22nd IEEE International Symposium on Parallel and Distributed Computing

A new Ad-Hoc parallel file system for HPC environments based on the Expand parallel file system

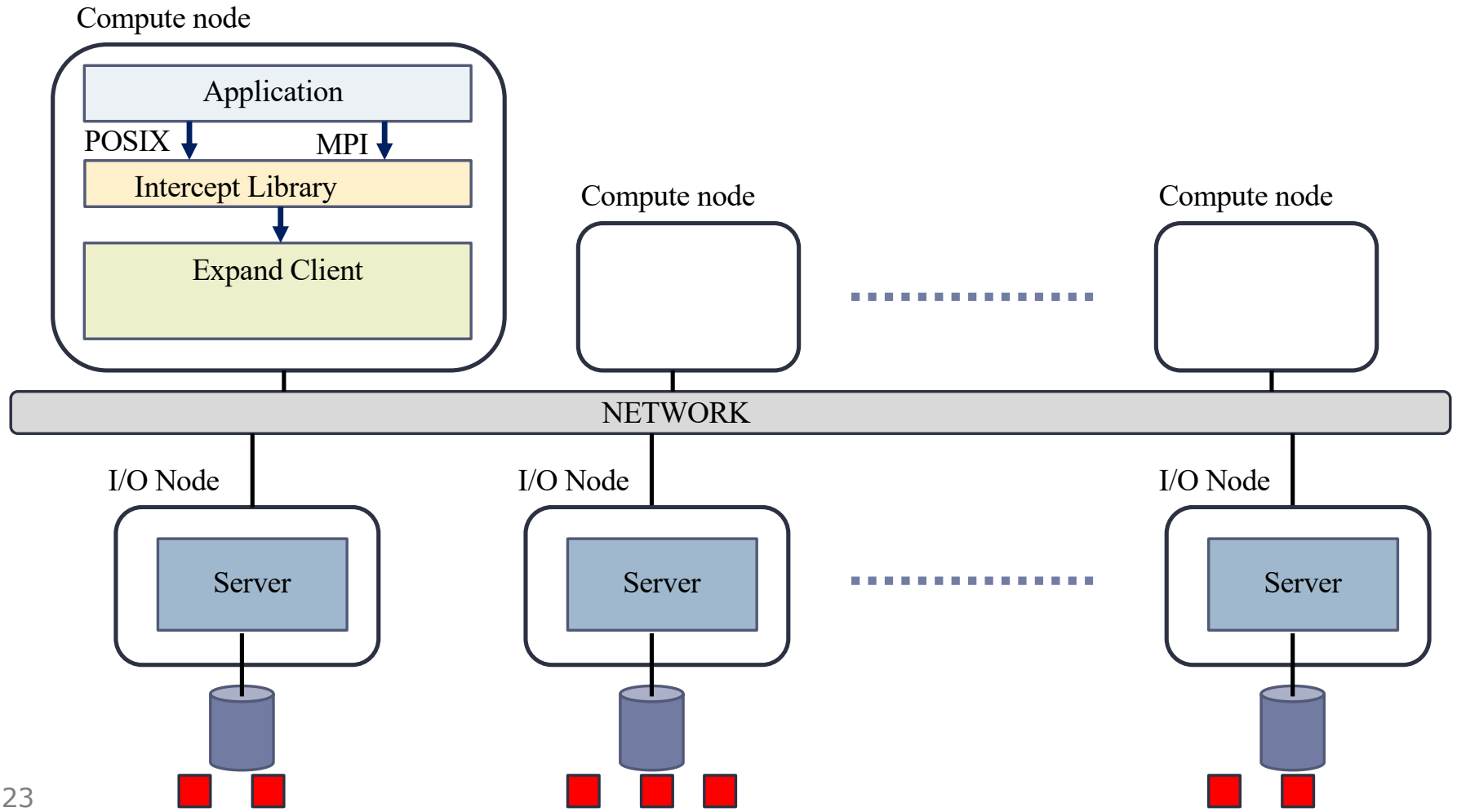
Félix García Carballeira
felix.garcia@uc3m.es

uc3m | Universidad Carlos III de Madrid

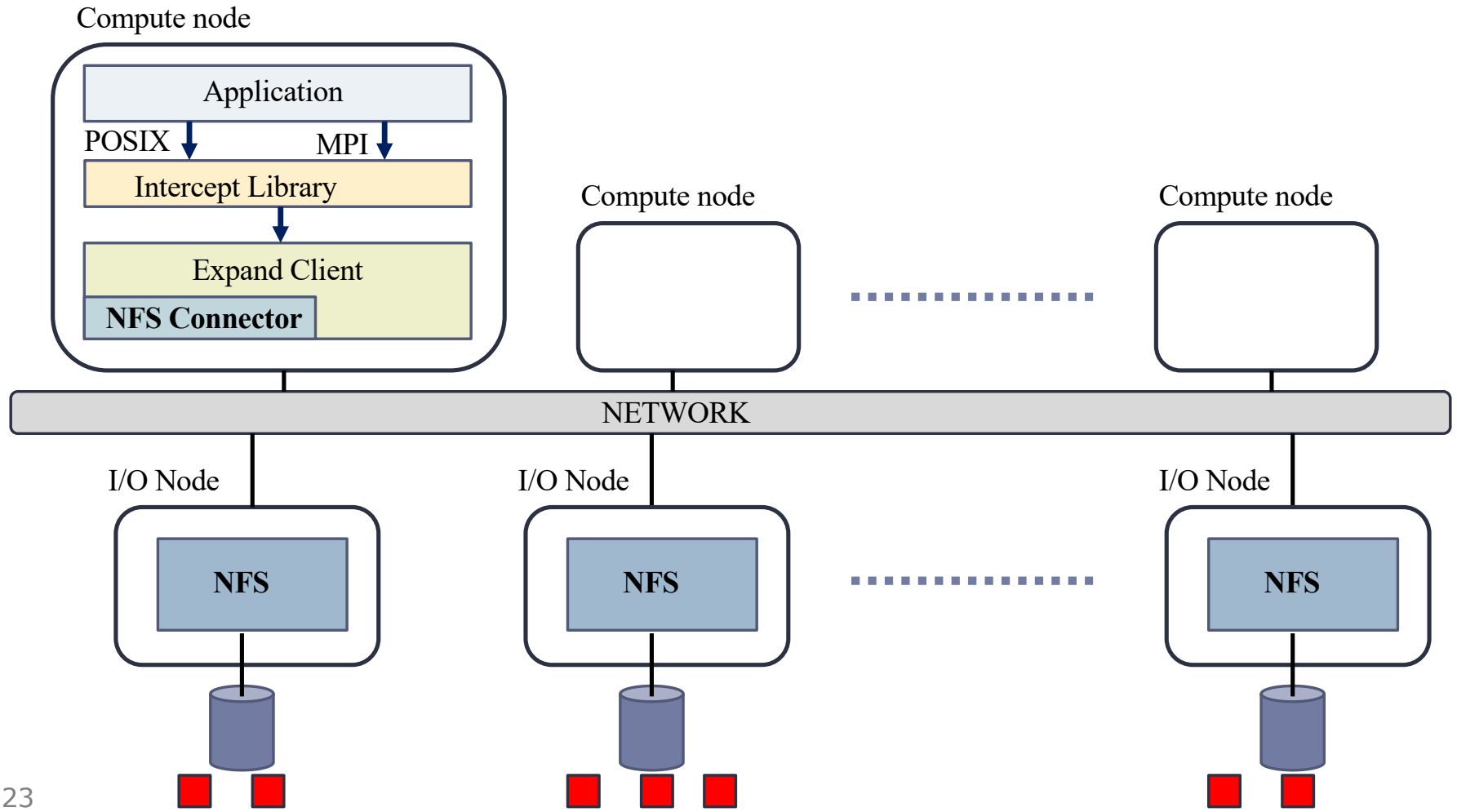
Agenda

- Introduction to Expand Parallel file system
- Expand as Ad-hoc parallel file system for large clusters and supercomputers
  - Motivation
  - Architecture
  - Data distribution
  - Metadata Management
- Performance evaluation
- Conclusions and future work

