ADMIRE
malleable data solutions for HPC

Adaptive multi-tier intelligent data manager for Exascale

admire-eurohpc.eu

Current progress in ad-hoc storage systems within ADMIRE project

Javier Garcia-Blas
uc3m Universidad Carlos III de Madrid

Marc-André Vef
Johannes Gutenberg Universität Mainz JG|U

HPC-IODC: HPC I/O in the Data Center Workshop

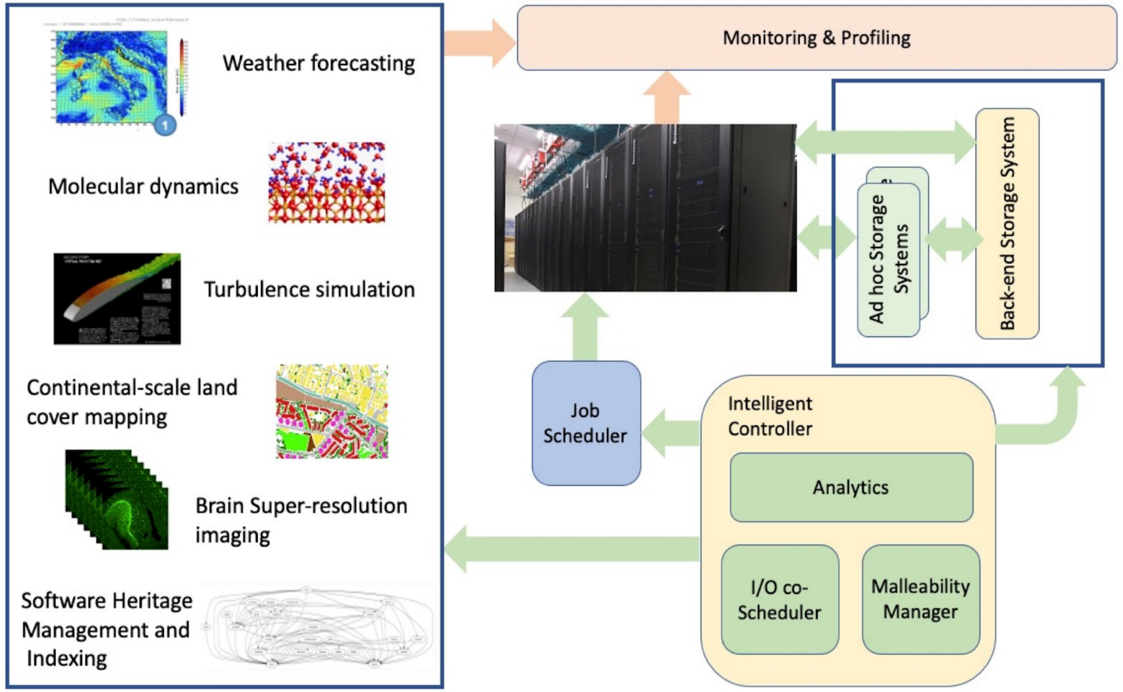
Grant Agreement number:
956748 — ADMIRE —
H2020-JTI-EuroHPC-2019-1

EuroHPC Joint Undertaking

Gobierno de España Ministerio de Ciencia e Innovación

The EuroHPC ADMIRE project

- € 7.9 million overall budget (2021-2024)
- 19 partners in 6 EU countries
- Key points:
  - Adaptive multi-tier data management
  - Computational and I/O malleability
  - Continuous monitoring
  - Focus on ad-hoc storage systems
  - Transparent data staging
  - Use cases from industry and academics



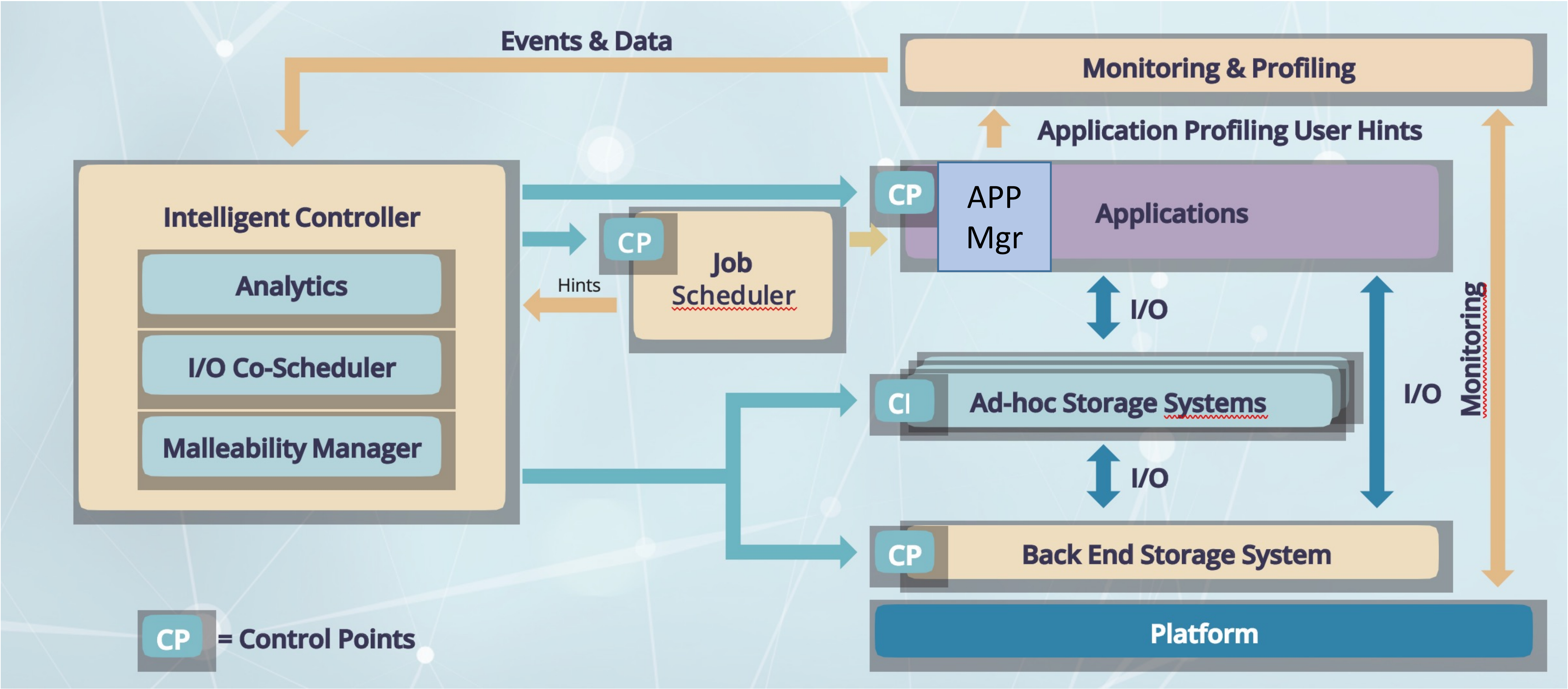
ADMIRE project architecture @ https://admire-eurohpc.eu

