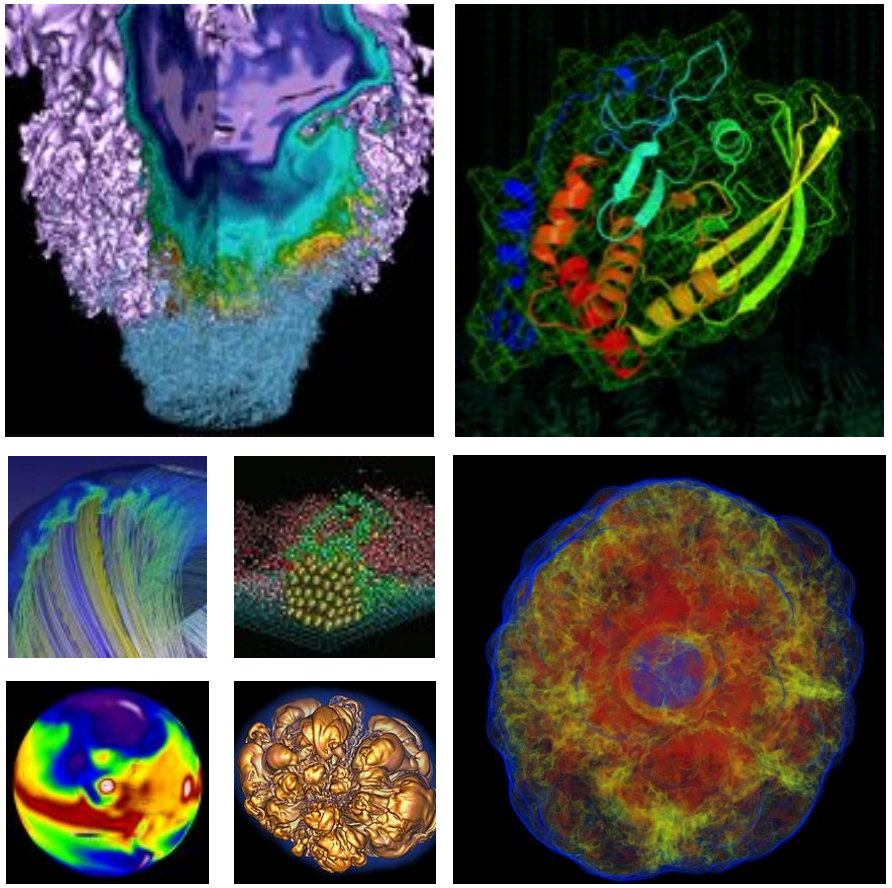
Accelerate your IO with the Burst Buffer



Debbie Bard
Data and Analytics Services
NERSC

CUG17 Tutorial

HPC memory hierarchy

Past

On Chip

Off Chip

CPU

Memory (DRAM)

Storage (HDD)

Future

On Chip

Off Chip

CPU

Near Memory (HBM)

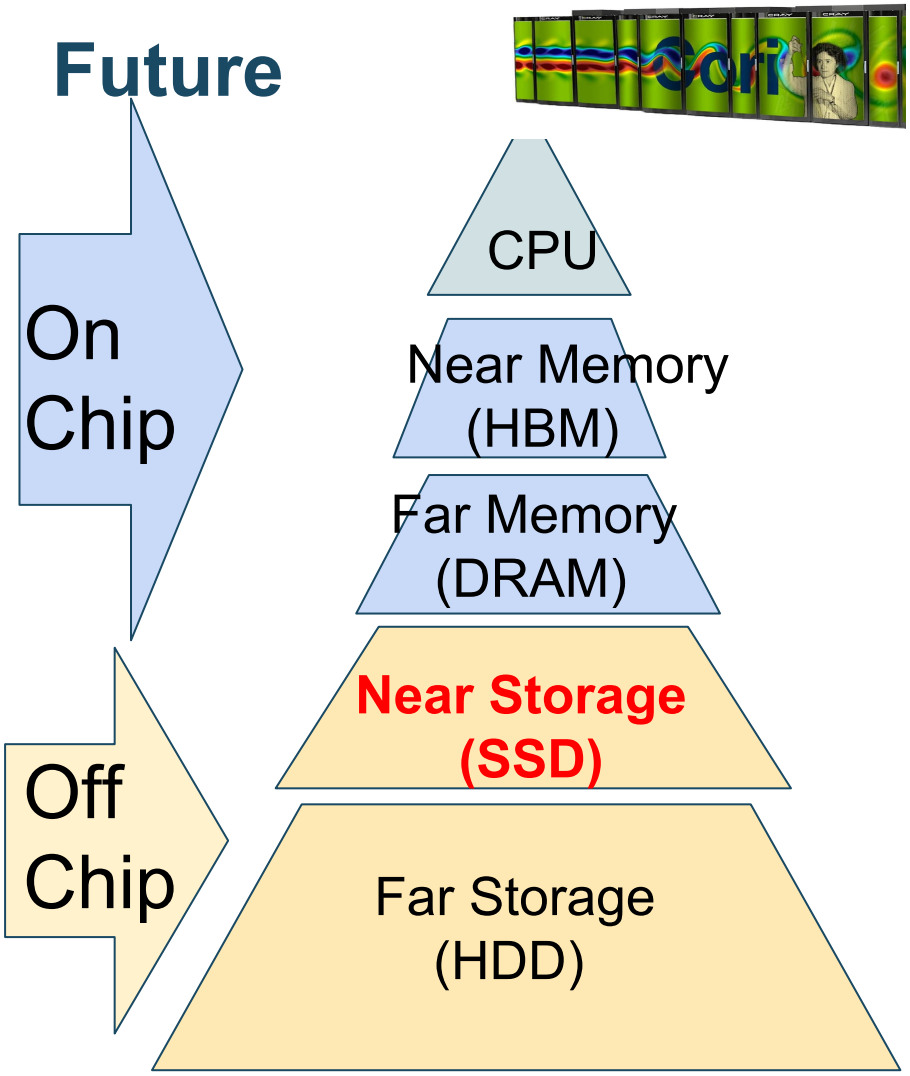
Far Memory (DRAM)

