ATPESC 2016

Argonne NATIONAL LABORATORY

BURST BUFFERS: A NERSC CASE STUDY

ROB LATHAM
PHIL CARNS
Argonne National Laboratory

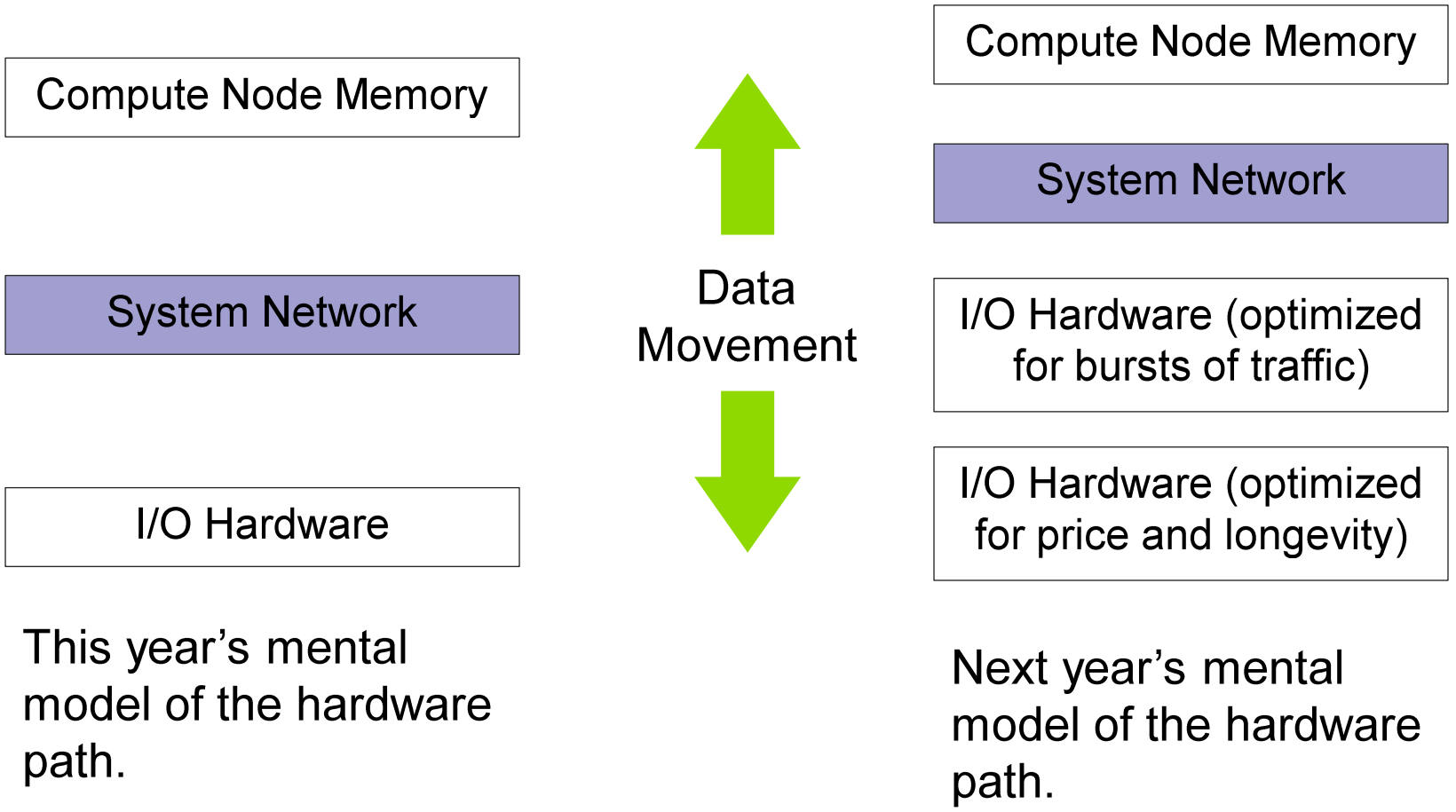
9:30-10:00am, August 11, 2016
St. Charles IL

WHAT'S A BURST?

