HPC and I/O

- Stefano Cozzini
- CNR-IOM and eXact lab srl

Agenda

- Introduction to I/O on HPC
- How to perform I/O on HPC

HPC and I/O

IOPS vs FLOPS

- HPC is today compute-centric
- But:
  - You can only compute as fast as you can move data
- Then:
  - scientific computing needs data accessibility rather than computing speed
  - HPC workflow will be soon be bounded by the speed of the storage system..

Source: IDC Direction 2013

Citations

“Very few large scale applications of practical importance are NOT data intensive.”

"A supercomputer is a device for converting a CPU-bound problem into an I/O bound problem." [Ken Batcher]

HPC I/O ecosystem

- HPC I/O system is the hardware and software that assists in accessing data during simulations and analysis and keeping data between these activities
- It composed by
  - Hardware: disks, disk enclosures, servers, networks, etc.
  - Software: parallel file system, libraries, parts of the OS

I/O subsystem on HPC ulysses

```
[cozzini@login2 ~]$ df -h
Filesystem                                  Size  Used Avail Use% Mounted on
/dev/mapper/sysVG-LV00                       20G  3.5G   15G  19% /
tmpfs                                        32G  208K   32G   1% /dev/shm
/dev/sda1                                   194M   92M   93M  50% /boot
/dev/mapper/sysVG-LV02                       48G   69M   46G   1% /tmp
/dev/mapper/sysVG-LV01                       48G  1.1G   45G   3% /var
10.6.0.6@o2ib2:10.6.0.7@o2ib2:/home          43T  9.3T   33T  22% /home
10.6.0.6@o2ib2:10.6.0.7@o2ib2:/scratch      256T  231T   23T  91% /scratch
10.6.0.1:/u/shared                          246G   90G  145G  39% /u/shared
10.6.0.1:/u/extra                           246G  6.4G  228G   3% /u/extra
```

scratch filesystems are used for short term storage for a single job or set of computational jobs and they often have the benefit of being fast and large.

Home filesystem (regular storage) slower and smaller but data there can be considered safe.

Flavors of I/O applications

- Two "flavors" of I/O from applications:
  - Defensive: storing data to protect results from data loss due to system faults
  - Productive: storing/retrieving data as part of the scientific workflow
  - Note: Sometimes these are combined (i.e., data stored both protects from loss and is used in later analysis)
- "Flavor" influences priorities:
  - Defensive I/O: Spend as little time as possible
  - Productive I/O: Capture provenance, organize for analysis

Why I need I/O for scientific computing ?

Scientific applications use I/O:

- to load initial conditions or datasets for processing (input)
- to store dataset from simulations for later analysis (output)
- checkpointing to files that save the state of an application in case of system failure

