JIT staging for ad-hoc file systems using I/O frequency predictions and Lustre's HSM

Marc-André Vef[1], Ahmad Tarraf[2], Maysam Rahmanpour[3], Ramon Nou[4], Reza Salkhordeh[3], André Brinkmann[3]

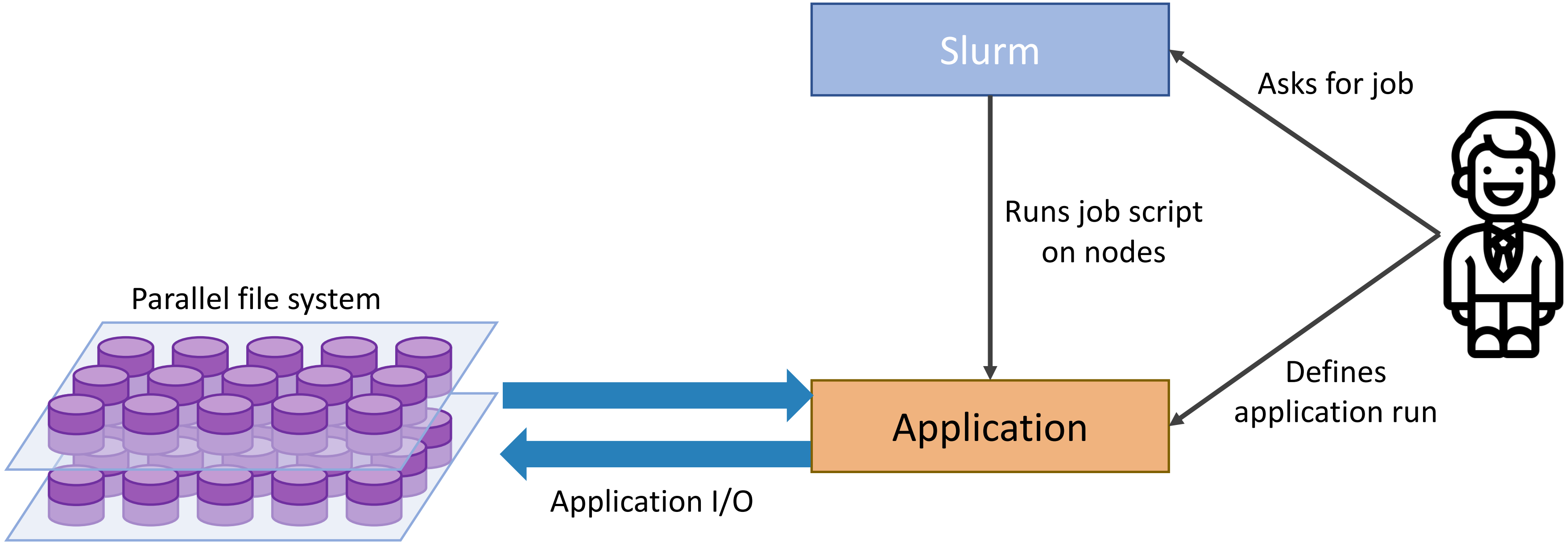
[1]DDN/Whamcloud
[2]Technical University Darmstadt
[3]Johannes Gutenberg University Mainz
[4]Barcelona Supercomputing Center

