CERN

# Ceph Storage at CERN

**Pablo Llopis Sanmillan, Dan van der Ster**
CERN IT Department

23 March 2022, GSI Storage Seminar

# Outline

I. What is Ceph and how does it work?

II. Ceph Use-Cases at CERN

III. CephFS for HPC

# I. What is Ceph?

*Slides credit: Sage Weil*

# WHAT IS CEPH?

<u>The buzzwords</u>

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

<u>The substance</u>

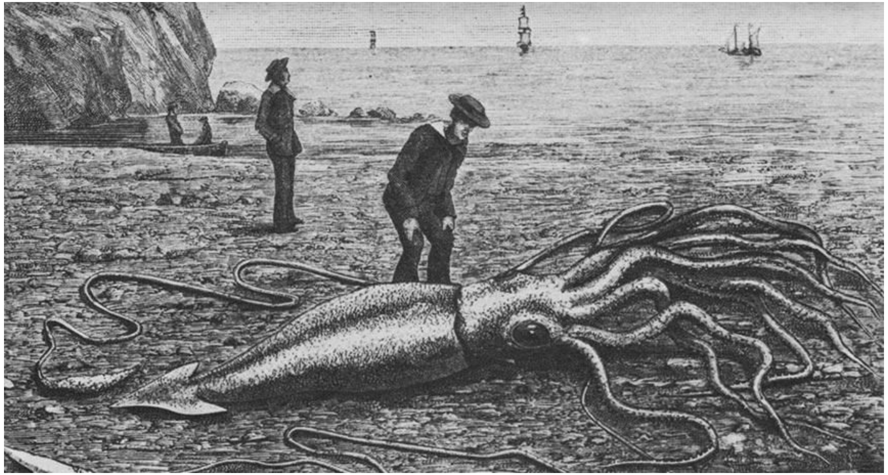
- Ceph is open source **software**
- Runs on commodity hardware
  - Commodity servers
  - IP networks
  - HDDs, SSDs, NVMe, NV-DIMMs, ...
- A single cluster can serve **object**, **block**, and **file** workloads

# CEPH IS RELIABLE

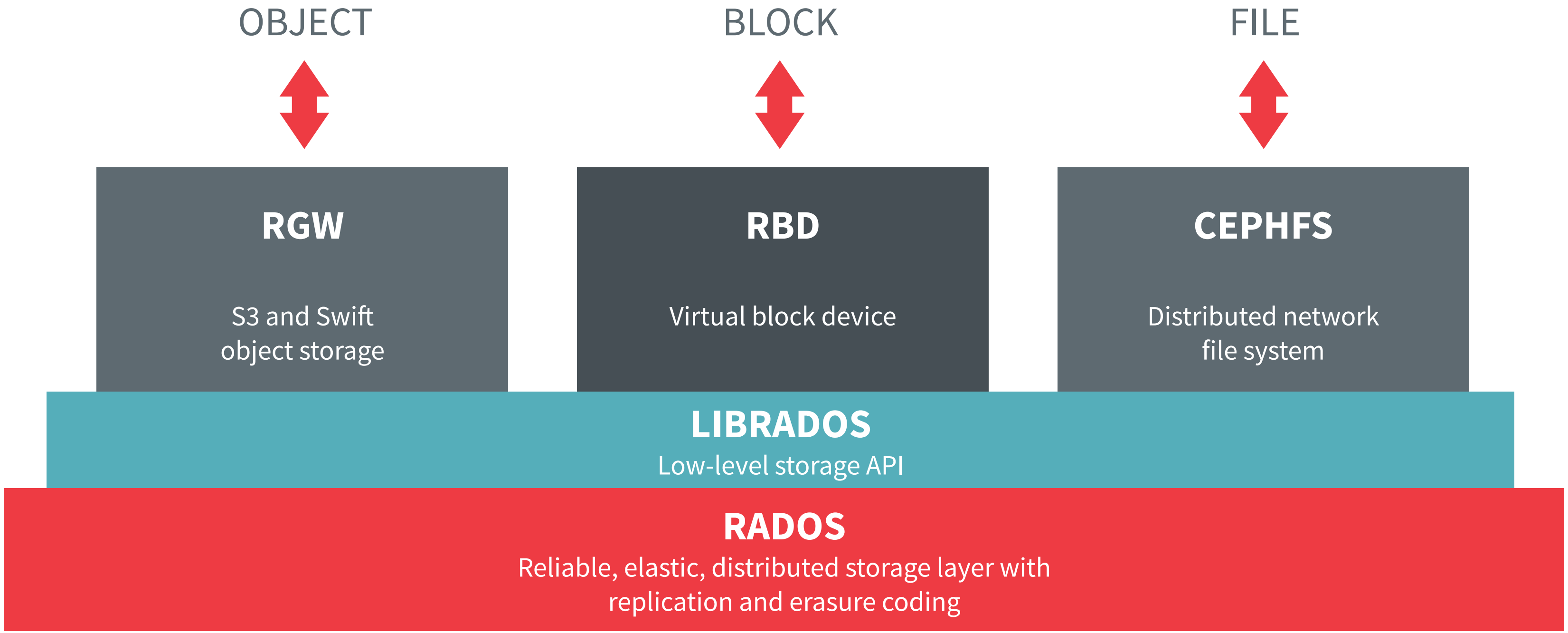
- **Reliable storage** service out of **unreliable components**
  - No single point of failure
  - Data durability via replication or erasure coding
  - No interruption of service from rolling upgrades, online expansion, etc.
- Favor consistency and correctness over performance

# 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



# THE CEPH FOUNDATION

Bloomberg
中国移动 China Mobile | 中移软件 CMSoft
XSKY
ZTE
DigitalOcean
intel
SOFTIRON
OVH
Western Digital
SUSE We adapt. You succeed.
redhat
SAMSUNG
arm
ubuntu Delivered by Canonical
cloudbase solutions
clyso cloud system solutions
BOSTON UNIVERSITY
CERN
CROSS cross.ucsc.edu
FASRC
croit
EasyStack open cloud computing
Intelligent Systems Services
linode
grnet
MONASH University
UNIVERSITY OF MICHIGAN
NRF SARAO
OSNEXUS
QCT
SRIE
VEXXHOST
Open Infrastructure FOUNDATION
Science & Technology Facilities Council
SWITCH

# RADOS

# RADOS

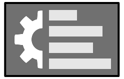
- **R**eliable **A**utonomic **D**istributed **O**bject **S**torage
  - Common storage layer underpinning object, block, and file services
- Provides low-level data object storage service
  - Reliable and highly available
  - Scalable (on day 1 and day 1000)
  - Manages all replication and/or erasure 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**

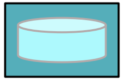
- Central authority for authentication, data placement, policy
- Coordination point for all other cluster components
- Protect critical cluster state with Paxos
- 3-7 per cluster



ceph-mgr

**Manager**

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



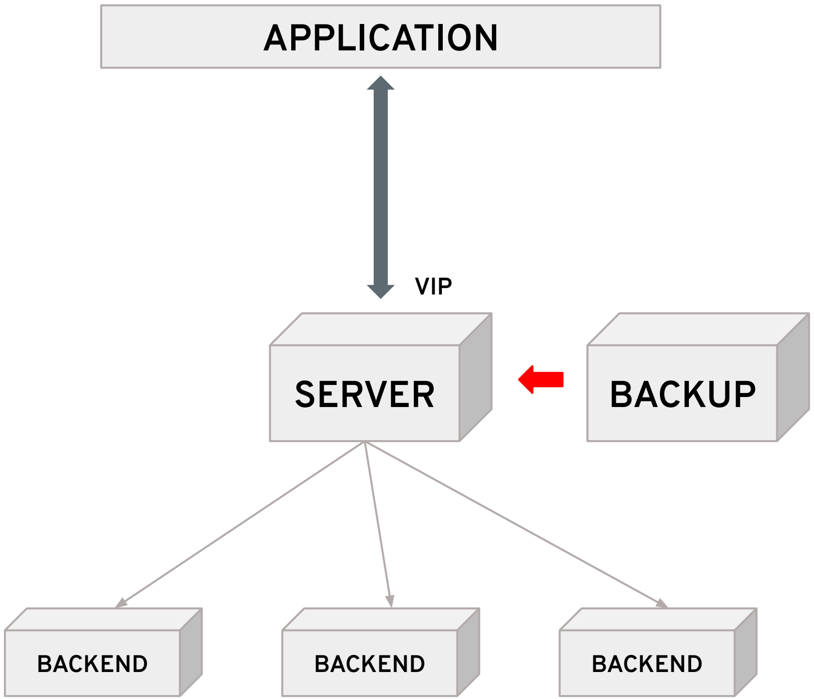
ceph-osd

**OSD (Object Storage Daemon)**

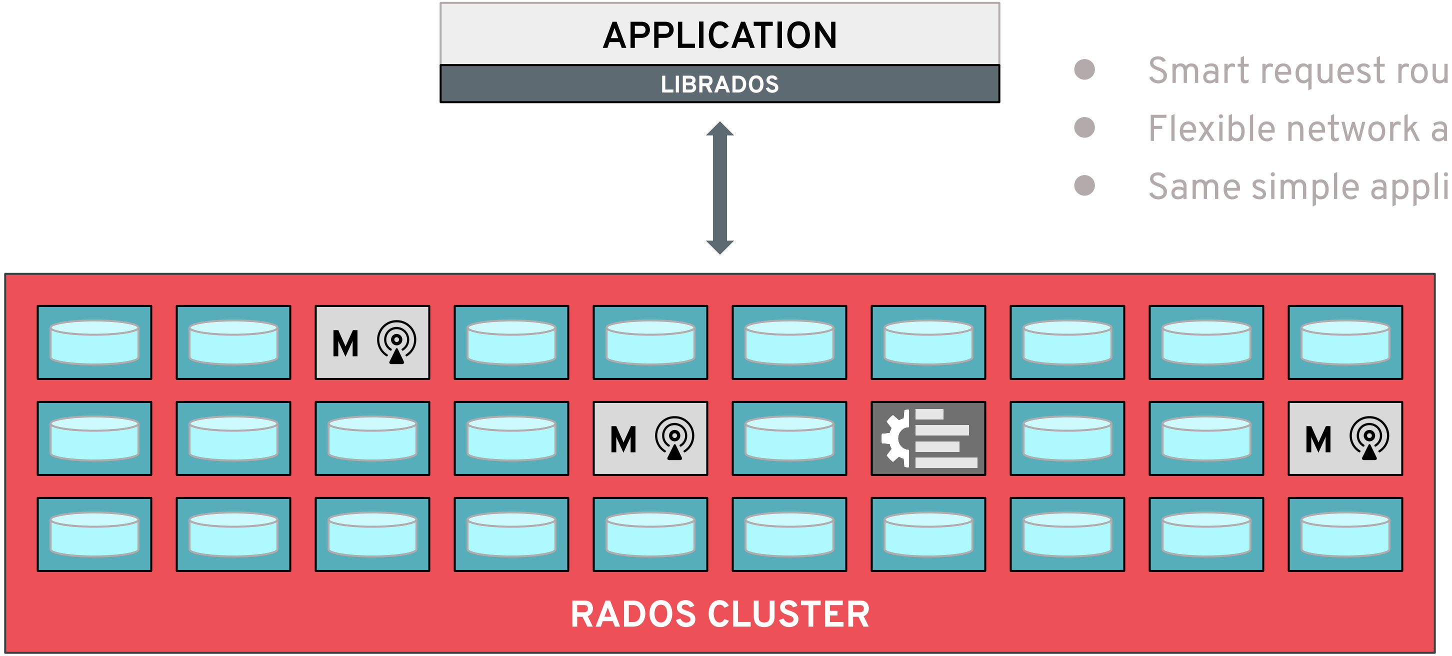
- Stores data on an HDD or SSD
- Services client IO requests
- Cooperatively peers, replicates, rebalances data
- 10s-1000s per cluster