Introduction to Expand

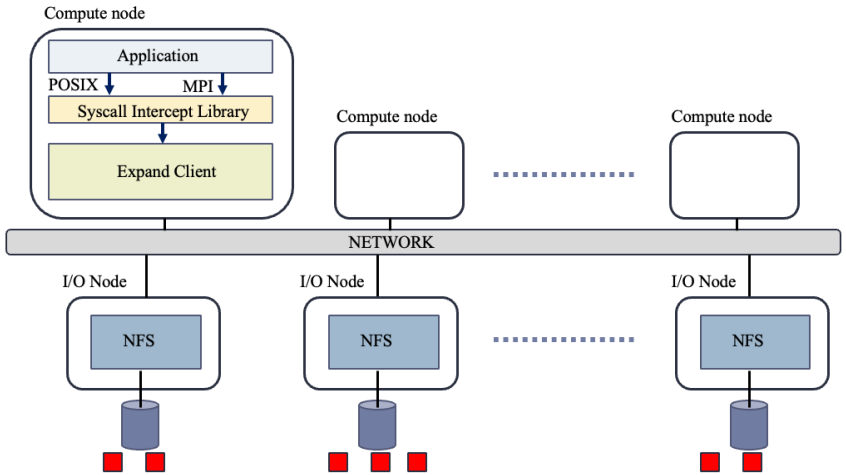


Expand: distributed systems and small clusters

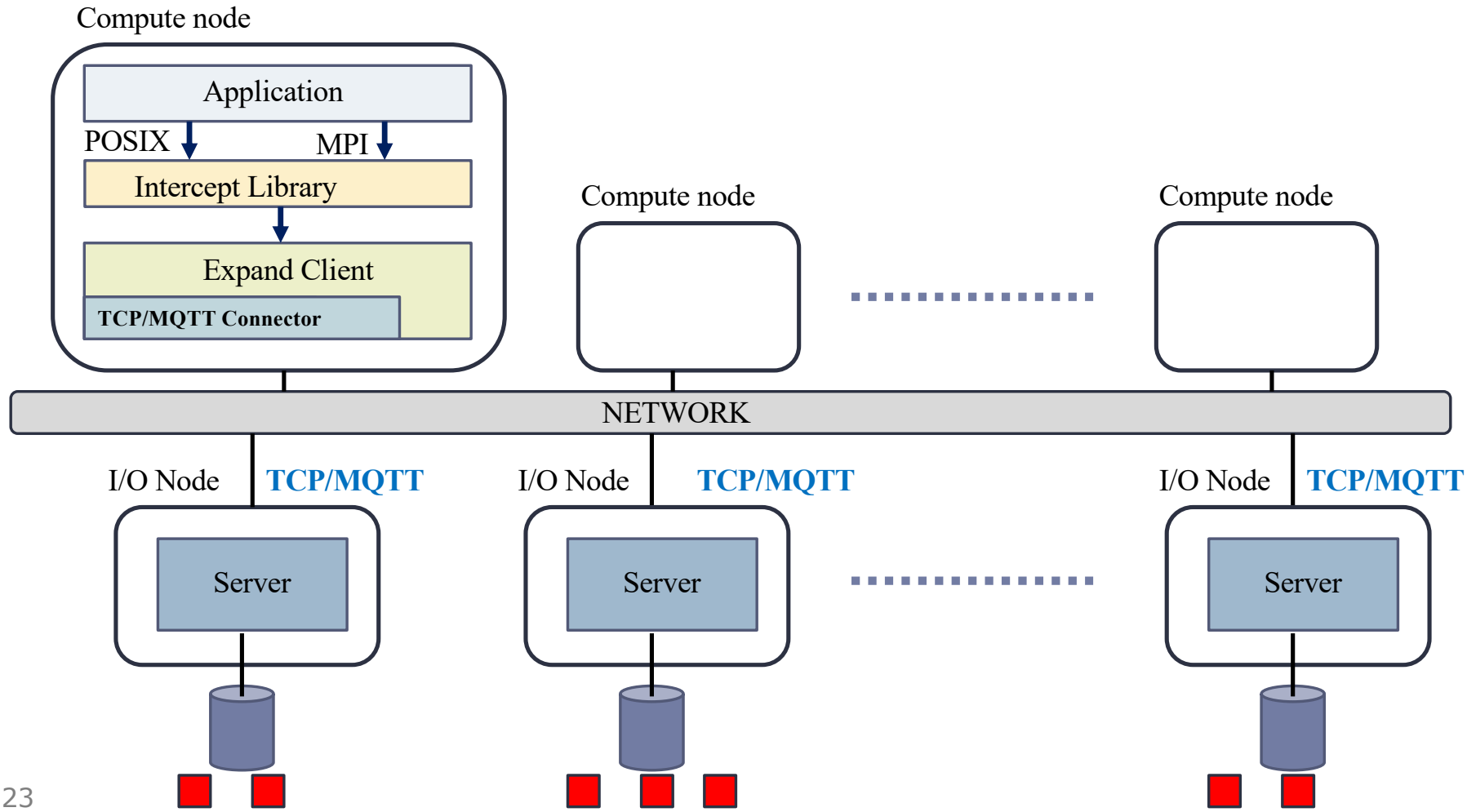


Expand: distributed systems and small clusters

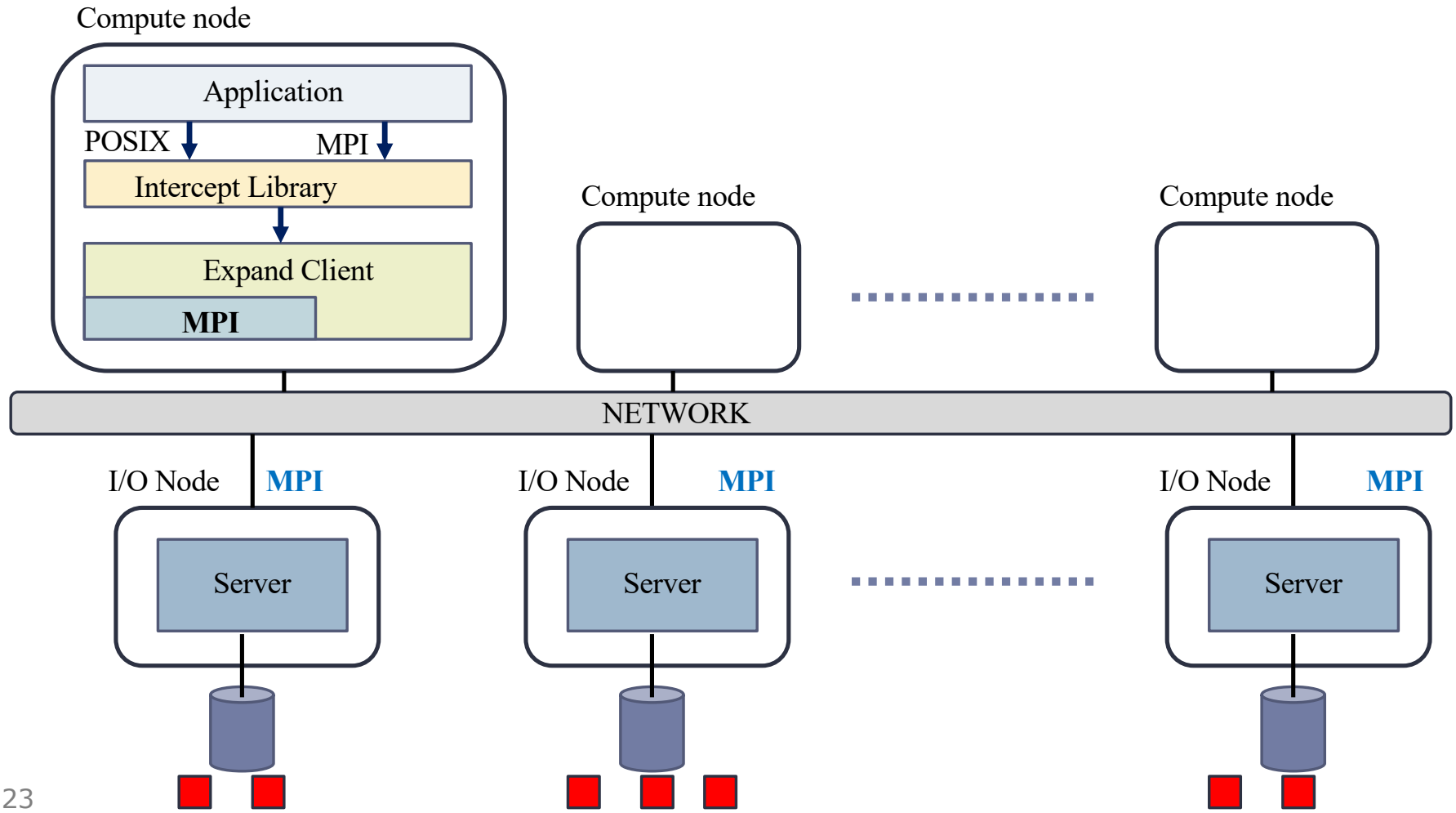
- Advantages of using NFS:
  - Standard server
  - No changes to NFS server are required
  - Independent of the operating system used in the clients
  - Allows the using of servers with different architectures and operating systems
  - The parallel file system construction is greatly simplified
  - NFS is very familiar to users: easy configuration



Expand: IoT environments

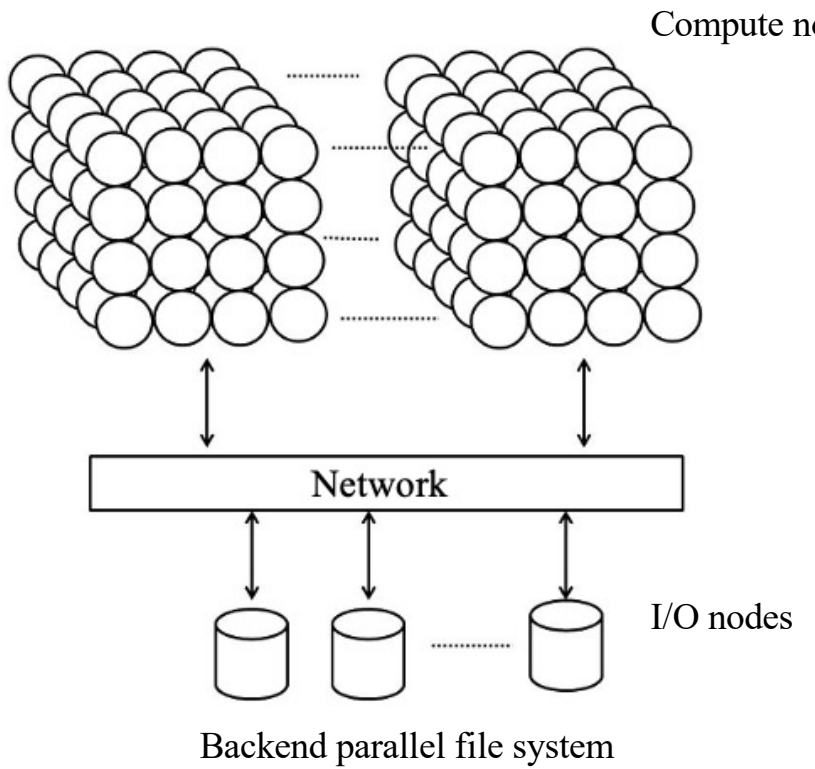


Expand: ad-hoc parallel file system for large clusters



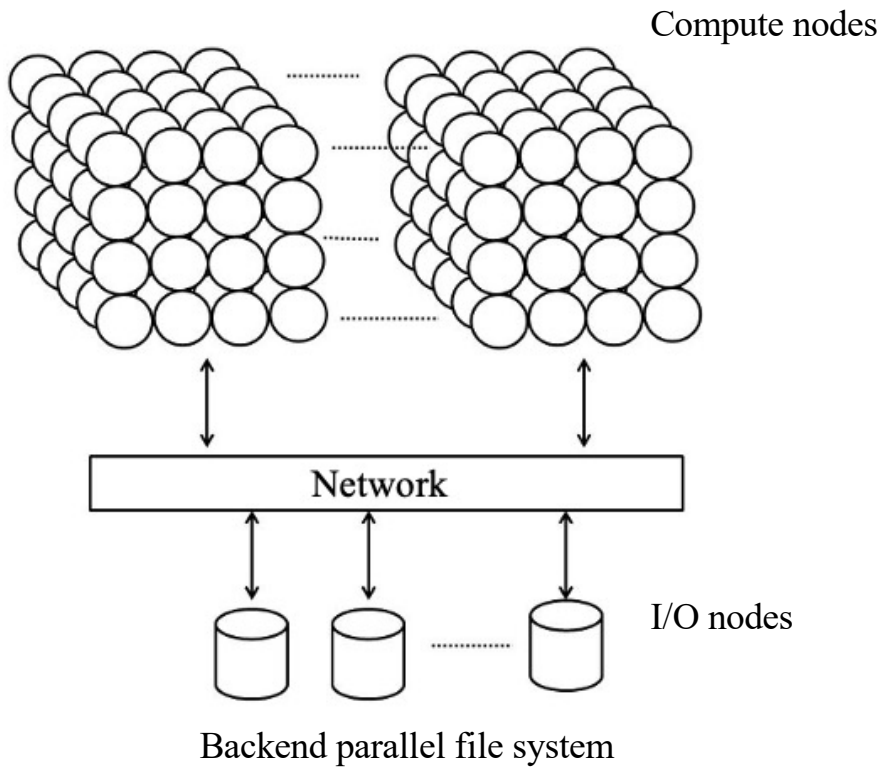
Motivation for developing an Ad-hoc file system

- Typical supercomputer architecture:
  - The number of compute nodes is much larger than the number of I/O nodes:
    - Possible bottleneck



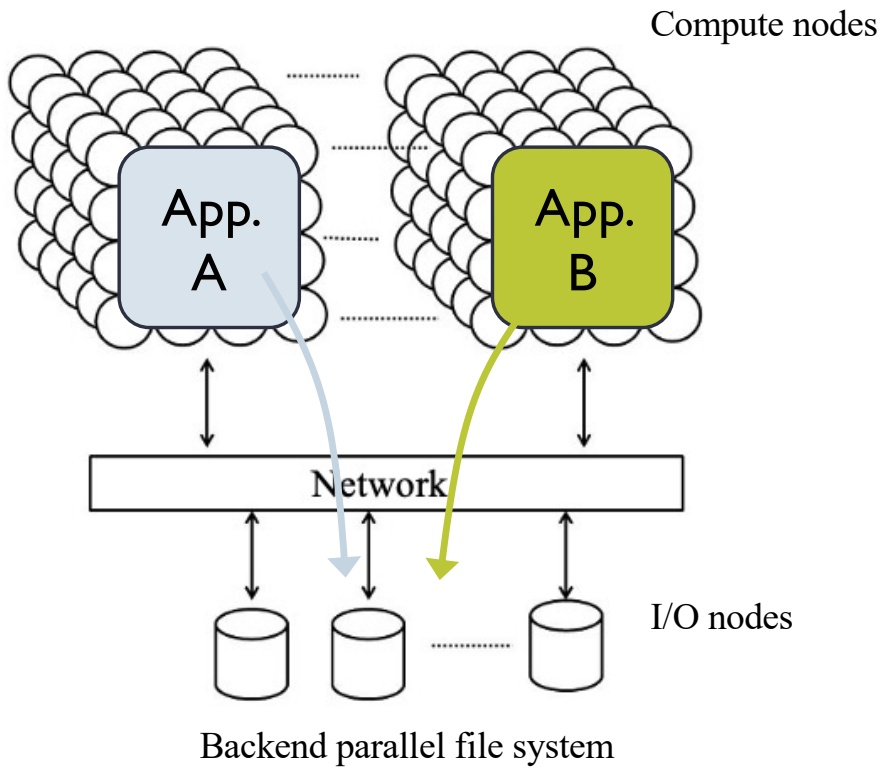
Motivation for developing an Ad-hoc file system

