MHPC
Master in High Performance Computing

Introduction to data management and storage for HPC
part2:
I/O on HPC and parallel FS

Stefano Cozzini

CNR/IOM and eXact-lab srl

SISSA
Scuola Internazionale Superiore di Studi Avanzati

ICTP The Abdus Salam International Centre for Theoretical Physics

MHPC
Master in High Performance Computing

Agenda

- Basic on File System
- Introduction to I/O on HPC
- How to perform I/O on HPC
- Distributed FS & other tricks
- Parallel FS: concepts
- Lustre FS
- Lustre example
- Lustre usage

SISSA ICTP

MHPC
Master in High Performance Computing

Basics on File system

SISSA
ICTP

MHPC
Master in High Performance Computing

File System

A file system is a set of methods and data structures used to organize, store, retrieve and manage information in a permanent storage medium, such as a hard disk. Its main purpose is to represent and organize resources storage.

SISSA ICTP

File System: elements

Name space: is a way to assign names to the items stored and organize them hierarchically.

API: is a set of calls that allow the manipulation of stored items

Security Model: is a scheme to protect, hide and share data.

Implementation: is the code that couples the logical model to the storage medium.

File Systems: Basic Concepts

Disk: A permanent storage medium of a certain size.

Block: The smallest unit writable by a disk or file system. Everything a file system does is composed of operations done on blocks.

Partition: A subset of all the blocks on a disk.

Volume: The term is used to refer to a disk or partition that has been initialized with a file system.

Superblock: The area of a volume where a file system stores its critical data.

File Systems: Basic Concepts (2/2)

Metadata: A general term referring to information that is about something but not directly part of it.

Journaling: write data to journal, commit to file system when complete in atomic operation

- reduces risk of corruption and inconsistency

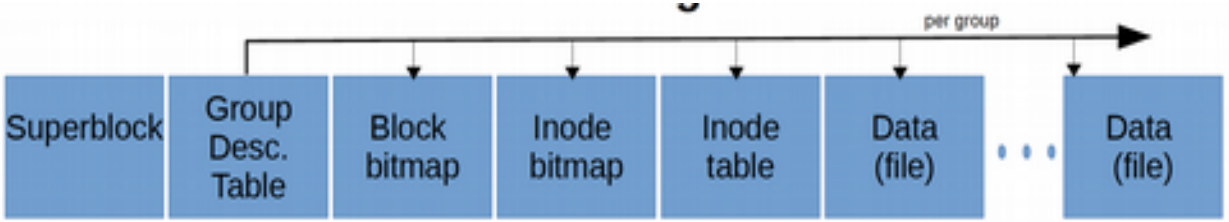
Attribute: A name and value associated with the name. The value may have a defined type (string, integer, etc.).

File Systems: modern concepts

- *Snapshot*: retain status of file system at given point in time by copying metadata and marking object data referred as *copy-on-write*
- *Deduplication*: identify identical storage objects, consolidate and mark them *copy-on-write*

File system : data layout

- How can retrieve data from the disk ?



- Superblock: filesystem type, block counts, inode counts, supported features...
- Group descriptor: location of group data structure, group status
- Group bitmap: one block of 1 bit for allocation status of each blocks
- Inode bitmap: [per group] one block of 1 bit for allocation status of each inode
- Inode Table: [per group] store information of the inode
- Data: your data, finally..

MHPC
Master in High Performance Computing

File system : data layout (2)

```
[root@elcid ~]# tune2fs -l /dev/sda1
tune2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          72228245-8322-4b2f-b043-317f5d9653df
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
// needs_recovery extent flex_bg sparse_super large_
// file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              38400
Block count:              153600
Reserved block count:     7680
Free blocks:              116833
Free inodes:              38336
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      37
Blocks per group:         32768 [...]
```

File system : data layout and inode

- Data structure pointed by the inode number, a unique identifier of a file in the file system
  - address of data block on the storage media
  - description of the file (POSIX)
    - Size of the file
    - Storage device ID
    - User ID of the file's owner.
    - Group ID of the file.
    - File type
    - File access right
    - Inode last modification time (ctime)
    - File content last modification time (mtime),
    - Last access time (atime).
    - Count of hard links pointed to the inode.
    - Pointers to the disk blocks that store the file's contents

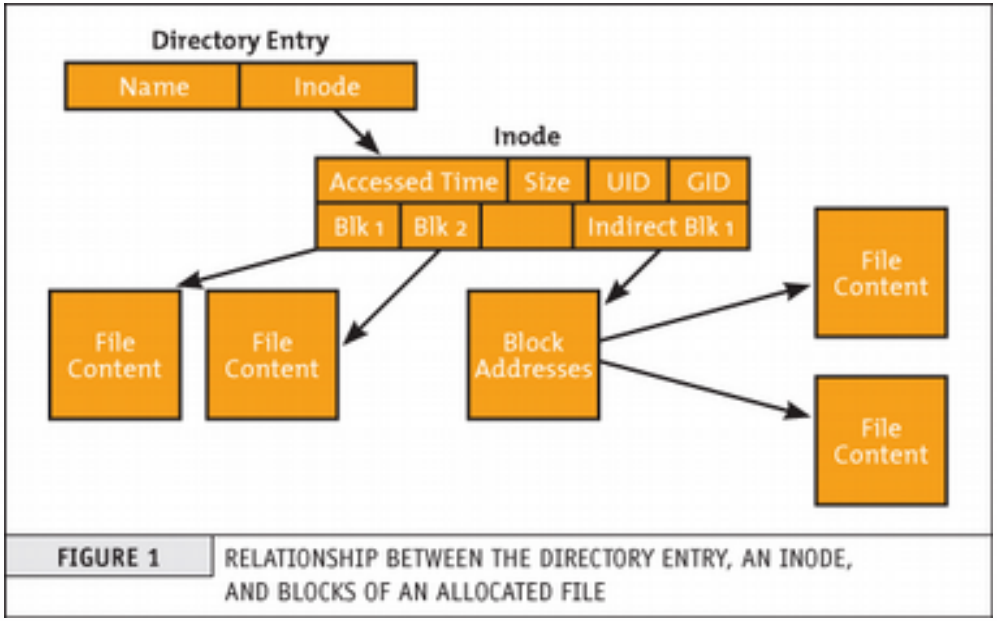


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

MHPC
Master in High Performance Computing

F-S useful command to check inodes

- ls -i
- stat filename
- df -i

SISSA ICTP

Data and metadata

Meta-data : Data to describe data attribute (and extended attribute)

- size, owner, creation date

Meta-data are the bottleneck of scalability

- How many times do you type ls in a day?
  How many times to you write a file?
- ltis scanning all the files in the directory !

(Local) Linux File Systems: few examples

- ext2, ext3, ext4
- Reiserfs
- Xfs
- ...

MHPC
Master in High Performance Computing

Posix

- API to access data and metadata (1988)
- POSIX interface is a useful, ubiquitous interface for building basic I/O tools.
- Standard I/O interface across many platforms.
  - open, read/write, close functions in C/C++/Fortran
- Sequential mode (single stream of accesses)

SISSA ICTP

Posix API (1/2)

Retrieve the list of all files in a directory: 1 call

`Readdir`

Retrieve the attribute of the all files from this directory:

`Fstat` 1 call per file !

Questions:

What happens if the content of a directory is modified by another process?

Posix API (2/2)

Posi assumes atomicity and ubiquity

Changes are visible immediately to all clients

Problem for parallel accesses:

- POSIX requires a strict consistency to sequential order : lock

  (Create a directory is an atomic operation with immediate global view)
- No support for non-continuous I/O
- No hint / prefetching

MHPC
Master in High Performance Computing

Introducing I/O on HPC

SISSA ICTP

HPC and I/O

Scientific applications use I/O:

- to store dataset from simulations for later analysis (output)
- to load initial conditions or datasets for processing (input)
- checkpointing to files that save the state of an application in case of system failure
- implement 'out-of-core' techniques for algorithms that process more data than can fit in system memory

MHPC
Master in High Performance Computing

Building block for HPC I/O systems

- A HPC I/O system should:
  - Present storage as a single, logical storage unit
  - Stripe files across disks and nodes for performance
  - Tolerate failures (in conjunction with other HW/SW)
  - Provide a standard interface
- Issues:
  - "Management problem": many disks 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

SISSA ICTP

MHPC
Master in High Performance Computing

HPC optimization works

- Most optimization work on HPC applications is carried out on:
  - Single node performance
  - Network performance (communication)
  - I/O only when it becomes a real problem

SISSA ICTP

MHPC
Master in High Performance Computing

I/O challenge in HPC

Large parallel machines should perform large calculations

=> Critical to leverage parallelism in all phases including I/O

(do you remember Amdahl law ?)