Near Storage (SSD)

Far Storage (HDD)

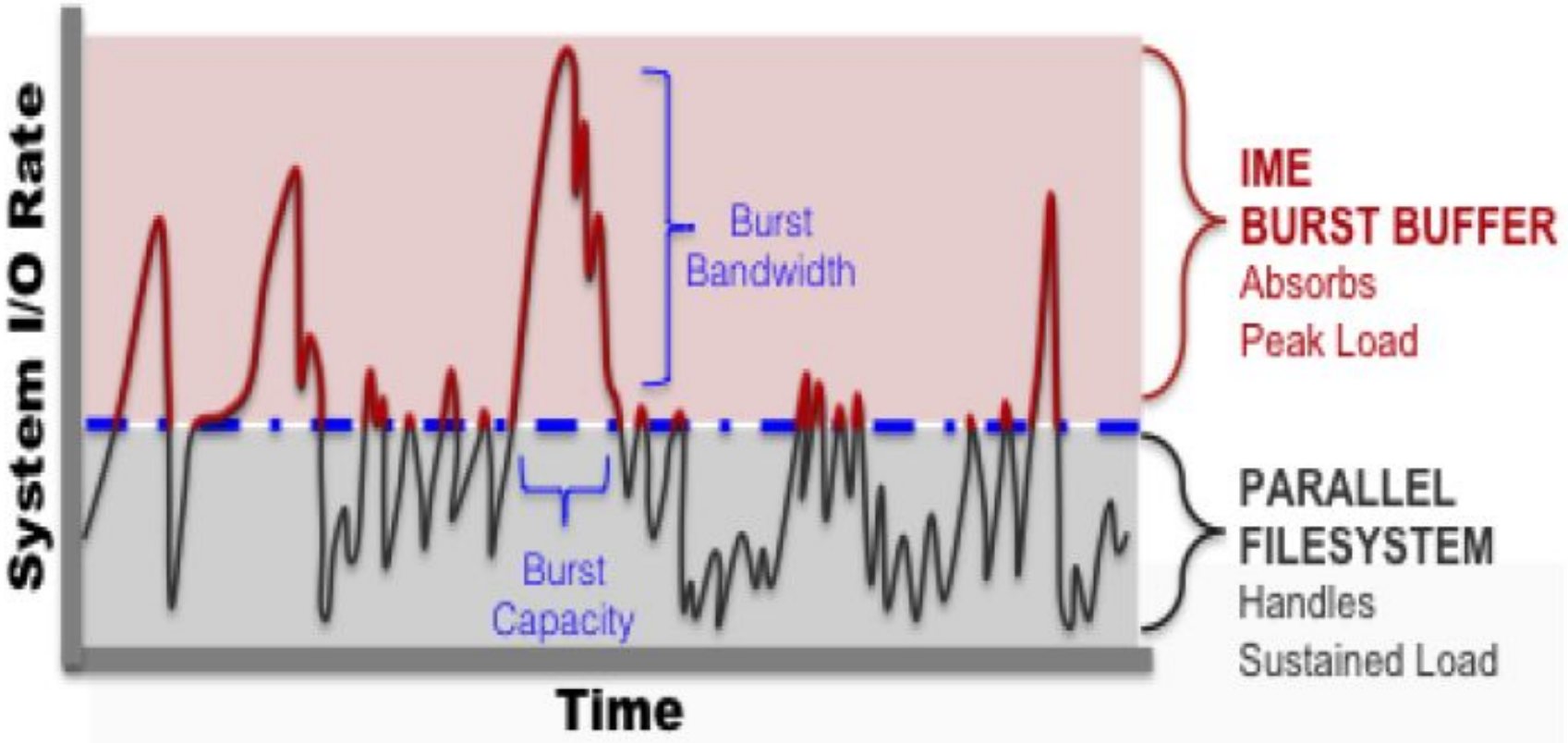
HPC memory hierarchy

- ***Silicon and system integration***
- **Bring everything – storage, memory, interconnect – closer to the cores**
- **Raise center of gravity of memory pyramid, and make it fatter**
  - *Enable faster and more efficient data movement*
  - *Scientific Big Data: Addressing Volume, Velocity*



Why an SSD Burst Buffer?

- **Motivation: Handle spikes in I/O bandwidth requirements**
  - Reduce overall application run time
  - Compute resources are idle during I/O bursts



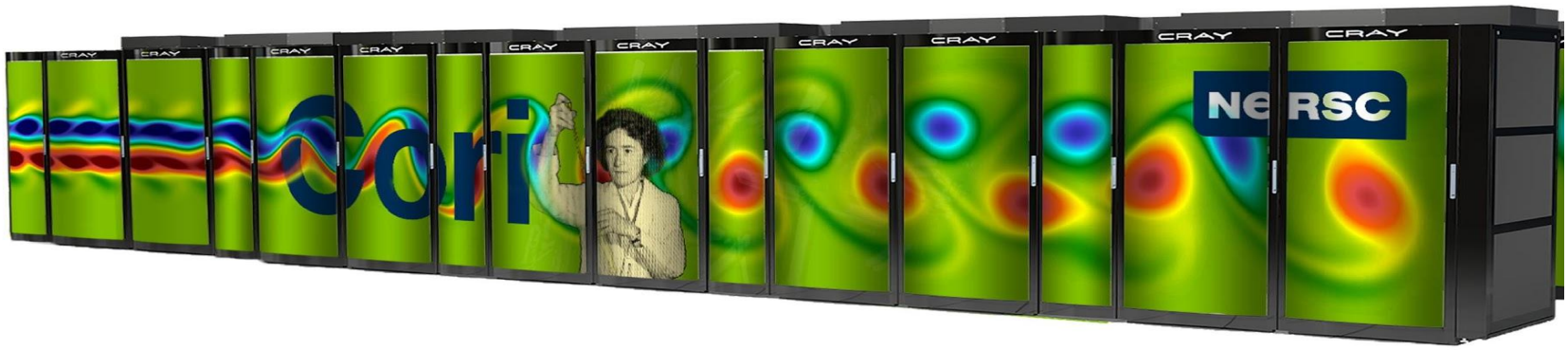
Why an SSD Burst Buffer?