23. September 2024

LAD24 LUSTRE ADMIN & DEV WORKSHOP
Paris 23.24.25/9/2024

Traditional usage of parallel file systems (PFS) in HPC

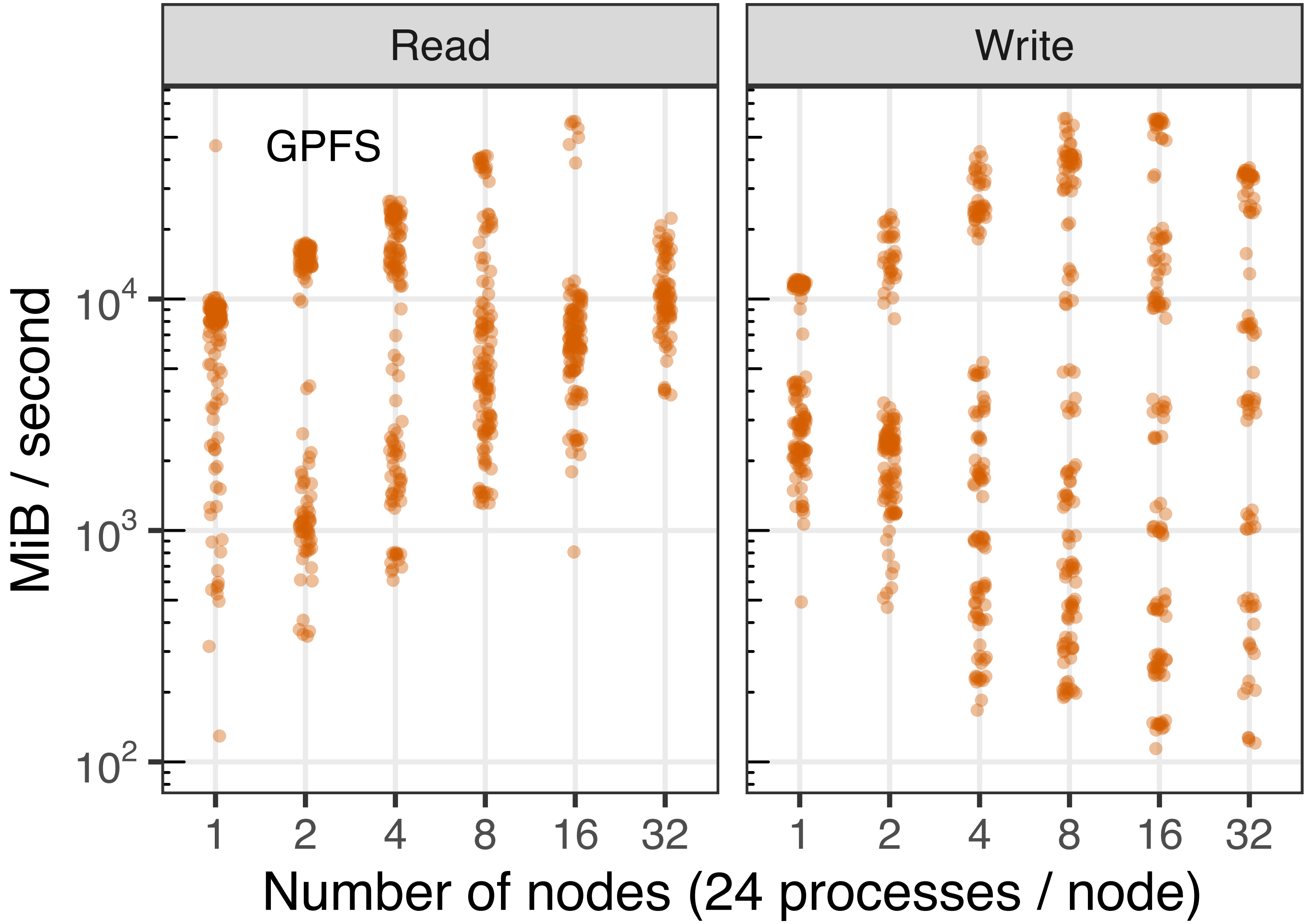
- Starting point …



PFSs are a shared I/O resource

I/O performance varies wildly for identical workloads

Applications suffer due to PFS load

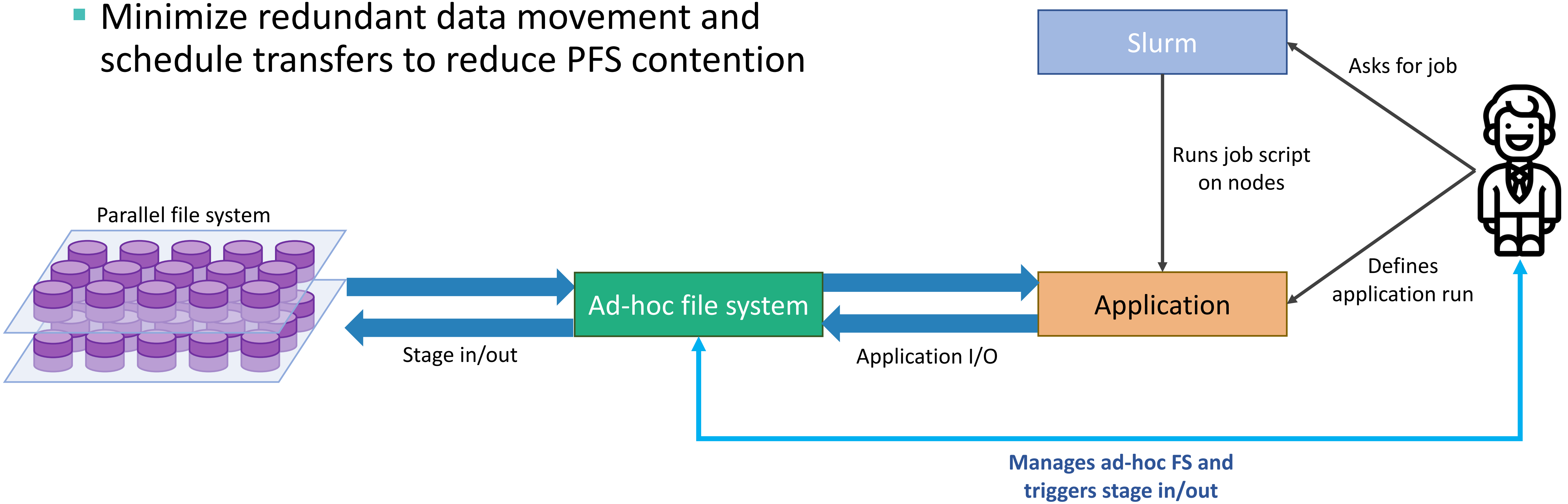


GPFS on MareNostrum 4 at the Barcelona Supercomputing Center

HPC applications using ad-hoc file systems

Ad-hoc file systems as a burst buffer for HPC applications:

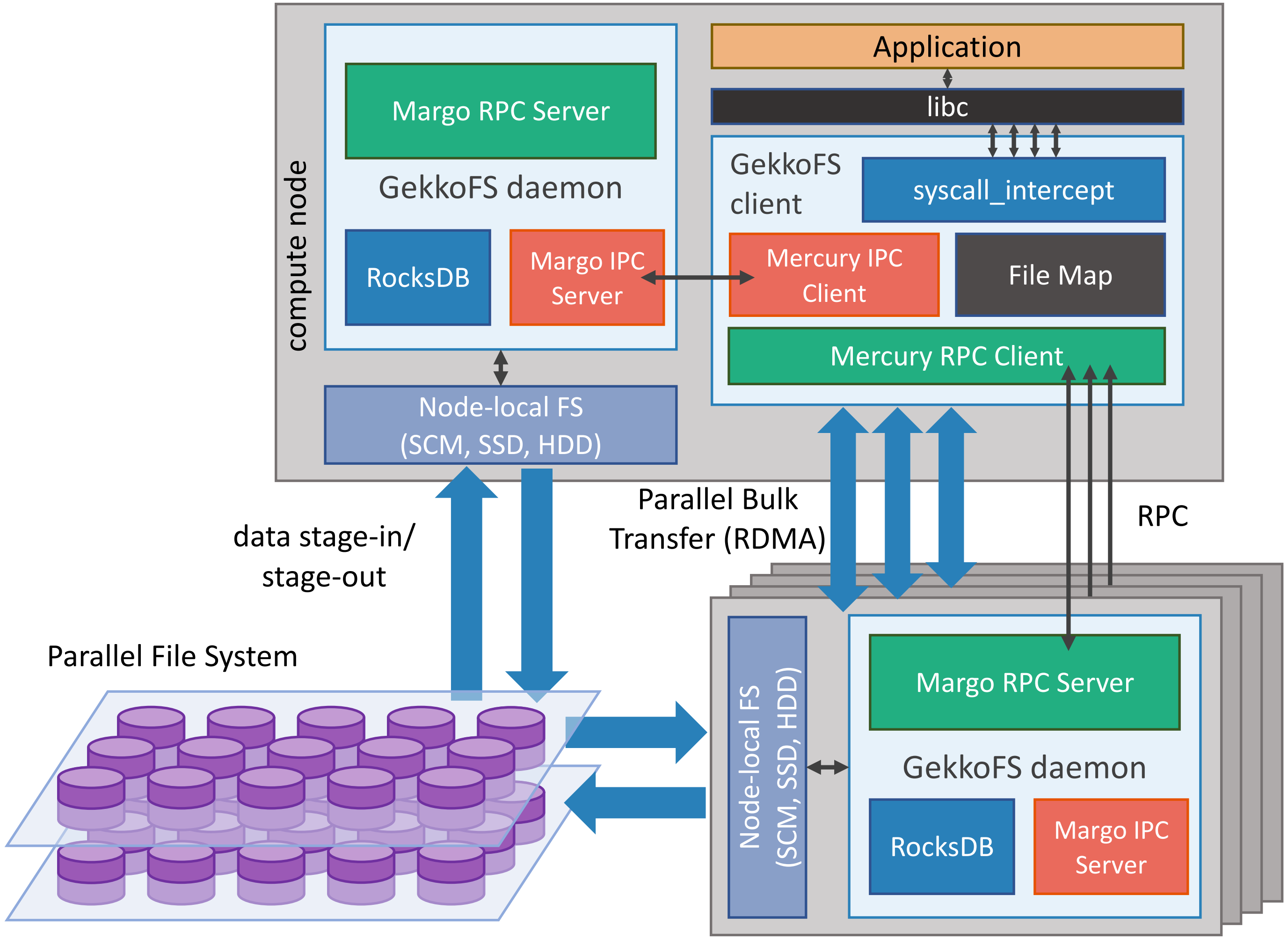
- Improve data locality: Do work where data lives and combine node-local SSDs
- Minimize uncoordinated PFS usage
- Minimize redundant data movement and schedule transfers to reduce PFS contention



GekkoFS

Simplified architecture

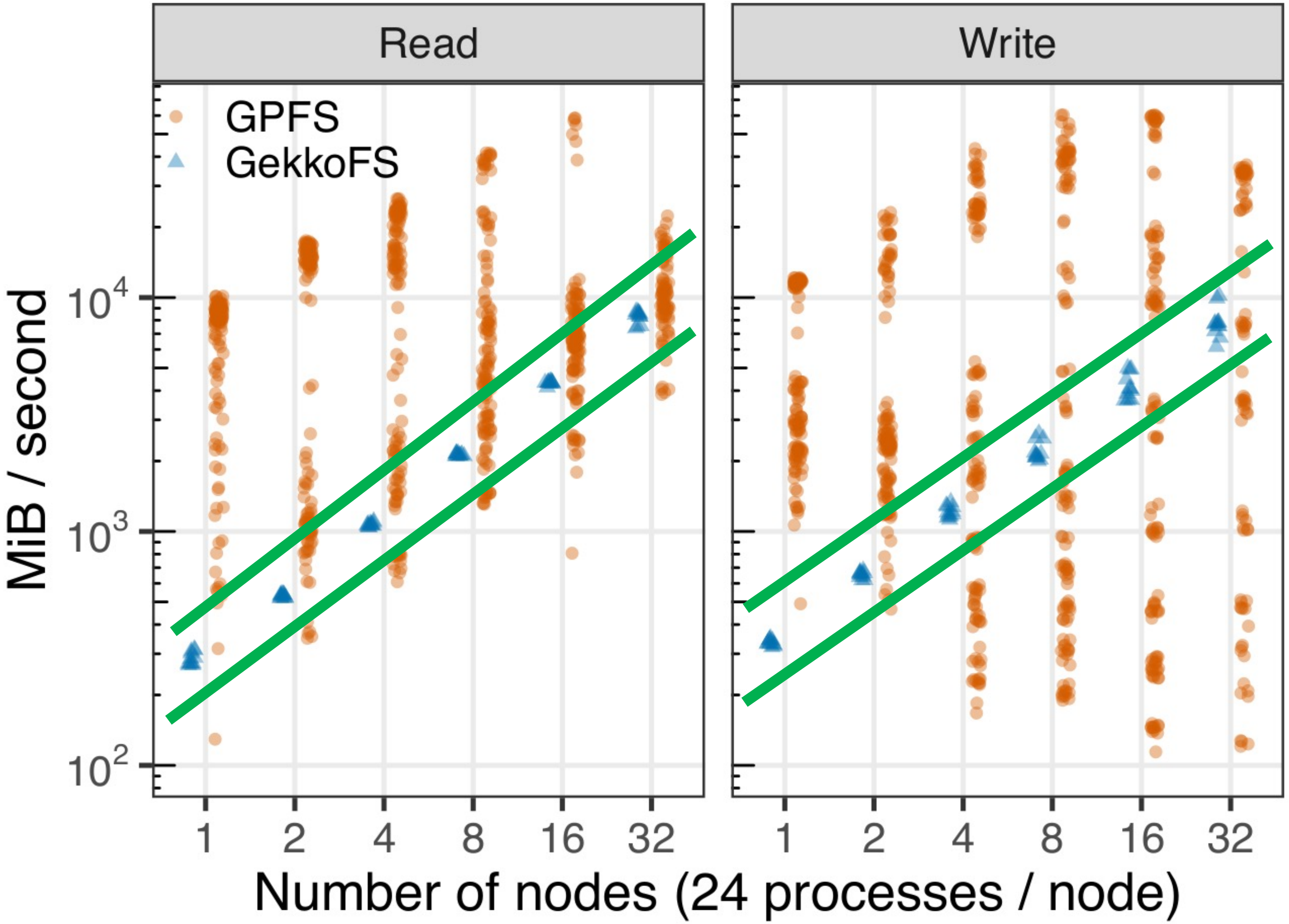
1. **Scalability**
2. **User space**
3. **Fast deployment**
4. **Hardware independence**