Motivation

- I/O-intensive HPC-based applications have been primarily based on distributed object-based file systems
  - **Separate data** from **metadata** management
  - Enable each client to **communicate in parallel** with multiple storage servers.
- Exascale I/O raises the throughput and storage capacity requirements by several orders of magnitude.
- Current challenges
  - Systems already developed for data analytics are not directly applicable to HPC due to the **fine-granularity** involved in scientific applications.
  - Semantic gap between the application requests and the way they are managed by the storage back-end at the block level.

The framework

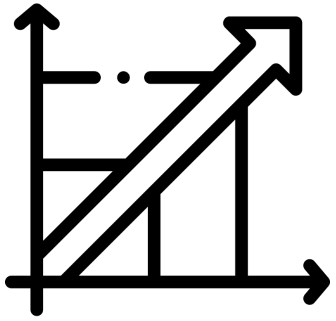
malleable data solutions for HPC



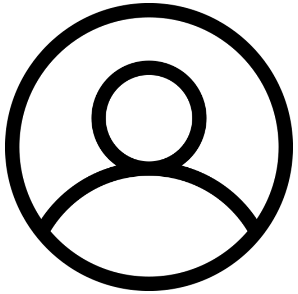
Adaptive I/O. Ad-hoc storage systems

- Deployed for an application or workflow
  - Implement temporary ad hoc storage systems with resilience features
  - Support fast node-local storage technologies (e.g., RAM, NVMe, SSD, ..)
- Four types of ad hoc storage systems
  - Distributed file system based on the **GekkoFS** burst buffer file system
  - Distributed object-oriented data store based on **dataClay**
  - **Hercules In-Memory Storage System** and
  - **Expand,** ad-hoc DFS based on MPI with external connectors**.**
- All ad-hoc storage systems support the same APIs
  - POSIX -> LD_PRELOAD
  - Custom I/O interfaces possible, but may require application changes
- General data staging approach in ADMIRE via I/O scheduler
  - Allows backend PFS independence