# LEGACY CLIENT/SERVER ARCHITECTURE

- Virtual IPs
- Failover pairs
- Gateway nodes

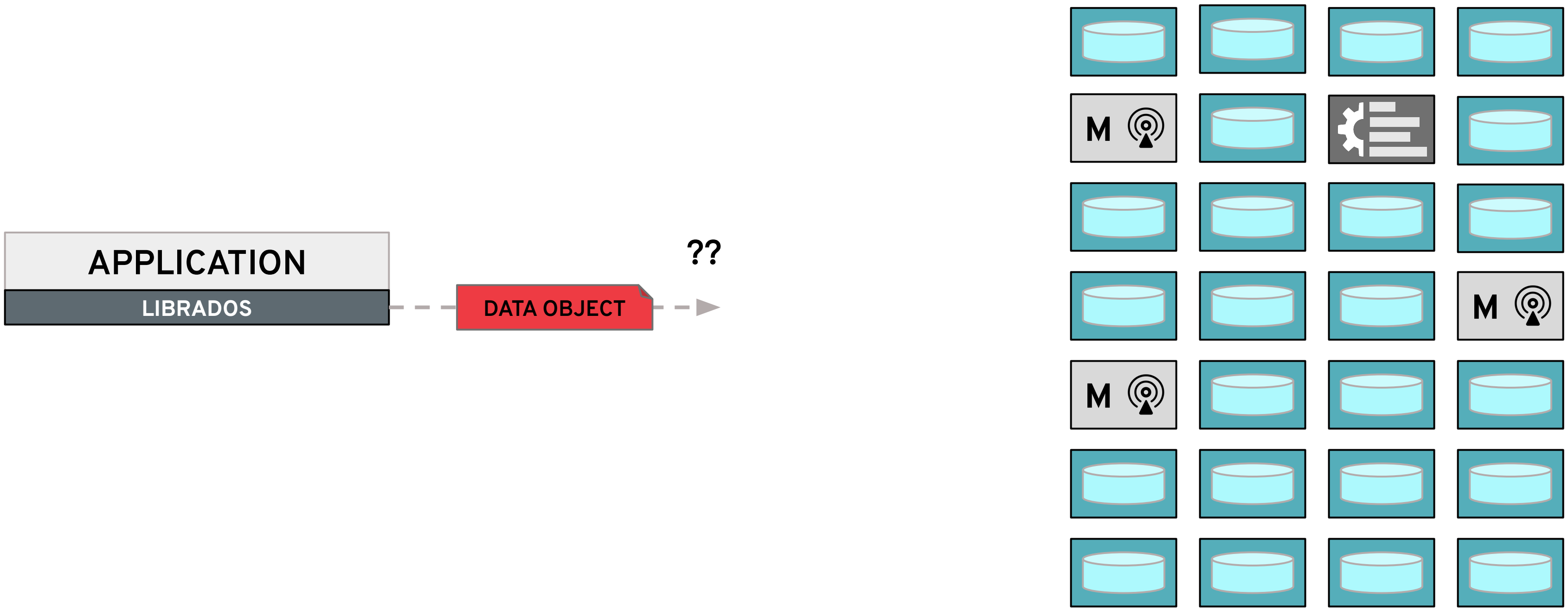


# CLIENT/CLUSTER ARCHITECTURE

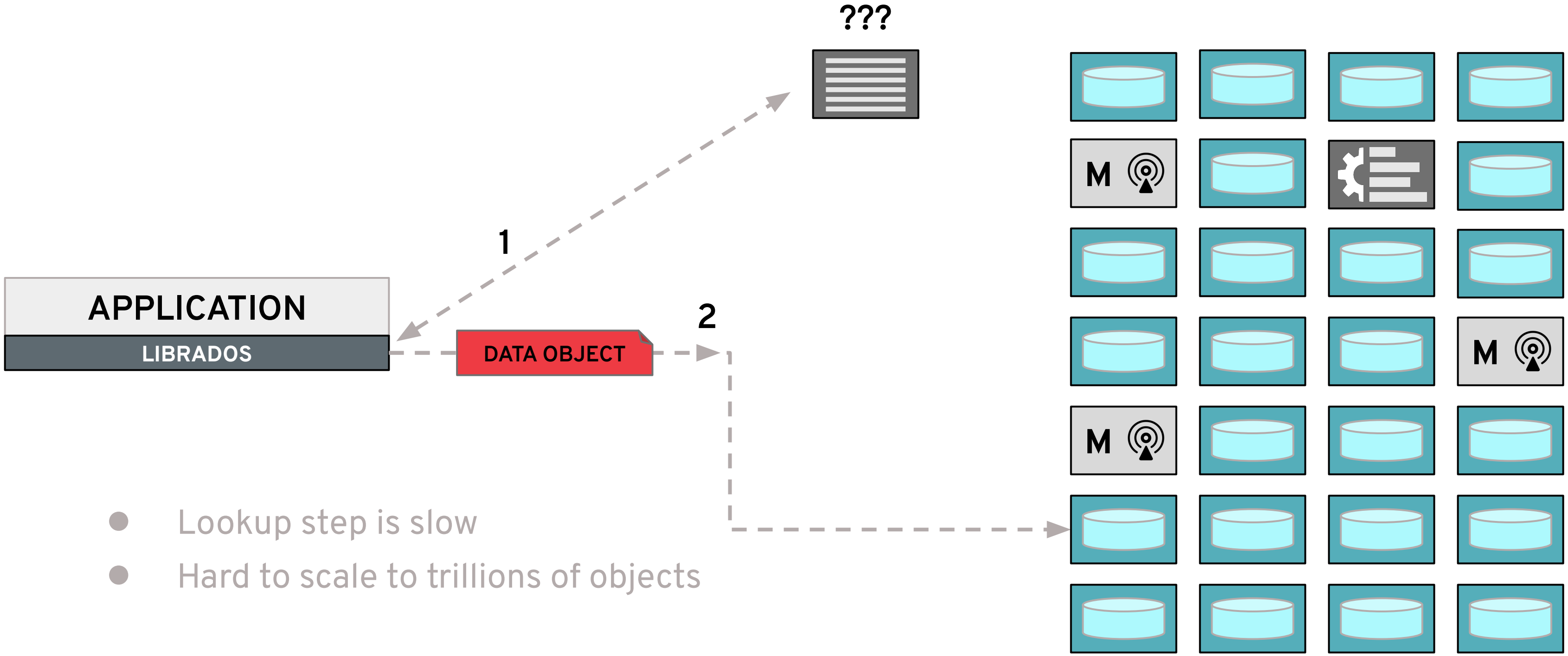



- Smart request routing
- Flexible network addressing
- Same simple application API

# DATA PLACEMENT

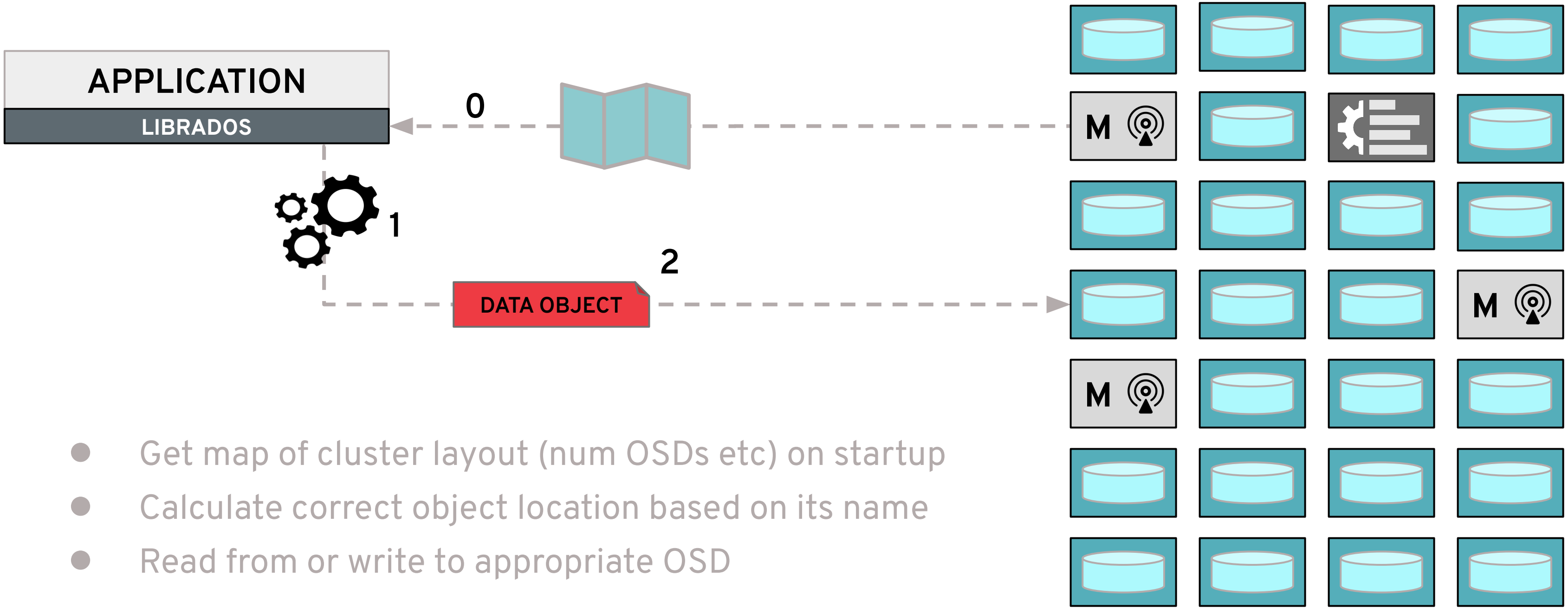


# LOOKUP VIA A METADATA SERVER?



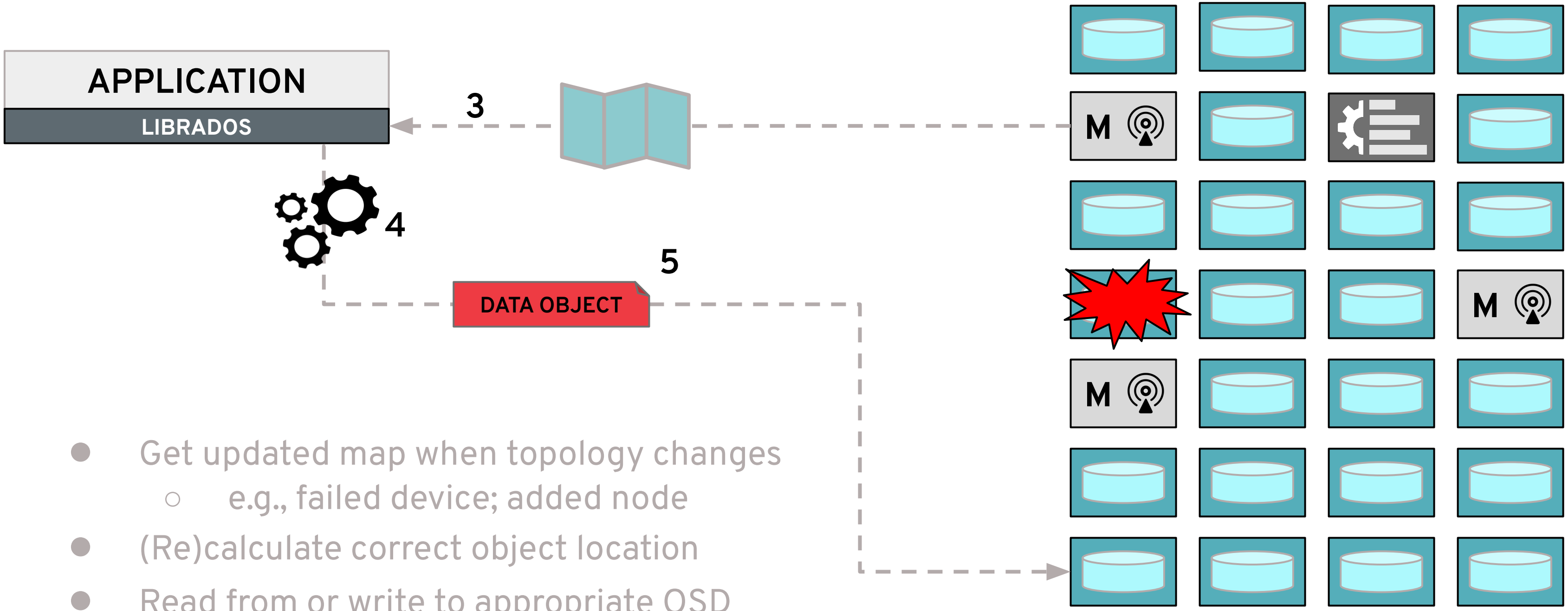

- Lookup step is slow
- Hard to scale to trillions of objects

# 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

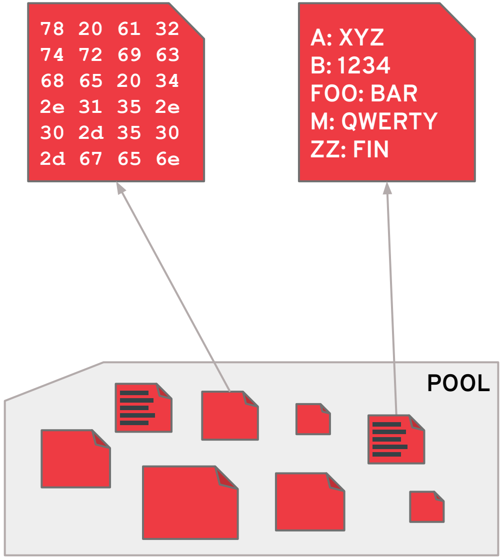
# MAP UPDATES WHEN TOPOLOGY CHANGES



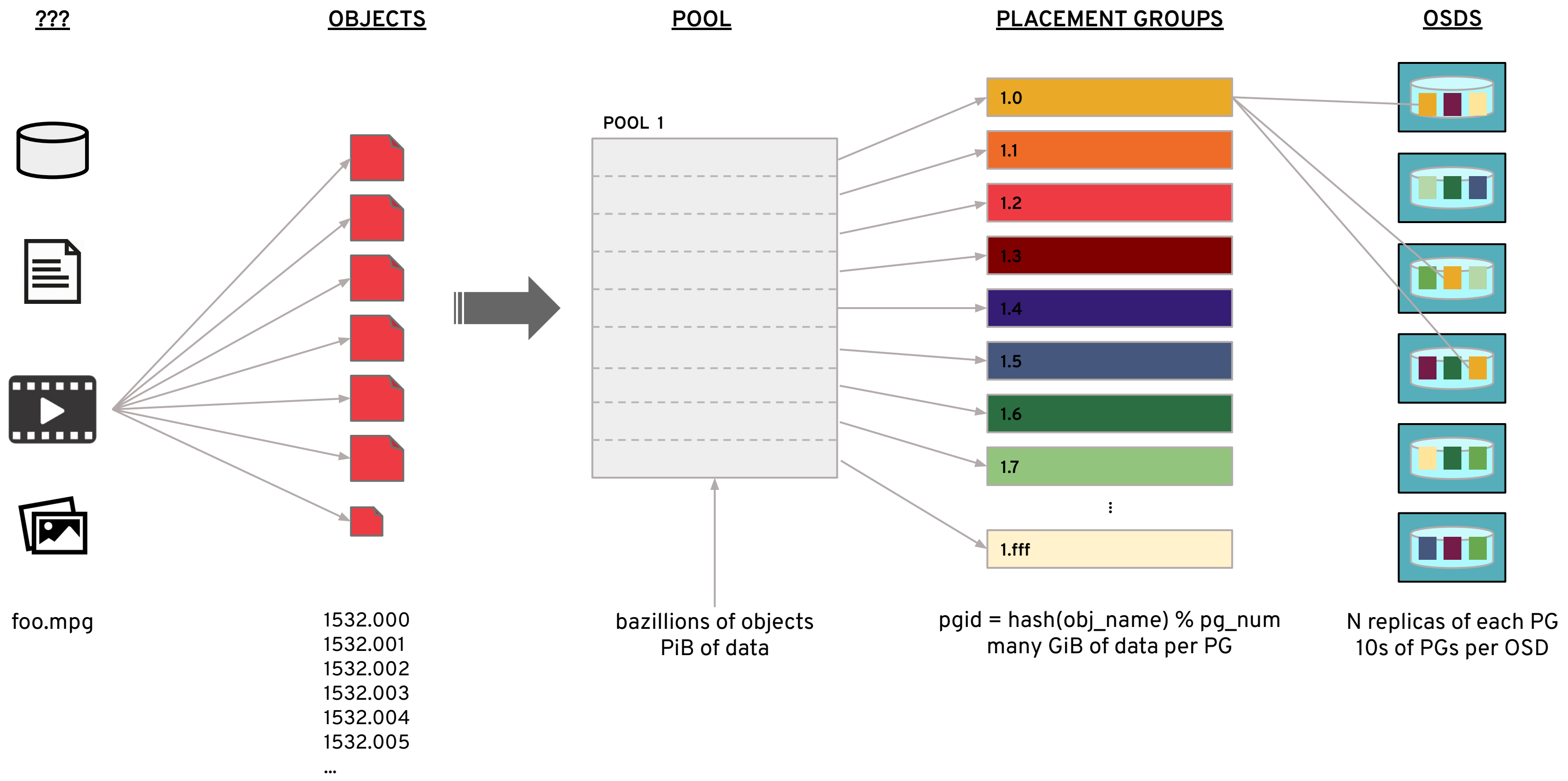

- Get updated map when topology changes
  - e.g., failed device; added node
- (Re)calculate correct object location
- Read from or write to appropriate OSD

# 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”

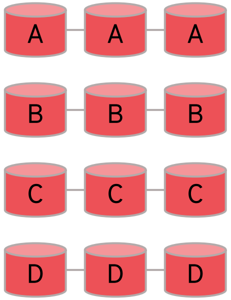


# ? → OBJECTS → POOLS → PGs → OSDs



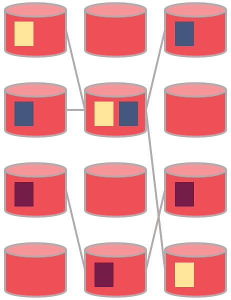
# WHY PLACEMENT GROUPS?

