Concurrent write support for Lustre Persistent Client Caching

Marc-André Vef, Maysam Rahmanpour, André Brinkmann

Johannes Gutenberg University Mainz, Germany

27th September 2022

LAD '22

Lustre Administrators & Developers Workshop

ADMIRE

malleable data solutions for HPC

EuroHPC Joint Undertaking

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

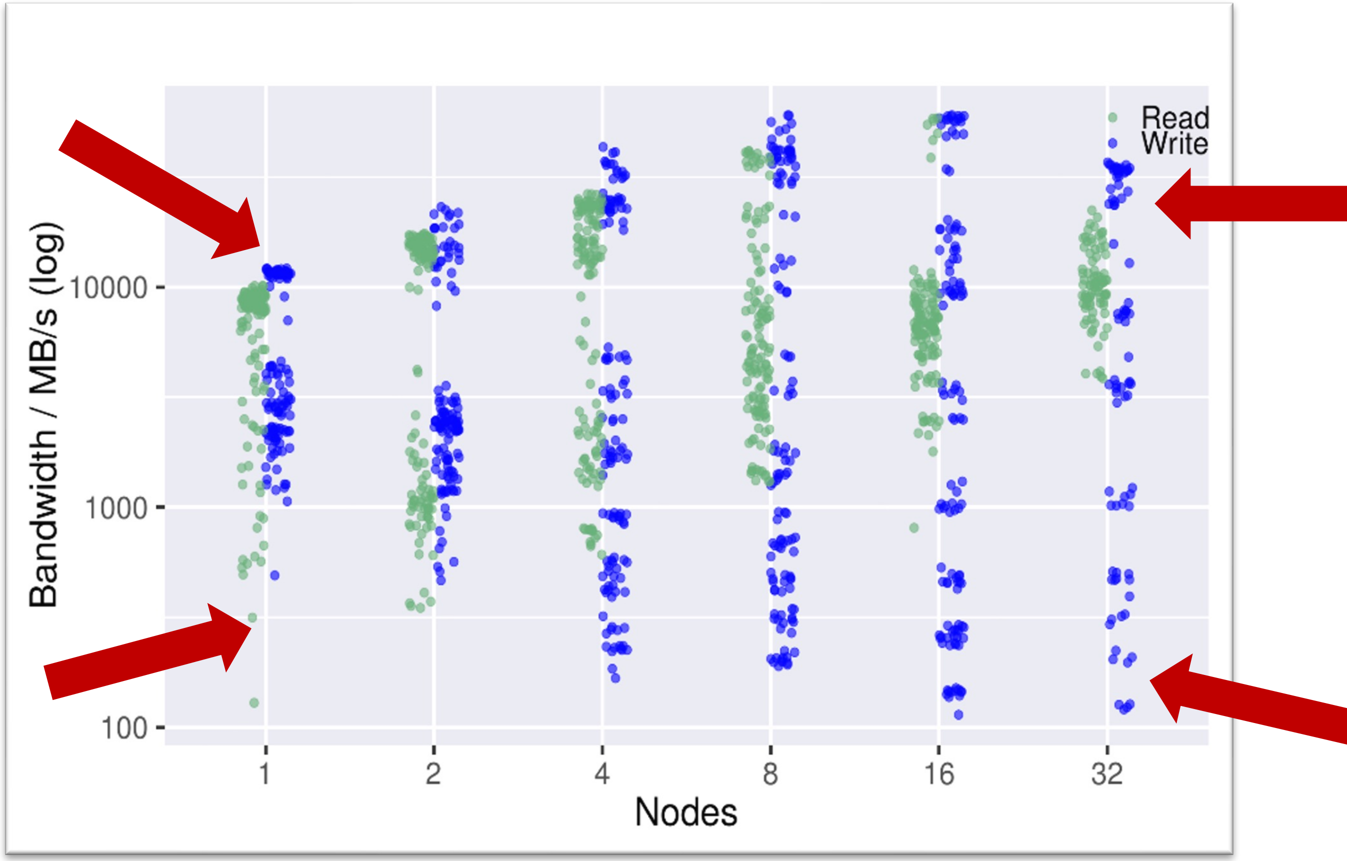
What will we talk about?

1. The cost of using the parallel file system
2. Role of ad hoc file systems
3. GekkoFS as an example
4. Ad hoc file system limitations in real life
5. Lustre Hierachical Storage Management (HSM) & Persistent Client Cache (PCC)
6. PCC limitations
7. Couple HSM, LPCC, and ad hoc file systems
8. Outlook

The cost of using the parallel file system

I/O performance varies wildly for identical workloads

Applications suffer due to PFS load!



Motivation

MareNostrum 4
Peak I/O bandwidth:
Read: 204,96 GB/s
Write: 120,89 GB/s

PFS BW per node (avg. 3456 nodes):
Read: 60,72 MB/s
Write: 35,81 MB/s

vs

Node-local Intel s3520 SSD:
Read: 450 MB/s
Write: 380 MB/s

From S. Moré, "Storage in MareNostrum 4: Petaflop System Administration" PATC 03/2019

- Minimize arbitrary PFS usage: exploit the available I/O stack
- Minimize redundant data movement and schedule transfers to reduce PFS contention
- Improve data locality: Do work where data lives!

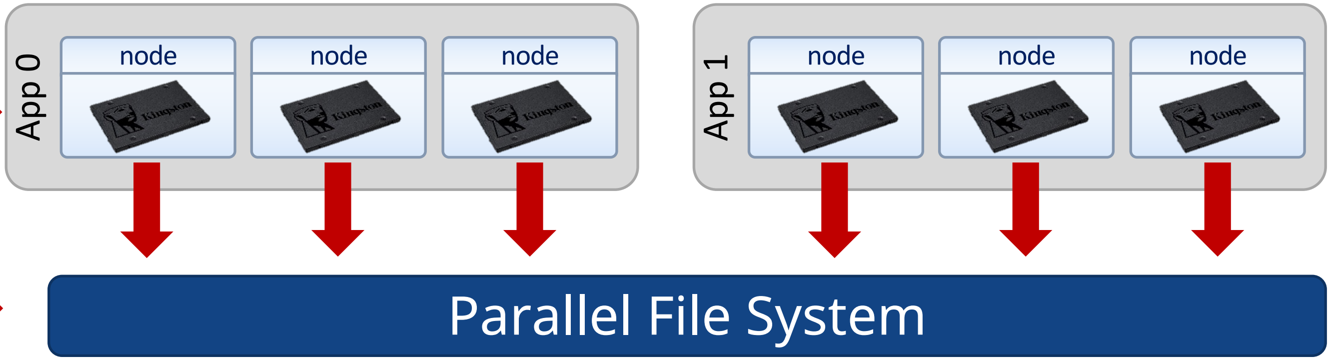
Goal

Data manipulations rely on the PFS

- Uncoordinated application I/O to/from PFS
- Node-local storage typically <u>ignored</u>
- Increased PFS contention and performance variability

node-local storage mostly unused

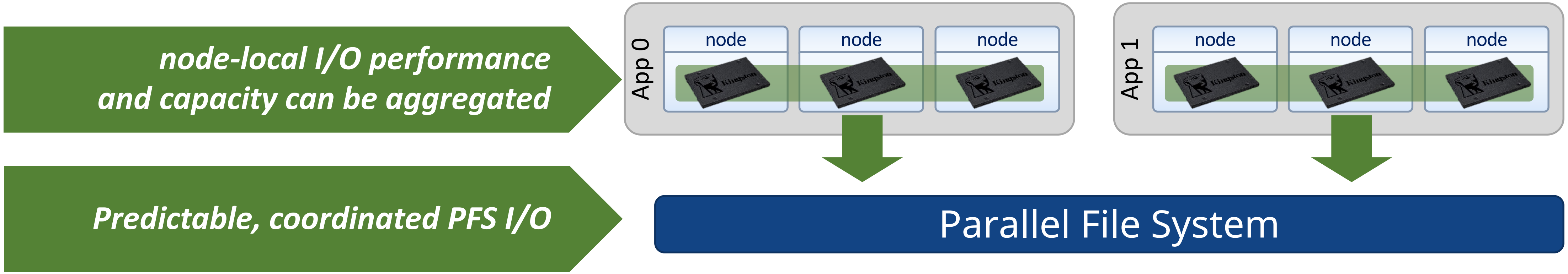
unpredictable and random PFS I/O



Goal

Data manipulations rely on node-local storage

- Coordinated application I/O: sequential stage-in (read) and stage-out (write) from/to PFS
- Harmful I/O patterns are absorbed by node-local storage
- Reduced PFS contention and performance variability