- Sudden surge in coordinated I/O activity (e.g., from a checkpoint phase)
- Example below (rather old now) quantifies recurring bursts of I/O activity from production jobs on a Blue Gene/P system in December 2011
  - Hundreds of GiBs or more written per burst 5 years ago, but application sizes and storage speeds have gone up since then
  - How do we architect systems to handle this?

Project	Procs	Nodes	Total Written	Run Time (hours)	Avg. Size and Subsequent Idle Time for Write Bursts>1 GiB				
					Count	Size	Size/Node	Size/ION	Idle Time (sec)
PlasmaPhysics	131,072	32,768	67.0 TiB	10.4	1	33.5 TiB	1.0 GiB	67.0 GiB	7554
					1	33.5 TiB	1.0 GiB	67.0 GiB	end of job
Turbulence1	131,072	32,768	8.9 TiB	11.5	5	128.2 GiB	4.0 MiB	256.4 MiB	70
					1	128.2 GiB	4.0 MiB	256.4 MiB	end of job
					421	19.6 GiB	627.2 KiB	39.2 MiB	70
AstroPhysics	32,768	8,096	8.8 TiB	17.7	1	550.9 GiB	68.9 MiB	4.3 GiB	end of job
					8	423.4 GiB	52.9 MiB	3.3 GiB	240
					37	131.5 GiB	16.4 MiB	1.0 GiB	322
					140	1.6 GiB	204.8 KiB	12.8 MiB	318
Turbulence2	4,096	4,096	5.1 TiB	11.6	21	235.8 GiB	59.0 MiB	3.7 GiB	1.2
					1	235.8 GiB	59.0 MiB	3.7 GiB	end of job

