DATA SCIENCE & SCIENTIFIC COMPUTING

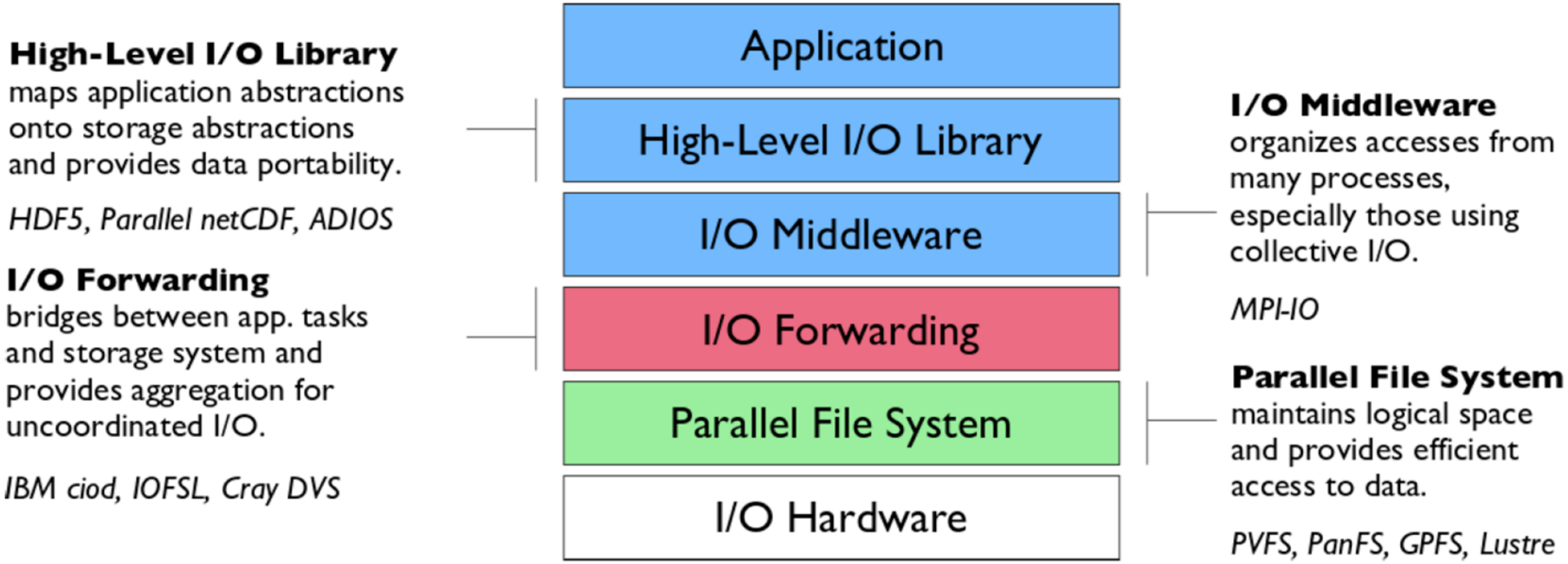
Lecture 11: storage for HPC systems: part 2

Stefano Cozzini
AreaSciencePark

Agenda of this lecture (part 2)

- HPC I/O system
- Parallel FS
- CEPH fs
- ORFEO storage
- Benchmarking I/O storage on ORFEO…

Software/Hardware stack for I/O



I/O middleware

- Match the programming model (e.g. MPI)
  - Facilitate concurrent access by groups of processes
  - Collective I/O
  - Atomicity rules
- Expose a generic interface
- Good building block for high-level libraries
- Efficiently map middleware operations into PFS ones
- Leverage any rich PFS access constructs, such as
  - Scalable file name resolution
  - Rich I/O descriptions

Overview of MPI I/O

- I/O interface specification for use in MPI apps
- Available in MPI-2.0 standard on
- Data model is a stream of bytes in a file
- Same as POSIX and stdio
- Features:
  - Noncontiguous I/O with MPI datatypes and file views
  - Collective I/O
  - Nonblocking I/O
- Fortran/C bindings (and additional languages)
- API has a large number of routines..

NOTE: you simply compile and link as you would any normal MPI program.

Why MPI is good for I/O ?

- Writing is like sending a message and reading is like receiving one.
- Any parallel I/O system will need to
  - define collective operations (*MPI communicators*)
  - define noncontiguous data layout in memory and file (*MPI datatypes*)
  - Test completion of nonblocking operations (*MPI request objects*)
- i.e., lots of MPI-like machinery needed

NOTE: you simply compile and link as you would any normal MPI program.

Parallel I/O using MPI ?

- Why do I/O in MPI?
- Why not just POSIX?
  - Parallel performance
  - Single file (instead of one file / process)
- MPI has replacement functions for POSIX I/O
- Multiple styles of I/O can all be expressed in MPI
  - Contigous vs non contiguous etc….

Building blocks for HPC I/O system

- A HPC I/O system should:
  - Present storage as a single, logical storage unit
    - (We do not want to look for different storage on different nodes)
  - Tolerate failures (in conjunction with other HW/SW)
    - (We do not want to stop production when a disk/server/inc card) breaks)
  - Provide a standard interface: (i.e. Posix compliant)
    - We do not want to change your code when you use an HPC
  - Stripe files across disks and nodes for performance
    - We do want to get parallel performance on parallel system

HPC I/O system

- HW:
  - Disks/ disk enclosure/ disk controllers
  - Server,
  - Networks etc..etc..
- Software:
  - distribute/parallel Filesystem,
  - libraries
  - some parts of O.S.

Scaling the Filesystem..

- Original POSIX environment was unshared, direct-attached storage
- RAID and Volume Managers aggregate devices safely

→ Scale performance within the same machine

- Distributed FS introduces a Network (Ethernet) between clients and server

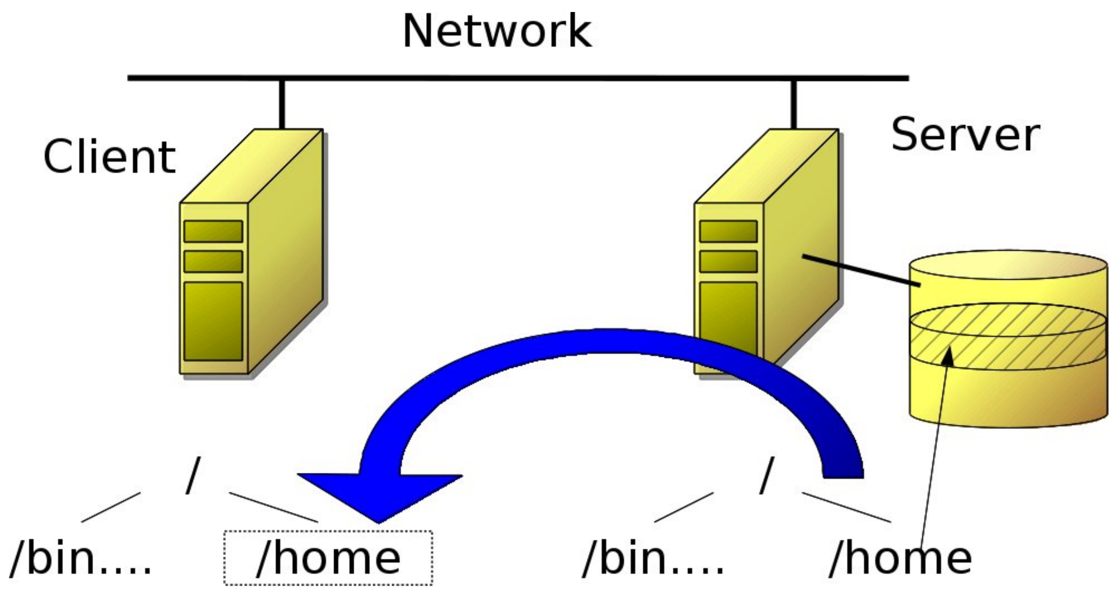
→ Able to coordinate access from multiple clients: scales over many client

- Parallel FS coordinates many clients and many servers
  - A special kind of networked file system that provides high-performance I/O when multiple clients share the file system

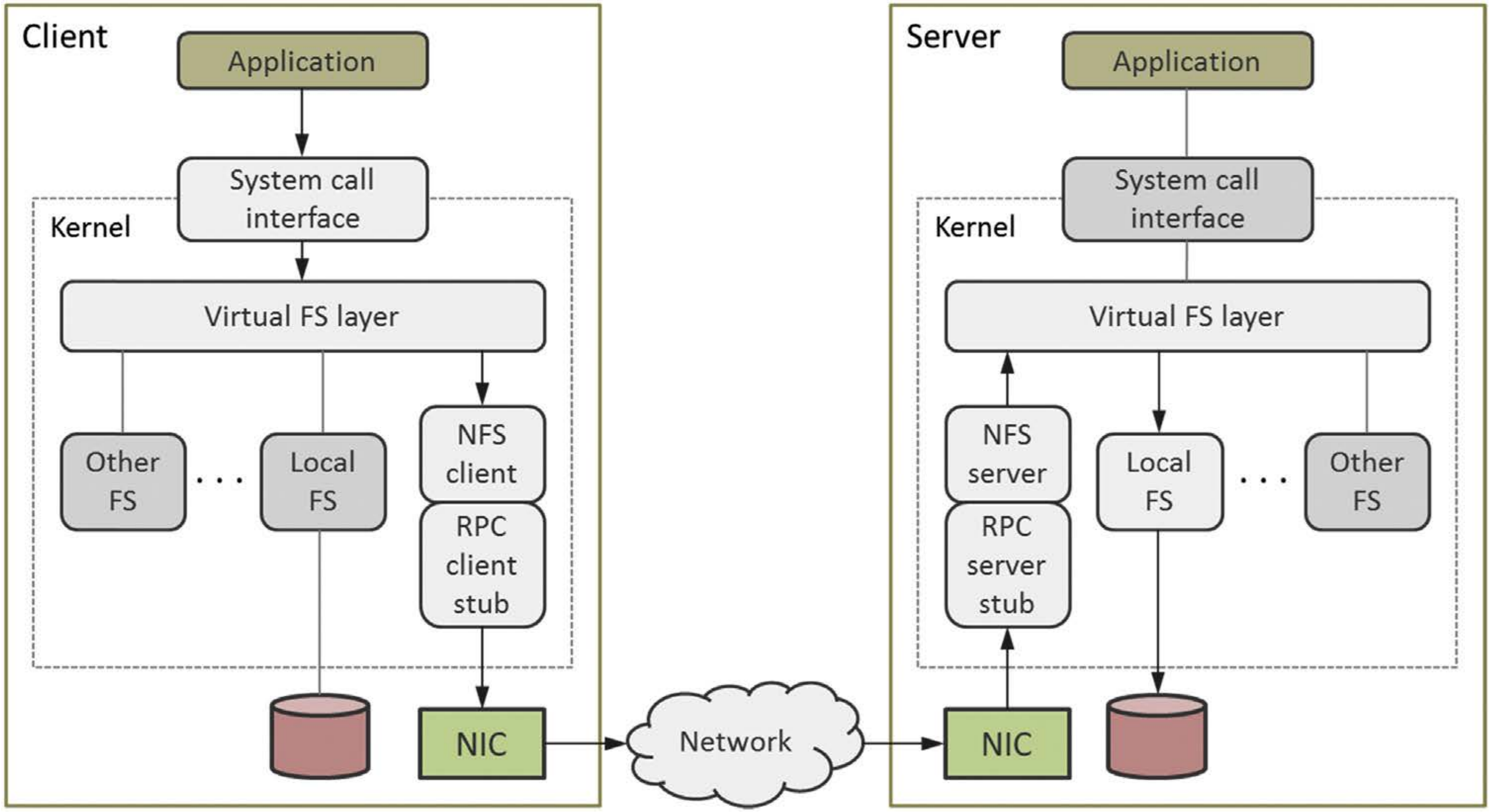
→ Able to scale in both capacity and performance

