Supporting NVRAM storage with active systemware

Adrian Jackson, Iakovos Panourgias

(EPCC, The Unversity of Edinburgh)

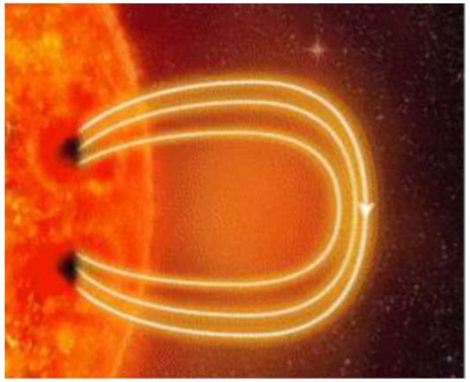
Ramon Nou, Alberto Miranda

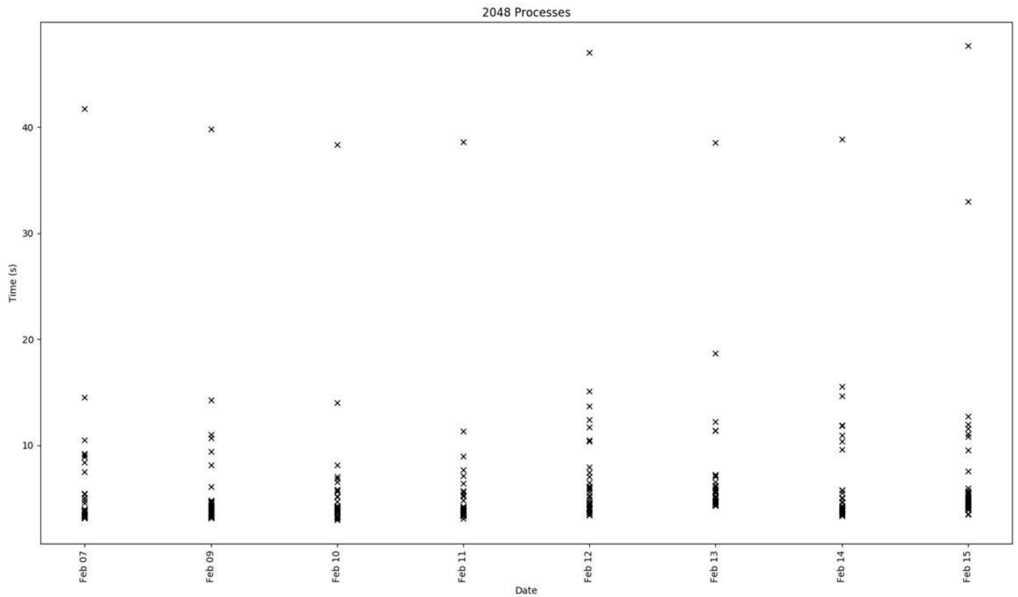
(BSC)

@adrianjhpc
a.jackson@epcc.ed.ac.uk
http://www.nextgenio.eu

I/O Performance – Small writes

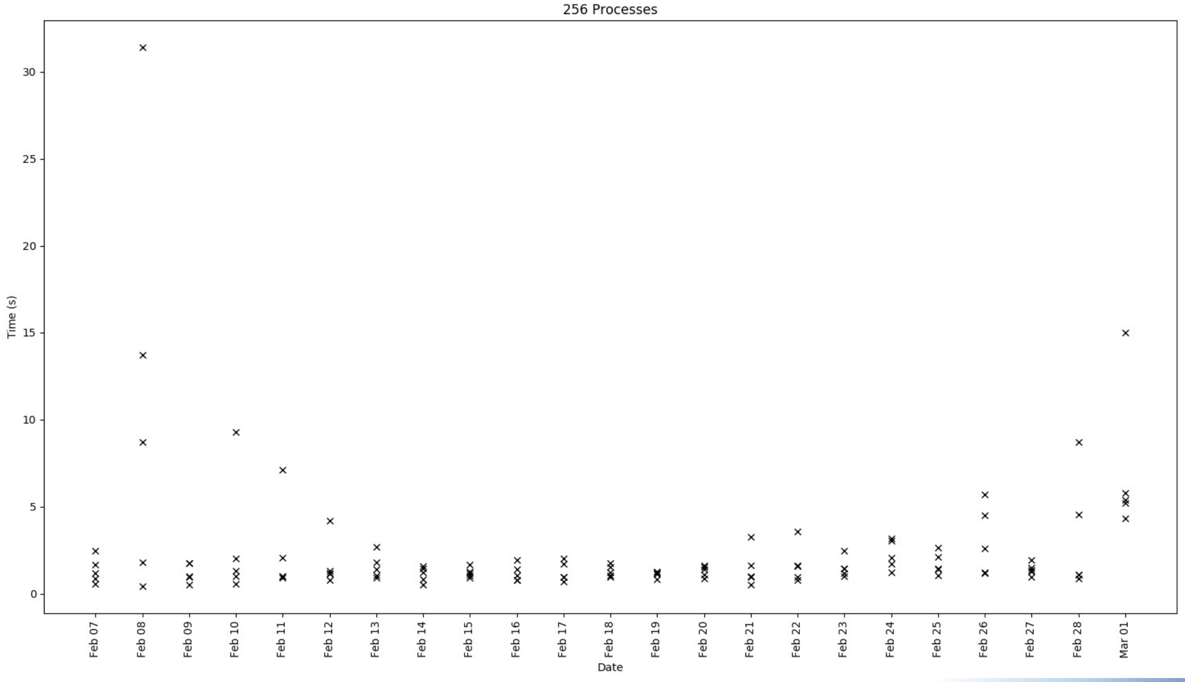
- Plot of average (across processes) run times of individual I/O regions for visualisation I/O
  - Same code executed for all runs
- I/O varies significantly in some cases:
  - Worst case ~12x
  - Best case ~2x



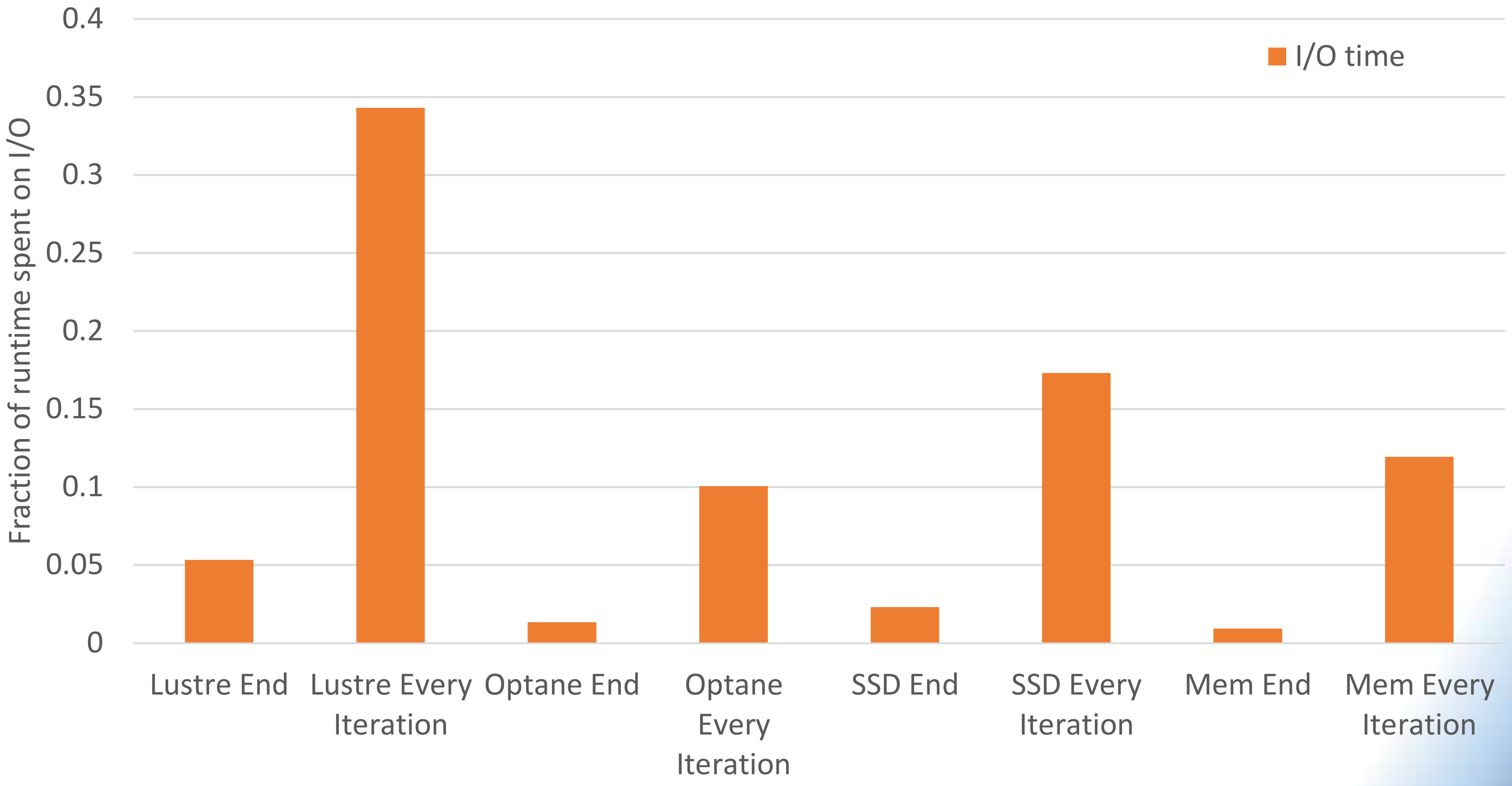


I/O Performance – Large writes

- Plot of run times of individual I/O regions for checkpoint I/O
  - Same code executed for all runs
- I/O varies in a similar pattern to the visualisation I/O
  - Variation more extreme (fastest is faster)
  - Average more consistent
- Checkpoint I/O less frequent but much quicker
  - Much higher data volumes