- **Motivation: Handle spikes in I/O bandwidth requirements**
  - Reduce overall application run time
  - Compute resources are idle during I/O bursts
- **Some user applications have challenging I/O patterns**
  - High IOPs, random reads, different concurrency… perfect for SSDs
- **Cost rationale: Disk-based PFS bandwidth is expensive**
  - Disk capacity is relatively cheap
  - SSD *bandwidth* is relatively cheap

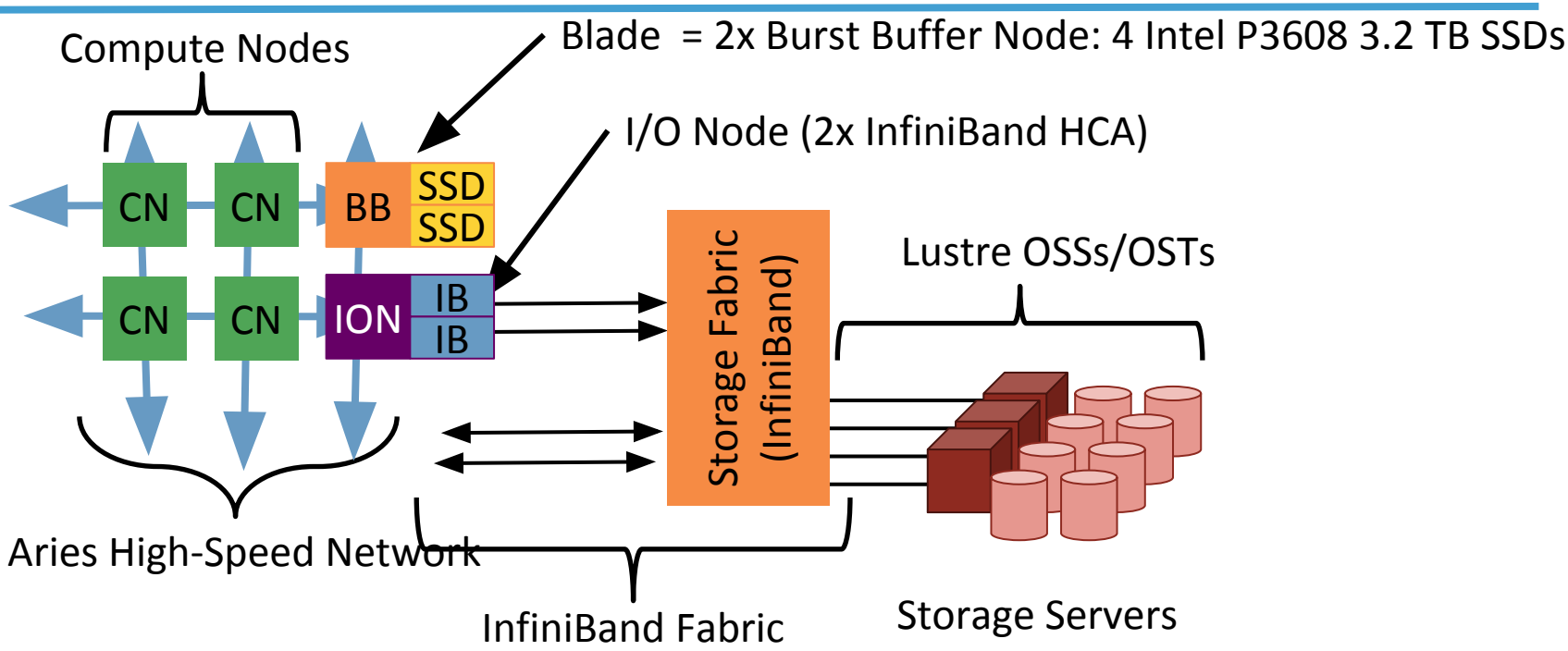
    =>Separate bandwidth and spinning disk
    - Provide high BW without wasting PFS capacity
    - Leverage Cray Aries network speed

Cori @ NERSC

- **NERSC at LBL, production HPC center for DoE**
  - >7000 diverse users across all DoE science domains
- **Cori – NERSCs Newest Supercomputer – Cray XC40**
  - 2,388 Intel Haswell dual 16-core nodes
  - 9,688 Intel Knights Landing Xeon Phi nodes, 68 cores
- **Cray Aries high-speed "dragonfly" topology interconnect**
- **Lustre Filesystem: 27 PB ; 248 OSTs; 700 GB/s peak performance**
- **1.8PB of Burst Buffer**

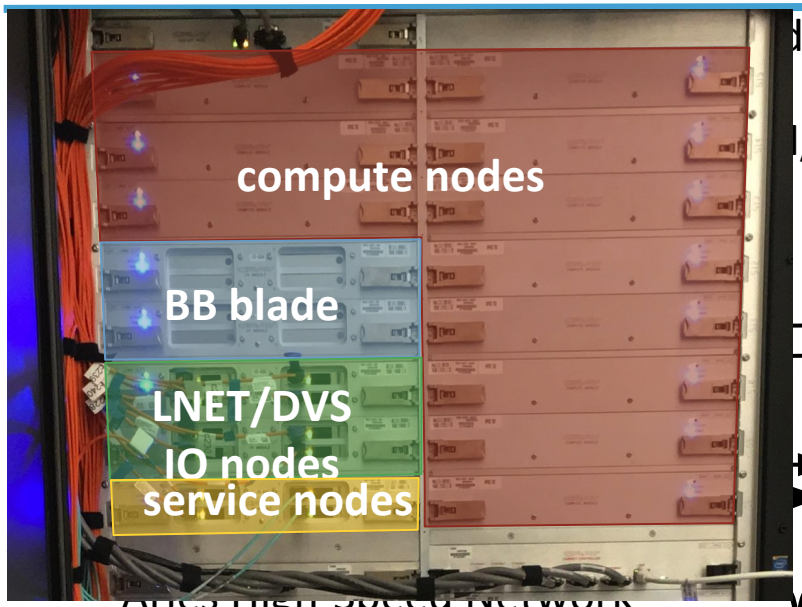


Burst Buffer Architecture



- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or 'persistent').
- Users see a POSIX filesystem
- Filesystem can be striped across multiple BB nodes (depending on allocation size requested)