SISSA ICTP

MHPC
Master in High Performance Computing

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

SISSA ICTP

MHPC

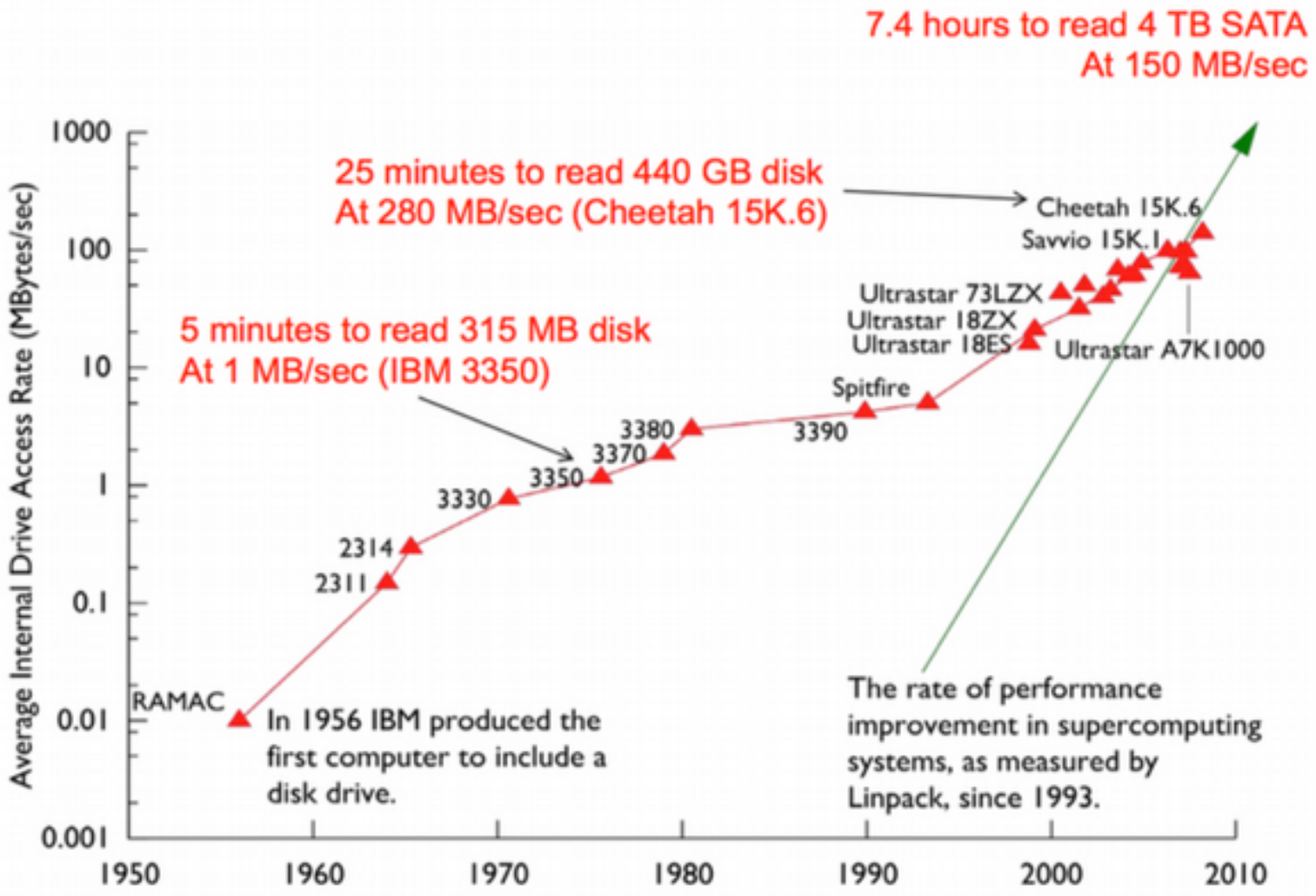


Figure by Rob Ross, Argonne National Laboratory

MHPC
Master in High Performance Computing

Factors which affect I/O

- How is I/O performed?
  - I/O Pattern
  - Number of processes and files.
  - Characteristics of file access.
- Where is I/O performed?
  - Characteristics of the computational system.
  - Characteristics of the file system.

SISSA ICTP

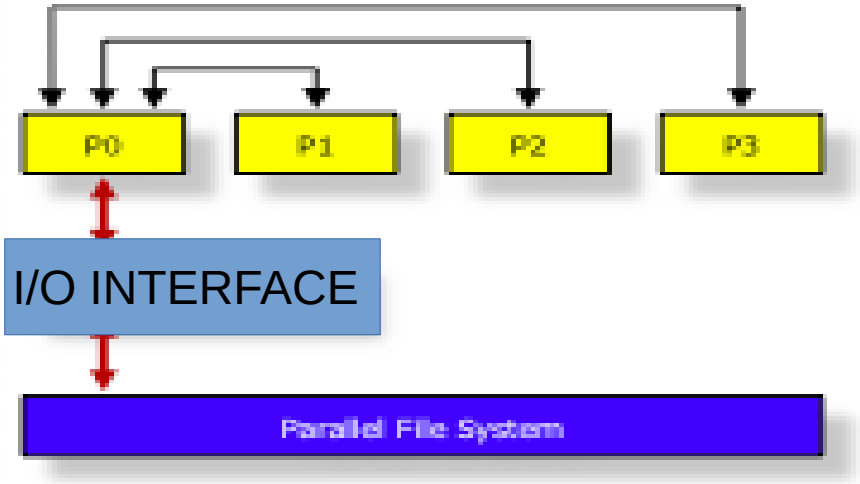
MHPC
Master in High Performance Computing

HOW TO PERFORM I/O on HPC

SISSA ICTP

Serial I/O : spokeperson

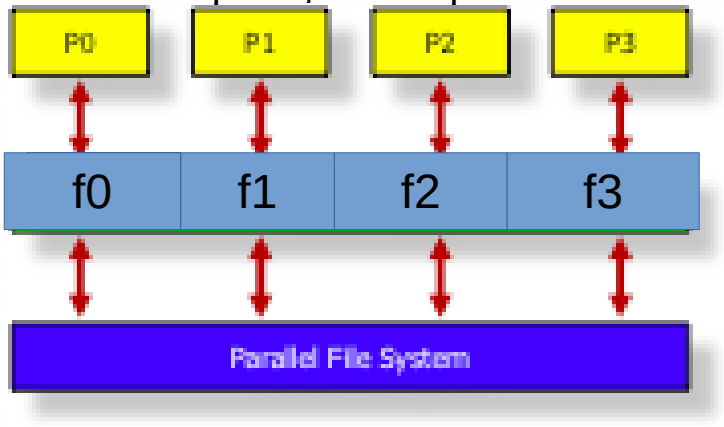
- One process performs I/O.
  - Data Aggregation or Duplication
  - Limited by single I/O process.
- Simple solution, easy to manage, but Pattern does not scale.
  - Time increases linearly with amount of data.
  - Time increases with number of processes.



Parallel I/O: File-per-Process

All processes perform I/O to individual files.

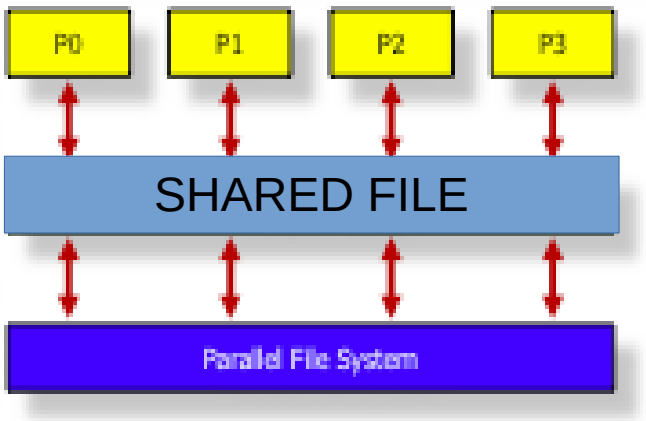
- Limited by file system.
  - Pattern does not scale at large number of processes
    - Number of files creates bottleneck with metadata operations.
    - Number of simultaneous disk accesses creates contention for file system resources.
- Manageability issues:
    - What about managing thousand of files ???
    - What about checkpoint/restart procedures on different number of processors ?



MHPC
Master in High Performance Computing

Parallel I/O

- Each process performs I/O to a single file which is shared.
- Performance Data layout within the shared file is very important.
- Possible contention for file system resources when large number of processors involved..

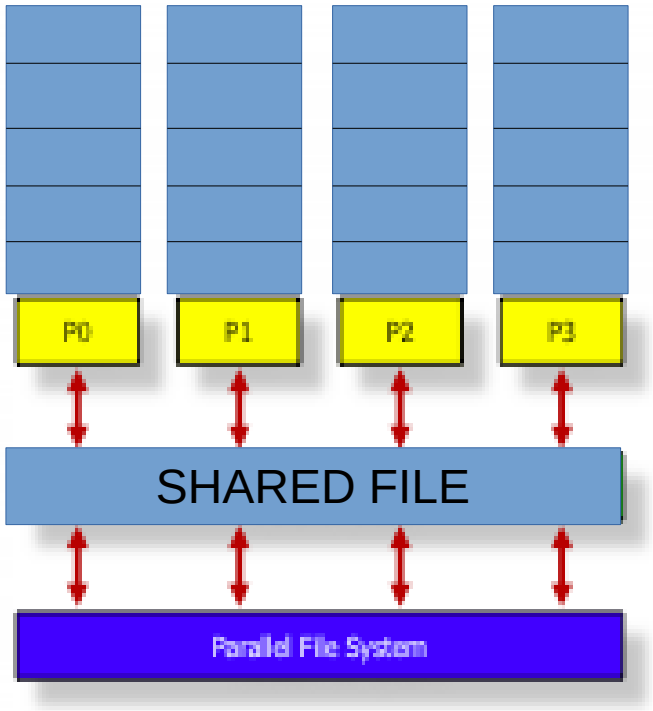


SISSA
ICTP

MHPC
Master in High Performance Computing

Parallel I/O on very large system..

- Accessing a shared filesystem from large numbers of processes could potentially overwhelm the storage system and not only..
- In some cases we simply need to reduce the number of processes accessing the storage system in order to match number of servers or limit concurrent access.