Enabling new I/O



Performance - STREAM

https://github.com/adrianjhpc/DistributedStream.git

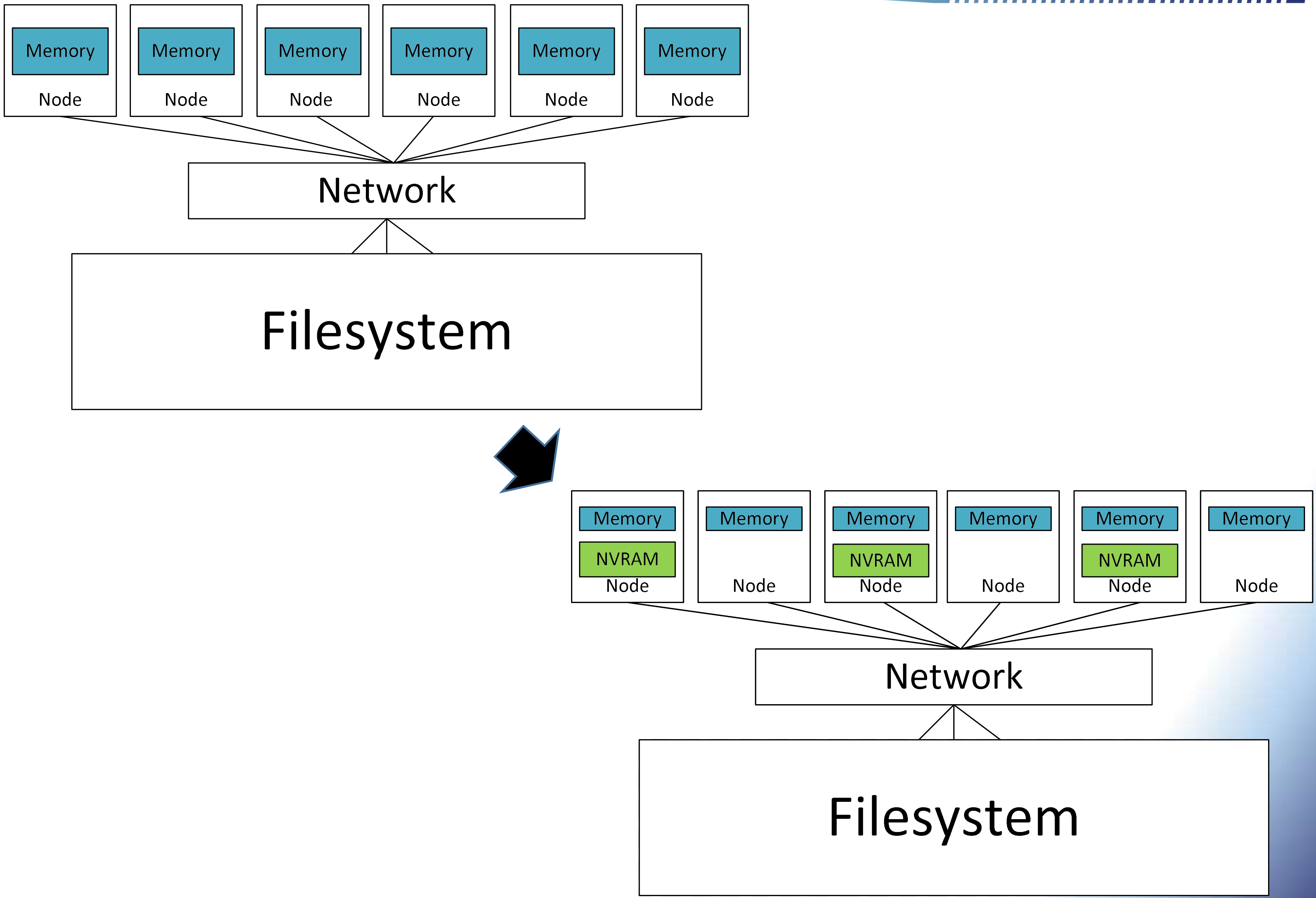
Mode	Min BW (GB/s)	Median BW (GB/s)	Max BW (GB/s)
App Direct (DRAM)	142	150	155
App Direct (DCPMM)	32	32	32
Memory mode	144	146	147
Memory mode	12	12	12

```
STREAM_TYPE     *a, *b, *c;
pmemaddr = pmem_map_file(path, array_length,
                         PMEM_FILE_CREATE|PMEM_FILE_EXCL,
                         0666, &mapped_len, &is_pmem)
a = pmemaddr;
b = pmemaddr + (*array_size+OFFSET)*BytesPerWord;
c = pmemaddr + (*array_size+OFFSET)*BytesPerWord*2;

#pragma omp parallel for
for (j=0; j<*array_size; j++){
   a[j] = b[j]+scalar*c[j];
}
pmem_persist(a, *array_size*BytesPerWord);
```

Optimising data usage

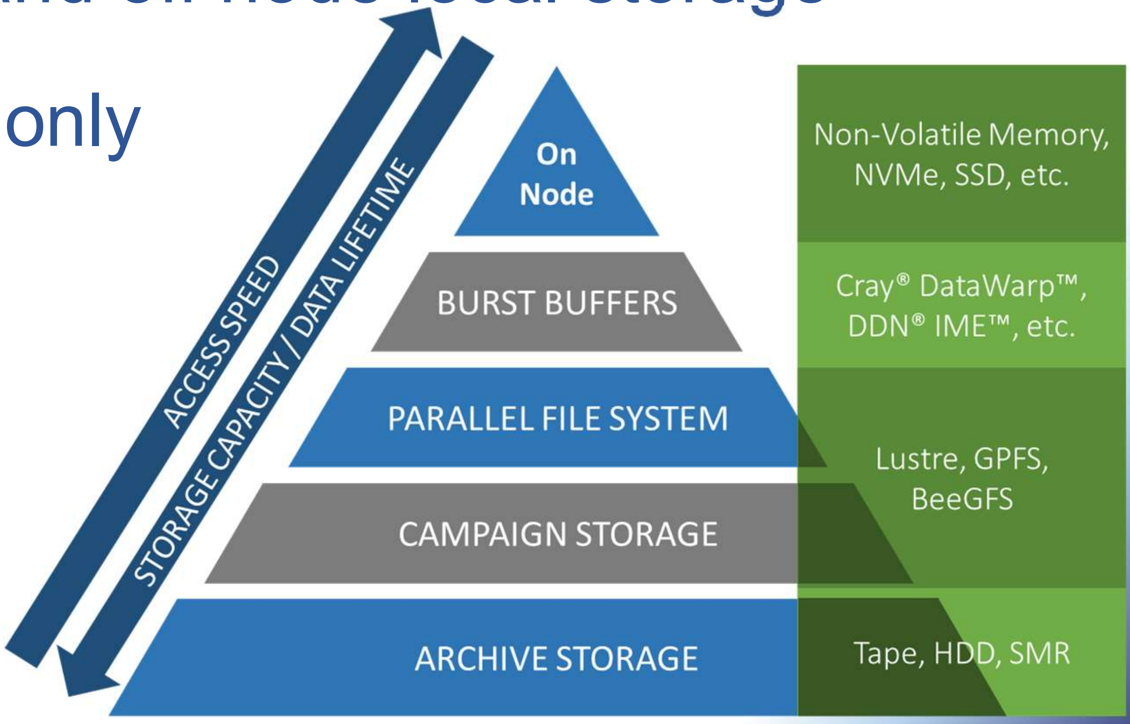
- Reducing data movement
  - Time and associated energy cost for moving data too and from external parallel filesystems
  - Move compute to data
- Considering full scientific workflow
  - Data pre-/post-processing
  - Multi-physics/multi-application simulations
  - Combined simulation and analytics
- Enable scaling I/O performance with compute nodes