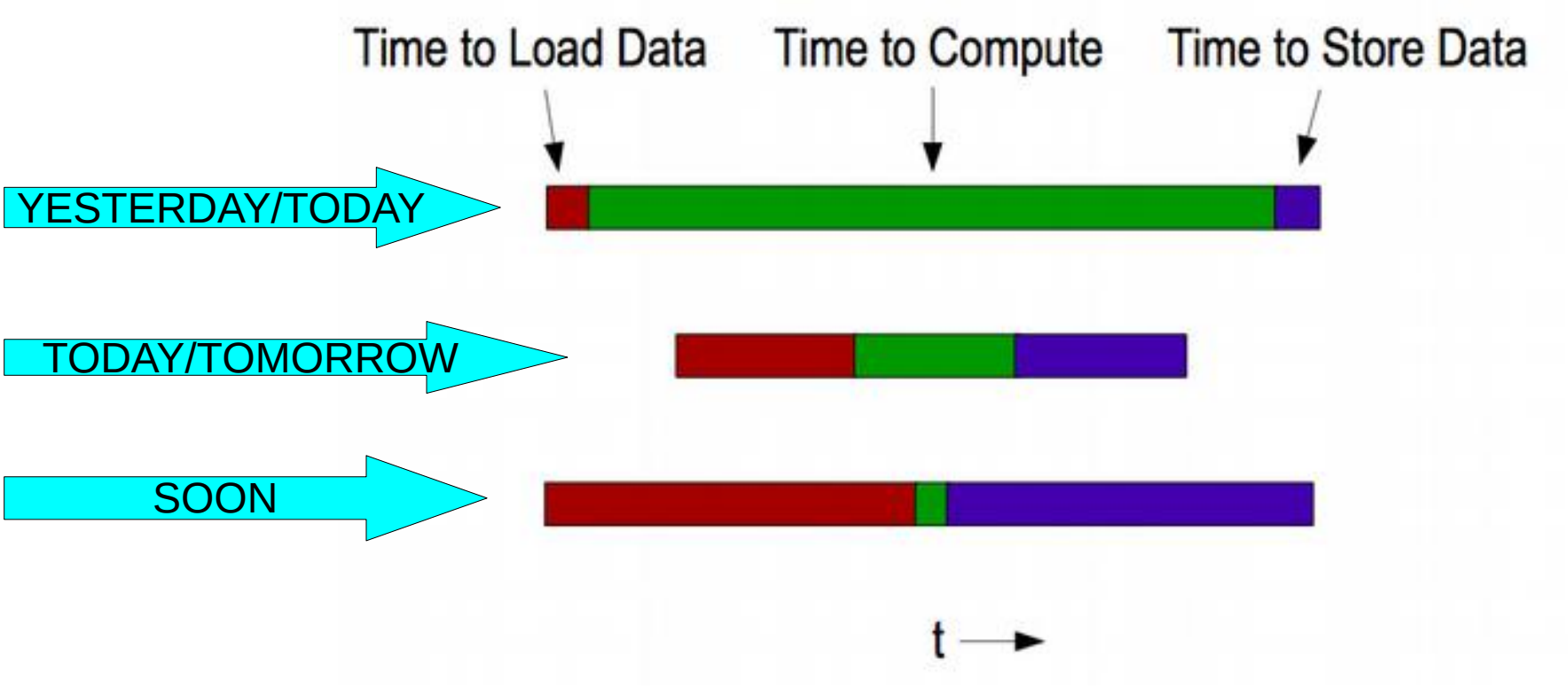
Preprocessing/Post-processing phases..

- Pre-/post processing:
  - Preparing input
  - Processing output
- These phases are becoming comparable or even larger in time than the computational phases..

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

Do we need to start optimizing I/O ?



We are not counting here pre/post processing phases !!

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 ?)

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.

Application dataset complexity vs I/O

- I/O systems have very simple data models
  - Tree-based hierarchy of containers
  - Some containers have streams of bytes (files)
  - Others hold collections of other containers (directories or folders)
- Applications have data models appropriate to domain
  - Multidimensional typed arrays, images composed of scan lines, variable length records
  - Headers, attributes on data
- How to map from one to the other ?

