www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Using NVM Transparently with a Unified Filesystem

NEXTGenIO & SAGE: Working towards Exascale I/O

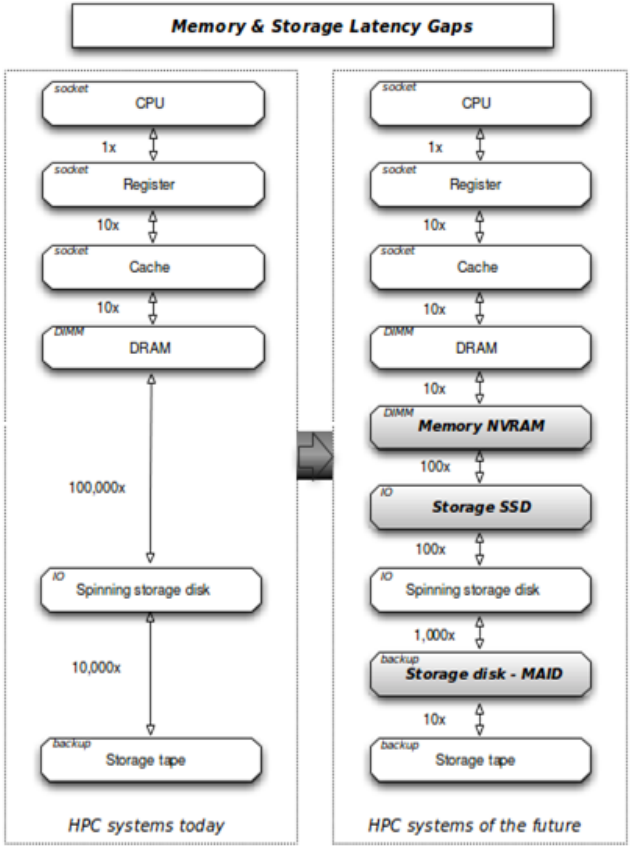
Ramon Nou (Storage Systems - BSC)

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

Barcelona, May 2017

I/O: a Fundamental Exascale Challenge!

- Petascale already struggles with I/O…
  - Parallelism beyond 100 million threads
  - Reading and writing data to parallel filesystems, checkpoints
- Next generation NVM technologies will change memory and storage hierarchies significantly
  - HPC systems and Data Intensive systems will merge.

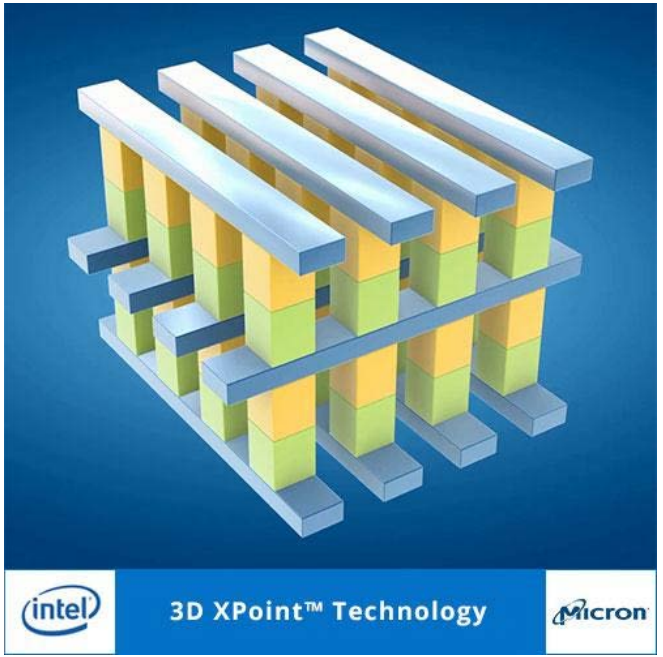


Burst Buffers

- Fast storage devices that temporarily store application data before sending it to PFS
  - Goal: Absorb peak I/O to avoid overloading 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

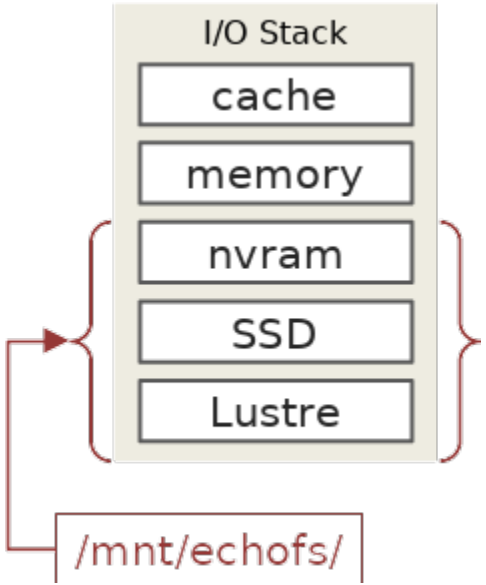
The NEXTGenIO EU Project

- New server architecture based on Intel's 3D XPoint™ NVM technology
- NVRAM as a fundamental component of compute nodes
- Low-latency/high-throughput interconnection network



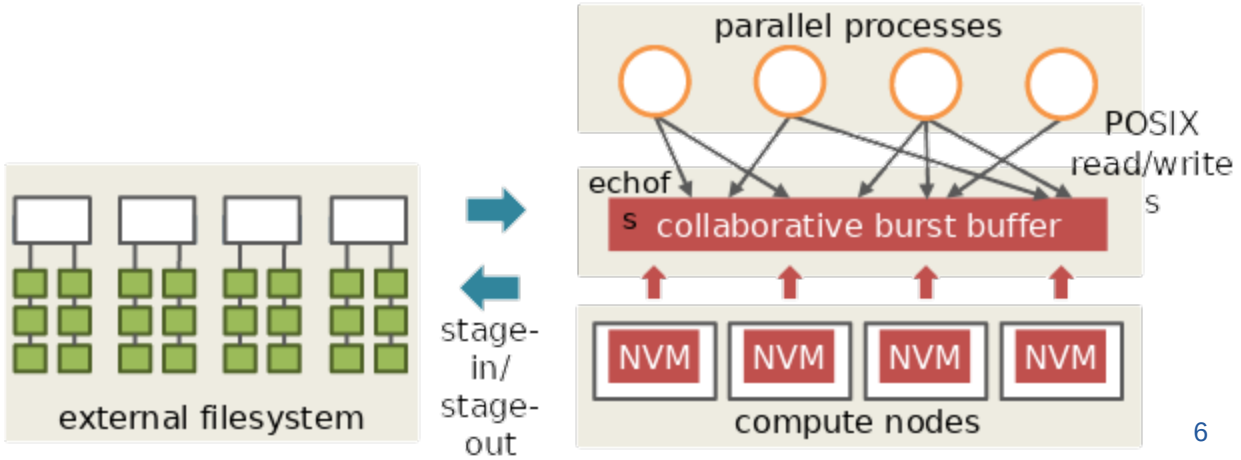
echofs: Features and Goals

- **First goal:** Allow legacy applications to transparently benefit from new storage layers
  - Accessible layers joined under unique mount point
  - New layers readily available to applications
  - I/O stack complexity hidden from applications
  - Automatic management of data location
  - POSIX interface



echofs: Features and Goals

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



echofs: Workflow

- User provides job I/O requirements through SLURM
  - Nodes required, files accessed, type of access (e.g. input/output/inout), expected lifetime of data (e.g. temporary/persistent).

- Compute nodes are allocated and echofs is mounted on them
  - The PFS's namespace is replicated → relative paths maintained

/PFS/user1/Application1 --> /ECHOFS/user1/Application1

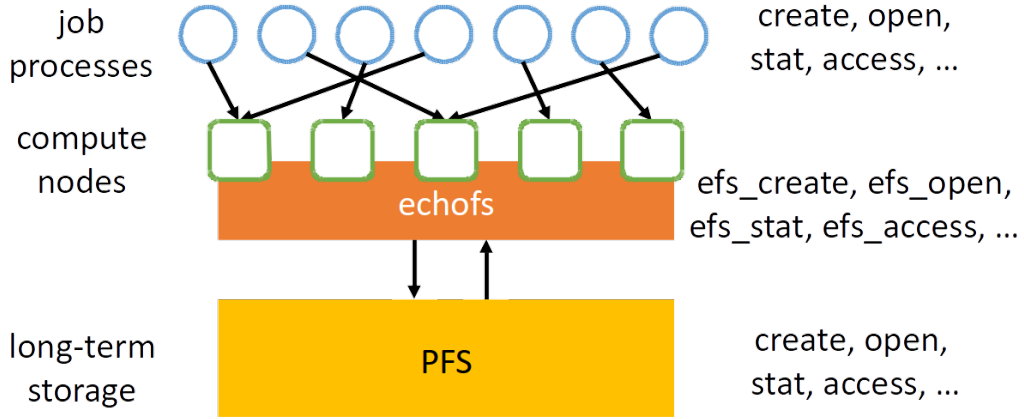
- Files specified by the job are copied from the PFS into NVM
- Job starts and I/O is redirected to NVM (or other levels)

echofs: Workflow

- After job completes, future of data is decided by echofs
  - Persistent data *eventually* transferred back to PFS
    - Orchestrated by DataScheduler NEXTGenIO component.
  - Temporary data is deleted from NVM
    - Reduces metadata issues to the PFS

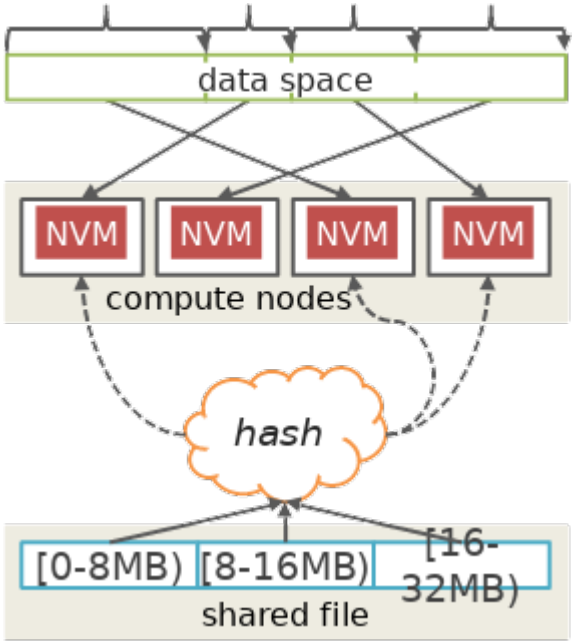
echofs: Architecture Overview

- No centralized Metadata Servers
  - Metadata operations for persistent files forwarded to PFS
    - *Forwarding rate adjusted to avoid overwhelming the PFS (e.g. a job creating millions of files)*
  - Metadata operations for temporary files “absorbed” by echofs
  - PFS manages metadata concurrency/consistency



echofs: Architecture Overview

- Distributed Data Servers
  - Data space is partitioned across compute nodes
  - Each node acts as a data server for its partition
  - Each node acts as a data client of other partitions
  - Low-latency network required for data transfers

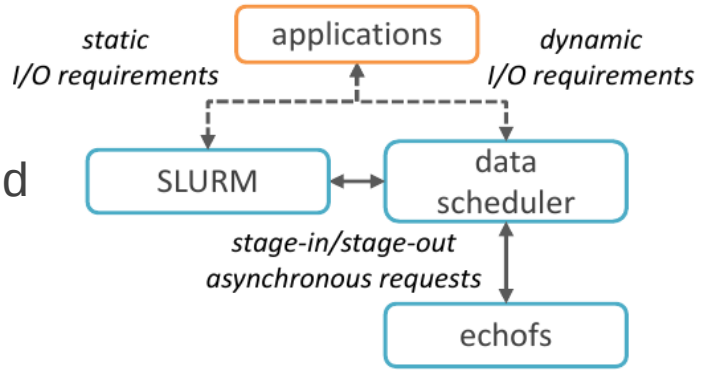


echofs: Architecture Overview

- Data Distribution
  - Static/pseudo-randomized data striping to collaborating nodes
  - No malleability (NextGenIO version)
- No replication ⇒ no distributed coherence mechanisms
  - Nodes act as a lock managers for their partitions to enforce POSIX semantics