Exploiting distributed storage

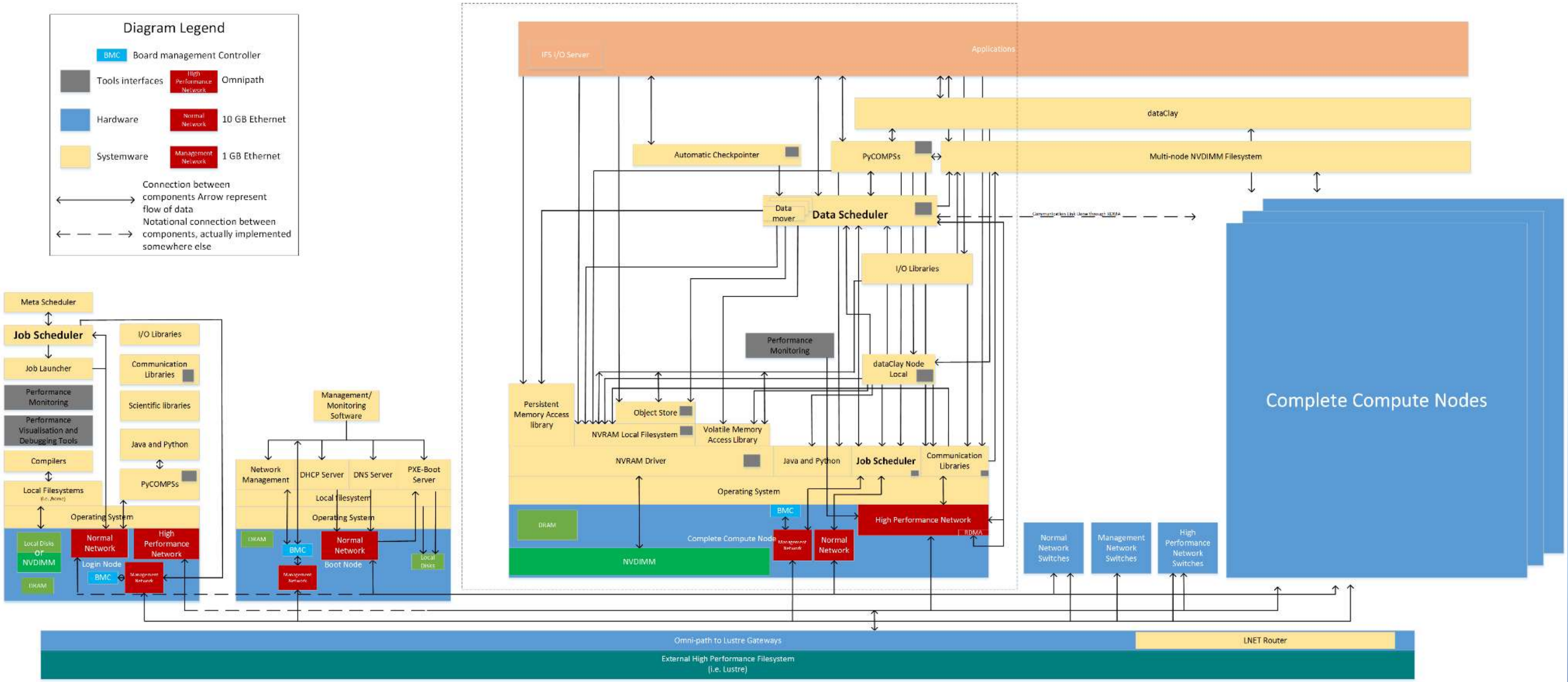


Challenges of compute local storage

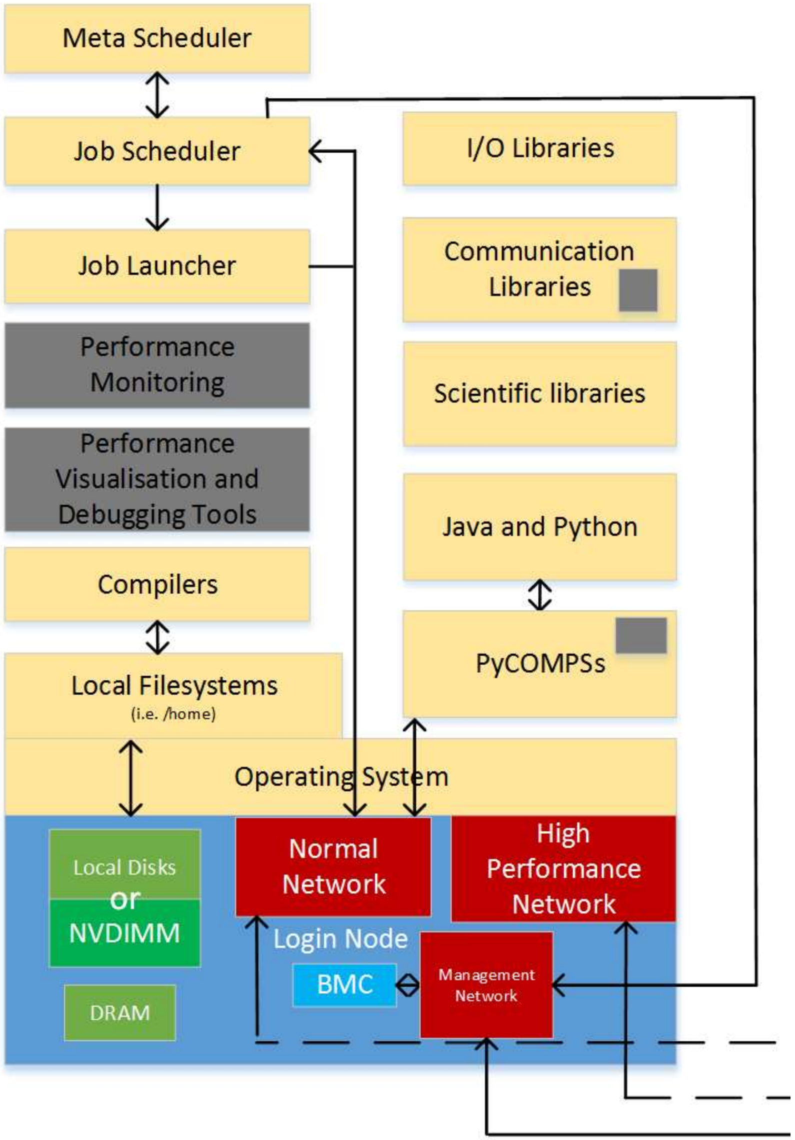
- No single namespace
- Enabling workflow jobs to run on same set of nodes
- Moving data on and off node local storage
- Ensuring data is only accessible by authorised users
- Understanding application performance



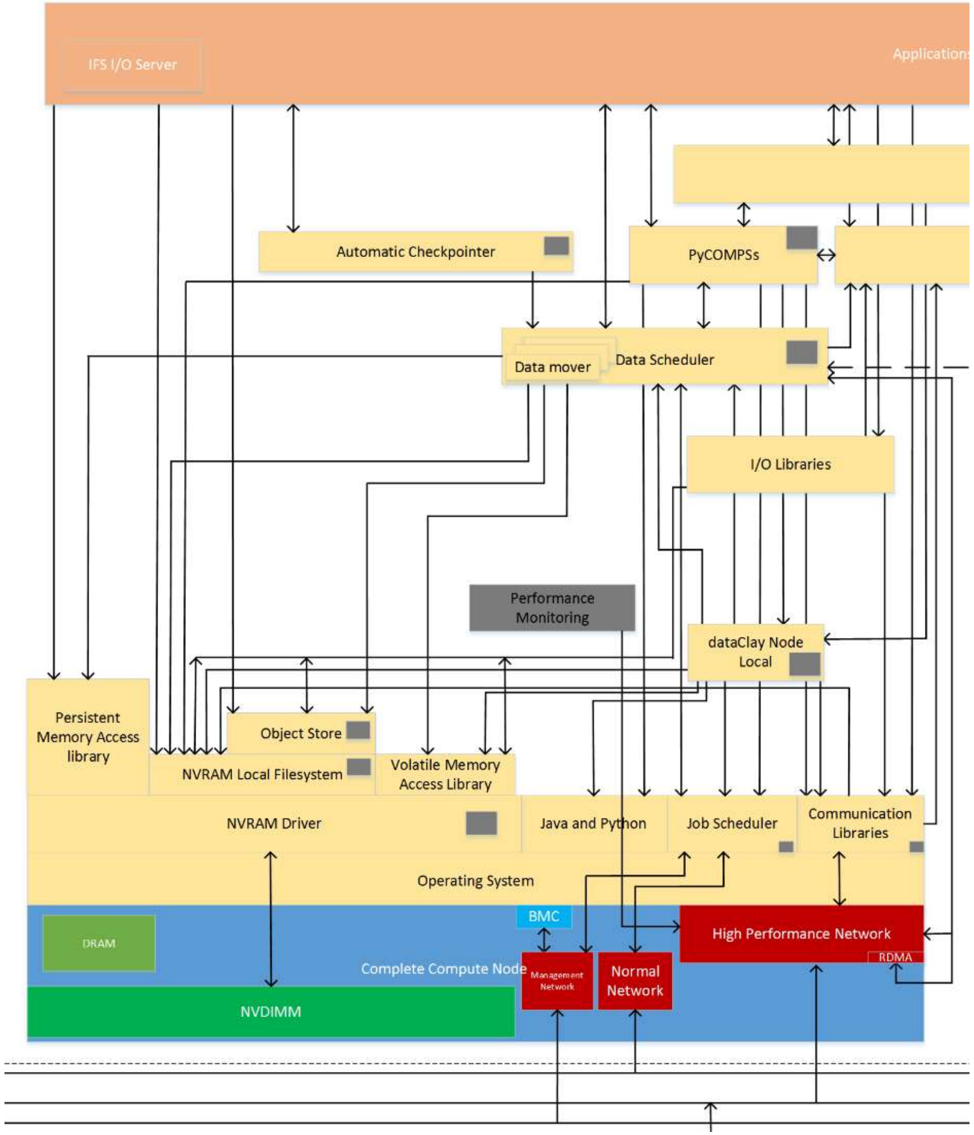
NEXTGenIO Systemware



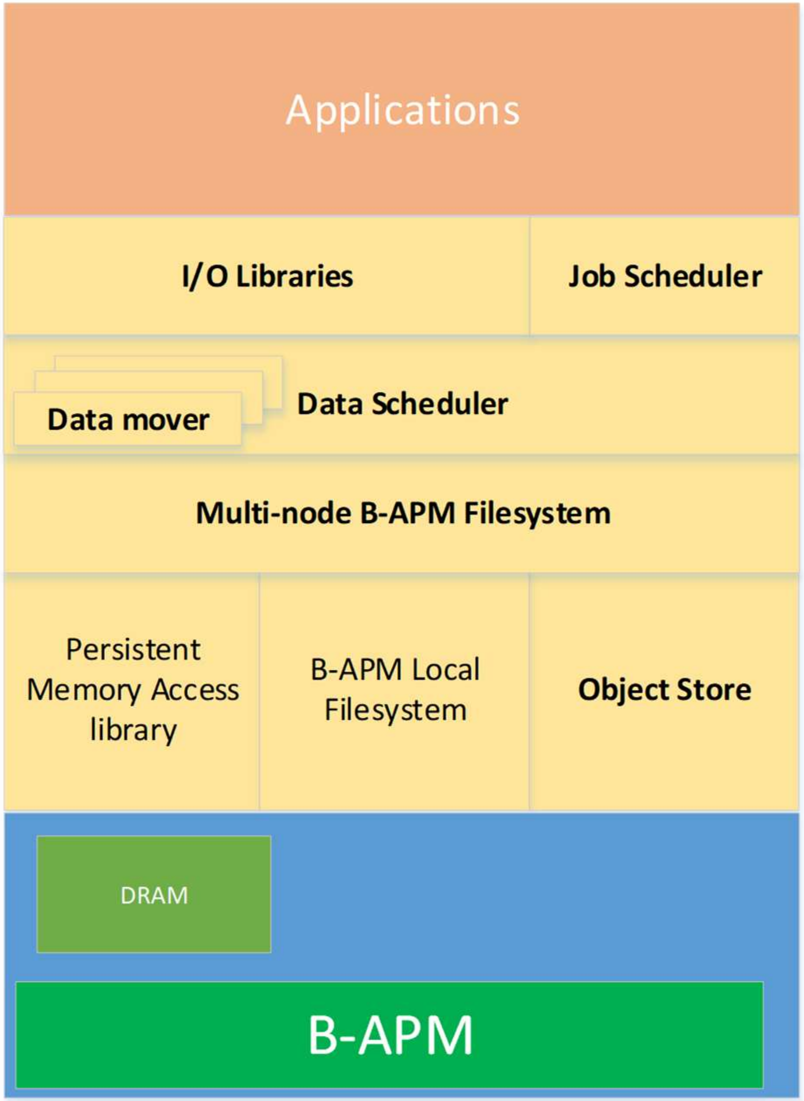
Login node systemware



Compute node systemware



Systemware architecture



norns

A data staging service and communication library to enable the scheduling of data-driven workflows in the multi-tiered HPC I/O stack