Challenges in Application I/O

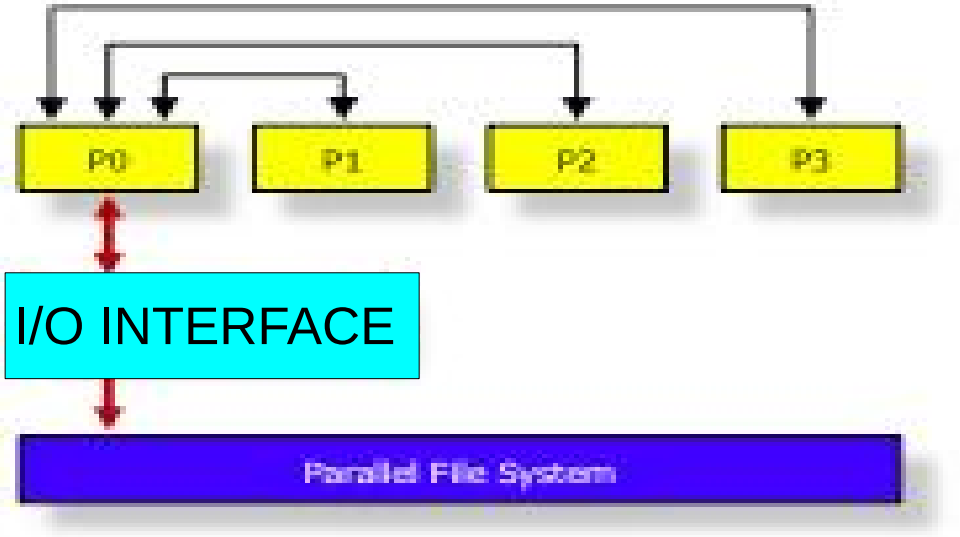
- Leveraging aggregate communication and I/O bandwidth of clients
  - but not overwhelming a resource limited I/O system with uncoordinated accesses!
- Limiting number of files that must be managed
  - Also a performance issue
- Avoiding unnecessary post-processing
- Often application teams spend so much time on this that they never get any further:
  - Interacting with storage through convenient abstractions
  - Storing in portable formats

Parallel I/O software is available to help fixing ALL these problem,

How to perform input/output on HPC ?

Serial I/O : spokeperson

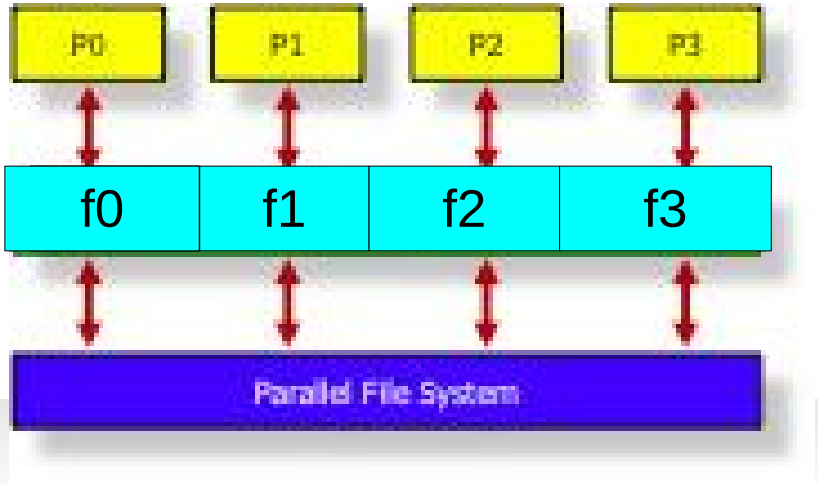
- One process performs I/O.
  - Data Aggregation or Duplication
  - Limited by single I/O process.
- Simple solution, easy to manage, but Pattern does not scale.
  - Time increases linearly with amount of data.
  - Time increases with number of processes.



Parallel I/O: File-per-Process

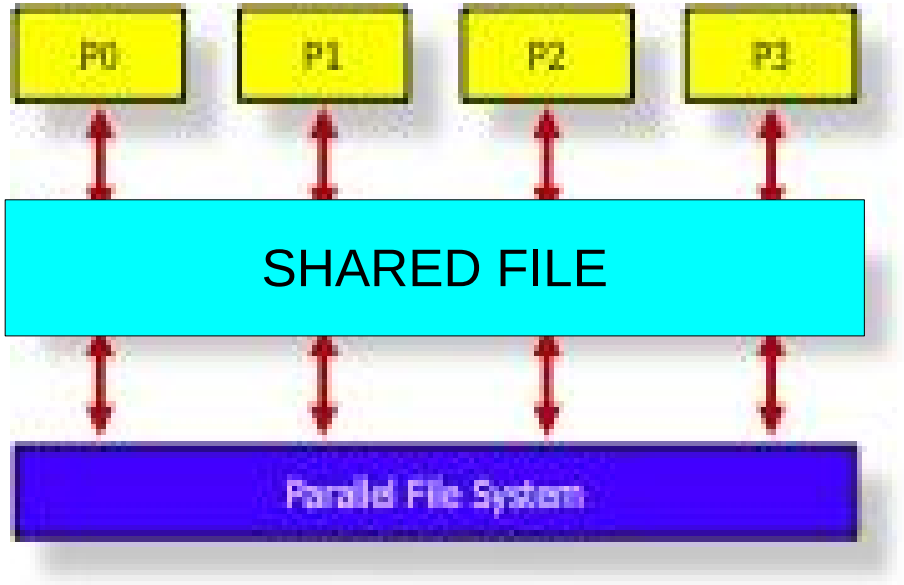
All processes perform I/O to individual files.