REVISITING THE PHYSICAL VIEW

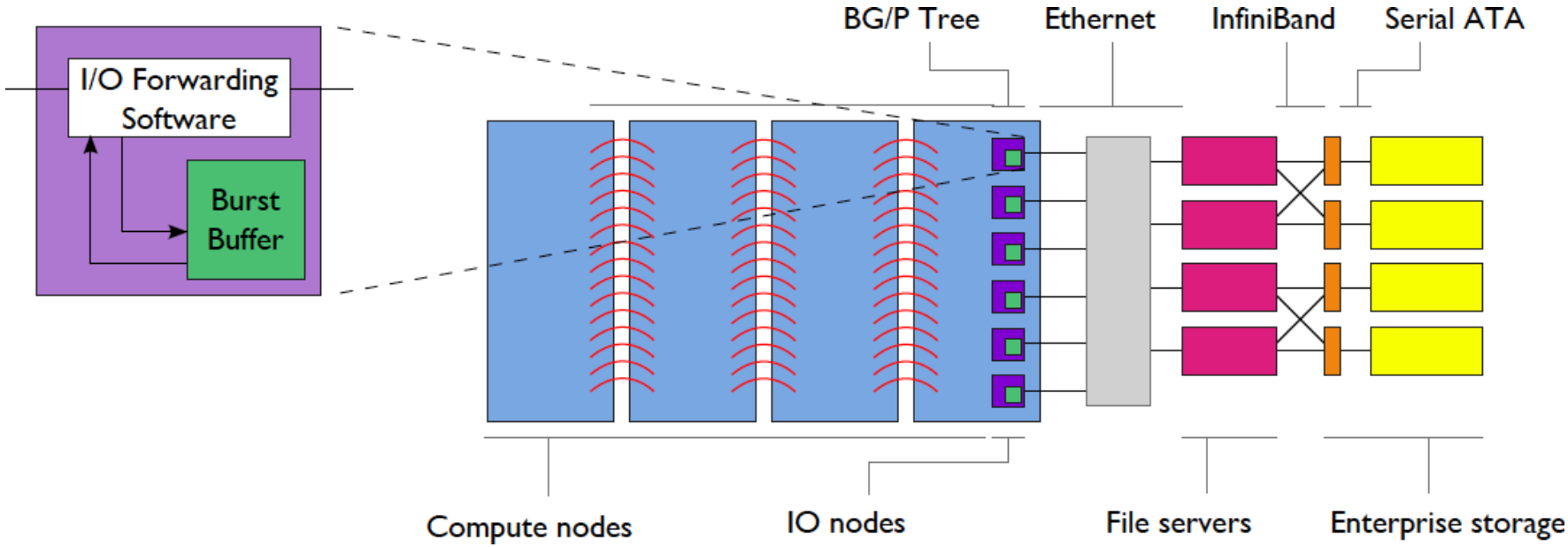


This year's mental model of the hardware path.

Next year's mental model of the hardware path.

ADDING IN SYSTEM STORAGE TO THE STORAGE MODEL

The inclusion of NVRAM storage in future systems is a compelling way to deal with the burstiness of I/O in HPC systems, reducing the peak I/O requirements for external storage. In this case the NVRAM is called a "burst buffer".



BURST BUFFERS: COMING SOON TO A PRODUCTION FACILITY NEAR YOU!

- As with “conventional” parallel file systems, burst buffer systems will vary across facilities. Near-term examples:
  - ALCF: deploying an IBM GSS-based system to transparently cache data on its way to the primary file systems
  - NERSC: deploying a Cray Datawarp system that explicitly associates fast storage nodes with jobs
- Commonalities:
  - Shorter path to compute nodes
  - Handle latency-bound access patterns more effectively
  - Solid state or NVRAM storage devices
  - Limited capacity
- In this presentation we will focus on NERSC as a use case
  - The burst buffer system on Cori is already available to early adopters
  - Concepts are similar across platforms, but implementations differ

Burst buffer early user program

- NERSC has a diverse user base:
  - Over 6500 users on more than 700 projects, running over 700 codes
- Great community interest in burst buffer access, ~30 proposals received
- Support 13 applications actively
  - 6 new effort from NERSC
  - 7 already had LBNL or NERSC staff involved.
- 16 applications not supported by NERSC staff, but do have early access to Cori Phase 1 and the BB.

BURST BUFFERS ARE HERE TO STAY

Multiple approaches to in-system storage and how to use it in upcoming Trinity and CORAL procurements

- LANL/Sandia: Trinity (2016)
  - Similar architecture to NERSC/Cori, dedicated burst buffer nodes
- ORNL: Summit (2018)
  - 800 GiB NVRAM per compute node
- LLNL: Sierra (2017)
  - 800 GiB NVRAM per compute node
- ANL: Theta (2016)
  - 128 GiB SSD per compute node
- ANL: Aurora (2018)
  - NVRAM per compute node and SSD burst buffers