Distributed file system

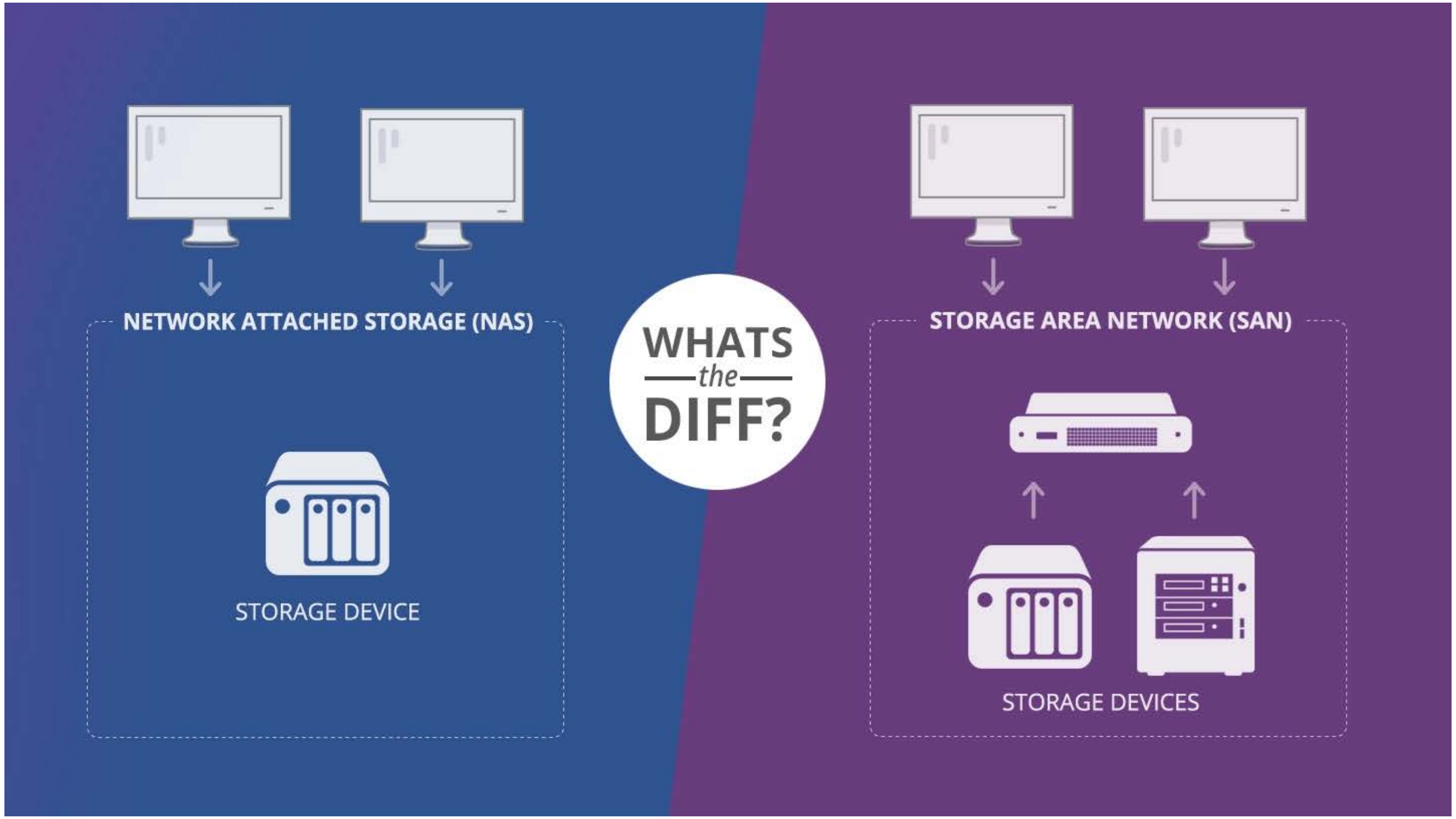
- Distributed file systems are file systems that are capable of handling I/O requests issued by multiple clients over the network.
- FS is “mounted” by several clients (compute nodes/login nodes..)
  - Example: Network File System
- Parallel access is possible but:
  - Network bandwidth limited..
- Locking issues



NFS architecture



SAN vs NAS..



Picture from: https://www.backblaze.com/blog/whats-the-diff-nas-vs-san/

SAN vs NAS

- SAN
  - Block level data access
  - Fiber channel is the primary media used with SAN.
  - SCSI is the main I/O protocol
  - SAN storage appears to the computer as its own storage
- NAS:
  - File Level Data access
  - Ethernet is the primary media used with NAS
  - NFS is used as the main I/O protocol in NAS
  - appears as a shared partition to the computer

SAN vs NAS

- SAN
- Block level data access
- Fiber channel is the primary media used with SAN.
- SCSI is the main I/O protocol
- SAN storage appears to the computer as its own storage

NAS

- NAS
- File Level Data access
- Ethernet is the primary media used with NAS
- NFS is used as the main I/O protocol in NAS
- appears as a shared partition to the computer

Issues in building HPC I/O systems

- "Management problem": many disks/ HW around our cluster but not easy to make them available to user in a clean/safe/cheap way.
- "Performance problems" : large dataset requires high performance I/O solutions

Scalability Limitation of I/O

- I/O subsystems are typically very slow compared to other parts of a supercomputer
  → You can easily saturate the bandwidth
- Once the bandwidth is saturated scaling in I/O stops
- Adding more compute nodes increases aggregate memory bandwidth and flops/s, but not I/O

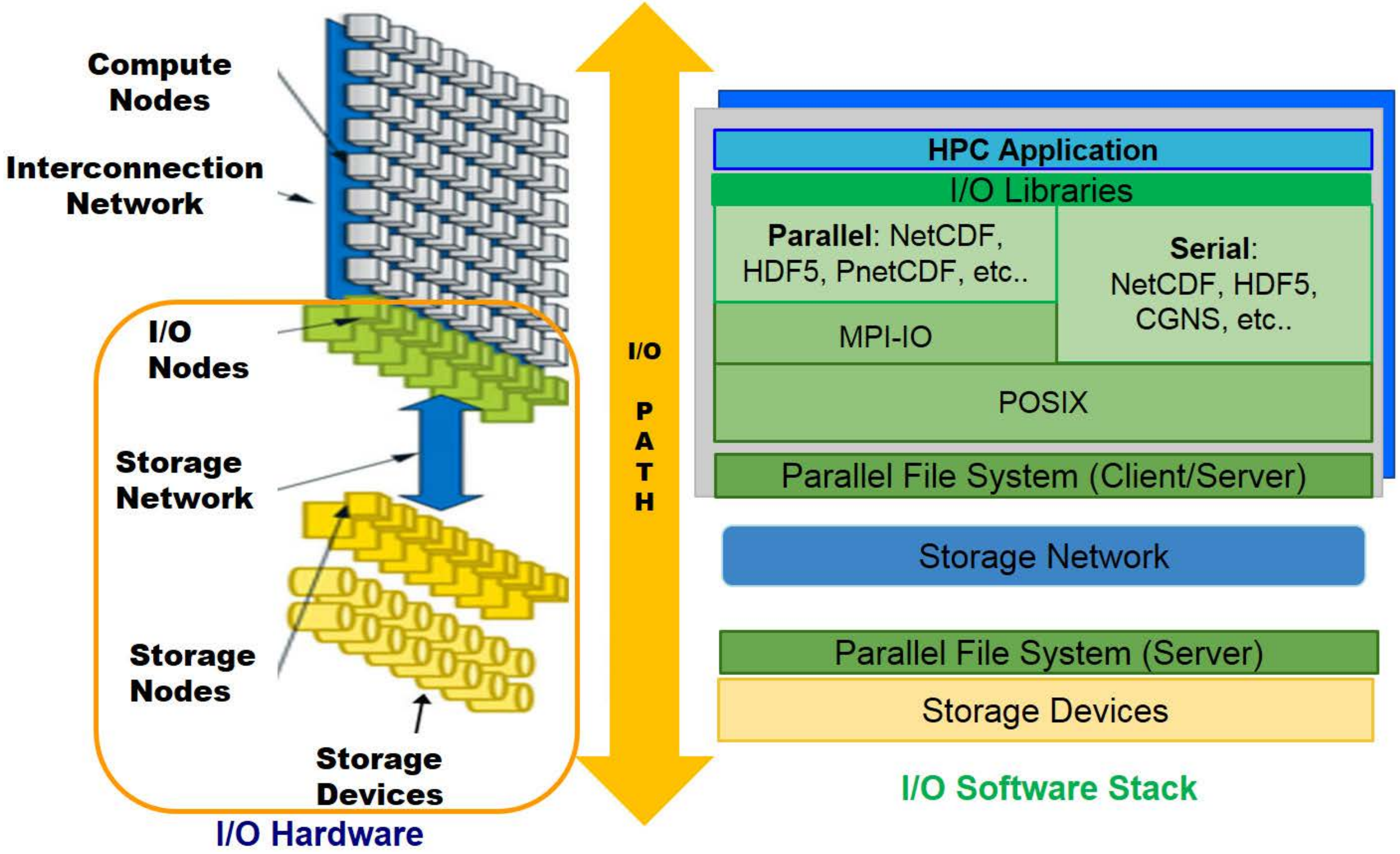
Parallel File System

Foundation of High Performance Computing 18

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

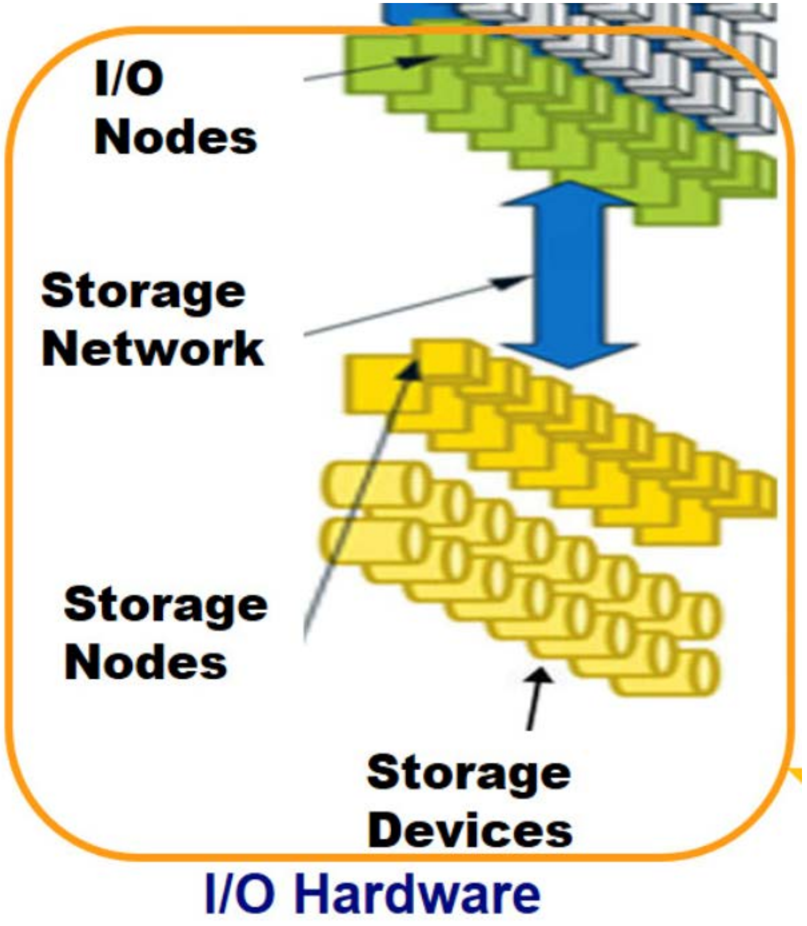
A graphical view:



Picture from: http://www.prace-ri.eu/best-practice-guide-parallel-i-o/#id-1.3.5

Parallel File System: I/O hardware

- Within ORFEO:
  - I/O nodes = Storage Nodes () = CEPH nodes
  - Storage Network= INFINIBAND network for CEPH
  - Metadata server hosted on I/O server (dedicated and/ or shared)
  - Storage nodes hosts some data:
  - Metadata server coordinates access by the clients to the data



Hardware to build a PFS:

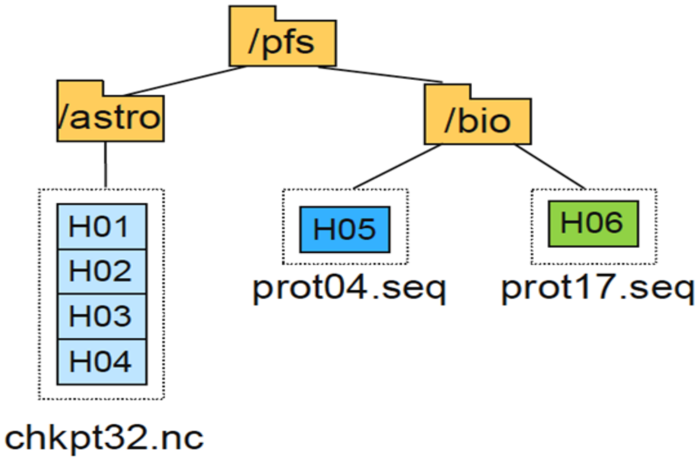
- Nodes, Disks, controllers, and interconnects
- Hardware defines the peak performance of the I/O system:
  - raw bandwidth
  - Minimum latency
- At the hardware level, data is accessed at the granularity of blocks, either physical disk blocks or logical blocks spread across multiple physical devices such as in a RAID array
- Parallel File Systems takes care of
  - managing data on the storage hardware,
  - presenting this data as a directory hierarchy,
  - coordinating access to files and directories in a consistent manner

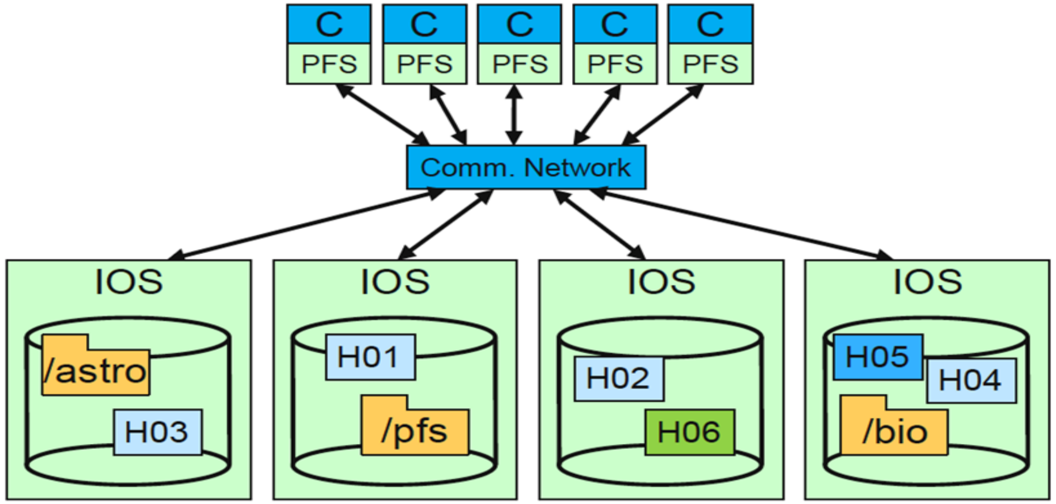
An important disclaimer..

- Parallel File Systems are usually optimized for high performance rather than general purpose use,
- Optimization criteria:
  - Large block sizes (≥ 64kB)
  - Relatively slow metadata operations (eg. fstat()) compared to reads and writes..)
  - Special APIs for direct access and additional optimizations

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 ?

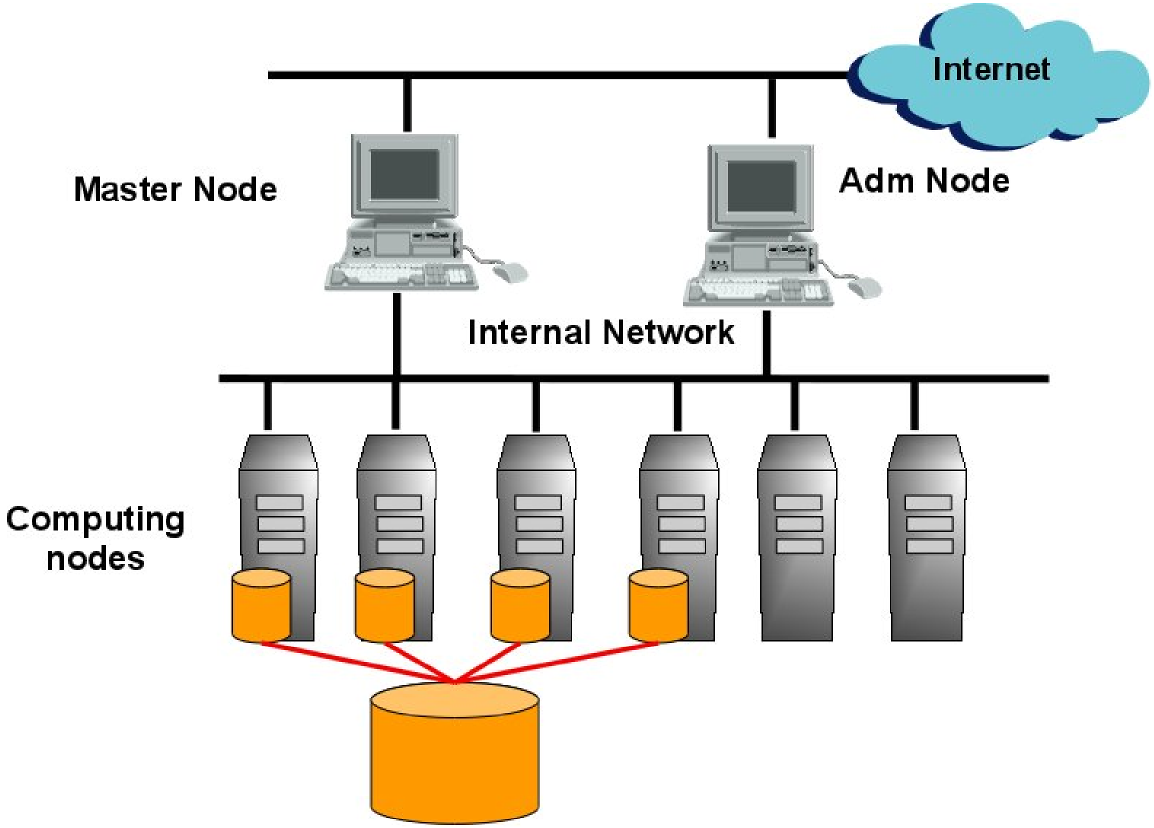
- BeeGFS
  - Developed at Fraunhofer Institute, freely available not open
  - http://www.fhgfs.com/cms/
- Lustre
  - open and Free owned by Intel DDN
  - Intel no longer sells tools to manage and support ($$$)
  - http://lustre.opensfs.org/
- GPFS (now known as Spectrum Scale)
  - IBM proprietary $$$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

ORFEO choice: CEPH

- A unique storage solution for both HPC and Cloud infrastructure
- Main Users: Bioinformatics with many files
- Open and free
- Scalable..

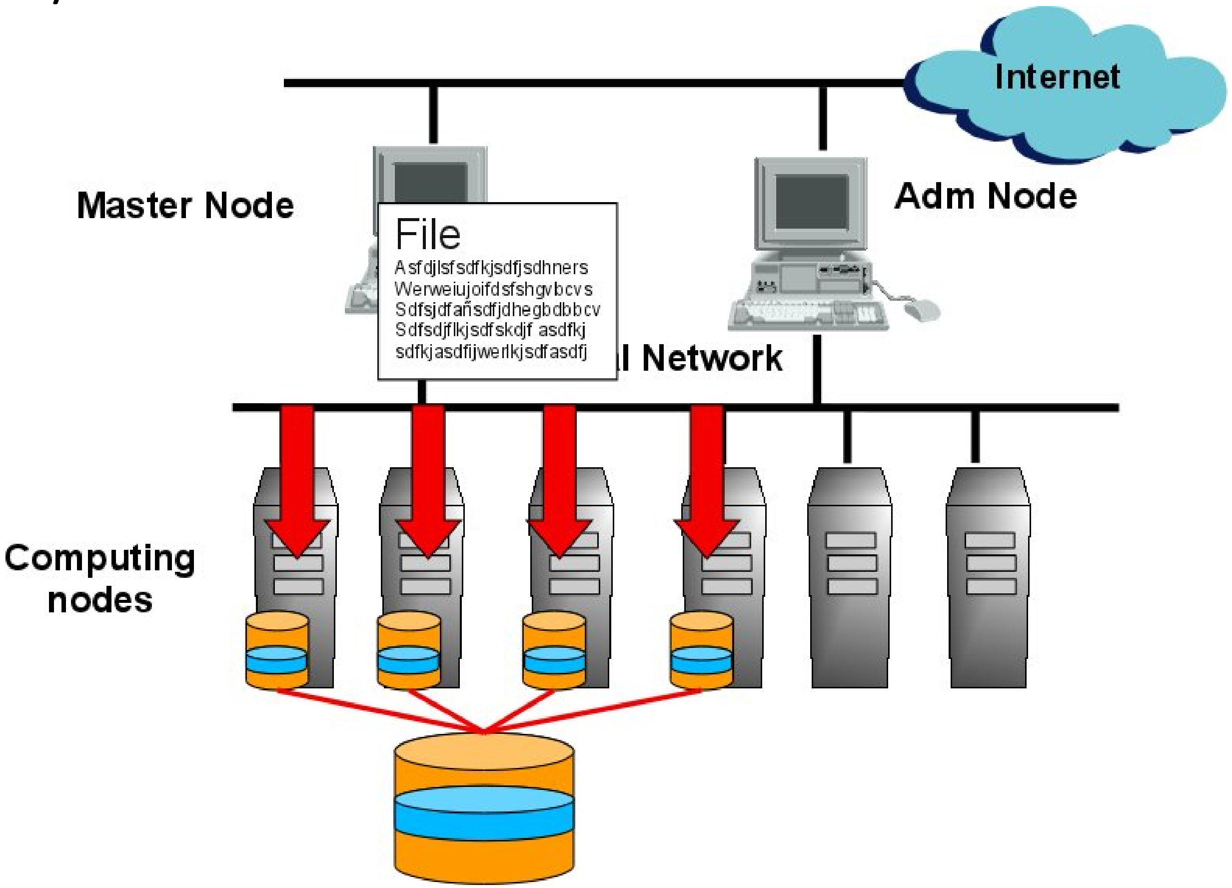
Parallel File Systems

- A parallel file system leverages all disks available across the network.