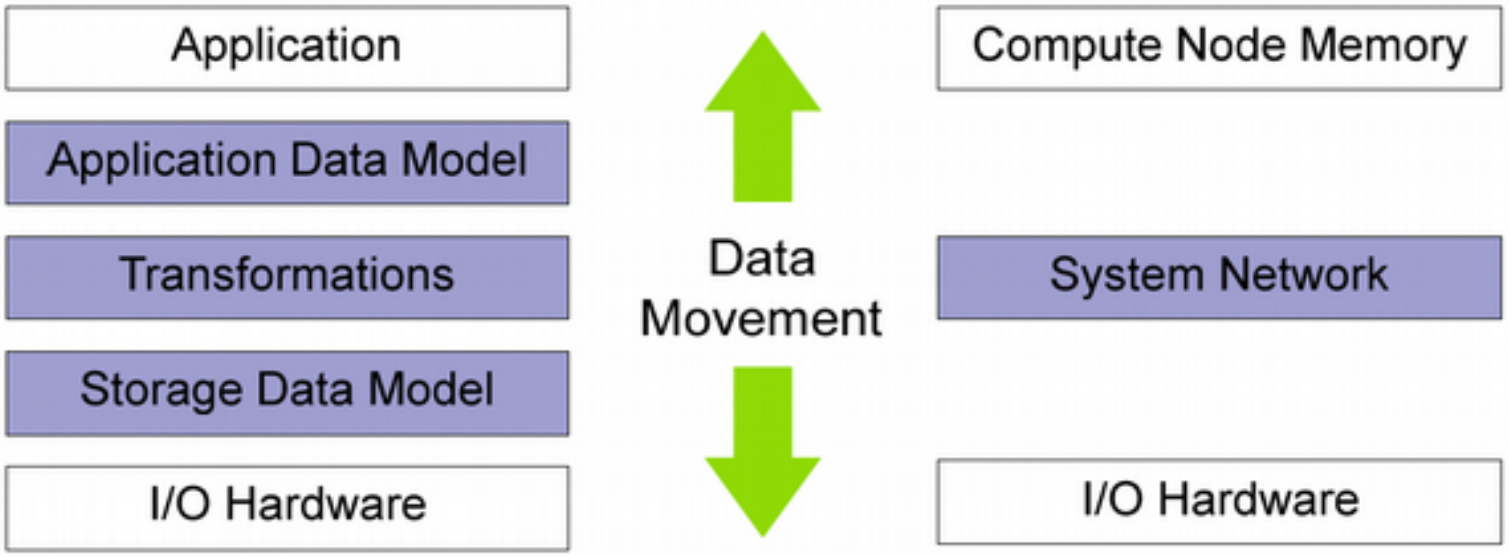


SISSA ICTP

MHPC
Master in High Performance Computing

Views of data access on HPC systems

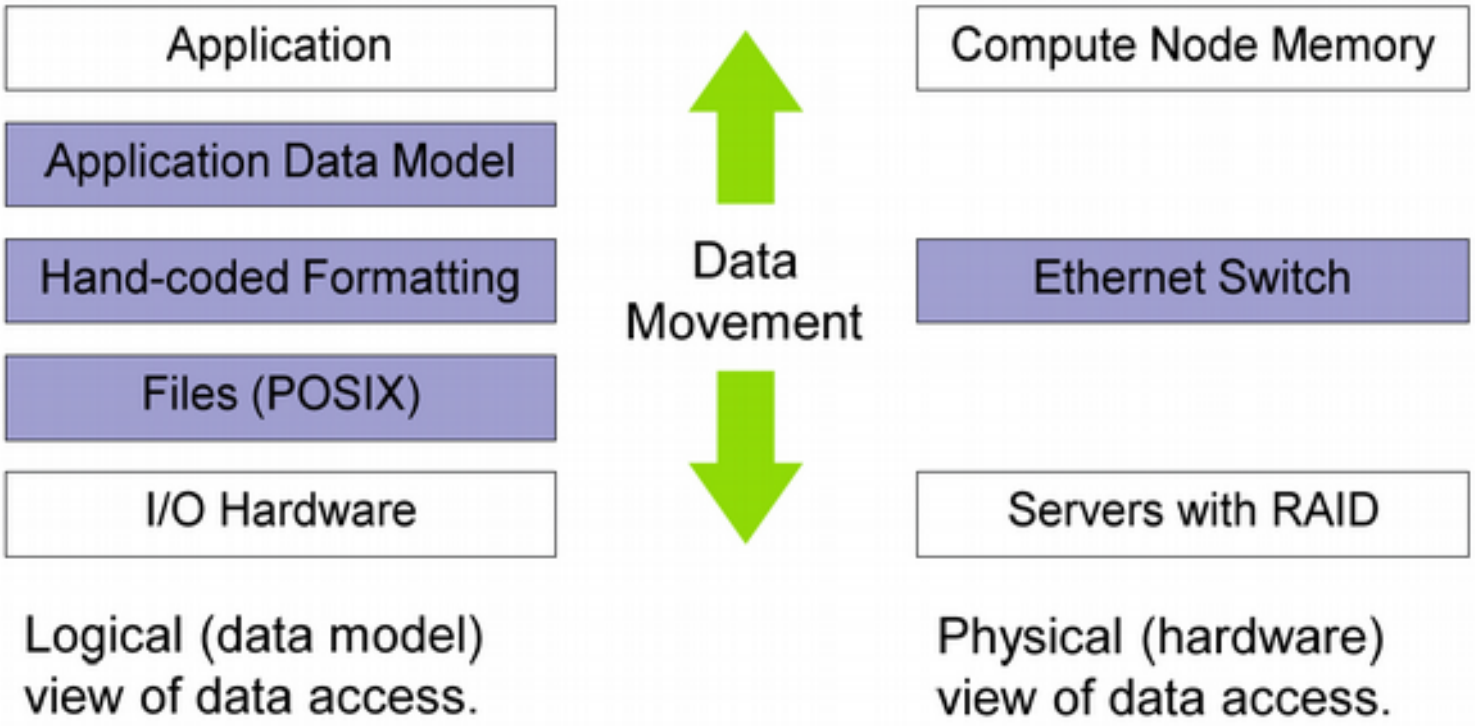
- Logical vs Physical View



SISSA ICTP

Data access in old HPC systems..

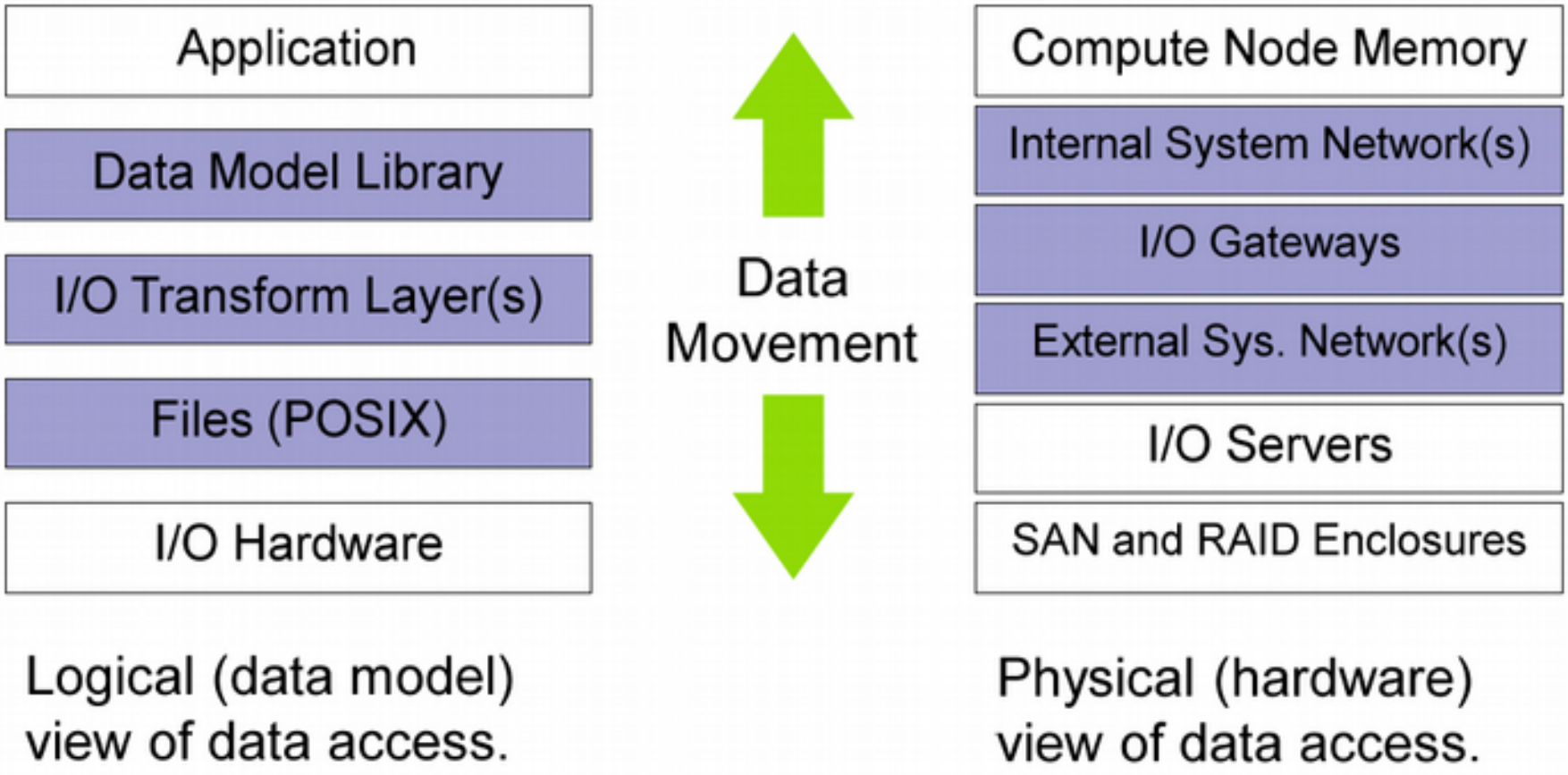
Applications teams wrote their own translation from their data model to files.



MHPC
Master in High Performance Computing

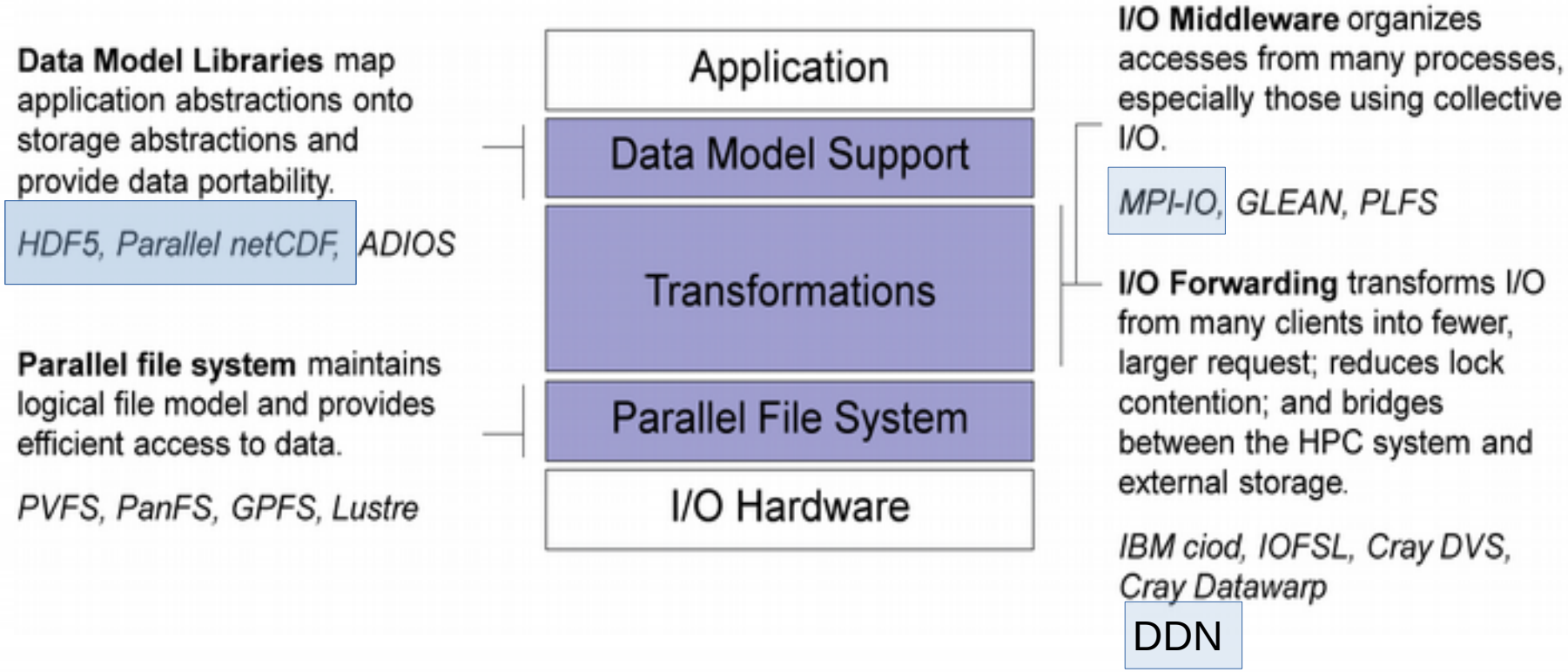
Data access in current HPC systems..