**REPLICATE DISKS**



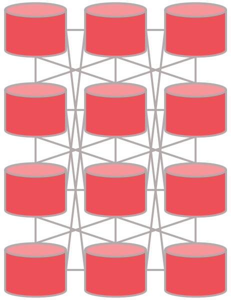

- Each device is mirrored
- Device sizes must match

**REPLICATE PGS**



- Each PG is mirrored
- PG placement is random

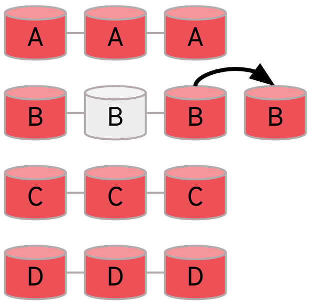
**REPLICATE OBJECTS**



- Each object is mirrored
- Object placement is random

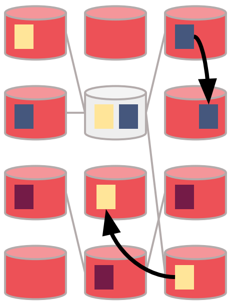
# WHY PLACEMENT GROUPS?

**REPLICATE DISKS**



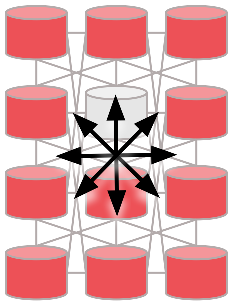

- Need an empty spare device to recover
- Recovery bottlenecked by single disk throughput

**REPLICATE PGS**



- New PG replicas placed on surviving devices
- Recovery proceeds in parallel, leverages many devices, and completes sooner

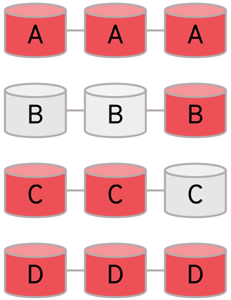
**REPLICATE OBJECTS**



- *Every* device participates in recovery

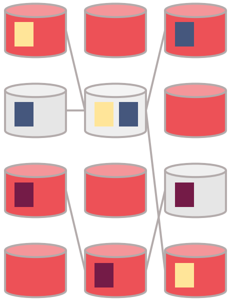
# WHY PLACEMENT GROUPS?

**REPLICATE DISKS**



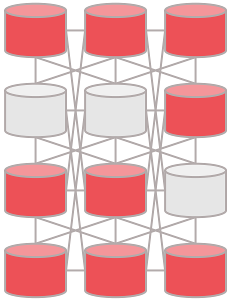

- Very few triple failures cause data loss (of an entire disk)

**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**

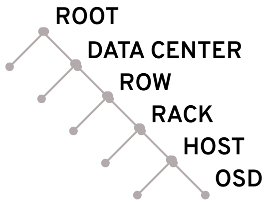
# KEEPING DATA SAFE

<u>"Declustered replica placement"</u>

- 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

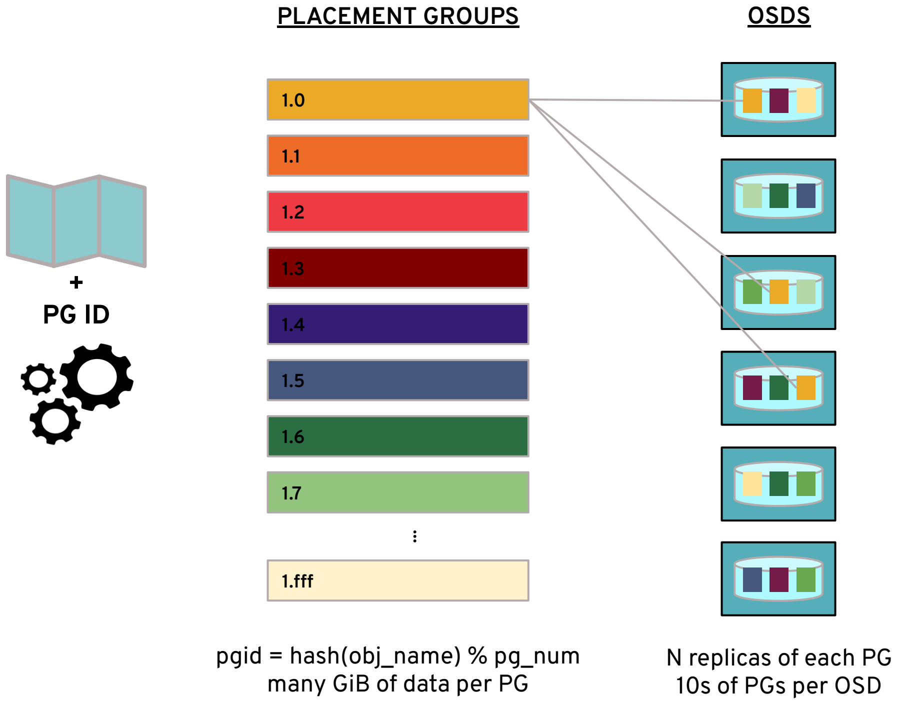
<u>Avoiding concurrent failures</u>

- 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



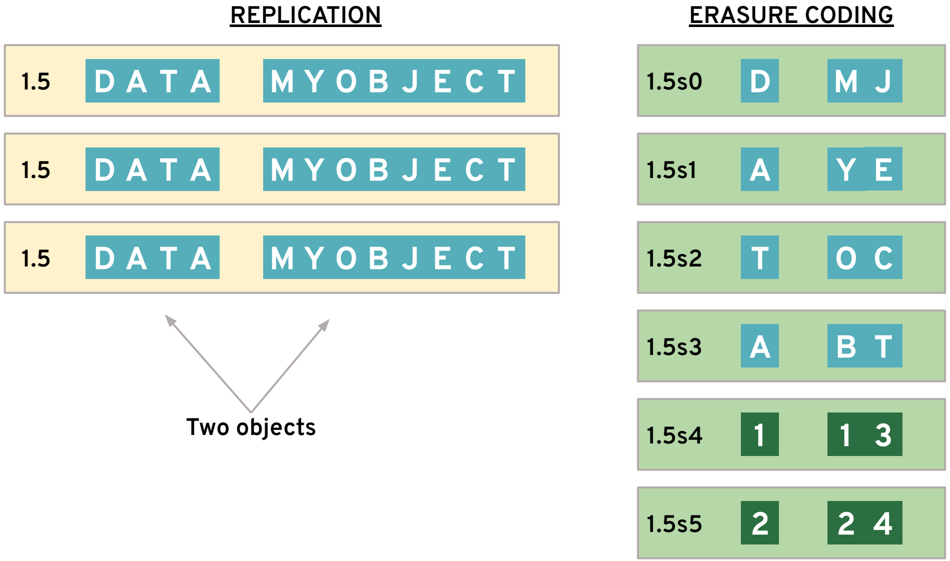
# PLACING PGs WITH CRUSH

- 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



# REPLICATION AND ERASURE CODING

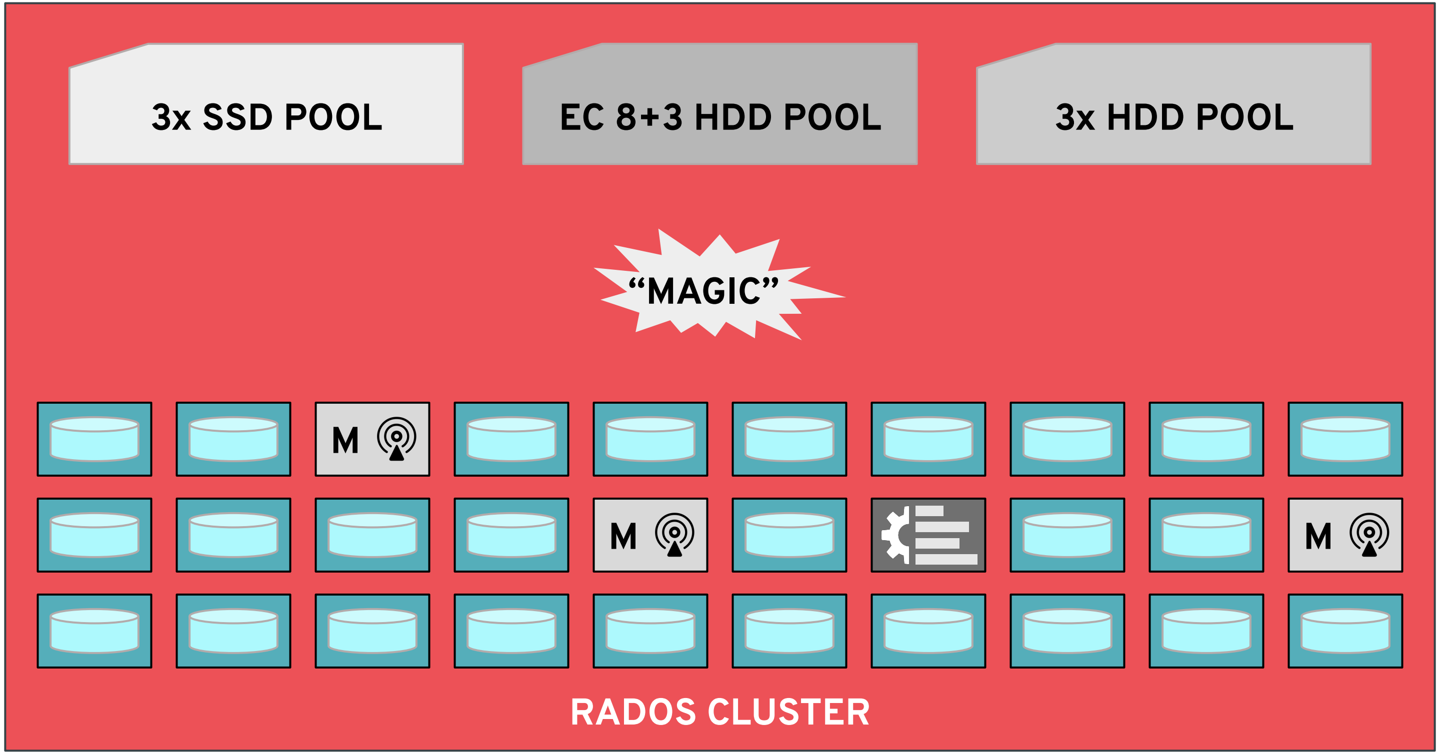
- Each RADOS pool must be durable
- Each PG must be durable
- Replication
  - Identical copies of each PG
  - Usually 3x (200% overhead)
  - Fast recovery--read any surviving copy
  - Can vary replication factor at any time
- Erasure coding
  - Each PG “shard” has different slice of data
  - Stripe object 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