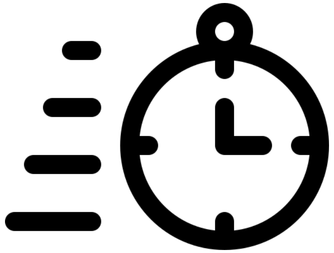
GekkoFS Architecture



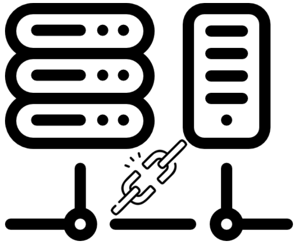
1. **Scalability**



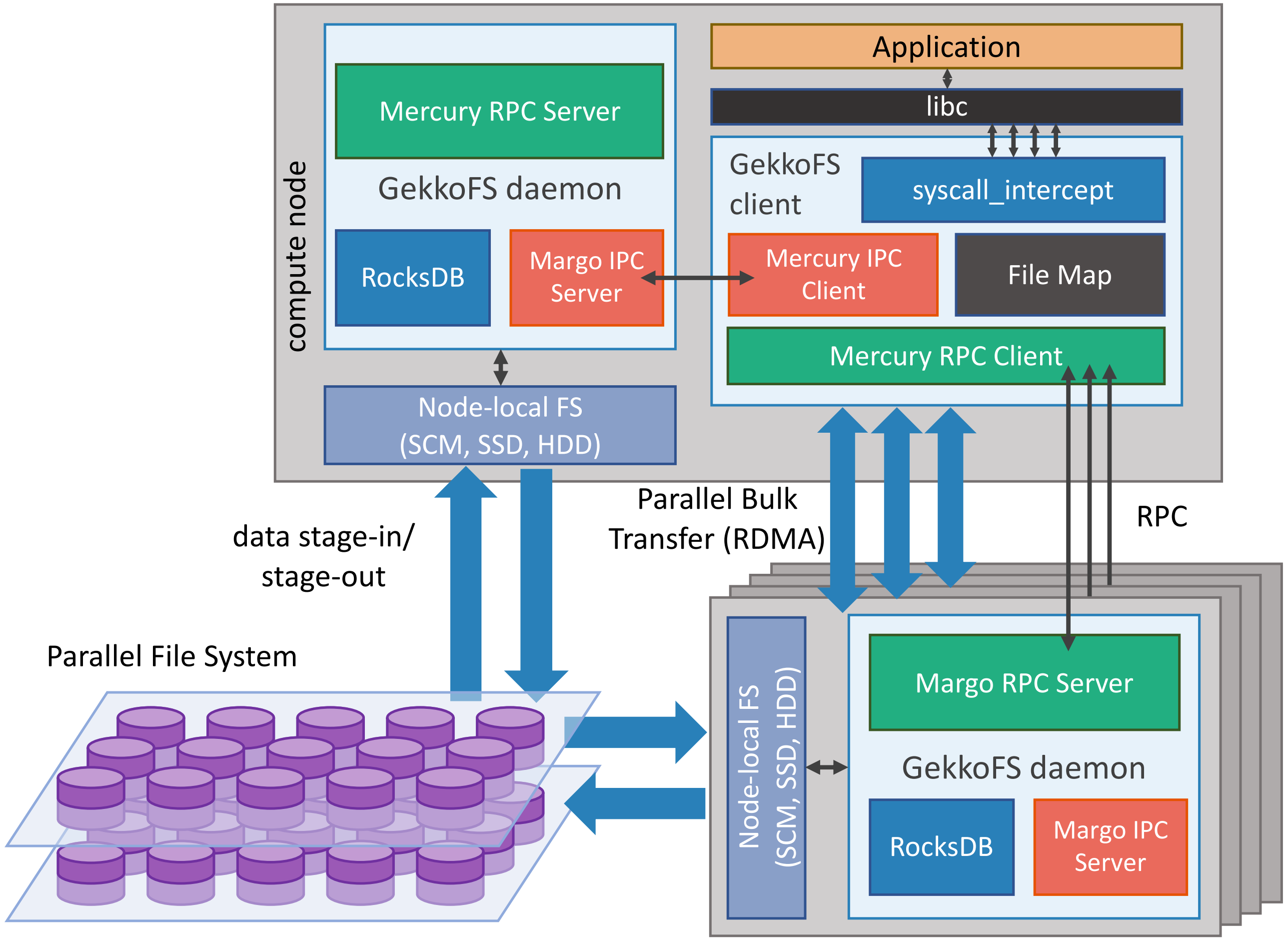
3. **User space**



2. **Fast deployment**



4. **Hardware independence**



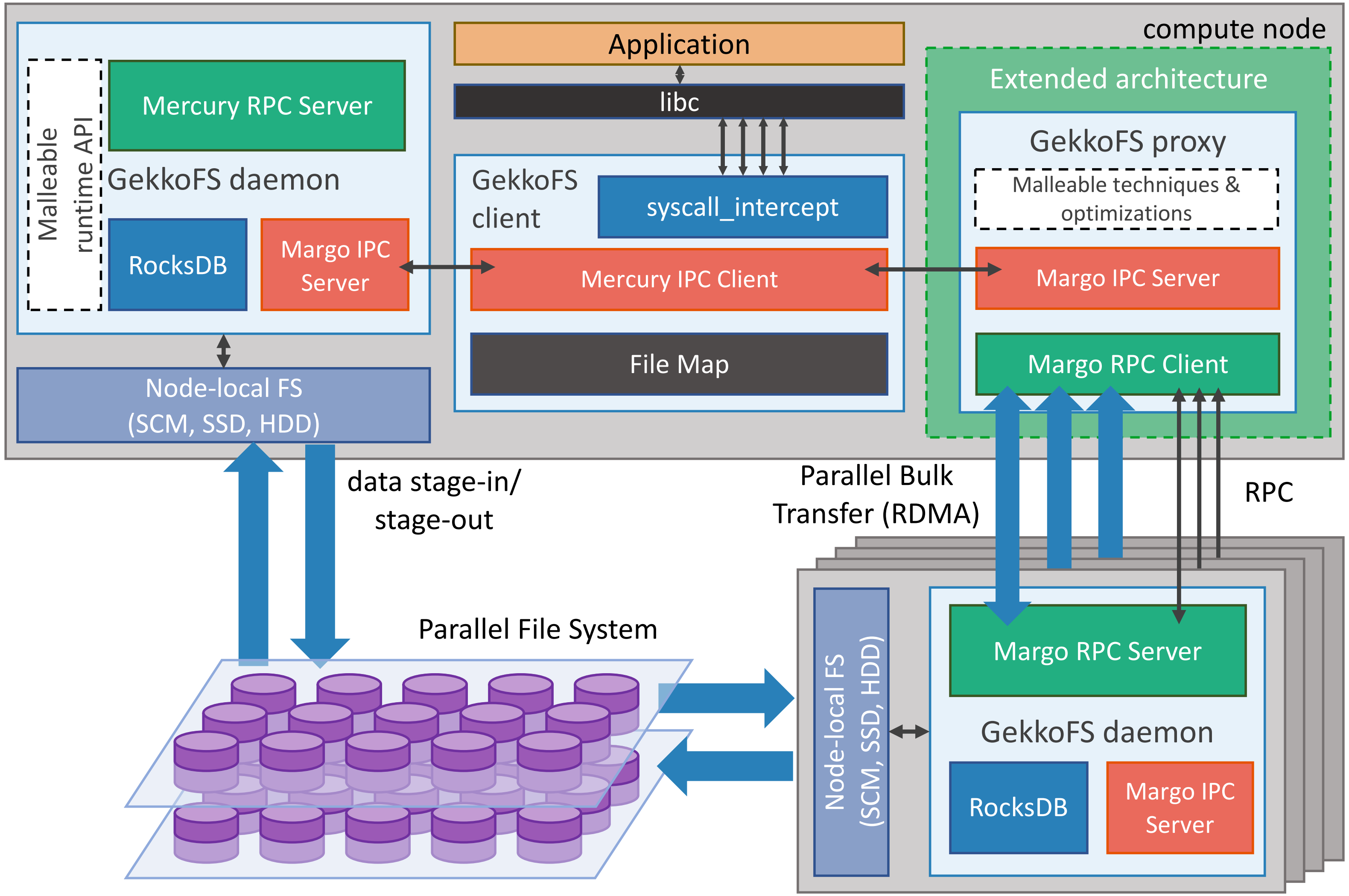
M.-A. Vef, N. Moti, T. Süß, M. Tacke, T. Tocci, R. Nou, A. Miranda, T. Cortes, A. Brinkmann. GekkoFS – A Temporary Burst Buffer File System for HPC Applications. In Journal of Computer Science and Technology (JCST), 2020

