ADMIRE malleable data solutions for HPC

Adaptive multi-tier intelligent data manager for Exascale

EuroHPC Joint Undertaking

admire-eurohpc.eu

ADMIRE Users Day

What are ad-hoc storage systems and the GekkoFS burst buffer file system

Marc-André Vef – JGU Mainz

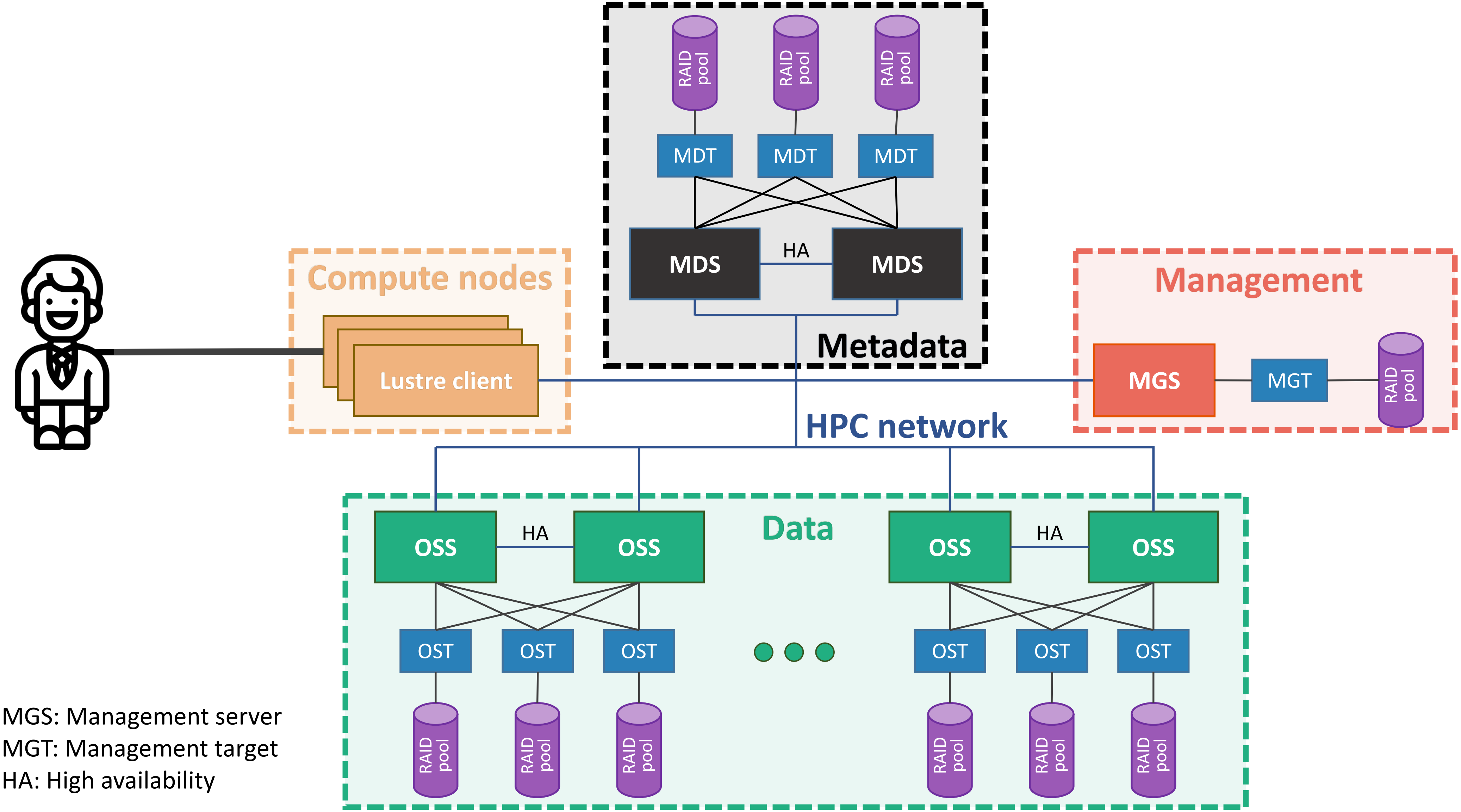
December 12th 2023

Barcelona Supercomputing Center

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

JG|U

The HPC parallel file system architecture (e.g., Lustre)

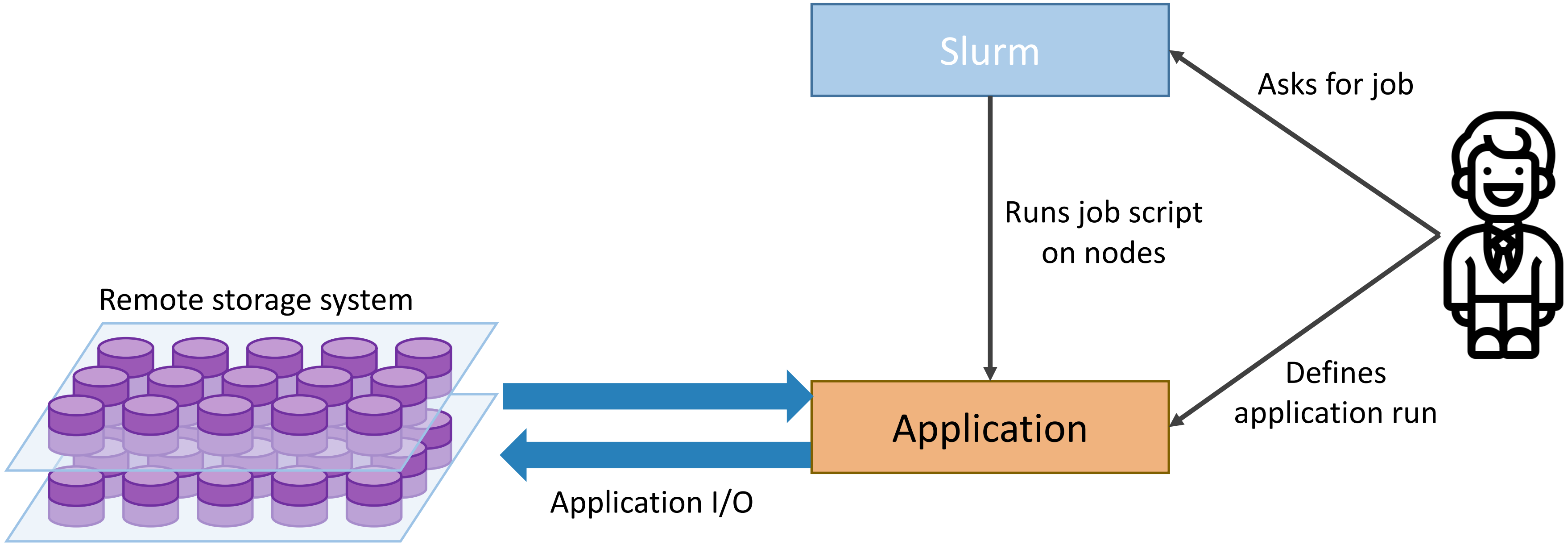


OSS: Object storage server
OST: Object storage target
MDS: Metadata server
MDT: Metadata target