Parallel File Systems

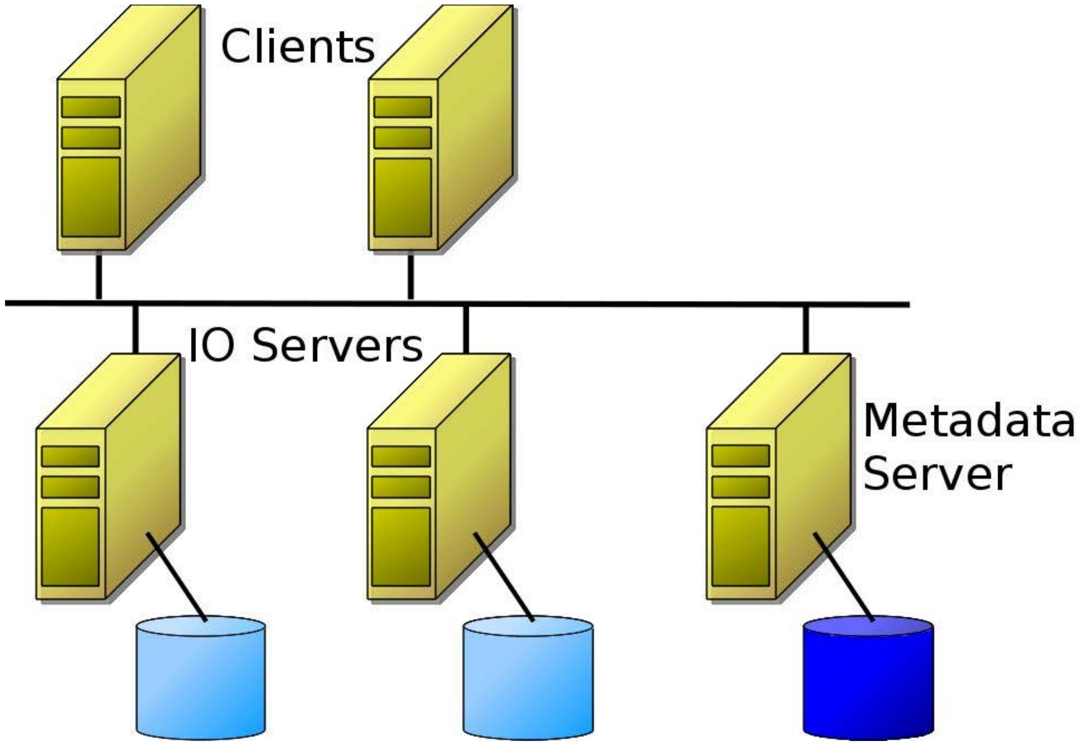
- A parallel file uses all I/O controllers at the same time.



Parallel File System: components

- In general, a Parallel File Systems has the following components

- Metadata Server
- I/O Servers
- Clients



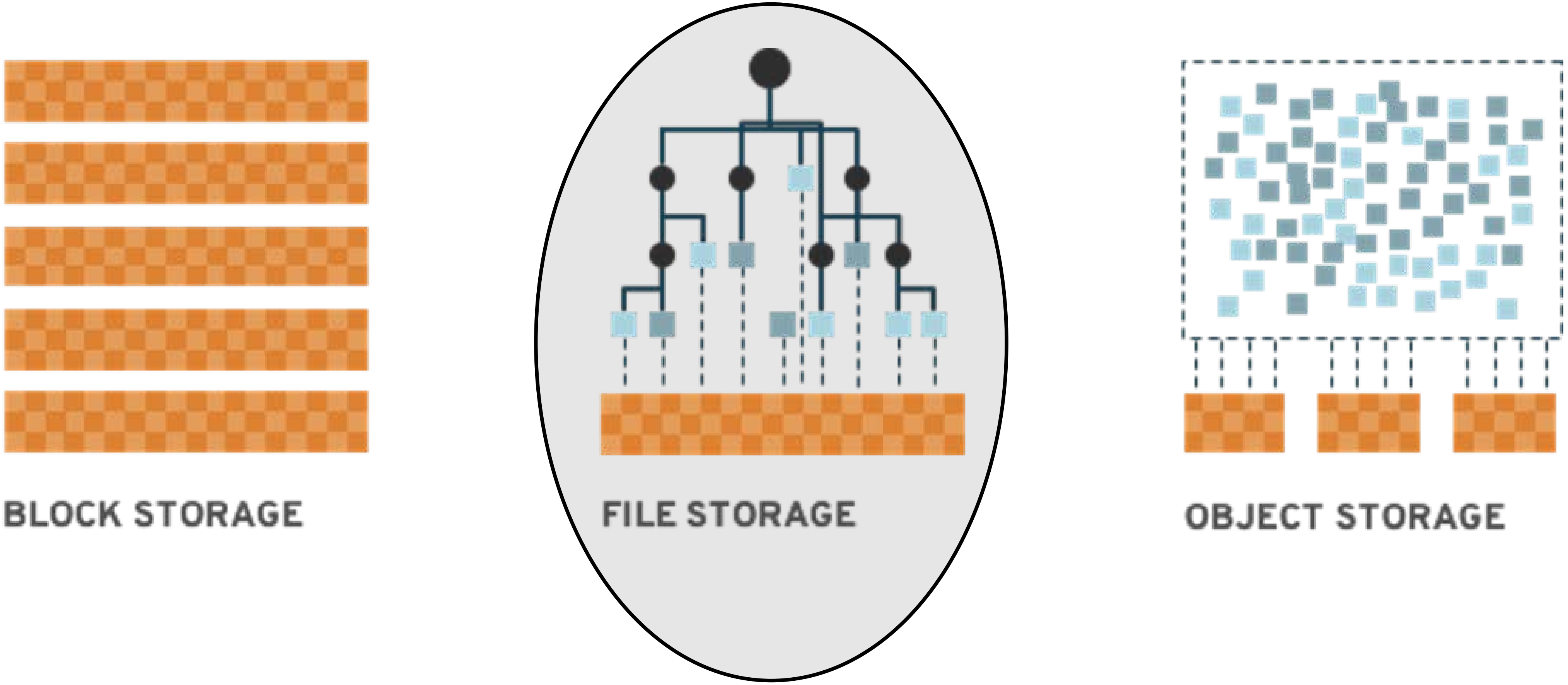
A short introduction to CEPH

Foundation of High Performance Computing

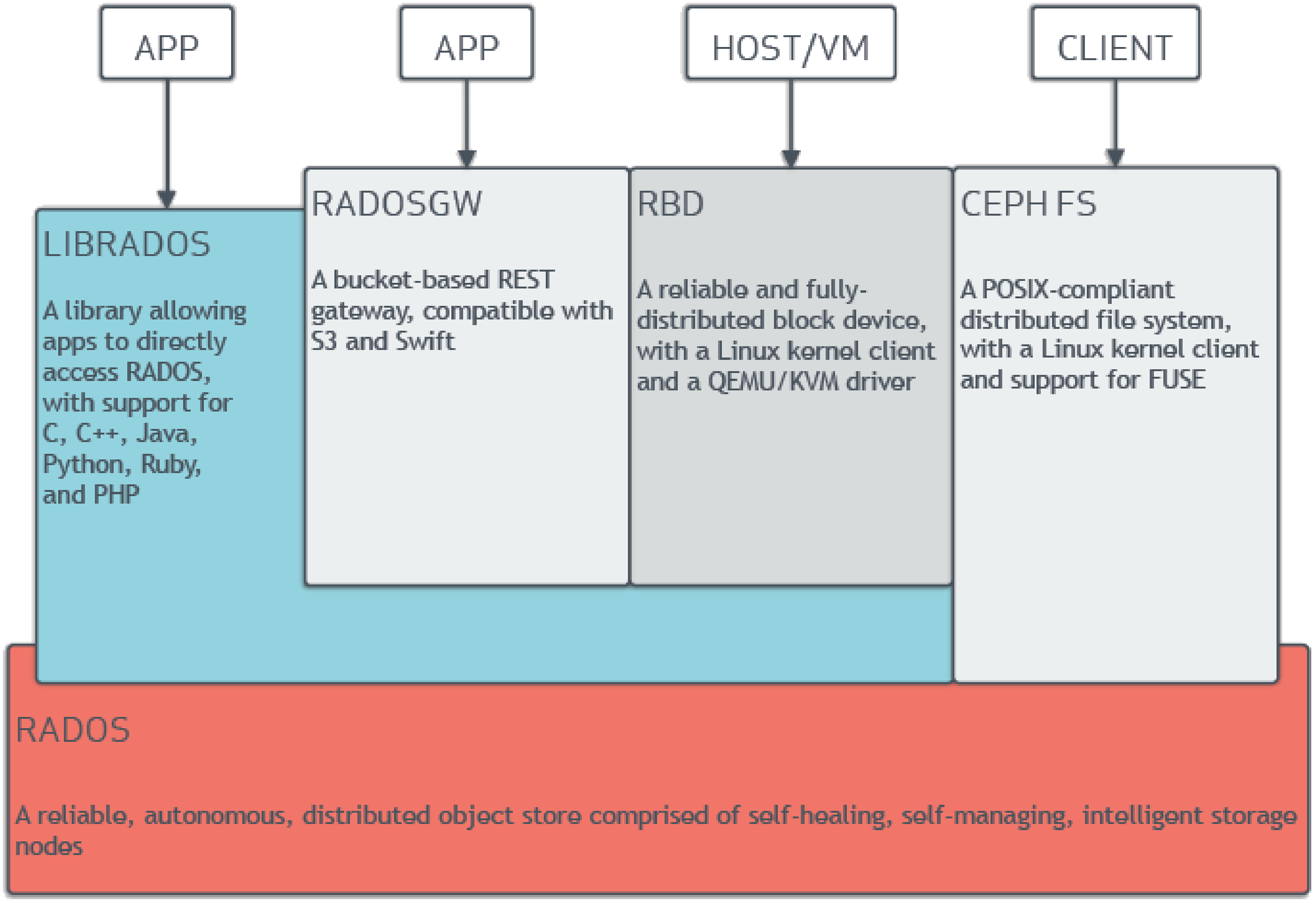
CEPH storage

- 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]

Access methods:



CEPH Storage 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 IO 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)

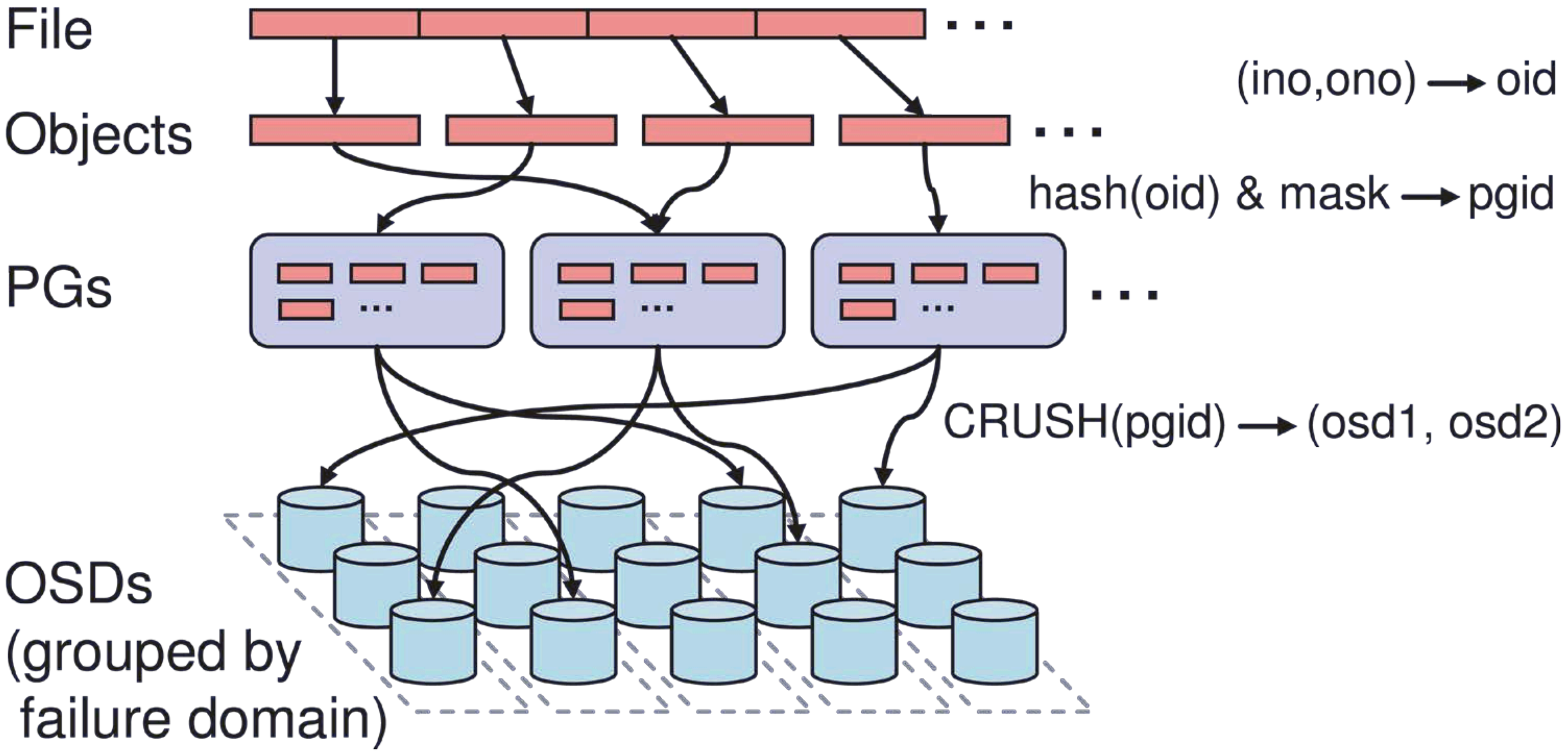
What are they doing ?