Greater /extended support on logical side, much more complexity on Hardware side



MHPC
Master in High Performance Computing

HPC I/O software stack



SISSA ICTP

MHPC
Master in High Performance Computing

I/O middleware

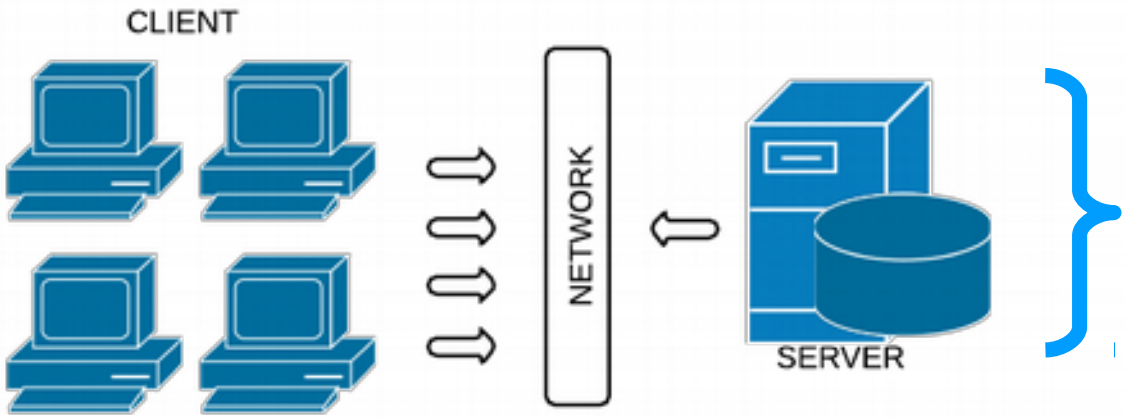
- Match the programming model (e.g. MPI)
- Facilitate concurrent access by groups of processes
  - Collective I/O
  - Atomicity rules
- Expose a generic interface
  - Good building block for high-level libraries
- Efficiently map middleware operations into PFS ones
  - Leverage any rich PFS access constructs,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

SISSA ICTP

MHPC
Master in High Performance Computing

Distributed file system

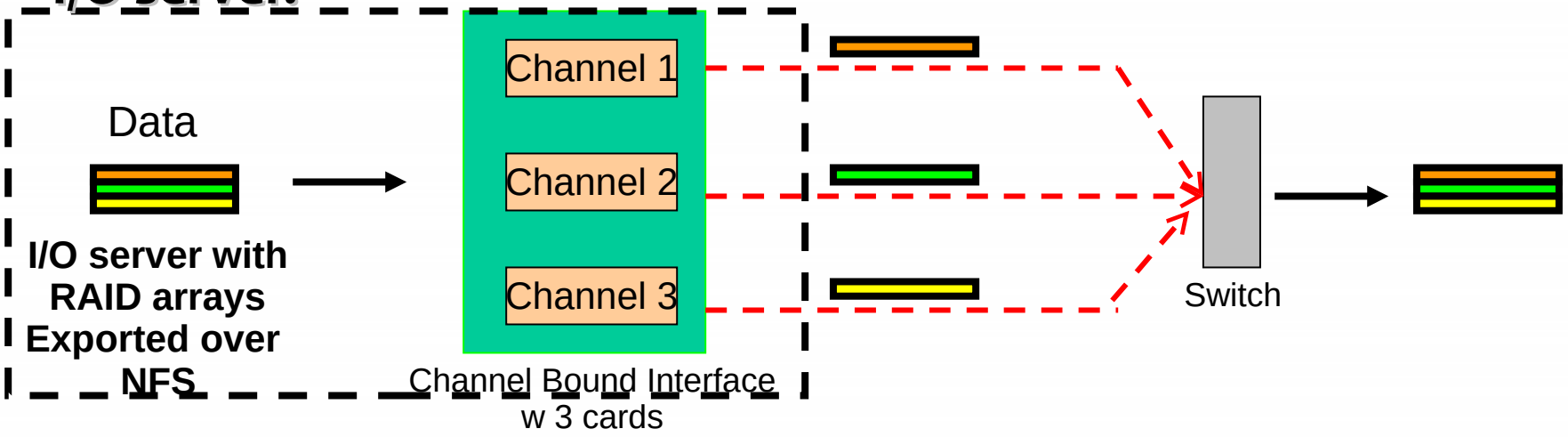
- FS hosted on a server (I/O server)
  - Mounted by several clients (compute nodes/login nodes..)
  - Example: NFS
    - Parallel acces is possible but:
    - Network bandwith limited..
    - Locking issues



SPOF!
bottleneck!

SISSA ICTP

I/O server.



I/O performance can be obtained/ increased by combing RAID (for I/O and/or security and channel bonding (for increased bandwidth).

Good: cheap and easy to maintain and configure.

Bad:

- performance is marginal (N disks give N I/O speedup at best), .
- Need a high end network switch.
- Performance degrades fast as the function of the number of users.
- Needs lots of memory on I/O server to allow for good buffering.
- Needs good UPS/Shutdown solution, to prevent disk corruption.

MHPC
Master in High Performance Computing

PFS for HPC

SISSA
ICTP

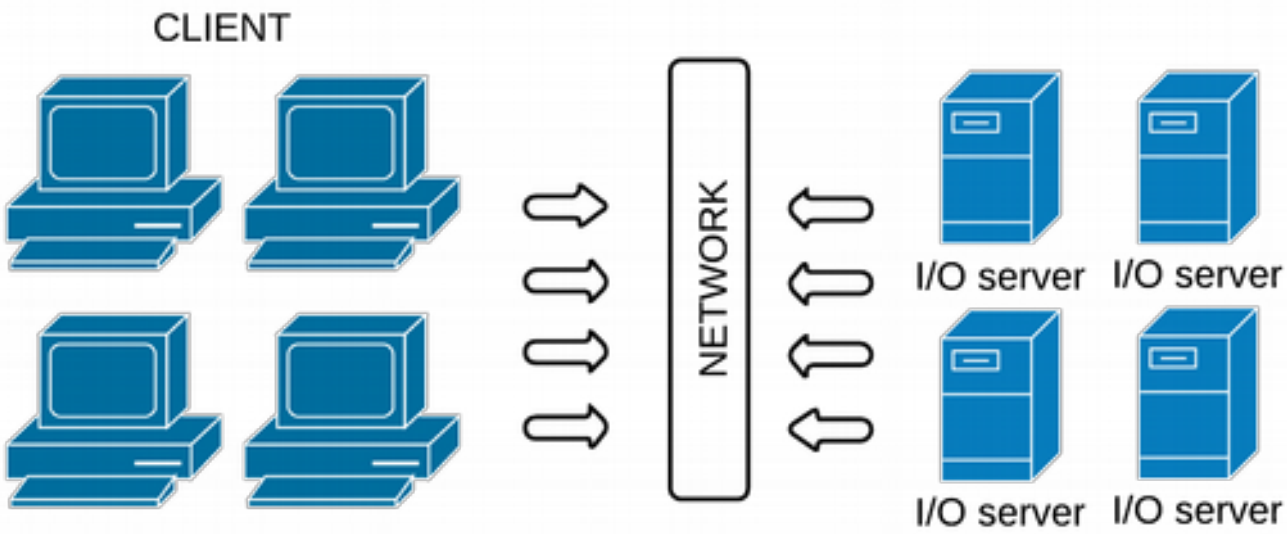
MHPC
Master in High Performance Computing

Parallel FS solution

- 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 make sense of data stored in the file system
- and optionally:
  - monitoring software that ensures continuous availability of all needed components
  - 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

SISSA ICTP

Parallel File System: Typical configuration



1) Each I/O server hosts some data
2) Metadata server coordinates access by the clients to the data
3) Metadata server hosted on I/O server (dedicated and/ or shared)

Parallel filesystem approach

- In a parallel solution..
  - you add storage servers, not only disk space
  - each added storage server brings in more memory, more processor power, more bandwidth
  - software complexity however is much higher
- there is no «easy» solution, like NFS...

MHPC
Master in High Performance Computing

A disclaimer...

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

SISSA ICTP

MHPC
Master in High Performance Computing

Hardware to build a PFS

- 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

SISSA ICTP

MHPC
Master in High Performance Computing

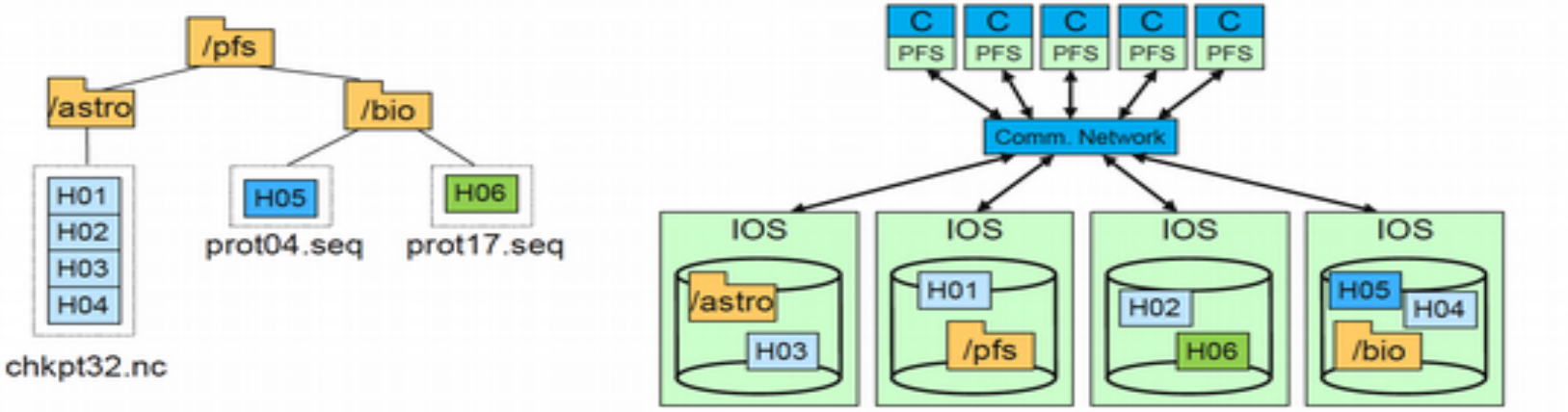
What is available on the market ?