MGS: Management server
MGT: Management target
HA: High availability

How application use the PFS

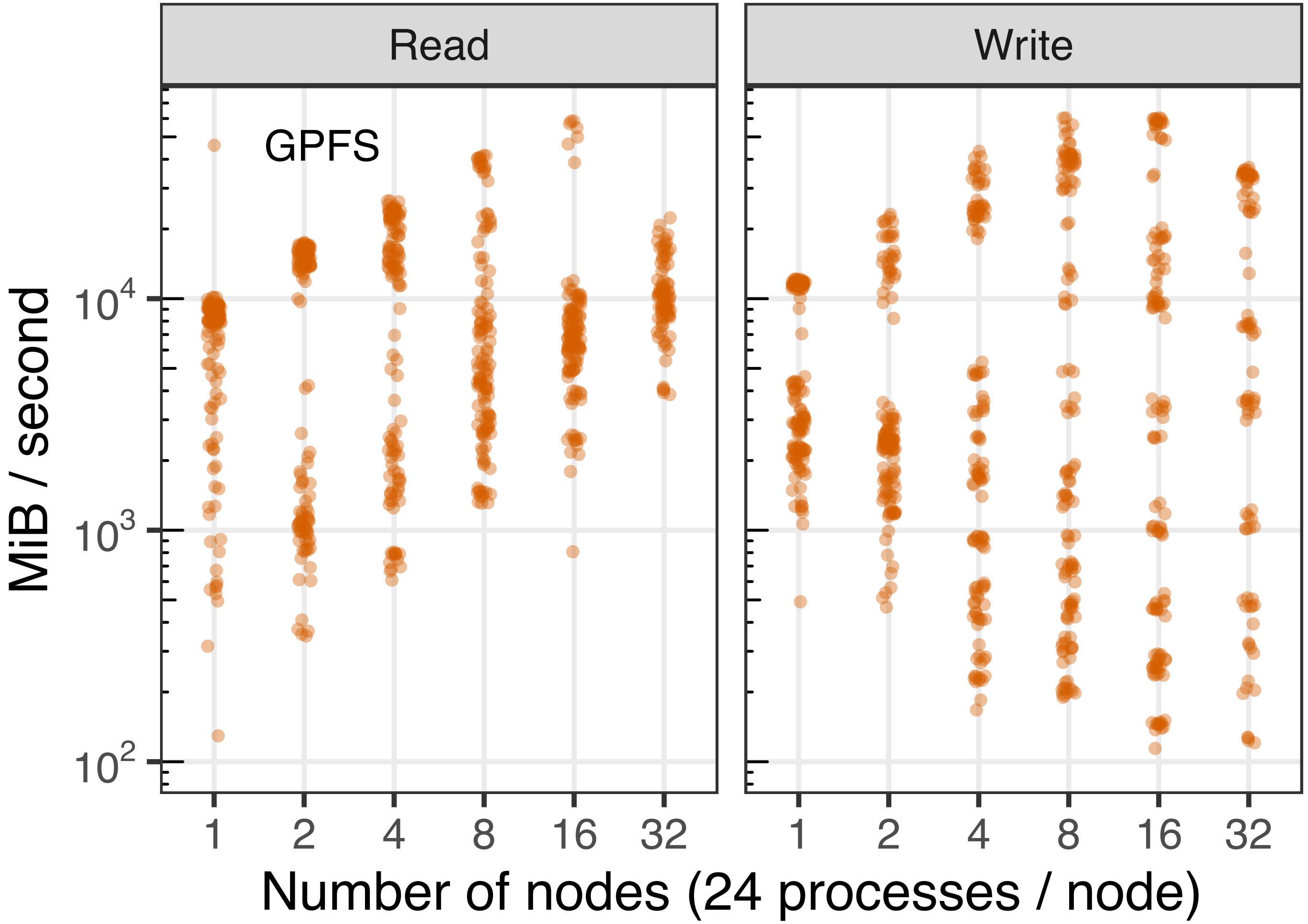
- User run applications with the PFS just as any other local file system



The cost of using the parallel file system (PFS)

I/O performance varies wildly for identical workloads

Applications suffer due to PFS load!



GPFS on MareNostrum 4 at the Barcelona Supercomputing Center

Can node-local storage help?

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

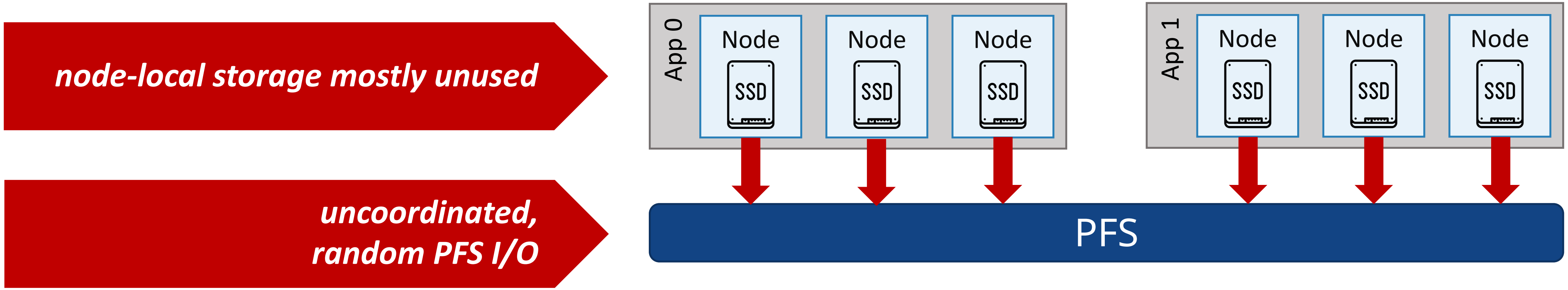
From S. Moré, "Storage in MareNostrum 4: Petaflop System Administration" PATC 03/2019

- Include often unused node-local storage into the HPC storage hierarchy
- Deploy a light-weight distributed file system per job
  - Temporary life-time
  - Input/Output are stage-in/staged-out
- Only offer FS features which are required by most (<u>not all</u>) applications
- Improve data locality: Do work where data lives!

