

High Performance Computing and Data Infrastructure

HPC parallel storage infrastructure



DATA SCIENCE &
ARTIFICIAL INTELLIGENCE



SCIENTIFIC &
DATA-INTENSIVE COMPUTING

2024-2025 @ Università di Trieste

Agenda

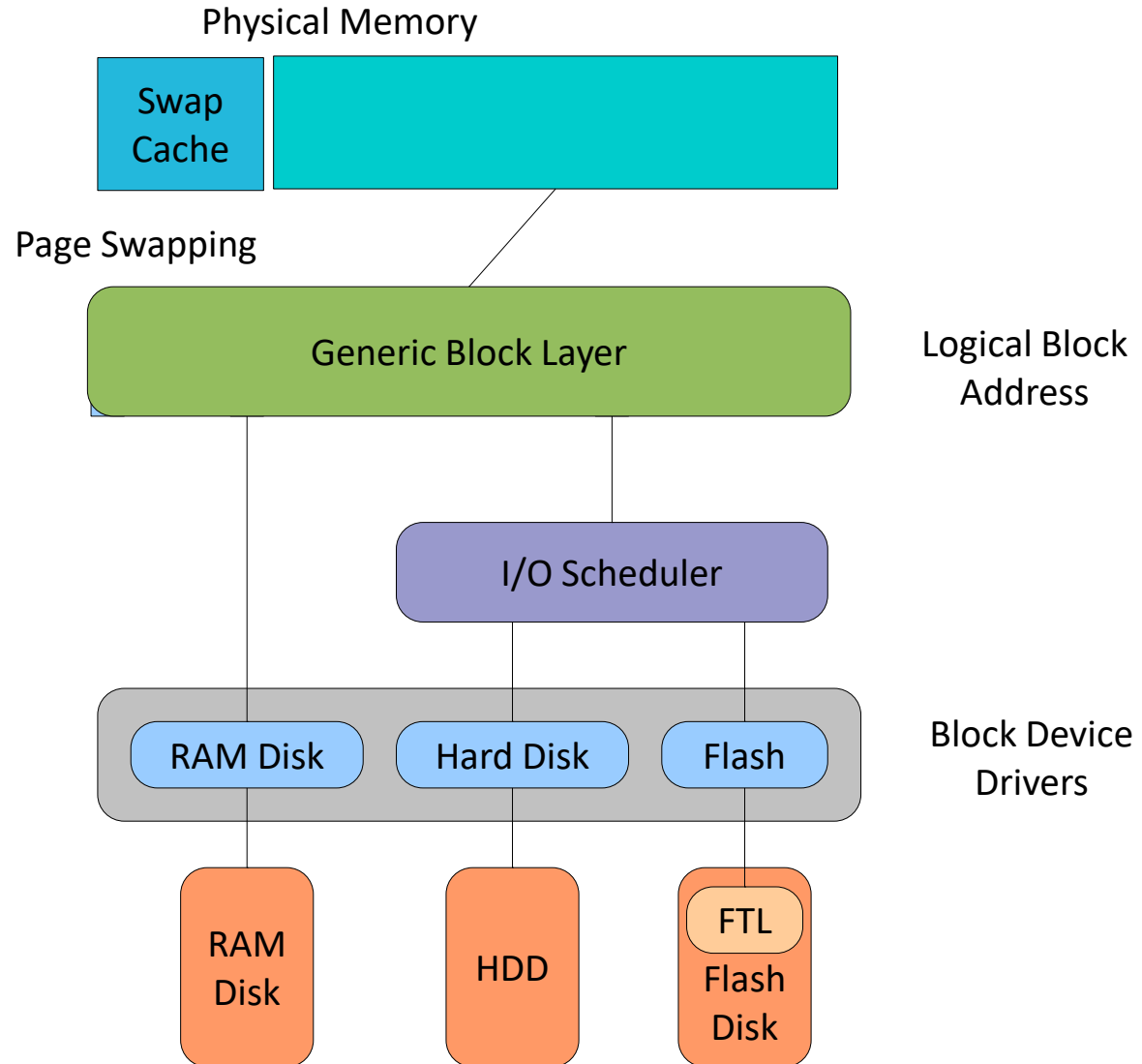
- Short recap:
 - Storage hierarchy
 - Storage building blocks
- CEPH filesystem

Storage Hierarchy

- Storage follows a hierarchy with multiple levels:
 - RAM disk, I/O buffers or file system cache
 - Local disk (flash based, spinning disk) (SATA, SAS, RAID, SSD, JBOD, ...)
 - Local network attached device or file system server (NAS, SAN NFS, CIFS, PFS, Lustre, GPFS, CEPH)
 - Tape based archival system (often with disk cache)
 - External, distributed file systems (Cloud storage)