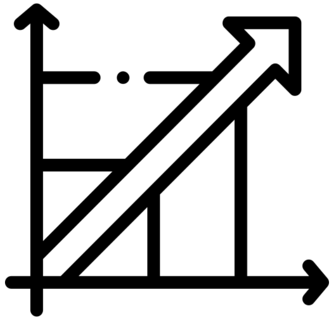


Marc-André Vef, Maysam Rahmanpour, André Brinkmann

GekkoFS

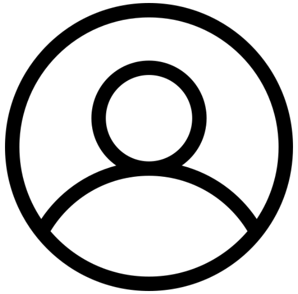
as an exemplarily ad hoc file system

Core challenges to be addressed



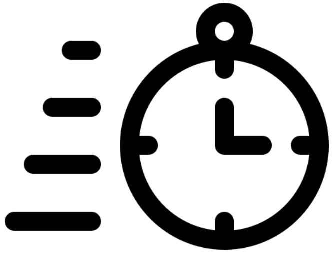
- No central components
- Linear scaling with # number

1. **Scalability**



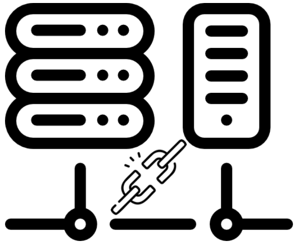
- User decides
- No administrative support

3. **User space**



- Wall time is important
- <10 seconds for deployment

2. **Fast deployment**



- Use accessible storage
- Use fast network fabrics

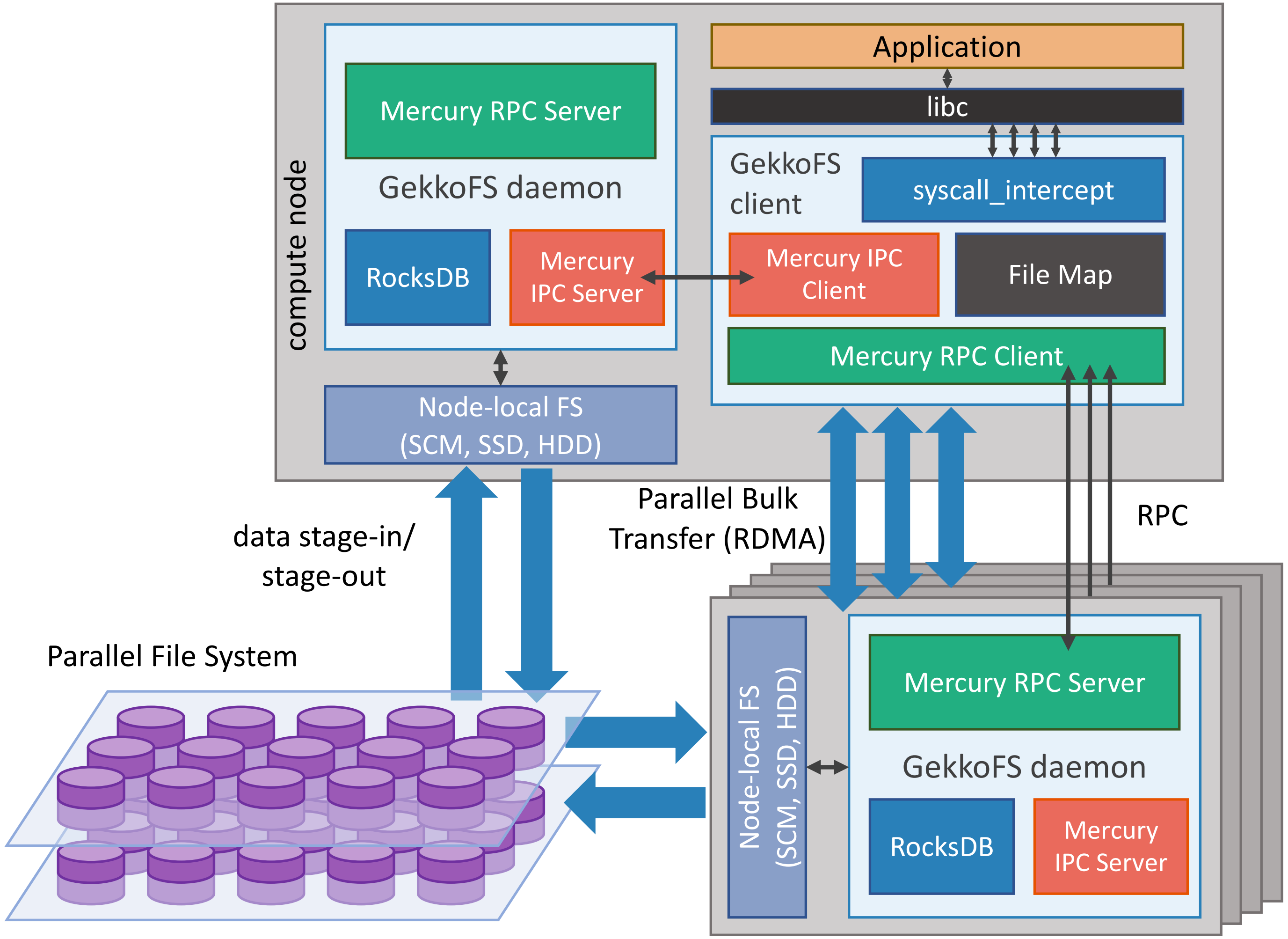
4. **Hardware independence**

GekkoFS architecture

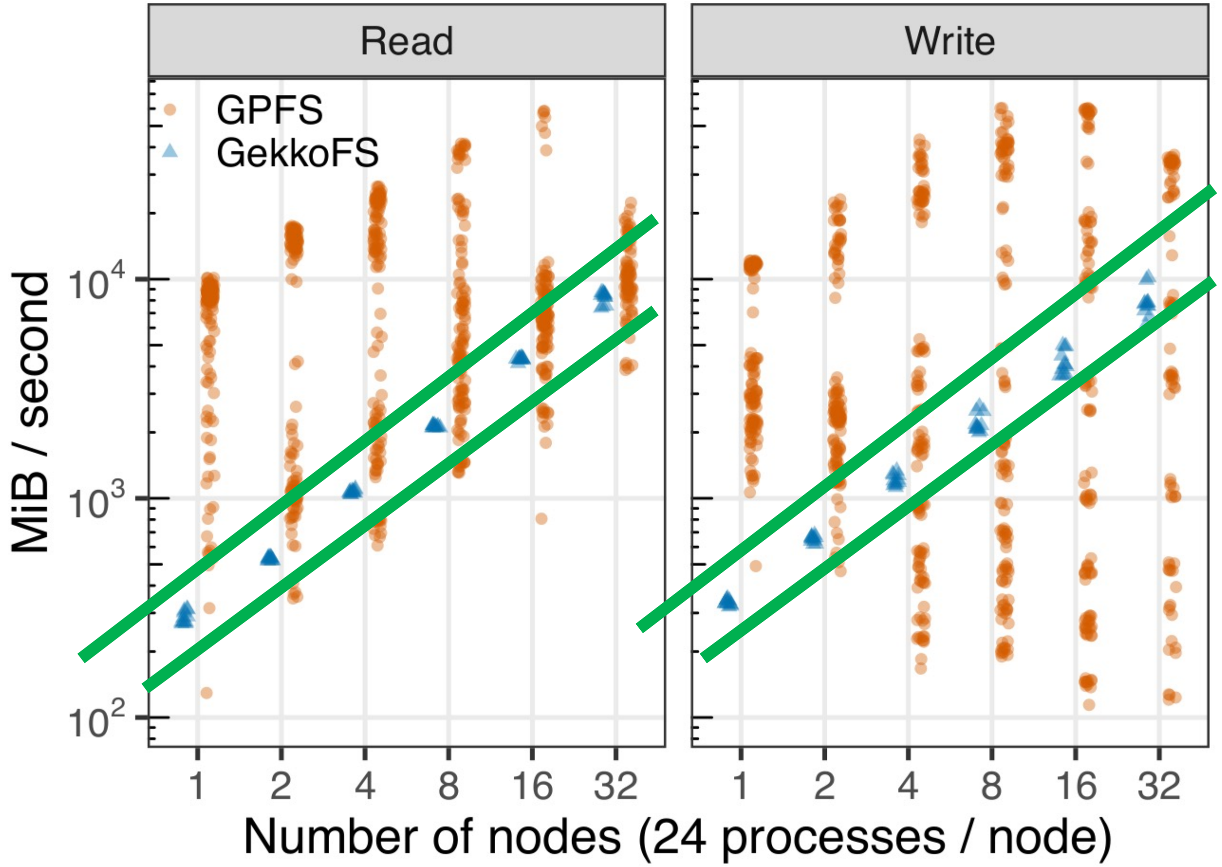
Mercury
A high-performance RPC framework from ANL
https://mercury-hpc.github.io