- BeeGFS/FHGFS
  - 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 tool to manage and support ($$$)
  - http://lustre.opensfs.org/
- GPFS
  - IBM proprietary $$$
  - http://www-03.ibm.com/systems/platformcomputing/products/gpfs/

SISSA ICTP

Parallel FS



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

MHPC
Master in High Performance Computing

Comparing parallel FS

- Block Management
- How Metadata is stored
- What is cached and where
- Fault tolerance mechanisms
- Management/Administration

- Performance
- Reliability
- Manageability
- Cost

SISSA ICTP

MHPC
Master in High Performance Computing

Recap on Metadata

- Metadata names files and describes where they are located in the distributed system
  - Inodes hold attributes and point to data blocks
  - Directories map names to inodes
- Metadata updates can create performance problems
- Different approaches to metadata are illustrated via the File Create operation

SISSA ICTP

MHPC
Master in High Performance Computing

Create a file on local FS

- 4 logical I/Os
  - Journal update
  - Inode allocation
  - Directory insert
  - Inode update
- Performance determined by journal updates
  - Details vary among systems

SISSA ICTP

MHPC
Master in High Performance Computing

Create a file on NFS

- RPC (remote procedure call)
  - Client to NFS server RPC
- NVRAM update (or other trick to)
  - Mirrored copy on peer via RDMA
- Reply to client
- Local I/O
  - Inode alloc, Inode update, Directory
  - Done in the background
- Performance from
  - NVRAM+RDMA

SISSA ICTP

Create a FS on Lustre

- Client to Server RPC
- Server creates local file to store metadata
  - Journal update, local disk I/O
- Server creates container object(s)
  - Object create transaction with OSS
  - OSS creates local file for object
- Performance dependent on
  - Local file systems on metadata server and OSS/OST

MHPC
Master in High Performance Computing

Lustre FS

SISSA
ICTP

MHPC
Master in High Performance Computing

Building blocks of Lustre FS

- Lustre Clients
  - – Remote clients that can mount the Lustre filesystem, e.g. your Linux cluster Compute nodes.
- Object Storage Targets (OST)
  - These are block devices that data will be distributed over. These are commonly RAID6 arrays of HDDs.
- Object Storage Server (OSS)
  - A dedicated server that is directly connected to one or more OSS. These are usually connected to the HPC facility by means via a high performance Network (infiniband)
- MetaData Server (MDS) – A single server per file system that is responsible for holding meta data on individual files
  - Filename and location
  - Permissions and access control
  - Which OSTs data is held on.
- MetaData Target (MDT)
  - block device where medata are physically stored

SISSA ICTP

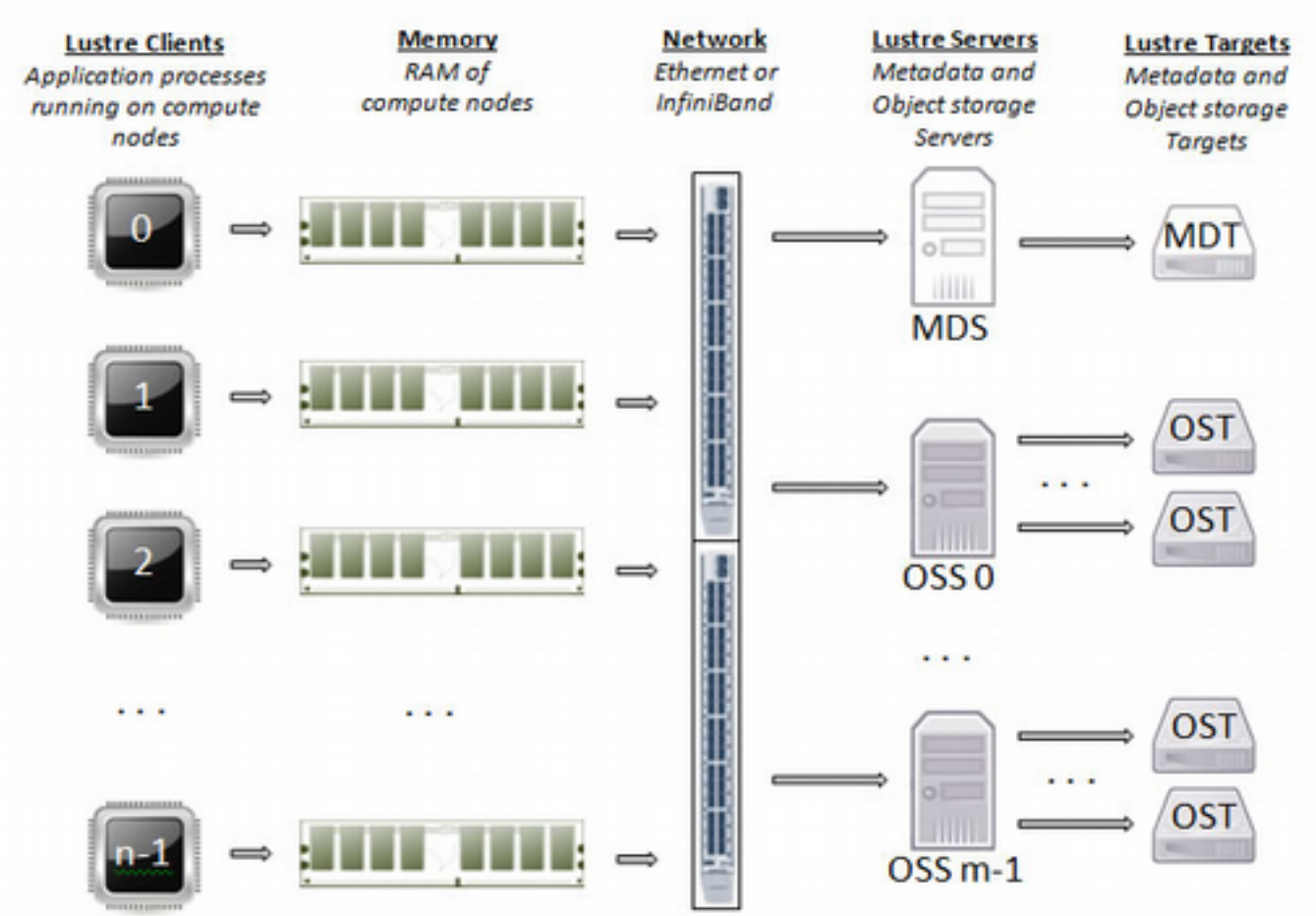
MHPC
Master in High Performance Computing

Lustre Principles

- Lustre is centralized
  - No copy of data or metadata.
  - Metadata stored on a shared storage called Metadata Target (MDT)
  - Attached to two Metadata Servers (MDS)
- MDS provides the illusion of shared name space
- Lustre is an overlay file system
- Data are stored on back-end file-systems: Object Storage Target (OST)
  - Attached to several Object Storage Servers (OSS) for active/active failover.
  - OSS are the servers that handle I/O requests.

SISSA ICTP

Lustre components



MHPC
Master in High Performance Computing

Lustre component characteristics

- Clients
  - 1 - 100,000 per system
  - Data bandwidth up to few GB/s; 1000's of metadata ops/s
  - No particular hardware or attached storage characteristics required
- Object Storage Servers (OSS)
  - 1 - 1000 per system
  - 500 MB/s..2.5 GB/s I/O bandwidth
  - Require good network bandwidth and adequate local storage capacity coupled with sufficient number of OSSs in the system
- Metadata Servers (MDS)
  - 2 (for redundancy) per system
  - 3,000 - 15,000 metadata ops/s
  - Utilize 1 – 2% of total file system capacity
  - Require memory-rich nodes with lots of CPU processing power

SISSA ICTP

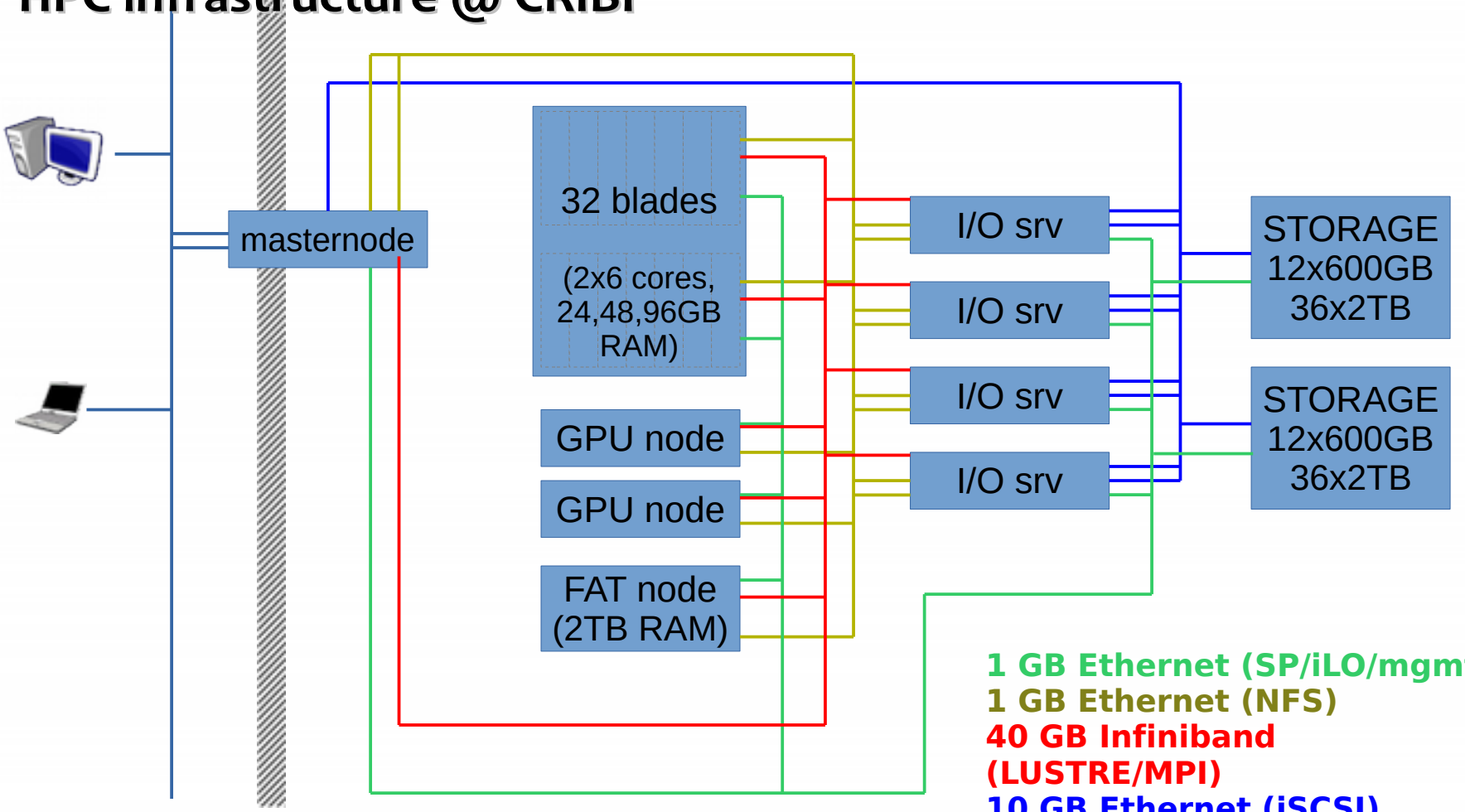
MHPC
Master in High Performance Computing