- Typical supercomputer architecture:
  - The number of compute nodes is much larger than the number of I/O nodes:
    - Possible bottleneck
  - Data away from applications:
    - Use of network
    - Reduces data access performance

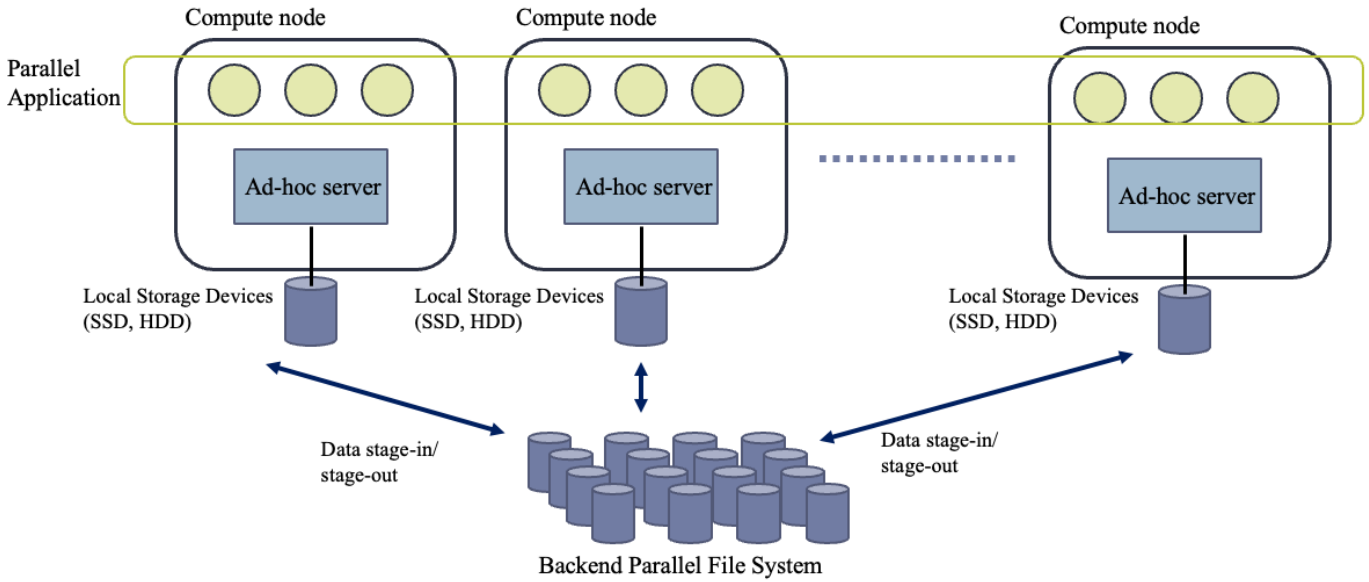


Motivation for developing an Ad-hoc file system

- Typical supercomputer architecture:
  - The number of compute nodes is much larger than the number of I/O nodes:
    - Possible bottleneck
  - Data away from applications:
    - Use of network
    - Reduces data access performance
  - Different applications running at the same time:
    - File system conflicts and interferences

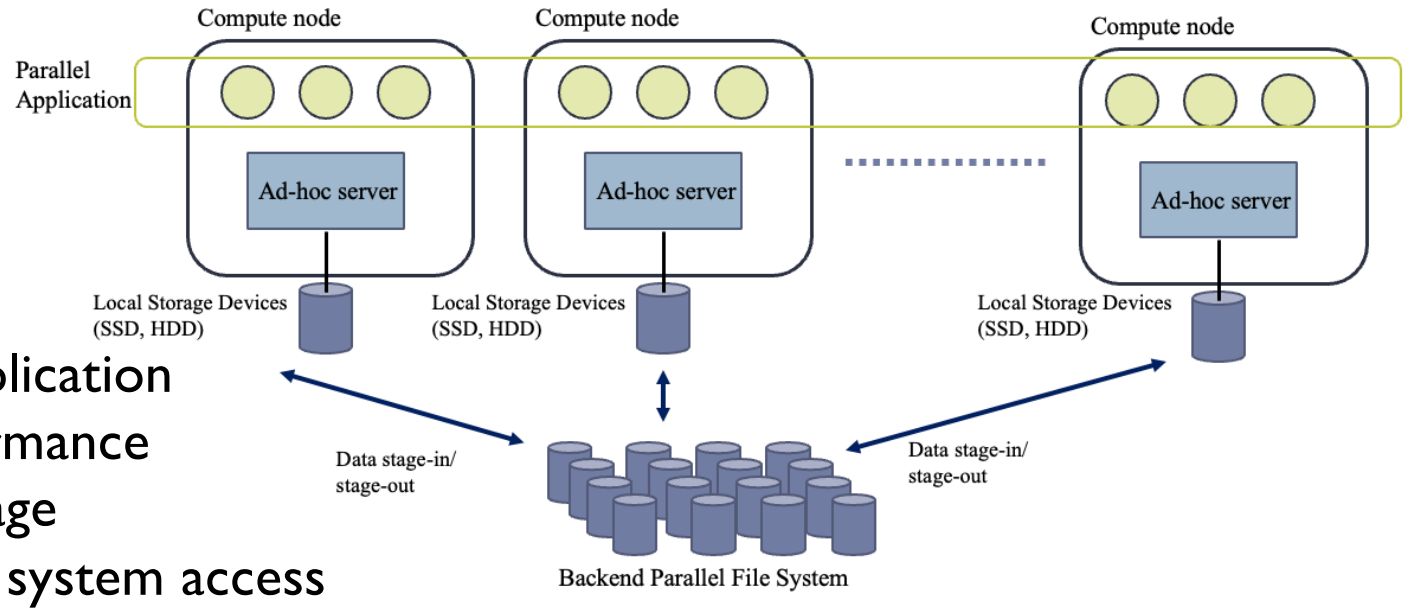


Ac-hoc file systems



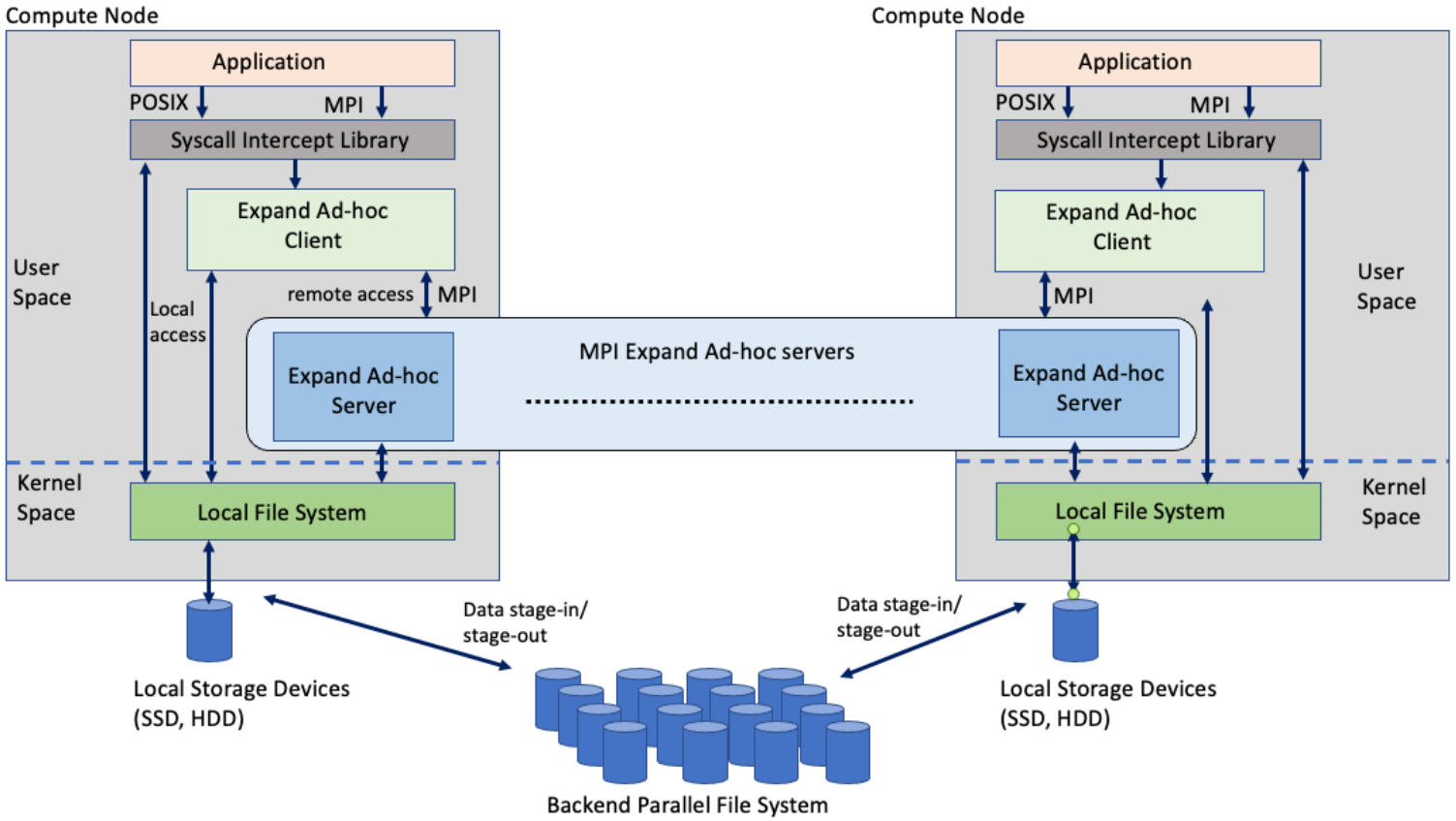
- Temporary storage system for single applications or workflows of applications that use the available storage of compute nodes (HDD, SSD or SHM)
- The servers are deployed on the application's compute nodes

Ac-hoc file systems: **advantages**



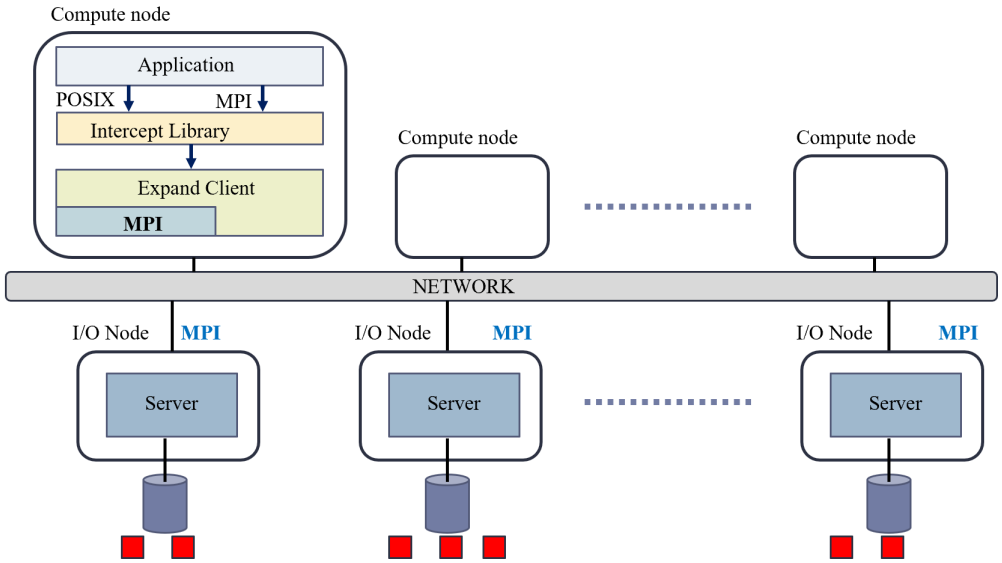
- Data close to the application
  - Increase the performance
- Reduces network usage
- Reduces backend file system access
  - Reduces bottlenecks
- Temporary data (checkpoints, intermediate data) does not need to be stored in the backend file system
- In application workflows data remains for the different workflow processes