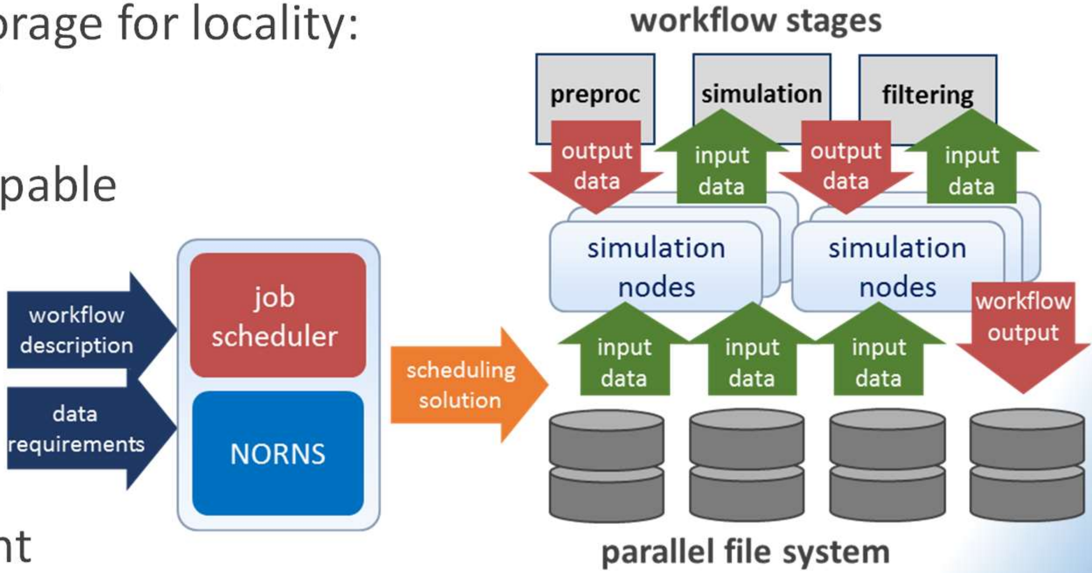
Integrates with SLURM to capture application data requirements

Leverages node-local storage for locality: SSDs, NVMe, NVRAM, …

Asynchronous RDMA-capable data transfers

Provides user-level APIs for data management

Provides admin-level APIs for tier management

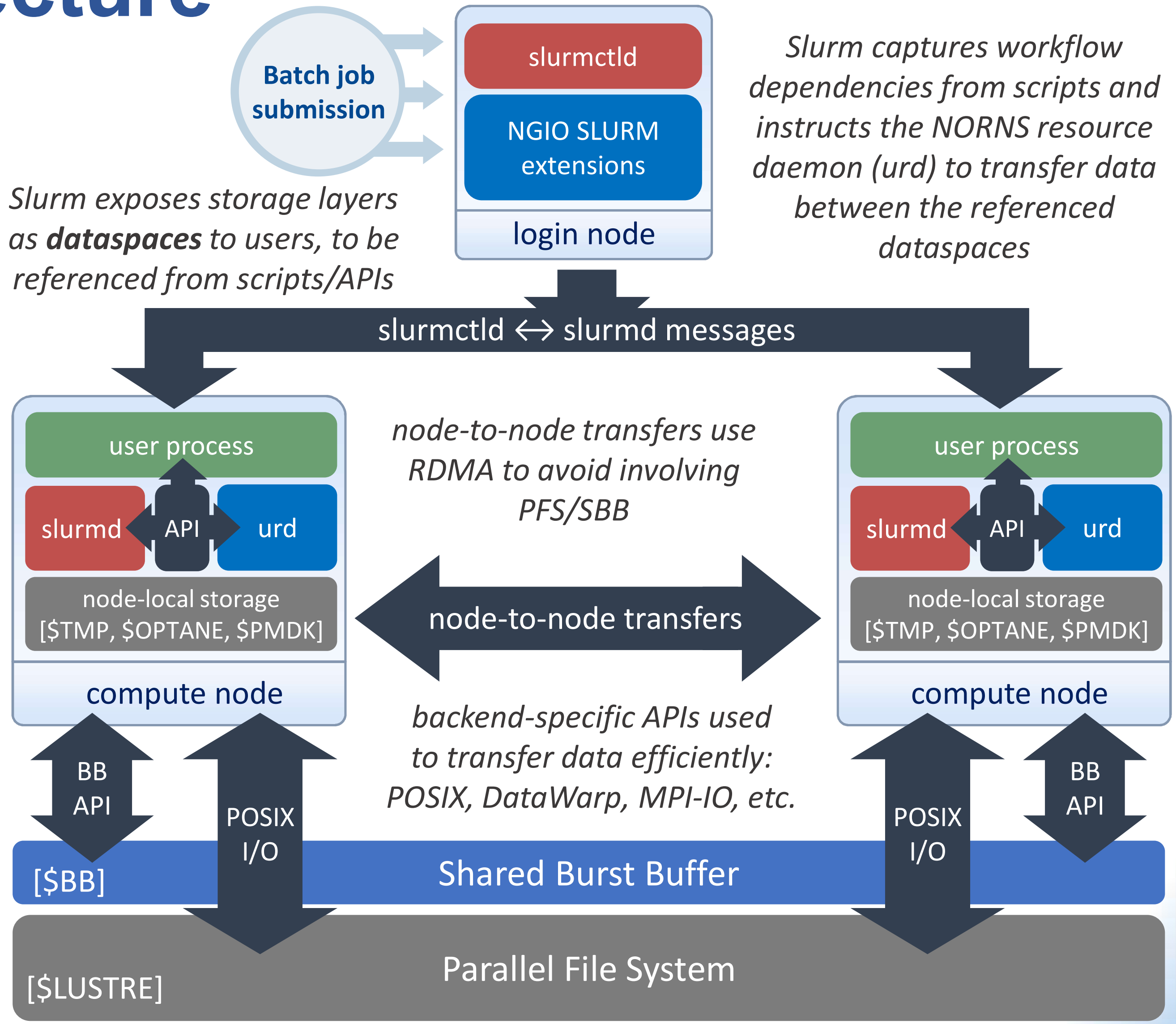


Barcelona Supercomputing Center
Centro Nacional de Supercomputación
epcc
nextgenio

Architecture

urd monitors storage layers and starts transfers as required

urd provides SLURM periodical reports of transfer status/ETAs to help with job scheduling decisions



Service components

- **SLURM extensions**
  - Capture job I/O requirements
  - Configure dataspaces required for job
  - Schedule, submits and tracks transfers through Admin API
  - Holds global view of data states associated with jobs and workflows
- **Resource control daemon**
  - Tracks dataspaces, storage tiers, jobs & processes
  - Accepts administrative requests, executes & monitors them
  - Accepts user requests, validates, executes & monitors them
  - Tracks request ETA & per-backend bandwidth
  - Can run as ‘root’ or as ‘norns’ user + capabilities

Service components

- **Administrative API (C)** [nornsctl.h, libnornsctl.so]
  - Management of dataspaces, storage tiers, jobs & processes
  - Tools for admin I/O task creation, monitoring & management
  - **Only for members of 'norns' group**

- **User-level API (C)** [norns.h, libnorns.so]
  - I/O task creation, monitoring & management
  - Dataspace queries

SLURM extensions

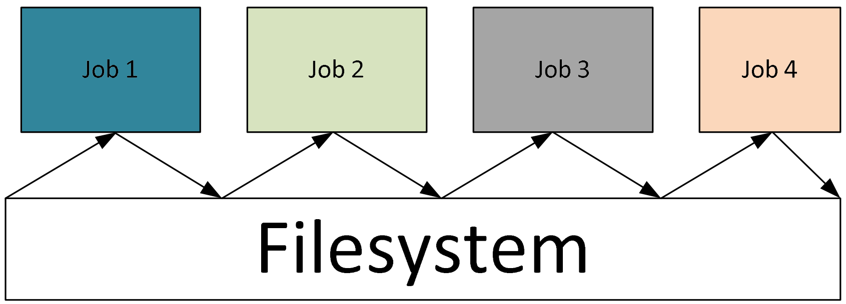
New options for srun, sbatch, salloc:

- SLURM tracks all workflow jobs; updating the prior- and post-dependencies and making sure they run in order
- If a workflow job fails; then all subsequent jobs fail (are deleted). Currently running jobs are not terminated

Option for job definition	Description
`#SBATCH --workflow-start`	Indicate that job starts a workflow
`#SBATCH --workflow-prior-dependency=JOBID+`	Make job depend on completion of prior jobs
`#SBATCH --workflow-end`	Indicate that job finalizes workflow

Using distributed storage

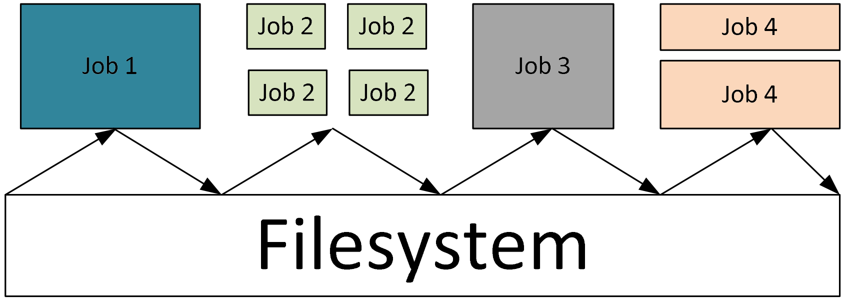
- New usage models
  - Resident data sets
    - Sharing preloaded data across a range of jobs
    - Data analytic workflows
    - How to control access/authorisation/security/etc….?
  - Workflows
    - Producer-consumer model



    - Remove filesystem from intermediate stages

Using distributed storage

- Workflows
  - How to enable different sized applications?



  - How to schedule these jobs fairly?
  - How to enable secure access?