Lustre Example

SISSA ICTP

MHPC
Master in High Performance Computing

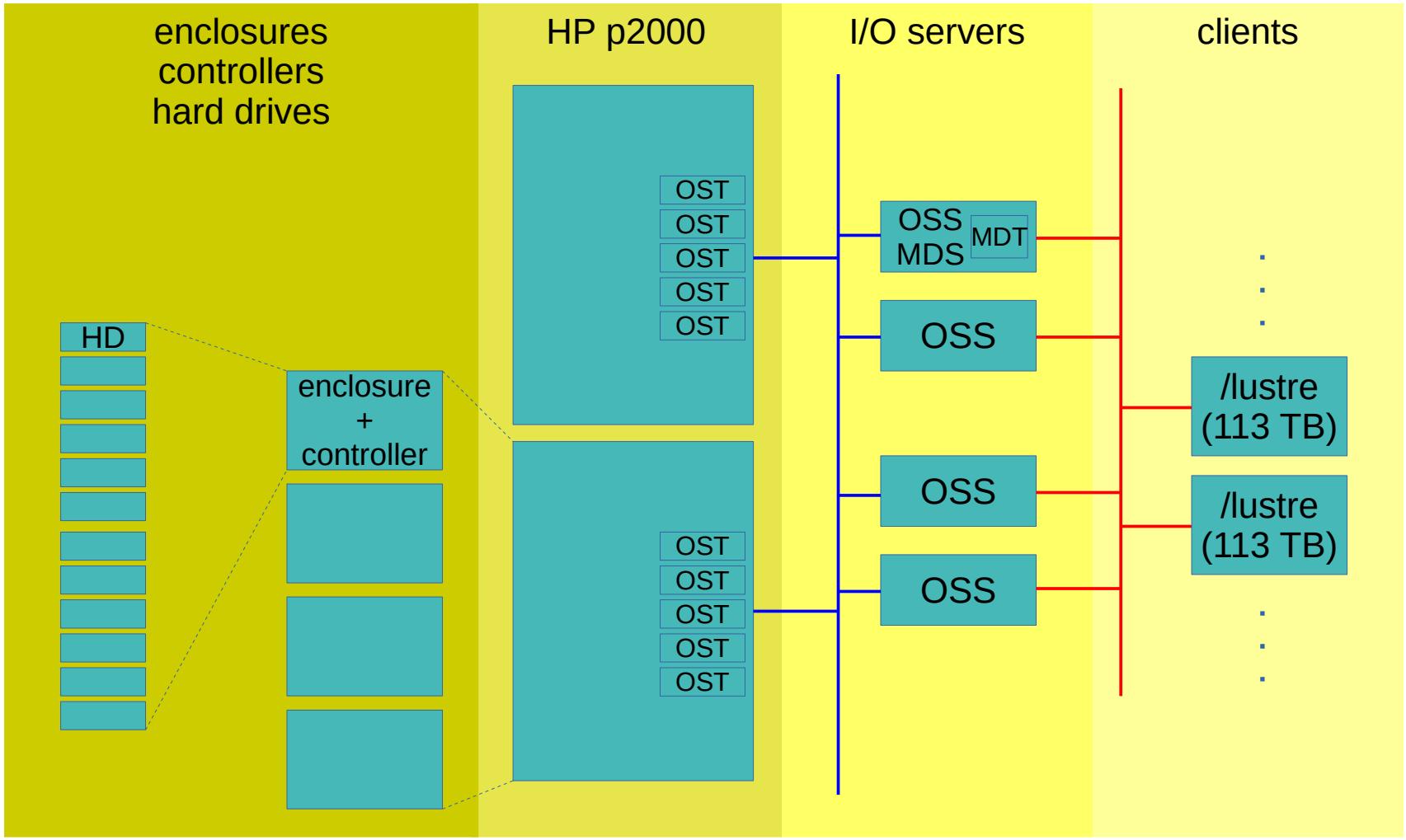
HPC infrastructure @ CRIBI



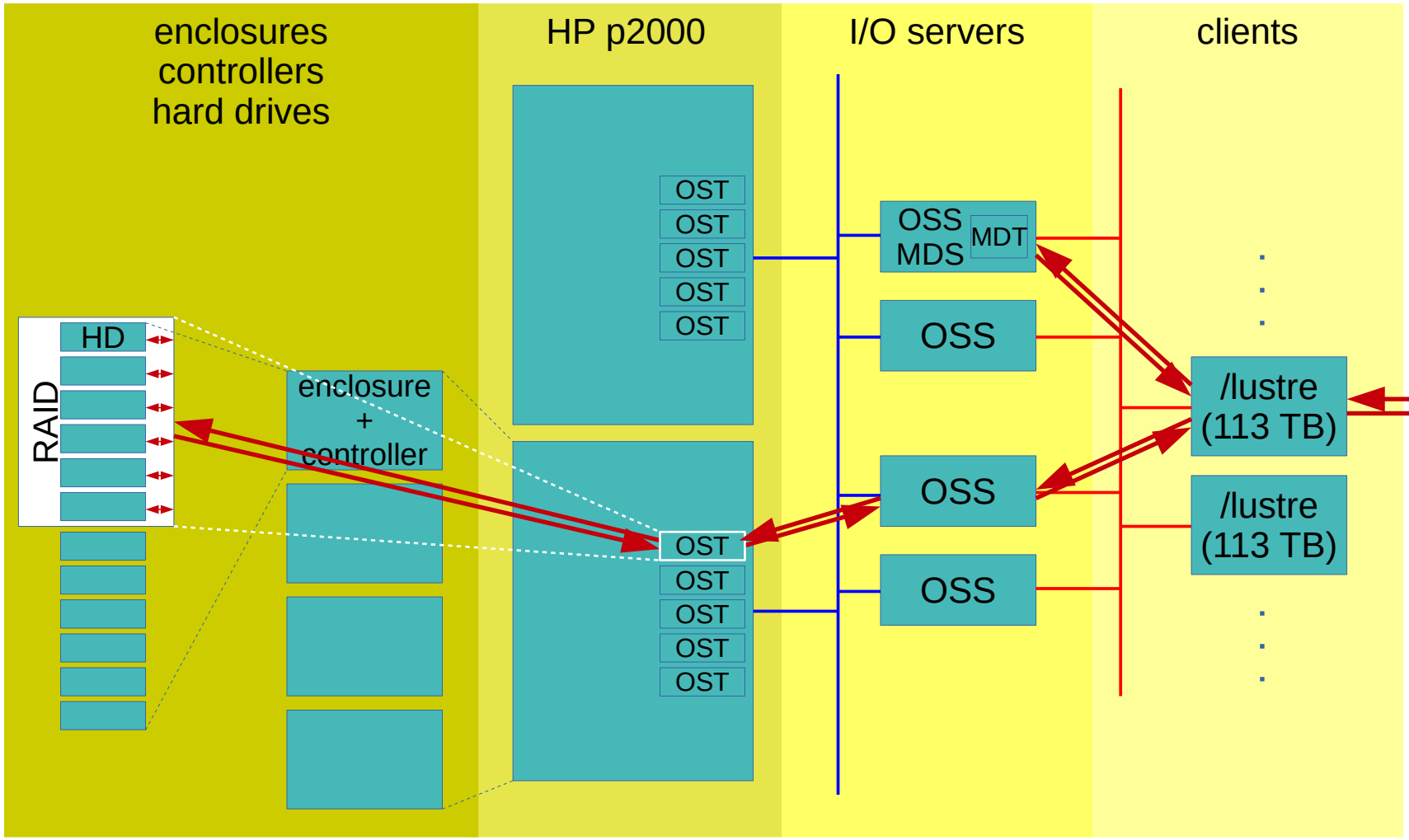
1 GB Ethernet (SP/iLO/mgmt)
1 GB Ethernet (NFS)
40 GB Infiniband (LUSTRE/MPI)
10 GB Ethernet (iSCSI)
1 GB (LAN)