Moving from this …

Data manipulations rely on the PFS

- Uncoordinated application I/O to/from PFS
- Increased PFS contention and perf. variability
- Node-local storage is typically ignored



node-local storage mostly unused
uncoordinated, random PFS I/O
App 0
App 1
Node
SSD
PFS

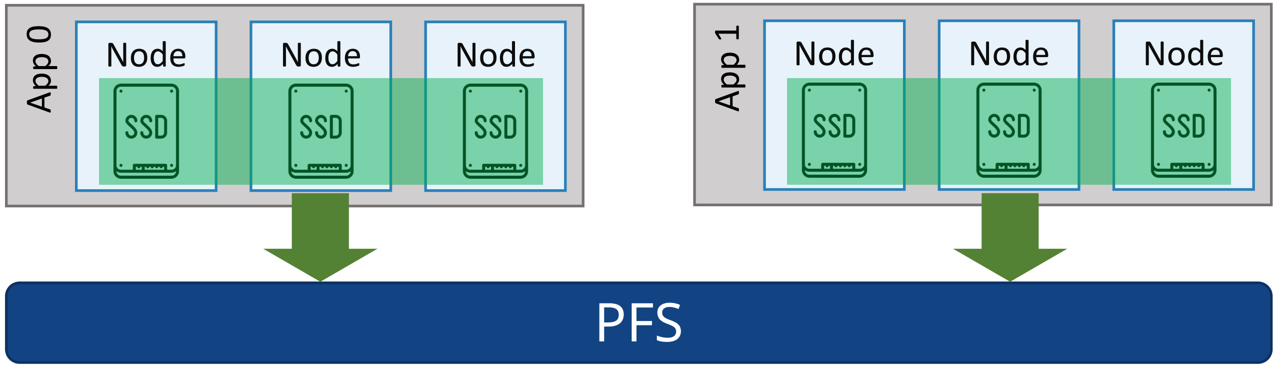
... to this

Data manipulations rely on node-local storage

- Deploy a distributed file system *ad-hoc*
  - Sequential stage-in (read) from the PFS
  - Sequential stage-out (write) to the PFS
- Harmful I/O is absorbed by node-local storage
- Reduced PFS contention and perf. variability

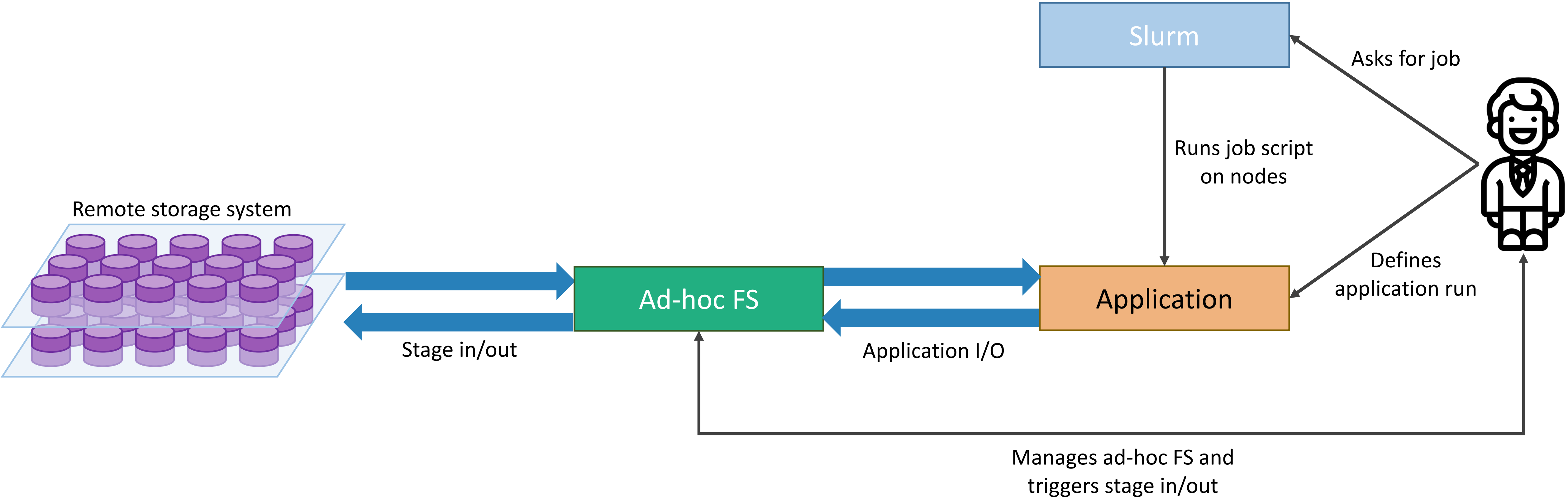
node-local I/O performance and capacity can be aggregated

predictable, coordinated PFS I/O



How application use ad-hoc FSs before ADMIRE?

- User must start ad-hoc FS and move data themselves



GekkoFS

Goals

1. **Scalability**
   - Linear scaling with number of nodes
   - Relax POSIX semantics

2. **User space**
   - User decides without administrative support
   - No VFS restrictions

3. **Fast deployment**
   - Wall time is important
   - Less than 1 minute for deployment

4. **Hardware independence**
   - Use available node-local storage
   - Support for native network protocols

Architecture

Mercury

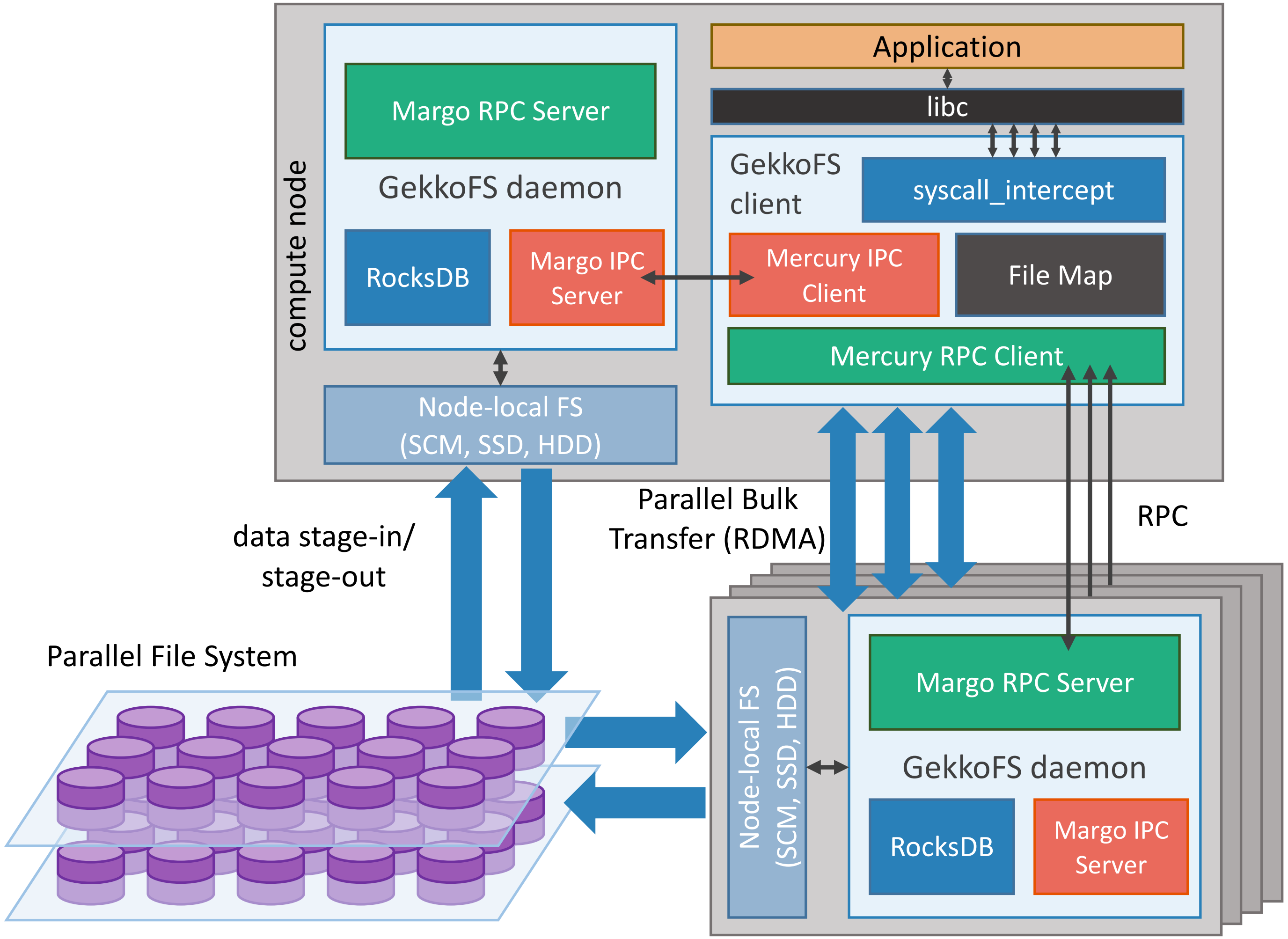
A high-performance RPC framework from ANL
https://mercury-hpc.github.io