SLURM extensions

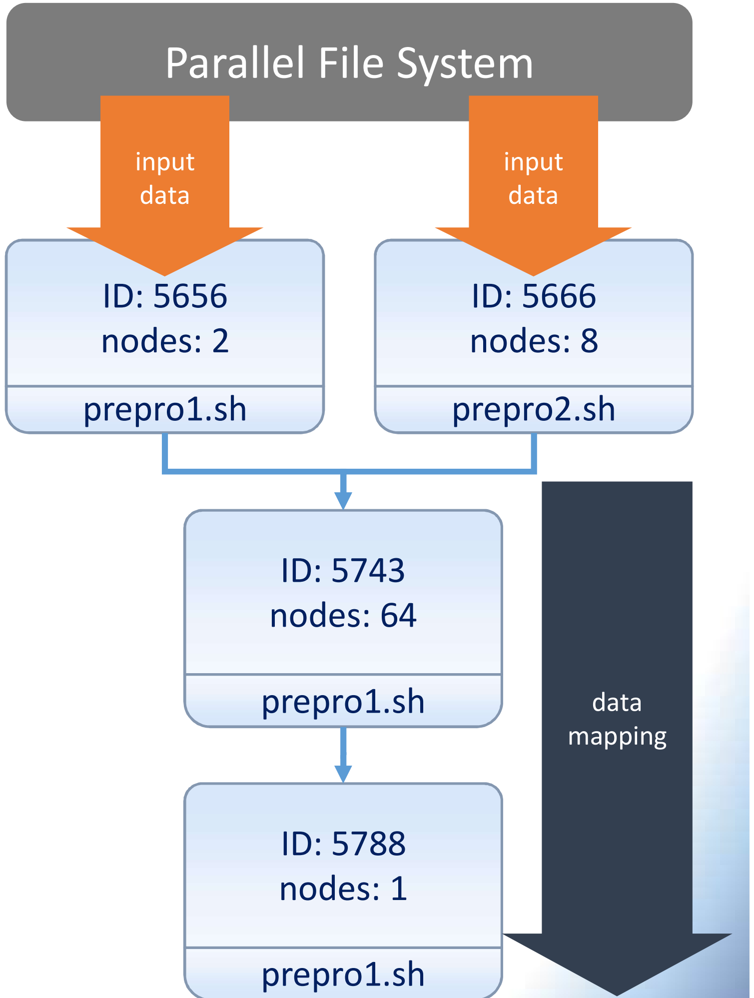
New options for data management:

- SLURM captures the dependencies and initiates the appropriate NORNS tasks to fulfill the transfers requested by users

Option for job definition	Description
`#NORNS stage_in origin destination mapping`	Stage in data from ORIGIN dataspace into DESTINATION according to a predefined MAPPING
`#NORNS stage_out origin destination mapping`	Stage out data from ORIGIN dataspace into DESTINATION according to a predefined MAPPING
`#NORNS persist [store|delete|share|unshare] location user`	Allow jobs to store data in node-local storage so that it can be shared among workflow jobs

Example: Definition of a workflow

```
[ bsc15455@mn1.bsc.es: ~ ] $
sbatch --nodes=2 \
       --workflow-start prepro1.sh

Job ID: 5656

[ bsc15455@mn1.bsc.es: ~ ] $
sbatch --nodes=8 \
       --workflow-start prepro2.sh

Job ID: 5666

[ bsc15455@mn1.bsc.es: ~ ] $
sbatch \
  --nodes=64 \
  --workflow-prior-dependency=5656,5666 \
  simulation.sh

Job ID: 5743

[ bsc15455@mn1.bsc.es: ~ ] $
sbatch --nodes=1 \
       --workflow-prior-dependency=5743 \
       --workflow-end postpro.sh

Job ID: 5788
```



Example: Resource mappings

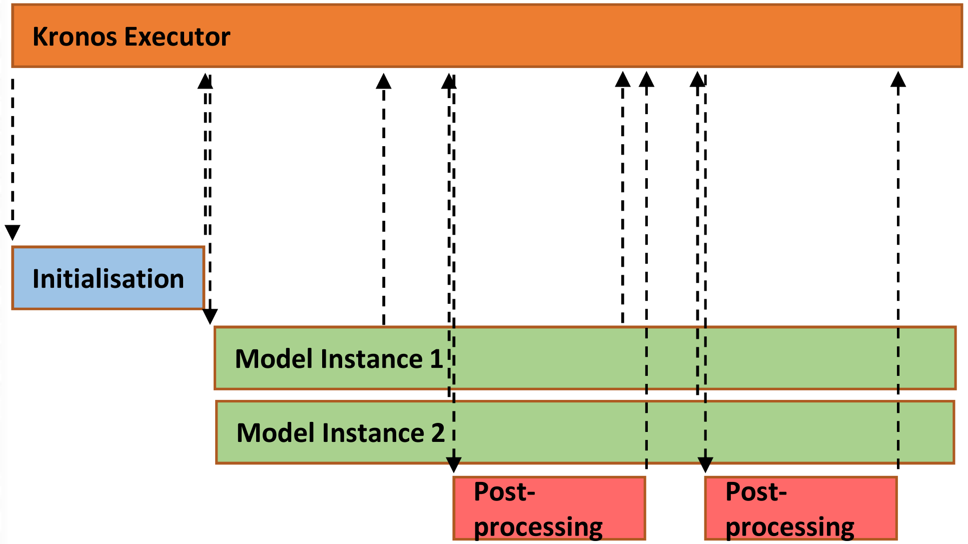
mapping.dat

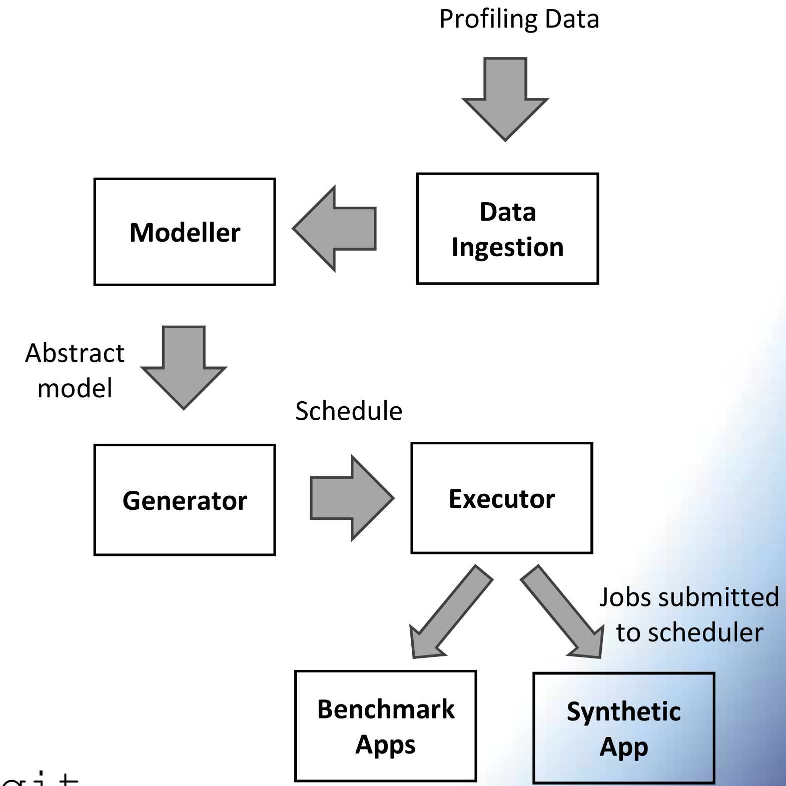
```
### INPUT/OUTPUT FILE MAPPINGS ###
['lustre://${HOME}/file/dir/checkpoint%[0-9]+%.out'];
0;pmdk0://;0,5
1;pmdk0://;1,6
2;pmdk0://;2,7
3;pmdk0://;3,8
4;pmdk0://;4,9,11
```

```
[ bsc15455@ngio-login1: ~ ] $ dsh -f nodes.list -c ls /mnt/pmdk0
ngio-cn00: checkpoint0.out  checkpoint5.out
ngio-cn01: checkpoint1.out  checkpoint6.out
ngio-cn02: checkpoint2.out  checkpoint7.out
ngio-cn03: checkpoint3.out  checkpoint8.out
ngio-cn04: checkpoint4.out  checkpoint9.out  checkpoint11.out
```

Simulators - Kronos

- Kronos is a workload modeller designed to enable exploration of software performance on different target systems