GekkoFS is open source:
https://storage.bsc.es/gitlab/hpc/gekkofs/

Challenges of LD_PRELOAD-based FS libraries

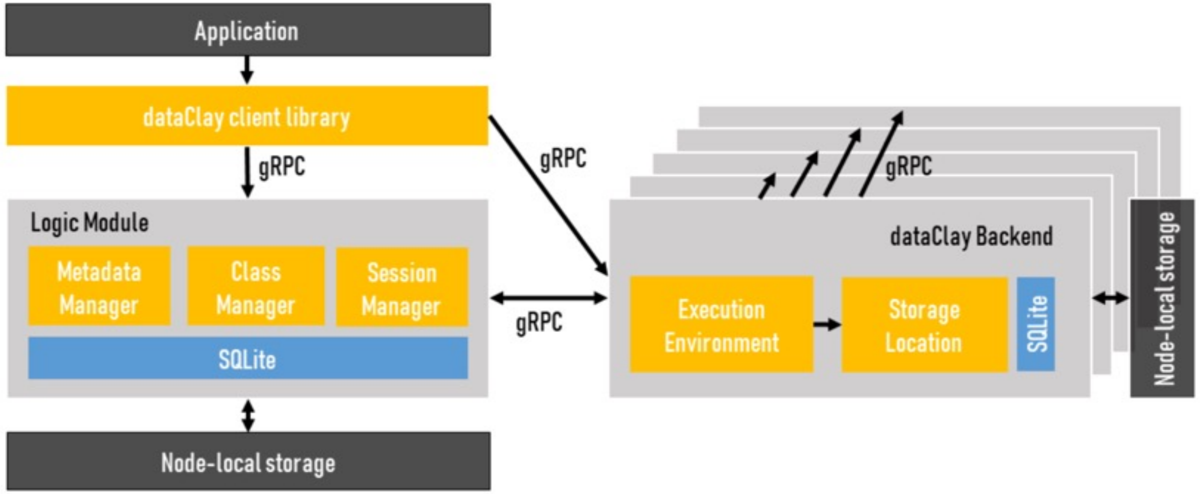
- Intercepts any function
  - Supporting all libc I/O functions is a daunting task
  - Other I/O interfaces, e.g., MPI I/O need special treatment
  - **Syscall_intercept helps to an extend**
- No valid mount point requires path checking
  - Detangle relative paths
  - **Separate file descriptor management necessary**
- Threads and forks don't mix
  - I/O library has no control over application or 3rd party library threads
  - `fork()` can lead to dead-locks
  - **Minimal interception library required**
- Malleability techniques limited
  - Any technique is confined to one process that can stop at any time
  - **Node-wide client solution necessary**

Redesign


compute node
Malleable runtime API
Mercury RPC Server
GekkoFS daemon
RocksDB
Margo IPC Server
Node-local FS (SCM, SSD, HDD)
Application
libc
GekkoFS client
syscall_intercept
Mercury IPC Client
File Map
Extended architecture
GekkoFS proxy
Malleable techniques & optimizations
Margo IPC Server
Margo RPC Client
data stage-in/ stage-out
Parallel Bulk Transfer (RDMA)
RPC
Parallel File System
Node-local FS (SCM, SSD, HDD)
Margo RPC Server
GekkoFS daemon
RocksDB
Margo IPC Server

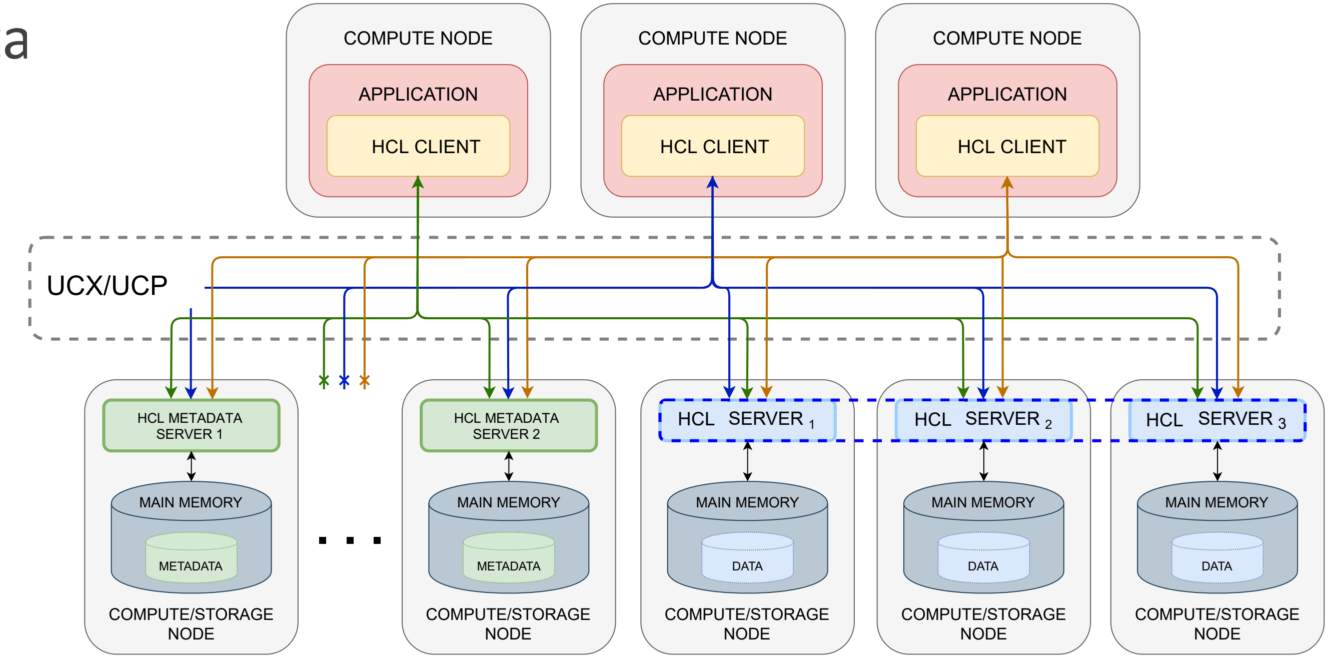
dataClay

- An active object store for HPC, Big Data, and Edge-to-Cloud applications
- Runs on heterogeneous infrastructures
- Brings computation to the data, enabling execution of custom user code
  - Objects = data + methods
  - Reduces communication costs and improves application performance
- Manages objects in-memory during execution
  - Avoids transformations and reduces disk accesses
- APIs
  - Object-oriented: transparent persistence (Python and Java bindings)
  - Object store: Get/Put operations