- A Ceph Monitor maintains a master copy of the cluster map. A cluster of Ceph monitors ensures high availability should a monitor daemon fail. Storage cluster clients retrieve a copy of the cluster map from the Ceph Monitor.
- A Ceph OSD Daemon checks its own state and the state of other OSDs and reports back to monitors.
- A Ceph Manager acts as an endpoint for monitoring, orchestration, and plug-in modules.
- A Ceph Metadata Server (MDS) manages file metadata when CephFS is used to provide file services.

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) algorith takes care of distributing objects and uses rules to determine the mapping of the PGs to the OSDs.

Distributed Object Storage



CRUSH

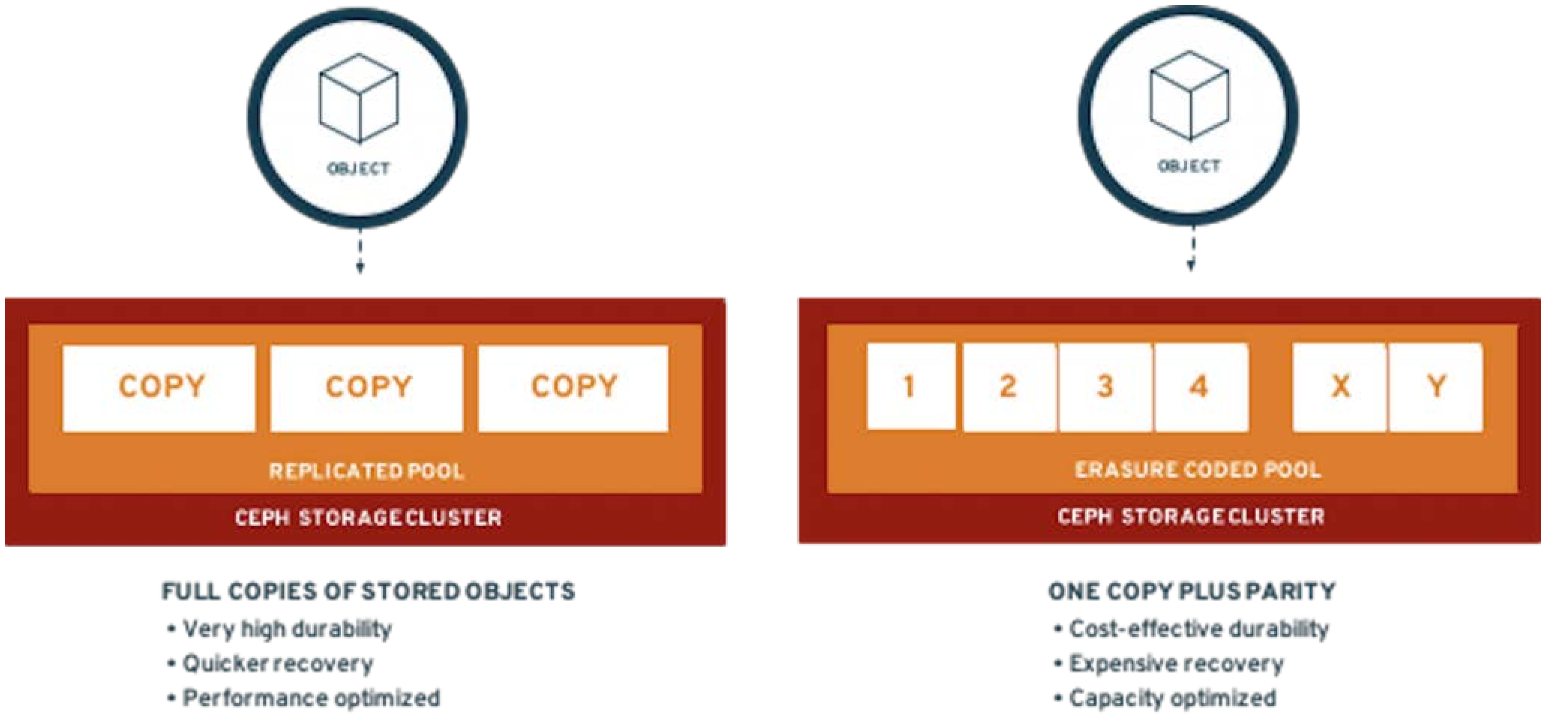
- CRUSH(x) -> (osdn1, osdn2, osdn3)
  - Inputs
    - x is the placement group
    - Hierarchical cluster map
    - Placement rules
  - Outputs a list of OSDs
- Advantages
  - Anyone can calculate object location
  - Cluster map infrequently updated

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

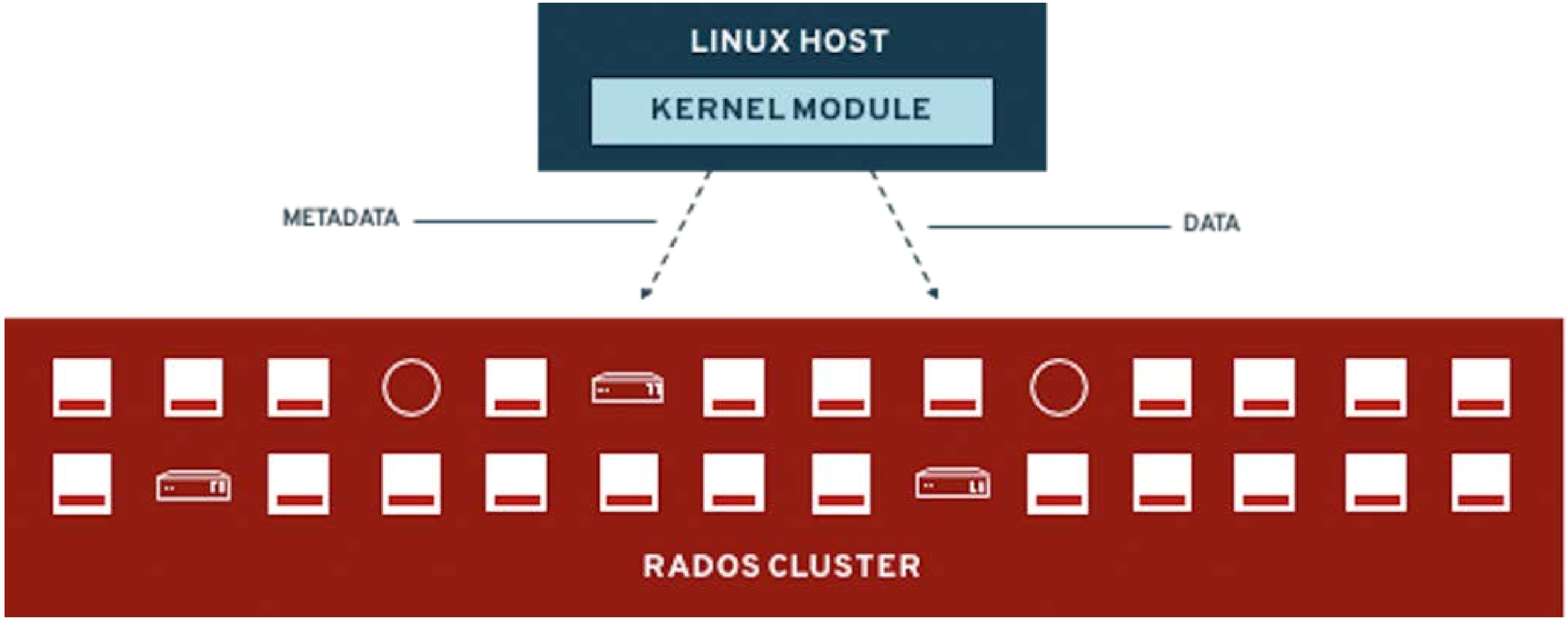


Erasure code vs replication

- 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)

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

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

ORFEO storage

Foundation of High Performance Computing 45

I/O subsystem on ORFEO:

- Home
  - once logged in, each user will land in its home in `/u/[name_of_group]/[name_of_user]
  - e.g. the home of user area is in /u/area/[name_of_users]
  - it's physically located on ceph large FS, and exported via infiniband to all the computational nodes
  - quotas are enforced with a default limit of 2TB for each users
  - soft link are available there for the other areas

```
[cozzini@login ~]$ ls -lrt
total 548398
lrwxrwxrwx 1 cozzini area        18 Apr  7  2020 fast -> /fast/area/cozzini
lrwxrwxrwx 1 cozzini area        21 Apr  7  2020 storage -> /storage/area/cozzini
lrwxrwxrwx 1 cozzini area        21 Apr 16  2020 scratch -> /scratch/area/cozzini
```

I/O subsystem on ORFEO:

- Scratch
  - it is large area intended to be used to store data that need to be elaborated
  - it is also physically located on ceph large FS, and exported via infiniband to all the computational nodes

```
[cozzini@login ~]$ df -h /scratch
Filesystem                                                              Size  Used Avail Use% Mounted on
10.128.6.211:6789,10.128.6.213:6789,10.128.6.212:6789,10.128.6.214:6789:/  598T   95T  503T  16% /large
```

- /fast
  - is a fast space available for each user, on all the computing nodes
  - is intended to be a **fast scratch area** for data intensive application

```
[cozzini@login ~] df -h /fast
Filesystem                                                              Size  Used Avail Use% Mounted on
10.128.6.211:6789,10.128.6.212:6789,10.128.6.213:6789,10.128.6.214:6789:/   88T  4.3T   83T   5% /fast
```

I/O subsystem on ORFEO:

- Long term storage:
  - it is NFS mounted via 50bit ethernet link
  - it is intended for long-term storage of final processed dataset
  - Plenty of room to be allocated..

```
[cozzini@login ~]$ df -h  | grep 231
10.128.2.231:/storage                                              37T   18T   19T  48% /storage
10.128.2.231:/illumina_run                                         46T   42T  4.1T  92%
/illumina_run
```

ORFEO storage: hardware

	FAST storage (NVMe)	FAST storage (SSD)	Standard storage (HDD)	Long term preservation
# of server	4		6	1
RAM	6 x 16GB		6 x 16GB	6 x 16GB
Disk per node	2x 1.6TB NVMe PCIe card	20 x 3.84TB	15 x 12TB	84 x 12TB + 42 x 12TB
Storage provider	CEPH parallel FS	CEPH parallel FS	CEPH parallel FS	Network FS (NFS)
RAW storage	**12TB**	**320 TB**	**1080 TB**	**1,512 TB**