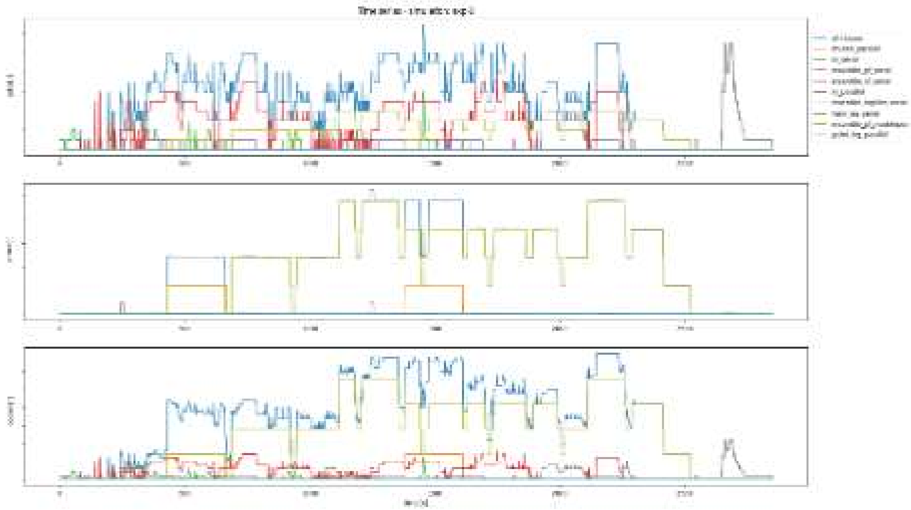




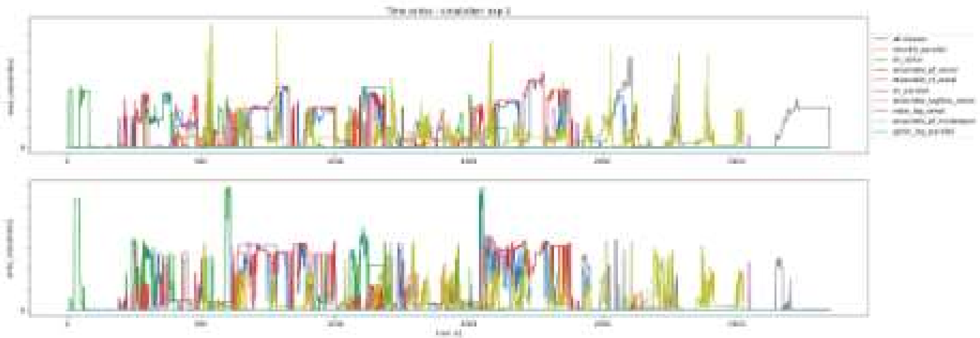
`https://github.com/ecmwf/kronos.git`

Simulators - Kronos

Post-processing



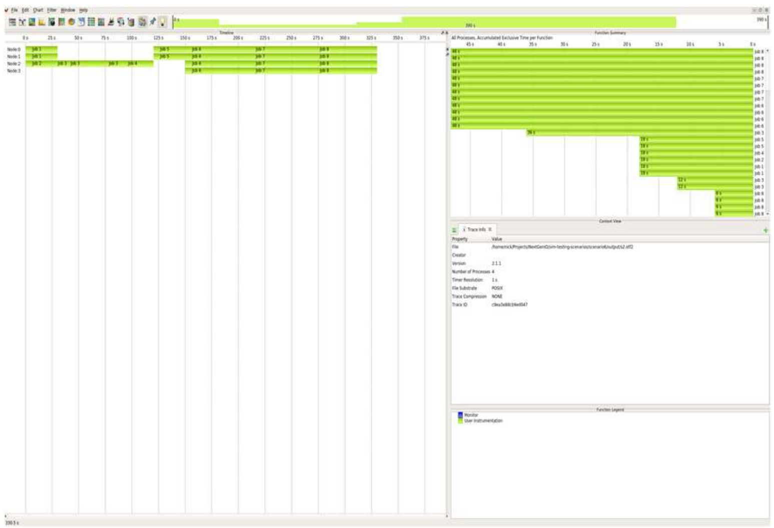
E.g. Workload execution profiles

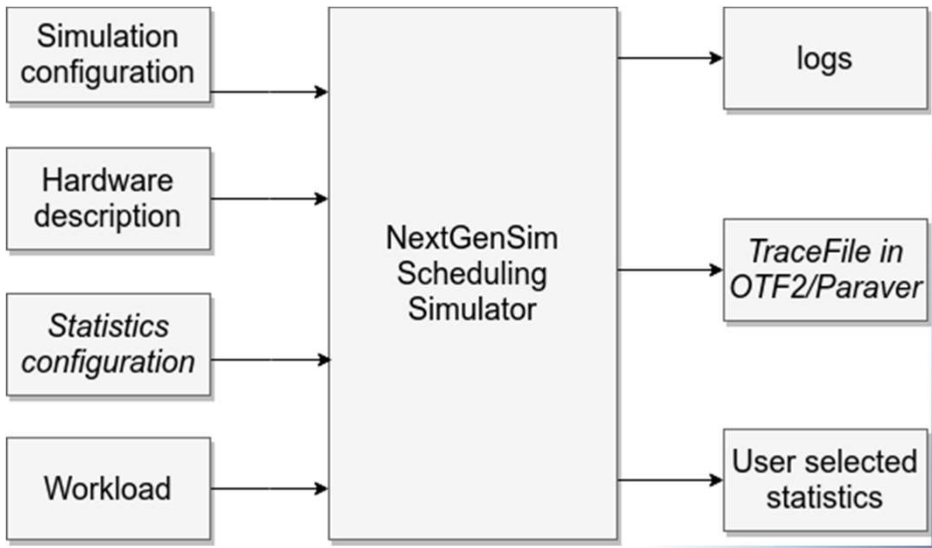


E.g. I/O time-profiles

Simulators – Scheduling

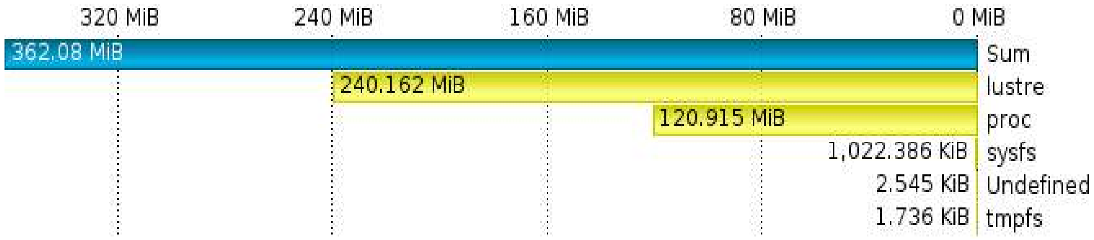
- NextGenSim: Simulating job scheduling with differing hardware and configurations
  - Explore performance benefits of utilising non-volatile memory in compute nodes

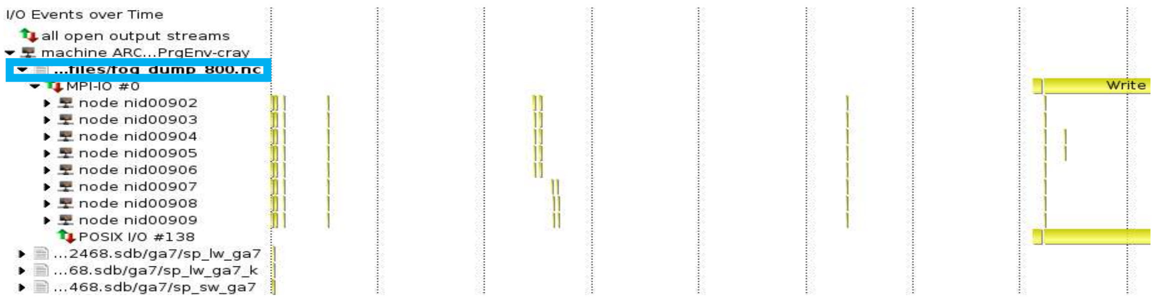


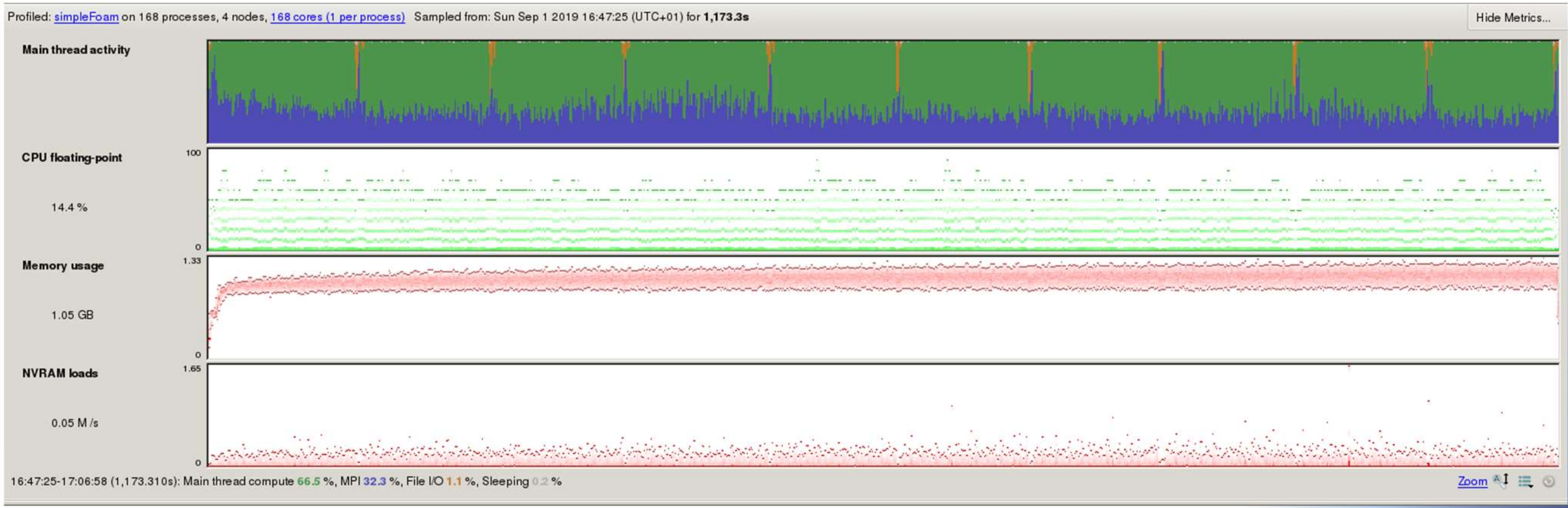


https://github.com/NGIOproject/NEXTGenSim.git

Performance and Debugging Tools

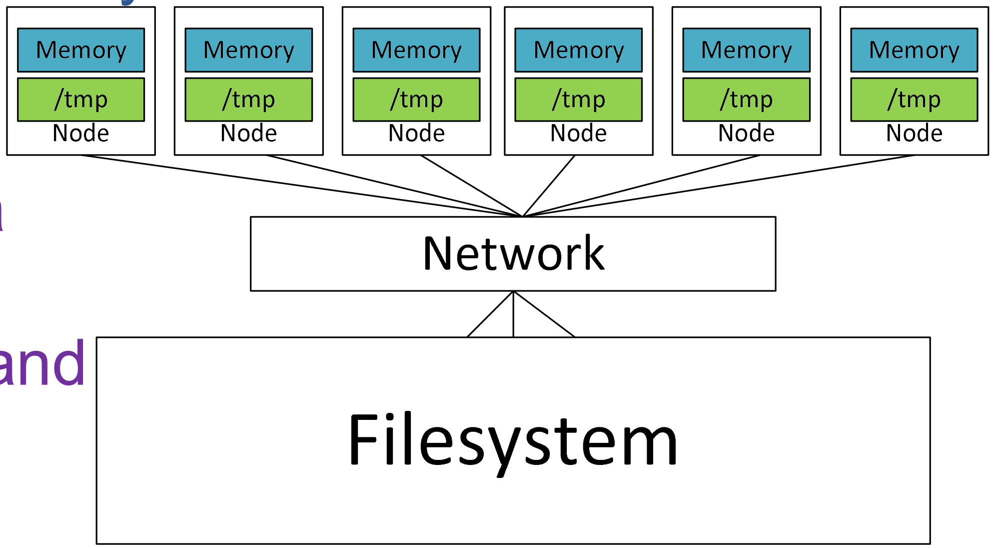






Using distributed storage

- Without changing applications
  - Large memory space/in-memory database etc…
  - Local filesystem