Same as with the memory hierarchy:
Register -> Cache (L1->L2->L3) -> RAM

Storage Hierarchy



RAM Disk

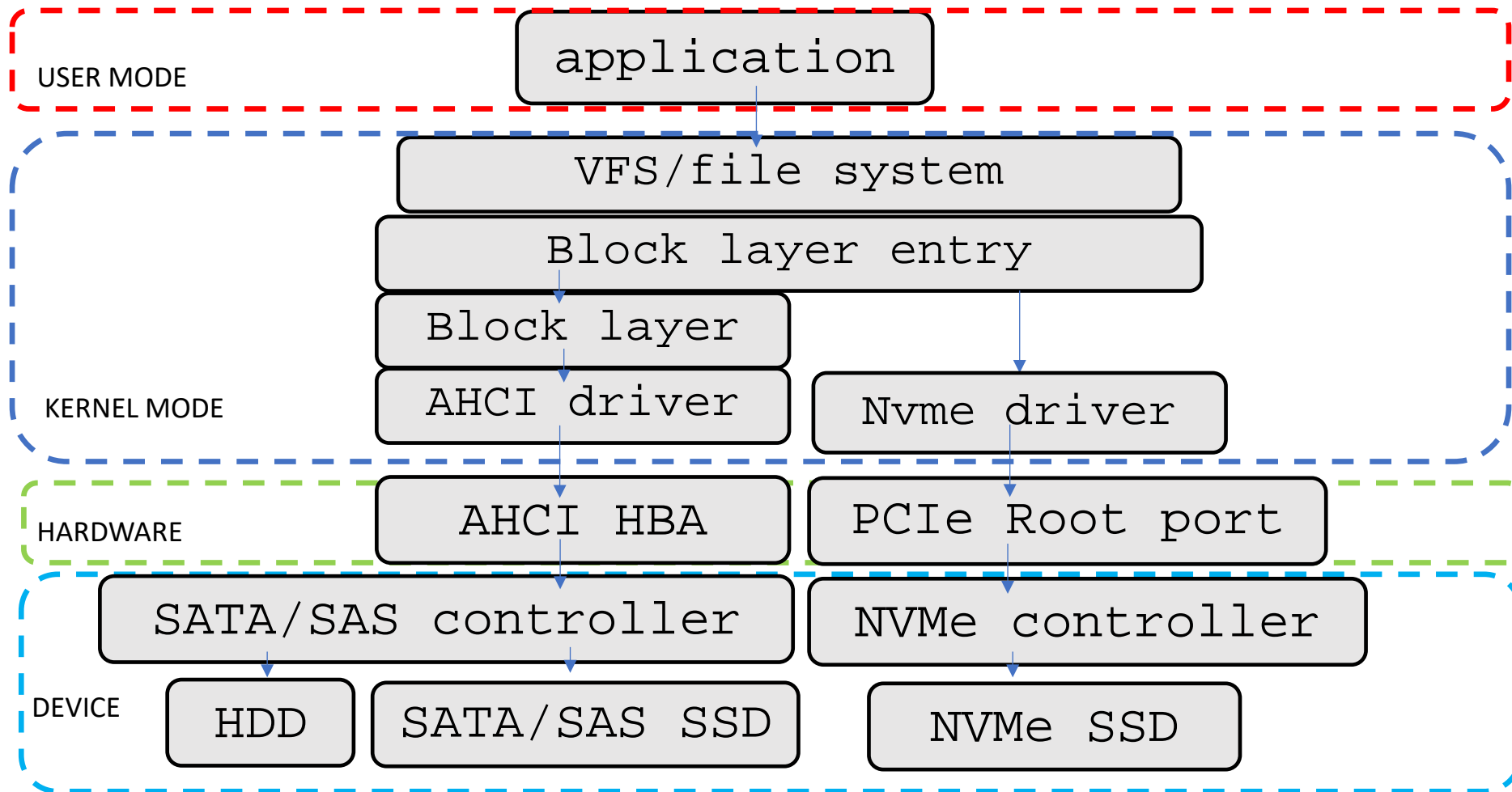
- Unix-like OS environments very frequently create (small) temporary files in /tmp, etc.
 - faster access and less wear with RAM disk
 - Linux provides “dynamic RAM disk” (tmpfs)
 - only existing files consume RAM
 - automatically cleared on reboot (-> volatile)

```
[cozzini@login ~]$ df
Filesystem            1K-blocks      Used    Available Use% Mounted on
devtmpfs              1915112         0    1915112    0% /dev
tmpfs                 1939960         0    1939960    0% /dev/shm
tmpfs                 1939960    25316    1914644    2% /run
tmpfs                 1939960         0    1939960    0%
/sys/fs/cgroup
/dev/vda1             41931756 11442916    30488840   28% /
```

NVMe (Non-volatile Memory express)

- NVMe is an *“optimized, high-performance, scalable host controller interface with a streamlined register interface and command set designed for non-volatile memory based storage.”*
- Designed to fix many of the issues of legacy SAS/SATA.
 - SATA /SAS protocols for mechanical drive
 - Now the bottleneck
- Physical connectivity is much simplified, with devices connected directly on the PCIe bus

NVMe (Non-volatile Memory express)



ORFEO storage: hardware

	FAST storage (NVMe)	FAST storage (SSD)	Standard storage (HDD)	Long term preservation
# of server	4		10	1
RAM	6 x 16GB		6 x 16GB	6 x 16GB
Disk per node	-4x 1.6TB NVMe PCIe card	20 x 3.84TB	18 x 12TB + 18x16TB + (on the 2 new server)	84 x 12TB + 84 x 12TB+ 84x 12TB
Storage provider	CEPH parallel FS	CEPH parallel FS	CEPH parallel FS	Network FS (NFS)
RAW storage	24TB	320 TB	1872 TB	3024 TB

ORFEO new storage: hardware

	FAST storage (NVMe)	Standard storage (HDD)
# of server	11	
Total RAM	192GB	
RAM per disk	32GB	8G
Total cores	28	
Cores for O.S	4	
Cores (storage)	6 core x disk	1 core x disk
Disk per node	2 x 15.36TB Enterprise NVMe Read Intensive U.2 Gen4	12 x 1 22TB Hard Drive SAS 12Gbps 7.2K 512e 3.5in Hot- Plug

The new ORFEO basic brick: NVME

- Device Type
 - SSD –NVME no hot swap
- Capacity
 - 15.36TB
- Form Factor
 - PCI-express + 2.5 inches
- Performance
 - To be measured



The ORFEO basic brick: HDD drive

- Device Type
 - Hard drive - hot-swap – nearline 7.w
- Capacity
 - 22 TB
- Form Factor
 - 3.5"
- Interface
 - SAS 12Gb/s
- Performance
 - Drive Performance :
 - Interface Transfer Rate : 1200 Mbps
 - Sustained Transfer Rate : 291/277 Mbps
 - Random Read 4kb Qd : 210 Iops
 - Random Write 4kb Qd : 565/565 Iops
 - Random 50/50 Read/write 4kb Qd : 214 Iops
 - Average Latency : 4.16 Ms
 - Buffer : 512 Mb
 - Rotational Speed : 7200 Rpm



RAID Parameters

Level	Description	Minimum # of drives	Space Efficiency	Fault Tolerance	Read Benefit	Write Benefit
RAID 0	Block-level striping without parity or mirroring.	2	1	0 (none)	nX	nX
RAID 1	Mirroring without parity or striping.	2	1/n	n-1 drives	nX	1X
RAID 4	Block-level striping with dedicated parity.	3	1-1/n	1 drive	(n-1)X	(n-1)X
RAID 5	Block-level striping with distributed parity.	3	1-1/n	1 drive	(n-1)X	(n-1)X
RAID 6	Block-level striping with double distributed parity.	4	1-2/n	2 drives	(n-2)X	(n-2)X
RAID 1+0/10	Striped set of mirrored sets.	4	*	needs 1 drive on each mirror set	*	*
RAID 0+1	Mirrored set of striped sets.	4	*	needs 1 working striped set	*	*