Hercules

- In-memory IO accelerator for volatile data
- Use cases:
  - HPC workflows.
  - Checkpointing engines.
  - Distributed locking.
- POSIX-based support for portability.
- Main features
  - Supports attached/detached storage servers.
  - Supports several data distribution policies.
  - Efficient storage backend based on GLIB.
  - Write on close policy.
  - Integrated with SLURM.
  - Data replication.



Javier Garcia-Blas, Genaro Sanchez-Gallegos, Cosmin Petre, Alberto Riccardo Martinelli, Marco Aldinuchi and Jesus Carretero. Hercules: scalable and network portable in-memory ad-hoc file system for data-centric and high-performance applications. EuroPAR conference. 2023.

https://gitlab.arcos.inf.uc3m.es/admire/imss

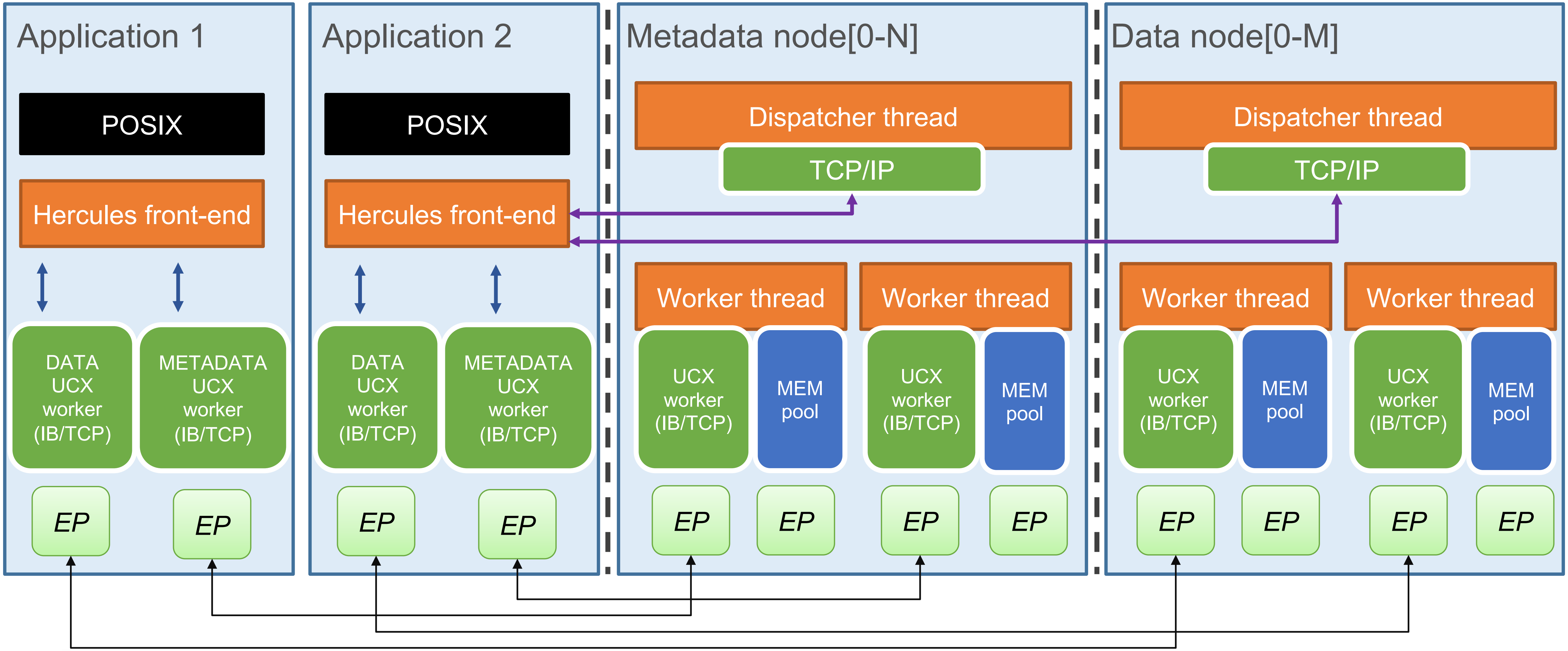
Portable communication layer

- Based on UCX
  - Unified Communication - X Framework (UCX) is a new acceleration library, integrated into the OpenMPI, OpenSHMEM, Dask, Charm++, …
- Key aspects
  - Multiple network interfaces/protocols available (TCP/IP, Omnipath, Infiniband supported).
  - Zero-copy message transfers of large data packages (>= 1 Mbytes).
  - Eliminated internal copies from application to network layer.
  - Asynchronous communication between peers.
  - RDMA QoS isolation.
  - End-point/two-sided-based communication.