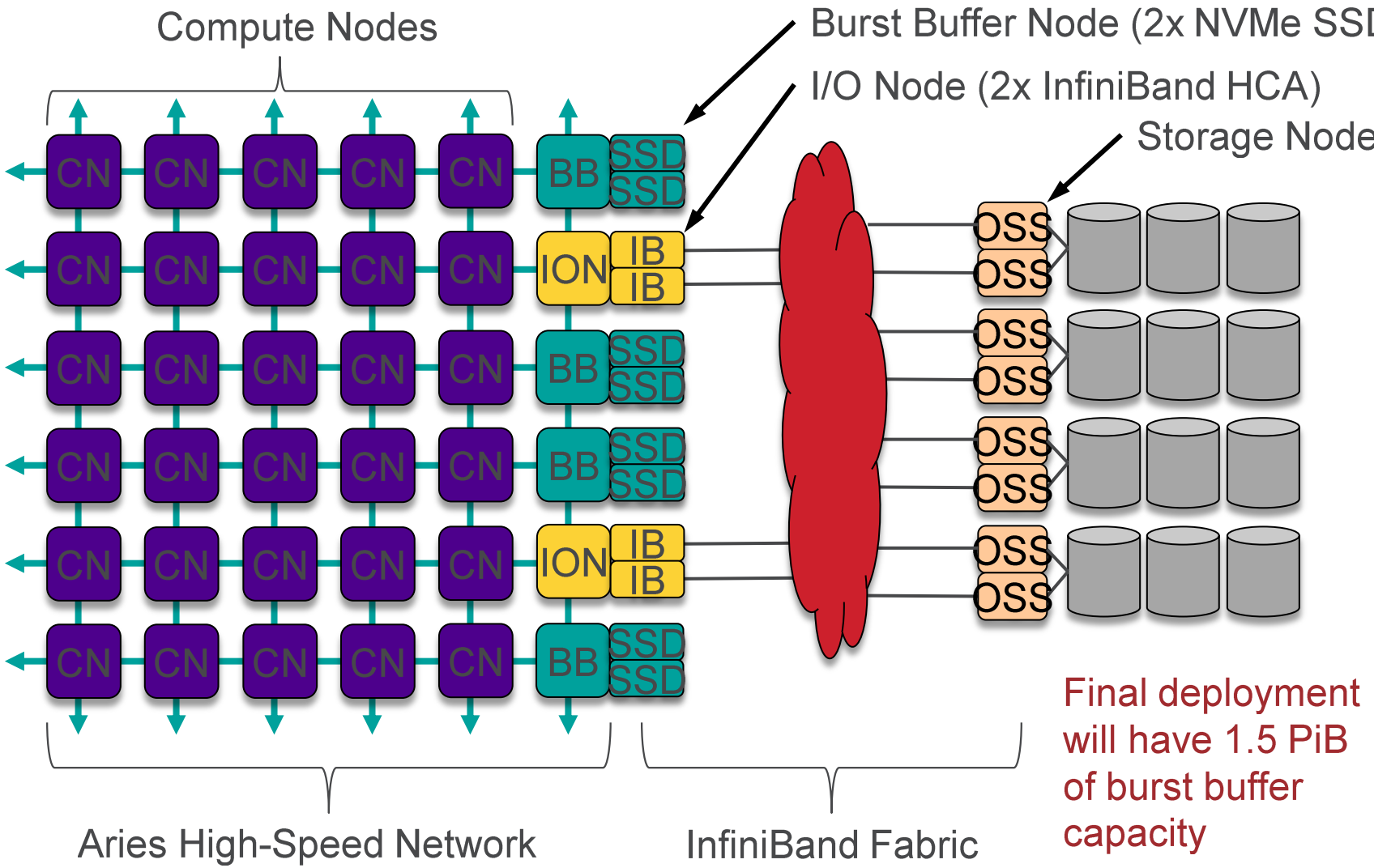
HOW WILL APPLICATIONS USE BURST BUFFERS?

- At a fundamental level it is just another file system
  - Use the same code and same libraries to access it that you normally would
  - Initial adoption will require minimal (if any) change to applications
  - But the performance could change radically!
- Job submission changes:
  - Are burst buffers provisioned per job?
  - Does data need to be explicitly staged to and from parallel file system?
- Will there be any changes to recommended strategies and best practices?
  - This is still evolving, no hard and fast rules yet

