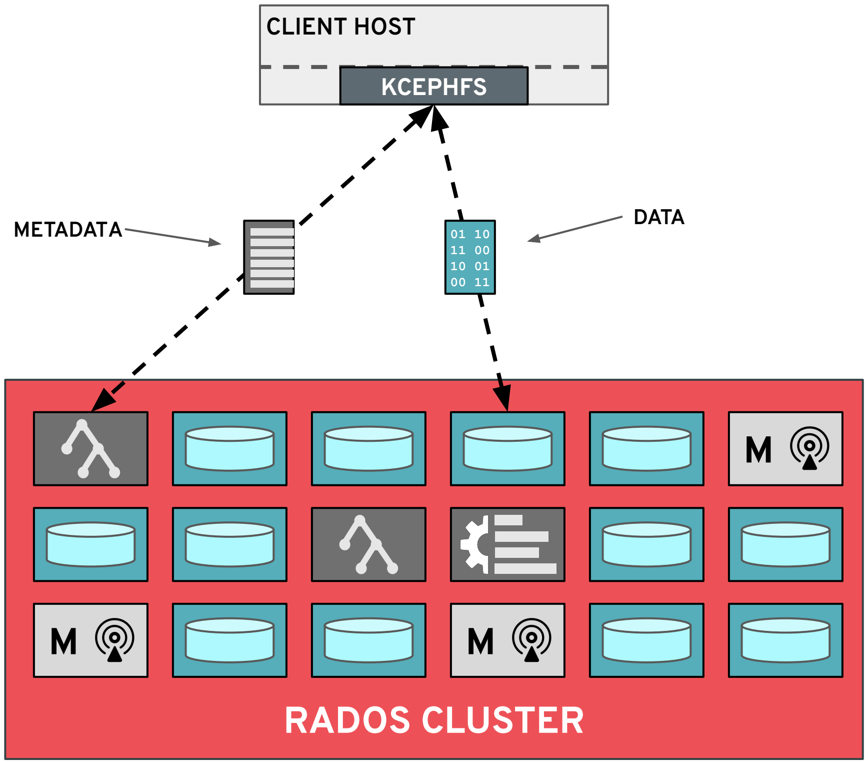
OTHER RBD FEATURES

- ‘rbd top’
  - Real-time view of IO activity
- Quotas
  - Enforced at provisioning time
- Namespace isolation
  - Restrict access to a private namespace of RBD images
- Import and export
  - Full image import/export
  - Incremental diff (between snapshots)
- Trash
  - Keep deleted images around for a bit before purging
- Linux kernel client
  - ‘rbd map myimage’ → /dev/rbd*
- NBD
  - ‘rbd map -t nbd myimage’ → /dev/nbd*
  - Run latest userspace library
- iSCSI gateway
  - LIO stack + userspace tools to manage gateway configuration
- librbd
  - Dynamically link with application

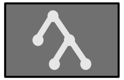
CEPHFS: FILE STORAGE

CEPHFS: CEPH FILE SYSTEM

- Distributed network file system
  - Files, directories, rename, hard links, etc.
  - Concurrent shared access from many clients
- Strong consistency and coherent caching
  - Updates from one node visible elsewhere, immediately
- Scale metadata and data independently
  - Storage capacity and IO throughput scale with the number of OSDs
  - Namespace (e.g., number of files) scales with the number of MDS daemons



CEPH-MDS: METADATA SERVER

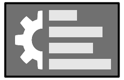


ceph-mds

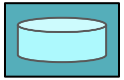
MDS (Metadata Server)

- Manage file system namespace
- Store file system metadata in RADOS objects
  - File and directory metadata (names, inodes)
- Coordinate file access between clients
- Manage client cache consistency, locks, leases
- Not part of the data path
- 1s - 10s active, plus standbys