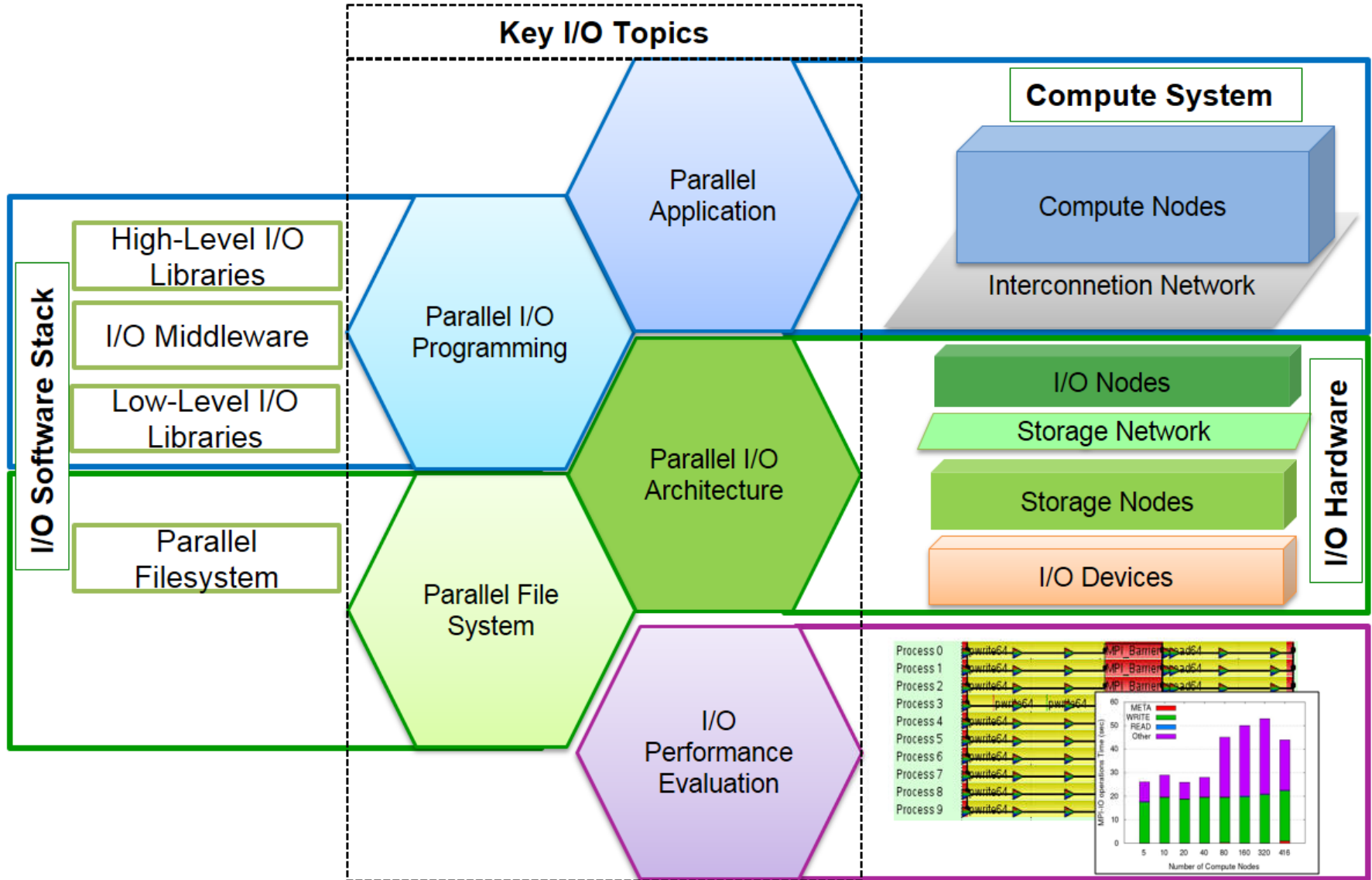
* depends on the # of mirrored/striped sets and # of drives

From <http://en.wikipedia.org/wiki/RAID>

RAID on ORFEO storage

- RAID 1 on all nodes for OS reliability
- RAID0 on DGX001 for scratch
- For actual storage: NONE
 - For CEPH FS redundancy managed at disk level (see later)
 - For long term storage redundancy managed at hardware/software layer within the NAS (see later)

Parallel I/O in HPC

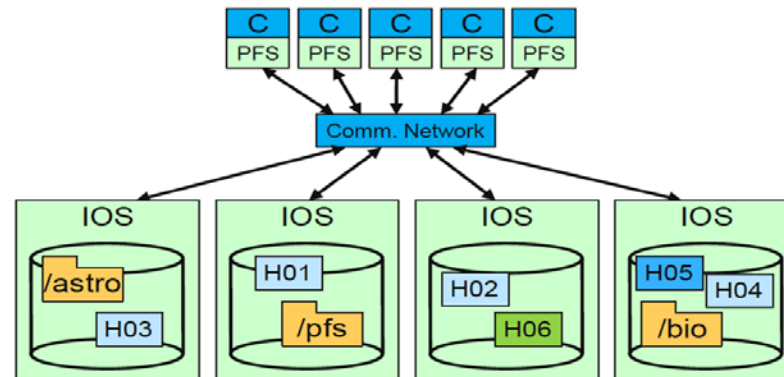
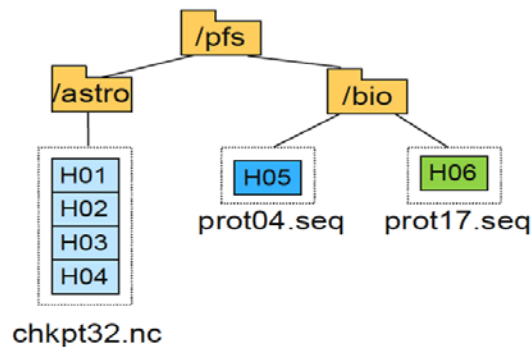


Elements of a PFS

- A parallel solution usually is made of
 - several Storage Servers that hold the actual filesystem data
 - one or more Metadata Servers that help clients to identify/manage data stored in the file system
 - a redundancy layer that replicates in some way information in the storage cluster, so that the file system can survive the loss of some component server
- and optionally:
 - monitoring software that ensures continuous availability of all needed components

Parallel FS approaches..