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



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/

What GekkoFS is not

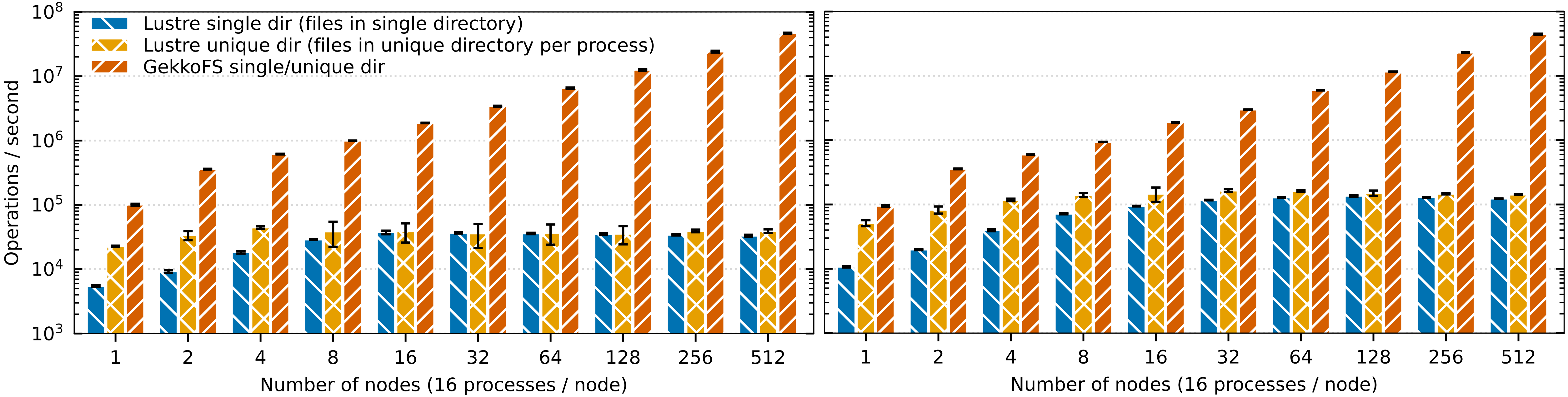
1. GekkoFS is not a long-term, general purpose PFS
   - **Ephemeral**: its lifetime is linked to an application/workflow
2. GekkoFS is not multi-user
   - Usable with **normal user privileges**
3. GekkoFS it not POSIX... mostly
   - GekkoFS **supports the POSIX I/O API** but discards some semantics in favor of performance
   - GekkoFS can also offer **specialized APIs**

What GekkoFS is

1. GekkoFS is a high-performance distributed file system for a single application
   - Allows **aggregating node-local storage** performance/capacity
   - Provides a **shared namespace** between nodes
2. GekkoFS is intended to be tuned for a specific application
   - Configurable **metadata management**: shared/non-shared, flat/hierarchical namespace, symlinks, access times updates, etc.
   - Configurable **data management**: data distribution, access consistency model, etc.
3. GekkoFS is easy to use
   - Runs in **user space** – easy installation and maintenance
4. GekkoFS is highly scalable
   - Performance of fully distributed mode **scales linearly** w.r.t. node count
   - Data based on chunks: Internal **access pattern transformation**
     - Shared file vs. file per process
     - Sequential vs. random

Evaluation on metadata performance

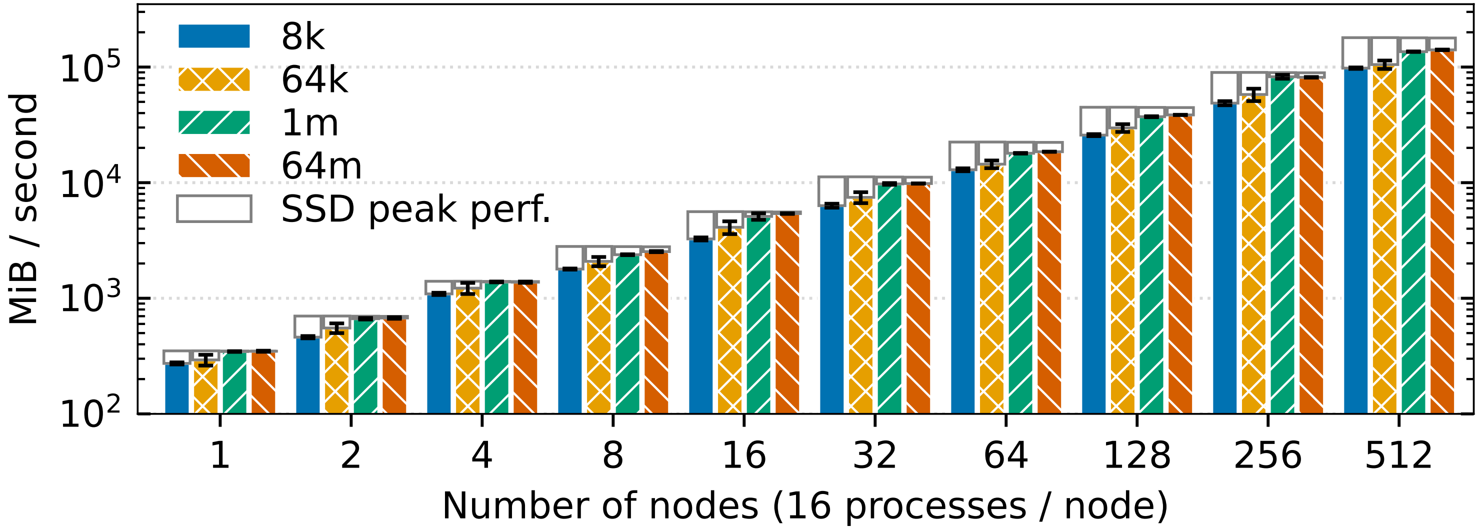
- GekkoFS weakly scaled (100K files per process)
- More than 819 million files in total at 512 nodes for GekkoFS