ceph-mon

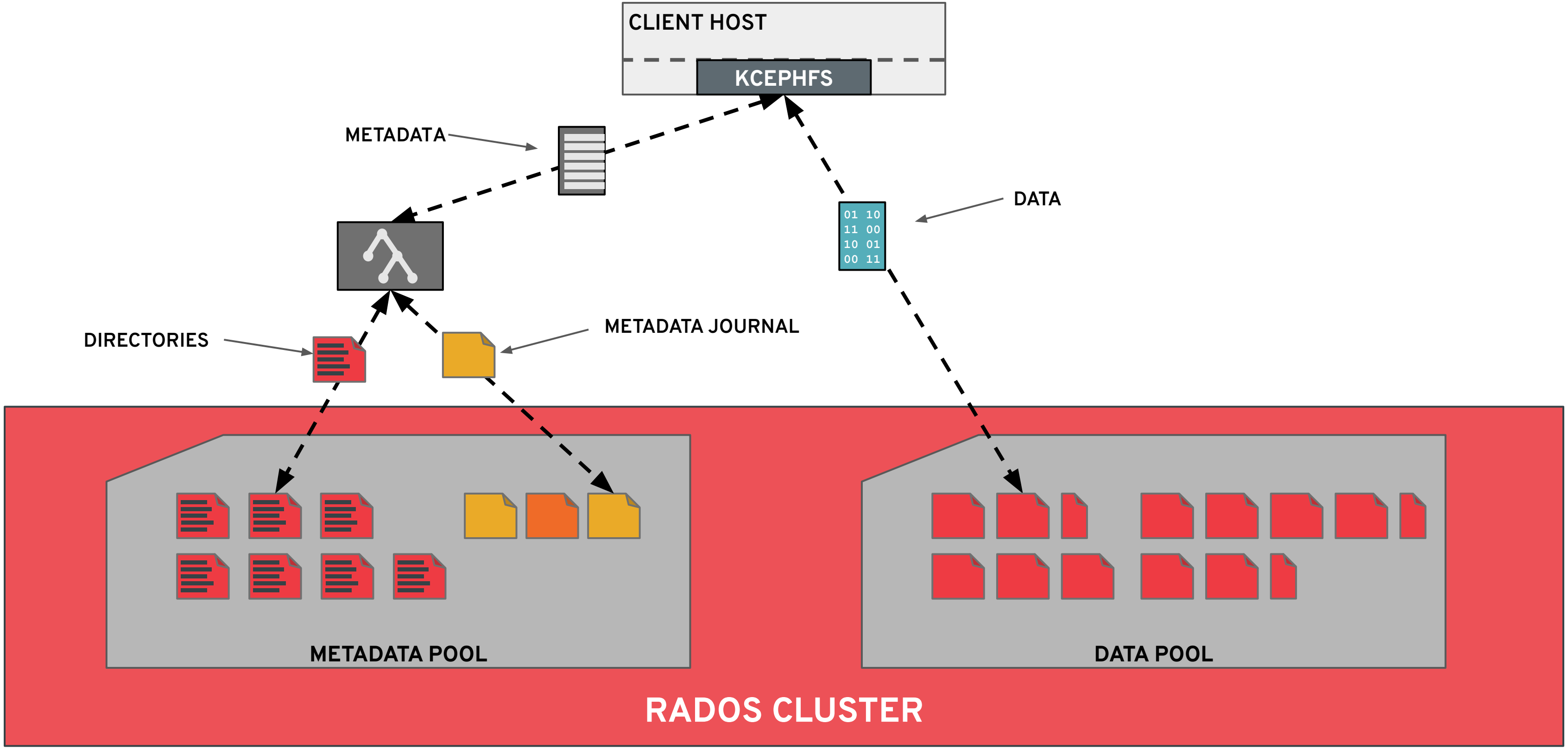


ceph-mgr



ceph-osd

METADATA IS STORED IN RADOS



CEPHFS SNAPSHOTS

- Snapshot any directory
  - Applies to all nested files and directories
  - Granular: avoid “volume” and “subvolume” restrictions in other file systems
- Point-in-time consistent
  - from perspective of POSIX API at *client*
  - *not* client/server boundary
- Easy user interface via file system
- Efficient
  - Fast creation/deletion
  - Snapshots only consume space when changes are made

```
$ cd any/cephfs/directory
$ ls
foo bar baz/
$ ls .snap
$ mkdir .snap/my_snapshot
$ ls .snap/
my_snapshot/
$ rm foo
$ ls
bar baz/
$ ls .snap/my_snapshot
foo bar baz/
$ rmdir .snap/my_snapshot
$ ls .snap
$
```