CORI BURST BUFFER SYSTEM: THE HARDWARE

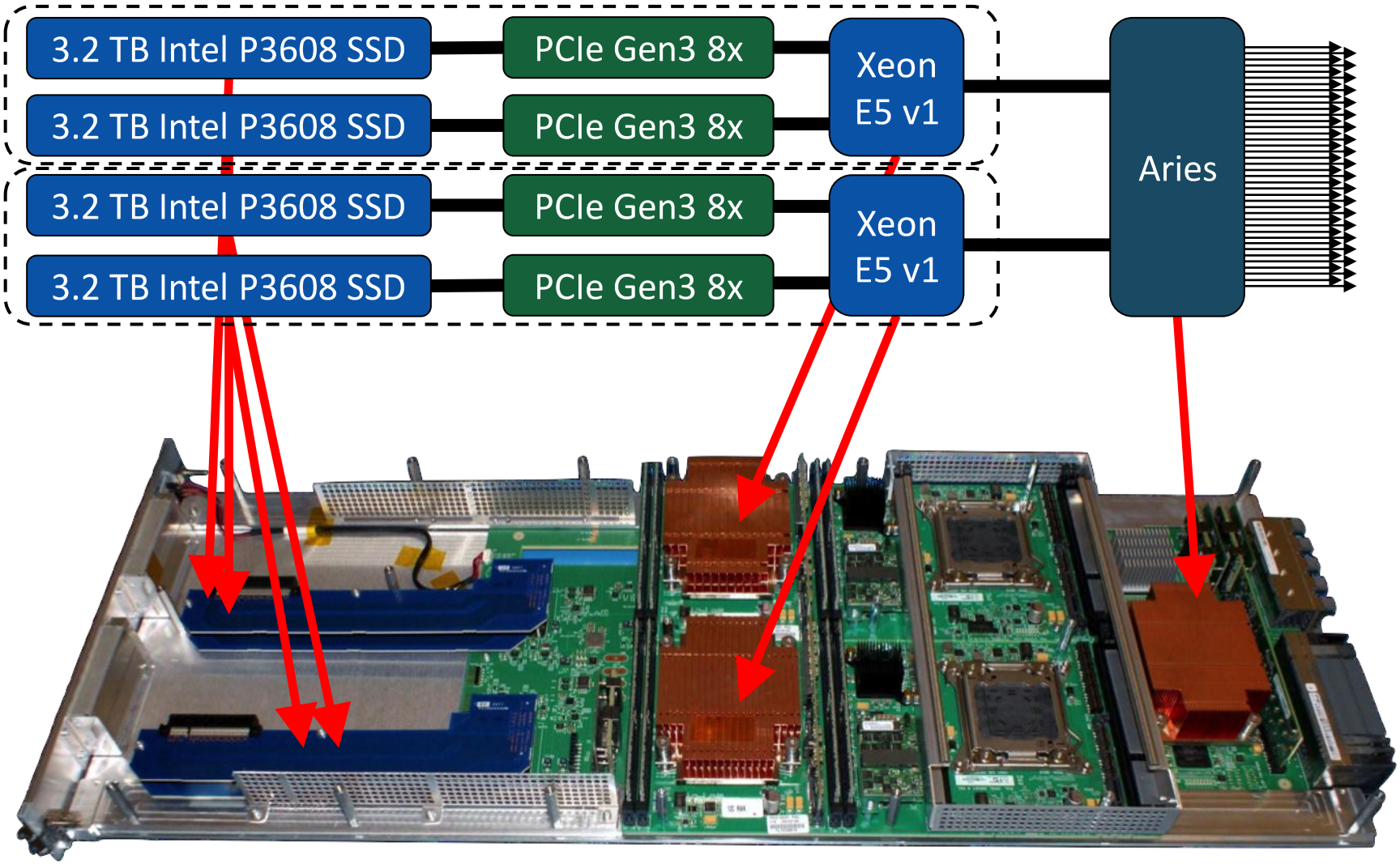
THANK YOU TO KATIE ANTYPAS, GLENN LOCKWOOD, AND WAHID BHIMJI OF NERSC FOR CONTRIBUTING MATERIAL TO THIS SECTION!