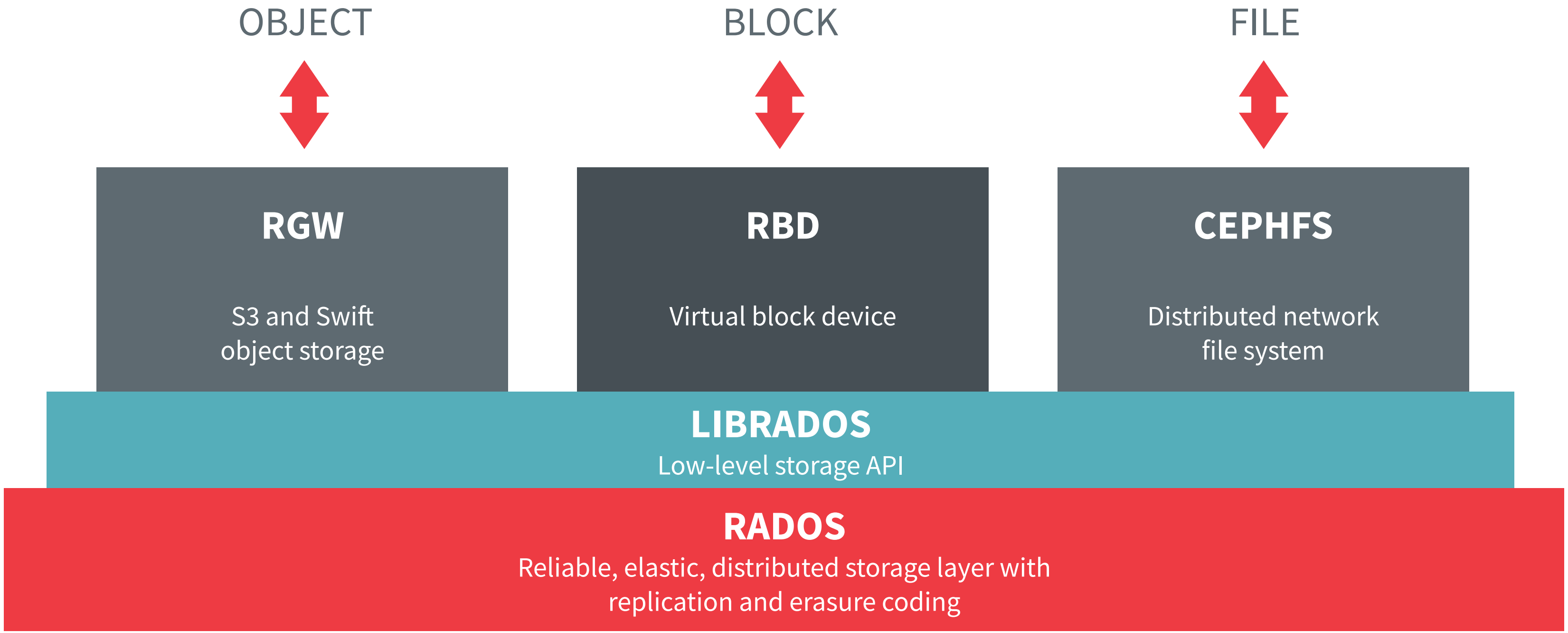
# SPECIALIZED POOLS

- 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

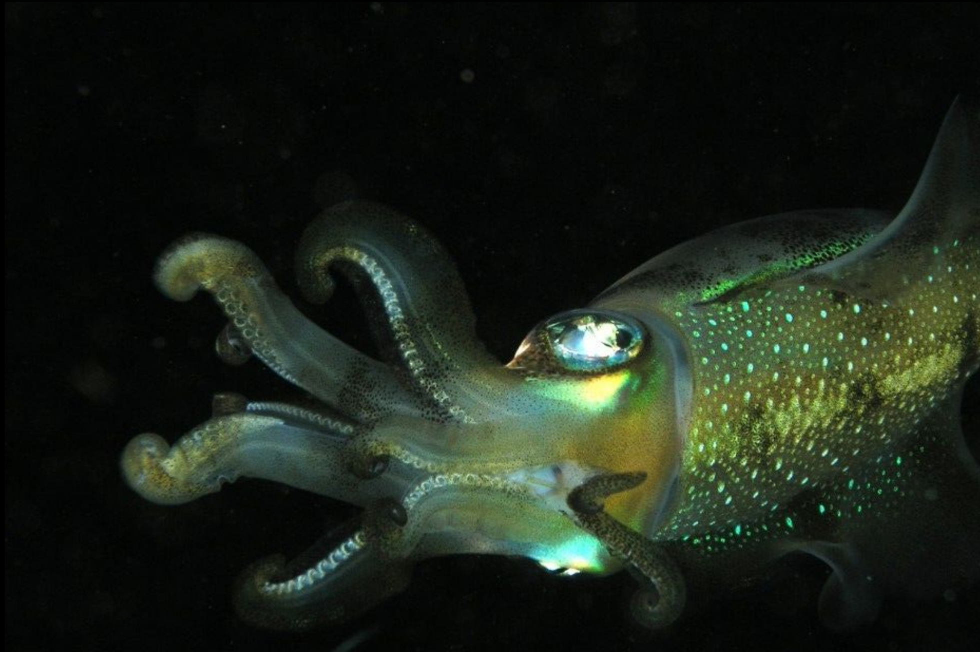
# RADOS VIRTUALIZES STORAGE



# PLATFORM FOR HIGH-LEVEL SERVICES

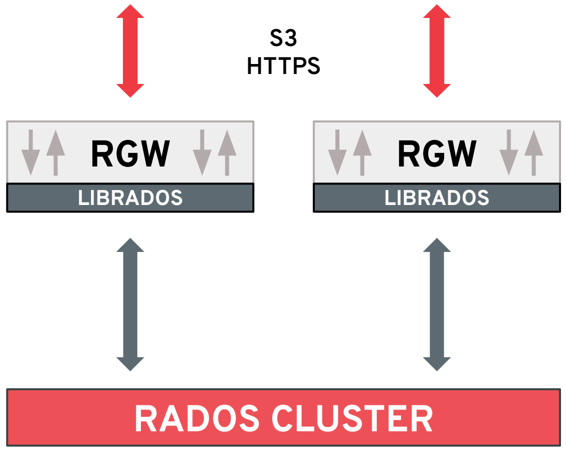


# RGW: OBJECT STORAGE

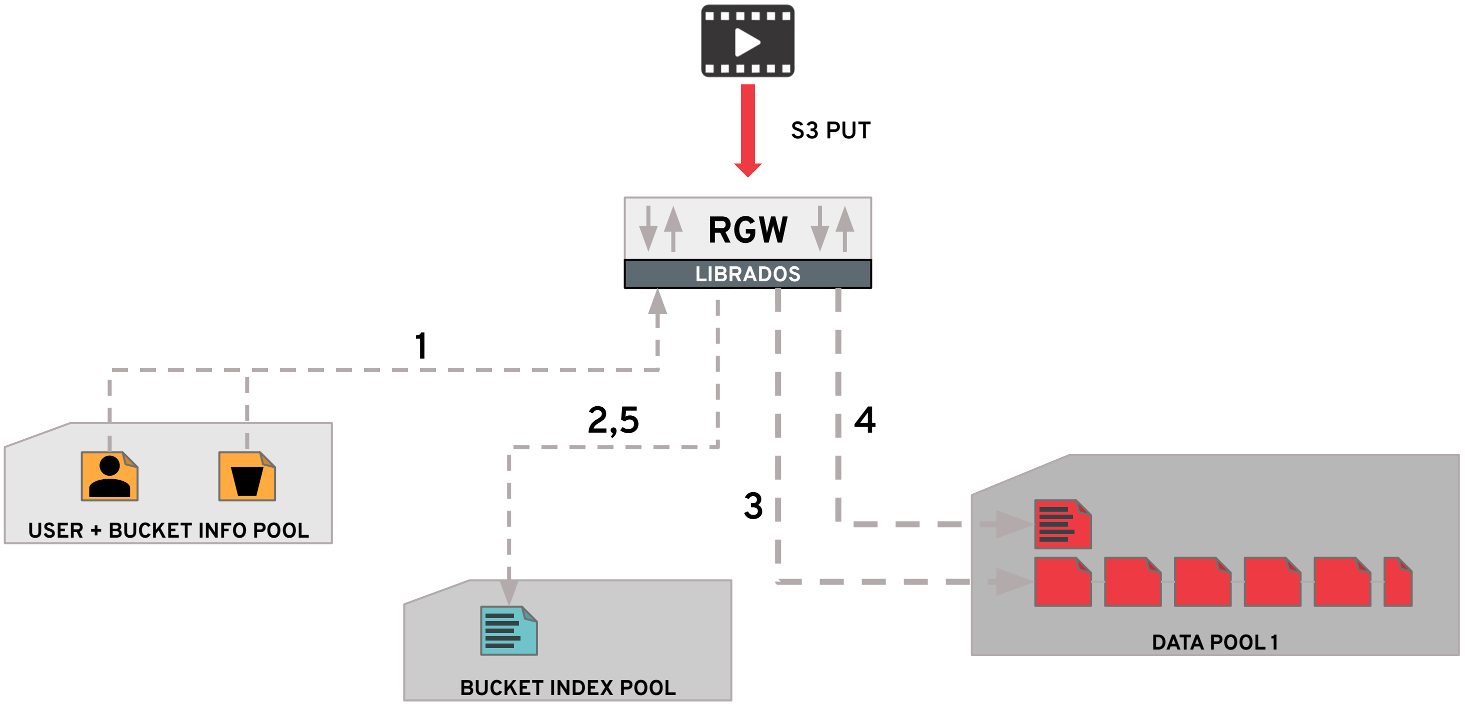


# RGW: RADOS GATEWAY

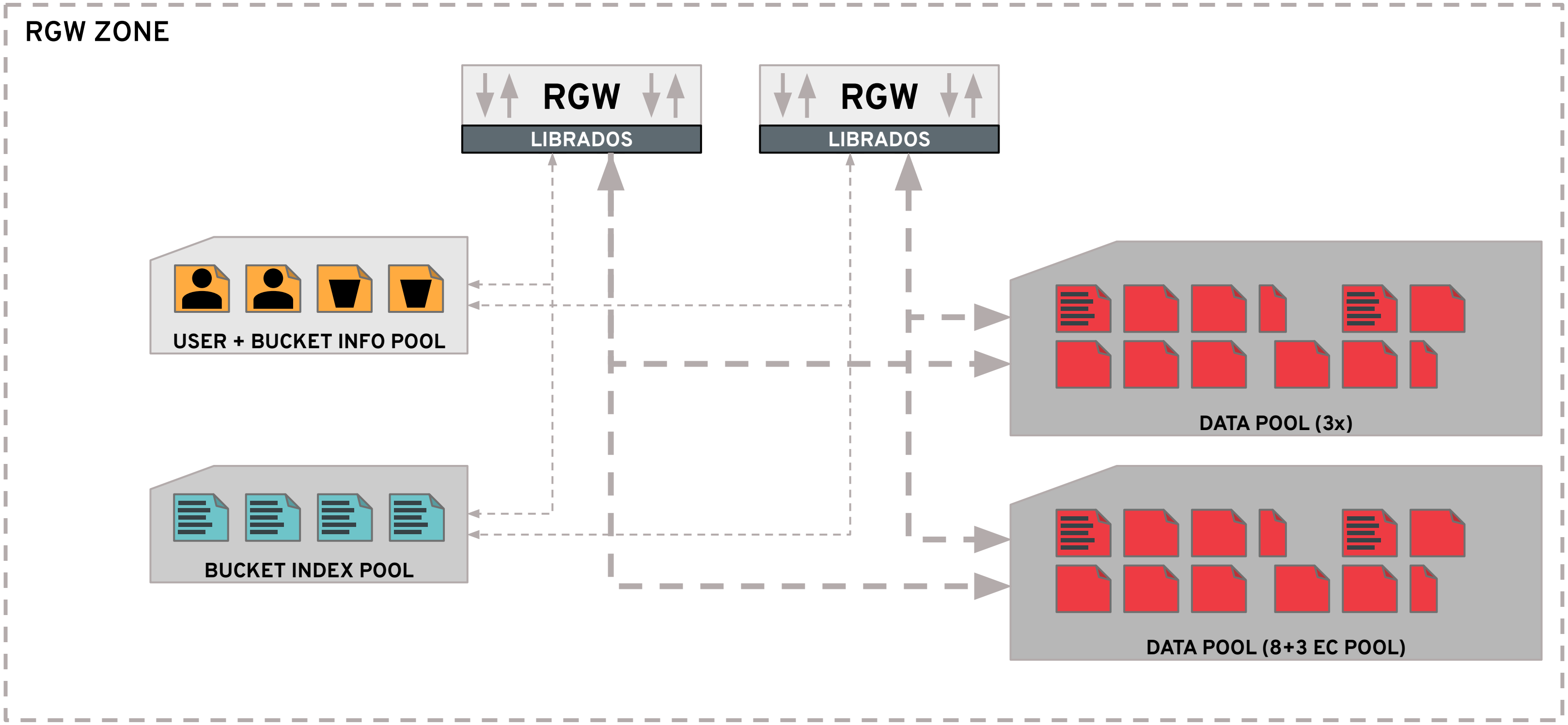
- S3 and Swift-compatible object storage
  - HTTPS/REST-based API
  - Often combined with load balancer to provide storage service to public internet
- Users, buckets, objects
  - Data and permissions model is based on a superset of S3 and Swift APIs
  - ACL-based permissions, enforced by RGW
- RGW objects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