- An example parallel file system, with large astrophysics checkpoints distributed across multiple I/O servers (IOS) while small bioinformatics files are each stored on a single IOS



What is available on the market ?

- BeeGFS
 - Developed at Fraunhofer Institute, freely available not open
 - <http://www.fhgfs.com/cms/>
- Lustre
 - open and Free owned by Intel DDN
 - Intel no longer sells tools to manage and support (\$\$\$)
 - <http://lustre.opensfs.org/>
- GPFS (now known as Spectrum Scale)
 - IBM proprietary \$\$\$
 - Very nice solution and expensive ones !
- And many others (WekaIO/MooseFS/Panasas... etc)

CEPH storage

- Open-source distributed storage solution
- Object based storage
- Highly scalable
- Built around the CRUSH algorithm, by Sage Weil – <http://ceph.com/papers/weil-crush-sc06.pdf>
- Supports multiple access methods [File, Block, Object]

Documentation for some of the following slides:

- <https://www.redbooks.ibm.com/redpapers/pdfs/redp5721.pdf>

Ceph Architecture

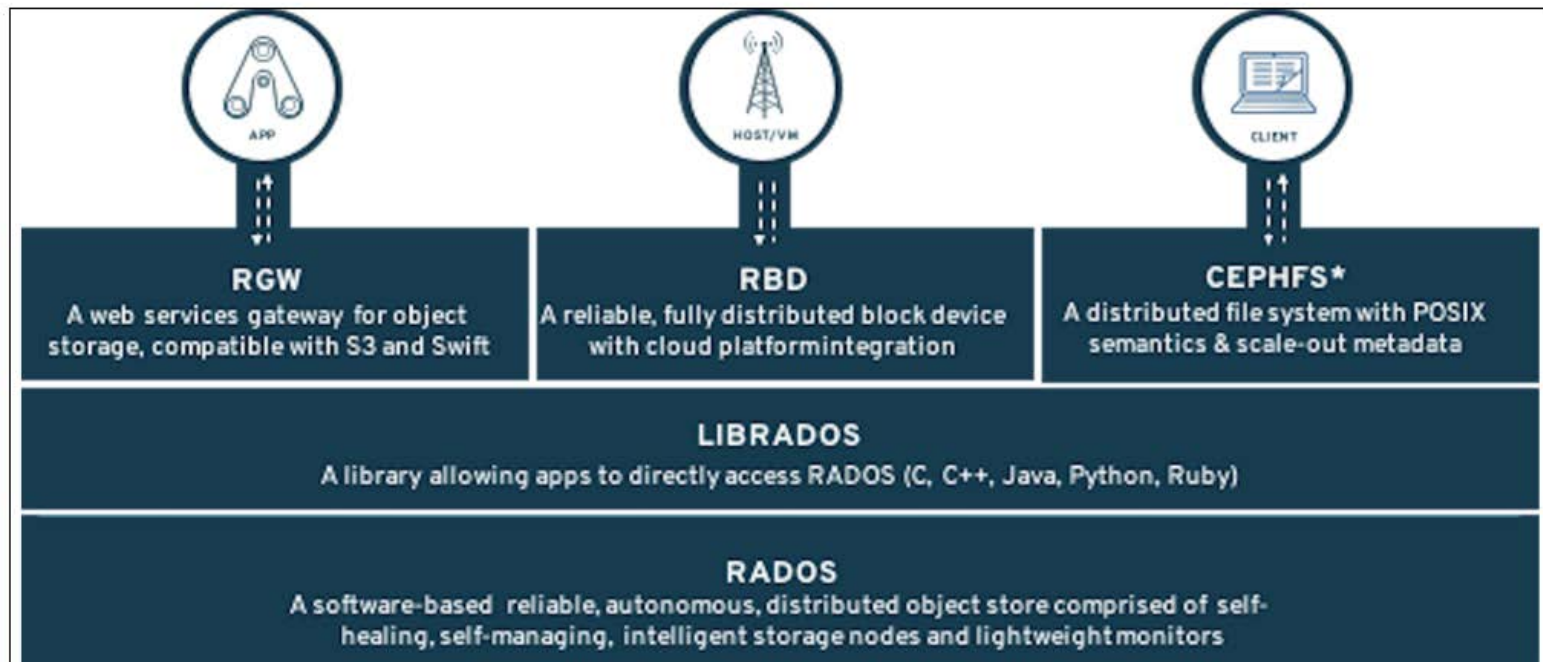


Figure 2-1 Ceph general architecture

CEPH storage cluster: RADOS

- **RADOS** (Reliable Autonomic Distributed Object Store)
 - This layer provides the CEPH software defined storage with the ability to store data (serve I/O requests, protect the data, check the consistency and the integrity of the data through built-in mechanisms).
- The RADOS layer is composed of the following daemons:
 - MONs or Monitors
 - OSDs or Object Storage Devices
 - MGRs or Managers
 - MDSs or Meta Data Servers (only for CEPHfs)

Monitors

- responsible for managing the state of the cluster.
- Ceph maintains its cluster state through a set of specialized maps, collectively referred to as the cluster map.
- Each map is assigned a unique version number (epoch), which starts at 1 and increments by 1 on every state change for the corresponding set of components.