ORFEO CEPH storage cluster

- 10 nodes:

Hosts List | Overall Performance

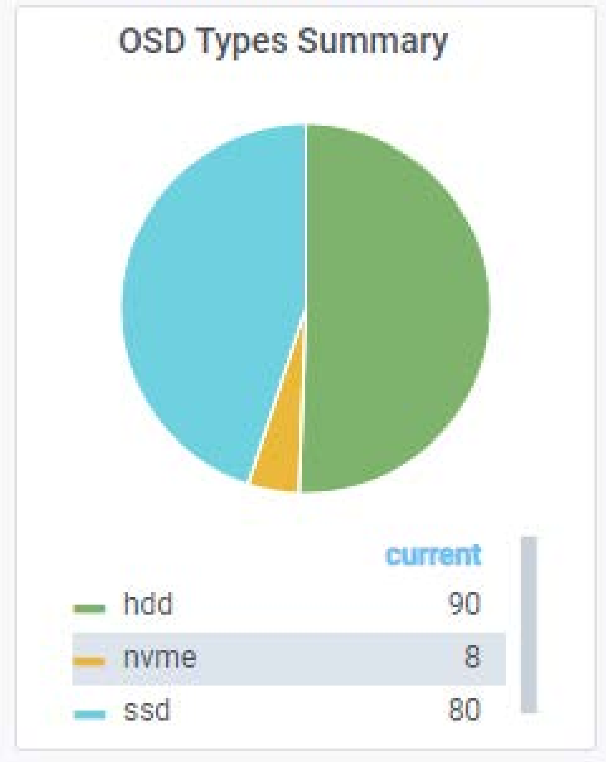
10

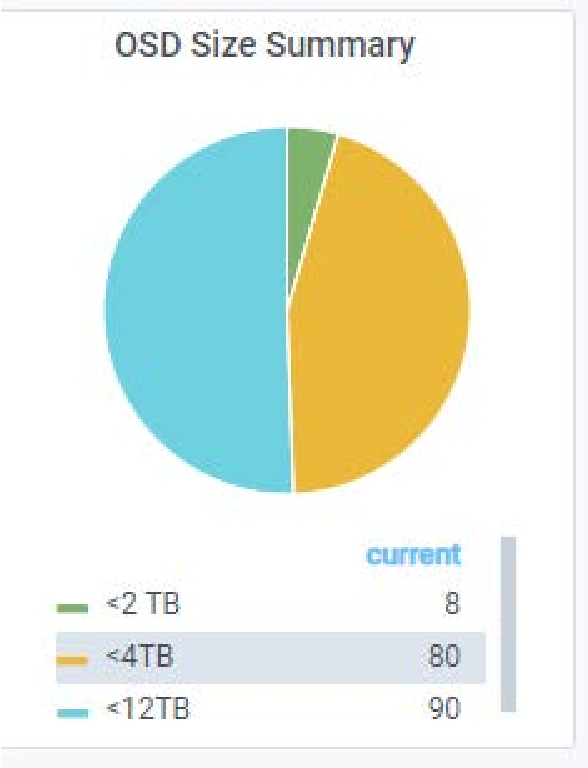
Hostname	Services
ct1ps-ceph005	osd.63 , osd.64 , osd.65 , osd.66 , osd.67 , osd.68 , osd.69 , osd.70 , osd.71 , osd.72 , osd.73 , osd.74 , osd.75 , osd.76 , osd.77
ct1ps-ceph006	osd.78 , osd.79 , osd.80 , osd.81 , osd.82 , osd.83 , osd.84 , osd.85 , osd.86 , osd.87 , osd.88 , osd.89 , osd.90 , osd.91 , osd.92
ct1ps-ceph007	osd.100 , osd.101 , osd.102 , osd.103 , osd.104 , osd.105 , osd.106 , osd.107 , osd.93 , osd.94 , osd.95 , osd.96 , osd.97 , osd.98 , osd.99
ct1ps-ceph008	osd.108 , osd.109 , osd.110 , osd.111 , osd.112 , osd.113 , osd.114 , osd.115 , osd.116 , osd.117 , osd.118 , osd.119 , osd.120 , osd.121 , osd.122
ct1ps-ceph009	osd.148 , osd.149 , osd.150 , osd.151 , osd.152 , osd.153 , osd.154 , osd.155 , osd.156 , osd.157 , osd.158 , osd.159 , osd.160 , osd.161 , osd.162
ct1ps-ceph010	osd.163 , osd.164 , osd.165 , osd.166 , osd.167 , osd.168 , osd.169 , osd.170 , osd.171 , osd.172 , osd.173 , osd.174 , osd.175 , osd.176 , osd.177
ct1ps-ceph001	mds.ct1ps-ceph001 , mgr.ct1ps-ceph001 , mon.ct1ps-ceph001 , osd.0 , osd.1 , osd.10 , osd.11 , osd.12 , osd.124 , osd.125 , osd.126 , osd.127 , osd.128 , osd.129 , osd.13 , osd.2 , osd.3 , osd.4 , osd.5 , osd.56 , osd.6 , osd.60 , osd.7 , osd.8 , osd.9 , rgw.ct1ps-ceph001
ct1ps-ceph002	mds.ct1ps-ceph002 , mgr.ct1ps-ceph002 , mon.ct1ps-ceph002 , osd.130 , osd.131 , osd.132 , osd.133 , osd.134 , osd.135 , osd.14 , osd.15 , osd.16 , osd.17 , osd.18 , osd.19 , osd.20 , osd.21 , osd.22 , osd.23 , osd.24 , osd.25 , osd.26 , osd.27 , osd.57 , osd.61 , rgw.ct1ps-ceph002
ct1ps-ceph003	mds.ct1ps-ceph003 , mgr.ct1ps-ceph003 , mon.ct1ps-ceph003 , osd.136 , osd.137 , osd.138 , osd.139 , osd.140 , osd.141 , osd.28 , osd.29 , osd.30 , osd.31 , osd.32 , osd.33 , osd.34 , osd.35 , osd.36 , osd.37 , osd.38 , osd.39 , osd.40 , osd.41 , osd.58 , osd.62 , rgw.ct1ps-ceph003
ct1ps-ceph004	mds.ct1ps-ceph004 , mgr.ct1ps-ceph004 , mon.ct1ps-ceph004 , osd.123 , osd.142 , osd.143 , osd.144 , osd.145 , osd.146 , osd.147 , osd.42 , osd.43 , osd.44 , osd.45 , osd.46 , osd.47 , osd.48 , osd.49 , osd.50 , osd.51 , osd.52 , osd.53 , osd.54 , osd.55 , osd.59 , rgw.ct1ps-ceph004

0 selected / 10 total

ORFEO CEPH storage cluster

- 178 OSDs





ORFEO CEPH Crush map

Cluster » CRUSH map

CRUSH map viewer

- default (root)
  - ct1ps-ceph005 (host)
  - ct1ps-ceph006 (host)
  - ct1ps-ceph007 (host)
  - ct1ps-ceph001 (host)
    - up osd.0 (osd)
    - up osd.1 (osd)
    - up osd.10 (osd)
    - up osd.11 (osd)
    - up osd.12 (osd)
    - up osd.124 (osd)
    - up osd.125 (osd)
    - up osd.126 (osd)
    - up osd.127 (osd)
    - up osd.128 (osd)
    - up osd.129 (osd)
    - up osd.13 (osd)
    - up osd.2 (osd)
    - up osd.3 (osd)
    - up osd.4 (osd)
    - up osd.5 (osd)
    - up osd.56 (osd)

osd.56 (osd)

crush_weight	1.454986572265625
depth	2
device_class	nvme
exists	1
id	56
primary_affinity	1
reweight	1
type_id	0

ORFEO CEPH storage cluster

- 4 monitors:

In Quorum

Name	Rank	Public Address	Open Sessions
ct1ps-ceph001	0	10.128.6.211:6789/0	
ct1ps-ceph002	1	10.128.6.212:6789/0	
ct1ps-ceph003	2	10.128.6.213:6789/0	
ct1ps-ceph004	3	10.128.6.214:6789/0	

4 total

Not In Quorum

Name	Rank	Public Address
No data to display		

0 total

ORFEO CEPH pools..

- 17 different pools

Pools List | Overall Performance

+ Create | 10

fast

large

Name	Type	Applications	PG Status	Replica Size	Erasure Coded Profile	Crush Ruleset	Usage	Read bytes	Write bytes
cephfs_data	replicated	cephfs	2048 active+clean	3		repl_ssd	17%		
cephfs_metadata	replicated	cephfs	256 active+clean	3		repl_ssd	0%		
cephfs_spin_data	erasure	cephfs	2 active+clean+scrubbing+dee 1 active+clean+scrubbing, 1021 active+clean	6	ec_hdd_4km2	cephfs_spin_data	23%		
cephfs_spin_metadata	replicated	cephfs	128 active+clean	3		repl_ssd	0%		
kub_metadata	replicated	cephfs	8 active+clean	3		repl_nvme	0%		
kub_data	replicated	cephfs	32 active+clean	3		repl_ssd	1%		
os-images	replicated	rbd	64 active+clean	3		repl_ssd	0%		
os-volumes-ssd	replicated	rbd	1024 active+clean	3		repl_ssd	0%		
os-vms	replicated	rbd	32 active+clean	3		repl_ssd	0%		
os-backups	replicated	rbd	32 active+clean	3		repl_ssd	0%		

0 selected / 17 total

‹ « 1 2 » ›

ORFEO CEPH pools..

- 17 different pools

Pools List | Overall Performance

Create | 10

Name	Type	Applications	PG Status	Replica Size	Erasure Coded Profile	Crush Ruleset	Usage	Read bytes	Write bytes
cephfs_data	replicated	cephfs	2048 active+clean	3		repl_ssd	17%		
cephfs_metadata	replicated	cephfs	256 active+clean	3		repl_ssd	0%		
cephfs_spin_data	erasure	cephfs	2 active+clean+scrubbing+dee 1 active+clean+scrubbing, 1021 active+clean	6	ec_hdd_4km2	cephfs_spin_data	23%		
cephfs_spin_metadata	replicated	cephfs	128 active+clean	3		repl_ssd	0%		
kub_metadata	replicated	cephfs	8 active+clean	3		repl_nvme	0%		
kub_data	replicated	cephfs	32 active+clean	3		repl_ssd	1%		
os-images	replicated	rbd	64 active+clean	3		repl_ssd	0%		
os-volumes-ssd	replicated	rbd	1024 active+clean	3		repl_ssd	0%		
os-vms	replicated	rbd	32 active+clean	3		repl_ssd	0%		
os-backups	replicated	rbd	32 active+clean	3		repl_ssd	0%		

0 selected / 17 total

‹ « 1 2 » ›

fast

large

ORFEO /fast and /large from CEPH

- /large
  - 90 disks (12TB each) –> 90 OSDs
  - Erasure code: 4+2 (1:1.666 ratio)
  - 1080 raw capacity →648 useful size
- /fast
  - 80 disks (4TB each) → 80 OSDs
  - Replication: 3 copies each object
  - 320TB raw capacity → 320/3 =~ 100TB useful size

ORFEO CEPH file system

- 3 different file-system

Filesystems

fast

large

Name	Created	Enabled
cephfs	2020-01-25 15:06:37.434728	true
cephfs_spin	2020-04-06 15:24:22.897778	true
kubefs	2020-05-12 16:34:13.390075	true

1 selected / 3 total

Details | Clients: 19 | Performance Details

Ranks

Rank	State	Daemon	Activity	Dentries	Inodes
0	active	ct1ps-ceph003	Reqs: 5.2 /s	1.7 M	1.5 M

1 total

Standby daemons	ct1ps-ceph004

Pools

Pool	Type	Size	Usage
cephfs_spin_data	data	652.8 TiB	23%
cephfs_spin_metadata	metadata	81.5 TiB	0%