Burst Buffer Architecture



de = 2x Burst Buffer Node: 4 Intel P3608 3.2 TB SSDs

I/O Node (2x InfiniBand HCA)

Storage Fabric (InfiniBand)

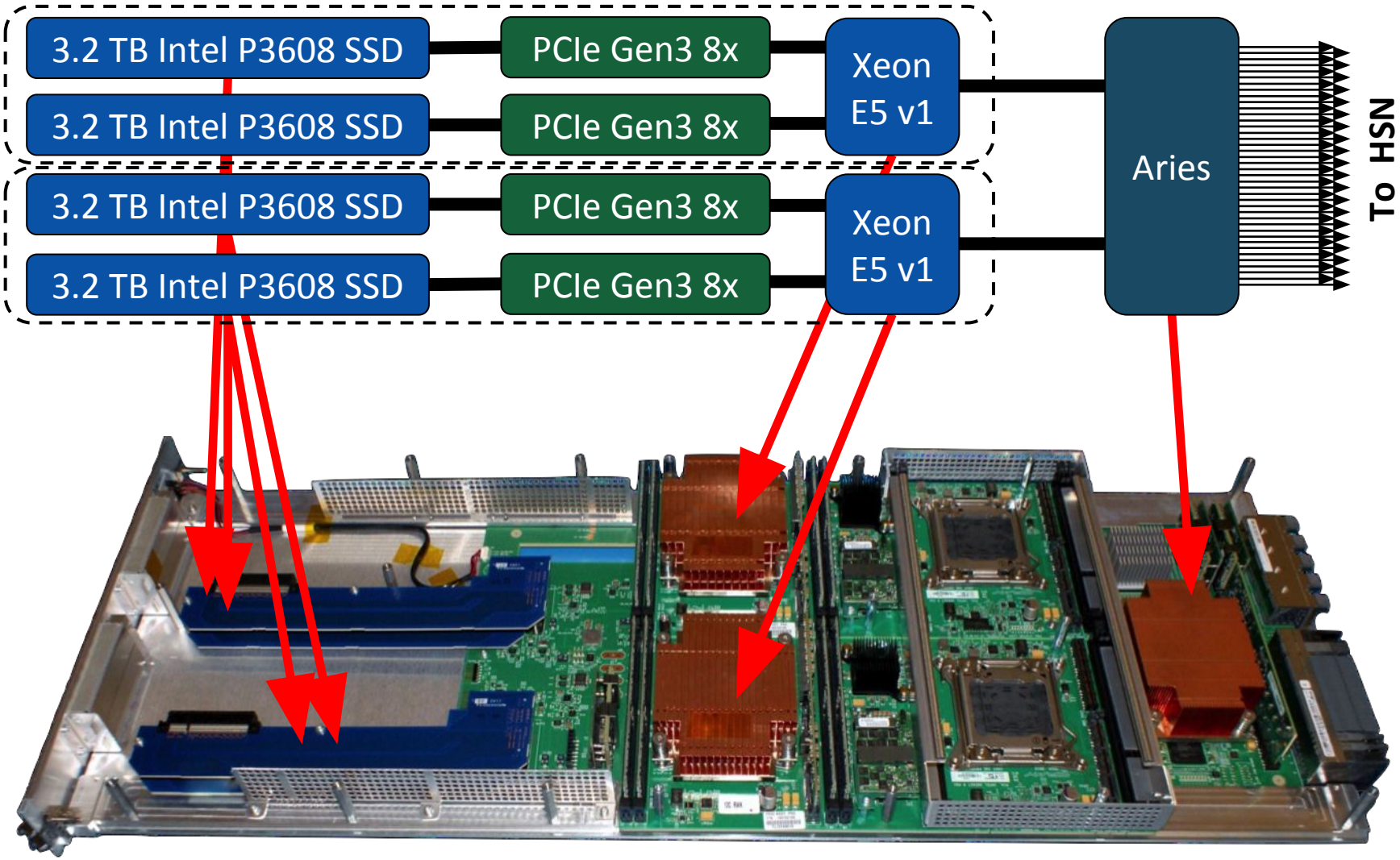
Lustre OSSs/OSTs

InfiniBand Fabric

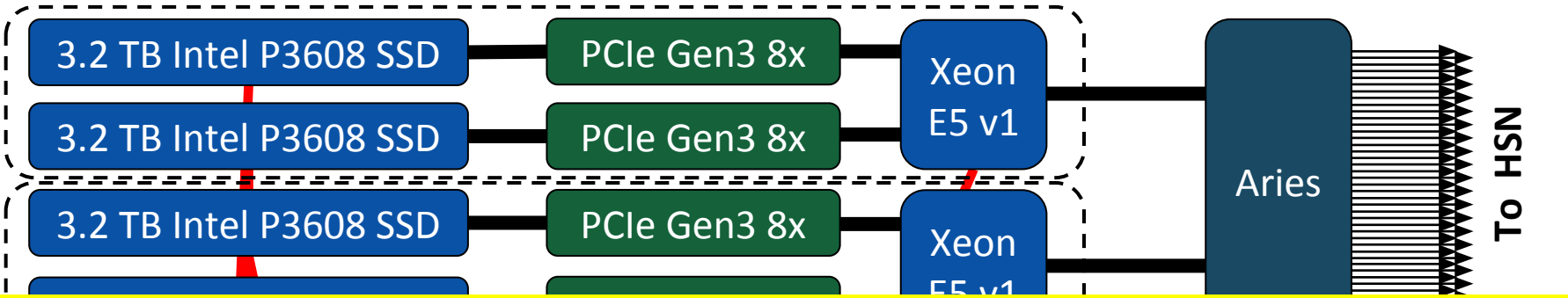
Storage Servers

- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or 'persistent').
- Users see a POSIX filesystem
- Filesystem can be striped across multiple BB nodes (depending on allocation size requested)