Monitors

- The MONs maintain the following maps:
 - MON map
 - MGR map
 - OSD map
 - MDS map
 - Placement group (PG) map
 - CRUSH map
- To ensure the integrity and consistency of map updates, the MONs employ the PAXOS algorithm, enabling them to reach a consensus among multiple MONs before validating and implementing any map changes.
- To prevent split-brain scenarios, the number of MONs that are deployed in a Ceph cluster must always be an odd number greater than two to ensure that most MONs can validate map updates.
- More than half of the MONs that are present in the Monitor Map (MONMap) must agree on the change that is proposed by the PAXOS quorum leader for the map to be updated

Managers

- The MGRs are integrated with the MONs, and collect the statistics within the cluster.
- The MGRs provide a pluggable Python framework to extend the capabilities of the cluster.
- The following list provides some of the existing MGR modules that are available:
 - Balancer module (dynamically reassign PGs to OSDs for better data distribution).
 - Auto-scaler module (dynamically adjust the number of PGs that are assigned to a pool).
 - Dashboard module (provide a UI to monitor and manage the Ceph cluster).
 - RESTful module (provide a RESTful API for cluster management).
 - Prometheus module (provide metrics support for the Ceph cluster)

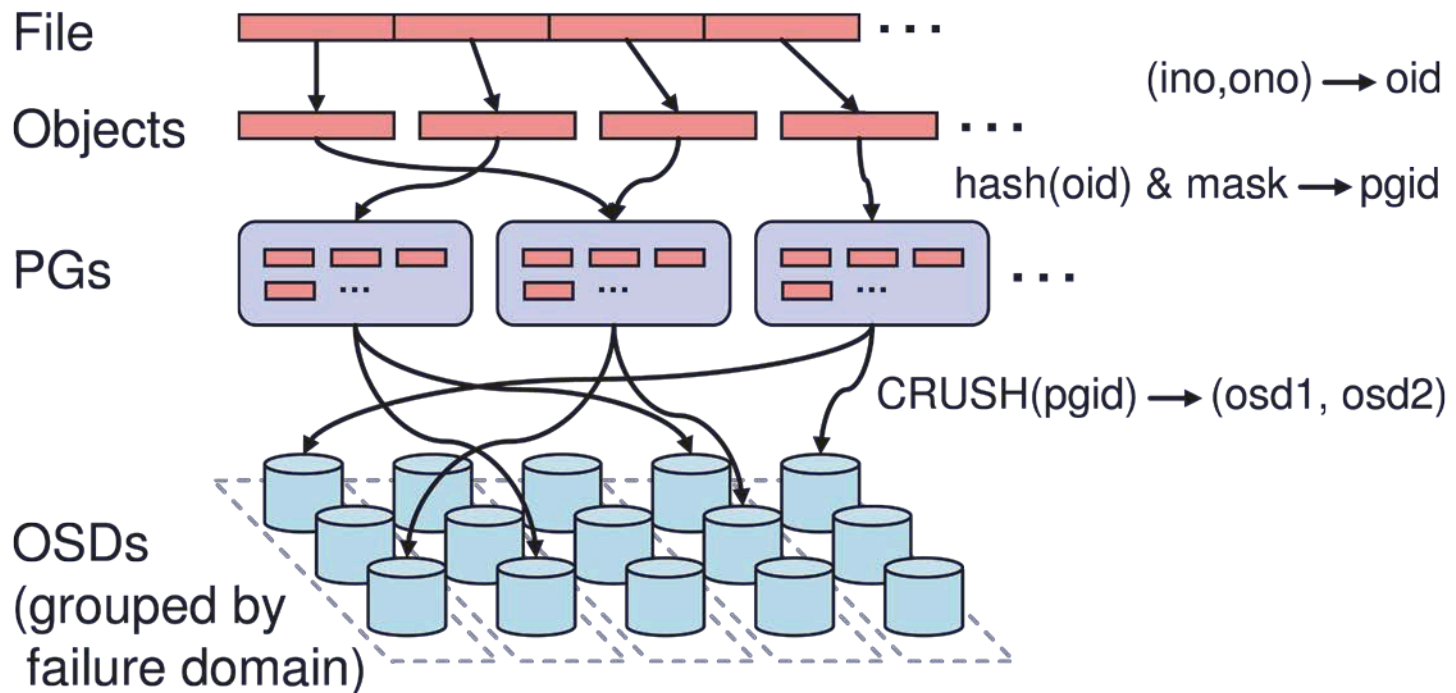
Object storage daemons (OSD)

- this component of the RADOS cluster is responsible for storing the data in RADOS and for serving the I/O requests that originate from the Ceph cluster clients.
- OSDs are responsible for the following tasks:
 - Serve I/O requests.
 - Protect the data (replication or erasure coding (EC) model).
 - Recover the data after failure.
 - Rebalance the data on cluster expansion or reduction.
 - Check the consistency of the data (scrubbing).
 - Check for bit rot detection (deep scrubbing).

Distributed Object Storage

- Files are split across objects
- Objects are members of placement groups
- Placement groups (PG) are distributed across OSDs.
- CRUSH (Controlled Replication Under Scalable Hashing) algorithm takes care of distributing objects and uses rules to determine the mapping of the PGs to the OSDs.

Distributed Object Storage



CRUSH

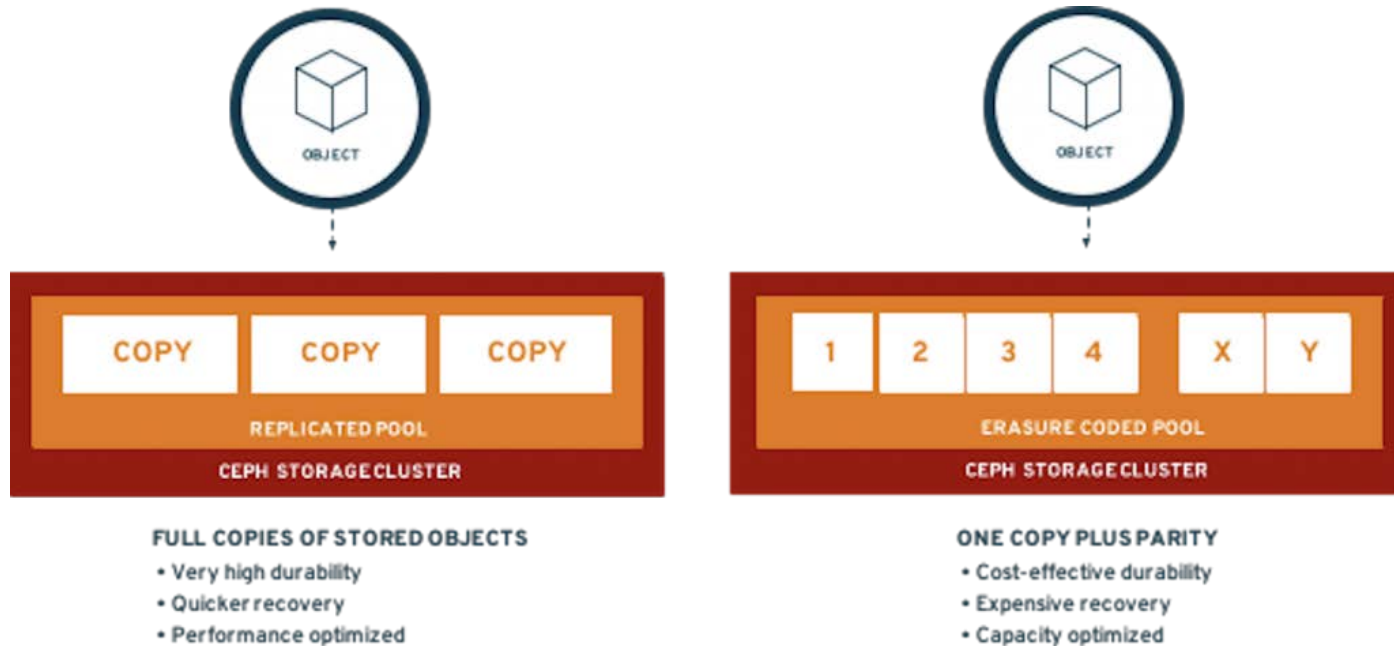
- CRUSH(x) -> (osdn1, osdn2, osdn3)
 - Inputs
 - x is the placement group
 - Hierarchical cluster map
 - Placement rules
 - Outputs a list of OSDs
- Advantages
 - Anyone can calculate object location
 - Cluster map infrequently updated