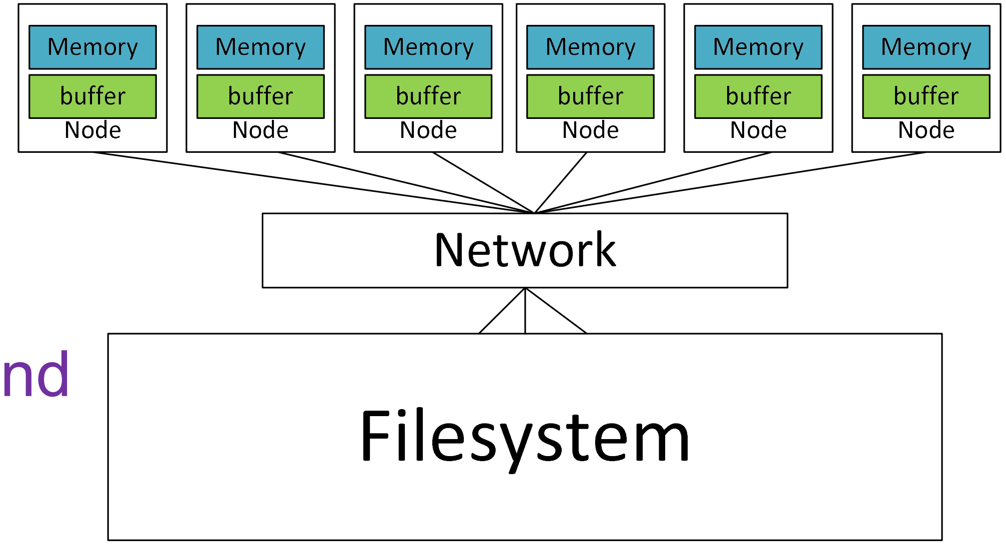


NGIO Data Scheduler (NORNS) and Slurm integration

- Users manage data themselves
- No global data access/namespace, large number of files
- Still require global filesystem for persistence

Using distributed storage

- Without changing applications
  - Filesystem buffer

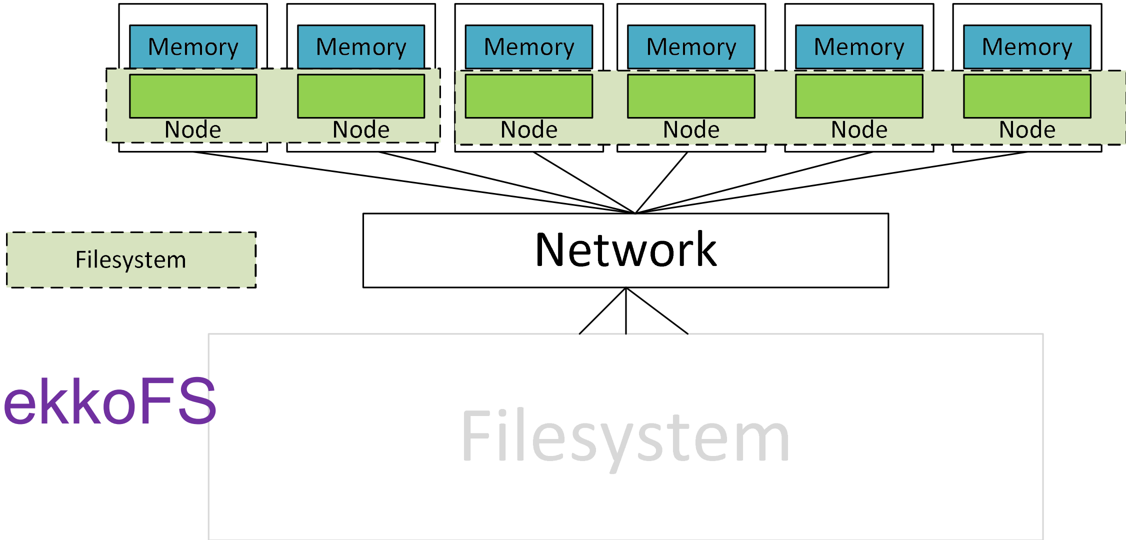


NGIO Data Scheduler (NORNS) and Slurm integration

- Pre-load data into NVRAM from filesystem
- Use NVRAM for I/O and write data back to filesystem at the end
- Requires systemware to preload and postmove data
- Uses filesystem as namespace manager

Using distributed storage

- Without changing applications
  - Global filesystem

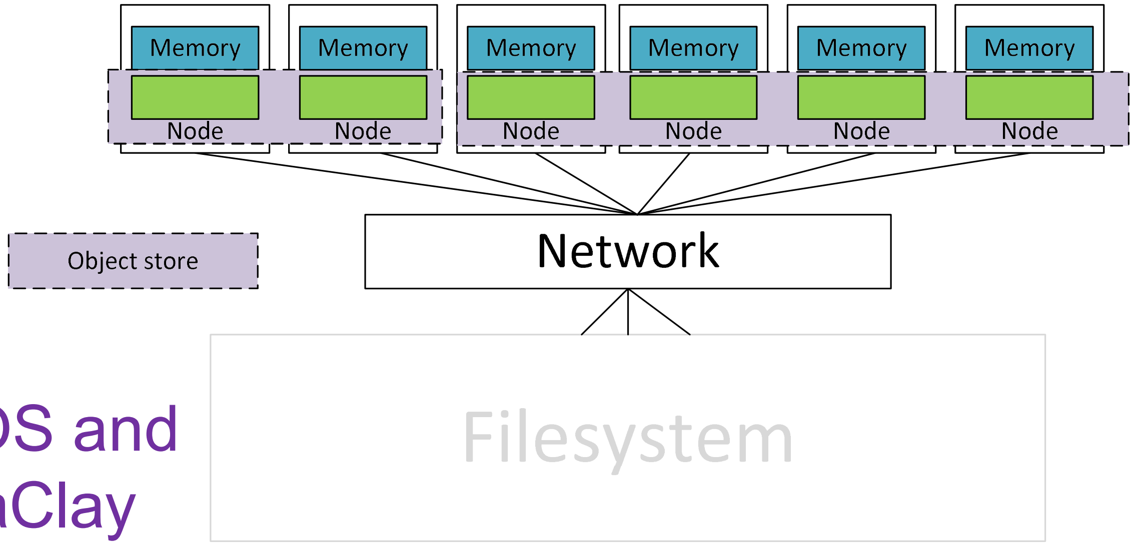


NGIO GekkoFS

- Requires functionality to create and tear down global filesystems for individual jobs
- Requires filesystem that works across nodes
- Requires functionality to preload and postmove filesystems
- Need to be able to support multiple filesystems across system

Using distributed storage

- With changes to applications
  - Object store



Intel DAOS and
BSC dataClay

- Needs same functionality as global filesystem
- Removes need for POSIX, or POSIX-like functionality

The challenge of distributed storage

- Enabling all the use cases in multi-user, multi-job environment is the real challenge
  - Heterogeneous scheduling mix
  - Different requirements on the B-APM
  - Scheduling across these resources
  - Enabling sharing of nodes
  - Not impacting on node compute performance
  - etc….
- Enabling applications to do more I/O
  - Large numbers of our applications don’t heavily use I/O at the moment
  - What can we enable if I/O is significantly cheaper

NGIO Prototype

- 34 node cluster with 3TB of Intel DCPMM per node
  - 2 CPUS per node, each with 1.5TB of DCPMM and 96GB of DRAM
- External Lustre filesystem

Performance – workflows

Synthetic workflow runtime (Lustre vs NVM)		
Component	Lustre	NVM
Producer	96 secs	64 secs
Consumer	74 secs	30 secs
Total	170 secs	94 secs

Sequential data producer/consumer
Working set: 100GiB data
2 configurations:
write/read to Lustre, separate nodes
write/read to NVM, same node

44.70% faster

High Performance Conjugate Gradient (HPCG) Benchmark
Profile: CPU and memory-bound
Targets: Single node

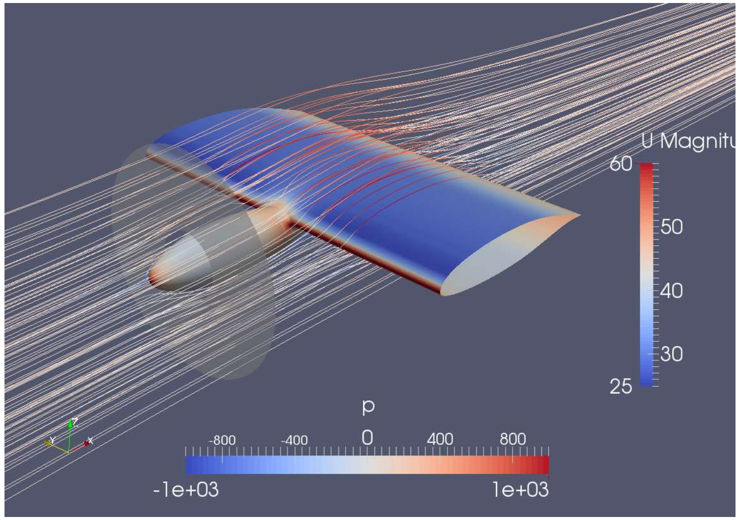
Performance impact on HPCG due to concurrent data staging	
Component	Runtime
HPCG (no staging)	122 secs
HPCG + stage in	142 secs
HPCG + stage out	137 secs

16.39% slower

12.29% slower

Performance – workflows

OpenFOAM simulation: *low-Reynolds number laminar turbulent transition modeling*
Input: mesh with ≈43M points
Stages: linear decomposition, parallel solver
768 MPI processes, 16 nodes
2 configurations:
① read/write to Lustre
② stage in, read/write on NVM, stage out