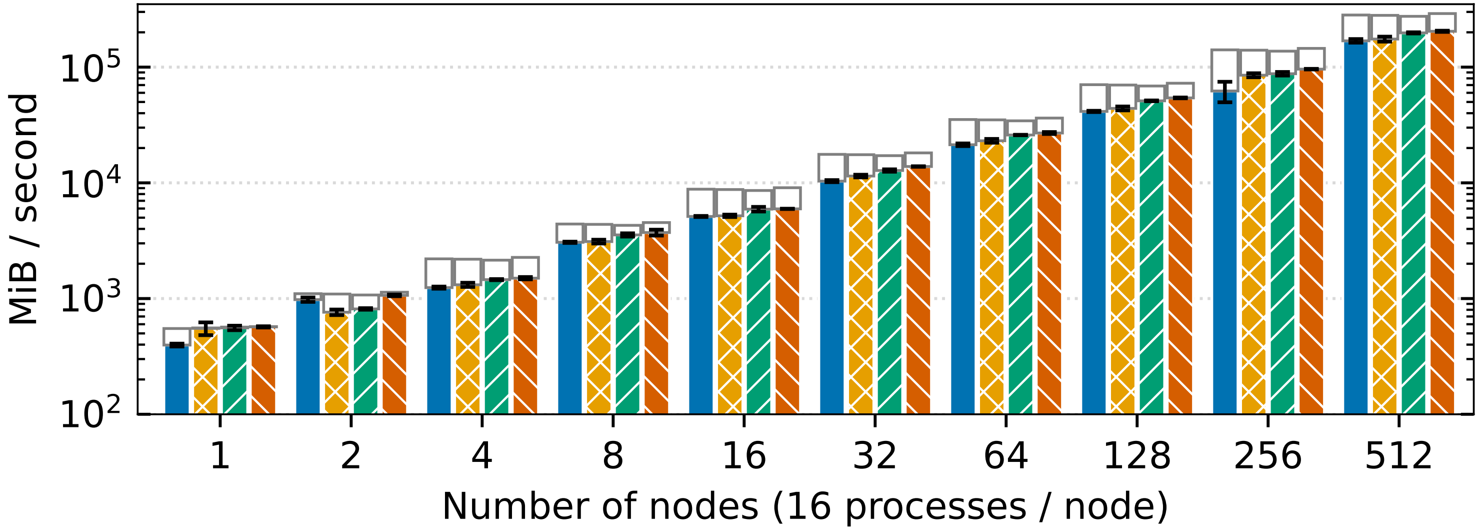
File create performance **File stat performance**