Cluster partitions

- The CEPH cluster is separated into logical partitions, known as pools. Each pool has the following properties that can be adjusted:
 - An ID (immutable)
 - A name
 - A number of PGs to distribute the objects across the OSDs
 - A CRUSH rule to determine the mapping of the PGs for this pool
 - Parameters associated with the type of protection
 - Number of copies for replicated pools
 - K and M chunks for Erasure Coding

Data protection

- Support two types: redundancy and erasure code



Erasure code vs replication

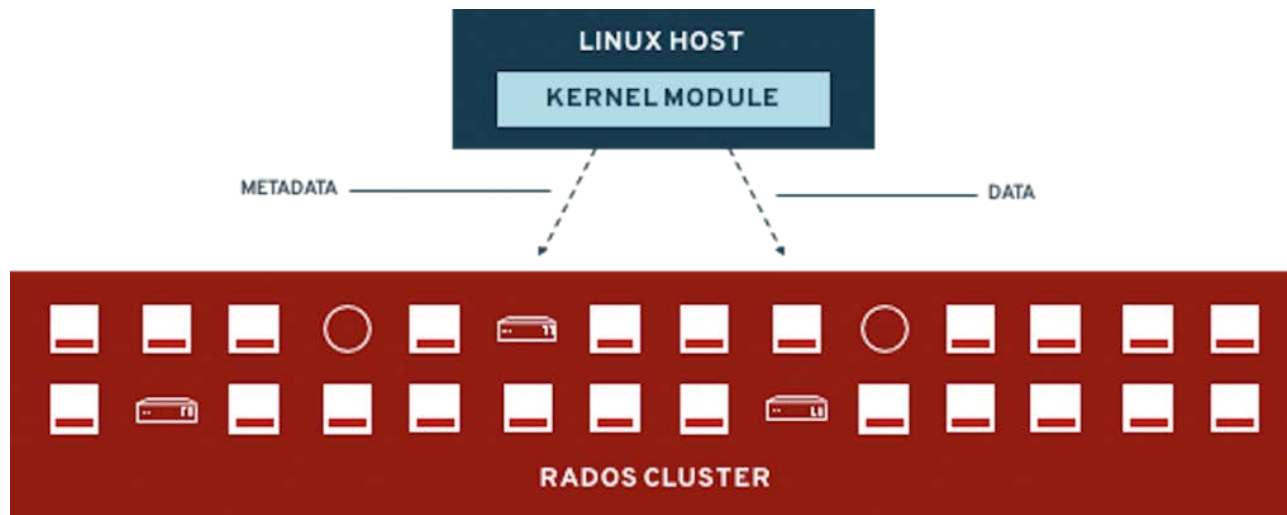
- Replicated pools provide better performance in **almost** all cases at the cost of a lower usable to raw storage ratio (1 usable byte is stored using 3 bytes of raw storage by default)
- Erasure Coding provides a cost-efficient way to store data with less performance.
- Standard Erasure Coding profiles
 - 4+2 (1:1.666 ratio)
 - 8+3 (1:1.375 ratio)
 - 8+4 (1:1.666 ratio)

Benefits of the two strategies

- Replicated model:
 - High durability with three copies
 - Quicker recovery
 - Performance optimized
- EC model:
 - Cost-effective durability with the multiple coding chunks
 - More expensive recovery
 - Capacity optimized

CEPHfs

- clients access a shared POSIX compliant filesystem.



Client access example:

1. Client sends *open* request to MDS
2. MDS returns capability, file inode, file size and stripe information
3. Client read/write directly from/to OSDs
4. MDS manages the capability
5. Client sends *close* request, relinquishes capability, provides details to MDS

CEPHfs and MDS

- MDS is the specific component required to maintain ACLs, ownership, and permissions for files and directories while mapping POSIX inode numbers to RADOS object names.
- CephFS stores both the data and the metadata (inodes and dentries) for the file system within RADOS pools.
- Using RADOS as a storage layer enables Ceph to leverage built-in RADOS object features such as watch and notify to easily implement an active-active or active-passive mechanism and a highly efficient online file system check mechanism.