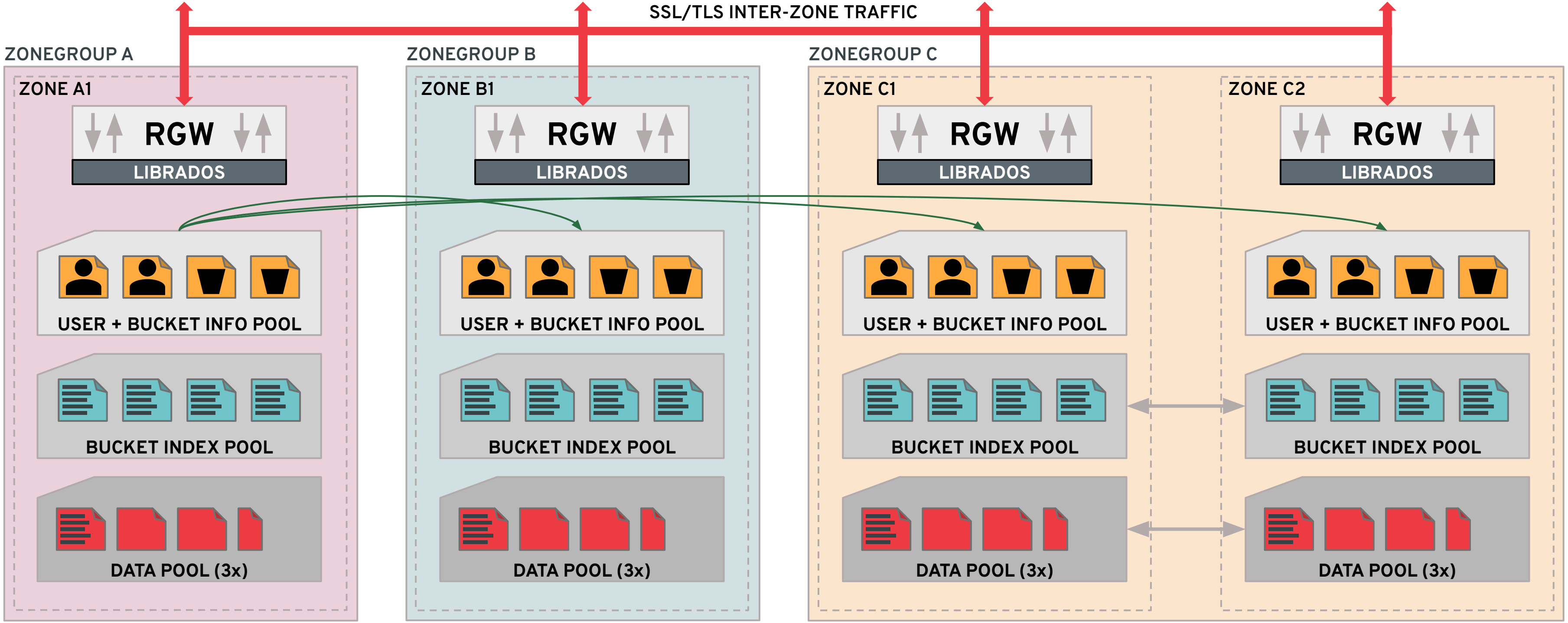
# RGW STORES ITS DATA IN RADOS



# 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

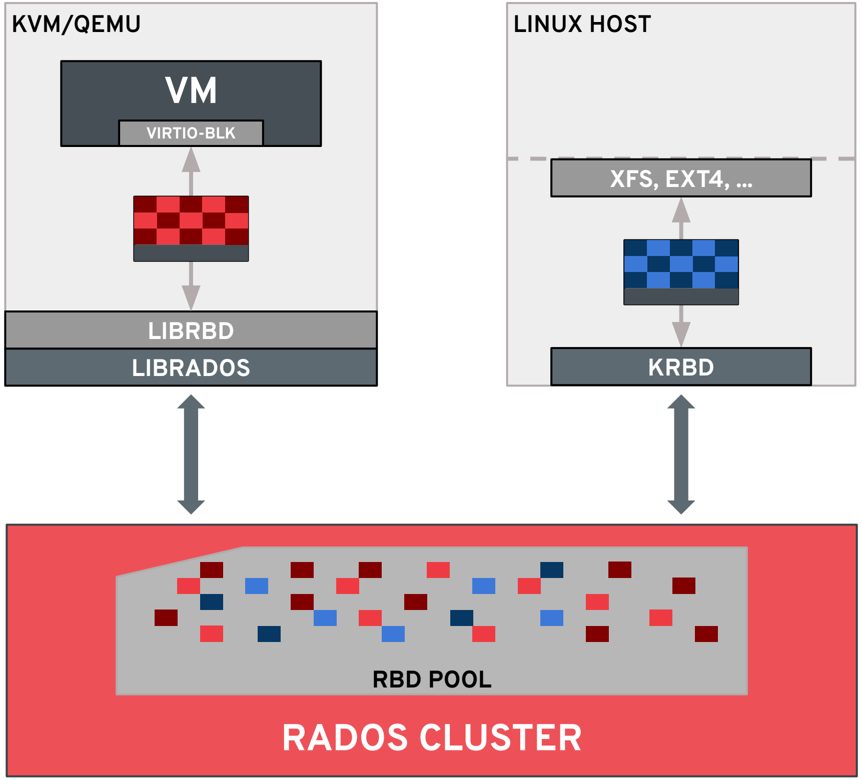
# OTHER RGW FEATURES

- 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

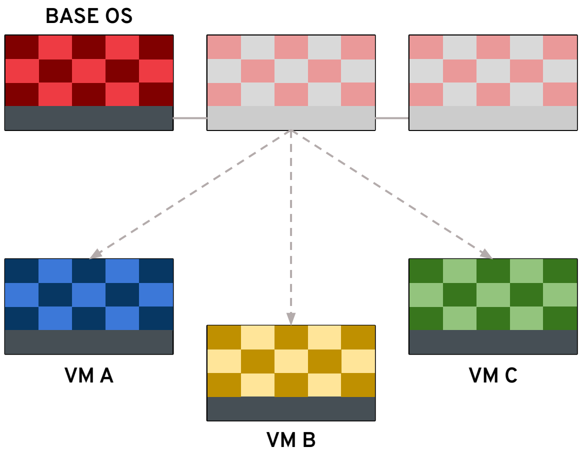
# RBD: RADOS BLOCK DEVICE

- 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, …

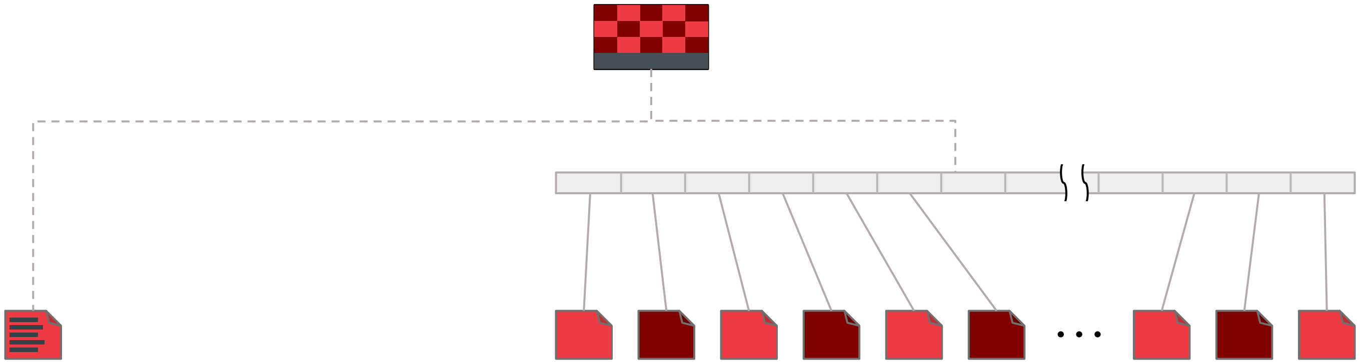


# SNAPSHOTS AND CLONES

- 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



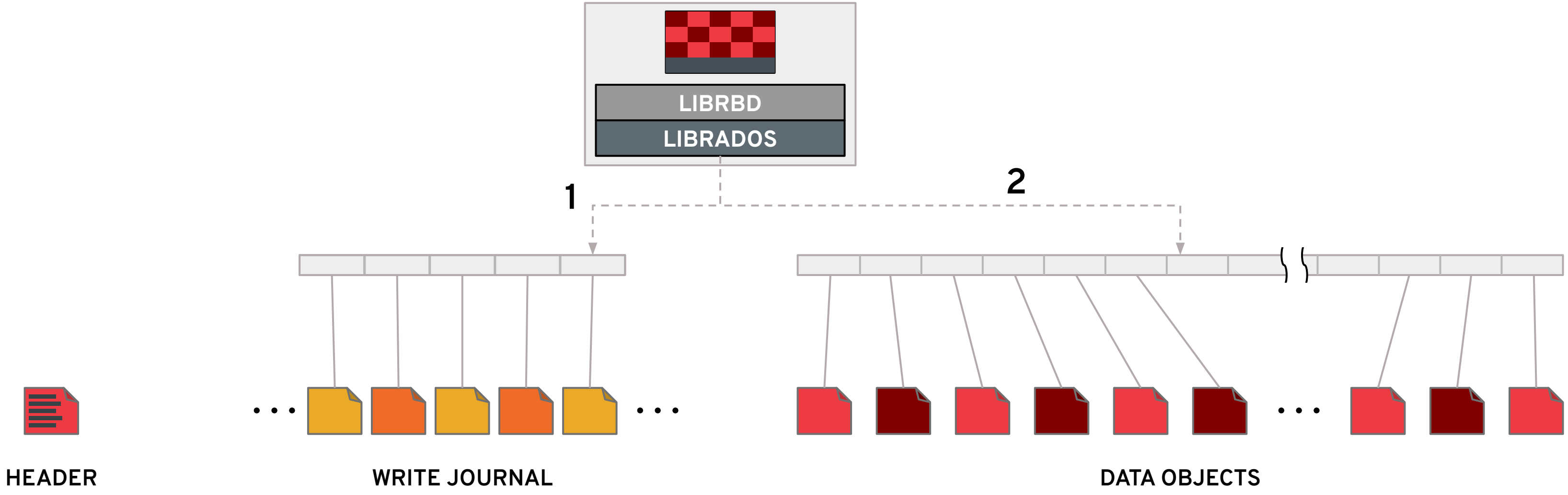
**HEADER**

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

**DATA OBJECTS**

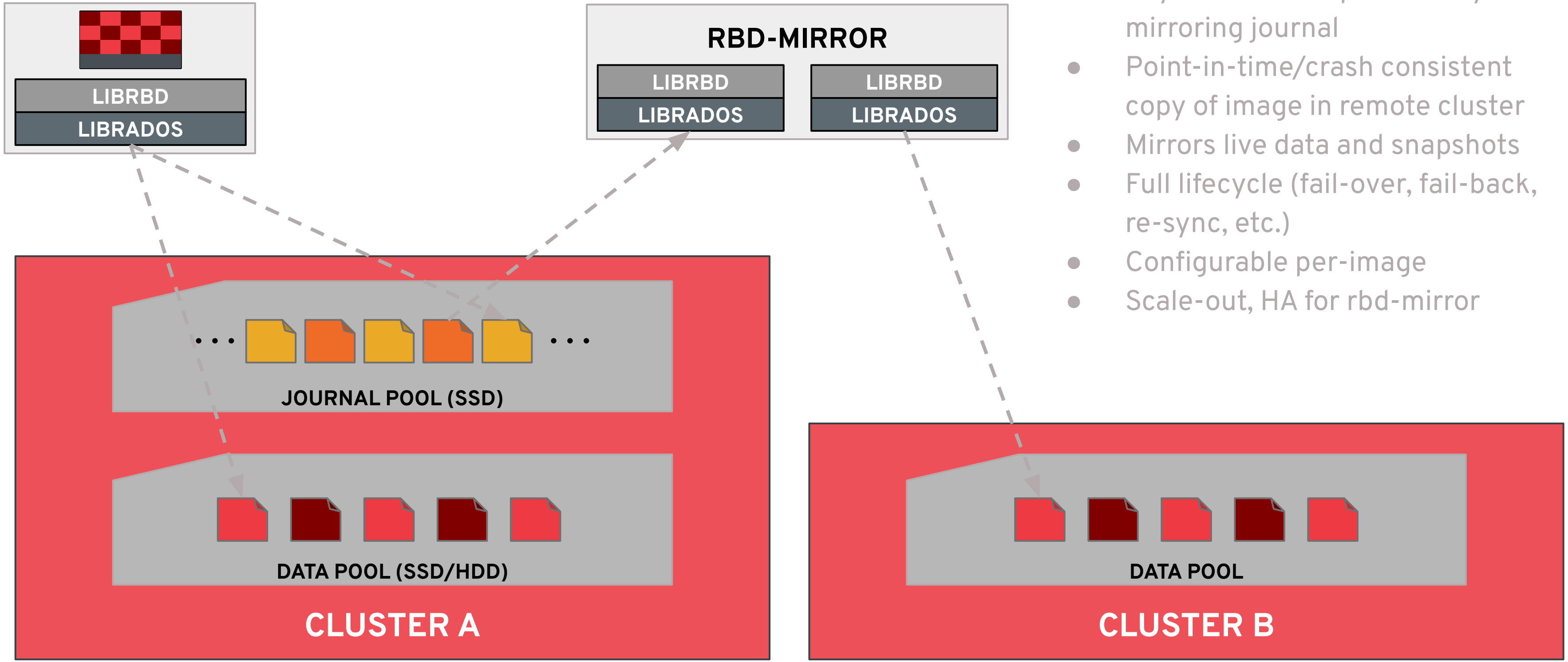
- Chunk of block device content
- 4 MB by default, but striping is configurable
- Sparse: objects only created if/when data is written
- Replicated or erasure coded, depending on the pool

# 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

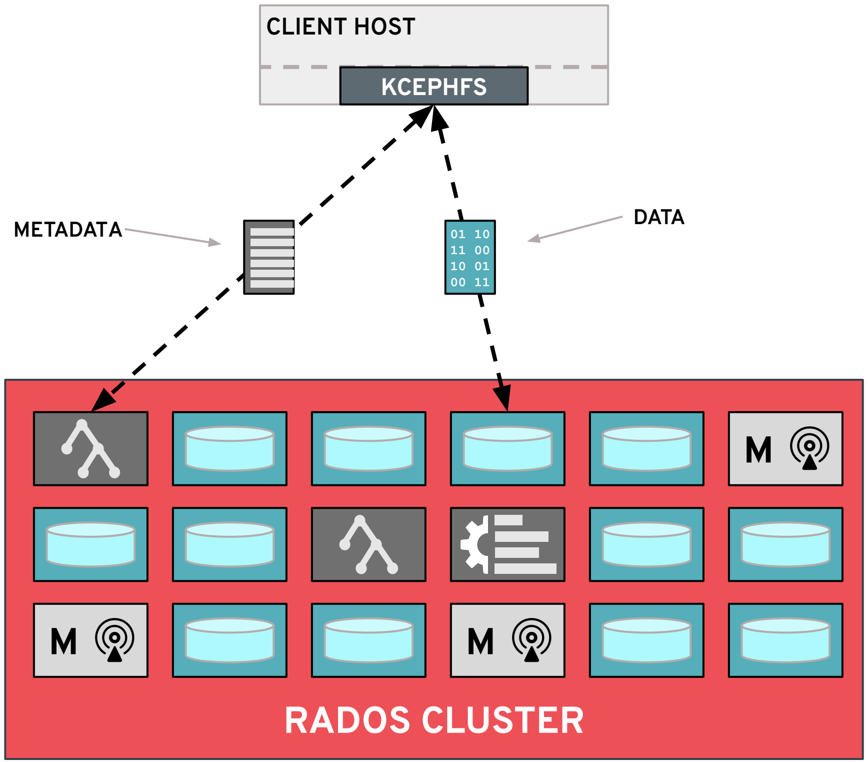
# OTHER RBD FEATURES

- ‘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

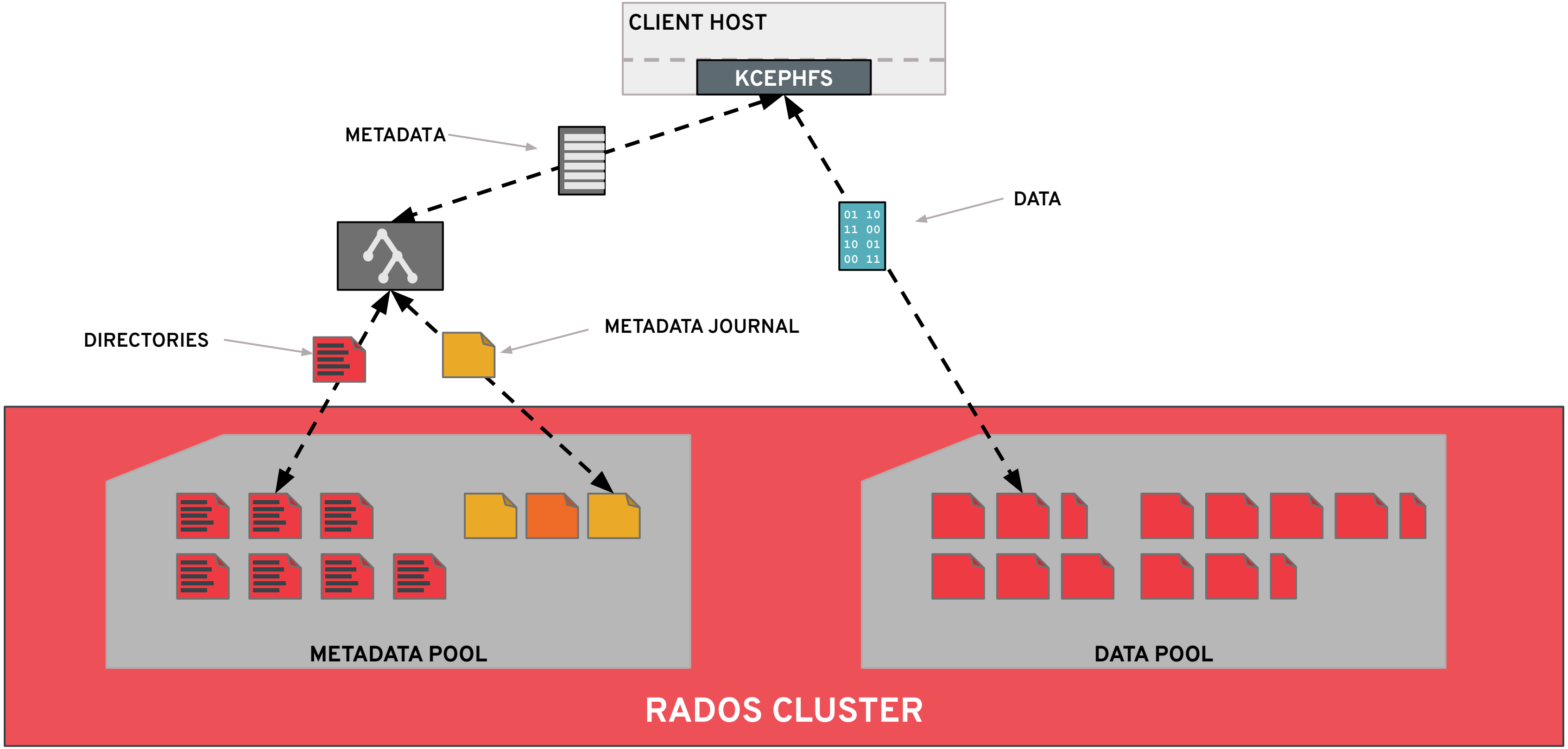
**ceph-mds**

<u>MDS (Metadata Server)</u>

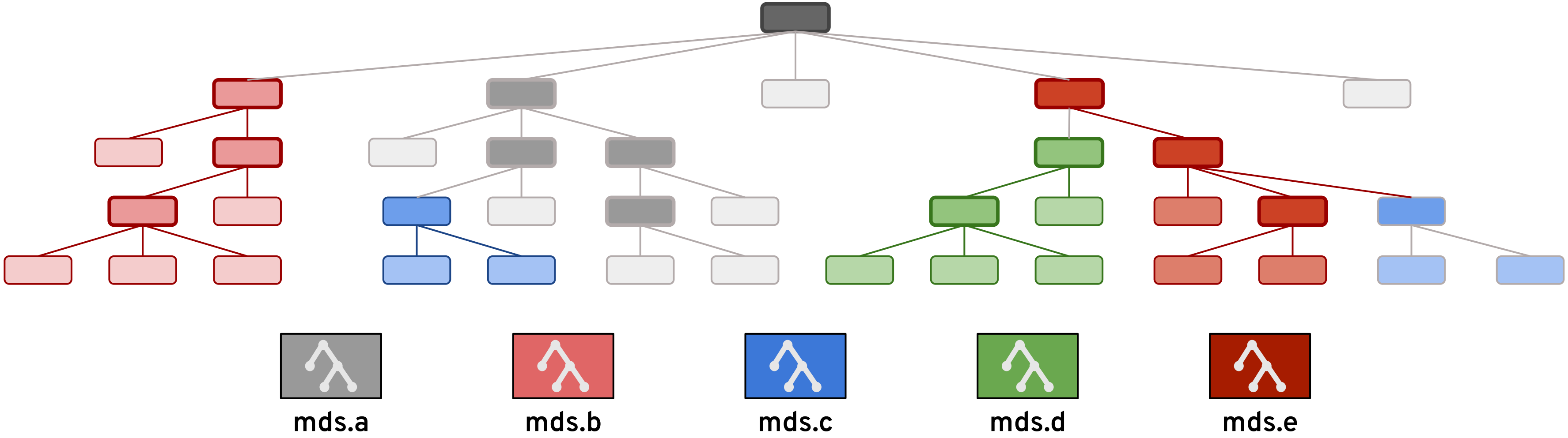
- Manage file system namespace
- Store file system metadata in RADOS objects
    - File and directory metadata (names, inodes)
- Coordinate file access between clients
- Manage client cache consistency, locks, leases
- Not part of the data path
- 1s - 10s active, plus standbys