OTHER CEPHFS FEATURES

- Multiple file systems (volumes) per cluster
  - Separate ceph-mds daemons
- xattrs
- File locking (flock and fcntl)
- Quotas
  - On any directory
- Subdirectory mounts + access restrictions
- Multiple storage tiers
  - Directory subtree-based policy
  - Place files in different RADOS pools
  - Adjust file striping strategy
- Lazy IO
  - Optionally relax CephFS-enforced consistency on per-file basis for HPC applications

- Linux kernel client
  - e.g., mount -t ceph $monip:/ /ceph
- ceph-fuse
  - For use on non-Linux hosts (e.g., OS X) or when kernel is out of date
- NFS
  - CephFS plugin for nfs-ganesha FSAL
- CIFS
  - CephFS plugin for Samba VFS
- libcephfs
  - Dynamically link with your application

COMPLETE STORAGE PLATFORM

OBJECT

BLOCK

FILE

RGW

S3 and Swift object storage

RBD

Virtual block device

CEPHFS

Distributed network file system

LIBRADOS

Low-level storage API

RADOS

Reliable, elastic, distributed storage layer with replication and erasure coding

MANAGEMENT

INTEGRATED DASHBOARD

Monitoring

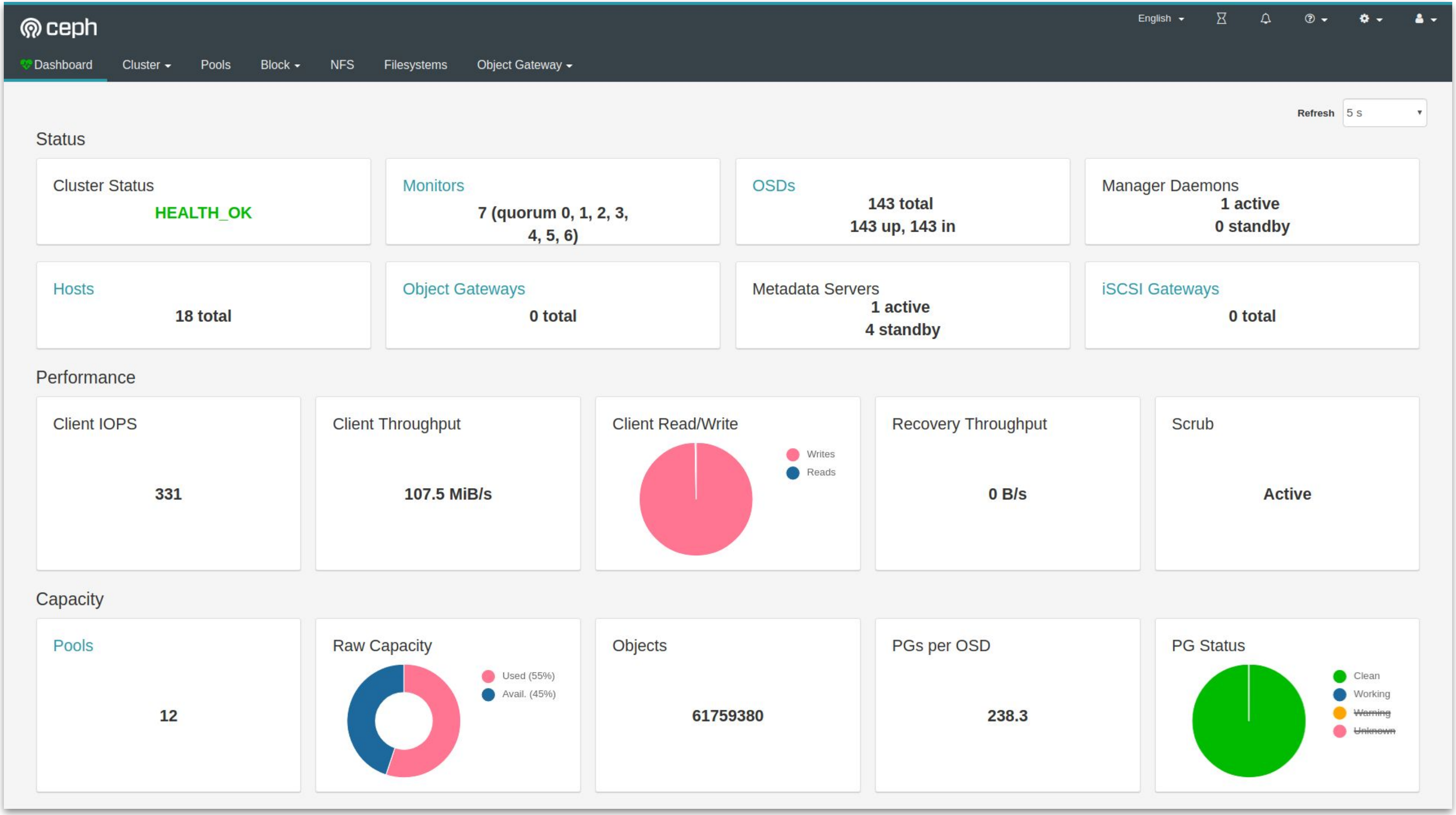
- Health
- IO and capacity utilization

Metrics

- Prometheus
- Grafana

Management

- Configuration
- Provisioning
- Day 2 tasks



A FEW OTHER MANAGEMENT FEATURES

- Internal health monitoring
  - Error and warning states
  - Alert IDs with documentation, mitigation steps, etc.
- Integrated configuration management
  - Self-documenting
  - History, rollback, etc.
- Device management
  - Map daemons to raw devices ($vendor_$model_$serial)
  - Scrape device health metrics (e.g. SMART)
  - Predict device life expectancy
  - Optionally preemptively evacuate failing devices
- Telemetry
  - Phone home anonymized metrics to Ceph developers
  - Cluster size, utilization, enabled features
  - Crash reports (version + stack trace)
  - Bugs are created based on crash reports
  - Performance metrics in-progress
  - Opt-in, obviously

INSTALLATION OPTIONS

- Cephadm
  - orchestration interface for installation and management using containers
  - https://docs.ceph.com/en/pacific/cephadm
- Rook
  - Run Ceph in Kubernetes