M.-A. Vef, N. Moti, T. Süß, T. Tocci, R. Nou, A. Miranda, T. Cortes, A. Brinkmann.
GekkoFS – A Temporary Distributed File System for HPC Applications.
In International Conference on Cluster Computing (CLUSTER), 2018

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

Performance variability revisited



Reduced I/O variability for GekkoFS

GPFS on MareNostrum 4 at the Barcelona Supercomputing Center

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 with I/O kernels & applications

S3D via PnetCDF (MN4)

- 20 nodes - 729 MPI processes; WRITE-only workload => 3476.14 GiB
- Bandwidth: 795.67 MiB/s vs 8651.79 MiB/s (+10x)

HACCIO (MN4)

- 20 nodes - Checkpoint – restart workload (2 TiB workload)
- WRITE Bandwidth: 932.691 MB/s vs 946.617 MB/s
- READ Bandwidth: 2458.82 MB/s vs 4208.82 MB/s (+2x)

NASBT-IO (MN4)

- 20 nodes - 729 MPI processes; WRITE-only workload => 2048.00 GiB
- WRITE Bandwidth: 746.75 MiB/s vs 13527.98 MiB/s (+18x)

NEK5000 computational fluid dynamics (MOGON II)

- 32 nodes - 512 MPI processes; WRITE-only workload => 59308 GiB
- WRITE Bandwidth: 347.33 MiB/s vs 1322.67 MiB/s (+3.8x)

WacomM++ Water Community Model (Mogon-NHR)

- 16 nodes – 256 MPI processes – byte-sized I/O operations on shared file
- Runtime: 267 seconds vs 86 seconds (+3.1x)

Ad-hoc file systems in real life — Challenges and possible solutions

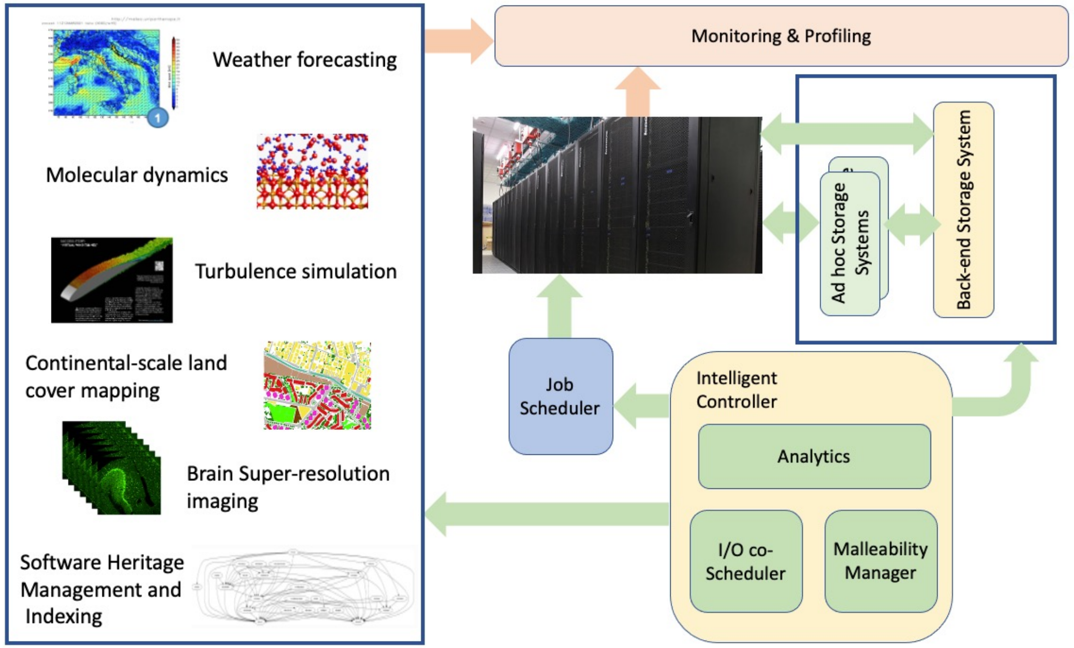
- **No** transparency and requires user interaction
  - Starting and stopping ad hoc file system
  - Data staging (data movement between namespaces)
  - Data is stored at two locations (threat of overwriting)
- The EuroHPC **ADMIRE** project
  - Adaptive multi-tier data management
  - Computational and I/O malleability
  - Focus on ad hoc storage systems

In this talk:

1. **GekkoFS-Lustre integration** (JGU, DDN collab)
2. **I/O trace analysis** (JGU, DDN, ANL collab)
3. **Just-in-time staging** (JGU, BSC, TUD collab)

EuroHPC Joint Undertaking

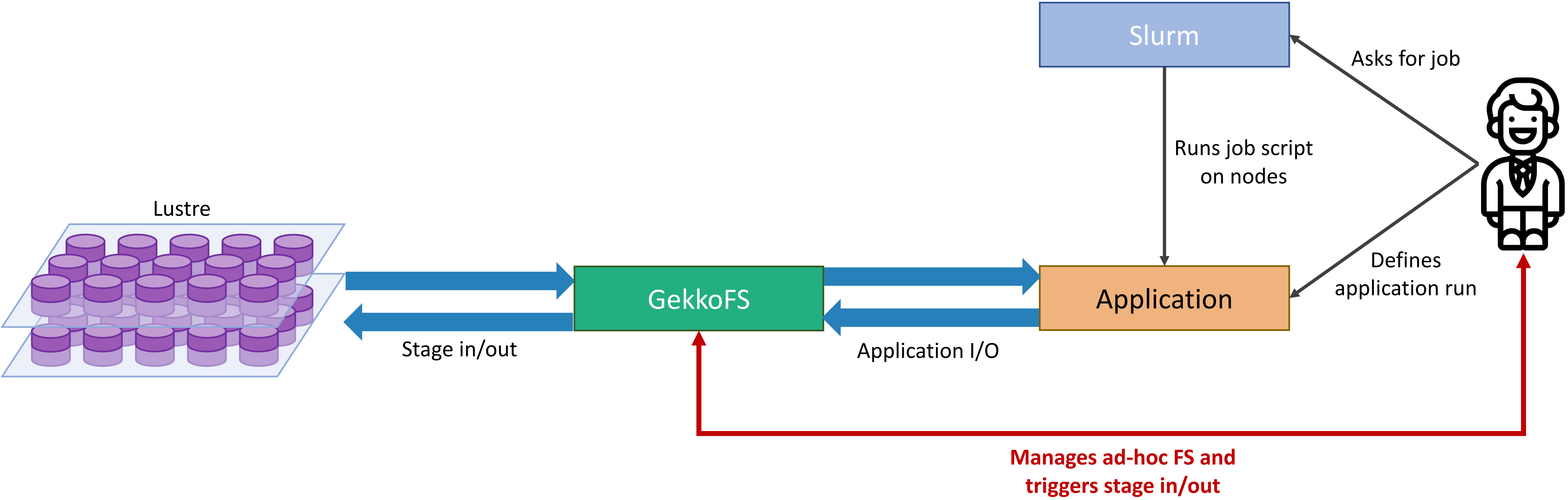
ADMIRE malleable data solutions for HPC



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

HPC applications using GekkoFS

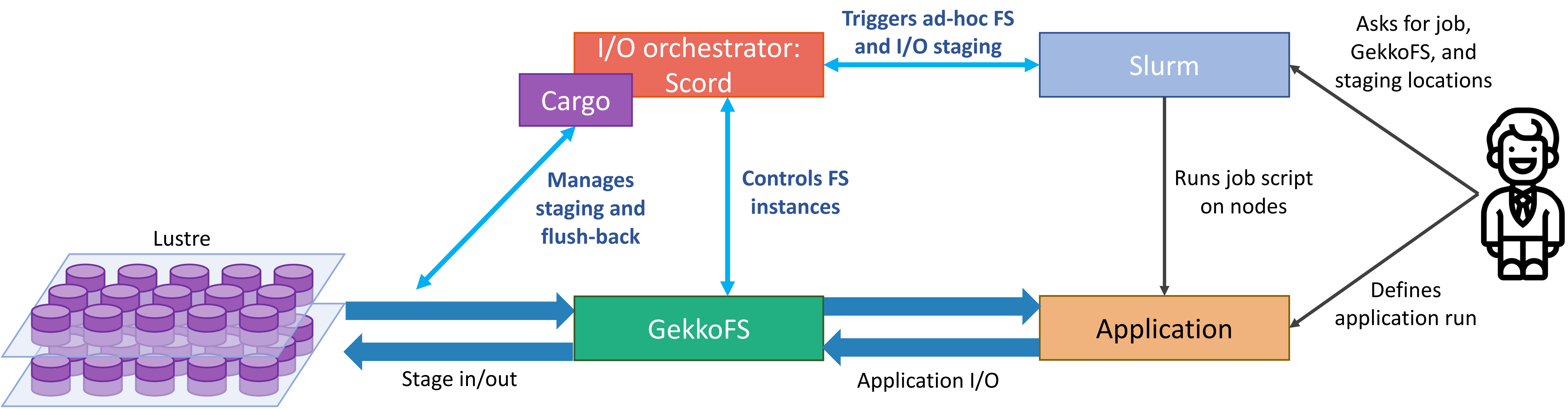
Users must start GekkoFS and move data themselves