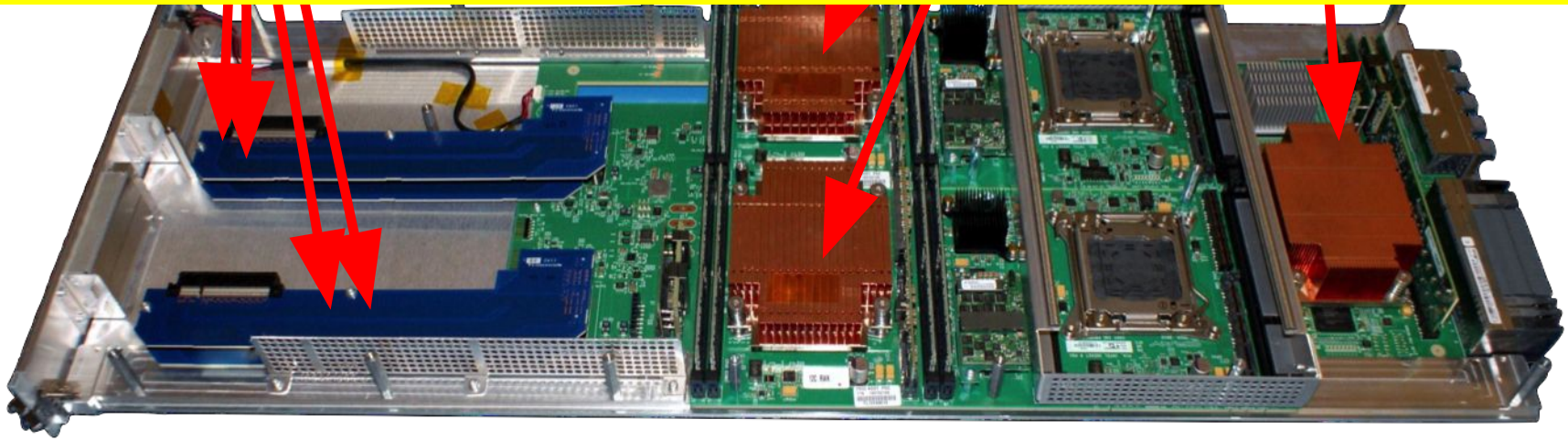
Burst Buffer Blade = 2xNodes



Burst Buffer Blade = 2xNodes

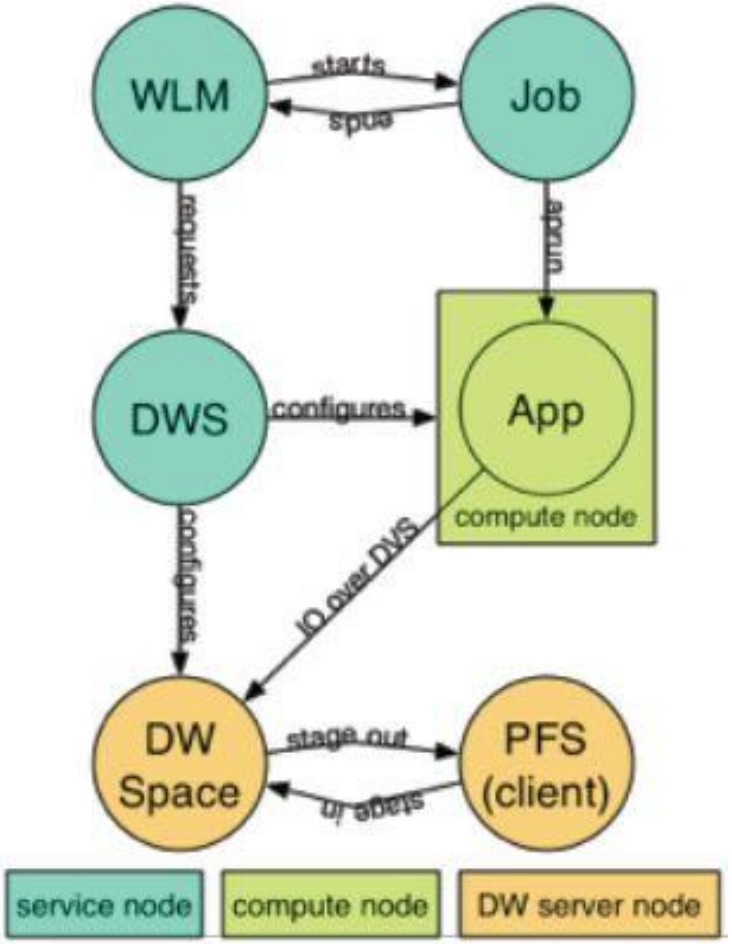


- ~1.8PiB of SSDs over 288 nodes
- Accessible from both HSW and KNL nodes



DataWarp: Under the hood

- Workload Manager (Slurm) schedules job in the queue on Cori
- DataWarp Service (DWS) configures DW space and compute node access to DW
- DataWarp Filesystem handles stage interactions with PFS (Parallel File System, i.e. scratch)
- Compute nodes access DW via a mount point