Evaluation on sequential data performance

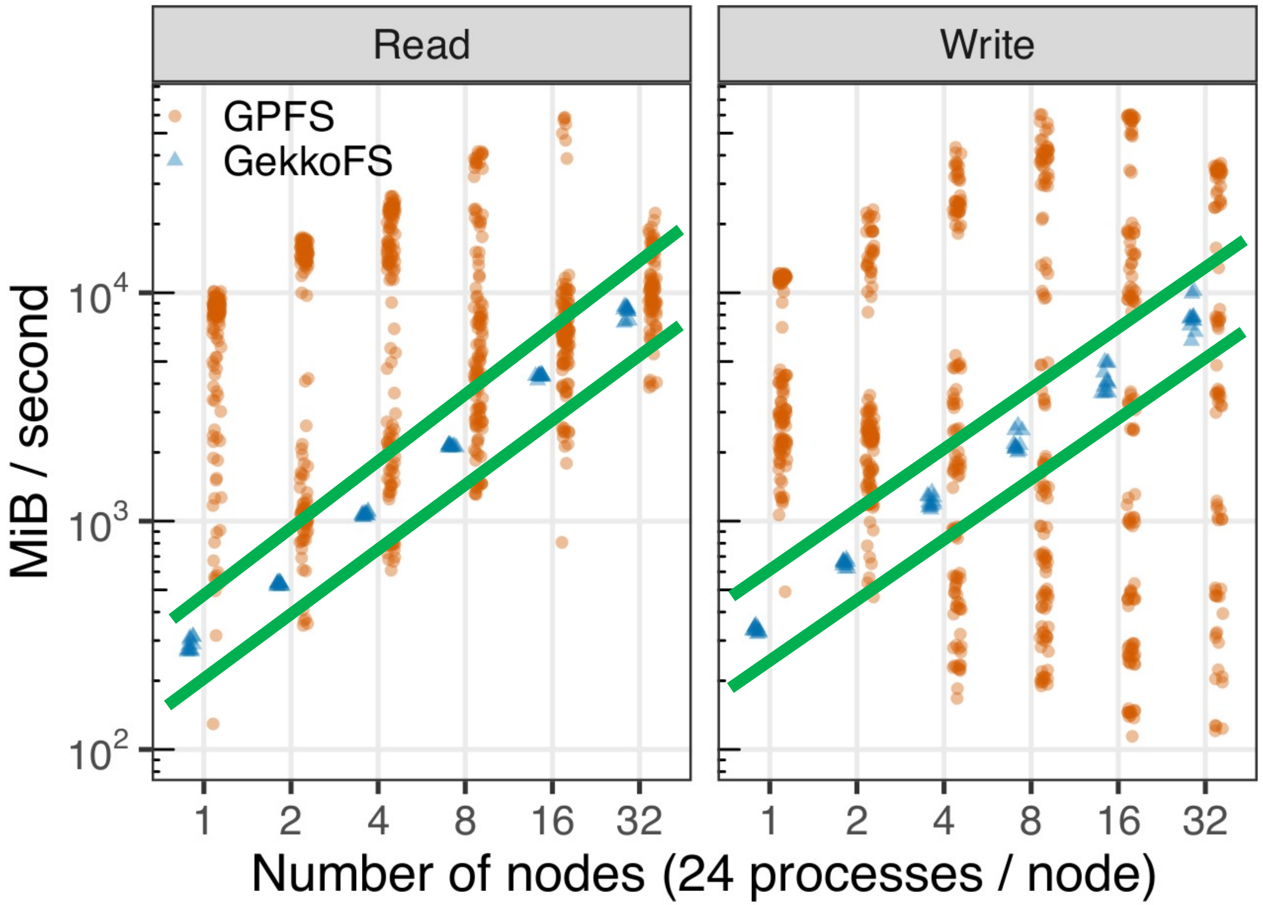
Write throughput



Read throughput



Performance variability revisited

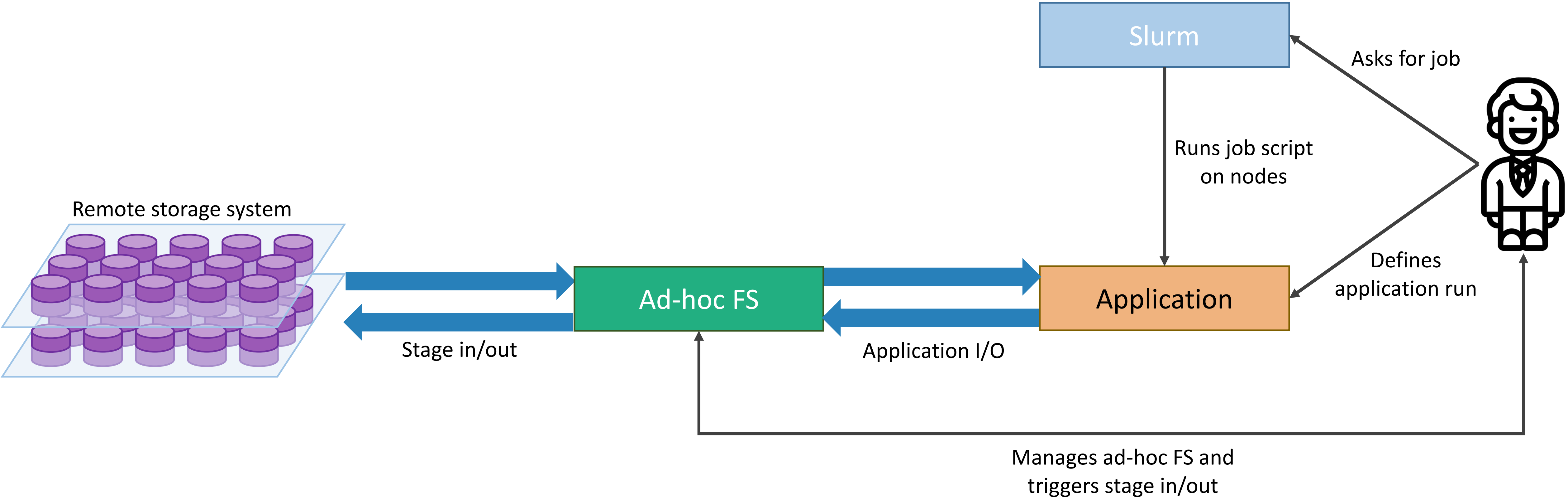


GPFS on MareNostrum 4 at the Barcelona Supercomputing Center

Reduced I/O variability for GekkoFS

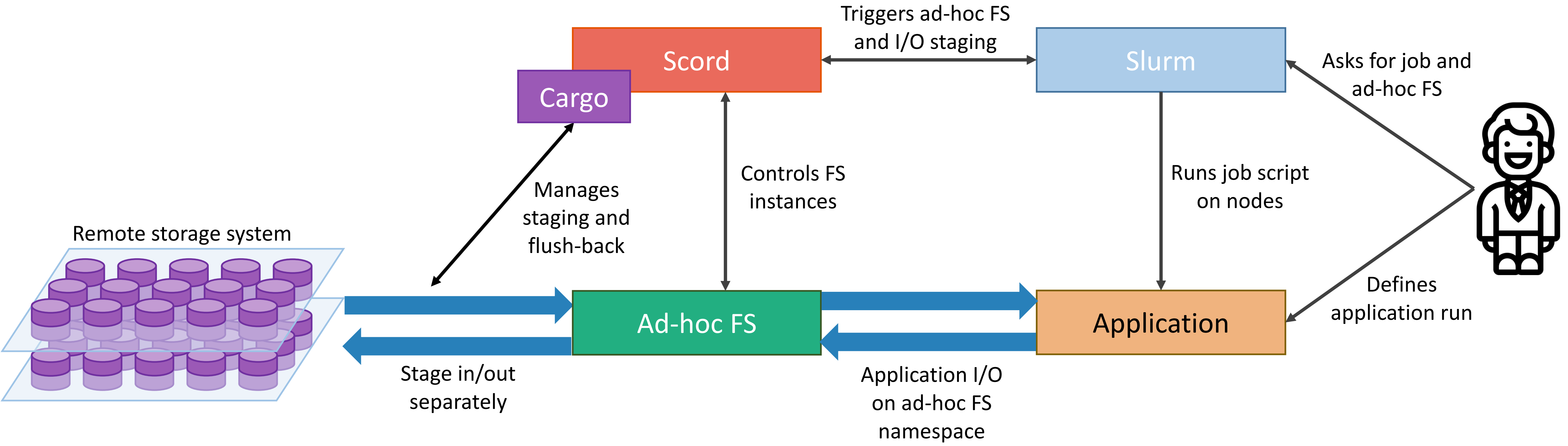
How application use ad-hoc FSs before ADMIRE?

- User must start ad-hoc FS and move data themselves



ADMIRE ad-hoc FS integration

- ADMIRE framework, i.e., Scord and Cargo, manages ad-hoc FSs and data transfer
- Scord (ad-hoc management) and Cargo (data transfer tool)



How does it work (outside of ADMIRE)?