GekkoFS with I/O orchestrator

See ADMIRE project

- I/O orchestrator (**Scord**) manages GekkoFS instances
- Scord uses **Cargo** for (parallel) data staging

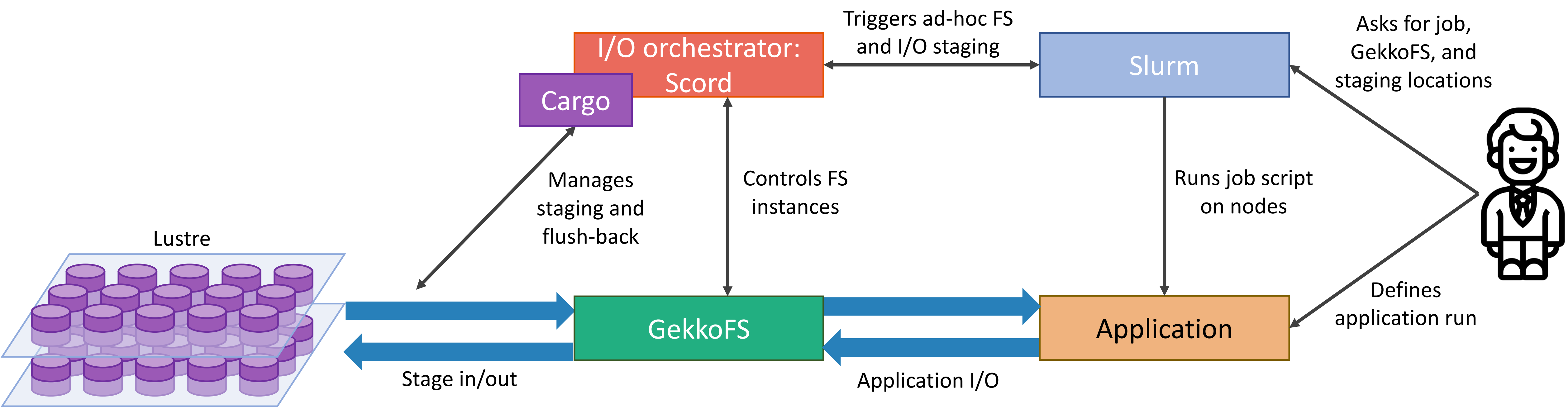


Scord: https://storage.bsc.es/gitlab/eu/admire/io-scheduler/
Cargo: https://storage.bsc.es/gitlab/hpc/cargo

GekkoFS with I/O orchestrator

See ADMIRE project

- I/O orchestrator (**Scord**) manages GekkoFS instances
- Scord uses **Cargo** for (parallel) data staging



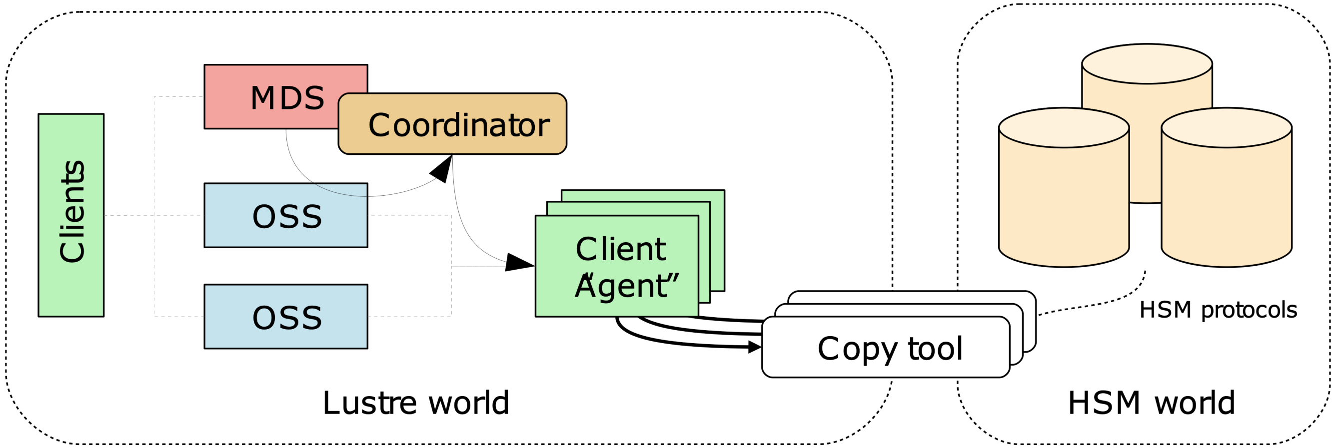
New SLURM args:

```
--ADM_adhoc_context= in_job:[shared|dedicated] | separate:[new|existing]
--ADM_adhoc_nodes= number_of_nodes
--ADM_input= “ORIGIN => TARGET”
--ADM_output= “ORIGIN => TARGET”
```

Scord: https://storage.bsc.es/gitlab/eu/admire/io-scheduler/
Cargo: https://storage.bsc.es/gitlab/hpc/cargo

Hierarchical Storage Management (HSM) in Lustre

- Lustre provides a framework to incorporate HSM-tiered storage (typically archiving)
- File data can exist in the HSM solution with its metadata residing in Lustre
- I/O operations on file triggers flush-back to Lustre (user transparency)
- Copy tool coordinates archiving and restore operations
- MDS Coordinator processes HSM requests



Overview of the Lustre file system HSM

Y. Qian, X. Li, S. Ihara, A. Dilger, C. Thomaz, S. Wang, W. Cheng, C. Li, L. Zeng, F. Wang, D. Feng, T. Süß, and A. Brinkmann.
LPCC: Hierarchical Persistent Client Caching for Lustre, SC'19.

Lustre Persistent Client Caching (LPCC)

- LPCC integrates into established HSM mechanisms, maintaining a unified namespace
- Layout lock mechanism to provide consistent cache services

Two caching modes:

- RW-PCC: read-write cache on **single** client
  - **No conflicting access allowed**
  - ➢ No parallel I/O from many nodes possible
- RO-PCC: read-only cache on **multiple** clients
  - **Concurrent access allowed but redundant data**
  - ➢ Can cause I/O overhead on parallel file system when many nodes cache the same data

Lustre Persistent Client Caching (LPCC)

- LPCC integrates into established HSM mechanisms, maintaining a unified namespace
- Layout lock mechanism to provide consistent cache services

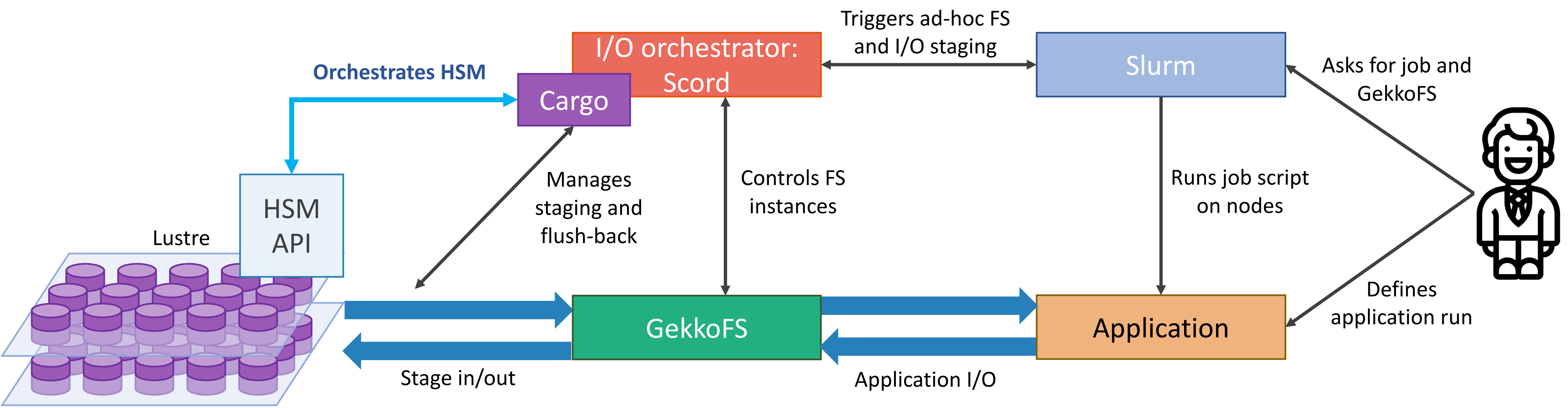
Two caching modes:

- RW-PCC: read-write cache on **single** client
  - **No conflicting access allowed**
  - ➢ No parallel I/O from many nodes possible
- RO-PCC: read-only cache on **multiple** clients
  - **Concurrent access allowed but redundant data**
  - ➢ Can cause I/O overhead on parallel file system when many nodes cache the same data

Combine Lustre HSM and ad-hoc file systems

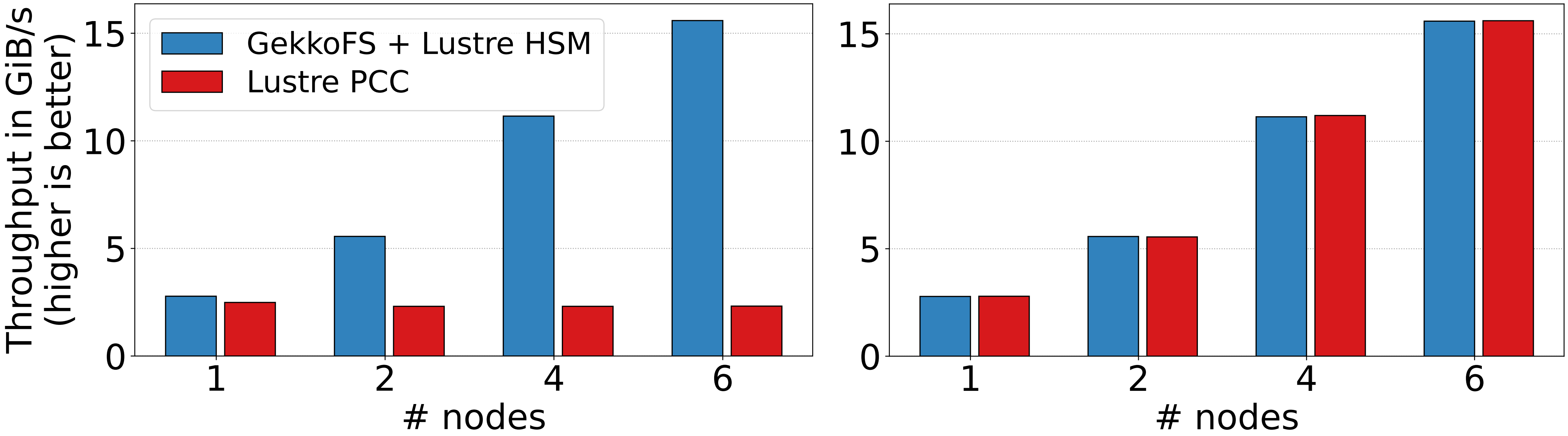
GekkoFS & Lustre HSM integration

- GekkoFS transparency via I/O orchestration and hierarchical storage management (HSM)



GekkoFS HSM & LPCC

- Experiments on separate partition and small test-Lustre @Mogon-NHR
- Preliminary write results via IOR (simulating application output)
- I/O req. size: 1 MiB ; 4 GiB per process ; 32 processes per node
- Concurrent RW mode allowed with GekkoFS



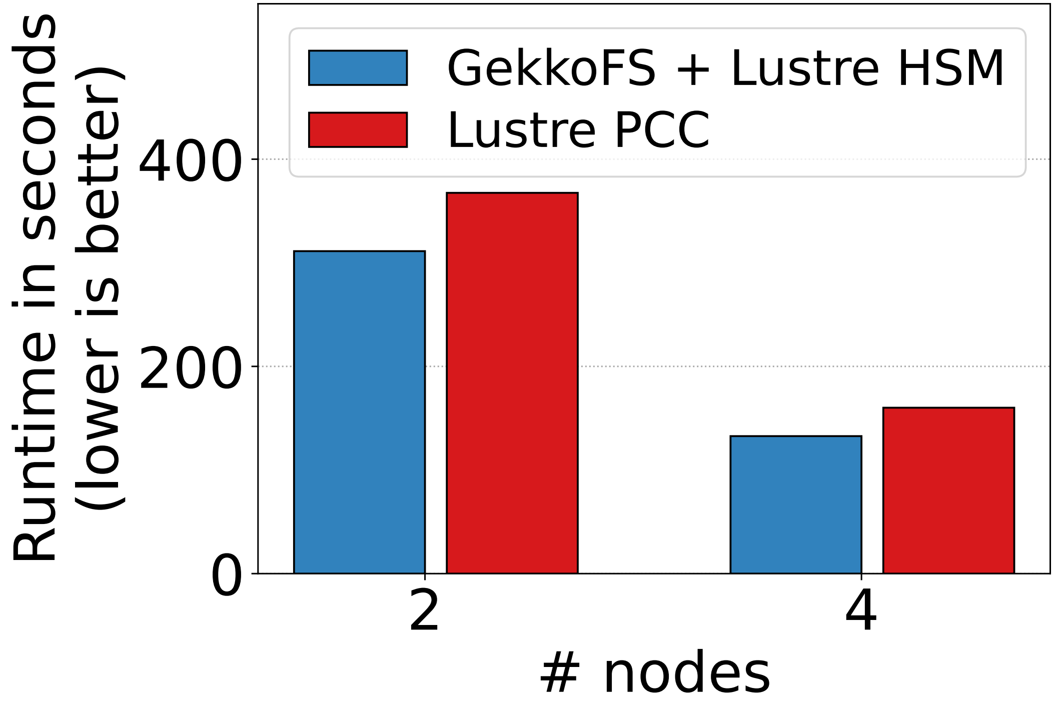
Single shared file **File per process**

GekkoFS HSM & Nek5000

- Preliminary results on a small-scale Nek5000 CFD simulation and small test-Lustre

Workload:

- 50 total compute steps ; 16 processes / node
- Processes write in a single shared file (~700 MiB)
- Output file generated every 5 steps
- I/O req. sizes decrease with more processes
- Additional statistics are written periodically



The I/O Trace Initiative

- The I/O Trace Initiative is an on-going community effort at fostering collaboration and improving knowledge of the I/O aspects in HPC applications
- A **JGU**-led cross-project initiative between ADMIRE, IO-SEA, and DeepSEA
- Additional collaborations with **DDN**, **CEA**, UC3M, ANL, and LBNL
- Submission, archiving, searching, analysis, and visualization tools
- 40+ traces including Molecular Dynamics, DL, ML, CFD and up to 130,000 rank runs
- Built on Darshan profiles
- Interactive heatmaps and Drishti integration for I/O recommendations
- **Full DOI support:** All traces include a Zenodo link
- Available at https://hpcioanalysis.zdv.uni-mainz.de

N. Moti, A. Brinkmann, M.-A. Vef, P. Deniel, J. Carretero, P. Carns, J.-T. Acquaviva, R. Salkhordeh.
The I/O Trace Initiative: Building a Collaborative I/O Archive to Advance HPC.
In International Parallel Data Systems Workshop (PDSW) @SC23 (2023)