UCX communication layer developed

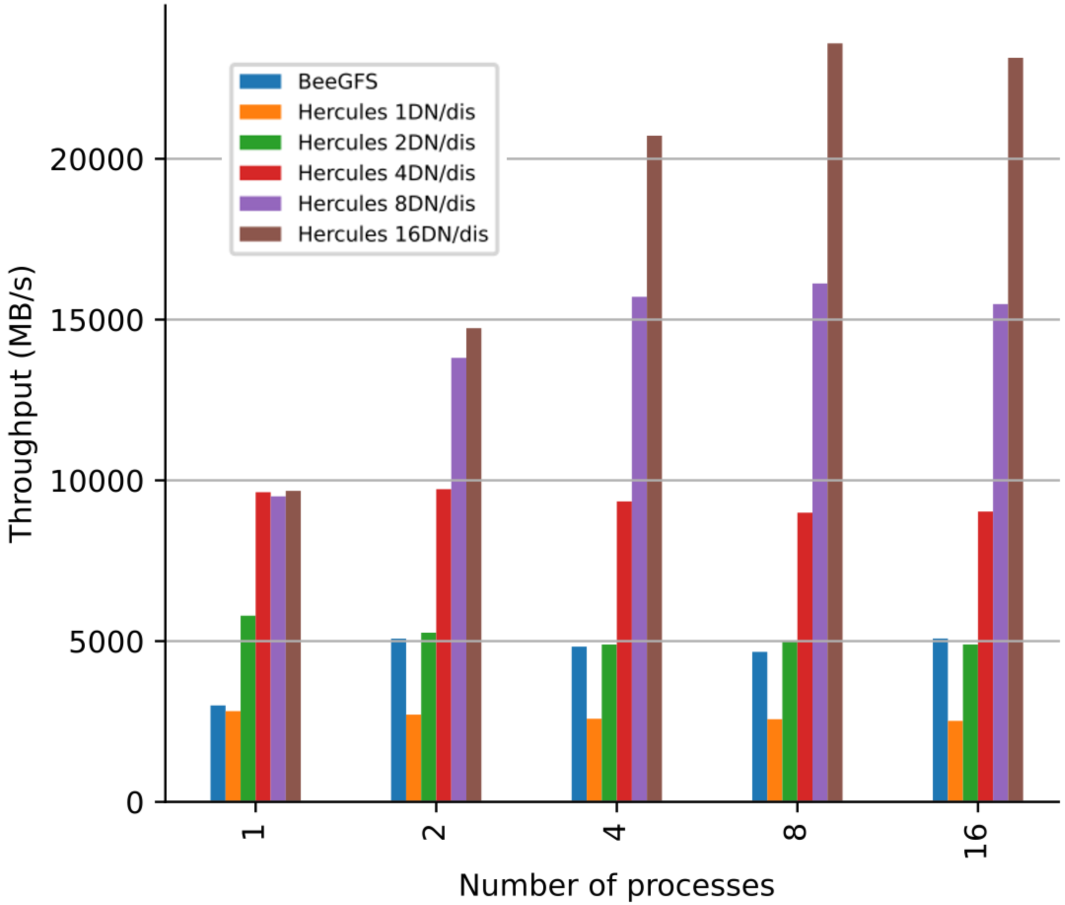
- Non-blocking/tag-based communication (MPI style)
  - **Asynchronous** communication between peers.
- Low-level communication schema (in contrast to Margo RPC)
- Frontend
  - Data and metadata UCX's workers enables **communication overlap**.
  - QoS
    - Interfaces and protocols can be enabled/disabled to adapt **network requirements**.
    - Communication can be upgraded/downgraded (Infiniband/Omnipath to TCP).
  - Communication parameters configured by using environment variables/config file.
- Backend
  - One single listener per worker thread.
  - Stores a pool of active end-points (two-sided communication).