- 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 ?



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..

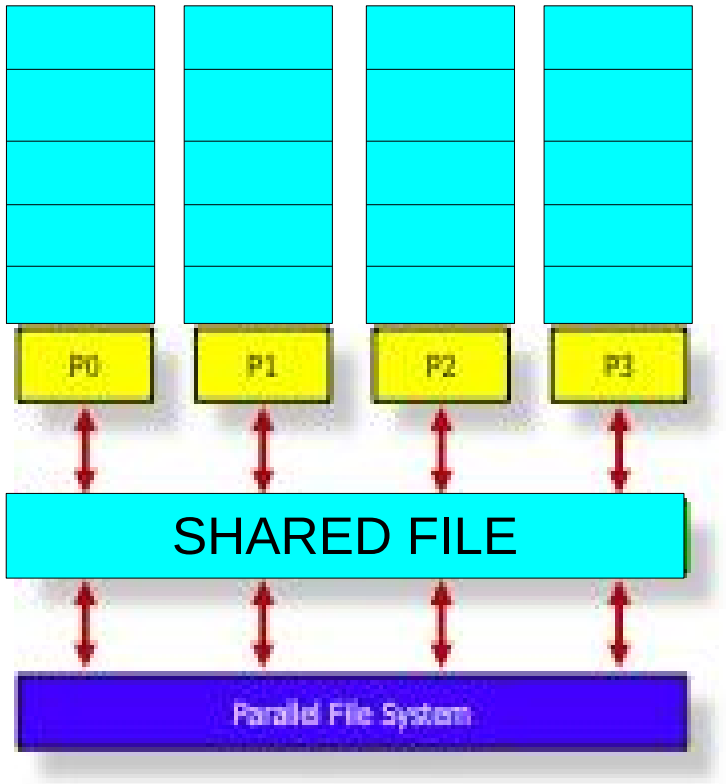


What does Parallel I/O mean ?

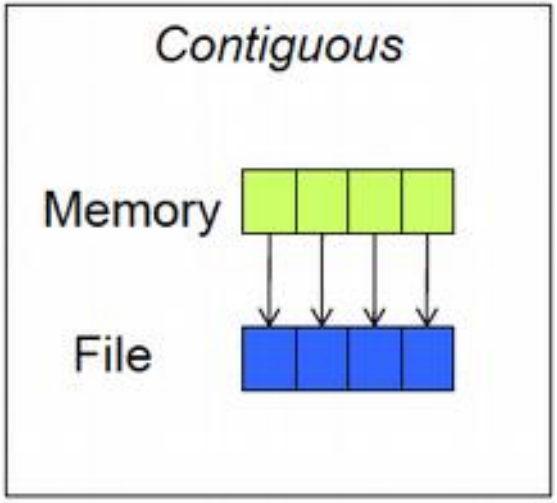
- At the program level:
  - Concurrent reads or writes from multiple processes to a common file
- At the system level:
  - A parallel file system and hardware that support such concurrent access