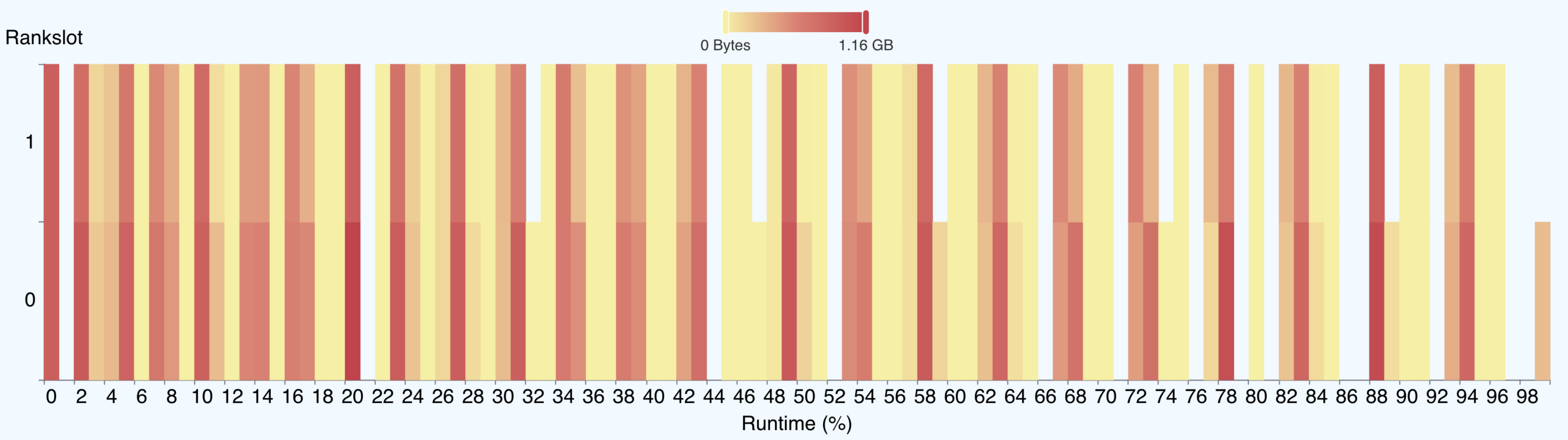
The I/O Trace Initiative

Heatmaps: Two examples

Wacomm++

Water Community Model

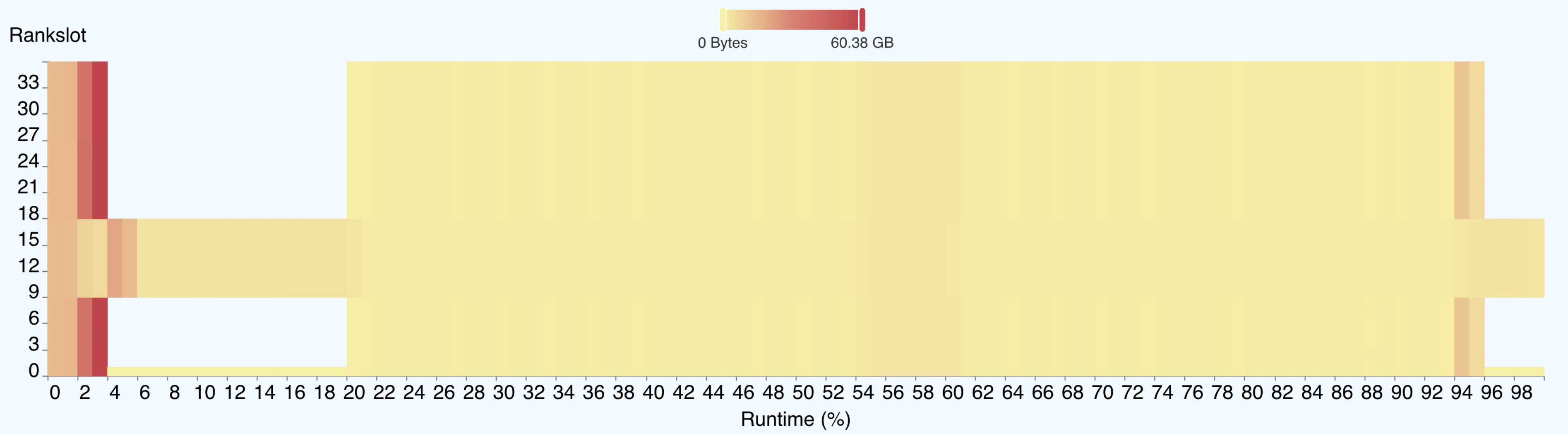
@Uni Napoli



Remote Sensing

Horovod

@Jülich



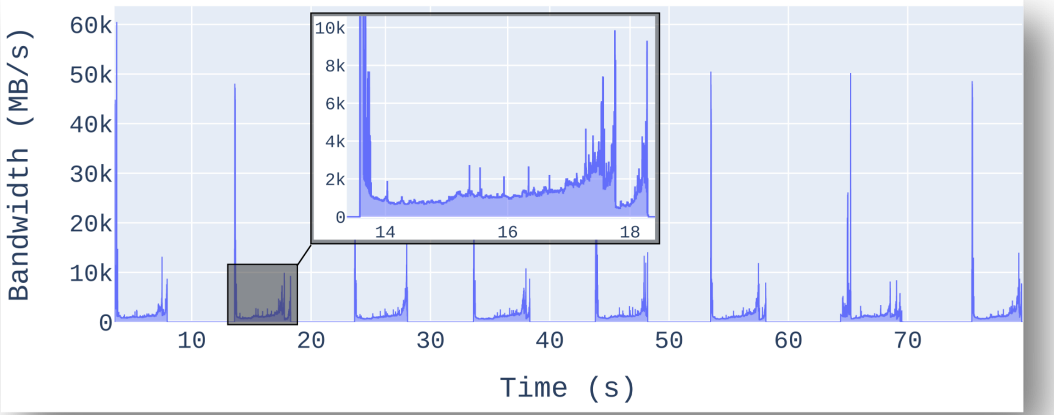
Frequency Techniques for I/O (FTIO)

Periodic I/O is often encountered in HPC

TECHNISCHE UNIVERSITÄT DARMSTADT

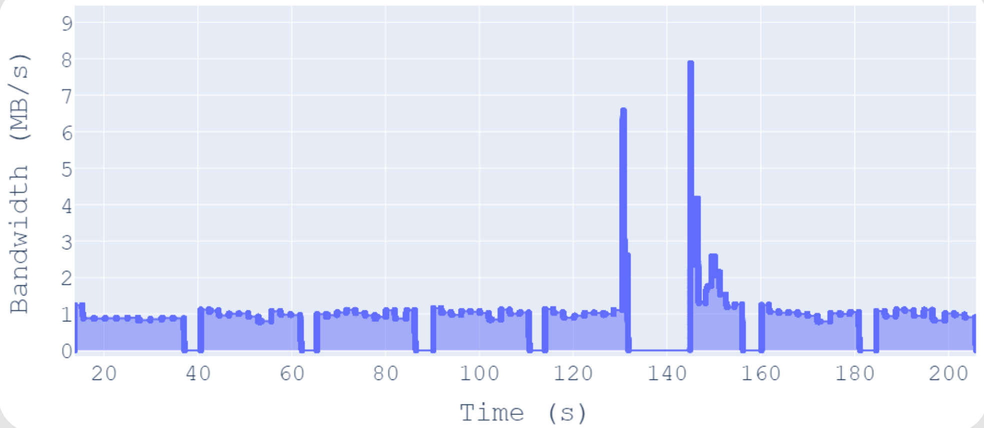
FTIO key points:

- Examine the I/O behavior in the frequency domain
- Describes the temporal behavior of the I/O phases through a single metric, namely the period (T_d)
- Online (prediction) and offline (detection)
- Additional metrics quantify the confidence in the results and further characterize the I/O behavior



Period (T_d) of I/O phases:
The time between the start of consecutive I/O phases

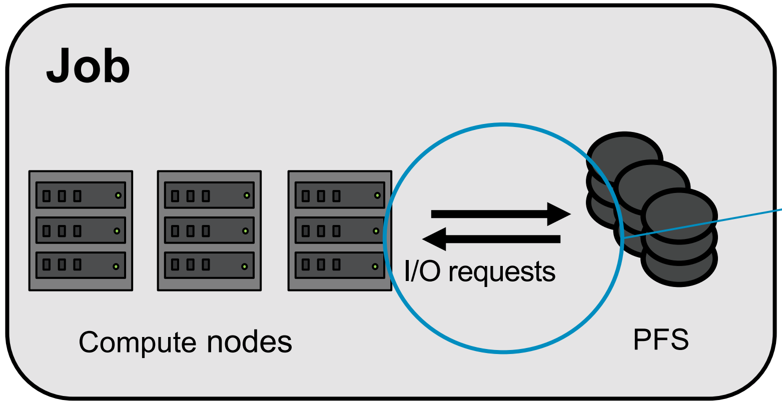
FTIO in a nutshell



signal $x(t)$

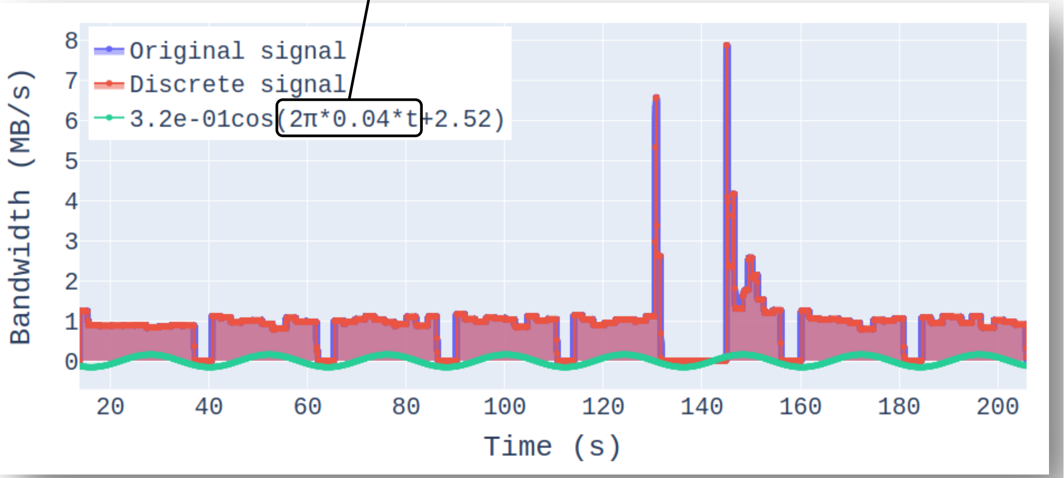
Frequency (f_d) = 0.04 Hz
→ Period (T_d) = 24.025 s