Hercules

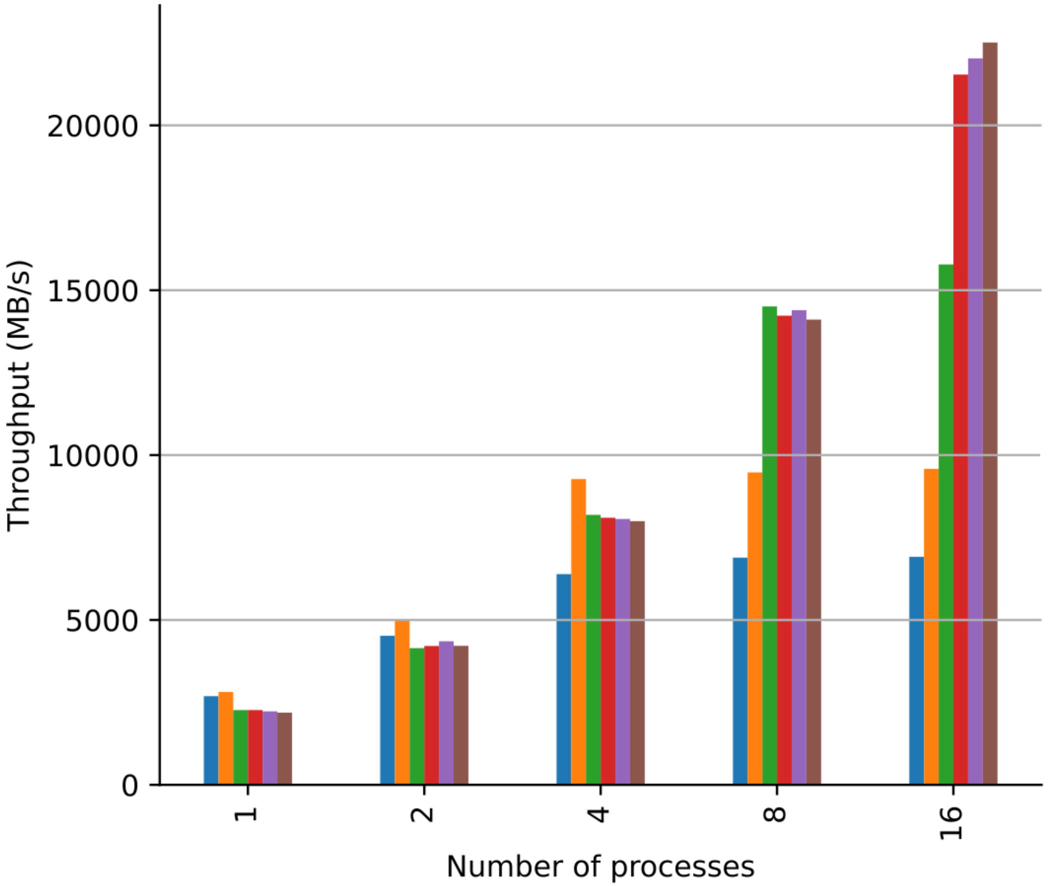


Application 1
POSIX
Hercules front-end
DATA UCX worker (IB/TCP)
METADATA UCX worker (IB/TCP)
EP
EP
Application 2
POSIX
Hercules front-end
DATA UCX worker (IB/TCP)
METADATA UCX worker (IB/TCP)
EP
EP
Metadata node[0-N]
Dispatcher thread
TCP/IP
Worker thread
Worker thread
UCX worker (IB/TCP)
MEM pool
UCX worker (IB/TCP)
MEM pool
EP
EP
EP
EP
Data node[0-M]
Dispatcher thread
TCP/IP
Worker thread
Worker thread
UCX worker (IB/TCP)
MEM pool
UCX worker (IB/TCP)
MEM pool
EP
EP
EP
EP

Evaluation



Write Throughput



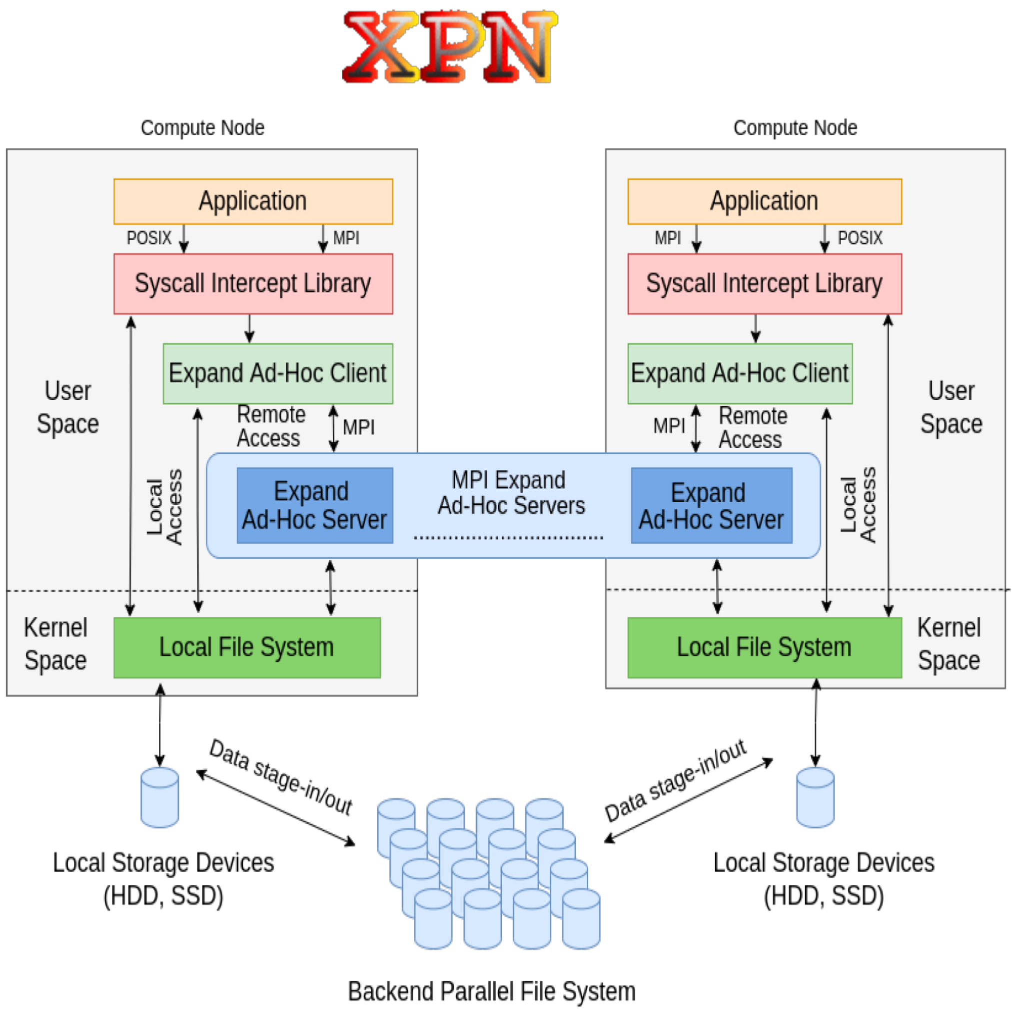
Read Throughput

Expand Parallel File System

- **Expand** ad-hoc file system
  - Parallel, distributed file system based on the use of existing standard storage servers.
  - MPI-based
  - Distribution based on subfile/server and Round Robin
  - Fully distributed metadata (B0 of each file) by hash
  - Aggressive parallel I/O ops
  - Client-servers may be co-located