**ceph-mon** **ceph-mgr** **ceph-osd**

# METADATA IS STORED IN RADOS



# SCALABLE NAMESPACE




- Partition hierarchy across MDSs based on workload
- Fragment huge directories across MDSs
- Clients learn overall partition as they navigate the namespace
- Subtree partition maintains directory locality
- Arbitrarily scalable by adding more MDSs

# 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
$
```

# CEPHFS RECURSIVE ACCOUNTING

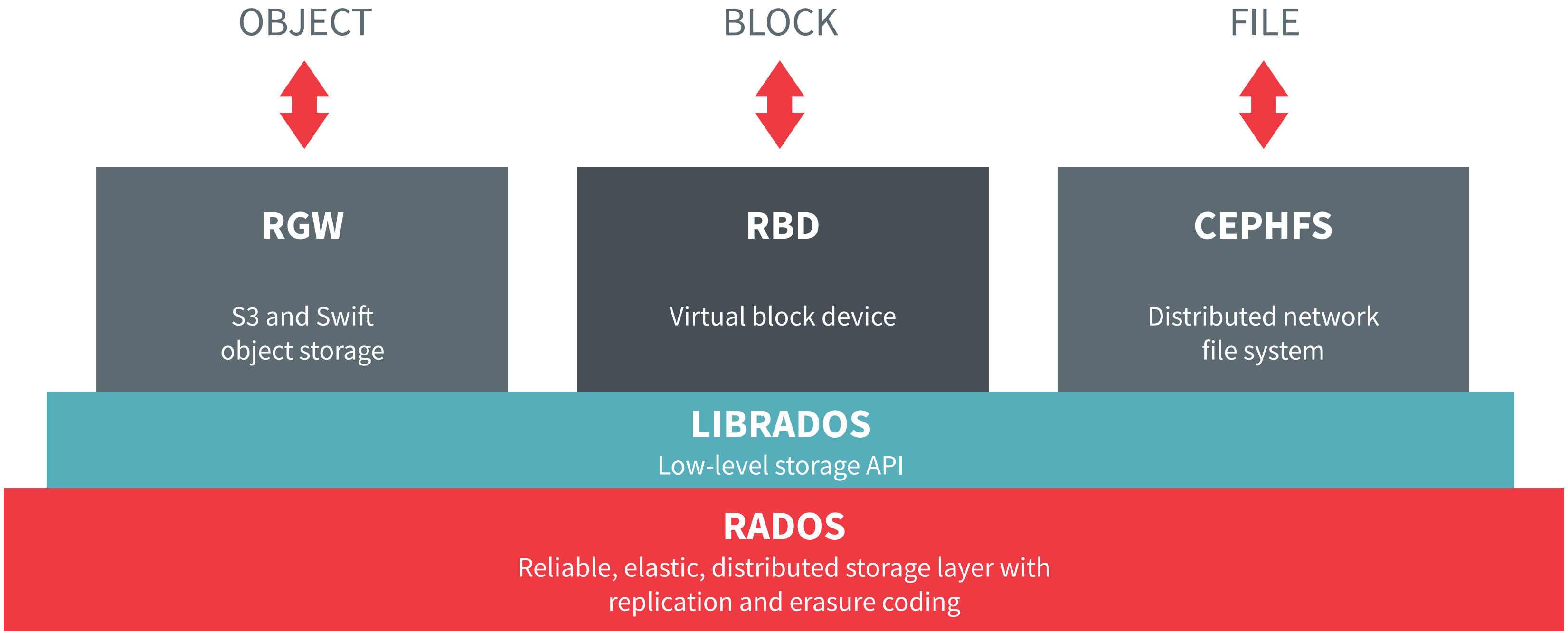
- MDS maintains recursive stats across the file hierarchy
  - File and directory counts
  - File size (summation)
  - Latest **ctime**
- Visible via virtual xattrs
- Recursive bytes as directory size
  - If mounted with ‘rbytes’ option
  - Unfortunately this confuses rsync; off by default
  - Similar to ‘du’, but free