+ Confidence (c_d) = 64.9%
+ Average bytes per phase …



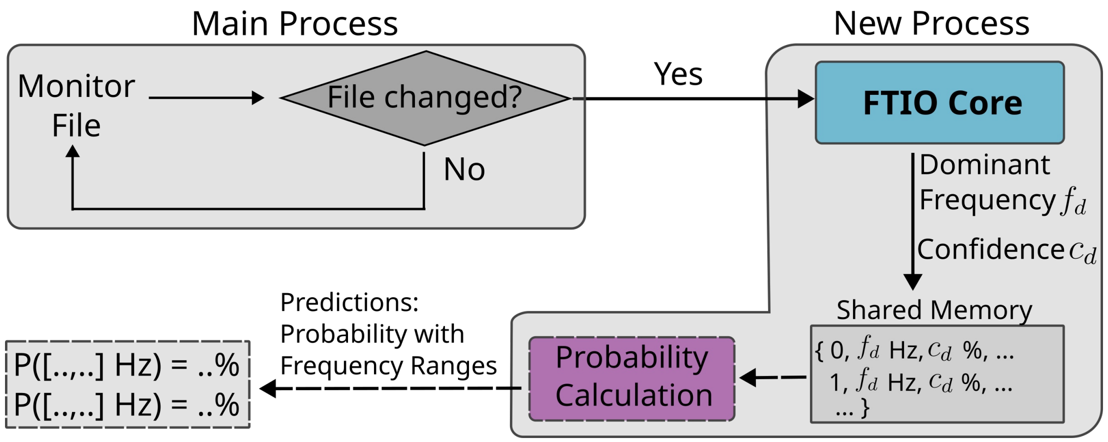
Trace File

FTIO



FTIO: Online version

- I/O per rank information send to FTIO via
  - Trace file
  - ZeroMQ (used by GekkoFS clients)
- Predicts the period during the execution
- Provides the frequency interval and confidence



More details on FTIO functionality and usage:

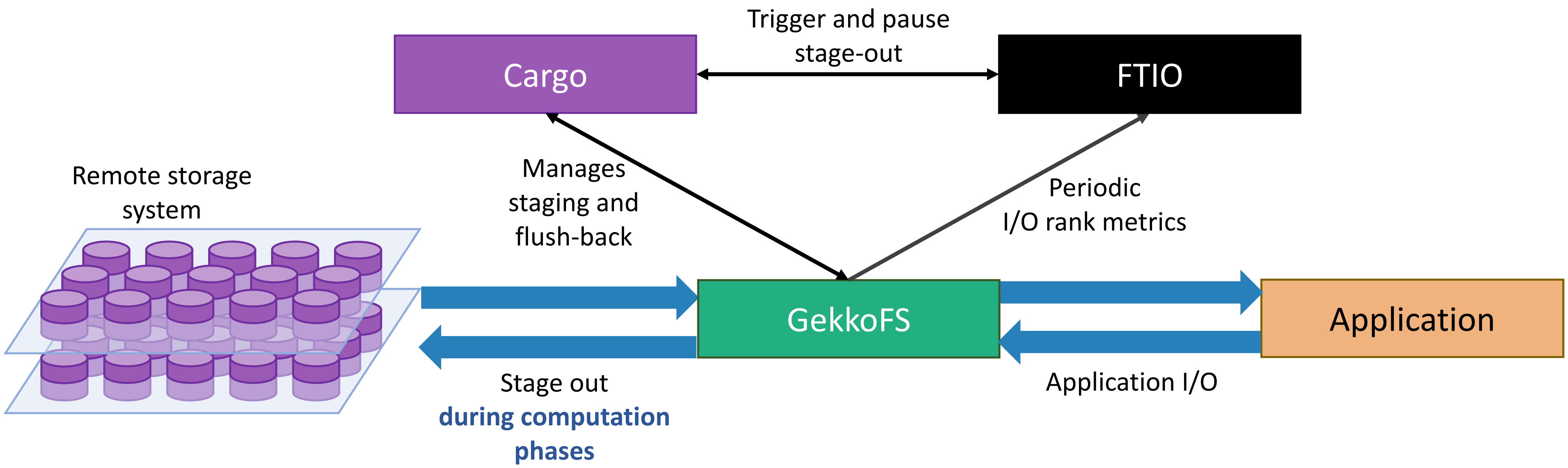
- Publication at IPDPS 2024
- Github: https://github.com/tuda-parallel/FTIO
- Youtube: https://youtu.be/QVPgXz1Kvyw

A. Tarraf, A. Bandet, F. Boito, G. Pallez, F. Wolf.
Capturing Periodic I/O Using Frequency Techniques.
In IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2024

GekkoFS, HSM, Cargo, and FTIO

Goals:

- Overlap staging with application's compute phases via FTIO triggering Cargo
- Reduce staging interference, i.e., don't stage during application I/O
- Reduce stage-out latency of ad-hoc file systems