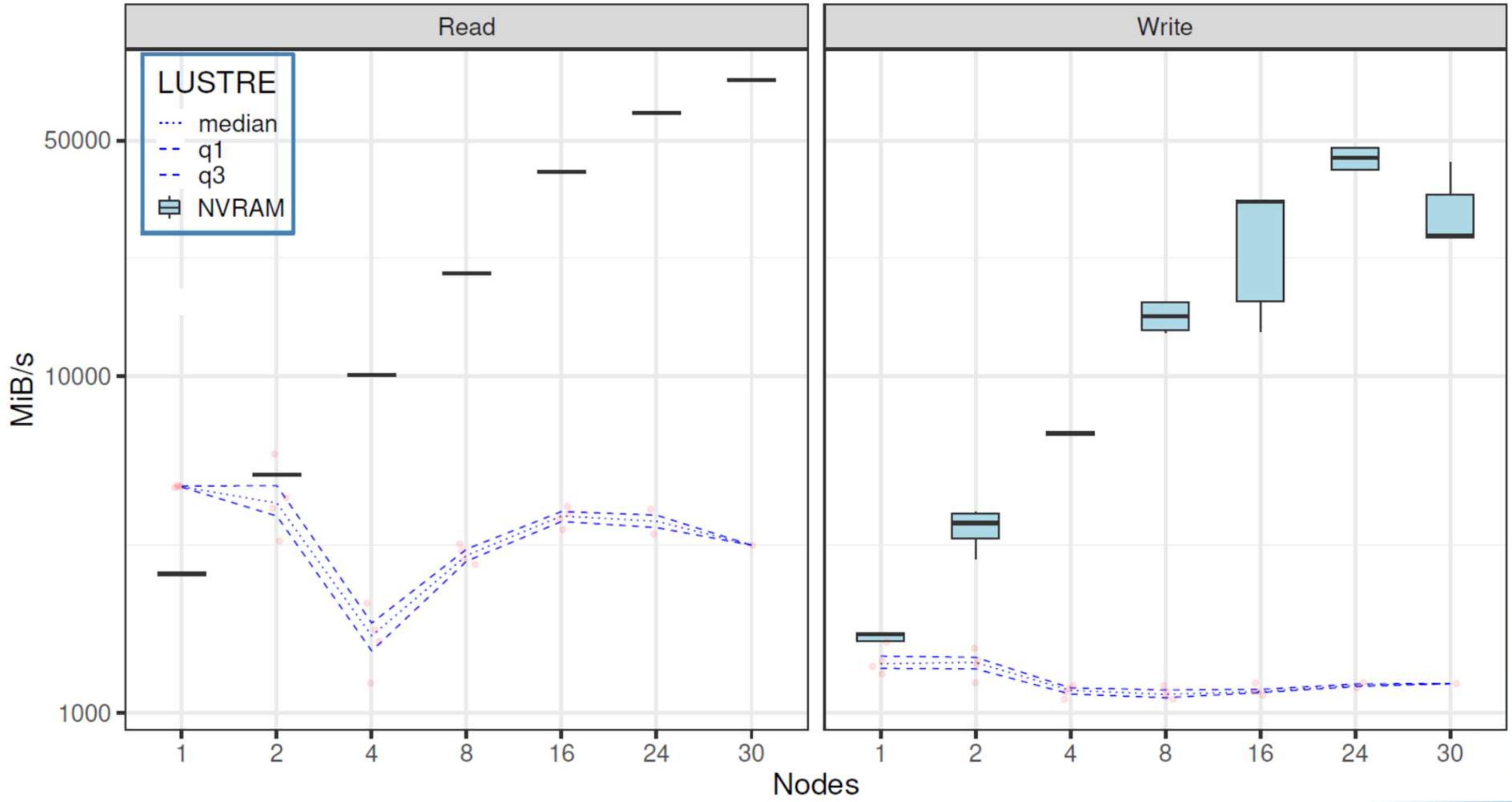
Performance benefits of data staging on OpenFOAM workflow

	16 nodes, 768 MPI procs			20 nodes, 960 MPI procs		
Stage	Lustre	NVM	Benefit	Lustre	NVM	Benefit
decomposition	1191 secs	1105 secs	–	1841 secs	1453 secs	–
data staging	–	32 secs	–	–	330 secs	–
solver	123 secs	66 secs	46% faster	664 secs	78 secs	88% faster
Total	1314 secs	1203 secs	8% faster	2505 secs	1861 secs	25% faster

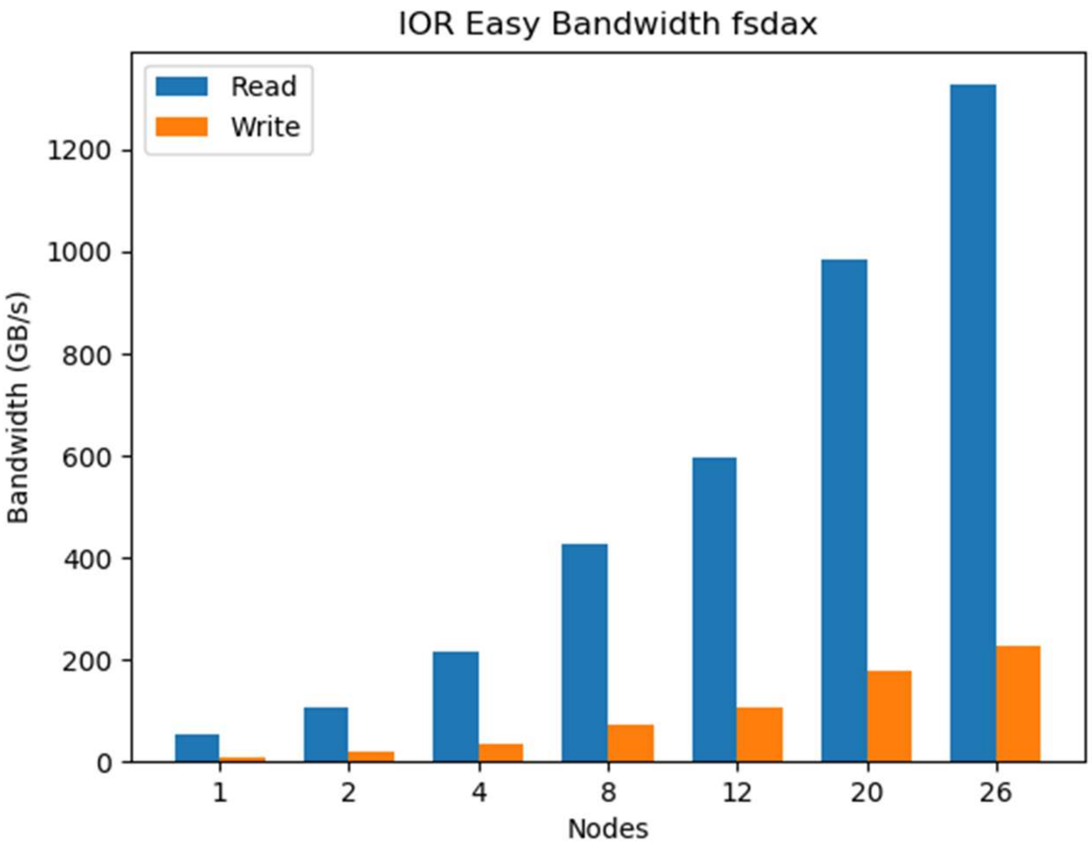
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance - IOR

- File per process



Performance – IOR Easy



Performance – IO-500

- 10 client nodes – 23 filesystem nodes GekkoFS
  - Client nodes are the same as the filesystem nodes
  - Currently using TCP/IP for communication
  - Only a single network rail

```
[RESULT] BW   phase 1            ior_easy_write 43.908 GB/s : time 342.83 seconds
[RESULT] BW   phase 2            ior_hard_write 4.449 GB/s : time 305.85 seconds
[RESULT] BW   phase 3             ior_easy_read 28.391 GB/s : time 530.21 seconds
[RESULT] BW   phase 4             ior_hard_read 21.788 GB/s : time  62.46 seconds
[RESULT-invalid] IOPS phase 1 mdtest_easy_write 1799.460 kiops : time 271.16 seconds
[RESULT] IOPS phase 2         mdtest_hard_write 140.924 kiops : time 325.54 seconds
[RESULT] IOPS phase 3                      find 606.030 kiops : time 866.72 seconds
[RESULT] IOPS phase 4          mdtest_easy_stat 1844.630 kiops : time 264.39 seconds
[RESULT] IOPS phase 5          mdtest_hard_stat 1773.000 kiops : time  29.92 seconds
[RESULT] IOPS phase 6        mdtest_easy_delete 904.720 kiops : time 549.28 seconds
[RESULT] IOPS phase 7          mdtest_hard_read 435.259 kiops : time 112.96 seconds
[RESULT] IOPS phase 8        mdtest_hard_delete 40.490 kiops : time 1122.59 seconds
One or more test phases invalid.  Not valid for IO-500 submission.
```

[SCORE-invalid] Bandwidth 18.6447 GB/s : IOPS 546.993 kiops : TOTAL 100.988

Performance IOR

- For comparison: Summit at ORNL

- **504** compute nodes, **2** MPI processes per node

[RESULT] BW phase 1	ior_easy_write	**2158.700** GB/s	time 362.34 seconds
[RESULT] BW phase 2	ior_hard_write	**0.572** GB/s	time 462.76 seconds
[RESULT] BW phase 3	ior_easy_read	1788.320 GB/s	time 437.39 seconds
[RESULT] BW phase 4	ior_hard_read	**27.403** GB/s	time 9.66 seconds
[RESULT] IOPS phase 1	mdtest_easy_write	3071.260 kiops	time 352.36 seconds
[RESULT] IOPS phase 2	mdtest_hard_write	24.375 kiops	time 327.20 seconds
[RESULT] IOPS phase 3	find	**21780.030** kiops	time 46.35 seconds
[RESULT] IOPS phase 4	mdtest_easy_stat	**28769.400** kiops	time 52.52 seconds
[RESULT] IOPS phase 5	mdtest_hard_stat	560.886 kiops	time 19.36 seconds
[RESULT] IOPS phase 6	mdtest_easy_delete	2699.000 kiops	time 398.34 seconds
[RESULT] IOPS phase 7	mdtest_hard_read	15354.700 kiops	time 17.02 seconds
[RESULT] IOPS phase 8	mdtest_hard_delete	26.504 kiops	time 181.79 seconds

[SCORE] Bandwidth **88.2049** GB/s : IOPS **1522.68** kiops : TOTAL **366.48**

OAK RIDGE National Laboratory

Open slide master to edit

Summary

- Enabling new technologies with HPC systems requires systemware support
- Transparently handling data for applications requires integration with job schedulers and data storage targets
- Tools are essential to allow exploitation of new hardware without requiring code change
- Tools very useful to evaluate design decision and approaches for applications and systems