```
$ sudo mount -t ceph 10.1.2.10:/ /mnt/ceph \
-o name=admin,secretfile=secret,rbytes
$ cd /mnt/ceph/some/random/dir
$ getfattr -d -m - .
# file: .
ceph.dir.entries="3"
ceph.dir.files="2"
ceph.dir.subdirs="1"
ceph.dir.rbytes="512000"
ceph.dir.rctime="1474909482.0924860388"
ceph.dir.rentries="17"
ceph.dir.rfiles="16"
ceph.dir.rsubdirs="1"
$ ls -alh
total 12
drwxr-xr-x  3 sage sage 4.5M Jun 25 11:38 ./
drwxr-xr-x 47 sage sage  12G Jun 25 11:38 ../
-rw-r--r--  1 sage sage   2M Jun 25 11:38 bar
drwxr-xr-x  2 sage sage 500K Jun 25 11:38 baz/
-rw-r--r--  1 sage sage   2M Jun 25 11:38 foo
```

# 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



# II. Ceph Use-Cases at CERN

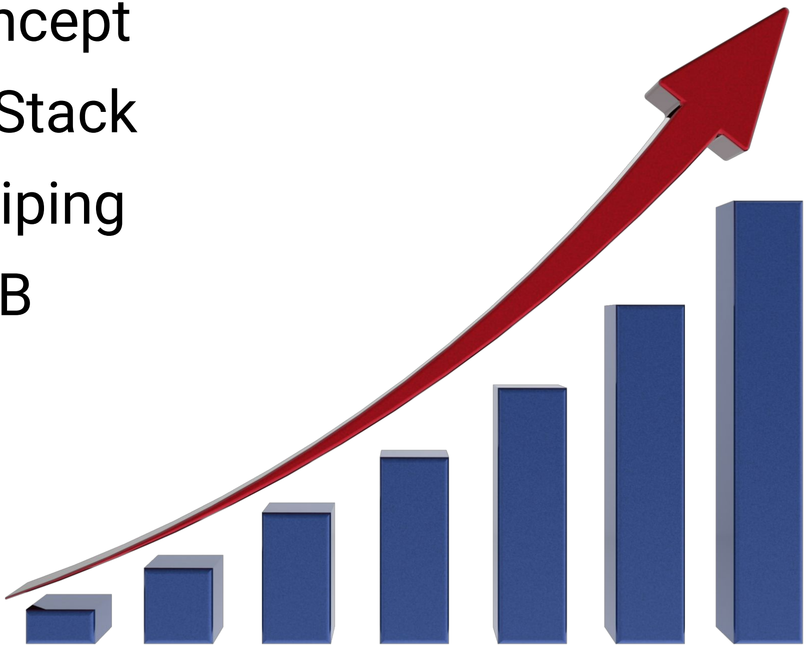
# CERN Computing Infrastructure

- High throughput scientific computing platform:
  - HTCondor batch system: ~250k CPU cores
  - EOS storage system: ~500 petabytes of raw storage
  - CTA tape system for long term archival: ~500 petabytes of tape
- IT infrastructure brings several storage needs:
  - Block Storage and NAS Filers for VMs and Databases
  - Object Storage for web or cloud native applications
  - HPC Scratch areas for MPI clusters
  - “Open Infrastructure"

# CERN IT Open Infrastructure

# Our Ceph History

- March 2013: 300TB proof of concept
- Dec 2013: 3 petabytes for OpenStack
- 2014-15: Erasure coding and striping
- 2016: Upgraded from 3PB to 6PB
- 2017: 8 production clusters
- 2018-19: CephFS and S3
- 2020+: scale out...



CERN

# Current Clusters in Prod (I)

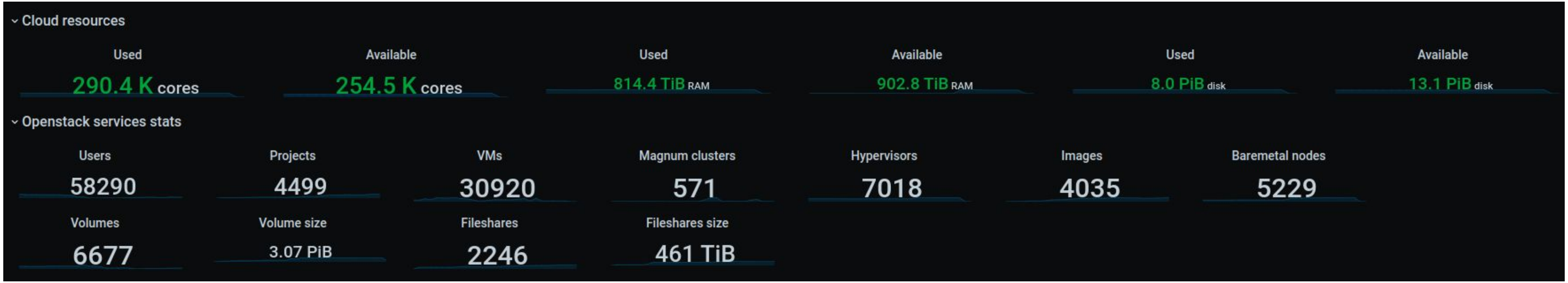
- **Block Storage** for OpenStack
  - Three hdd (w/ssd rocksdb) clusters: 24 petabytes raw (3x replication)
  - Three all-flash clusters: 1.2 petabytes raw (2+2 erasure coding)
  - Integrated to OpenStack as multiple QoS types (IOPS throttles) and availability zones
- **S3 Object Storage**
  - Two clusters in different data centres: 12.5 petabytes raw
  - Data stored in 4+2 erasure coding on HDDs, bucket indices on SSDs
  - Currently independent realms; Working on zonegroup replication now.

# Current Clusters in Prod (II)

- **CephFS**
  - Two general purpose hdd (w/ ssd rocksdb) clusters: 6.3 petabytes raw (3x replication)
  - One general purpose all-flash cluster: 500TB raw (3x replication)
  - Several targeted all-flash clusters: hyperconverged DB tests, groupware, HPC, …
- General experience is that Ceph is **robust and performant**
  - Data remains consistent after infrastructure outages; failure recovery is basically transparent
  - Hardware replacement and flexibility demonstrated across three procurement cycles

# CERN IT OpenStack Cloud

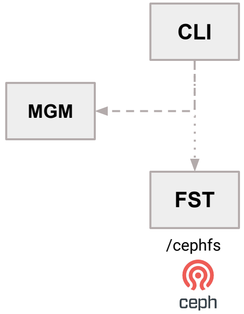
- Since 2013, hosting 90% of CERN's computing resources for scientific and IT needs




openstack®

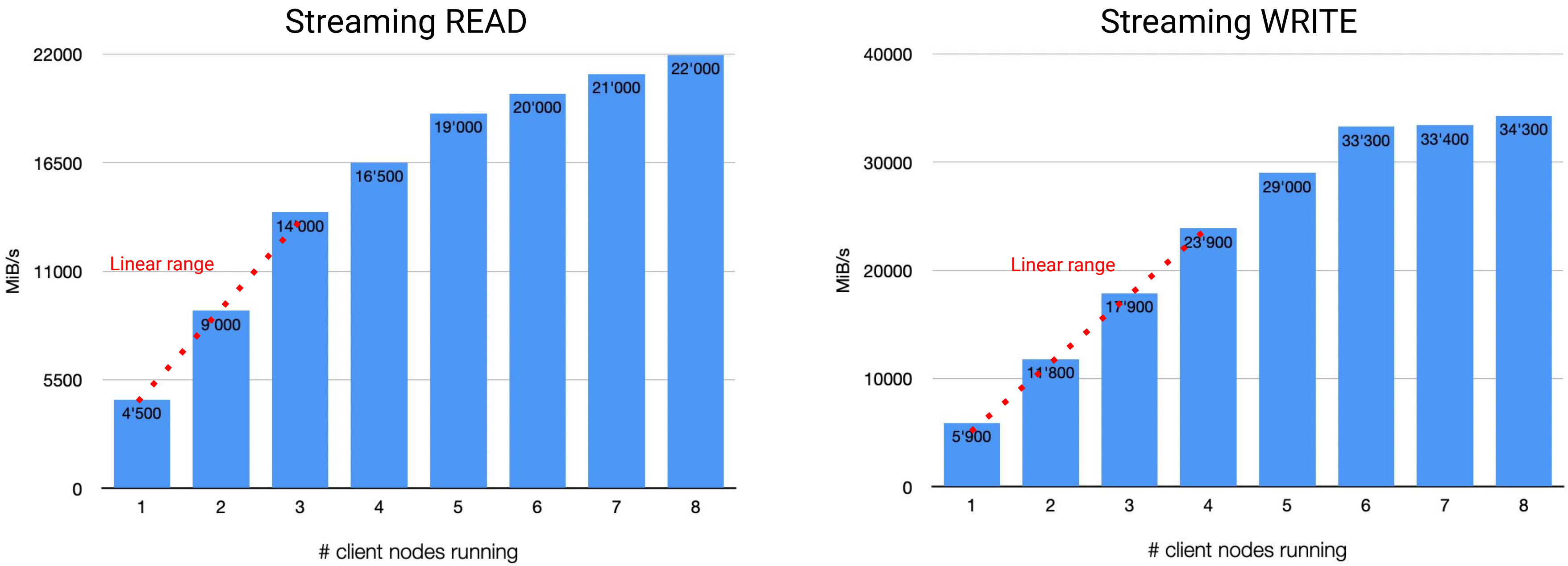
# EOS on CephFS

- EOS storage **developed at CERN for physics** and regular users: **350PB**
- Is it feasible / useful to layer EOS on top of a Ceph backend?
    - Best of both worlds: feature-rich EOS for scientific users + flexible object storage on Ceph
- EOS is clustered storage built upon Xrootd:
    - Files can be replicated or erasure-coded; metadata in “QuarkDB”
    - FST (analogous to the Ceph OSD) normally stores files in a local XFS
        - Files stored using a simple inode hash naming convention
- It’s therefore **straightforward to use CephFS** in the FST
    - Durability is delegated to CephFS
    - EOS configured to store data with a single replica

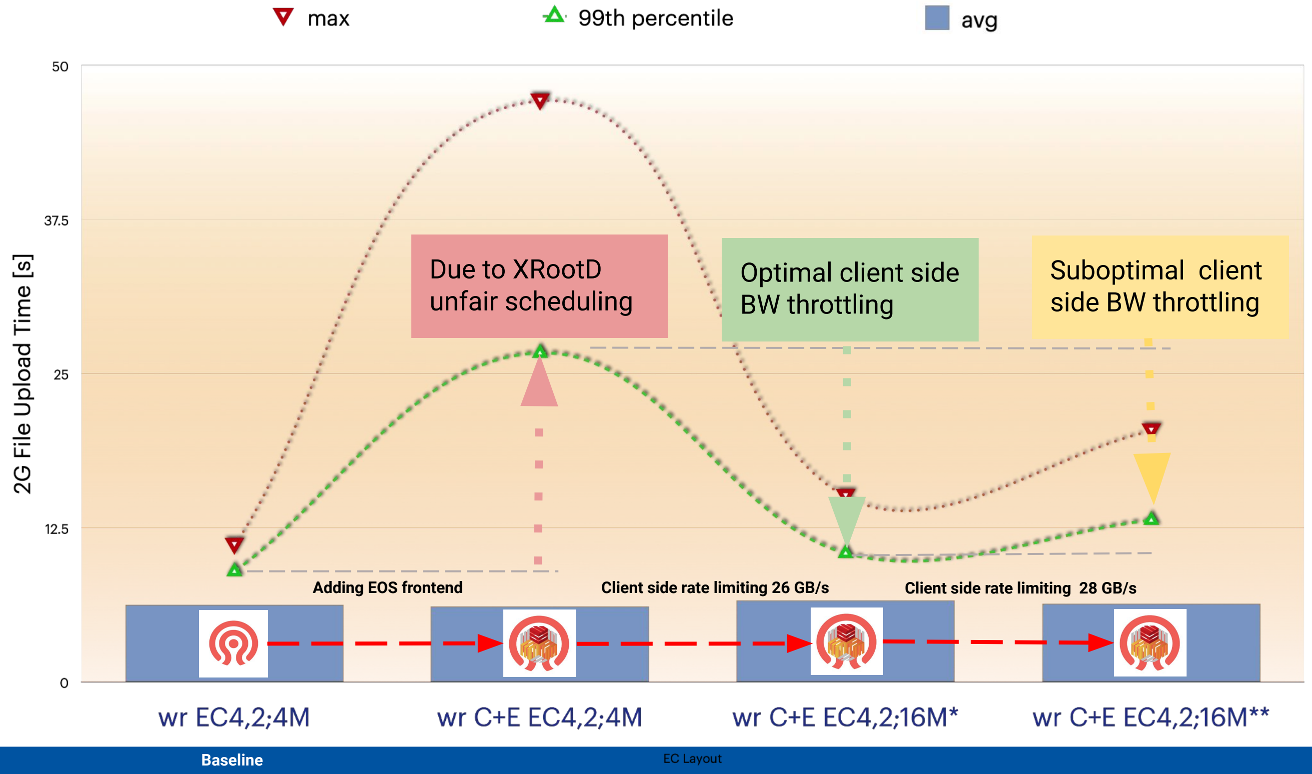



Peters, A.J., van der Ster, D.C. Evaluating CephFS Performance vs. Cost on High-Density Commodity Disk Servers. Comput Softw Big Sci 5, 25 (2021). https://doi.org/10.1007/s41781-021-00071-1

# PoC: CephFS Scalability Measurements



# CephFS+EOS Write Performance Impact?

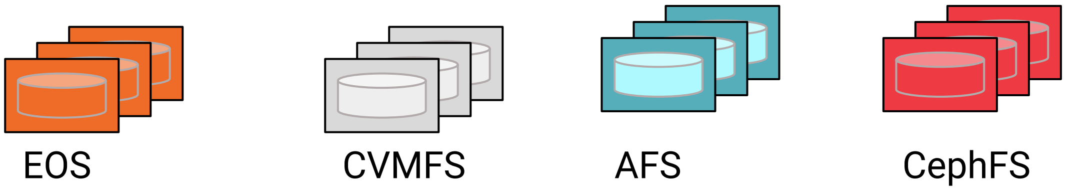



CEPHFS + EOS

# III. CephFS for HPC

# Why use CephFS for HPC?

- At CERN we're already running many network filesystems
  - No desire to introduce yet another (e.g. Lustre)



# Why use CephFS for HPC?

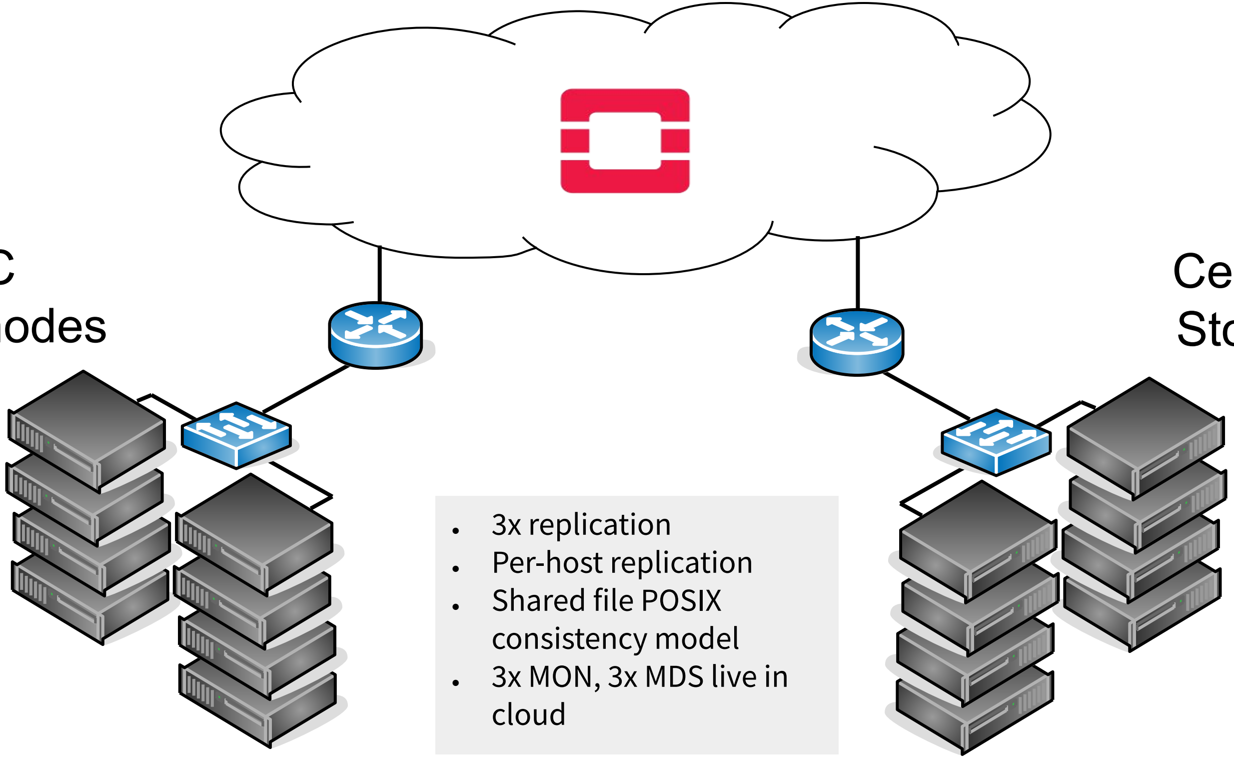
- HPC cluster procurement process optimization
  - We aren’t ordering an HPC cluster from a vendor, it’s 100% DIY HPC + Open Source tech.
  - HW procured as a large order (meant for HTC, HPC, Storage..)
  - Low-latency interconnect added on top

# Why use CephFS for HPC?

- Desire to evaluate CephFS as a multi-purpose filesystem
  - Today: ~ 4 years of experience running CephFS on production for CERN IT's HPC service

HPC Workernodes

CephFS Storage



- 3x replication
- Per-host replication
- Shared file POSIX consistency model
- 3x MON, 3x MDS live in cloud

# Evolution of CephFS as HPC scratch space

- **Shared** CephFS cluster for IT services

BUT…

- **Contention**. I/O-intensive applications affecting other IT services and vice-versa
- CephFS cluster "far" in the network
  - Less resilient to **network issues**
  - Greater I/O **latency**

**ceph-fuse** mounts

BUT…

- Ceph-fuse **not very performant**
- Ceph-fuse issues with **stuck mounts** and stale data after/during network issues