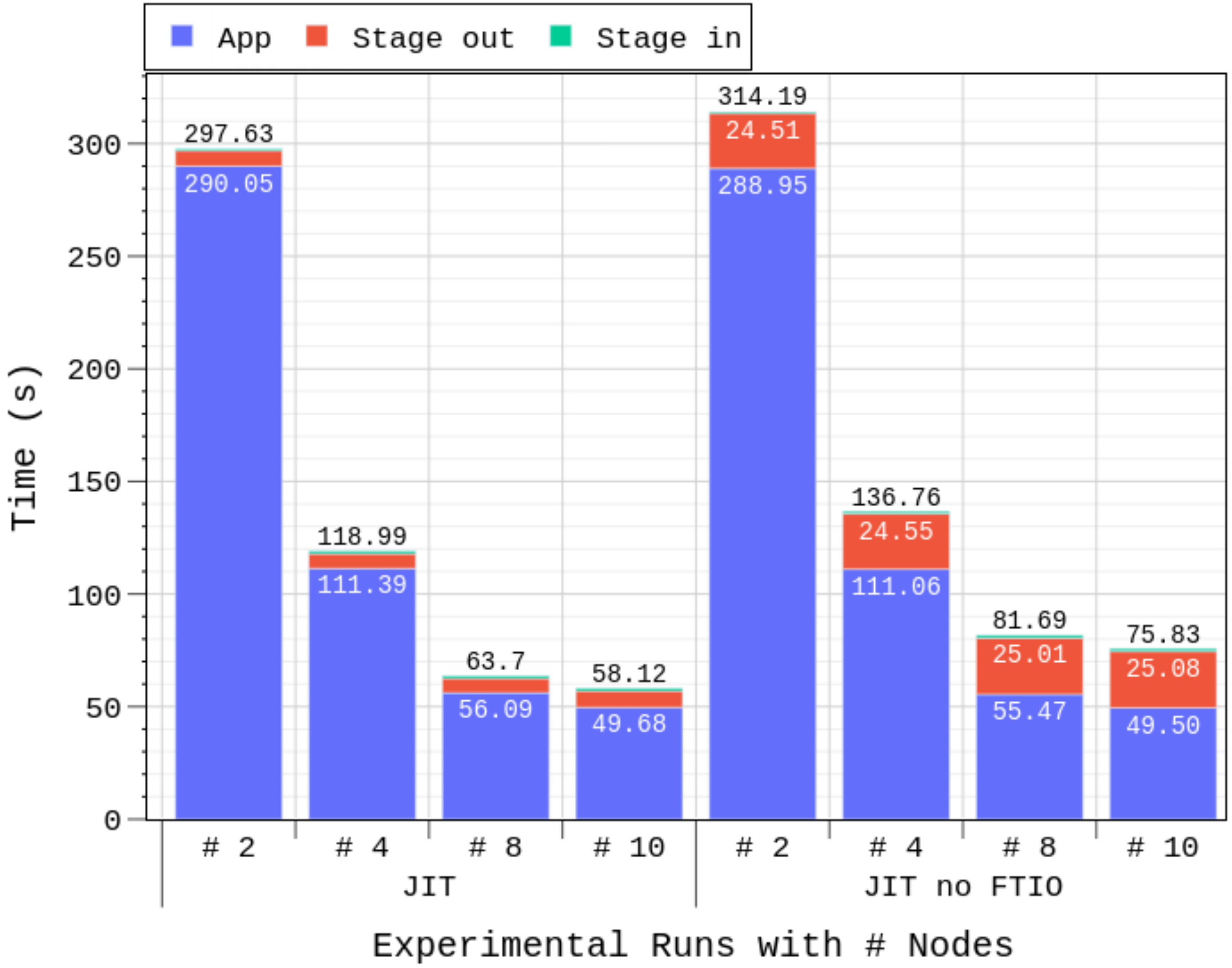


Staging with and without FTIO

- Preliminary results on a small-scale Nek5000 CFD simulation
- Nek5000 generates a single output file every 5 compute steps over 50 total steps (~700 MiB per file)
- JIT with FTIO overlaps compute phases with stage-out

Next steps:

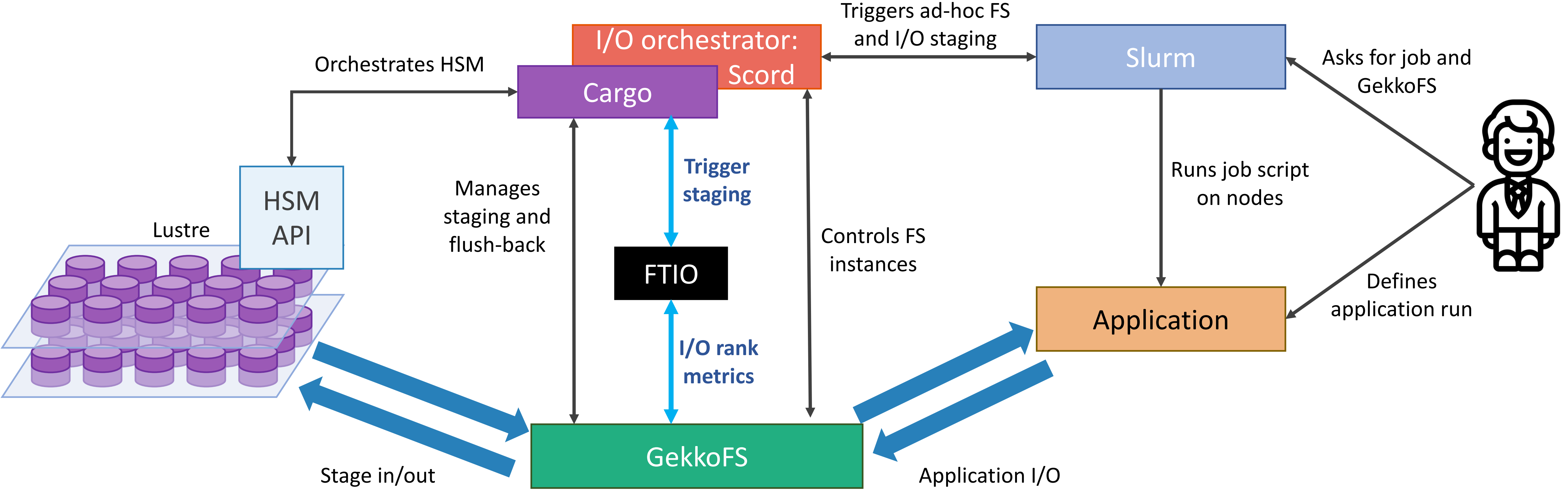
- Larger scales
- More applications
- Comparison to PFS-only runs



Full proposed architecture

Work-in-progress

- Combine ad-hoc file system transparency and intelligent data staging
- Full integration of HSM-Cargo-FTIO pending …



Conclusion

- Ad-hoc file systems can provide I/O advantages
- In their basic form, they are challenging to use
- Transparency and intelligent data staging is required
  - Transparent deployment and staging
  - Integrate ad-hoc file systems with Lustre’s HSM mechanisms in a single namespace
  - Predict I/O phases to overlap data staging with computational phases
- Next: Finalize integration and large-scale experiments
- Resources
  - I/O Trace Initiative: https://hpcioanalysis.zdv.uni-mainz.de
  - GekkoFS: https://storage.bsc.es/gitlab/hpc/gekkofs/
  - FTIO: https://github.com/tuda-parallel/FTIO
  - Scord: https://storage.bsc.es/gitlab/eu/admire/io-scheduler/
  - Cargo: https://storage.bsc.es/gitlab/hpc/cargo

Thank You!

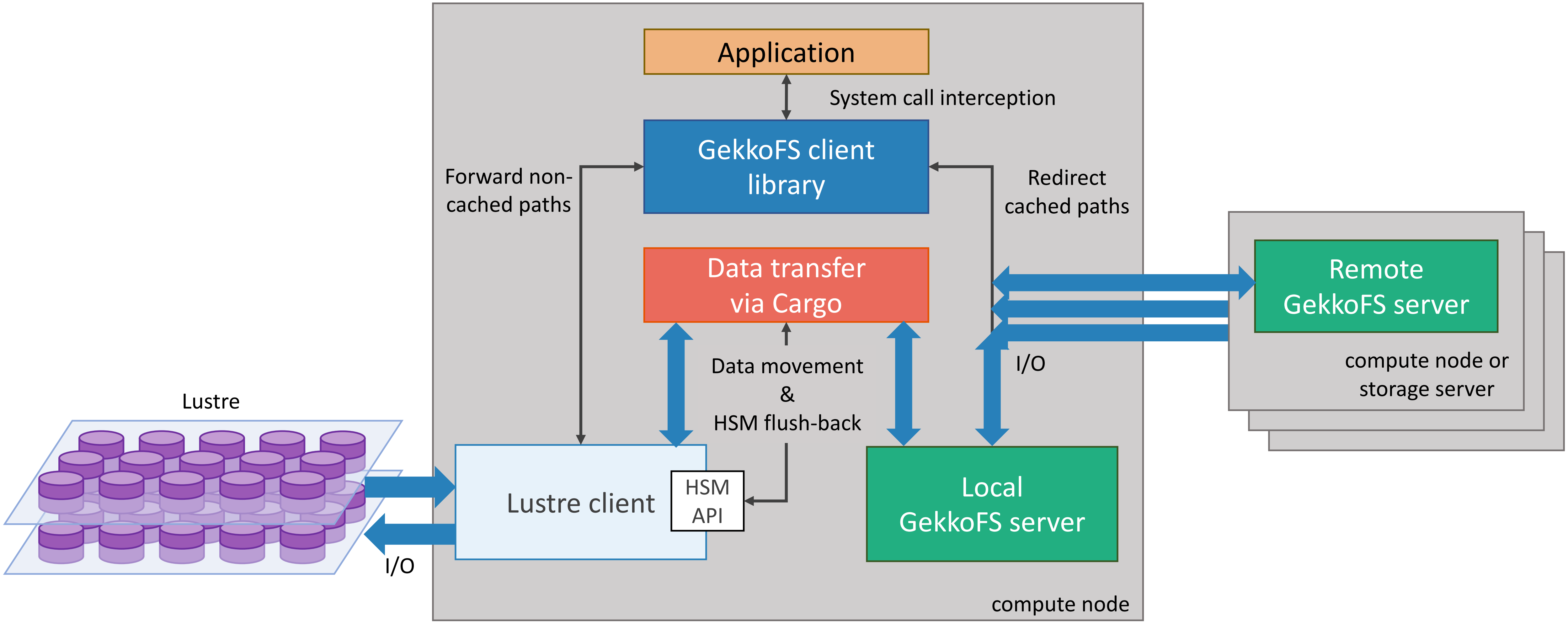
Questions?

Marc-André Vef	mvef@ddn.com
Ahmad Tarraf	ahmad.tarraf@tu-darmstadt.de
Maysam Rahmanpour	mrahmanpour@uni-mainz.de
Ramon Nou	ramon.nou@bsc.es
Reza Salkhordeh	rsalkhor@uni-mainz.de
André Brinkmann	brinkman@uni-mainz.de

Some of the icons in this presentation have been designed using free resources from flaticon.com

Lustre HSM and GekkoFS integration

Zoomed-in



FTIO core & offline version