F. García, A. Calderón, J. Carretero, J. Fernández, J. M. Pérez. The Design of the Expand Parallel File System. International Journal of High Performance Computing Applications. Vol. 17. Nº 1, Spring 2003. Pags. 21-37

https://arcos-xpn.github.io



Performance of Ad-hoc Expand

❑ IOR in MareNostrum

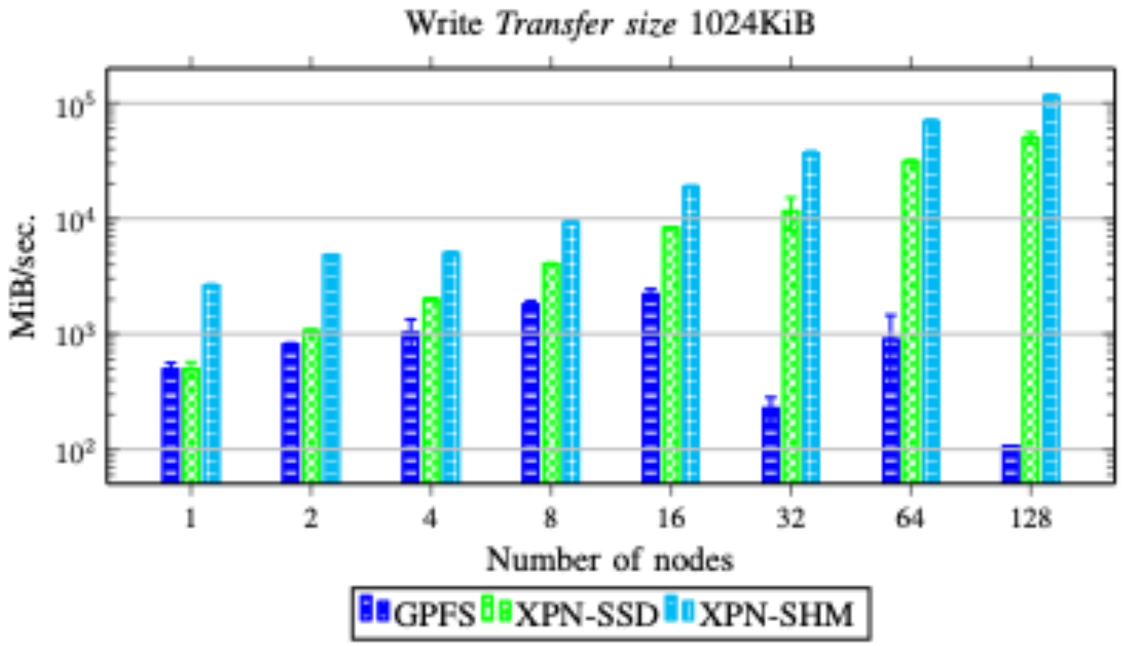


Fig. 4. GPFS vs. Expand Ad-Hoc. Bandwidth (MiB/sec.) writing with different transfer sizes (64KiB, 512KiB, and 1MiB), compute nodes (1, 2, 4, 8, 16, 32, 64, and 128), with 8 client process per node and shared file. Results in logarithmic scale

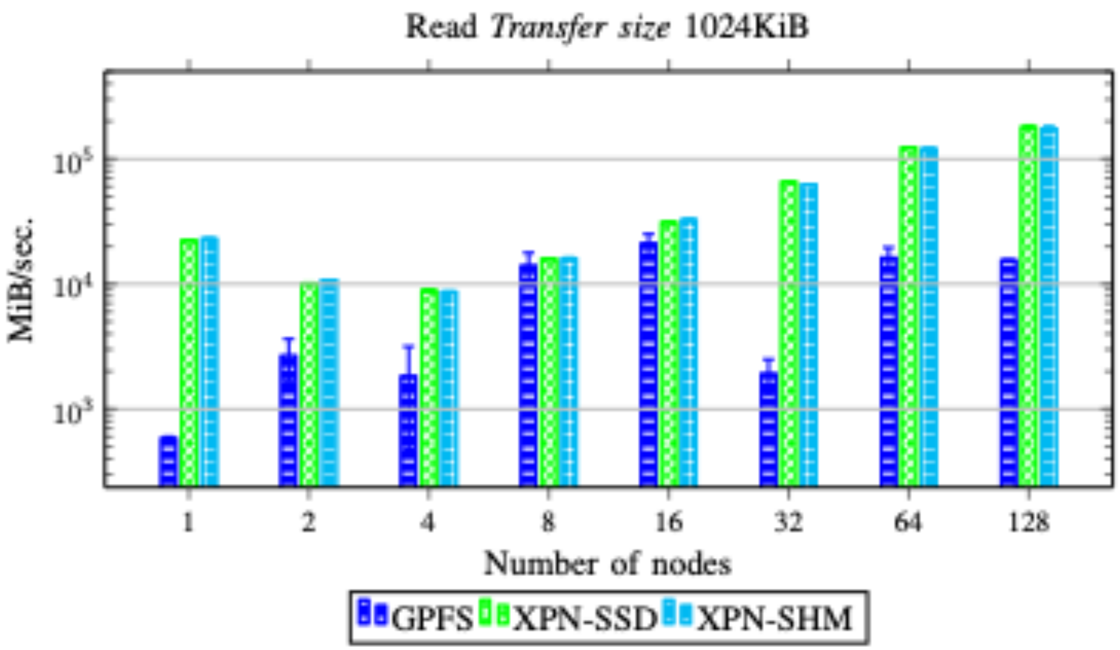


Fig. 5. GPFS vs. Expand Ad-Hoc. Bandwidth (MiB/sec.) reading data with different transfer sizes (64KiB, 512KiB, and 1MiB), compute nodes (1, 2, 4, 8, 16, 32, 64, and 128), with 8 client process per node and shared file. Results in logarithmic scale

ADMIRE malleable data solutions for HPC

Current efforts

- ❑ Malleability
  - ➢Expanding/shrinking the number of data nodes.
- ❑ Monitoring
  - ➢Integration with existing monitoring tools.
- ❑ Replication / Mirroring
- ❑ Error correction schemes
  - ➢ECC/parity.
  - ➢To implement distributed parity calculations on advanced data placement schemes.

ADMIRE
malleable data solutions for HPC

Adaptive multi-tier intelligent data manager for Exascale

admire-eurohpc.eu

Current progress in ad-hoc storage systems within ADMIRE project

Javier Garcia-Blas

uc3m Universidad Carlos III de Madrid

Marc-André Vef

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

HPC-IODC: HPC I/O in the Data Center Workshop

Grant Agreement number:
956748 — ADMIRE —
H2020-JTI-EuroHPC-2019-1

EuroHPC Joint Undertaking

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN