www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

echofs: Enabling Transparent Access to Node-local NVM Burst Buffers for Legacy Applications

[…with the scheduler's collaboration]

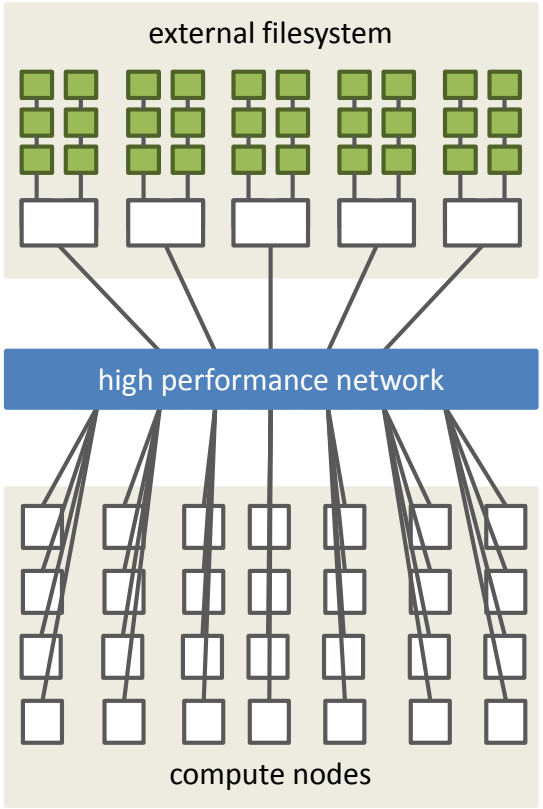
Alberto Miranda, PhD
Researcher on HPC I/O
alberto.miranda@bsc.es

The NEXTGenIO project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951

Dagstuhl, May 2017

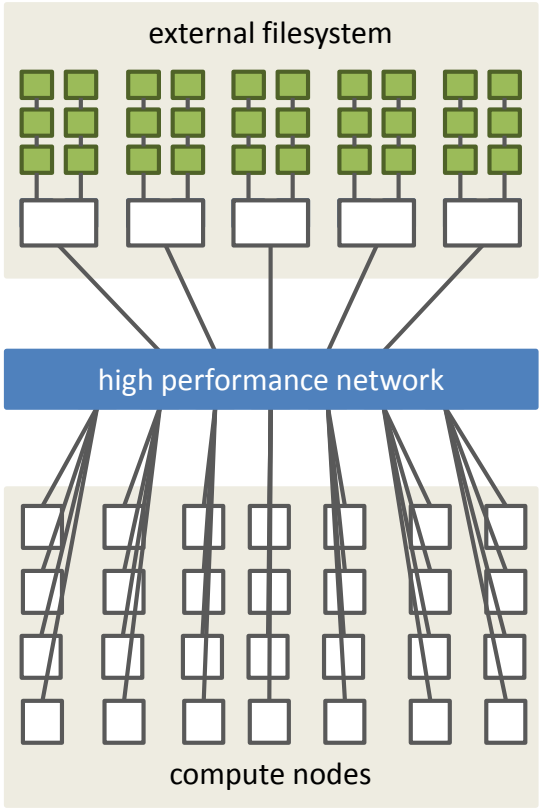
I/O -> a fundamental challenge;

- Petascale already struggles with I/O…
  - Extreme parallelism w/ millions of threads
  - Job's input data read from external PFS
  - Checkpoints: periodic writes to external PFS
  - Job's output data write to external PFS
- HPC & Data Intensive systems merging
  - Modelling, simulation and analytic workloads increasing…