Architecture of the Expand Ad-Hoc



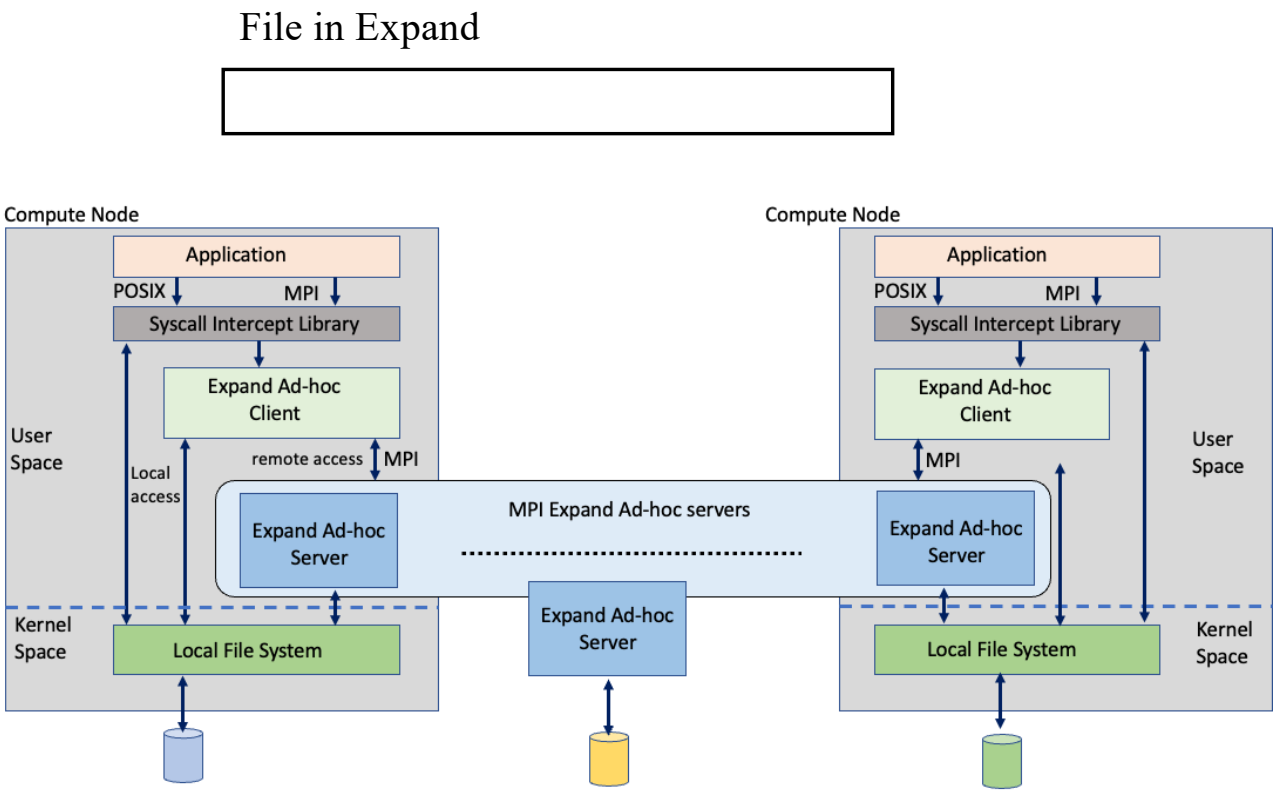
Architecture of the Expand Ad-Hoc

- Advantages of using MPI:
  - Standard interface and portable to different platforms
  - Offers good performance in clusters



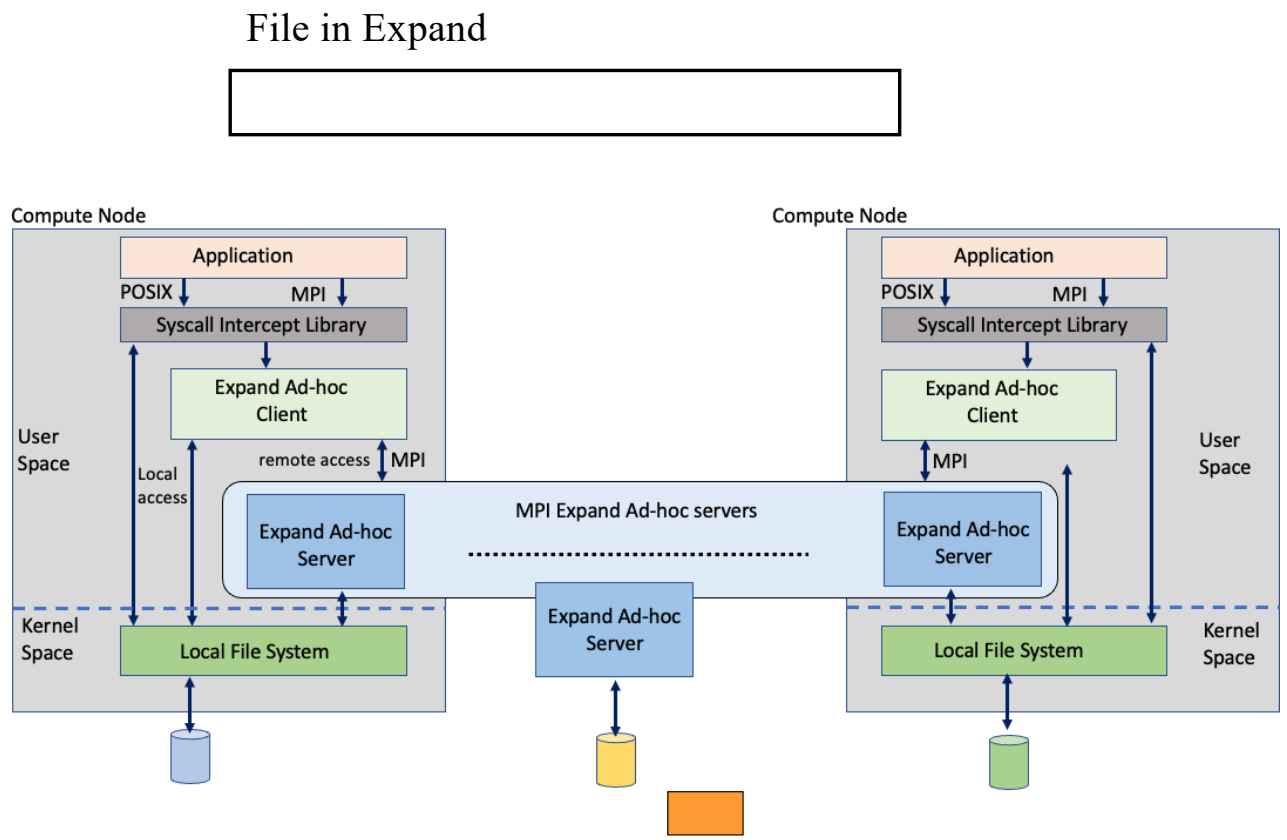
Data distribution

- File in Expand:
  - Metadata subfile
  - Several data subfiles
- Data distributed on different servers



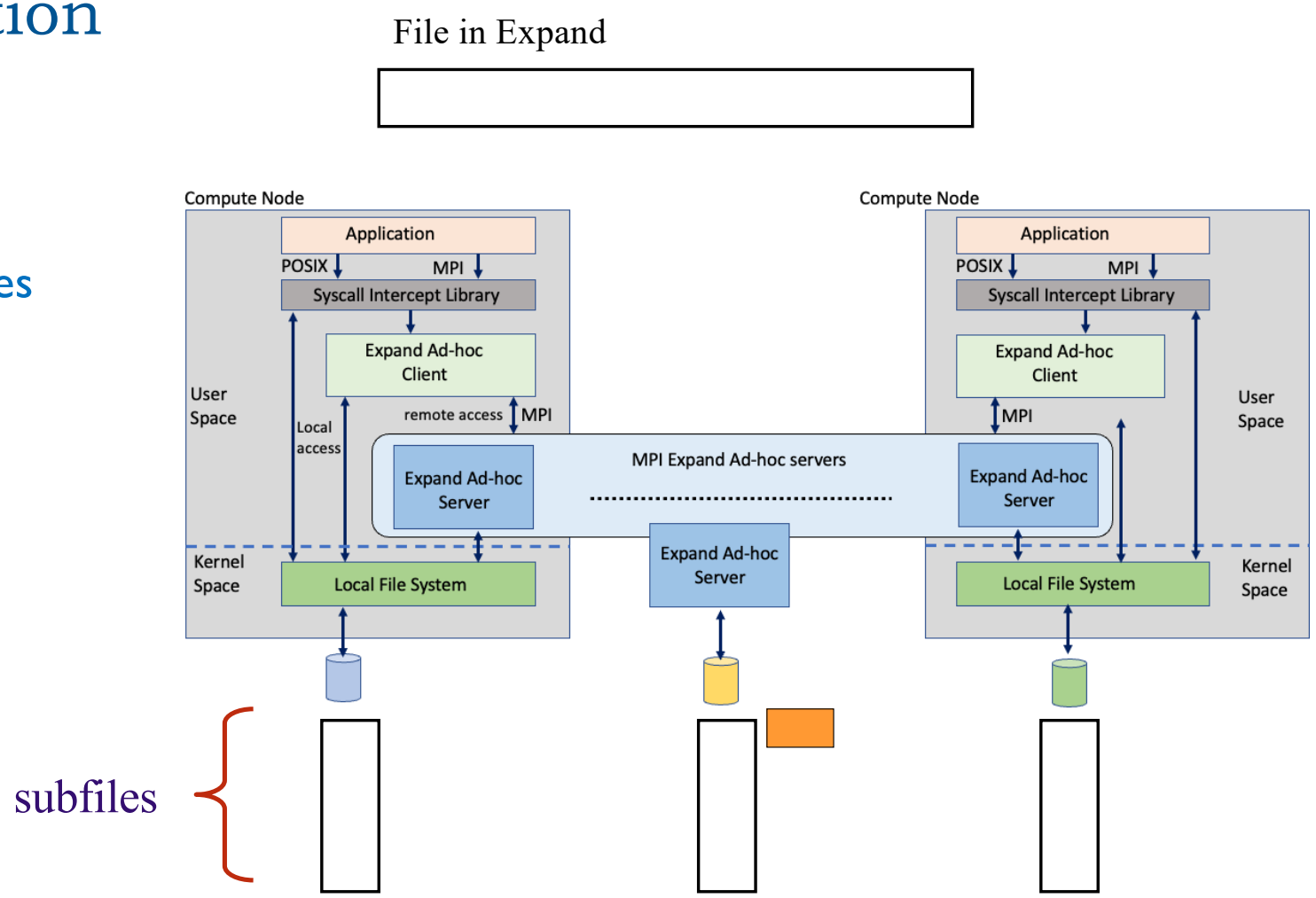
Data distribution

- File in Expand:
  - Metadata subfile
  - Several data subfiles
- Data distributed on different servers