RocksDB
A persistent key-value store for fast storage from Facebook
http://rocksdb.org

syscall_intercept
A system call interception library from Intel
https://github.com/pmem/syscall_intercept



Performance variability revisited (MN4)

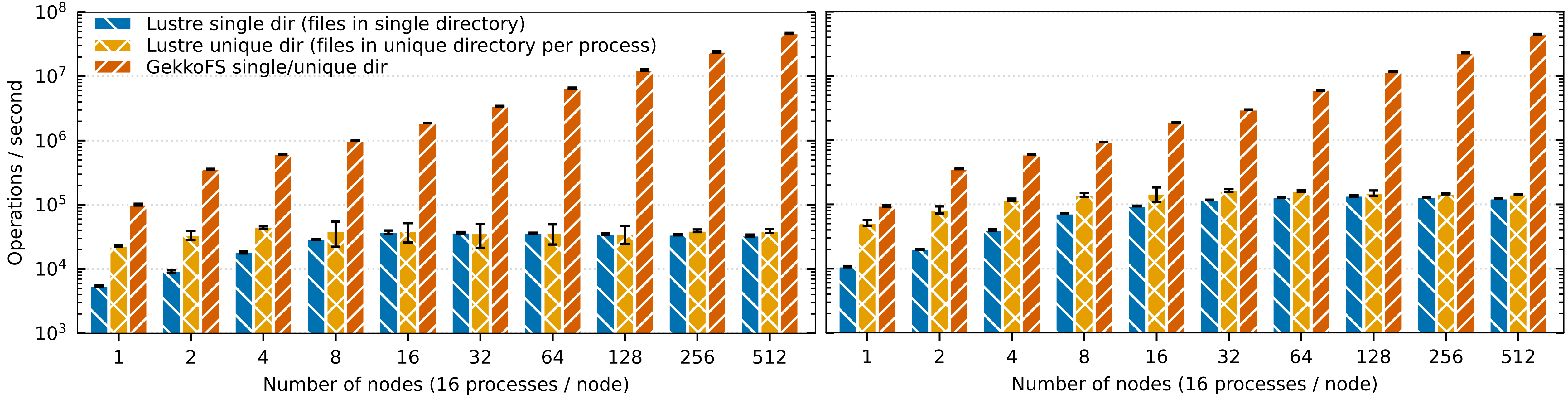


I/O performance variability is greatly reduced

Metadata performance

GekkoFS vs. BeeGFS @ MOGON II

- GekkoFS weakly scaled (100K files per process)
  - More than 819 million files in total at 512 nodes for GekkoFS



File create performance

File stat performance

Ranked 4th in IO500 10-node challenge @ SC'19

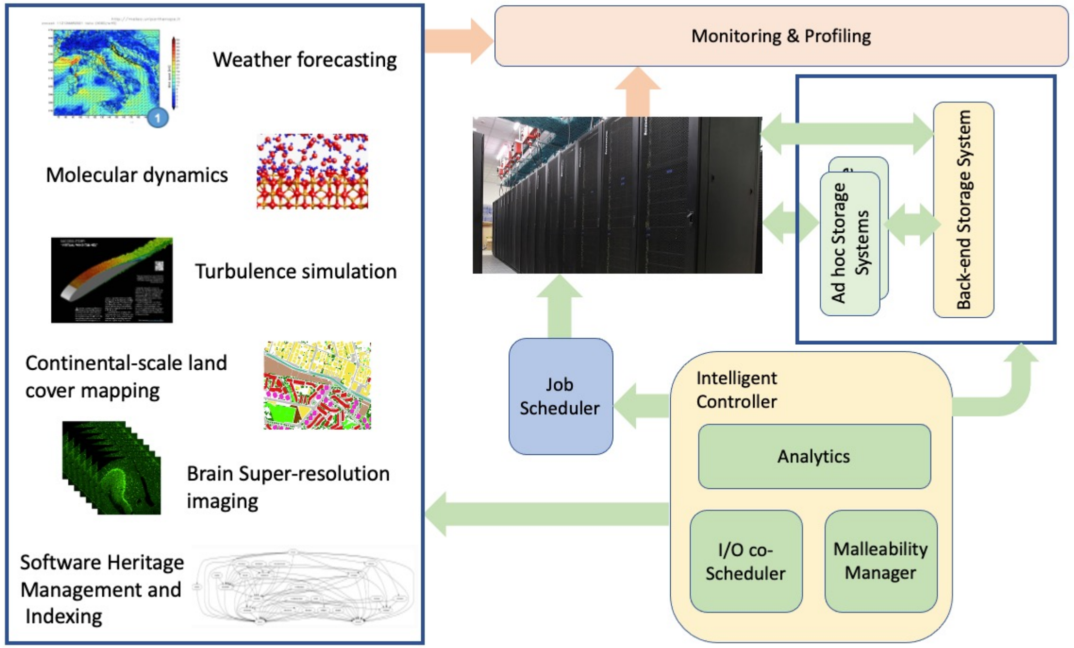
Ad hoc file systems in real life

Challenges and possible solutions

- **Not** transparent usage and requires user interaction
  - Starting and stopping ad hoc file system
  - Data staging
  - 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
  - Lustre integration (DDN and JGU collaboration)

EuroHPC Joint Undertaking

ADMIRE malleable data solutions for HPC



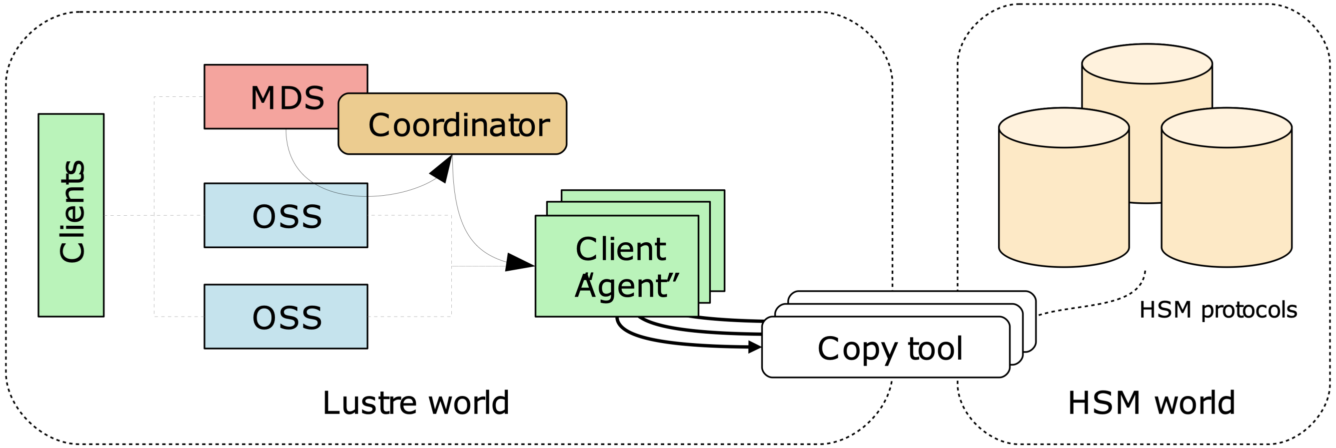
EuroHPC ADMIRE project architecture.
https://admire-eurohpc.eu

Proposal:
Combine the benefits of Lustre HSM, PCC, and ad hoc file systems

Lustre
Hierarchical Storage Management
&
Persistent Client Cache

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 flushback 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)