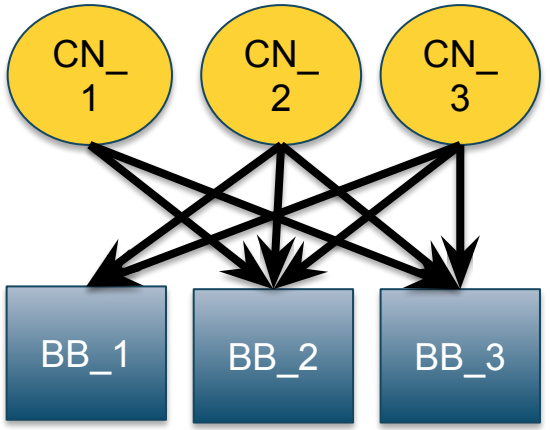


Two kinds of DataWarp Instances

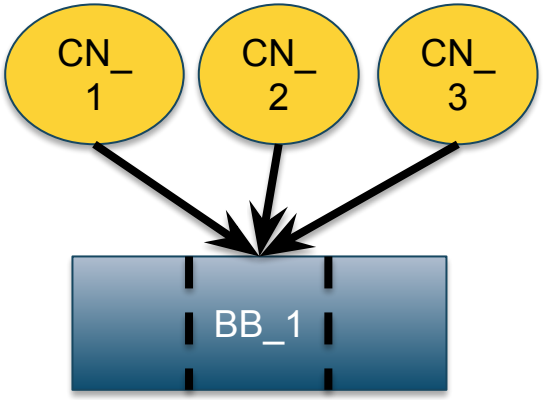
- **"Instance": an allocation on the BB**
- **Can it be shared? What is its lifetime?**
  - **Per-Job Instance**
    - Can only be used by job that creates it
    - Lifetime is the same as the creating job
    - Use cases: PFS staging, application scratch, checkpoints
  - **Persistent Instance**
    - Can be used by any job (subject to UNIX file permissions)
    - Lifetime is controlled by creator
    - Use cases: Shared data, PFS staging, Coupled job workflow
    - ***NOT for long-term storage of data!***

Two DataWarp Access Modes

- **Striped ("Shared")**
  - Files are striped across all DataWarp nodes
  - Files are visible to all compute nodes
    Aggregates both capacity and BW per file
  - One DataWarp node elected as the metadata server (MDS)



- **Private**
  - Files are assigned to one or more DataWarp node (can chose to stripe)
  - File are visible to *only the compute node that created them*
  - Each DataWarp node is an MDS for one or more compute nodes



How to use DataWarp

- **Principal user access: SLURM Job script directives: `#DW`**
  - Allocate job or persistent DataWarp space
  - Stage files or directories in from PFS to DW; out DW to PFS
  - Access BB mount point via `$DW_JOB_STRIPED`, `$DW_JOB_PRIVATE`, `$DW_PERSISTENT_STRIPED_name`
- **We'll go through this in more detail later….**
- **User library API – `libdatawarp`**
  - Allows direct control of staging files asynchronously
  - C library interface
  - https://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/#toc-anchor-8
  - https://github.com/NERSC/BB-unit-tests/tree/master/datawarpAPI

Why not node-local SSDs?

- **Average >1000 jobs running on Cori at any time**
- **Diverse workload**
  - –Many NERSC users are IO-bound
  - –Small-scale compute jobs, large-scale IO needs
- **Persistent BB reservations enable medium-term data access without tying up compute nodes**
  - –Multi-stage workflows with differing concurrencies can simultaneously access files on BB.
- **Easier to stream data directly into BB from external experiment**
- ***Configurable BB makes sense for our user load***

Benchmark Performance on Cori

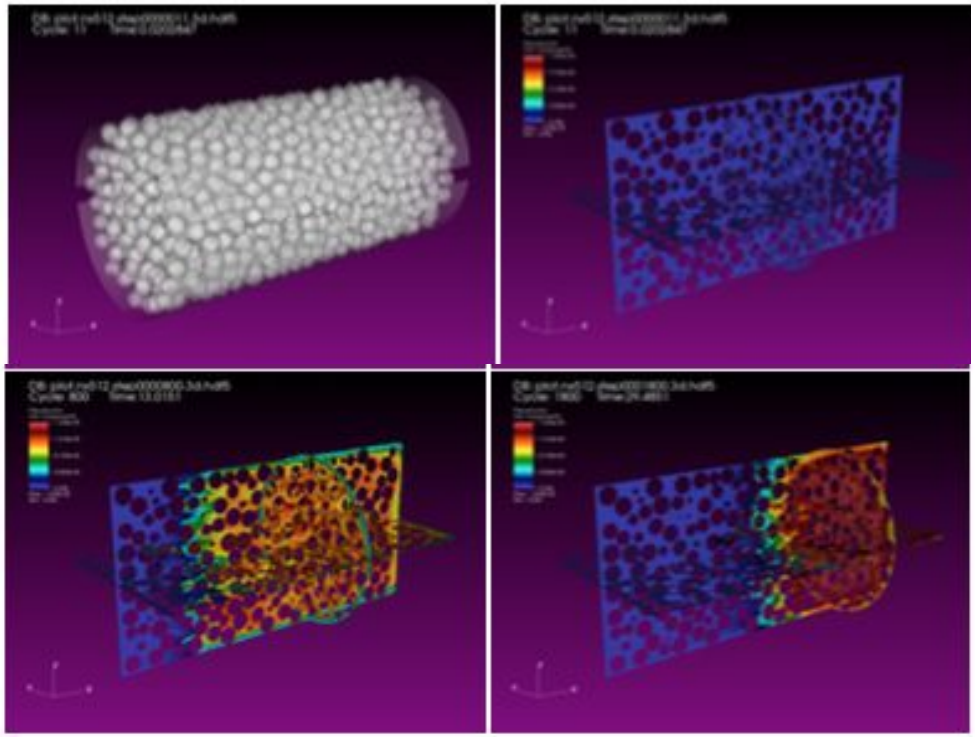
- **Burst Buffer is now doing very well against benchmark performance targets**
  - Out-performs Lustre significantly
  - Fastest IO system in the world!

	IOR Posix FPP		IOR MPIO Shared File		IOPS	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Best Measured (287 Burst Buffer Nodes : 11120 Compute Nodes; 4 ranks/node)*	1.7 TB/s	1.6 TB/s	1.3 TB/s	1.4 TB/s	28M	13M

**Bandwidth tests: 8 GB block-size 1MB transfers IOPS tests: 1M blocks 4k transfer*

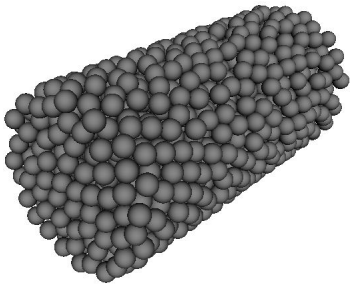
Burst Buffer enables Workflow coupling and visualization

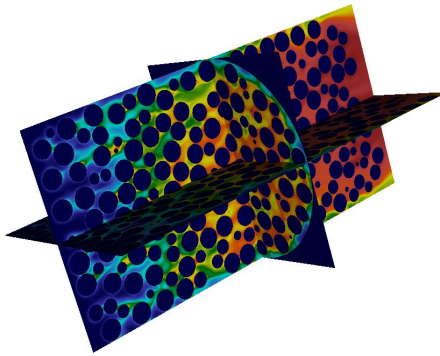
- **Success story: Burst Buffer can enable new workflows that were difficult to orchestrate using Lustre alone.**



Workflows Use Case: ChomboCrunch + VisIT

- **ChomboCrunch simulates pore-scale reactive transport processes associated with carbon sequestration**
  - –Flow of liquids through ground layers
  - –All MPI ranks write to single shared HDF5 '.plt' file.
  - –Higher resolution -> more accurate simulation - more data output (O(100TB))

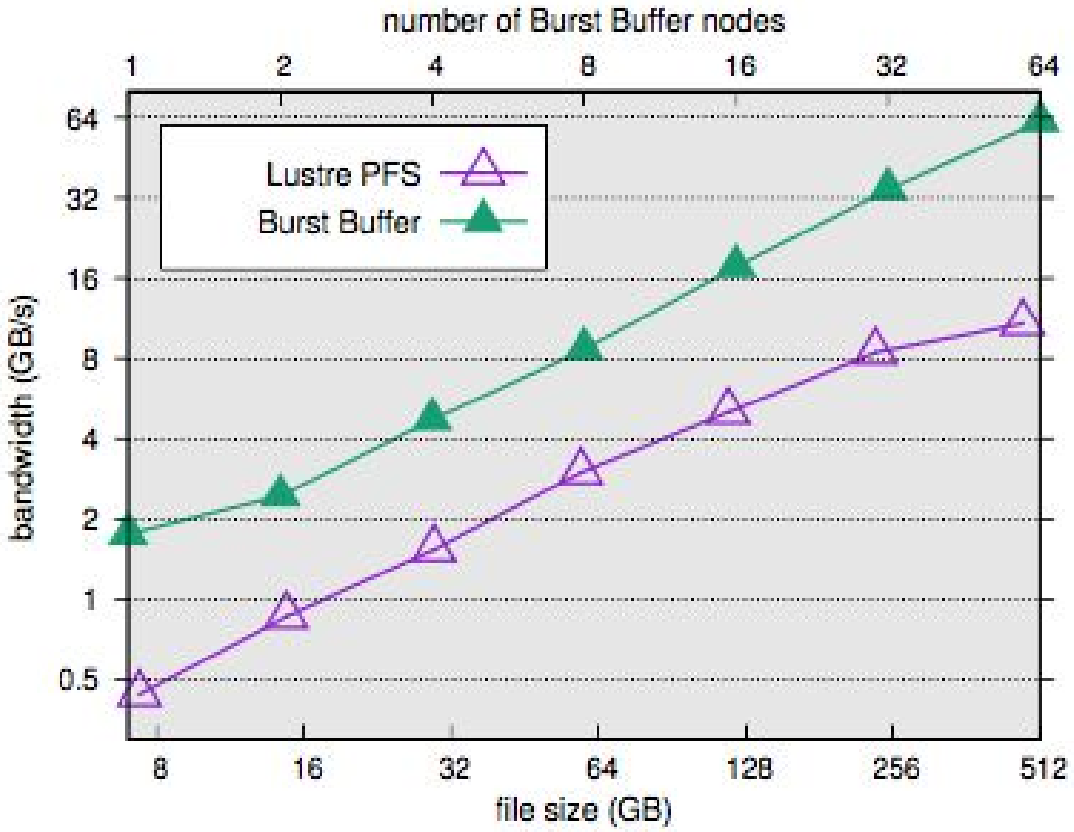




- **VisIT – visualisation and analysis tool for scientific data**
  - Reads '.plt' files produces '.png' for encoding into movie
- **Before: used Lustre to *store* intermediate files.**

Workflows Use Case: ChomboCrunch + VisIT

- **Burst Buffer significantly out-performs Lustre for this application at all resolution levels**
  - Did not require any additional tuning!
- **Bandwidth achieved is around a quarter of peak, scales well.**

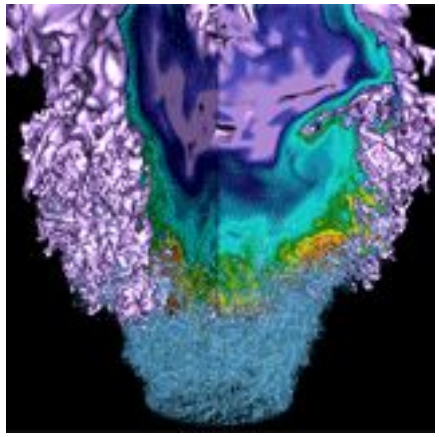


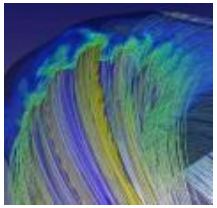
Compute node/BB node scaled: 16/1 to 1024/ 64
Lustre results used a 1MB stripe size and a stripe count of 72 OSTs