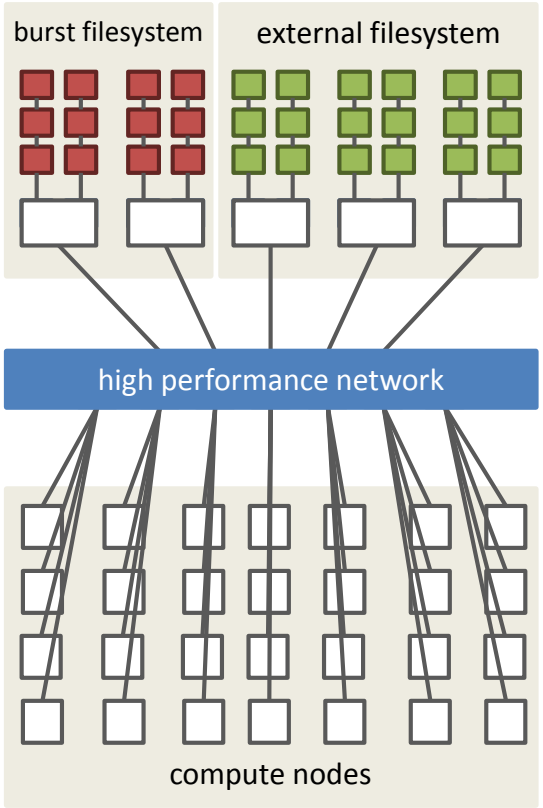
I/O -> a fundamental challenge;

- Petascale already struggles with I/O…
  - Extreme parallelism w/ millions of threads
  - Job's input data read from external PFS
  - Checkpoints: periodic writes to external PFS
  - Job's output data write to external PFS
- HPC & Data Intensive systems merging
  - Modelling, simulation and analytic workloads increasing…
- **And it will only get worse at Exascale…**



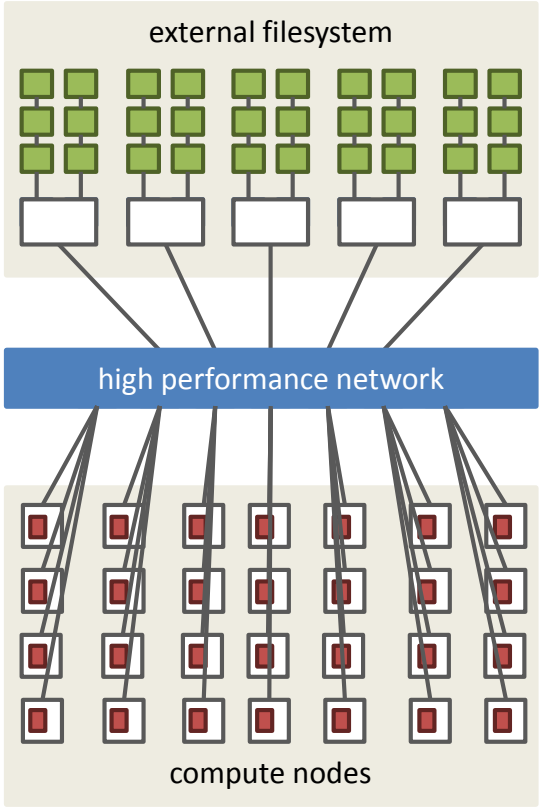
Burst Buffers -> remote;

- Fast storage devices that temporarily store application data before sending it to PFS
  - Goal: Absorb peak I/O to avoid overtaxing PFS
  - Cray Datawarp, DDN IME
- Growing interest to add them into next-gen HPC architectures
  - NERSC's Cori, LLNL's Sierra, ANL's Aurora, …
  - Typically a separate resource to PFS
  - Usage/allocation/data movements/etc become user responsibility



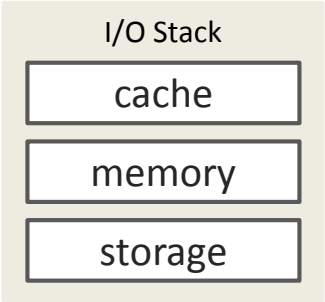
Burst Buffers -> on-node;

- Non-volatile coming to the node
  - Argonne's Theta has 128GB SSDs in each compute node



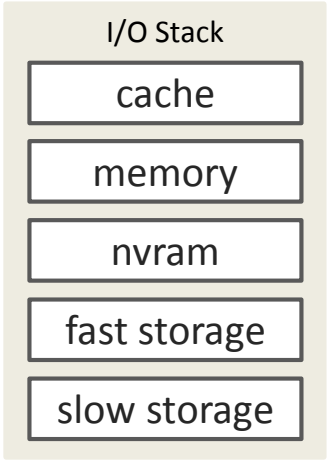
NEXTGenIO EU Project [http://www.nextgenio.eu]

- Node-local, high-density NVRAM becomes a fundamental component
  - Intel's 3DXPoint™
  - Capacity much larger than DRAM
  - Slightly slower than DRAM but significantly faster than SSDs
  - DIMM form factor → standard memory controller
  - No refresh → no/low energy leakage



NEXTGenIO EU Project [http://www.nextgenio.eu]

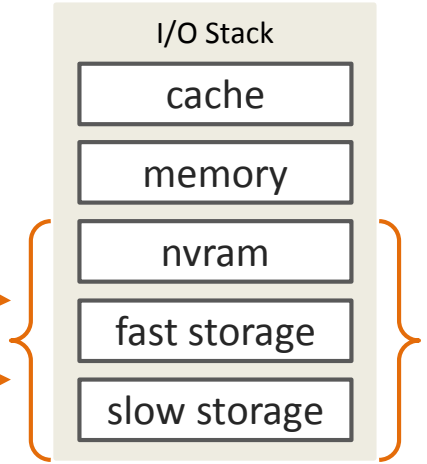
- Node-local, high-density NVRAM becomes a fundamental component
  - Intel's 3DXPoint™
  - Capacity much larger than DRAM
  - Slightly slower than DRAM but significantly faster than SSDs
  - DIMM form factor → standard memory controller
  - No refresh → no/low energy leakage



NEXTGenIO EU Project [http://www.nextgenio.eu]

- Node-local, high-density NVRAM becomes a fundamental component
  - Intel's 3DXPoint™
  - Capacity much larger than DRAM
  - Slightly slower than DRAM but significantly faster than SSDs
  - DIMM form factor → standard memory controller
  - No refresh → no/low energy leakage

1. How do we manage access to these layers?

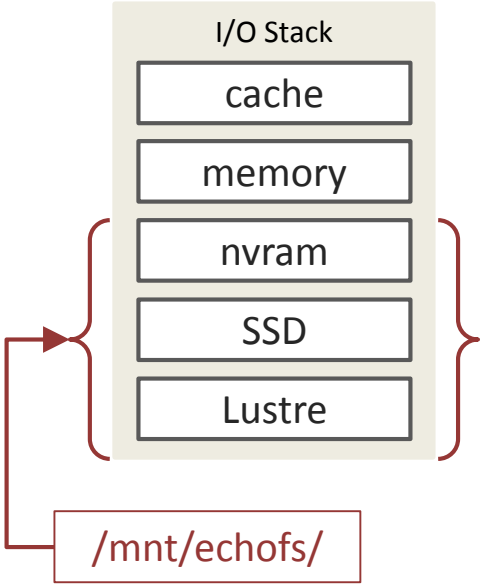


2. How can we bring the benefits from these layers to legacy code?

OUR SOLUTION: MANAGING ACCESS THROUGH A USER-LEVEL FILESYSTEM

echofs -> objectives;

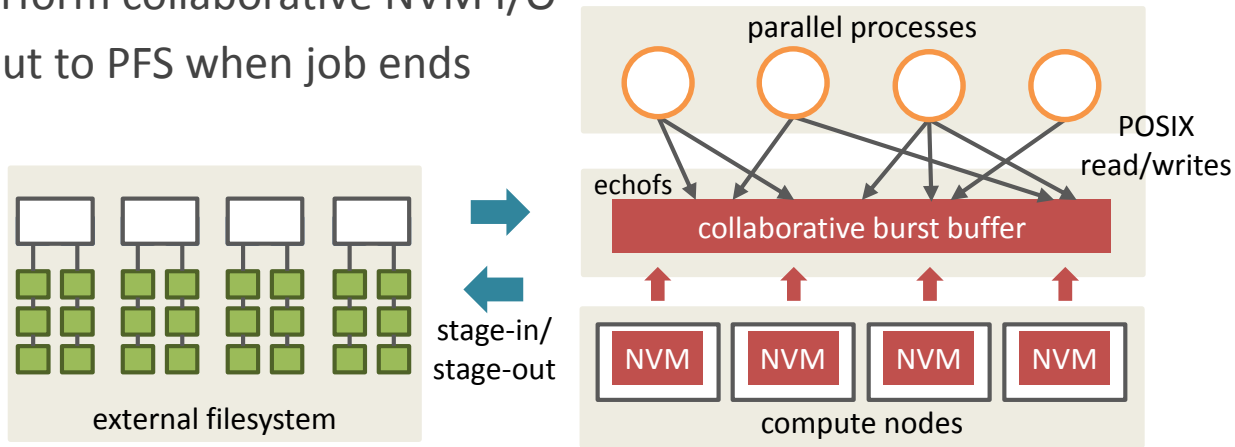
- **First goal:** Allow legacy applications to transparently benefit from new storage layers
  - Accessible storage layers under unique mount point
  - Make new layers readily available to applications
  - I/O stack complexity hidden from applications
  - Allows for automatic management of data location
  - POSIX interface [sorry]



PFS namespace is “echoed”
/mnt/PFS/User/App →
/mnt/ECHOFS/User/App

echofs -> objectives;

- **Second goal:** construct a collaborative burst buffer by joining NVM regions assigned to a batch job by scheduler [SLURM]
  - Filesystem's lifetime linked to batch job's lifetime
  - Input files staged into NVM before job starts
  - Allow HPC jobs to perform collaborative NVM I/O
  - Output files staged out to PFS when job ends



echofs -> intended workflow;

- User provides job I/O requirements through SLURM
  - Nodes required, files accessed, type of access [in|out|inout], expected lifetime [temporary|persistent], expected “survivability”, required POSIX semantics [?], …
- SLURM allocates nodes and mounts echofs across them
  - Also forwards I/O requirements through API
- echofs builds the CBB and fills it with input files
  - When finished, SLURM starts the batch job

echofs -> intended workflow;

- User provides job I/O requirements through SLURM
  - Nodes required, files accessed, type of access [in|out|inout], expected lifetime [temporary|persistent], expected “survivability”, required POSIX semantics [?], …
- SLURM allocates nodes and mounts echofs across them
  - Also forwards I/O requirements through API
- echofs builds the CBB and fills it with input files
  - When finished, SLURM starts the batch job

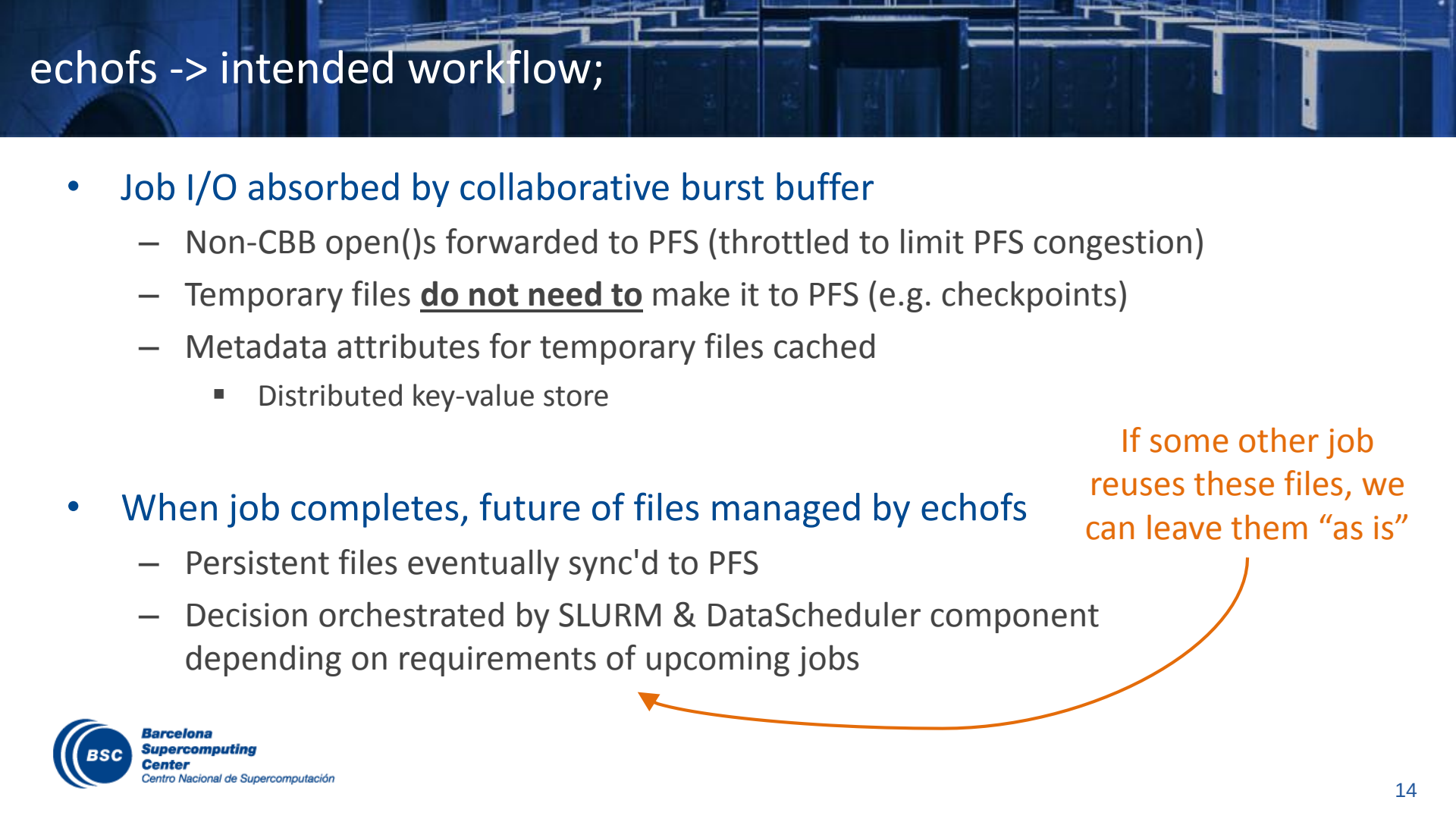
We can’t expect optimization details from users, but maybe for them to offer us enough hints…

echofs -> intended workflow;

- Job I/O absorbed by collaborative burst buffer
  - Non-CBB open()s forwarded to PFS (throttled to limit PFS congestion)
  - Temporary files **do not need to** make it to PFS (e.g. checkpoints)
  - Metadata attributes for temporary files cached
    - Distributed key-value store

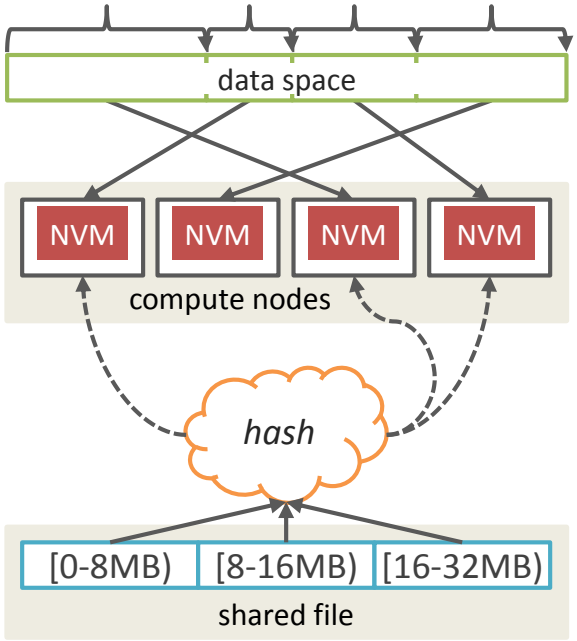
- When job completes, future of files managed by echofs
  - Persistent files eventually sync'd to PFS
  - Decision orchestrated by SLURM & DataScheduler component depending on requirements of upcoming jobs

If some other job reuses these files, we can leave them “as is”



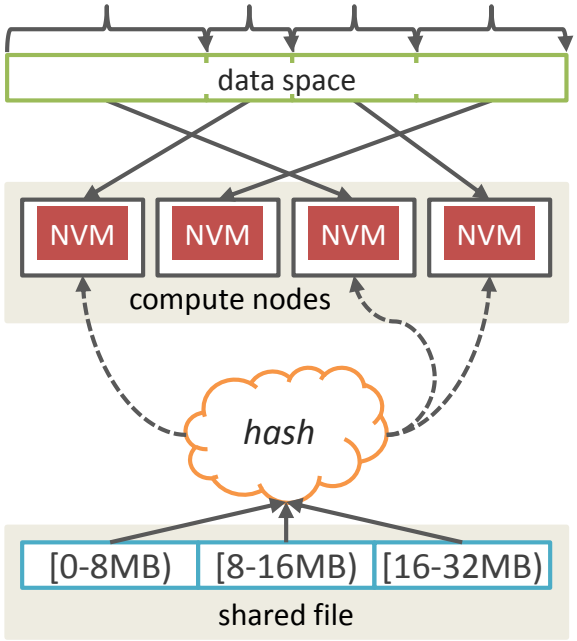
echofs -> data distribution;

- Distributed data servers
  - Job's data space partitioned across compute nodes
  - Each node acts as data server for its partition
  - Each node acts as data client for other partitions
- Pseudo-random file segment distribution
  - No replication ⇒ avoid coherence mechanisms
  - Resiliency through erasure codes (eventually)
  - Each node acts as lock manager for its partition



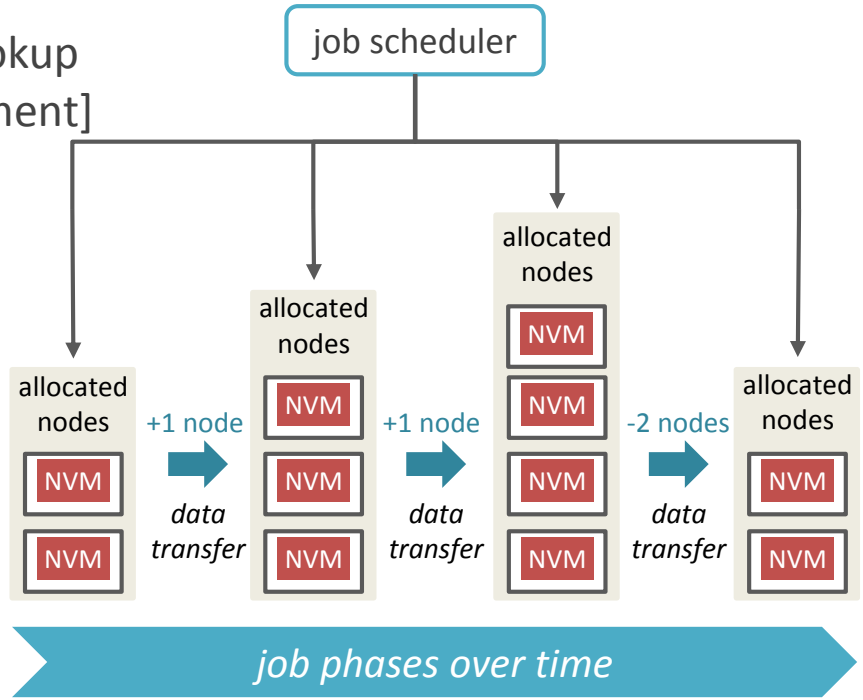
echofs -> data distribution;

- Why pseudo-random?
  - Efficient & decentralized segment lookup [no metadata request to lookup segment]
  - Balance workload w.r.t. partition size
  - Allows for collaborative I/O



echofs -> data distribution;