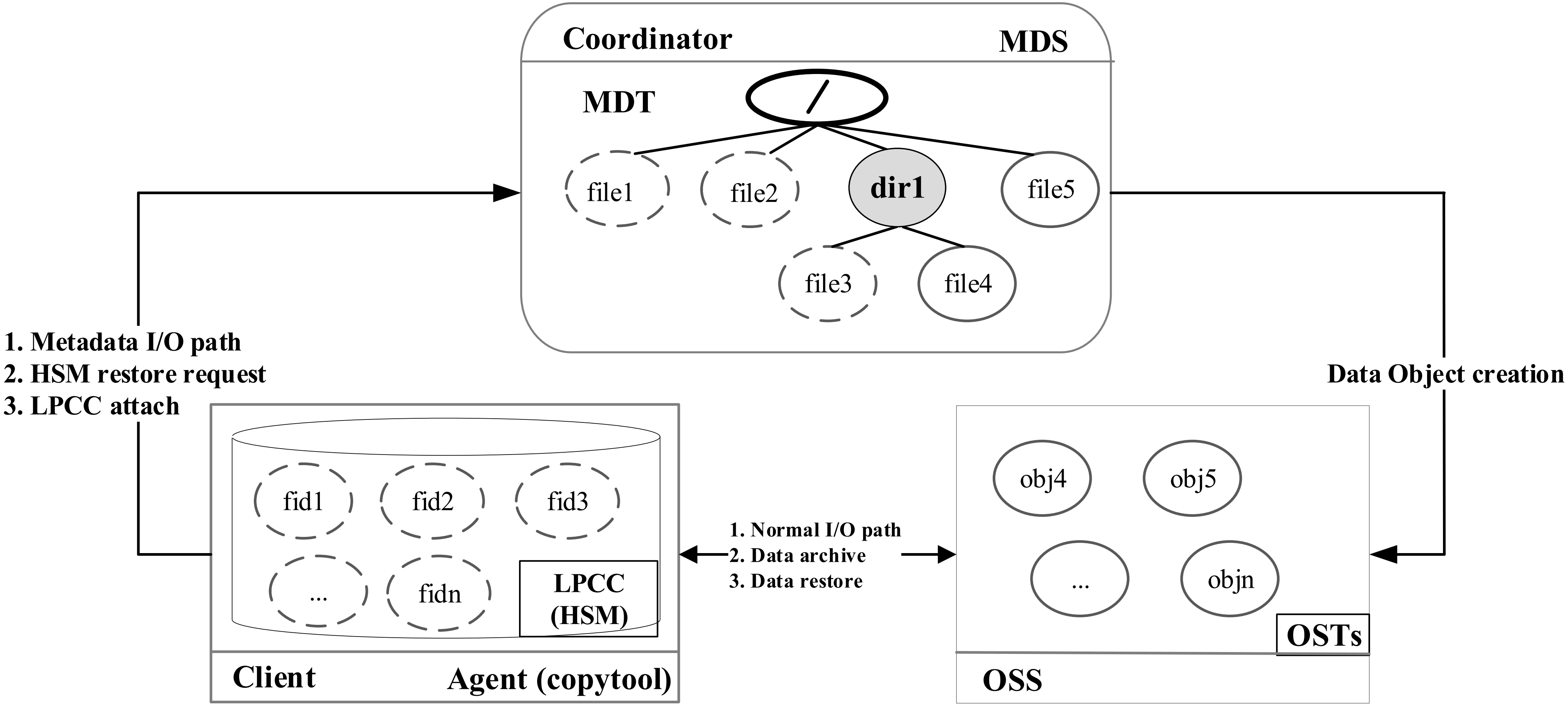
Motivation and goals

- Node-local storage media often remain **unused**
- **Transparently** include fast node-local storage into Lustre
- **Increase** I/O performance for I/O workflows and **decrease** I/O interference

Features

- LPCC integrates into established HSM mechanisms
- Layout lock mechanism to provide consistent cache services
- Maintain global unified namespace
- Two caching modes
  - RW-PCC: read-write cache on **single** client
  - RO-PCC: read-only cache on **multiple** clients

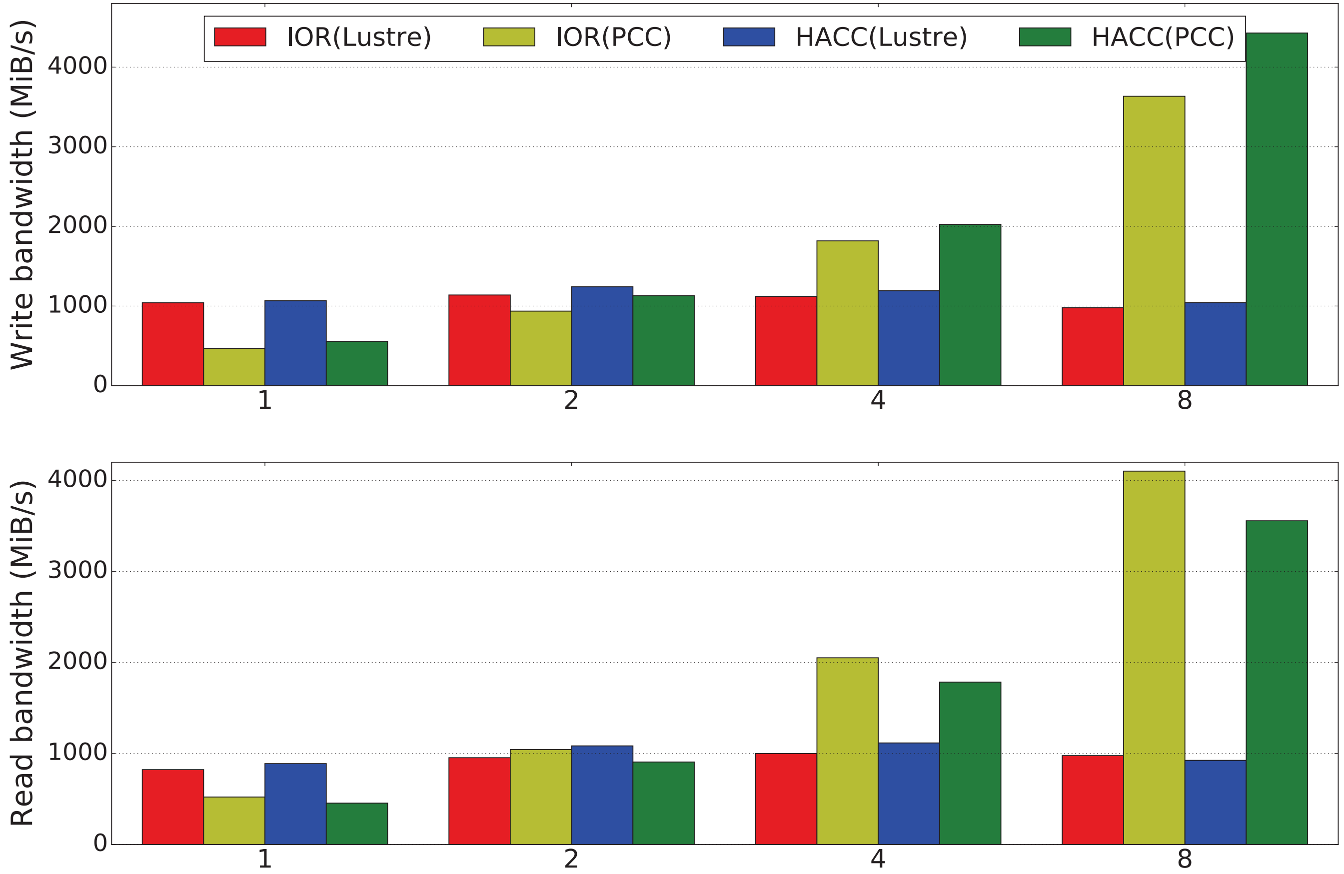
Lustre Persistent Client Caching (LPCC)



Overview of LPCC architecture

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: RW-PCC scalability evaluation