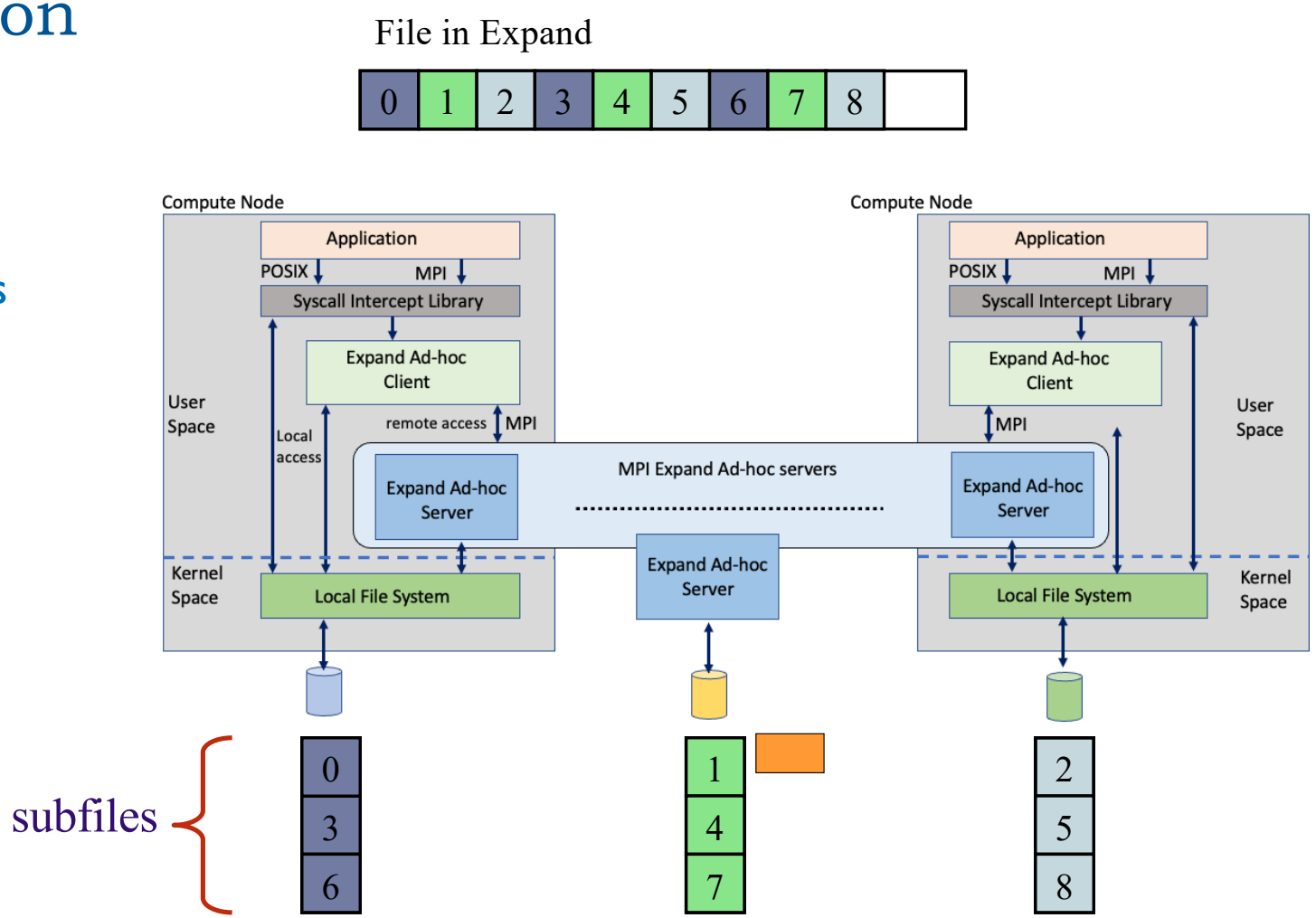
Data distribution

- File in Expand:
  - Metadata subfile
  - Several data subfiles
- Data distributed on different servers

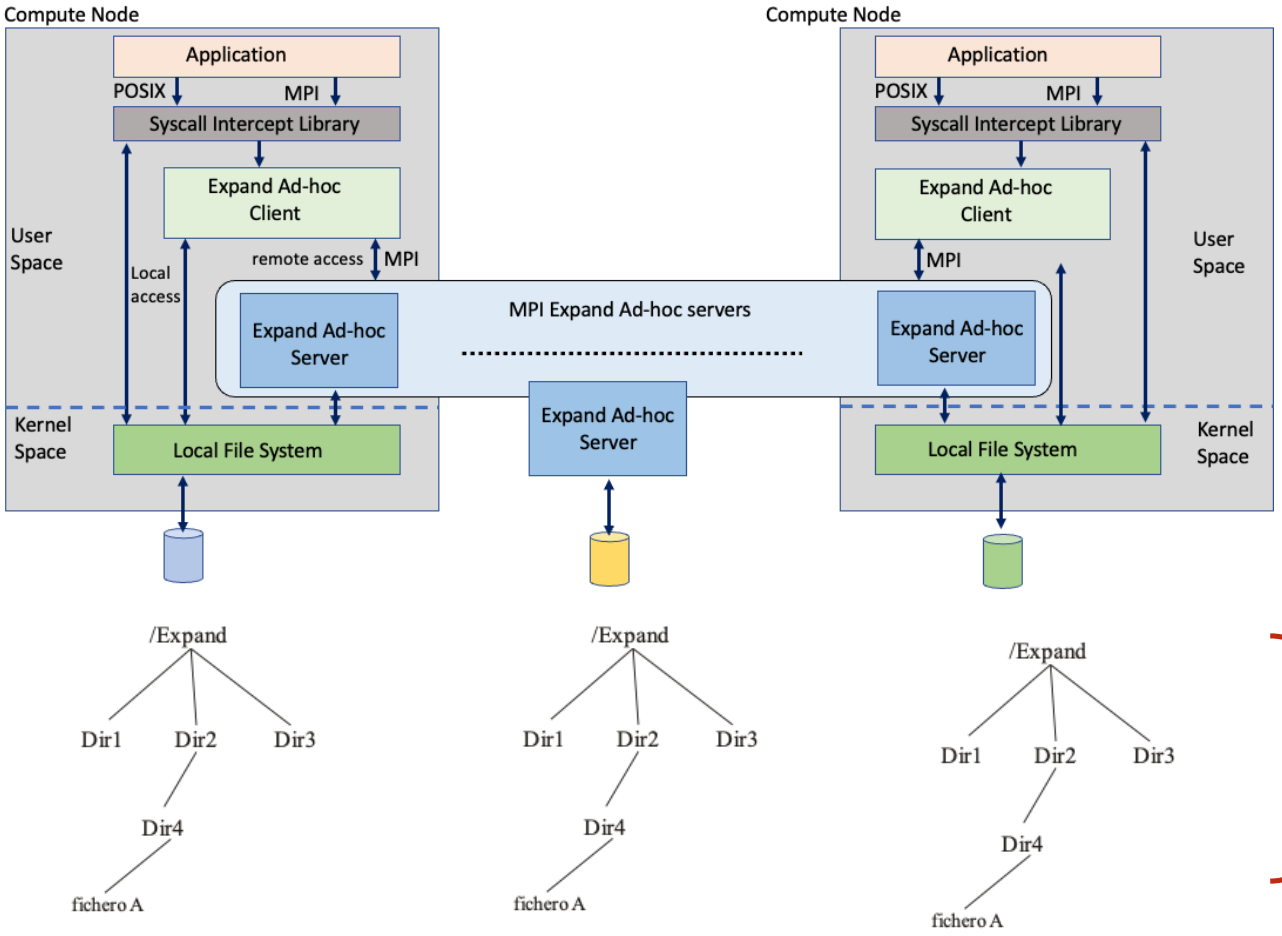


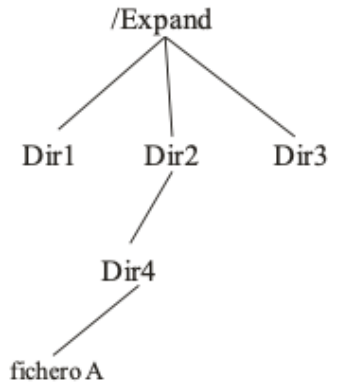
Data distribution

- File in Expand:
  - Metadata subfile
  - Several data subfiles
- Data distributed on different servers



Directory structure





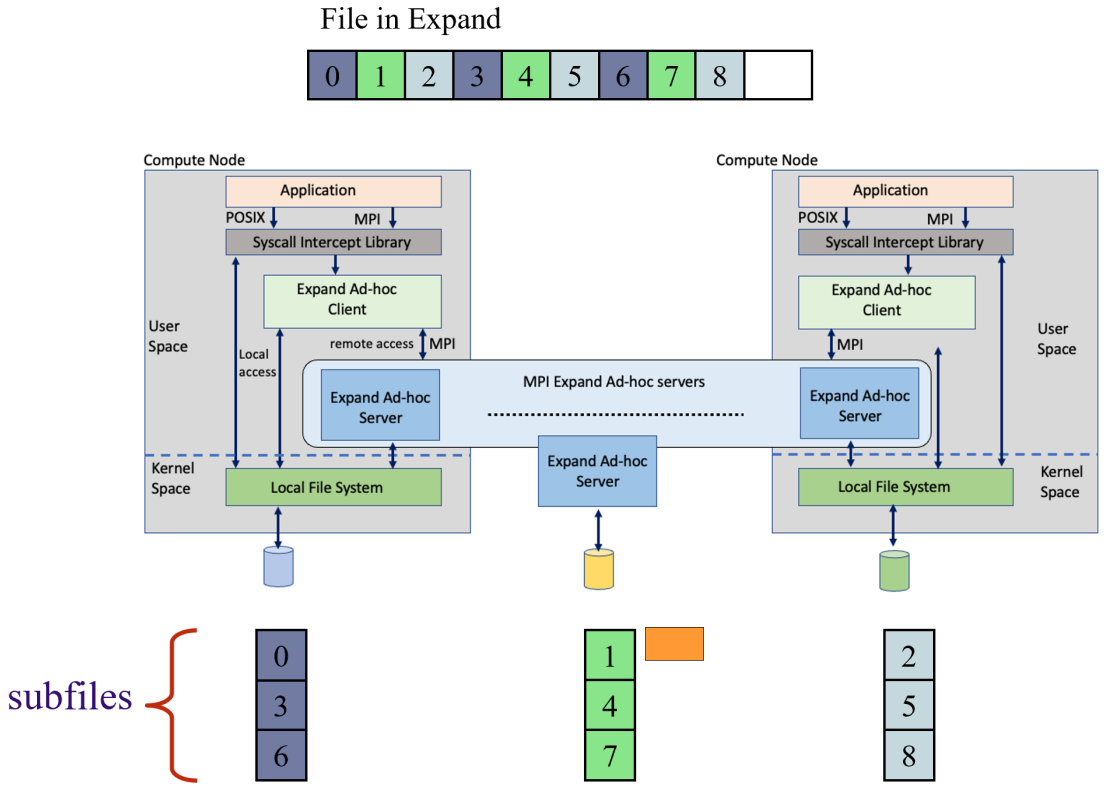
Logical View

Mapping

Physical View

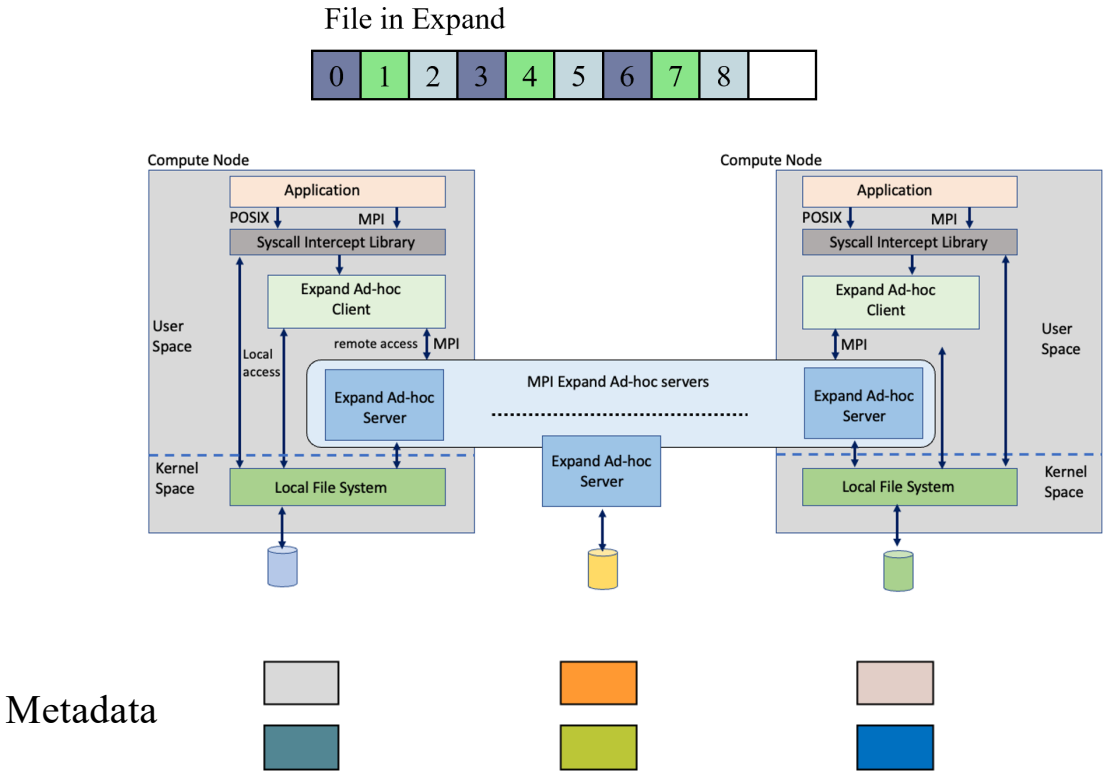
Metadata management

- Distributed metadata management:
  - Two levels
  - No locks
  - No metadata manager
- Metadata distributed among servers:
  - Master node
  - Hash function(name)
  - Load balancing