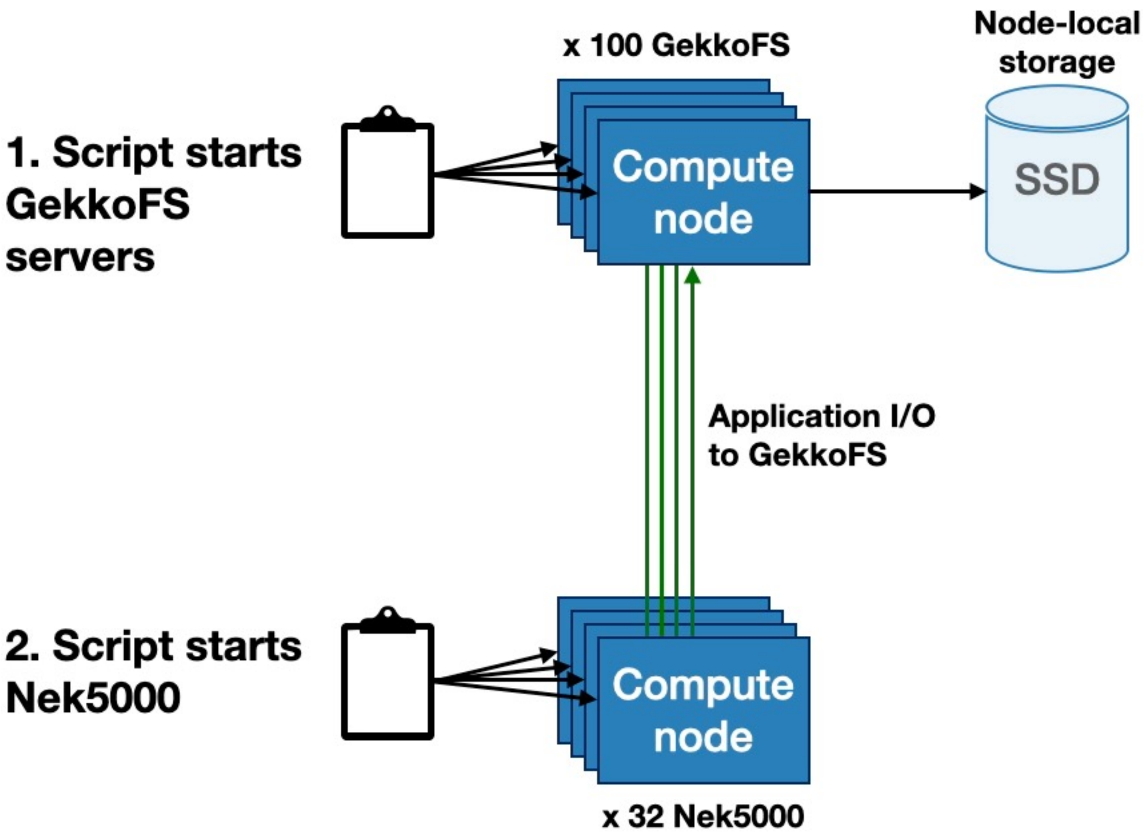
- Spack or manual installation (see Readme)
- Get the GekkoFS source code here
  - `git clone --recurse-submodules https://storage.bsc.es/gitlab/hpc/gekkofs.git`
- Start/stop the servers
  - `gkfs -c ~/gkfs.conf start/stop`
  - `gkfs_daemon -r <data_path> -m <gkfs_mount_path> -H <hostfile_path>` (manual)
- Set the hosts file on a path accessible to all clients (aka the file system instance)
  - `export LIBGKFS_HOSTS_FILE=<hostfile_path>`
- Use `LD_PRELOAD` to use the GekkoFS client
  - `LD_PRELOAD=<ipath>/libgkfs_intercept.so cp ~/some_input_data <gkfs_mount_path>/some_input_data`