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.*

LPCC limitations

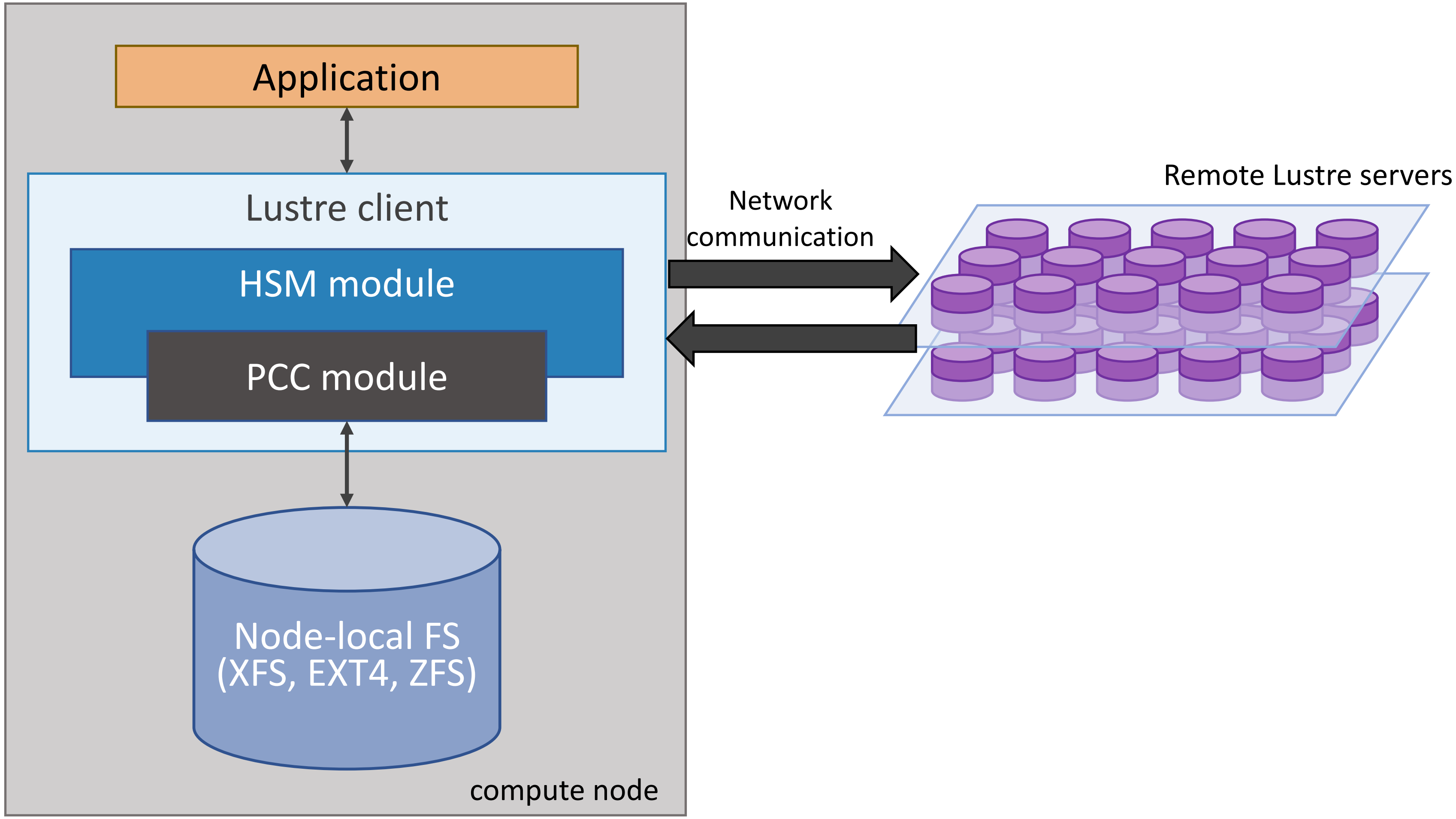
- LPCC offers caching in the context of a single node
- RW-PCC: One node can use the same resource
  - **No conflicting access allowed**
  - ➢ No parallel I/O from many nodes possible
- RO-PCC: Multiple nodes can cache the same resource
  - **Same access allowed but redundant data**
  - ➢ Can cause severe I/O overhead on parallel file system when many nodes cache the same data
- Cache capacity and I/O performance **restricted** by node-local storage
- Metadata (except file size) is **not** cached

Distributed ad hoc file systems can offer a solution to these limitations

The naïve coupling approach

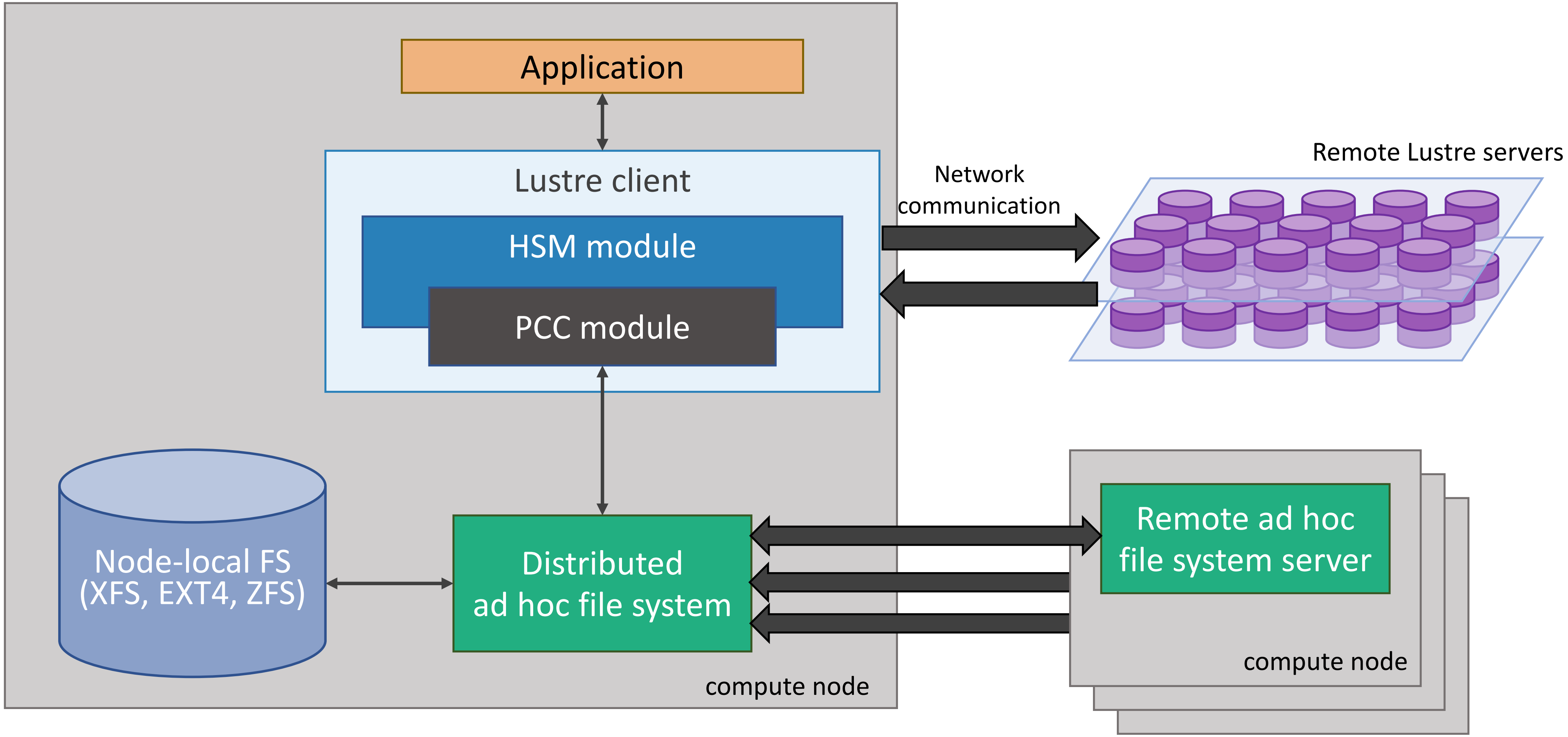
The naïve approach

- Replace node-local storage with distributed storage



The naïve approach

- Replace node-local storage with distributed storage



The naïve approach

- Requires **extensive** implementation effort at core Lustre components
  - RW/RO-cache code logic must be **rebuild**
  - **Redirection** of I/O accesses from remote nodes to ad hoc file system instance
- **No benefit** for metadata workloads
- ➢ **Significant overhead** for Quality Assurance
- **Too specific** for a general "Lustre – ad hoc file system" coupling approach