Metadata management

- Distributed metadata management:
  - Two levels
  - No locks
  - No metadata manager
- Metadata distributed among servers:
  - Master node
  - Hash function(name)
  - Load balancing



Evaluation

- **Benchmark used: IOR**
  - Open-source benchmark
  - A popular and effective way to evaluate the performance of distributed and parallel file systems is by using a variety of input/output (I/O) loads
- IOR access pattern: file sharing
  - All processes share the same file
  - Challenge for most parallel file systems

Evaluation

- Platform:
  - MareNostrum 4. Barcelona Supercomputing Center (BSC)
- Main properties:
  - *Nodes*: 3,456
  - *Total cores*: 165,888
  - *Main memory*: 384.75 TB
  - *SSD:* 240 GB
  - *Interconnection networks*: 100Gb Intel Omni-Path (Full-Fat Tree)
  - *Peak Performance*: 11.15 Petaflops
  - *Parallel file system*: GPFS

Evaluation

- File systems evaluated:
  - **Expand Ad-Hoc** (512 KB of block size)
  - **GekkoFS** (512 KB of block size)

> M.-A. Vef, N. Moti, T. Süß, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann, "GekkoFS: a temporary distributed file system for HPC applications," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 319–324.

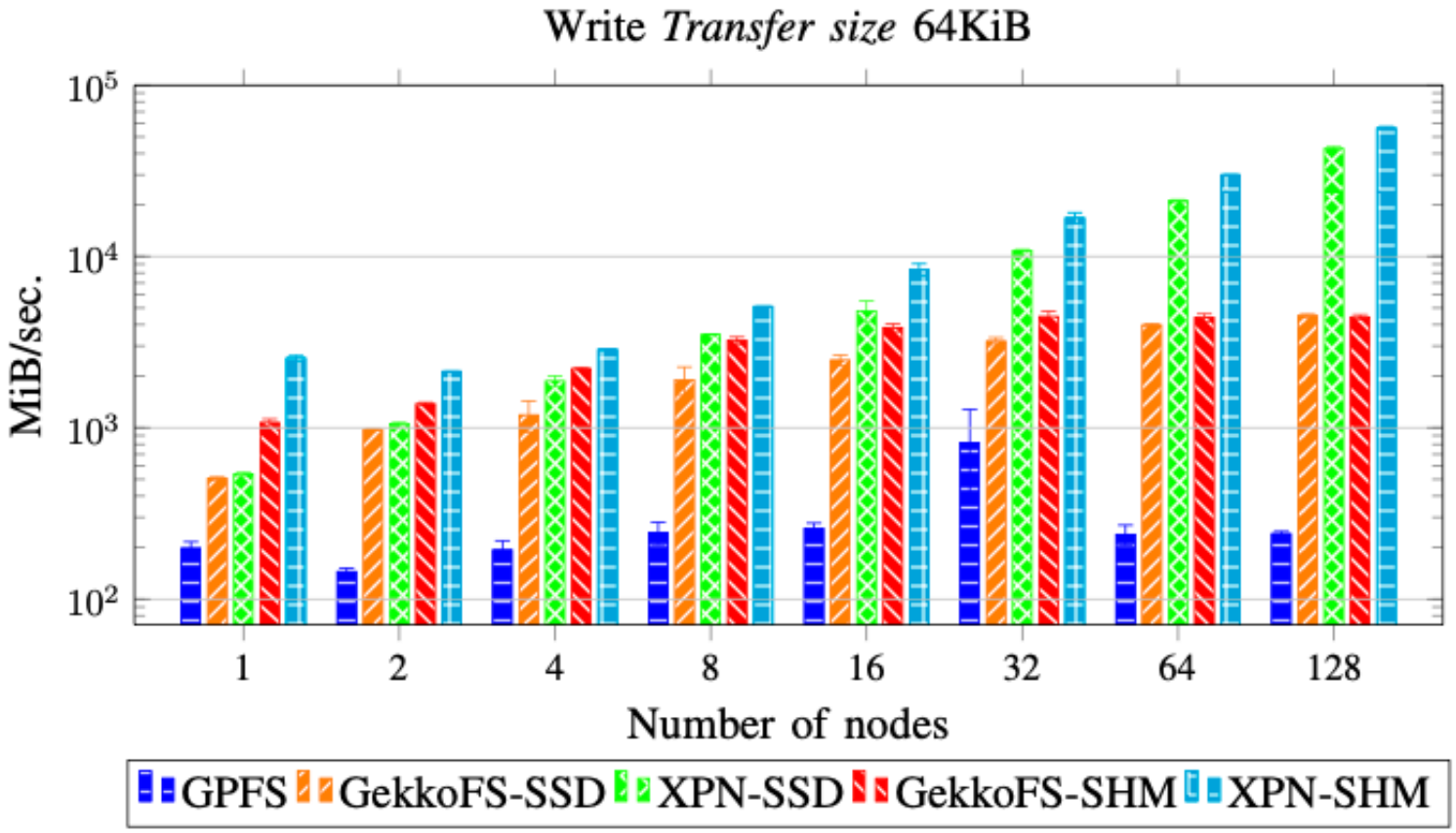
  - **GPFS**: parallel file system used in MareNostrum 4

Evaluation

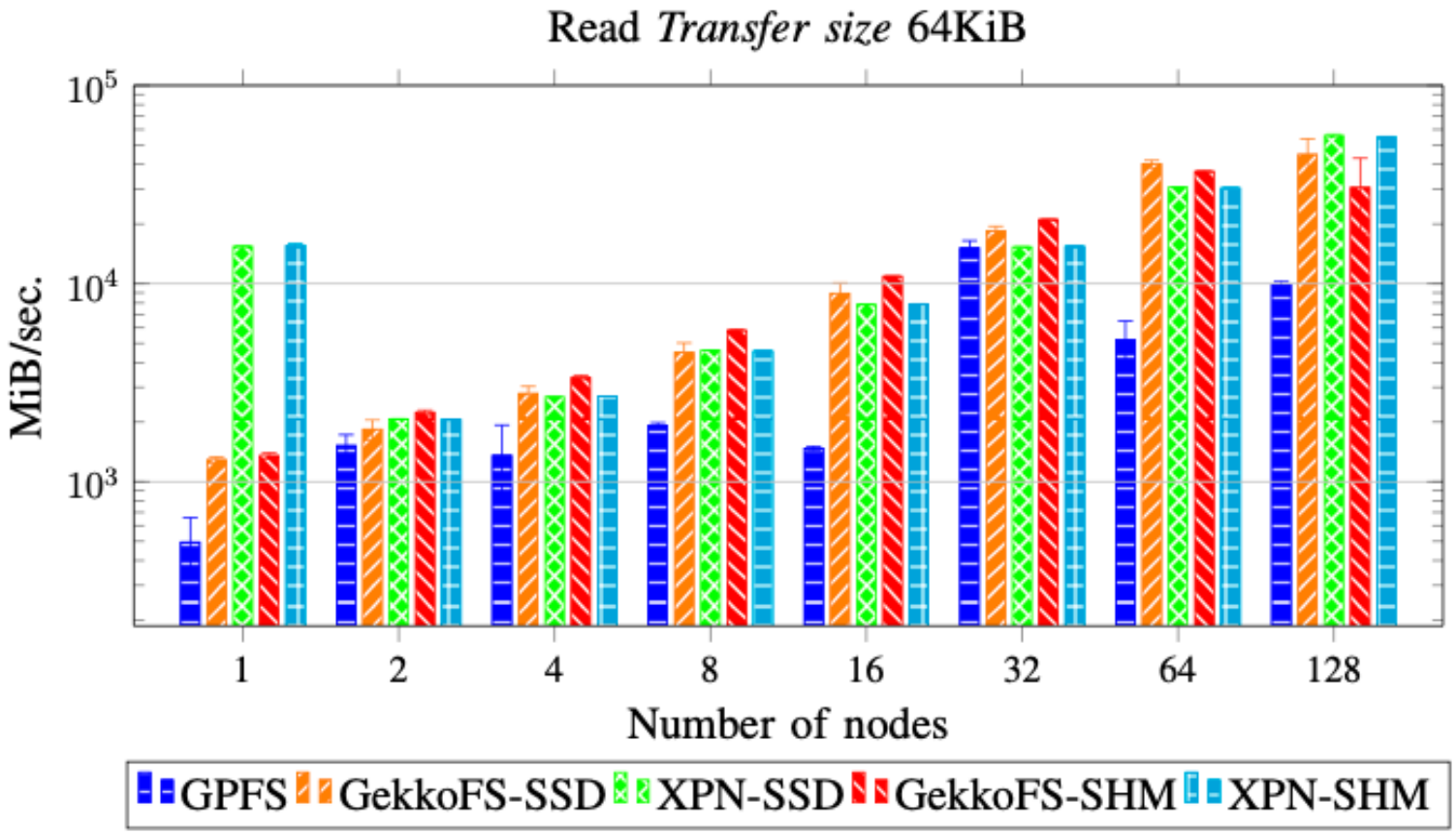
- Configuration:
  - Compute nodes: 1, 2, 4, ... up to 128 compute nodes
  - Local storage: SSD and Shared Memory (SHM) for GekkoFS and Expand
  - Transfer size used in IOR: 64 KiB, 512 KiB, and 1 MiB.
  - Client processes per compute node: 8
  - Operations: read and write in parallel on a shared file
  - Size written by each client: 4 GiB (resulting in a 4 TiB)
  - Number of processes: 8 up to 1024 processes

- All results in logarithmic scale
- All tests have been executed 10 times and the average results and standard deviation are shown in all figures.