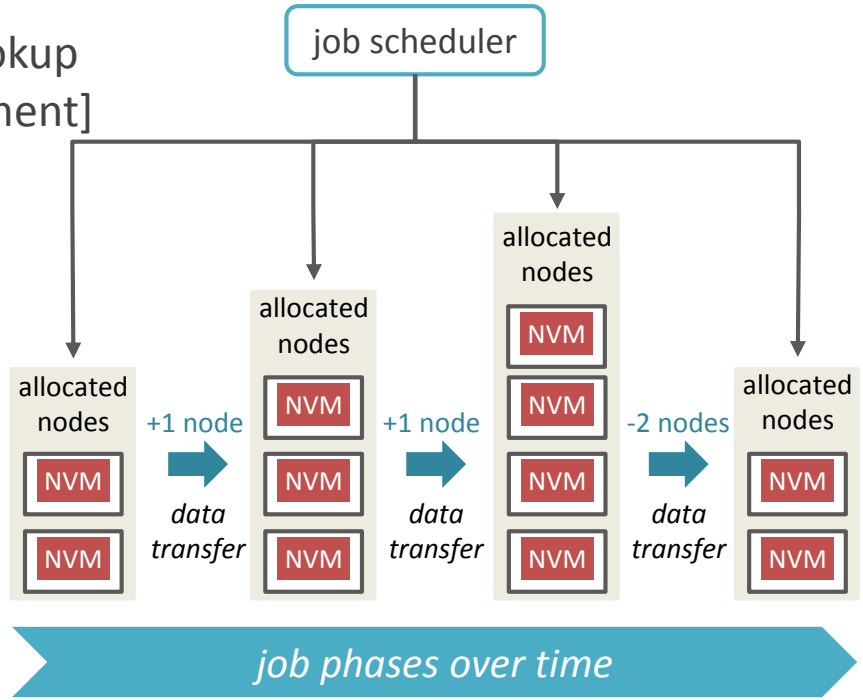
- Why pseudo-random?
  - Efficient & decentralized segment lookup [no metadata request to lookup segment]
  - Balance workload w.r.t. partition size
  - Allows for collaborative I/O
  - **Guarantees minimal movements of data if node allocation changes [future research on elasticity]**



echofs -> data distribution;

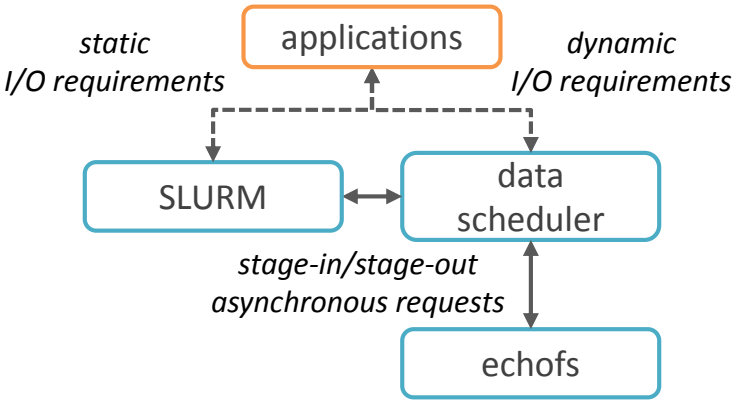
- Why pseudo-random?
  - Efficient & decentralized segment lookup [no metadata request to lookup segment]
  - Balance workload w.r.t. partition size
  - Allows for collaborative I/O
  - **Guarantees minimal movements of data if node allocation changes [future research on elasticity]**

Other strategies would be possible depending on job semantics



echofs -> integration with batch scheduler;

- Data Scheduler daemon external to echofs
  - Interfaces SLURM & echofs
    - Allows SLURM to send requests to echofs
    - Allows echofs to ACK these requests
  - Offers an API to [non-legacy] applications willing to send I/O hints to echofs
  - In the future will coordinate w/ SLURM to decide when different echofs instances should access PFS
    [data-aware job scheduling]



Summary;

- Main features:
  - Ephemeral filesystem linked to job lifetime
  - Allows legacy applications to benefit from newer storage technologies
  - Provides aggregate I/O for applications

- Research goals:
  - Improve coordination w/ job scheduler and other HPC management infrastructure
  - Investigate ad-hoc data distributions tailored for each job I/O
  - Scheduler-triggered specific optimizations for jobs/files

Food for thought;

- POSIX compliance is hard…
  - But maybe we don't need **FULL COMPLIANCE** for **ALL** jobs…
- Adding I/O-awareness to the scheduler is important…
  - Allows wasting I/O work already done…
  - … but requires user/developer collaboration (tricky…)
- User-level filesystems/libraries solve very specific I/O problems…
  - Can we reuse/integrate these efforts? Can we learn what works for a specific application, characterize it & automatically run similar ones in a "best fit" FS?

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

For further information please contact

alberto.miranda@bsc.es, ramon.nou@bsc.es, toni.cortes@bsc.es