SISSA ICTP

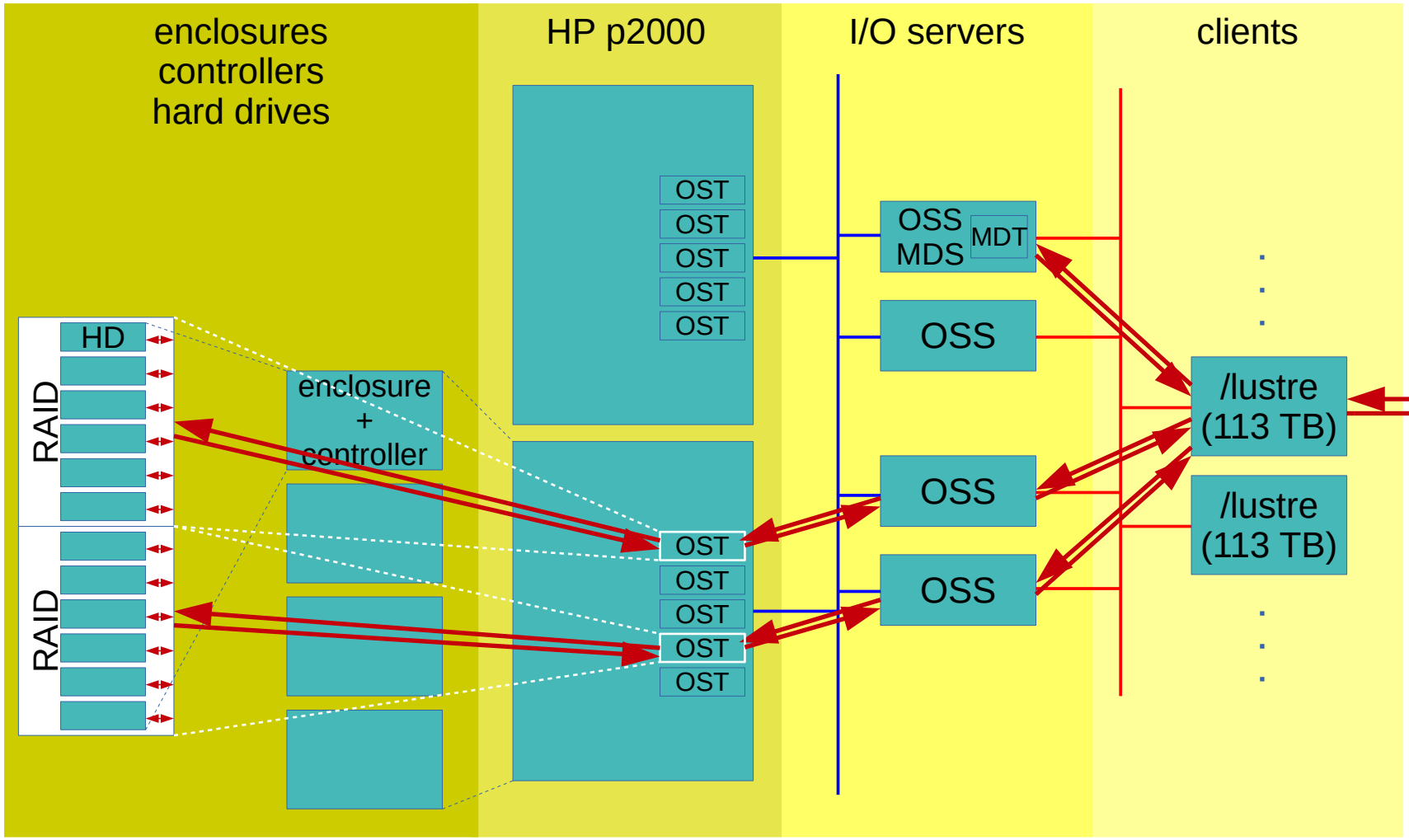
LUSTRE and storage solution



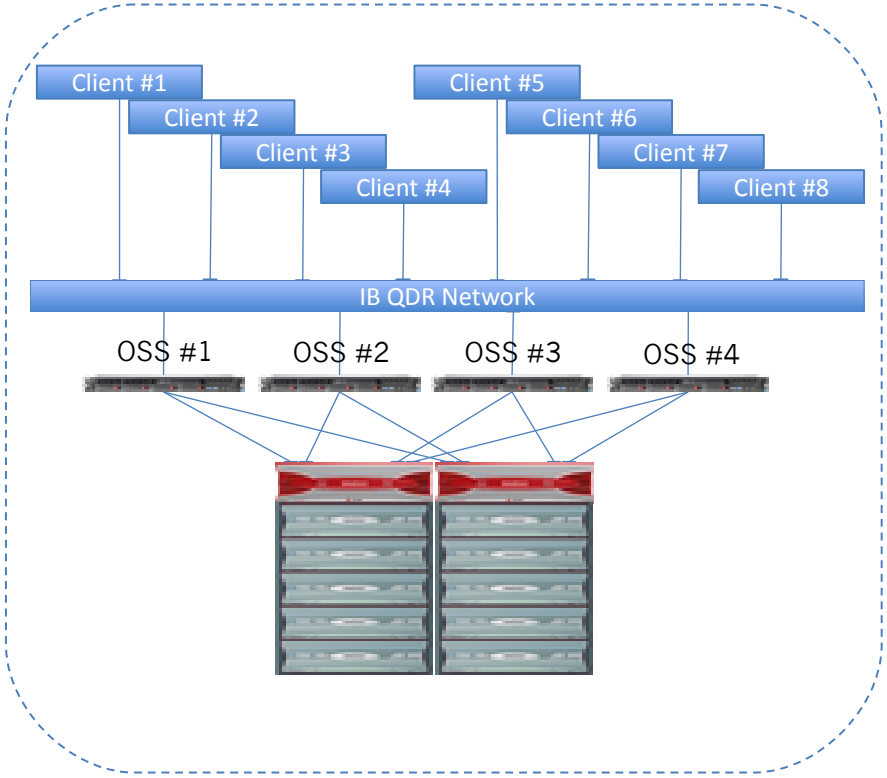
accessing LUSTRE filesystem



why "parallel" filesystem?



overall LUSTRE performance

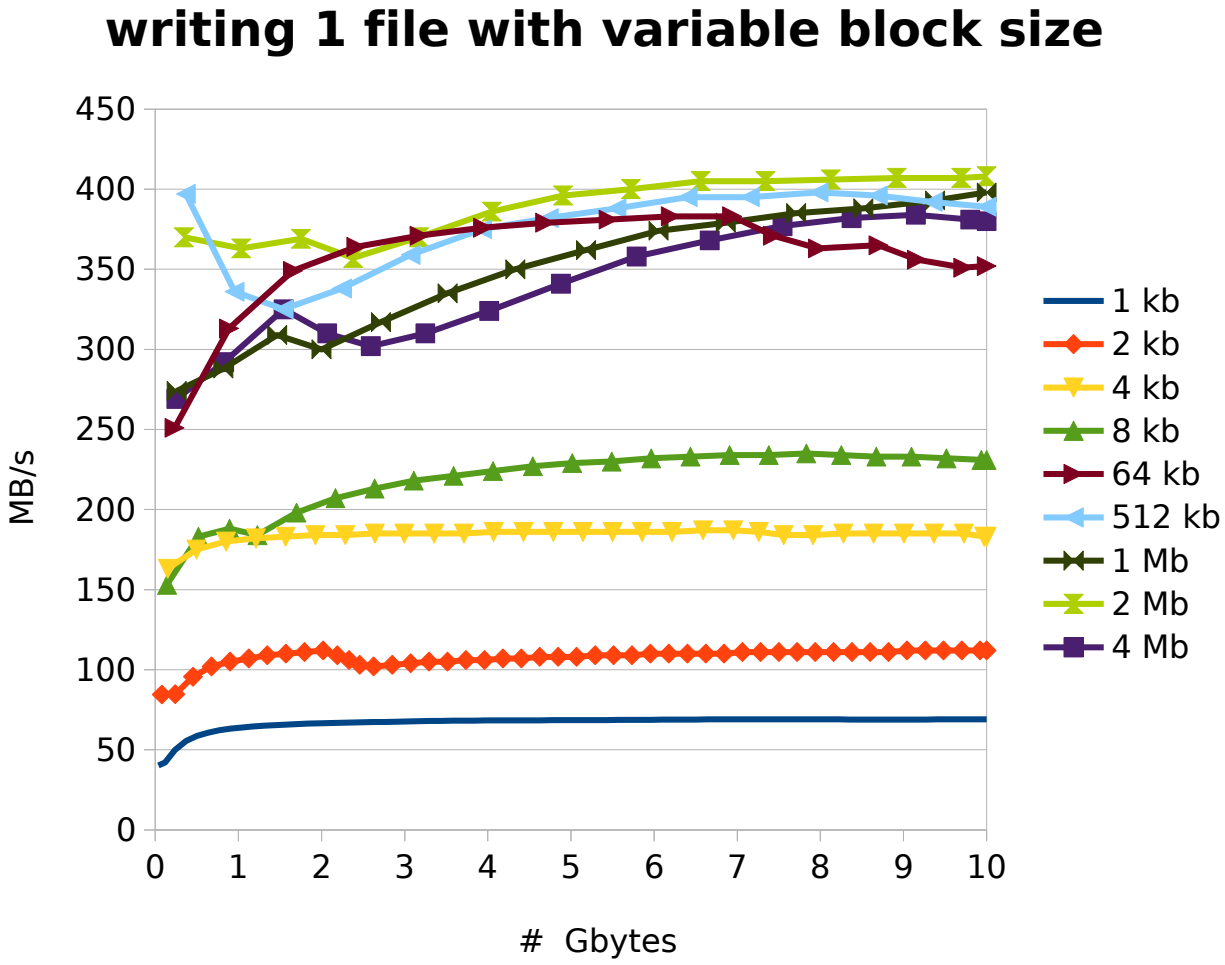


- sequential write/read by iozone
- 1 ~ 8 clients, 1 ~ 4 proc/client
- 32 GB files writing
- 64 GB files reading

~ 1.7 GB/sec writing
32 clients, 32 GB files

~ 1.2 GB/sec reading
32 clients, 64 GB files

LUSTRE can be disappointing too...