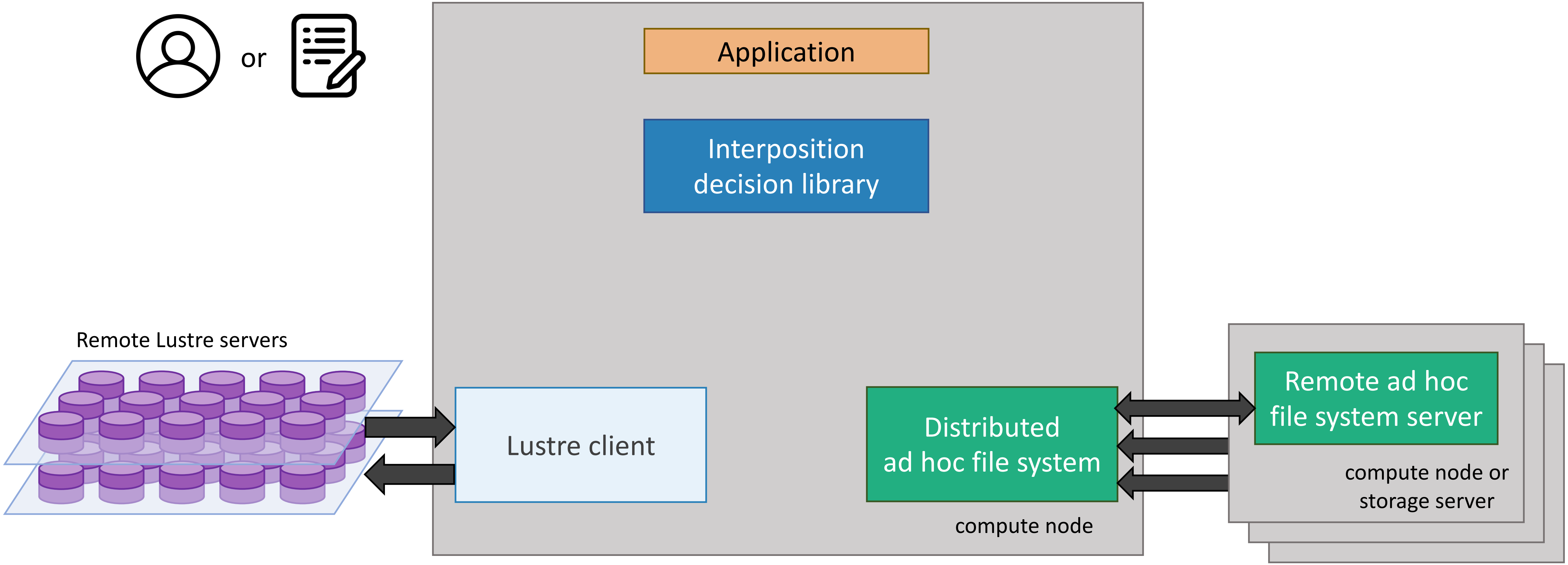
Overall unsatisfactory solution

Proposal

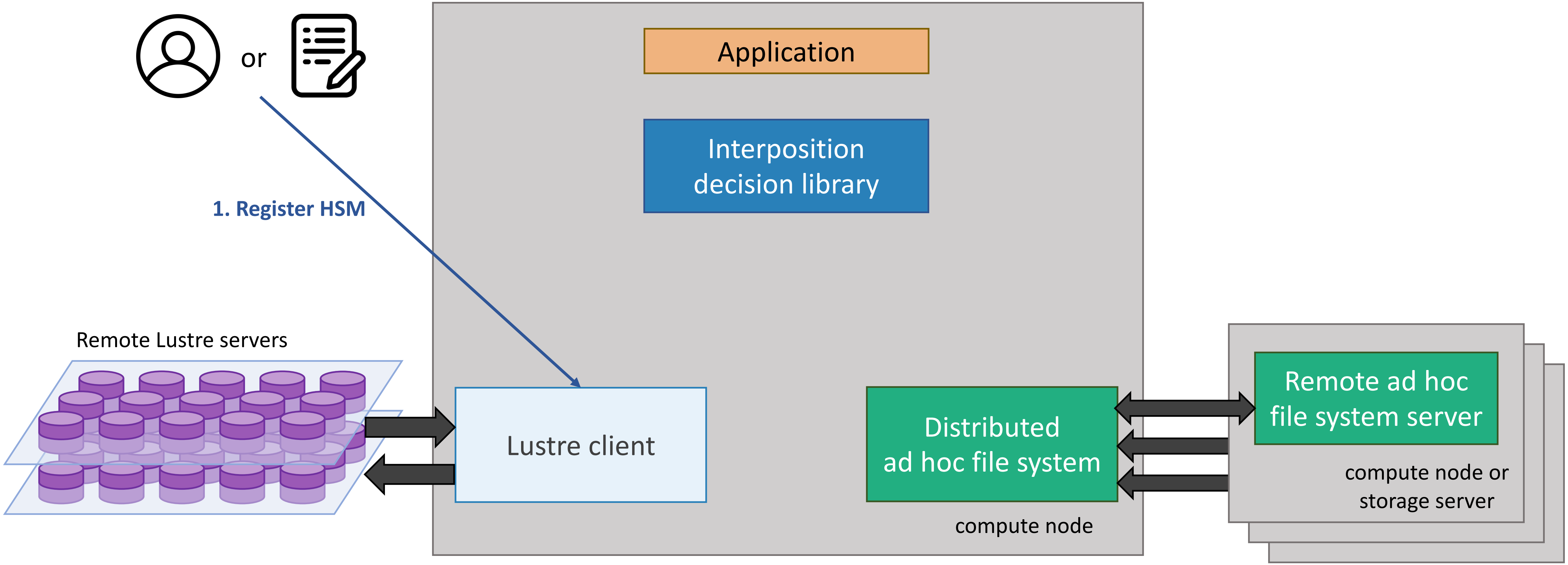
Proposal

- Instead, **build on** existing Lustre's HSM mechanisms
- **Avoid** extensive Lustre modifications
- **Reuse** LPCC's policy monitor and prefetching algorithms
- Semi-**transparent**: Users and administrators can define caching behavior
- **Leverage** GekkoFS's interception layer
- **Compatibility** with any distributed ad hoc file system

Proposal

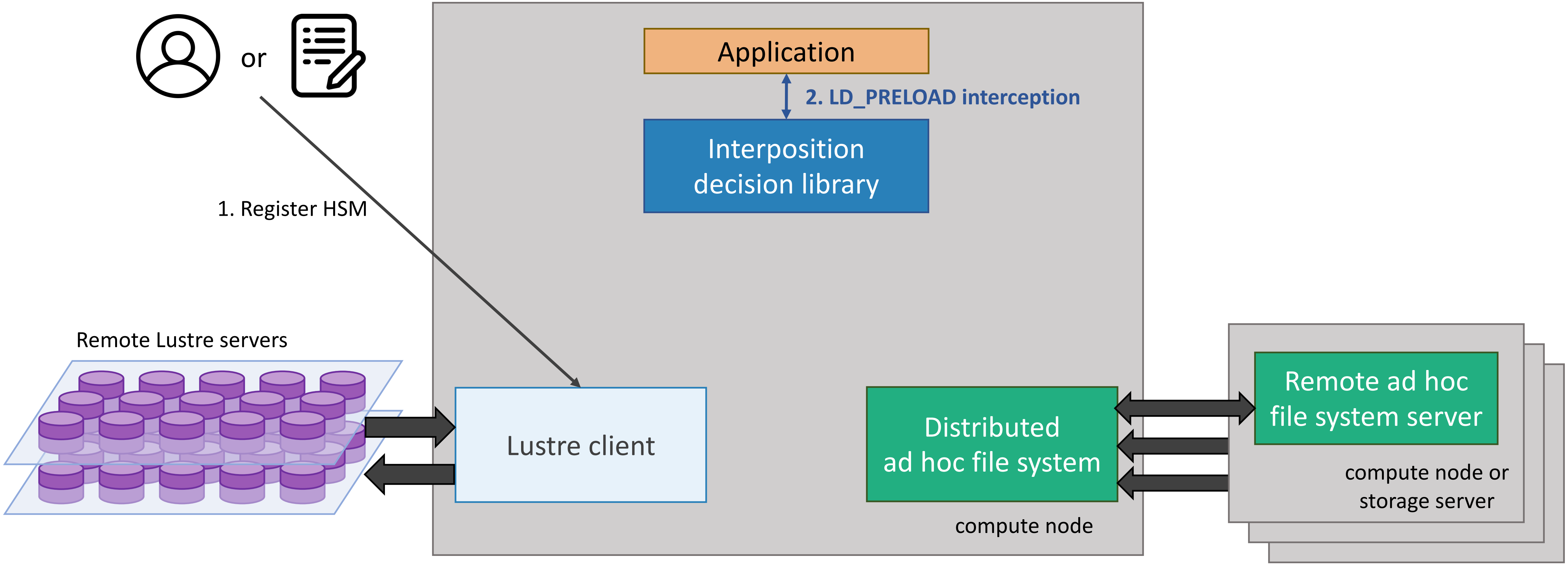

or
Application
Interposition decision library
Remote Lustre servers
Lustre client
Distributed ad hoc file system
compute node
Remote ad hoc file system server
compute node or storage server