MDS

- Active Metadata Server
 - An active MDS responds to CephFS client requests to provide the client with the exact RADOS object name and its metadata for an i-node number while maintaining a cache of all accesses to the metadata.
 - When a new inode or dentry is created, updated, or deleted, the cache is updated and the inode or dentry is recorded, updated, or removed in a RADOS pool that is dedicated to CephFS metadata.
- Standby Metadata Server
 - When the standby MDS becomes active, it replays the journal to reach a consistent state.
 - If the active MDS becomes inactive or responds to an administrative command that puts it in an inactive state, a standby MDS becomes the active MDS for a CephFS.
- Metadata Server journaling
 - Every action on the CephFS metadata is streamed into a journal that is shared among MDSs and located in the CephFS metadata pool before the file system operation is committed.
 - This journal is used to maintain the consistency of the file system during an MDS failover operation because events can be replayed by the standby MDS to reach a file system state that is consistent with the state that was last reached by the now defunct, previously active MDS.
 - They are handled faster by all types of physical disk drives, and sequential consecutive writes can be merged for even better performance
 - The performance of the metadata pool where the journal is maintained is of vital importance to the level of performance that is delivered by a CephFS subsystem
 - a best practice to use flash device-based OSDs to host the placement groups (PGs) of the metadata pool.
 - The journal is composed of multiple RADOS objects in the metadata pool, and journals are striped across multiple objects for better performance.
 - Each active MDS maintains its journal for performance and resiliency reasons. Old journal entries are automatically trimmed by the active MDS.

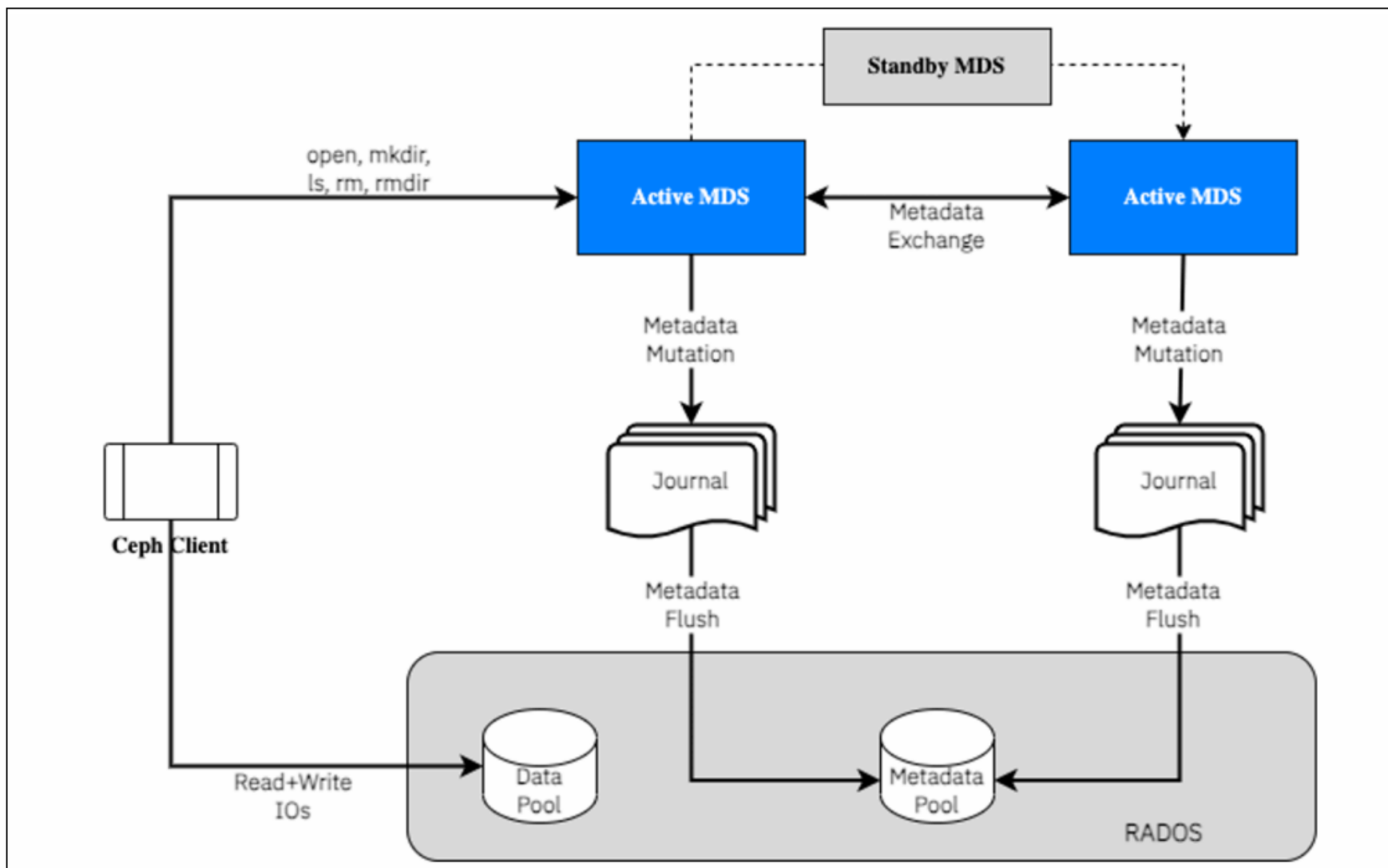


Figure 3-29 Ceph File System pools and components

CEPHfs

- Ceph supports one or more file systems in a single storage cluster.
- Creating multiple file systems is aligned with the CRUSH customization mapping specific hardware capabilities or layout:
 - Lightning-fast file system
 - Fast file system
 - Home directory file system
 - Archival file system

Synchronization


- Adheres to POSIX
- Includes HPC oriented extensions
 - Consistency / correctness by default
 - Optionally relax constraints via extensions
 - Extensions for both data and metadata
- Synchronous I/O used with multiple writers or mix of readers and writers

[Home](#) > [High Performance Computing](#) > Conference paper


An I/O Analysis of HPC Workloads on CephFS and Lustre

Conference paper | First Online: 03 December 2019

pp 300–316 | [Cite this conference paper](#)

[Alberto Chiusole](#), [Stefano Cozzini](#) , [Daniel van der Ster](#), [Massimo Lamanna](#) & [Graziano Giuliani](#)

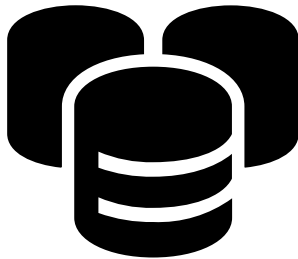
 Part of the book series: [Lecture Notes in Computer Science](#) ((LNTCS, volume 11887))

 Included in the following conference series:
[International Conference on High Performance Computing](#)

ORFEO's CEPH cluster[s]

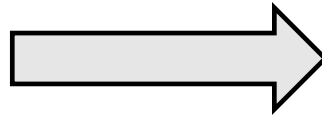
Since we are updating the CEPH version, we have 2 different CEPH clusters.