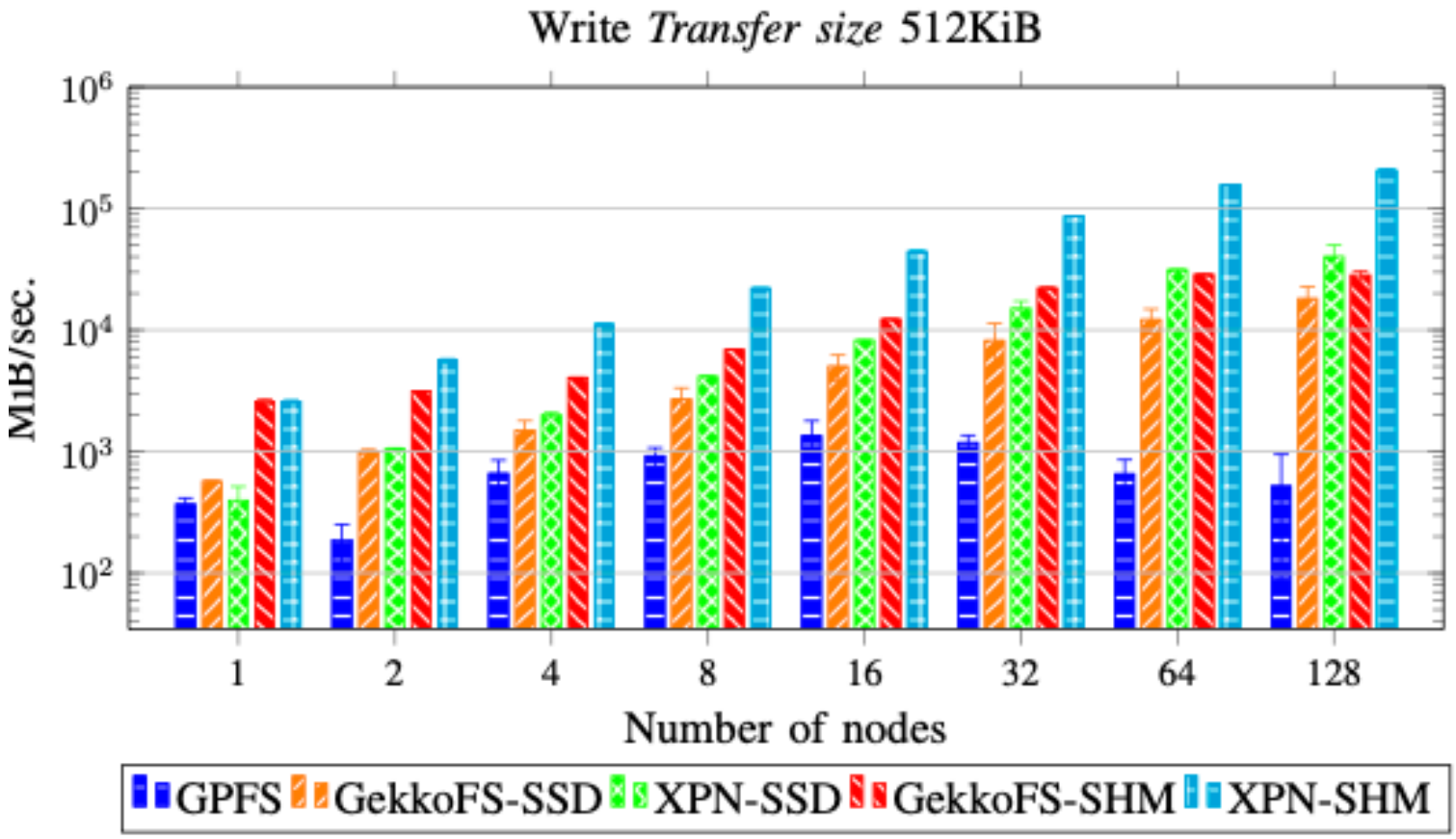
Results


Write *Transfer size* 64KiB
MiB/sec.
10^5
10^4
10^3
10^2
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
XPN-SHM

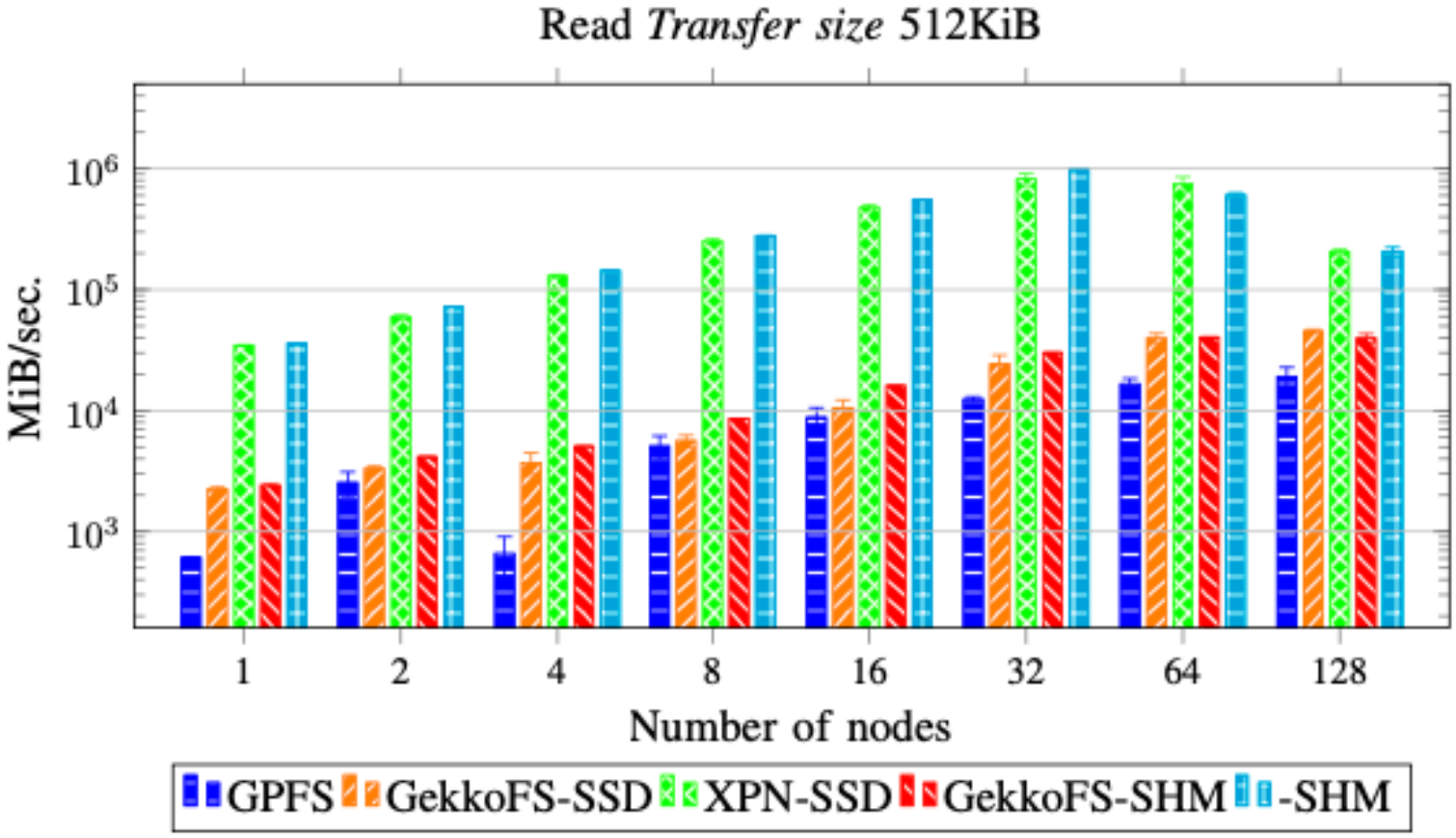
Results


Read *Transfer size* 64KiB
MiB/sec.
10^5
10^4
10^3
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
XPN-SHM

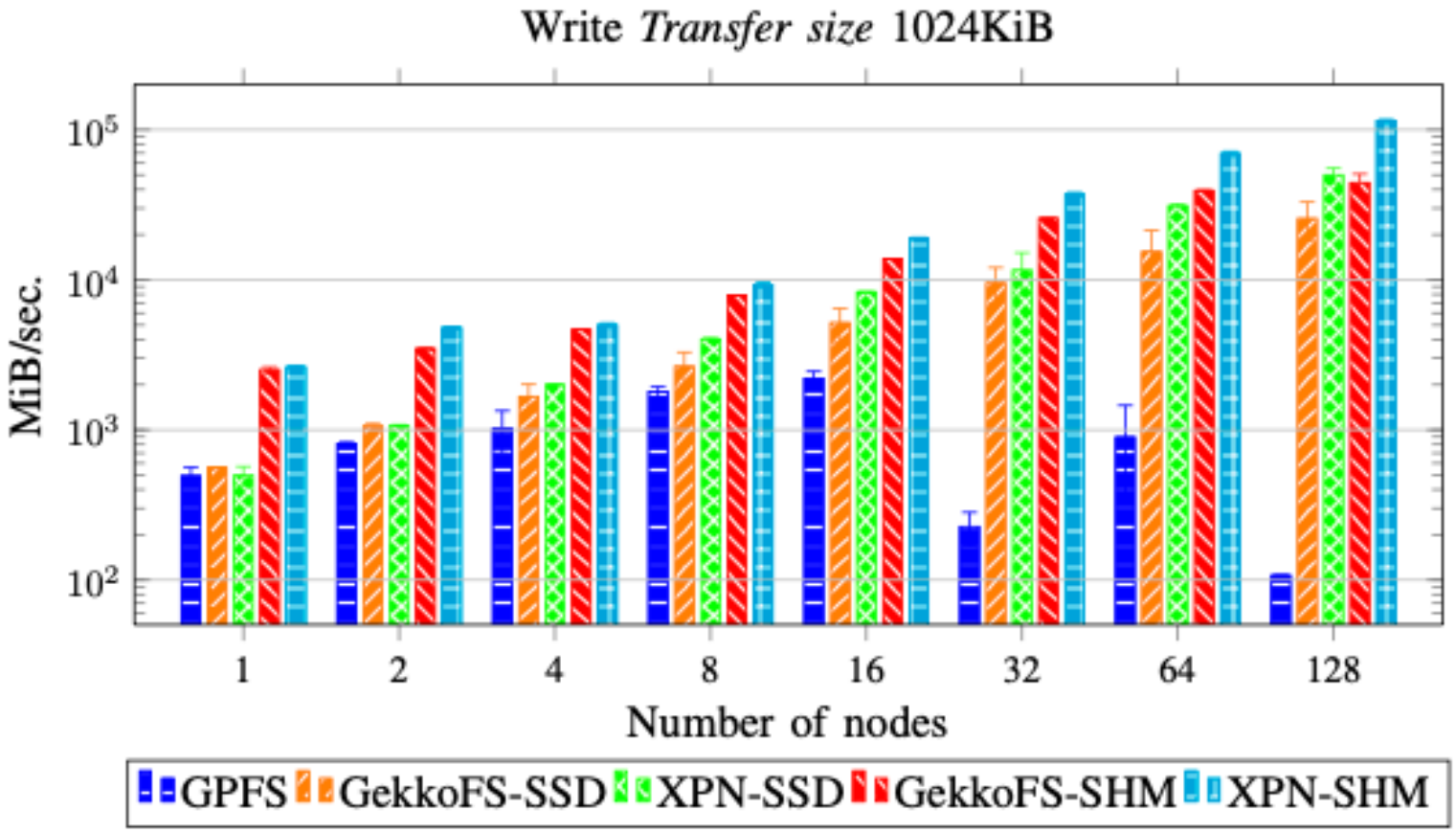
Results


Write *Transfer size* 512KiB
MiB/sec.
10^6
10^5
10^4
10^3
10^2
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
XPN-SHM