Proposal



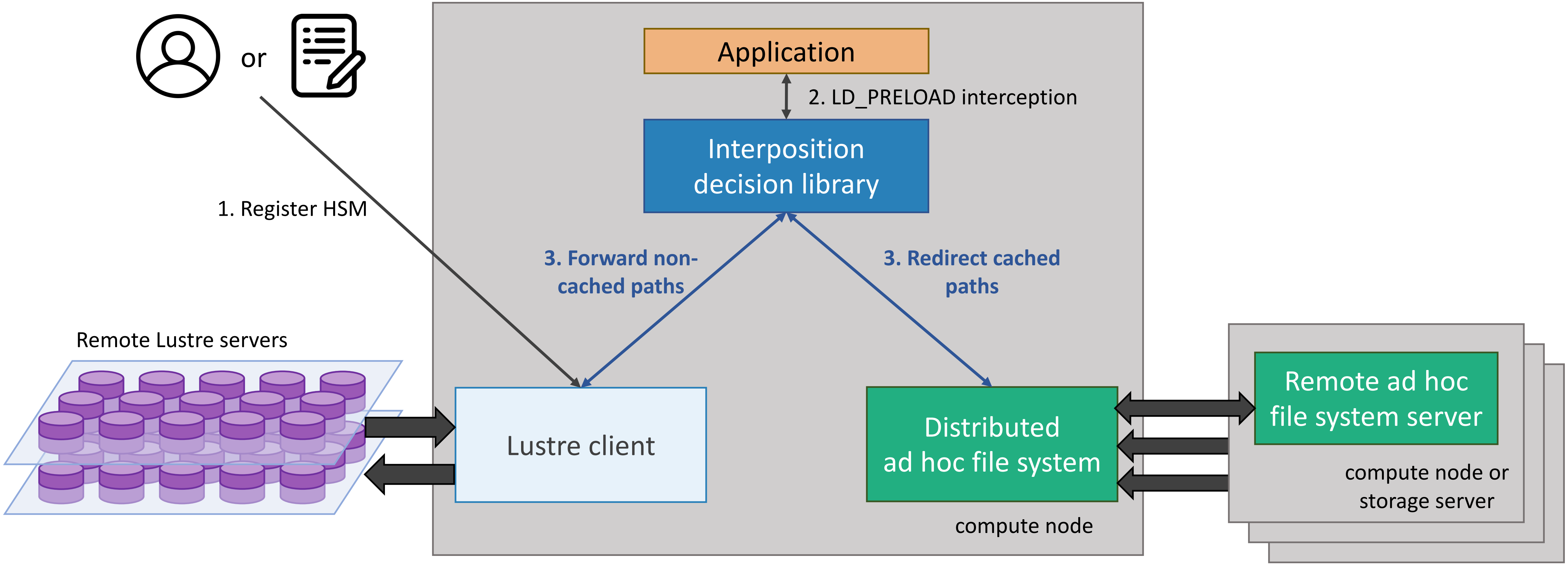
1. Register HSM paths (without release) to be handled by ad hoc file system

Proposal



2. Application preloads decision library

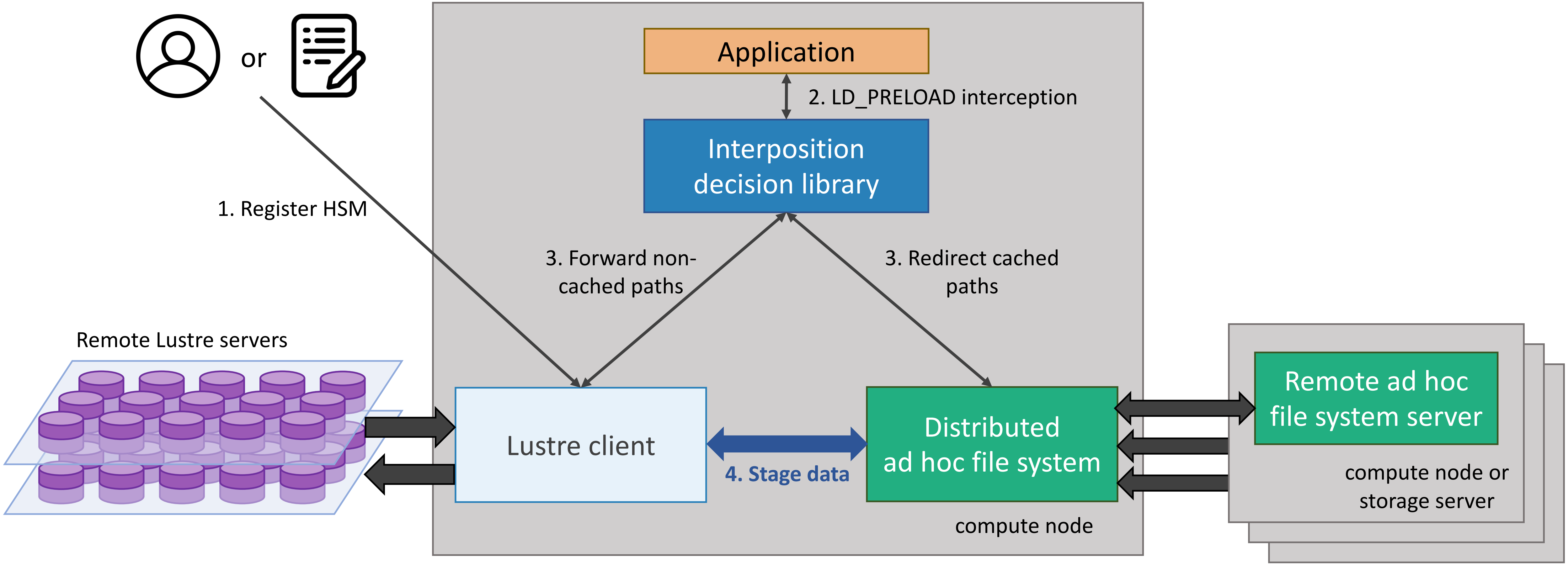
Proposal



3. Decision library branches requests to Lustre or ad hoc file system

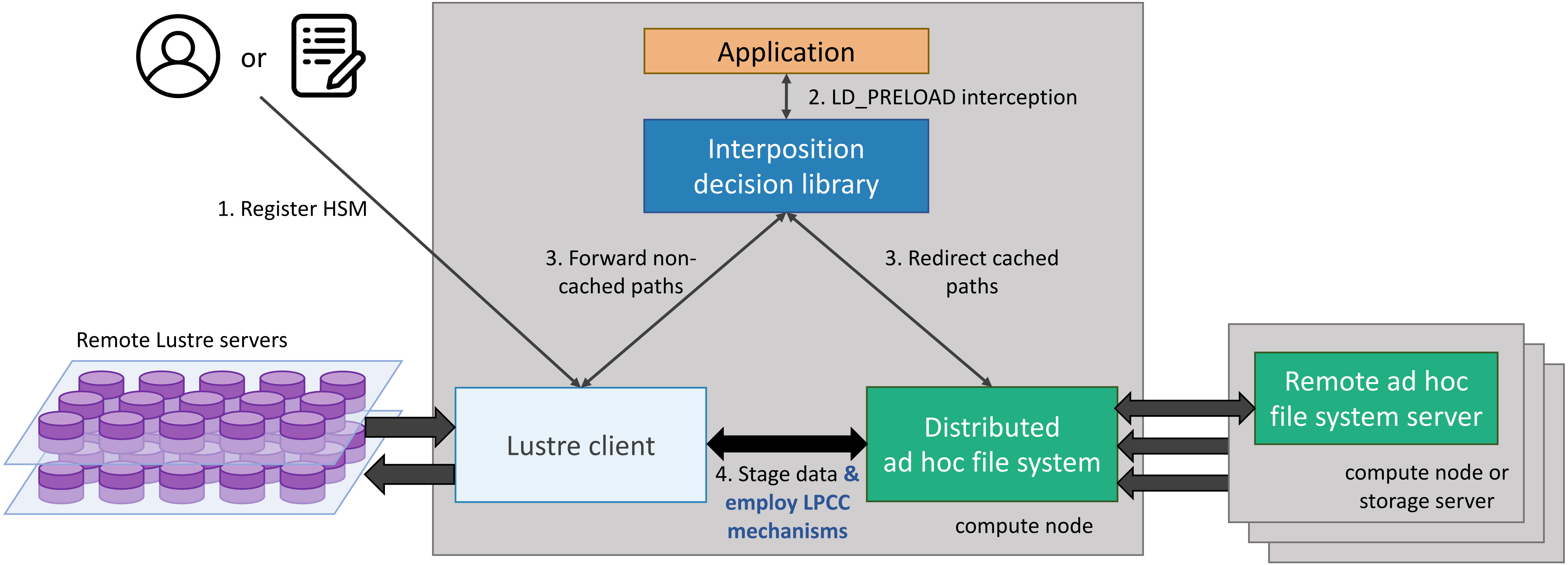
Marc-André Vef, Maysam Rahmanpour, André Brinkmann