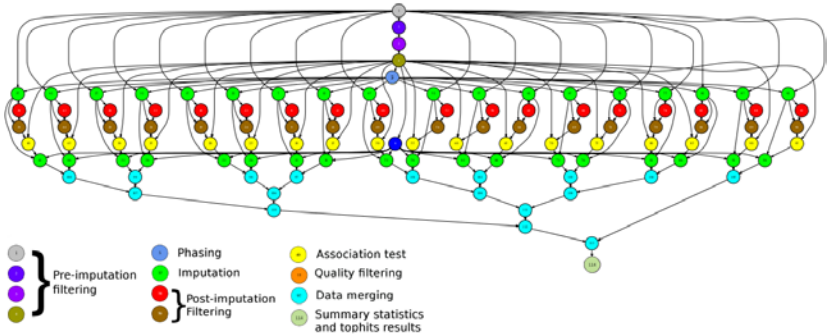
echofs: integration with batch scheduler

- 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



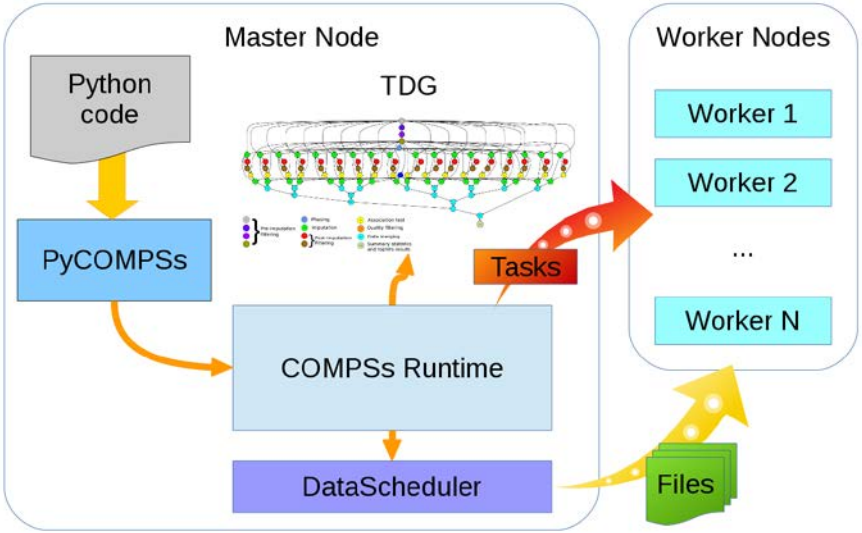
echofs: PyCOMPSs collaboration

- PyCOMPSs
  - Python binding of COMPSs, a task based programming model and runtime which aims to ease the development of parallel applications for distributed infrastructures, such as Clusters and Clouds.
  - Sequential programming and task definition with decorators.
  - At execution time the runtime system is able to exploit the inherent parallelism of applications at task level respecting the existing data dependencies.



echofs: PyCOMPSs collaboration

- PyCOMPSs and DataScheduler
  - PyCOMPSs will be able to put specific files in specific nodes using the DataScheduler API.
    - Actions could be translated to fs (echofs...) data movements.



Conclusion

- Main features:
  - Ephemeral filesystem linked to job lifetime
  - Allows legacy applications to benefit from newer storage technologies
  - Provides aggregate I/O for applications
  - Allows to “shape” the I/O using hints.

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

For further information please contact

ramon.nou@bsc.es / alberto.miranda@bsc.es /
francisco.conejero@bsc.es / toni.cortes@bsc.es