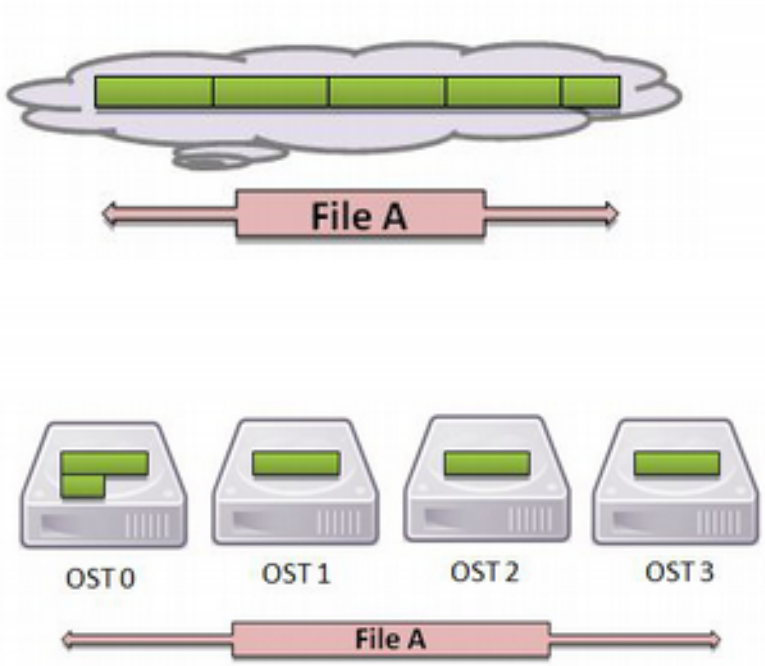
MHPC
Master in High Performance Computing

Using Lustre

SISSA
ICTP

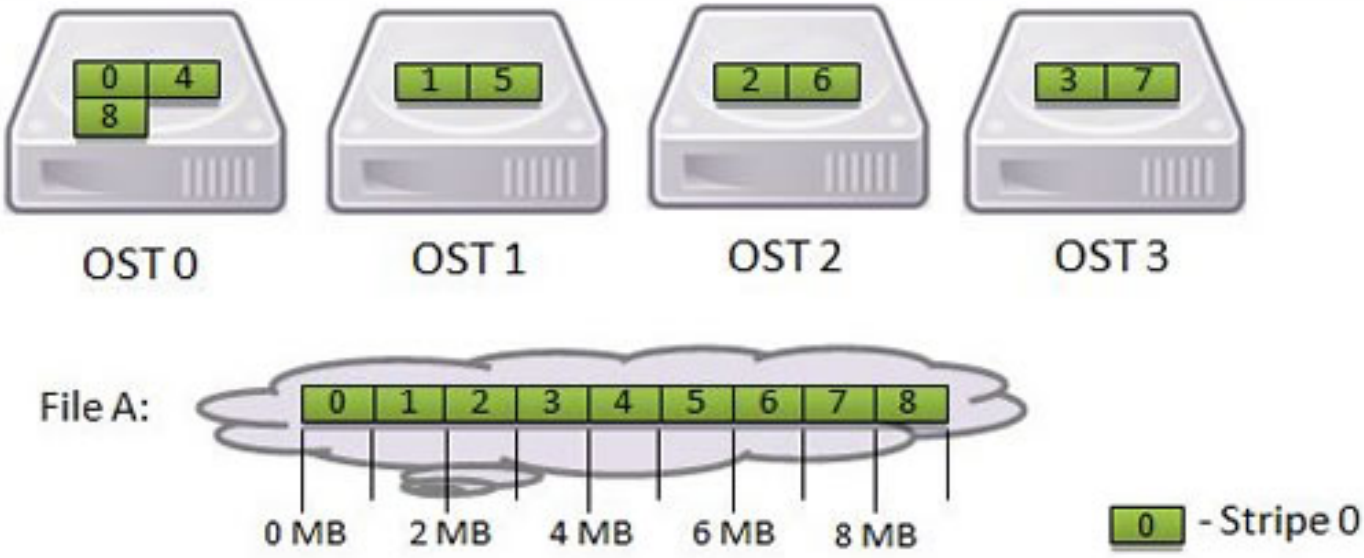
file striping in lustre

- Distribute a file over several OST..



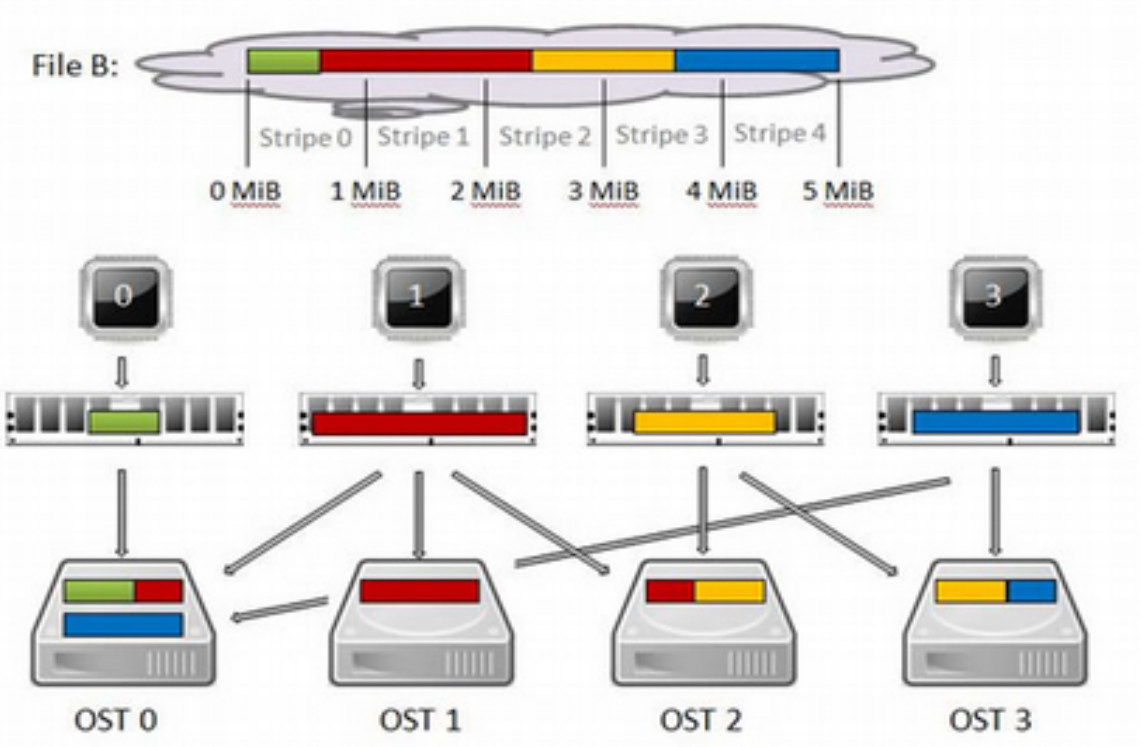
Aligned stripes

- A file large 9Mb
- Stripe size: 1Mb
- Four OST to stripe:



Non aligned stripes

- A file B 5Mb
- Written in parallel with different offset:



MHPC
Master in High Performance Computing

Basic Lustre User command

- Lustre's lfs utility provides several options for monitoring and configuring your Lustre environment.
- You can:
  - List OSTs in the File System
  - Search the Directory Tree
  - Check Disk Space Usage
  - Get Striping Information
  - Set Striping Patterns

```
>lfs -help
```

SISSA ICTP

MHPC
Master in High Performance Computing

lfs basic usage

- `lfs df [-h] [-i]`
  - To check size and number of inodes [-i]

```
[root@m001 ~]# lfs df
UUID                   1K-blocks        Used   Available Use% Mounted on
lustre-MDT0000_UUID    157258928      494864   156764064   0% /lustre[MDT:0]
lustre-OST0000_UUID   6266049040    27197816  6238851144   0% /lustre[OST:0]
lustre-OST0001_UUID   6266049040    26622240  6239425692   0% /lustre[OST:1]
lustre-OST0002_UUID   6266049040    27687756  6238357652   0% /lustre[OST:2]
lustre-OST0003_UUID   6266049040    27897188  6238150736   0% /lustre[OST:3]

filesystem summary:  25064196160   109405000 24954785224   0% /lustre
```

- `lfs quota -u username /your_lustreFS`

SISSA ICTP

MHPC
Master in High Performance Computing

lfs command..

- `lfs osts`
  - Show available OST on filesystem
- `lfs find`

```
[root@m001 ~]# lfs find  /lustre -mtime +30 -type f -print --maxdepth 1
/lustre/wrf-benchmark.tgz
/lustre/pio.tgz
/lustre/wrf-crma-tests.tgz
```

- `lfs getstripe <directory|filename>`
  - Give you info about striping
- `lfs setstripe`
  - Allow you to set the striping

SISSA ICTP

lfs getstripe.

```
[cozzini@elcid cozzini]$ lfs help getstripe
getstripe: To list the striping info for a given file or files in a
directory or recursively for all files in a directory tree.
usage: getstripe [--ost|-O <uuid>] [--quiet | -q] [--verbose | -v]
                 [--stripe-count|-c] [--stripe-index|-i]
                 [--pool|-p] [--stripe-size|-S] [--directory|-d]
                 [--mdt-index|-M] [--recursive|-r] [--raw|-R]
                 <directory|filename> ...
```

```
[cozzini@elcid cozzini]$ lfs getstripe datafile
datafile
lmm_stripe_count:   1
lmm_stripe_size:    1048576
lmm_layout_gen:     0
lmm_stripe_offset:  1
        obdidx           objid           objid           group
             1         4157803       0x3f716b               0
```

lfs getstripe.

```
[cozzini@elcid cozzini]$ lfs help setstripe
setstripe: Create a new file with a specific striping pattern or
set the default striping pattern on an existing directory or
delete the default striping pattern from an existing directory
usage: setstripe -d <directory>   (to delete default striping)
 or
usage: setstripe [--stripe-count|-c <stripe_count>]
                 [--stripe-index|-i <start_ost_idx>]
                 [--stripe-size|-S <stripe_size>]
                 [--pool|-p <pool_name>]
                 [--block|-b] <directory|filename>
       stripe_size:  Number of bytes on each OST (0 filesystem default)
                     Can be specified with k, m or g (in KB, MB and GB
                     respectively)
       start_ost_idx: OST index of first stripe (-1 default)
       stripe_count: Number of OSTs to stripe over (0 default, -1 all)
       pool_name:    Name of OST pool to use (default none)
       block:   Block file access during data migration
```

MHPC
Master in High Performance Computing

Lustre – some advices to check

- Use ls -l only where absolutely necessary
- Avoid using wild cards with GNU commands,
- Place small files on single OST
- Place directories containing many small files on single OSTs
- Set the stripe count and size appropriately for shared files
- Set the stripe count appropriately for applications which utilize a file-per-process

SISSA ICTP

MHPC
Master in High Performance Computing

What you can do with setstripe ?

- 1.setting striping on a single file:
- 2.setting stripe on a directory:
  - Files within directory hinerits the same striping policy
- Note: you cannot change the striping policy of a file..
- To do this:
  - create a new file with appropriate striping policy and then move the original file over this.

SISSA ICTP

MHPC
Master in High Performance Computing

tips to use lustre at best

identify bottlenecks

- HDs, RAID, controllers, volumes and virtual disks, 10GE, iscsi, LUSTRE fs, local caching/buffers, InfiniBand or Gigabit network, LUSTRE client, access patterns

when accessing files:

- read/write in large chunks, not line by line

when running multiple I/O operations:

- optimize data distribution and disk access (i.e. mpi-i/o)

SISSA ICTP