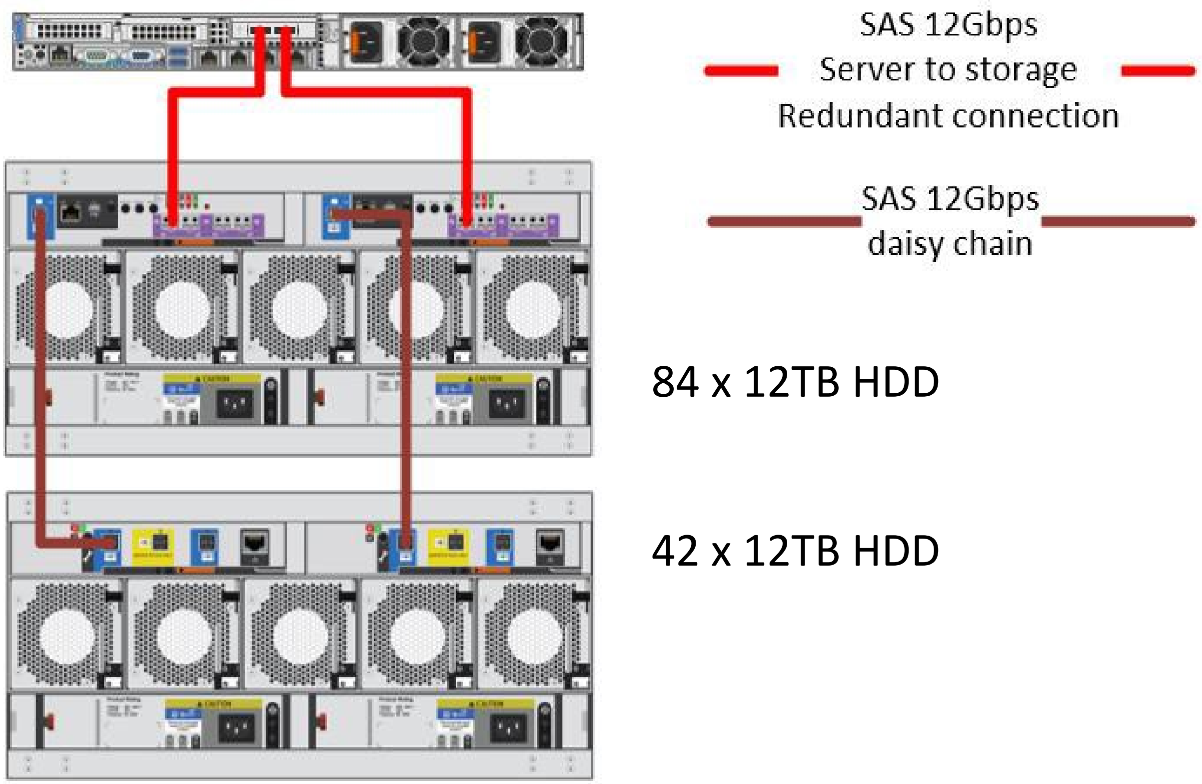
2 total

ORFEO: long term storage

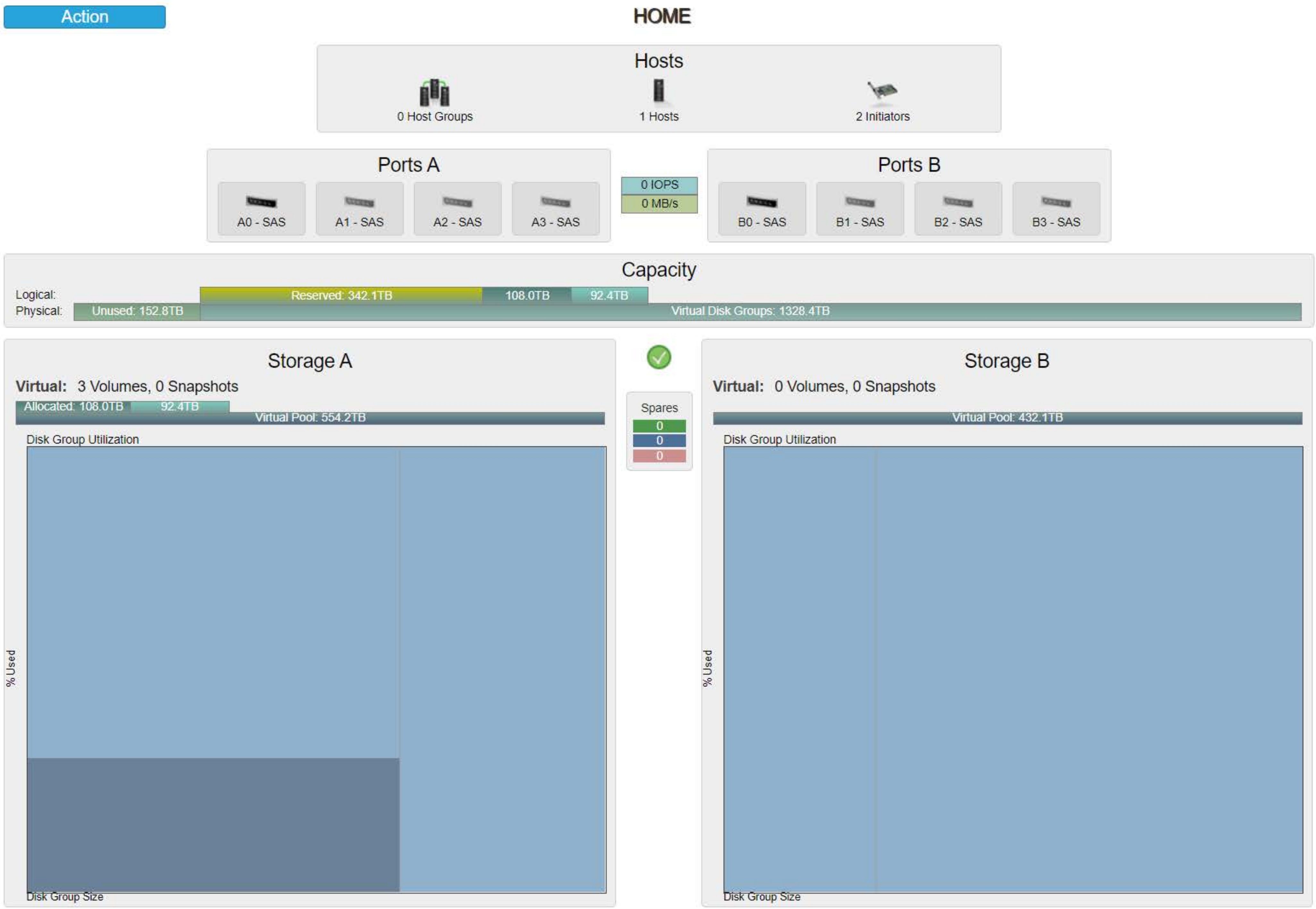
- A NAS for ORFEO Cluster
- Internally:
  - An entry level SAN

ORFEO: long term storage

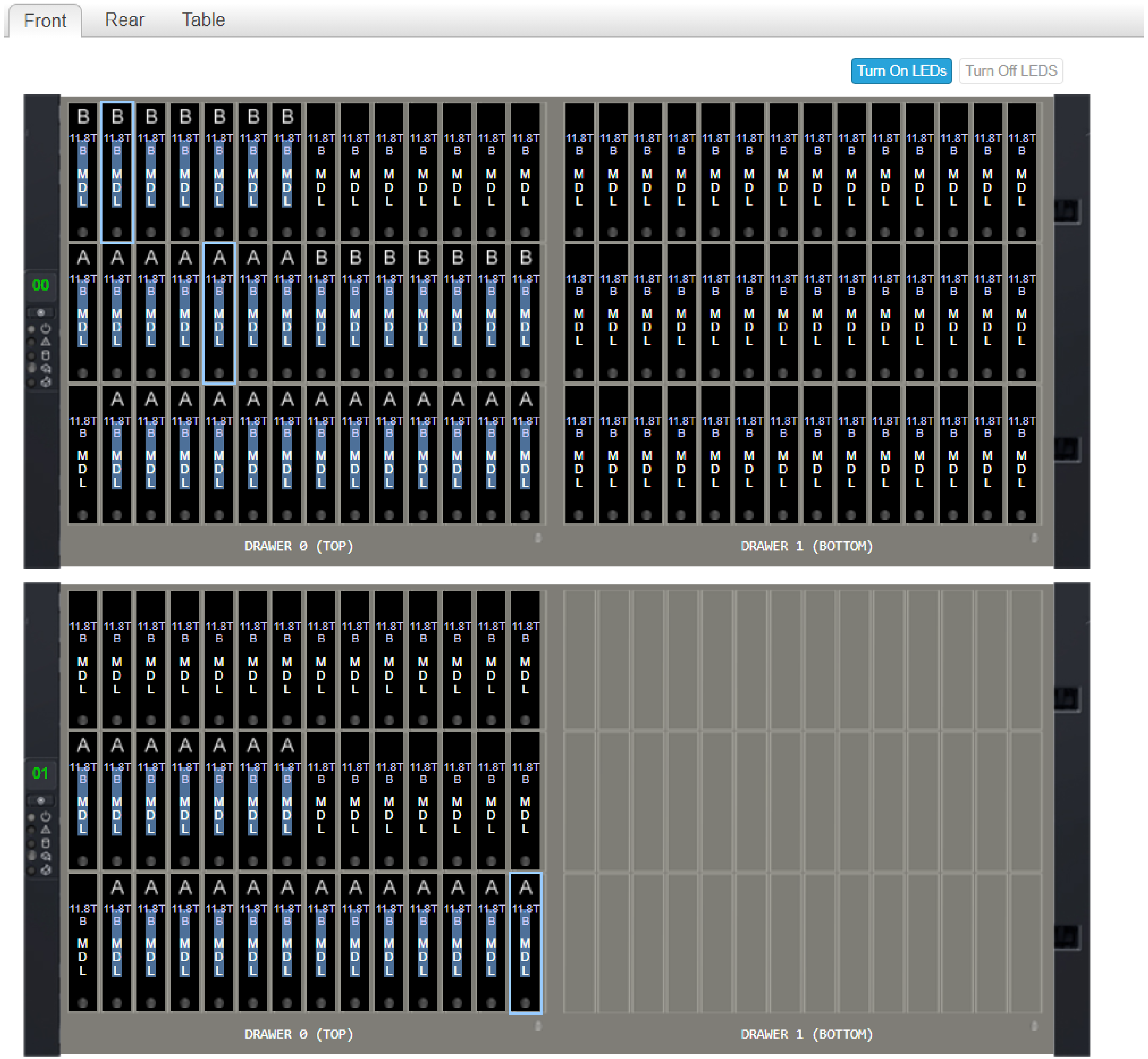
- Dell EMC PowerEdge R640 ⇨
- Dell EMC PowerVault ME4084 ⇨
- + Dell EMC PowerVault ME4084 ⇨




```
>df -h
10.128.2.231:/storage         37T   18T   20T  48% /storage
10.128.2.231:/illumina_run    46T   42T  4.2T  92% /illumina_run
```


Action
HOME
Hosts
0 Host Groups
1 Hosts
2 Initiators
Ports A
A0 - SAS
A1 - SAS
A2 - SAS
A3 - SAS
0 IOPS
0 MB/s
Ports B
B0 - SAS
B1 - SAS
B2 - SAS
B3 - SAS
Capacity
Logical:
Reserved: 342.1TB
108.0TB
92.4TB
Physical:
Unused: 152.8TB
Virtual Disk Groups: 1328.4TB
Storage A
Virtual: 3 Volumes, 0 Snapshots
Allocated: 108.0TB
92.4TB
Virtual Pool: 554.2TB
Disk Group Utilization
% Used
Disk Group Size
Spares
0
0
0
Storage B
Virtual: 0 Volumes, 0 Snapshots
Virtual Pool: 432.1TB
Disk Group Utilization
% Used
Disk Group Size