# Evolution of CephFS as HPC scratch space

- **Shared** CephFS cluster for IT services

BUT…
  - I/O-intensive applications affecting other IT services and vice-versa
  - CephFS cluster “far” in the network
    - Less resilient to network issues
    - Greater I/O latency

→ Transitioned to **dedicated** CephFS cluster for increased network failure **resiliency** and **performance**

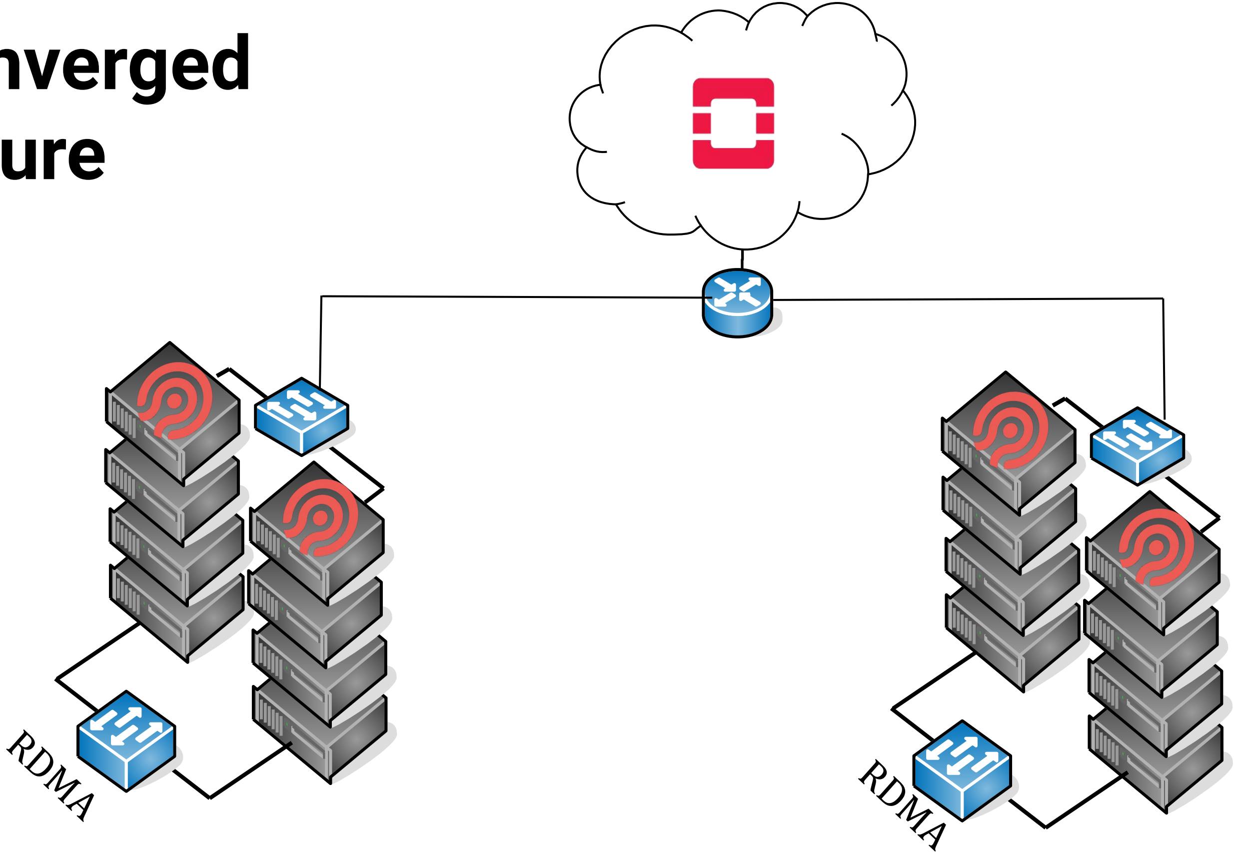
**ceph-fuse** mounts

BUT…
  - Ceph-fuse not very performant
  - Ceph-fuse issues with stuck mounts and stale data after/during network issues

→ Transitioned to **kernel** mounts for greater **performance**, greater **stability**, and much improved **resiliency** to network issues.

# Hyperconverged architecture

Ceph-osd daemons run on HPC compute nodes



# CephFS performance tuning

**Network-locality**: Client/MDS/Disk locality has more than 10x impact on performance

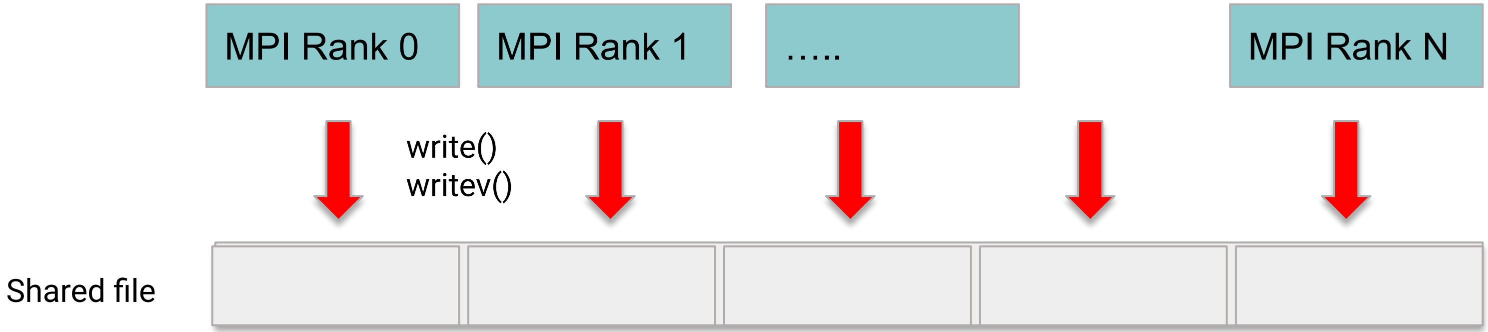
**Replication factor**: Tuning replica count had an impact on write latency.

**Automatic MDS balancing**: Works, but manual pinning can do better if you know the workload

**Lazy I/O**: Much improved performance for single-shared-file collective I/O

# CephFS performance tuning: Lazy I/O

- **Lazy I/O** refers to a mode in which **POSIX semantics are relaxed**
- For shared file collective I/O, coherency is delegated to the application
- Allows **lock-free parallel writes**
- CephFS mode with lazy I/O support added to IOR [https://github.com/hpc/ior]



# Limitations and future plans

- Impact of the Hyperconverged architecture on MPI collectives
  - openQCD is a very tightly coupled HPC application with excellent scalability.
  - Very **sensitive to OS noise**
  - **20%** performance impact from system noise (e.g. Hyperconverged)
- Burst Buffers
  - To significantly **reduce** or remove the **impact** of independent workloads on each other.
- How does a Hyperconverged solution affect day-to-day IT operations?

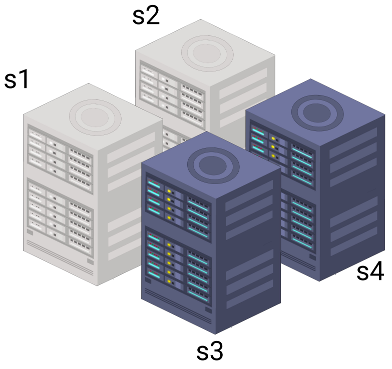
# Impact on automation and IT operations

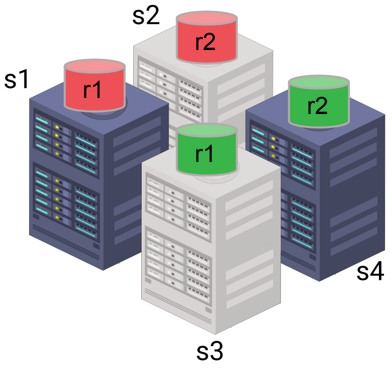
Hyperconverged increases complexity for **transparent operations**

(e.g. kernel reboot campaigns)

Batch System + Distributed Storage System

Drain & shutdown nodes **independently**

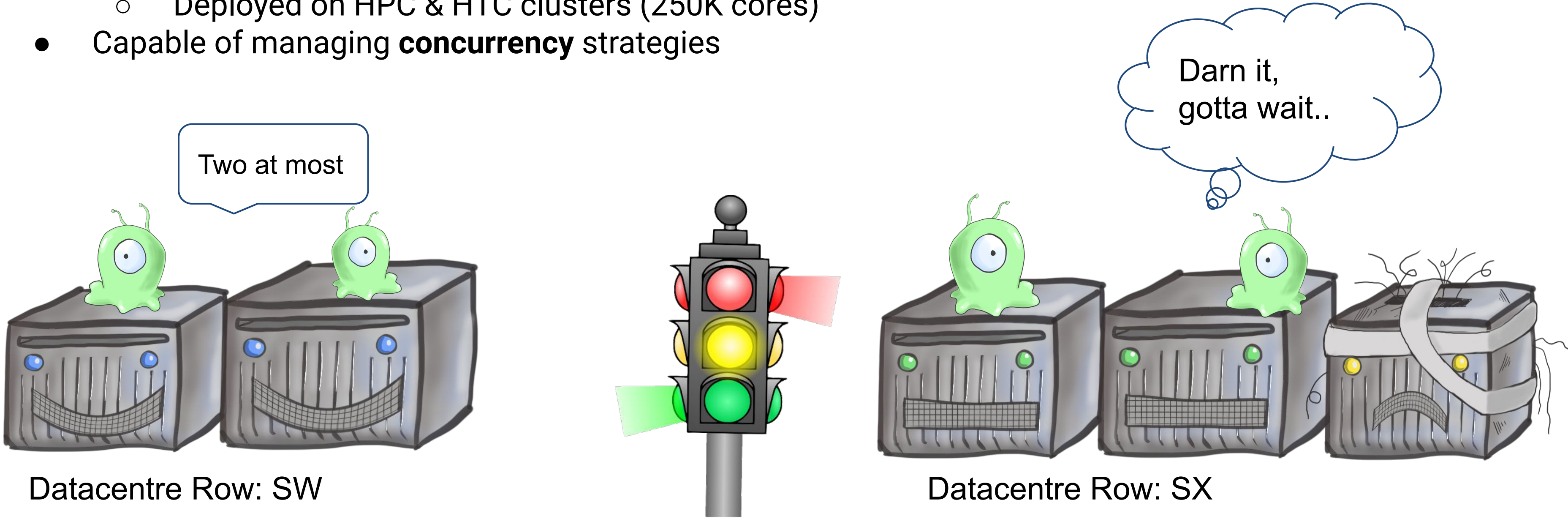



Shutdown nodes according to **data replication** strategy

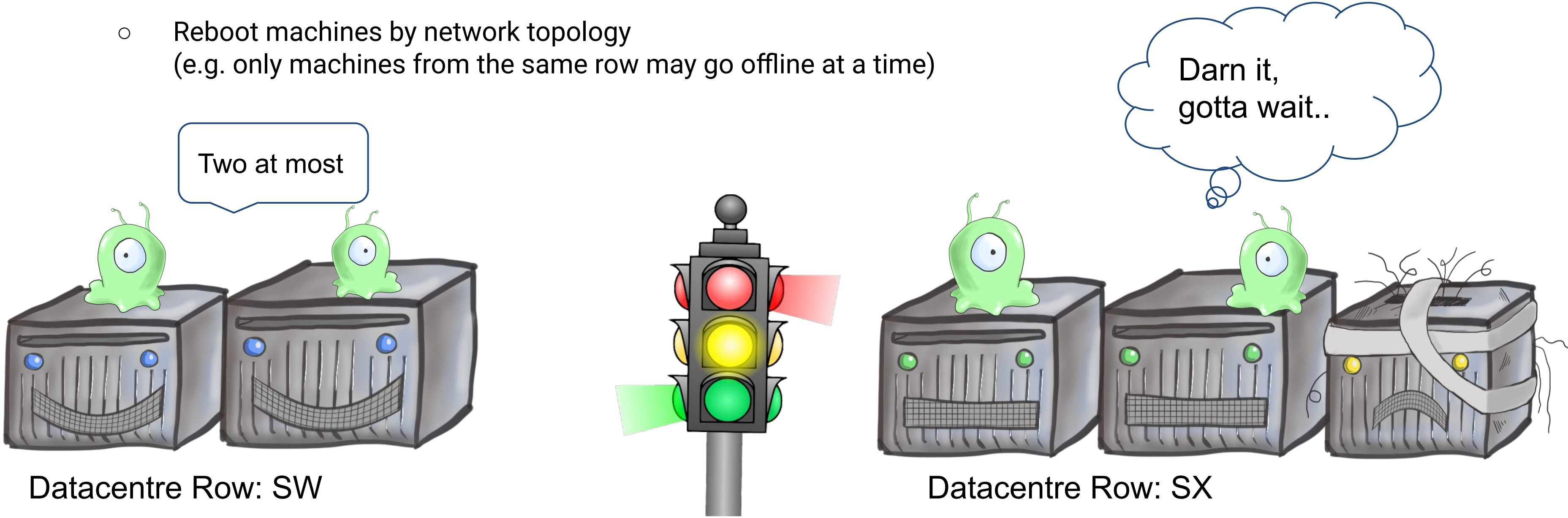
# Impact on automation and IT operations

- **Brainslug** is an **automation tool** written at CERN
- Lightweight daemon running on every node
  - Machine **state manager** (Slurm & HTCondor)
  - Deployed on HPC & HTC clusters (250K cores)
- Capable of managing **concurrency** strategies



# Impact on automation and IT operations

- **Brainslug** is capable of **orchestration** based on user-defined **concurrency strategies**
  - Limit number of nodes draining/offline at a time
    (e.g. drain+reboot 10% at a time)
  - Reboot machines by network topology
    (e.g. only machines from the same row may go offline at a time)



# IIII. Final Words


ceph
cephalocon
2022
July 11-13, 2022 • Portland, OR + Virtual

# Thank you!
# Any Questions?

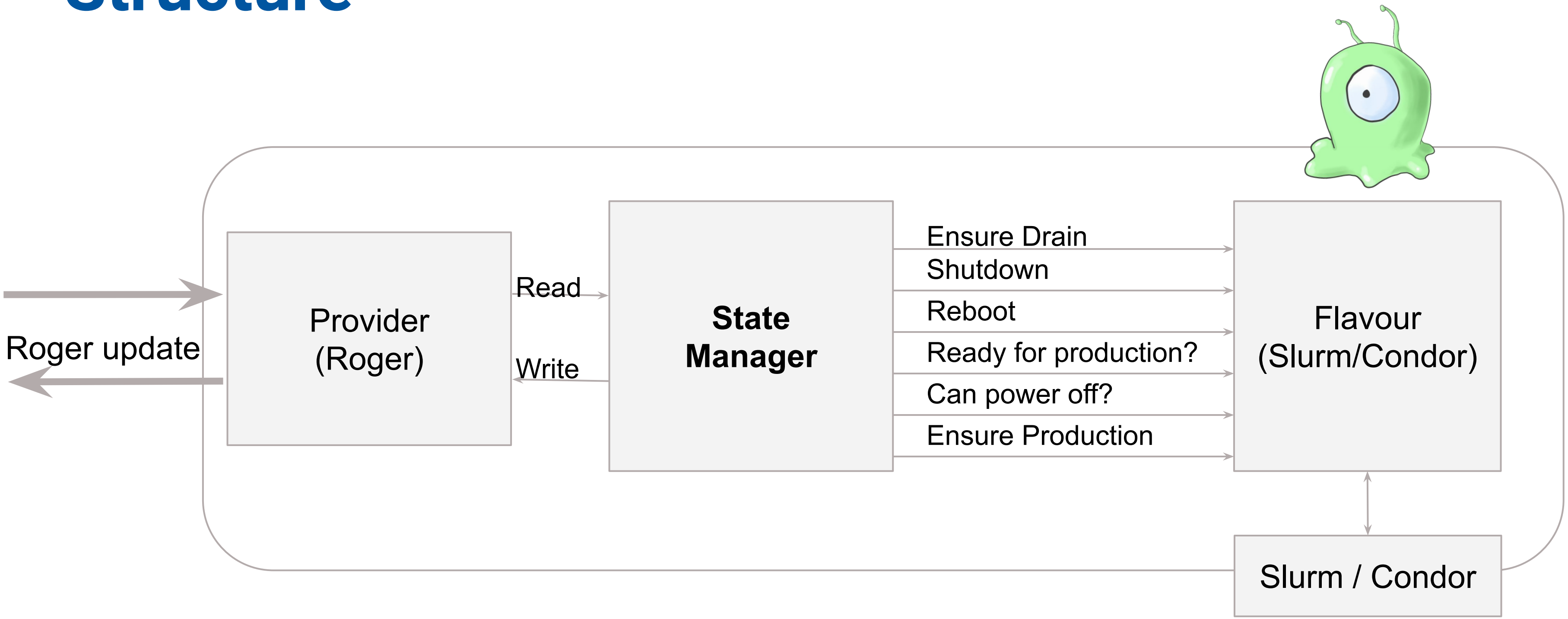
Feel free to get in touch:

daniel.vanderster@cern.ch
pablo.llopis@{gmail.com,cern.ch}

https://ceph.io

# Extra Slides

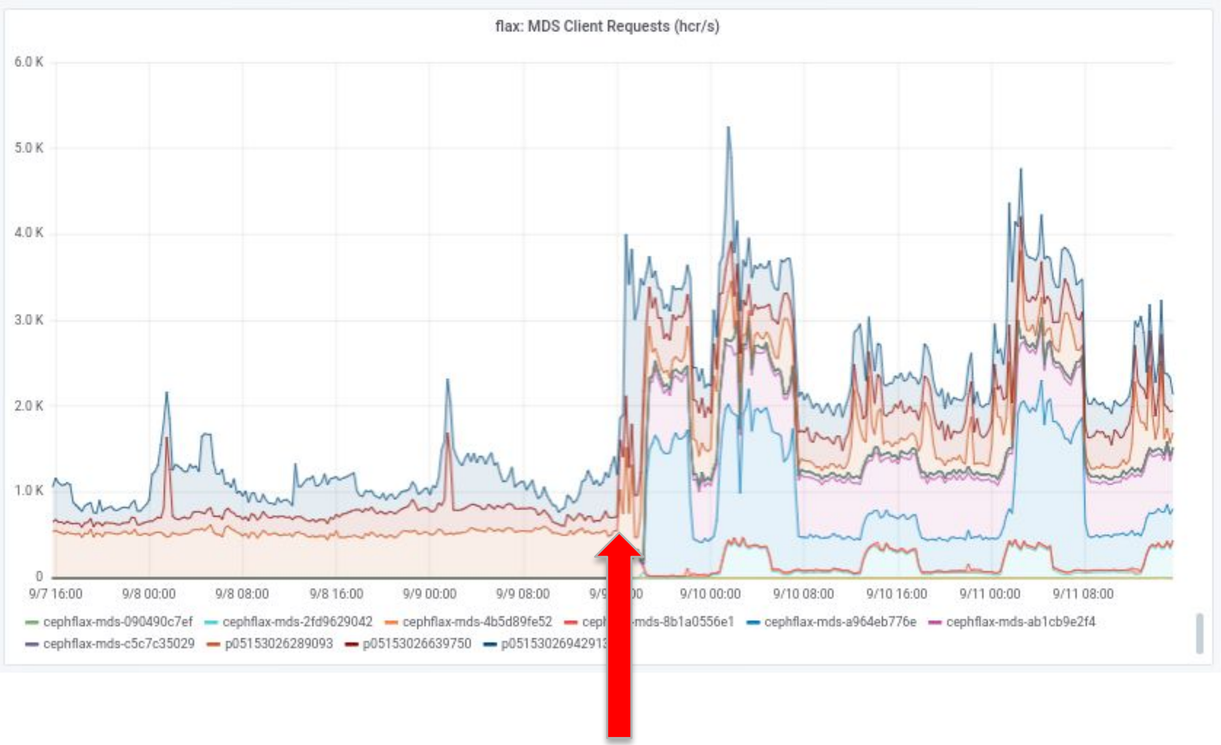
# Structure


Roger update
Provider
(Roger)
Read
Write
State
Manager
Ensure Drain
Shutdown
Reboot
Ready for production?
Can power off?
Ensure Production
Flavour
(Slurm/Condor)
Slurm / Condor

# CephFS Performance

- In previous years we invested in profiling and tuning CephFS for HPC
  - **Automatic MDS balancing**: works, but manual pinning can do better if you know the workload
  - **Keep things local**: Client/MDS/Disk locality has more than 10x impact on performance
  - **LazyIO for parallel IO**: relaxed consistency hints managed by the application
- Tuning for the **IO-500 benchmark** as published at SuperComputing
  - ior: throughput tests for single or multi-file parallel IO
  - mdtest/find: metadata performance tests

# CephFS Scale-Out MDS in Practice




From 3 to 10 active MDS's

# CephFS and IO-500

IO500

| # | information | | | | | | | io500 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | institution | system | storage vendor | filesystem type | client nodes | client total procs | data | score | bw GiB/s | md kIOP/s |
| 1 | University of Cambridge | Data Accelerator | Dell EMC | Lustre | 512 | 8192 | zip | 620.69 | 162.05 | 2377.44 |
| 2 | Oak Ridge National Laboratory | Summit | IBM | Spectrum Scale | 504 | 1008 | zip | 330.56 | 88.20 | 1238.93 |
| 3 | JCAHPC | Oakforest-PACS | DDN | IME | 2048 | 2048 | zip | 275.65 | 492.06 | 154.41 |
| ... | | | | | | | | | | |
| 34 | SUSE | TigerShark | SUSE, Intel, Lenovo | CephFS | 14 | 98 | zip | 8.38 | 3.58 | 19.60 |
| 35 | Clemson University | Palmetto | Dell | BeeGFS | 48 | 48 | zip | 7.64 | 2.93 | 19.88 |
| 36 | CERN | Bytecollider | | CephFS | 64 | 64 | zip | 7.56 | 2.83 | 20.16 |