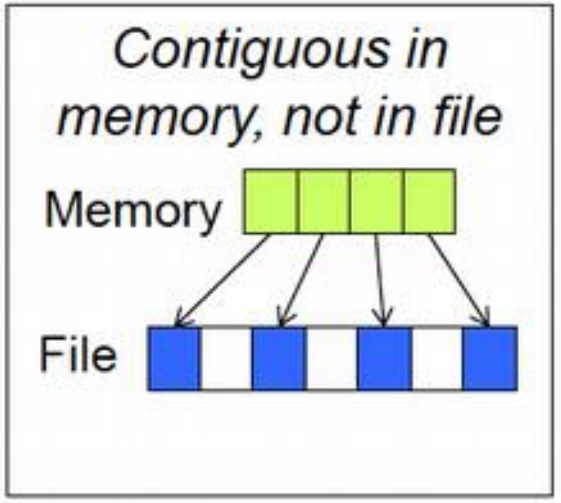
Parallel I/O on very large system..

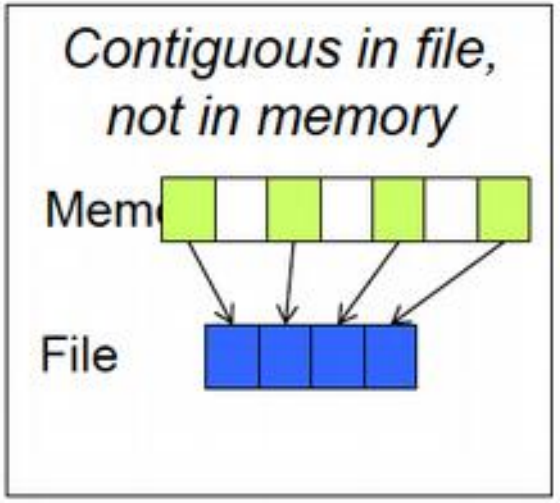
- Accessing a shared filesystem from large numbers of processes could potentially overwhelm the storage system and not only..
- In some cases we simply need to reduce the number of processes accessing the storage system in order to match number of servers or limit concurrent access.