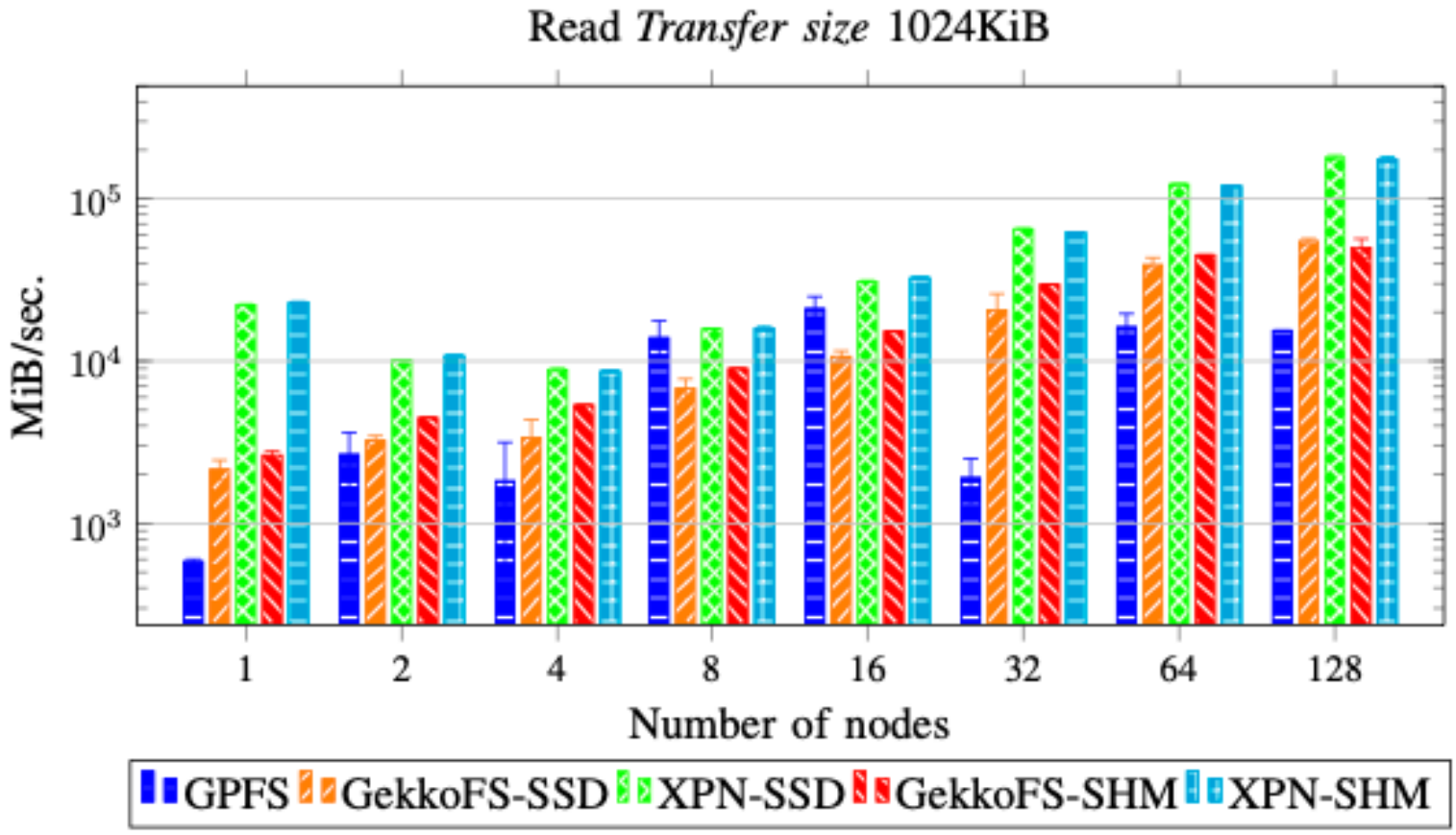
Results


Read *Transfer size* 512KiB
MiB/sec.
10^6
10^5
10^4
10^3
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
-SHM

Results


Write *Transfer size* 1024KiB
MiB/sec.
10^5
10^4
10^3
10^2
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
XPN-SHM

Results


Read *Transfer size* 1024KiB
MiB/sec.
10^5
10^4
10^3
1
2
4
8
16
32
64
128
Number of nodes
GPFS
GekkoFS-SSD
XPN-SSD
GekkoFS-SHM
XPN-SHM

Conclusions and Future Works

- Main conclusions:
  - Expand Ad-Hoc is available as an open-source project in the following GitHub repository:
    - https://github.com/xpn-arcos/xpn
  - Expand Ad-Hoc uses user-space MPI-based data servers on the same compute nodes on which applications are running
  - Expand Ad-Hoc provides a good performance compared to other solutions
- Future work:
  - Fault tolerant support

22nd IEEE International Symposium on Parallel and Distributed Computing

A new Ad-Hoc parallel file system for HPC environments based on the Expand parallel file system

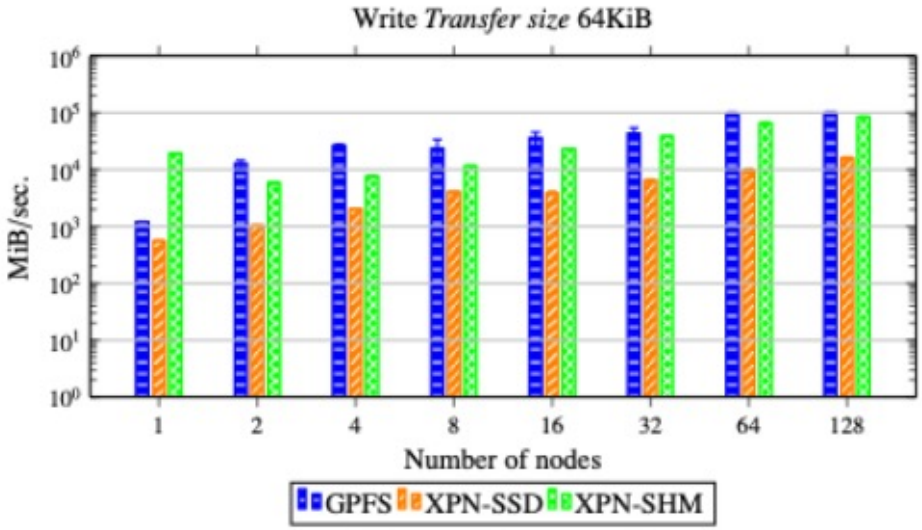
Félix García Carballeira
felix.garcia@uc3m.es

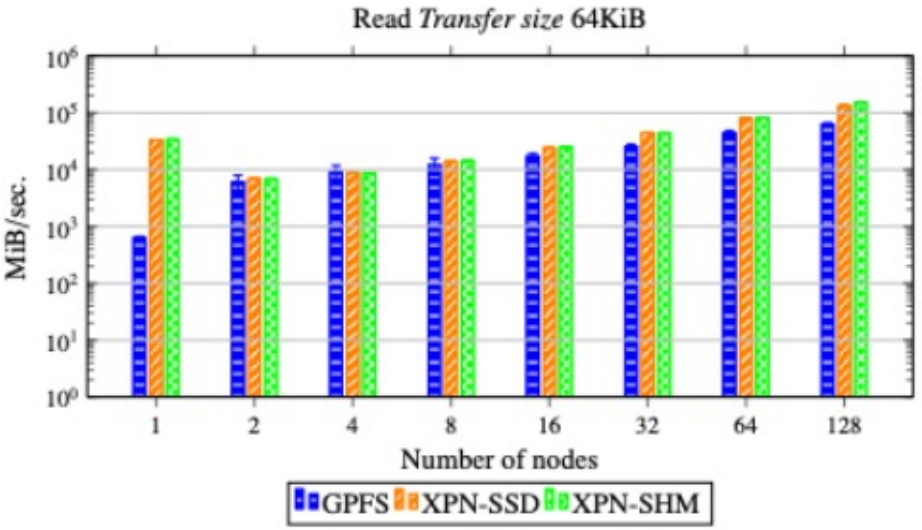
uc3m | Universidad Carlos III de Madrid

Evaluation

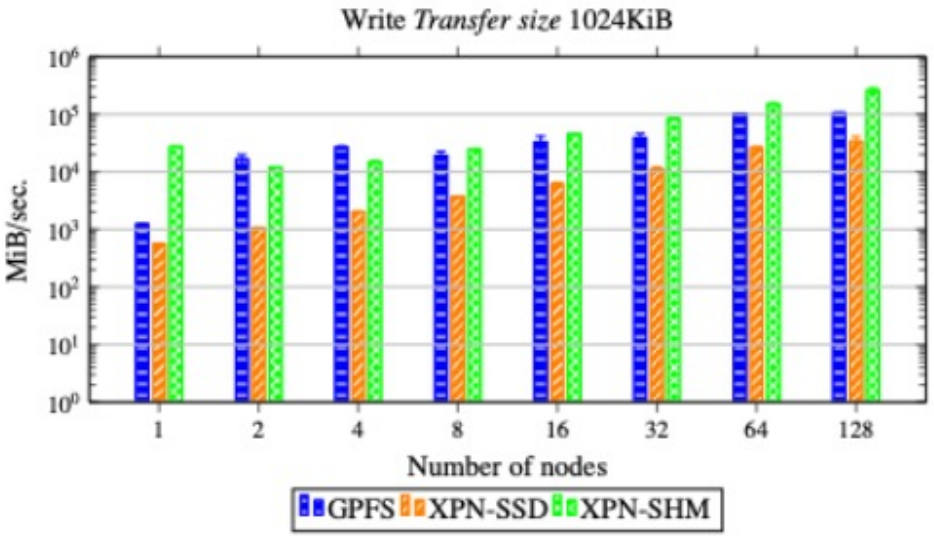
- Other results not included in the article:
  - A file per process. Each process accesses an individual file

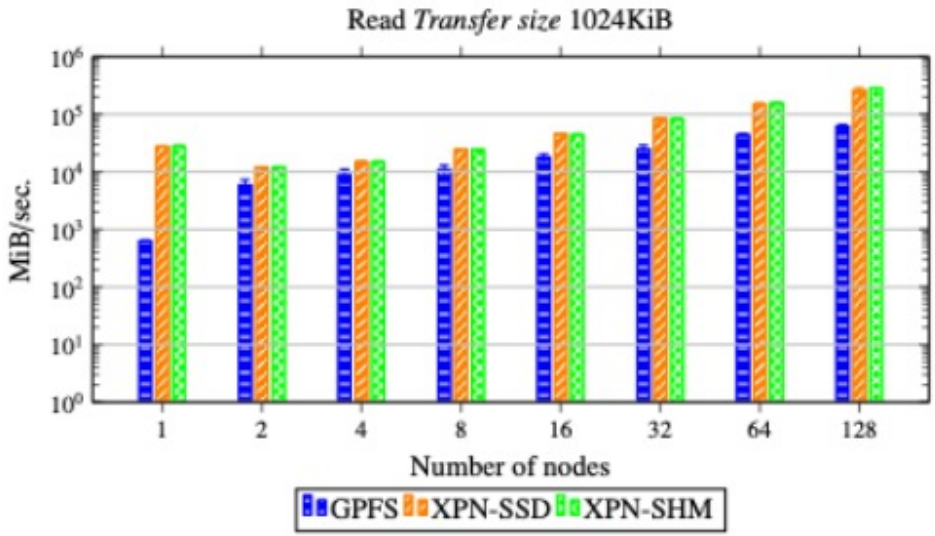
Results


Write *Transfer size* 64KiB
MiB/sec.
10^0
10^1
10^2
10^3
10^4
10^5
10^6
1
2
4
8
16
32
64
128
Number of nodes
GPFS
XPN-SSD
XPN-SHM


Read *Transfer size* 64KiB
MiB/sec.
10^0
10^1
10^2
10^3
10^4
10^5
10^6
1
2
4
8
16
32
64
128
Number of nodes
GPFS
XPN-SSD
XPN-SHM

Results


Write *Transfer size* 1024KiB
MiB/sec.
10^0
10^1
10^2
10^3
10^4
10^5
10^6
1
2
4
8
16
32
64
128
Number of nodes
GPFS
XPN-SSD
XPN-SHM


Read *Transfer size* 1024KiB
MiB/sec.
10^0
10^1
10^2
10^3
10^4
10^5
10^6
1
2
4
8
16
32
64
128
Number of nodes
GPFS
XPN-SSD
XPN-SHM