  - https://rook.io/
- Other methods
  - ceph-ansible
  - DeepSea
  - Puppet
  - https://docs.ceph.com/en/pacific/install/index.html#other-methods

COMMUNITY AND ECOSYSTEM

OPEN DEVELOPMENT COMMUNITY

- Ceph is open source software!
  - Mostly LGPL2.1/LGPL3
- We collaborate via
  - GitHub: https://github.com/ceph/ceph
  - https://tracker.ceph.com/
  - E-mail: dev@ceph.io
  - #ceph-devel on irc.oftc.net
- We meet a lot over video chat
  - See schedule at http://ceph.io/contribute
- We publish releases periodically
- We work with downstream distributions
  - Debian, SUSE, Ubuntu, Red Hat

WE INTEGRATE WITH CLOUD ECOSYSTEMS

openstack®

kubernetes

KVM

ROOK

CEPH EVENTS

Ceph Days

- One-day regional event
- ~10 per year
- 50-200 people
- Normally a single track of technical talks
- Mostly user-focused

http://ceph.io/cephdays

Cephalocon

- Two-day global event
- Once per year, in the spring
- 300-1000 people
- Multiple tracks
- Users, developers, vendors

http://ceph.io/cephalocon

Coming up https://ceph.io/en/community/events/2022/cephalocon-portland/

CEPH FOUNDATION

- Organization of industry members supporting the Ceph project and community
- 34 members
  - Vendors
  - Cloud companies
  - Major users
  - Academic and government institutions
- Event planning
- Upstream CI infrastructure
- Community hardware test lab
- Documentation

PREMIER MEMBERS

Bloomberg®

中国移动 China Mobile | 中移软件 CMSoft

DigitalOcean

intel®

OVH

redhat.

SAMSUNG

SOFTIRON

SUSE®
We adapt. You succeed.

Western Digital.

XSKY

ZTE

GENERAL MEMBERS

ASSOCIATE MEMBERS

BOSTON UNIVERSITY

CERN

CROSS
cross.ucsc.edu

FAS RC

grnet

MONASH University

UNIVERSITY OF MICHIGAN

NRF
National Research Foundation

SARAO

Science & Technology Facilities Council

SWITCH

FOR MORE INFORMATION

- http://ceph.io/
- Twitter: **@ceph**
- Docs: http://docs.ceph.com/
- Mailing lists: http://lists.ceph.io/
  - ceph-announce@ceph.io → announcements
  - ceph-users@ceph.io → user discussion
  - dev@ceph.io → developer discussion
- IRC: irc.oftc.net
  - #ceph, #ceph-devel
- GitHub: https://github.com/ceph/
- YouTube ‘Ceph’ channel