Proposal



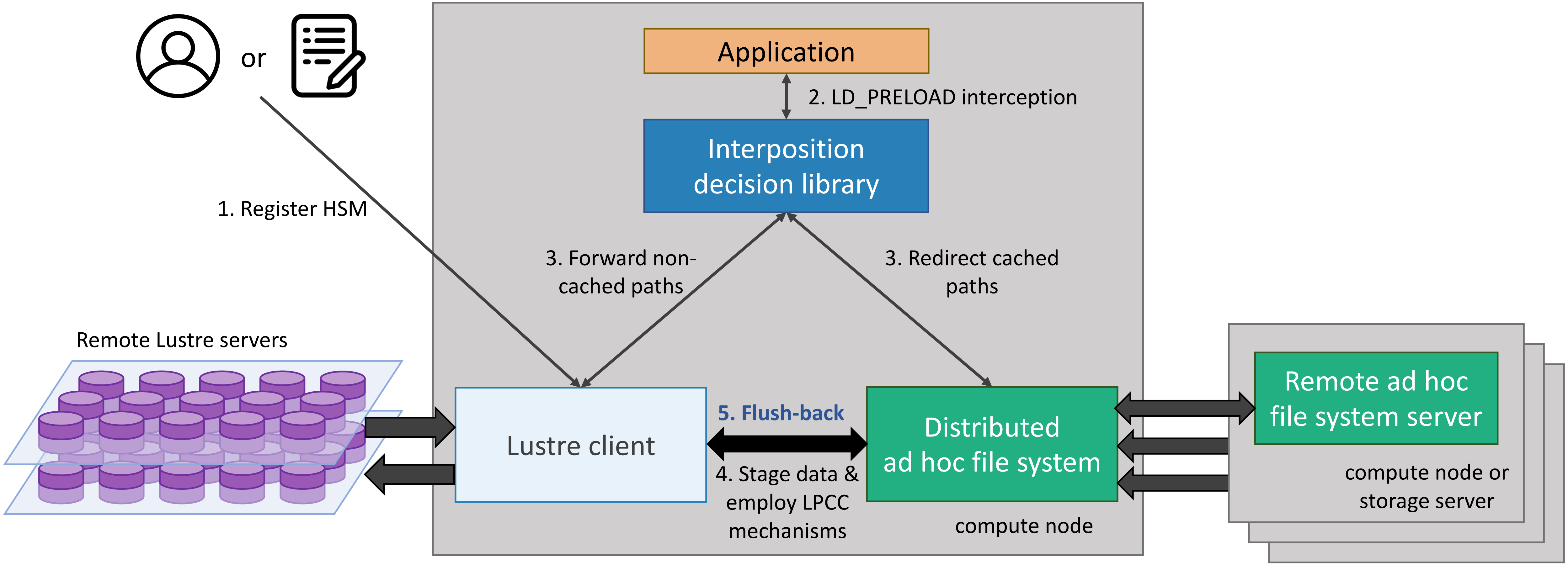
4. Lazily stage data from Lustre to ad hoc file system

Proposal



4. (cont.) Leverage Lustre policy monitor, copy tool, and prefetching (TBD)

Proposal



5. Flush-back to Lustre on conflicting access

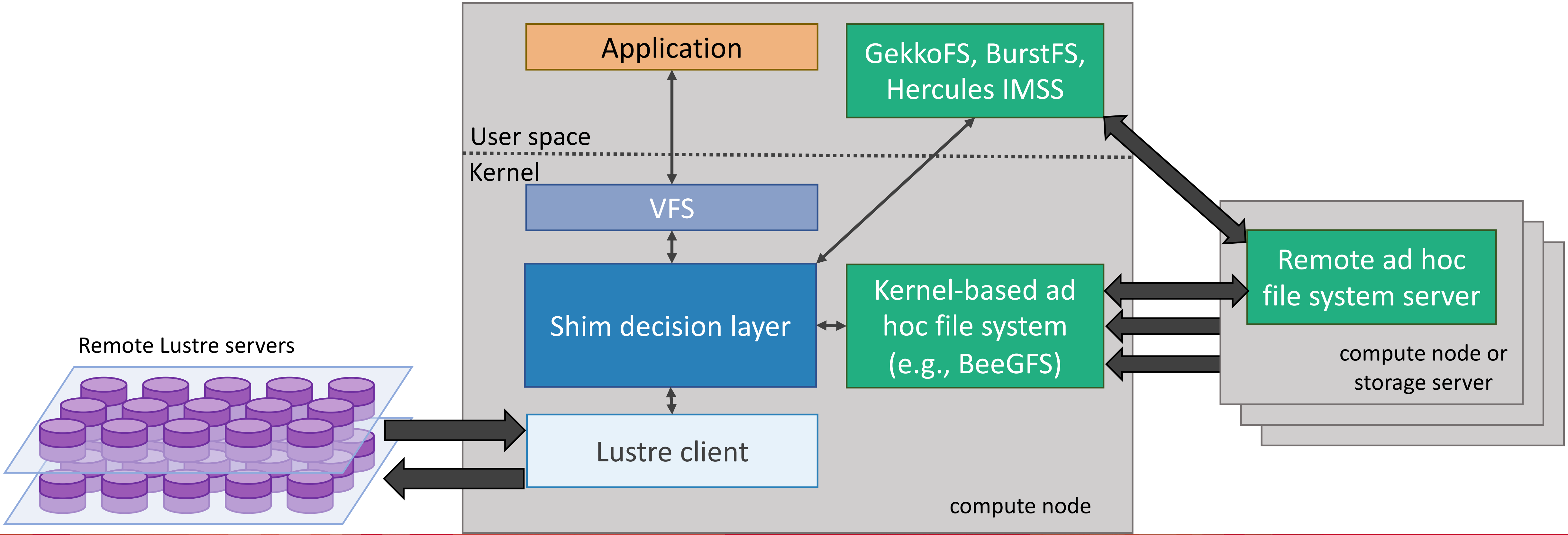
Possible future Kernel integration

Architecture

- Interposition library can **cause issues** for some applications
  - ➢ Kernel-based solution **preferred**
- Move decision library to the Kernel as a **shim layer** (or pseudo file system)

Architecture

- Interposition library can **cause issues** for some applications
  - ➢ Kernel-based solution **preferred**
- Move decision library to the Kernel as a **shim layer** (or pseudo file system)



Conclusion

- Ad hoc file system can **relieve I/O load** and **interference** of parallel file systems
- Due to their architectures, they often provide **linear scalability**
- But ad hoc file system usage remains a **challenge** in practice
  - Transparency
  - Data staging
- Lustre HSM and LPCC can **help**, giving staging responsibility to the ad hoc file system
- Discussion
  - What are the pitfalls you expect?
  - What are your requirements to use such a system?
  - What are the typical use cases you see?

We greatly appreciate any feedback!

Thank You

JGU

- Marc-André Vef vef@uni-mainz.de
- Maysam Rahmanpour mrahmanp@uni-mainz.de
- André Brinkmann brinkman@uni-mainz.de

Acknowledgements: Supported by

Some of the icons in this presentation have been designed using resources from flaticon.com by the authors Freepik and Bingge Liu