Ceph 14.2.22
(*nautilus*)
OLD

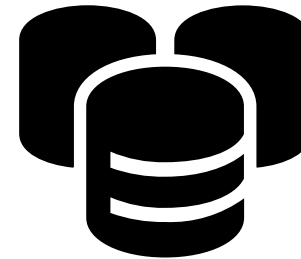


Current status

- 4 nodes equipped with SSD and NVME OSDs.
- They serve **/fast** and **/kubefs** filesystem.



Ceph 17.2.3
(*quincy*)
NEW



Current status

- 8 nodes equipped with HDD and NVME OSDs.
- They serve the **/cephfs**. Filesystem composed by 2 pool.

ORFEO's old **CEPH** cluster.

It is composed by 4 nodes with a total raw capacity of 320 TB.

It currently serve the `/fast` filesystem mounted on the computational nodes:

```
[root@login01 ~]# mount | grep fast
10.128.6.211:6789,10.128.6.212:6789,10.128.6.213:6789,10.128.6.214:6789:/ on /fast
```

Hint: use `mount` and `df` to explore the file system.

ORFEO's new **CEPH** cluster.

It is composed by 8 nodes with a total capacity of 1.6 PB.

It currently serve the *cephfs* filesystem mounted on the computational nodes:

```
[root@login01 ~]# mount | grep cephfs  
10.128.6.227:6789,10.128.6.228:6789:/ on /orfeo/cephfs
```

`/orfeo/cephfs` serves the **home** and the **scratch** folders.
Each folder has a dedicated pool.

How to manage a migration

- **Step 0:** Move the data. To migrate the data we will stop the services and move the data (*/fast* and */kubefs*) to the new cluster.

How to manage a migration

- **Step 0:** Move the data. To migrate the data we will stop the services and move the data (*/fast* and */kubefs*) to the new cluster.



Problem: the new cluster has only mechanical HDD, and the SSD are on the old cluster, how can we handle this?

How to manage a migration

- **Step 0:** Move the data. To migrate the data we will stop the services and move the data (*/fast* and */kubefs*) to the new cluster.



Problem: the new cluster has only mechanical HDD, and the SSD are on the old cluster, how can we handle this?

- **Step 1:** Bind */fast* and */kubefs* to a **dedicated** pool. Decommission the old cluster and add the 4 old nodes to the new cluster.

How to manage a migration

- **Step 0:** Move the data. To migrate the data we will stop the services and move the data (*/fast* and */kubefs*) to the new cluster.



Problem: the new cluster has only mechanical HDD, and the SSD are on the old cluster, how can we handle this?

- **Step 1:** Bind */fast* and */kubefs* to dedicated pool. Decommission the old cluster and add the 4 old nodes to the new cluster.
- **Step 2:** Change the CRUSH rule of */fast* and */kubefs* pools specifying the binding to OSDs labeled as SSD. Ceph will manage the data movement, users will not experience any downtime.

ORFEO's new FS

</

One single filesystem with 3 different pools:




- *fsmain_data_home*
- *fsmain_data_scratch*
- *fsmain_metadata*

*CEPH pool definition:
"Pools are logical
partitions that are used
to store objects."*

ORFEO's new CEPH pools

Name 	Data Protection 	Applications 	PG Status 	Usage 
fsmain_metadata	replica: x2	cephfs	32 active+clean	<div><div></div></div> 3.03%
fsmain_data_scratch	EC: 6+2	cephfs	2044 active+clean, 4 active+clean+scrubbing+	<div><div></div></div> 14.66%
fsmain_data_home	replica: x3	cephfs	255 active+clean, 1 active+clean+scrubbing+	<div><div></div></div> 3.5%

Metadata pool detail

	Name 	Data Protection 	Applications 
▼	fsmain_metadata	replica: ×2	cephfs
<div>DetailsPerformance DetailsConfiguration</div>			
application_metadata		cephfs	
auid		0	
cache_min_evict_age		0	
cache_min_flush_age		0	
cache_mode		none	
cache_target_dirty_high_ratio_micro		600000	
cache_target_dirty_ratio_micro		400000	
cache_target_full_ratio_micro		800000	
create_time		2023-06-01T11:14:33.804246+0000	
crush_rule		replicated_nvme	
data_protection		replica: ×2	

Scratch pool detail

▼	fsmain_data_scratch	EC: 6+2	cephfs	2044 active+clean, 4 active+clean+scrubbing-
---	---------------------	---------	--------	--

Details

Performance Details

application_metadata	cephfs
aud	0
cache_min_evict_age	0
cache_min_flush_age	0
cache_mode	none
cache_target_dirty_high_ratio_micro	600000
cache_target_dirty_ratio_micro	400000
cache_target_full_ratio_micro	800000
create_time	2023-06-01T10:54:51.603198+0000
crush_rule	scratch_rule
data_protection	EC: 6+2

CRUSH rules details

SCRATCH RULE

```
{
  "rule_id": 1,
  "rule_name": "scratch_rule",
  "type": 3,
  "steps": [
    {
      "op": "set_chooseleaf_tries",
      "num": 5
    },
    {
      "op": "set_choose_tries",
      "num": 100
    },
    {
      "op": "take",
      "item": -2,
      "item_name": "default~hdd"
    },
    {
      "op": "chooseleaf_indep",
      "num": 0,
      "type": "host"
    },
    {
      "op": "emit"
    }
  ]
},
```

NVME RULE

```
{
  "rule_id": 3,
  "rule_name": "replicated_nvme",
  "type": 1,
  "steps": [
    {
      "op": "take",
      "item": -9,
      "item_name": "default~nvme"
    },
    {
      "op": "chooseleaf_firstn",
      "num": 0,
      "type": "host"
    },
    {
      "op": "emit"
    }
  ]
},
```

ORFEO CEPH summary

- */orfeo/cephs*

- 144 OSDs for 1.6 PB raw capacity

- Scratch pool: Erasure code 6+2 ->(1:1.25 ratio)

- Home pool: replica 3 -> (1:3 ratio)

- */fast*

- 80 OSDs for 320 TB raw capacity

- Replication: 3 copies each object ~ 100TB useful size