All this is abstracted away in ADMIRE!

GekkoFS demo with Nek5000

Experimental setup

Nek5000 with GekkoFS



1. Script starts GekkoFS servers

x 100 GekkoFS

Compute node

Node-local storage

SSD

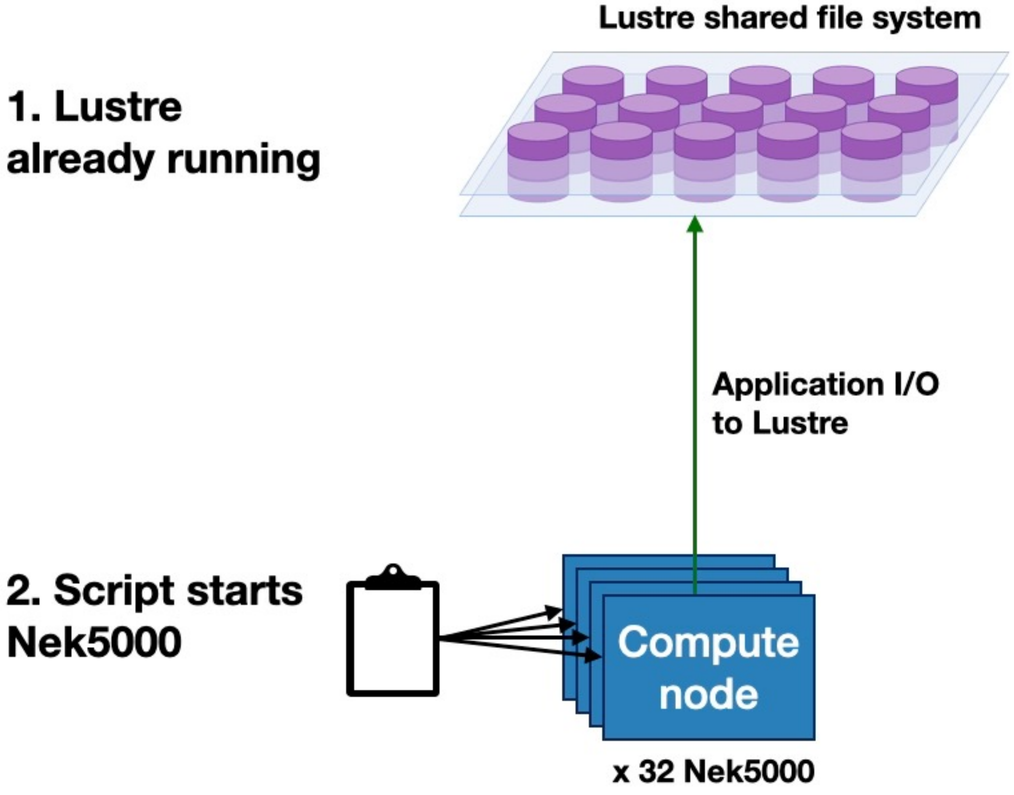
Application I/O to GekkoFS

2. Script starts Nek5000

Compute node

x 32 Nek5000

Nek5000 with Lustre



1. Lustre already running

Lustre shared file system

Application I/O to Lustre

2. Script starts Nek5000

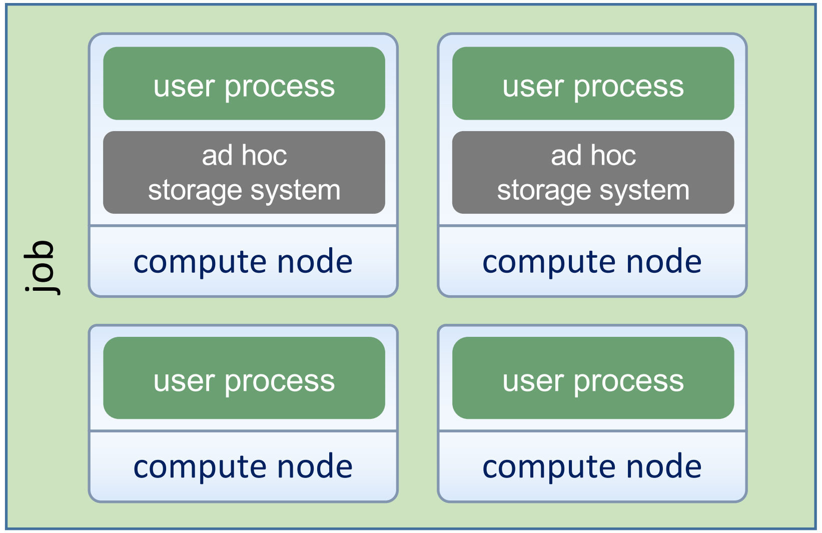
Compute node

x 32 Nek5000

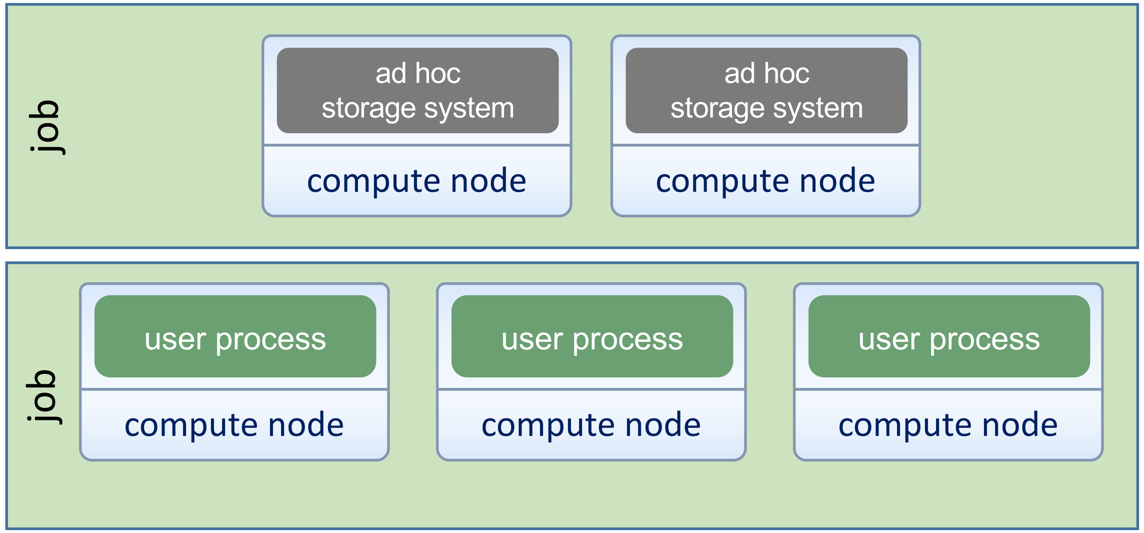
How to use ad-hoc file systems in ADMIRE

- New Slurm commands to ask for an ad-hoc file system in several options
- Ad-hoc storage options on lifetime and resource allocation

```
--ADM_adhoc_context= in_job:[shared|dedicated] | separate:[new|existing]
--ADM_adhoc_nodes= number_of_nodes
```



“in_job” → reuse allocation fully/partially



“separate” → separate allocation for I/O

How to use ad-hoc file systems in ADMIRE

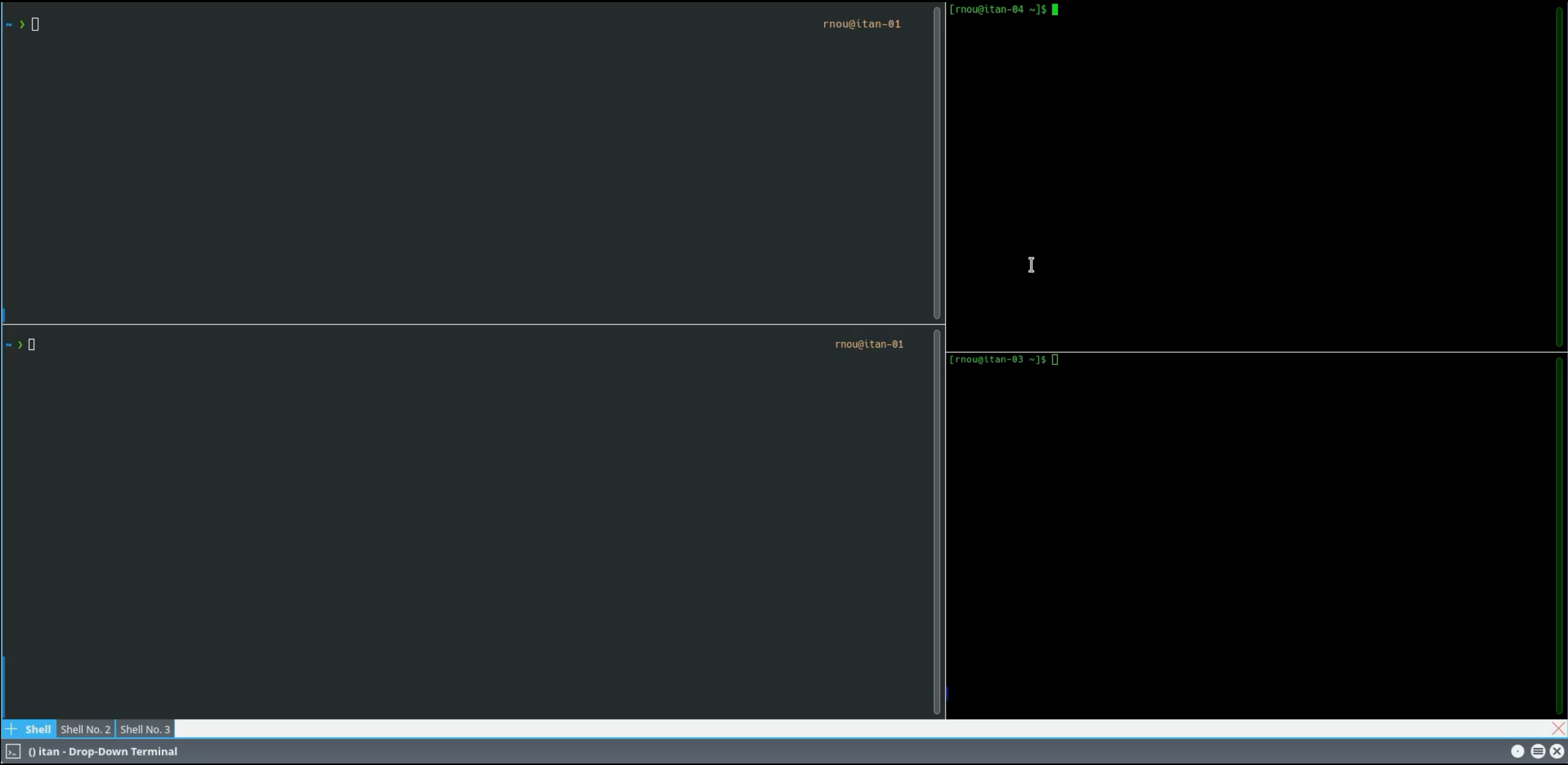
- New Slurm commands to ask for an ad-hoc file system in several options
- Specifying used/generated datasets and transfers between tiers

```
--ADM_input= “ORIGIN => TARGET”
--ADM_output= “ORIGIN => TARGET”
--ADM_inout= “ORIGIN <=> TARGET”
```

- Examples

```
$ sbatch --ADM_input=“lustre:/some/dir => gekkofs:/some/other/dir”
$ sbatch --ADM_output=“gekkofs:/some/dir => lustre:/some/other/dir”
$ sbatch --ADM_inout=“lustre:/some/dir <=> gekkofs:/some/other/dir”
```

Transparent ad-hoc file systems management in ADMIRE



```
~ ❯                                                   rnou@itan-01
```

```
~ ❯                                                   rnou@itan-01
```

```
[rnou@itan-04 ~]$
```

```
[rnou@itan-03 ~]$
```

Thank you!

Marc-André Vef – JGU Mainz

vef@uni-mainz.de

admire_wp2@listserv.uc3m.es

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

JG|U