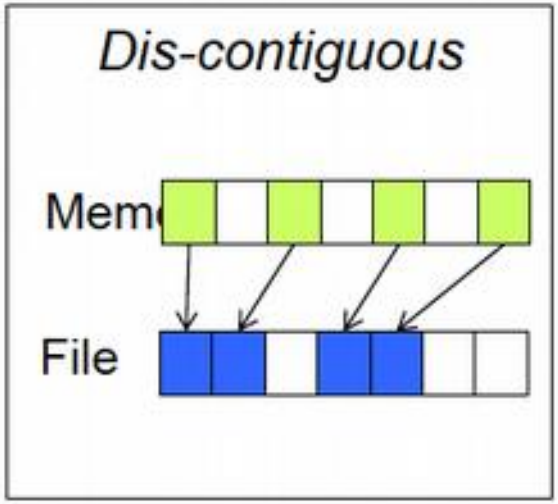


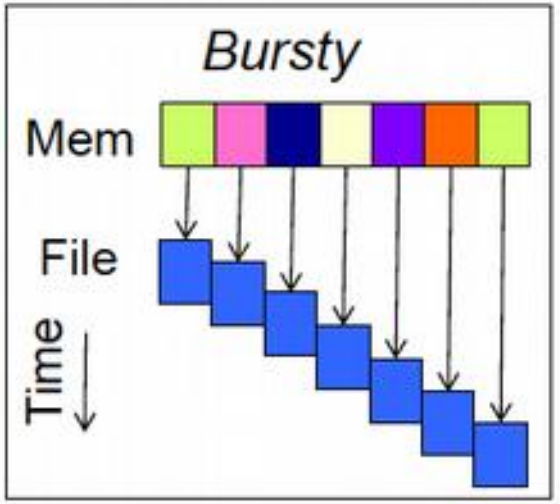
Access Patterns

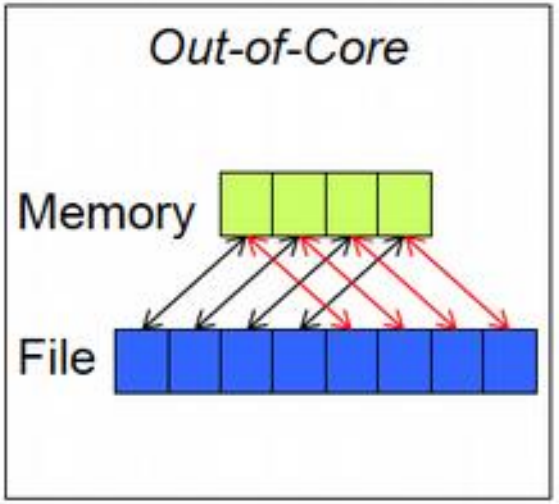




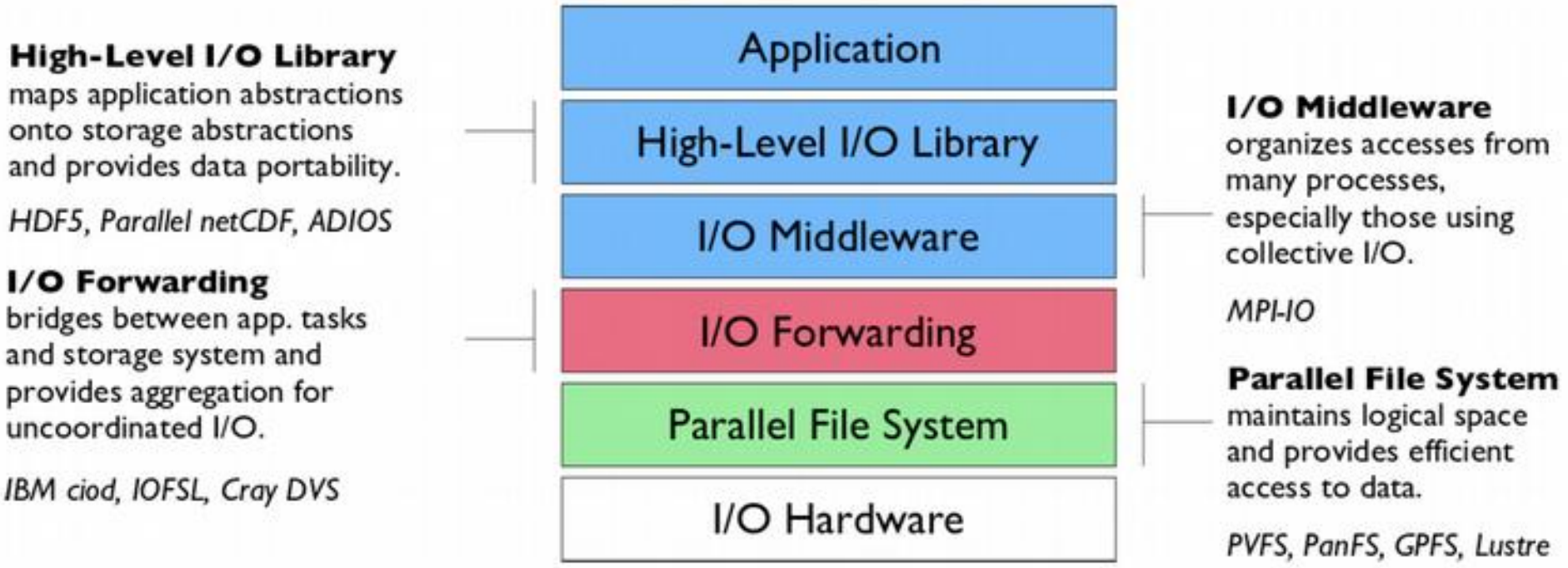








Software/Hardware stack for I/O



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