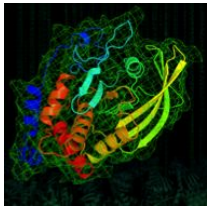
Summary

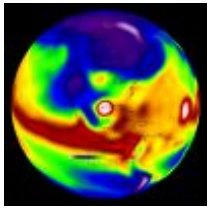
- **Burst Buffer technology has been available to NERSC users for over 18 months**
  - Some initial teething problems, but a productive collaboration with Cray has meant nearly all users see an improvement in their IO (compared to Lustre) out of the box
- **For the rest of this tutorial we'll hear more of user experiences from KAUST, the administrators perspective, then you'll have a chance to run some simple examples on Cori's Burst Buffer before we have a detailed use case of a complex workflow, from KAUST.**

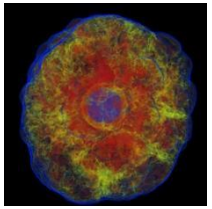
Extra slides

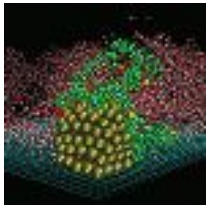












Summary

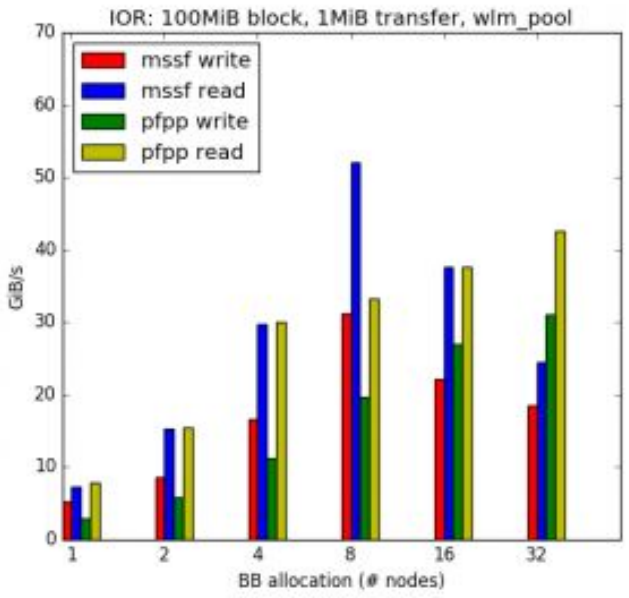
- **NERSC has the first Burst Buffer for open science in the USA**
  - –And the first in the world that is being tested for real use cases!
- **Users are able to take advantage of SSD performance**
  - –Some tuning may be required to maximise performance
- **Many bugs now worked through**
  - –But care is needed when using this new technology!
- **User experience today is generally good**
  - –Let us know if you're interested in using the Burst Buffer…

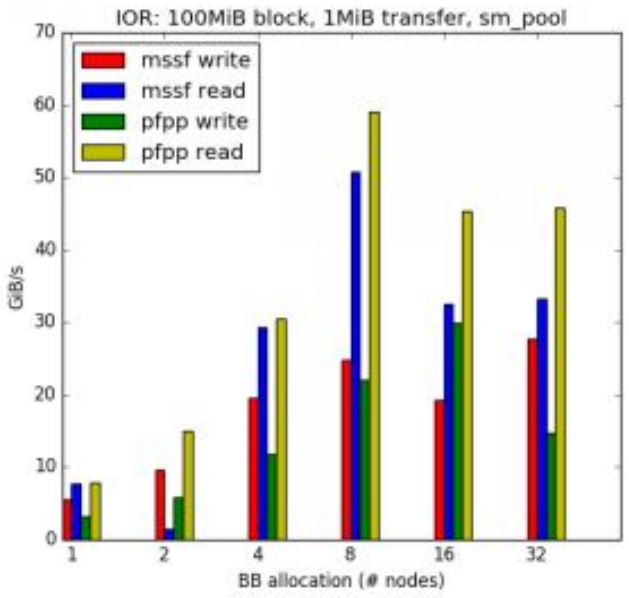
Striping, granularity and pools

- **DataWarp nodes are configured to have "granularity"**
  - Minimum amount of data that will land on one node
- **Two "pools" of DataWarp nodes, with different granularity**
  - wlm_pool (default): 200GiB
    - `#DW jobdw capacity=1000GB access_mode=striped type=scratch pool=wlm_pool`
  - sm_pool: 20.14 GiB
    - `#DW jobdw capacity=1000GB access_mode=striped type=scratch pool=sm_pool`
- **For example, 1.2TiB will be striped over 6 BB nodes in wlm_pool, but over 60 BB nodes in sm_pool**
  - No guarantee that allocation will be spread evenly over SSDs - may see >1 "grain" on a single node

Performance tips

- **Stripe your files across multiple BB servers**
  - To obtain good scaling, need to drive IO with sufficient compute - scale up # BB nodes with # compute nodes





Resources

- **NERSC Burst Buffer Web Pages**

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/

- **Example batch scripts**

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/

- **Burst Buffer Early User Program Paper**

http://www.nersc.gov/assets/Uploads/Nersc-BB-EUP-CUG.pdf

SSD write protection

- **SSDs support a set amount of write activity before they wear out**
- **Runaway application processes may write an excessive amount of data, and therefore, "destroy" the SSDs**
- **Three write protection policies**
  - –Maximum number of bytes written in a period of time
  - –Maximum size of a file in a namespace
  - –Maximum number of files allowed to be created in a namespace
- **Log, error, log and error**
  - –`EROFS` (write window exceeded)
  - –`EMFILE` (maximum files created exceeded)
  - –`EFBIG` (maximum file size exceeded)