SYSTEM


Front
Rear
Table
Turn On LEDs
Turn Off LEDS
00
DRAWER 0 (TOP)
DRAWER 1 (BOTTOM)
01
DRAWER 0 (TOP)
DRAWER 1 (BOTTOM)

POOLS

Clear Filters | Export to CSV | Show All | Showing 1 to 2 of 2 entries(1 selected)

Name	Health	Size	Class	Avail	Volumes	Disk Groups
A	OK	356.9TB	Virtual	248.9TB	3	1
B	OK	112.7TB	Virtual	112.7TB	0	1

Related Disk Groups

Clear Filters | Export to CSV | Show All | Showing 1 to 1 of 1 entries(1 selected)

Name	Health	Pool	RAID	Class	Disk Description	Size	Free	Current Job	Status	Disks
dgB01	OK	B	ADAPT	Virtual	SAS MDL	112.7TB	112.7TB	VRSC (58%)	FTOL	14

Related Disks

Clear Filters | Export to CSV | Show All | Showing 1 to 14 of 14 entries

Location	Health	Description	Size	Usage	Disk Group	Status
0.21	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.22	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.23	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.24	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.25	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.26	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.27	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.28	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.29	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.30	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.31	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.32	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.33	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up
0.34	OK	SAS MDL	11.7TB	VIRTUAL POOL	dgB01	Up

Action

VOLUMES

Clear Filters | Export to CSV | Show 10 | Showing 1 to 3 of 3 entries(1 selected)

Group	Name	Pool	Type	Size	Allocated
-	illumina_run	A	base	50.4TB	45.9TB
-	long_term_storage	A	base	109.9TB	40.5TB
-	storage	A	base	39.9TB	21.5TB

Snapshots | Maps | Replication Sets | Schedules

Clear Filters | Export to CSV | Show 10 | Showing 1 to 1 of 1 entries

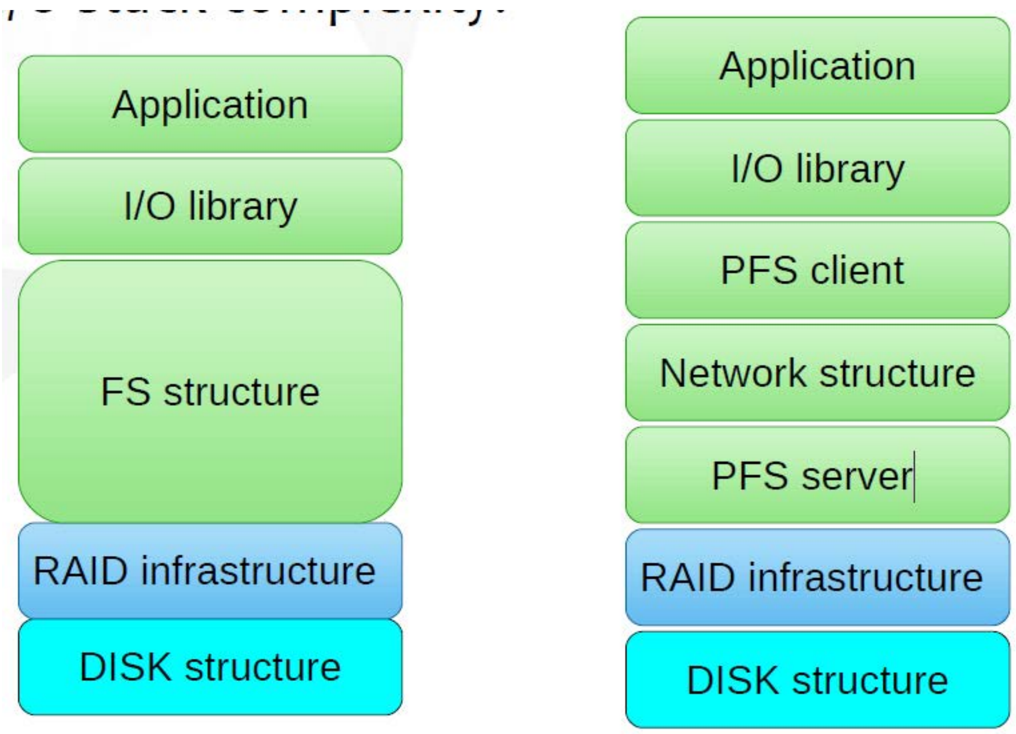
Group.Host.Nickname	Volume	Access	LUN	Ports
ceph9.*	storage	read-write	2	0,1,2,3

```
[root@login bin]# df -h | grep storage
10.128.2.231:/storage                37T   18T   20T  48% /storage
```

Benchmarking I/O on ORFEO

I/O benchmarking…

- It is becoming more and more important
- I/O performance tends to be trickier than CPU/memory ones



How to test a complex I/O infrastructure ?

- Benchmark all the single component of the infrastructure
- Compare simple component Peak performance with measured numbers
- Combine all numbers together to get a performance model and some expected value
- Perform the high level benchmark and compare against what you evaluated.

I/O microbenchmarks to play..

- Measures one fundamental operation in isolation
  - Read throughput, write throughput, creates/sec, etc.
- Good for:
  - Tuning a specific operation
  - Post-install system validation
  - Publishing a big number in a press release
- Not as good for:
  - Modeling & predicting real application performance
  - Measuring broad system performance characteristics
- Example to play
  - IOR: https://github.com/hpc/ior
  - iozone (www.iozone.org)
  - Mdtest (included in the IOR)

Estimate I/O performance of ORFEO storage..

- Peak performance estimate:
  - Network:
    - Infiniband Network from server toward clients: 12GB/sec
  - Disks:
    - HDD: 150 MB/sec (estimate)
    - SDD: 600 MB/sec (estimate)

→

Fast: 80x600=32 GB/sec without replicas

Large: 90x150= 13GB/sec without erasure code:

Measure performance of ORFEO storage..

- Acceptance tests:

  /fast without replica with 56 disks:

  - ~ 20 GB/seconds

Iozone

- Compilation trivial: see tutorial.
- Things to try:
- Test to run:
  - Iozone -a (basic testing)
  - Large file (large than memory to avoid caching effects)

```
iozone -i 1 -i 0 -s 32g -r 1M -f ./32gzero2
```

- Short introduction of basic flags:

  http://www.thegeekstuff.com/2011/05/iozone-examples/

IOR: the de-facto I/O benchmark for HPC

HPC IO Benchmark Repository build error

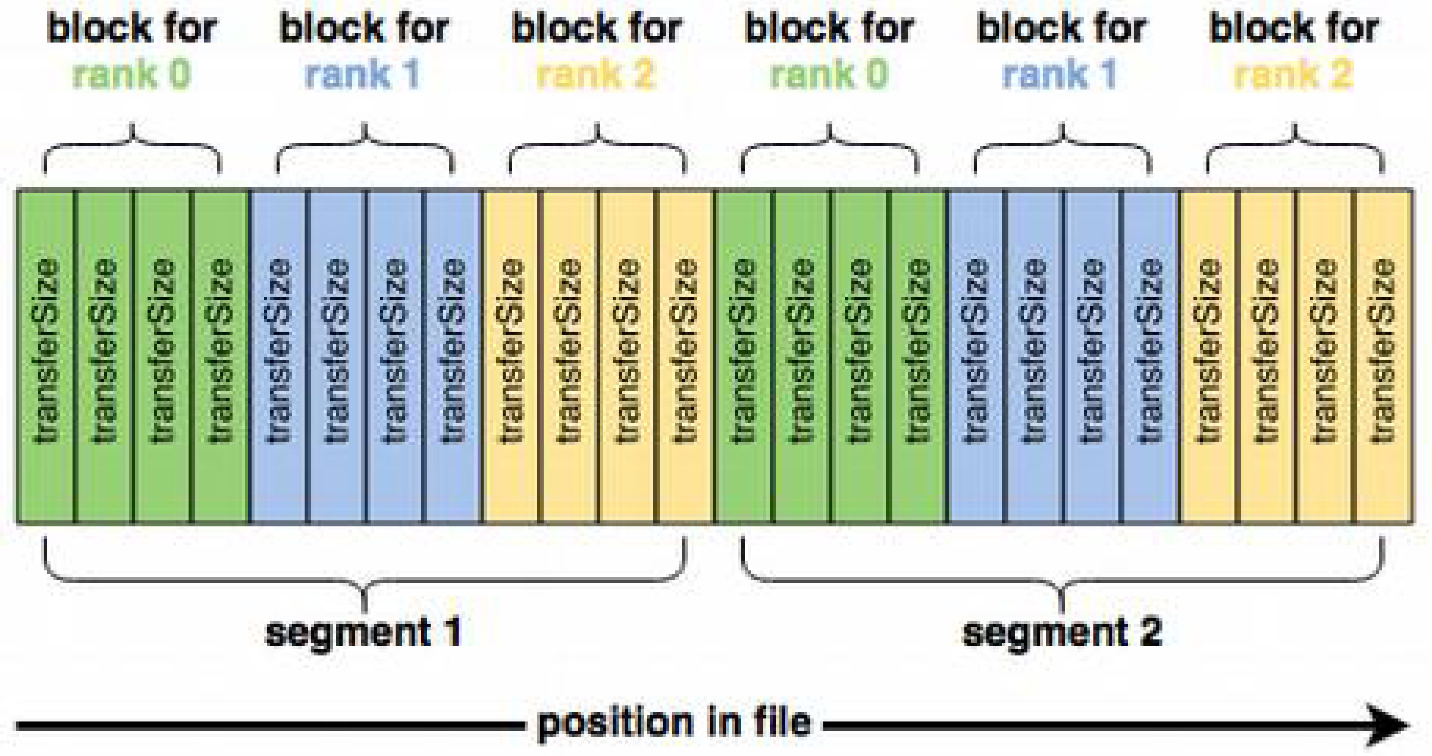
This repository contains the IOR and mdtest parallel I/O benchmarks. The official IOR/mdtest documention can be found in the `docs/` subdirectory or on Read the Docs.

Building

1. If `configure` is missing from the top level directory, you probably retrieved this code directly from the repository. Run `./bootstrap` to generate the configure script. Alternatively, download an official IOR release which includes the configure script.
2. Run `./configure`. For a full list of configuration options, use `./configure --help`.
3. Run `make`
4. Optionally, run `make install`. The installation prefix can be changed via `./configure --prefix=...`.

IOR basic usage:

- IOR writes data sequentially with the following parameters:
  - blockSize (-b)
  - transferSize (-t)
  - segmentCount (-s)
  - numTasks (-n)



IOR number to collect..

- Compare performance of HDF5 vs MPIIO vs POSIX..
- Possible experiments:
  - mpirun -np 32 IOR -a [POSIX|MPIIO|HDF5] -i 3 -d 32 -k -r -E -o yourfile_name -s 1 -b 60G -t 1m
  - mpirun -np 32 IOR -a [POSIX|MPIIO|HDF5] -i 3 -d 32 -k -r -E -o yourfile_name -s 1 -b 16G -t 1m
  - mpirun -np 32 IOR -a [POSIX|MPIIO|HDF5] -i 3-d 32 -k -r -E -o yourfile_name -s 1 -b 4G -t 1m

MD test

- How much does it cost metadata operations ?
- Example to run:

```
mdtest -n 10 -i 200 -y -N 10 -t -u –d $test_directory
```

  - -n: every process will creat/stat/remove # directories and files
  - -i: number of iterations the test will run
  - -y: sync file after writing
  - -N: stride # between neighbour tasks for file/dir stat (local=0)
  - -t: time unique working directory overhead
  - -u: unique working directory for each task
  - -d: the directory in which the tests will run

The end