BURST BUFFER ARCHITECTURE CONCEPT



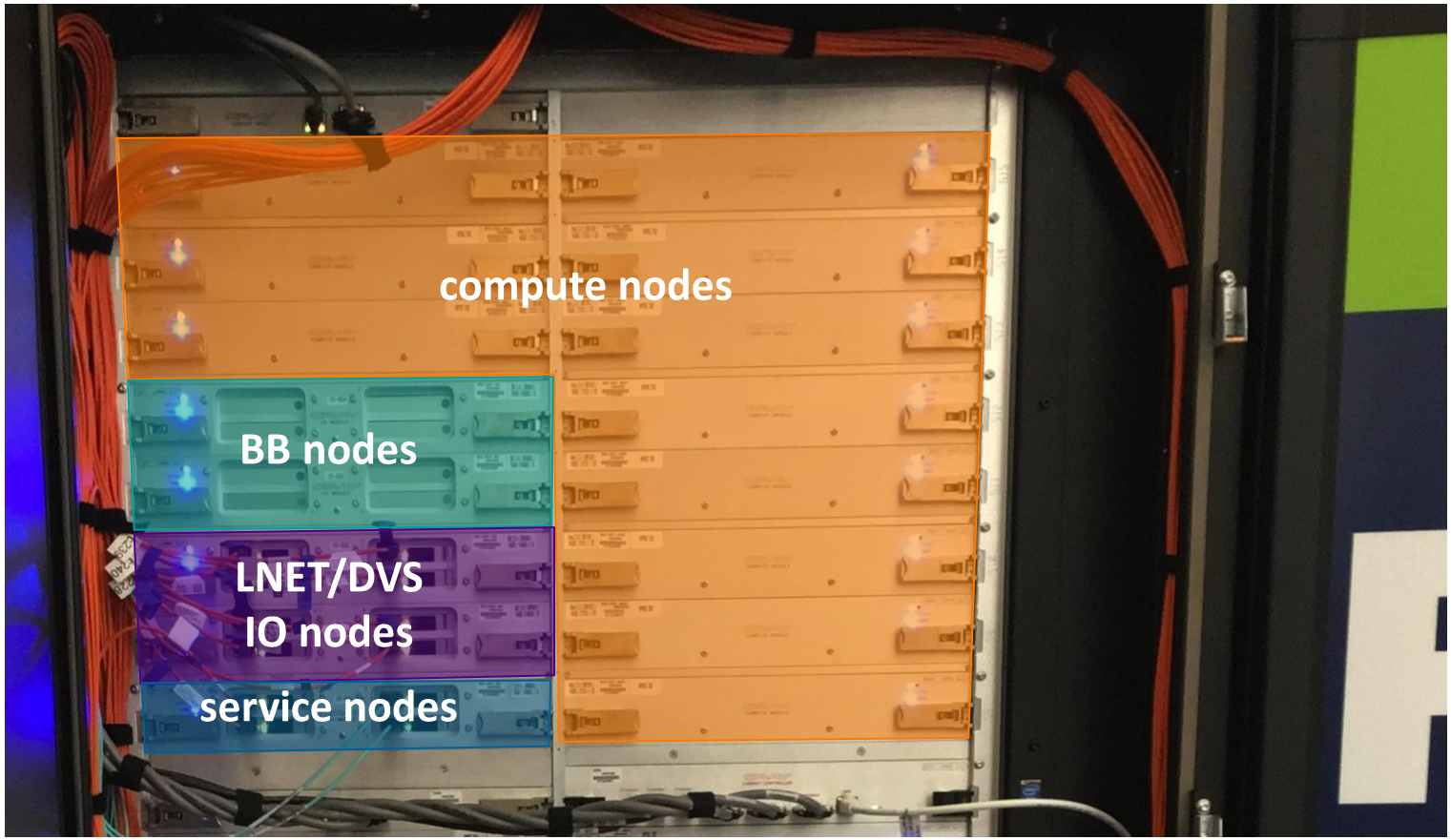
Final deployment will have 1.5 PiB of burst buffer capacity

Burst Buffer Blade = 2xNodes



Burst Buffer Architecture Reality

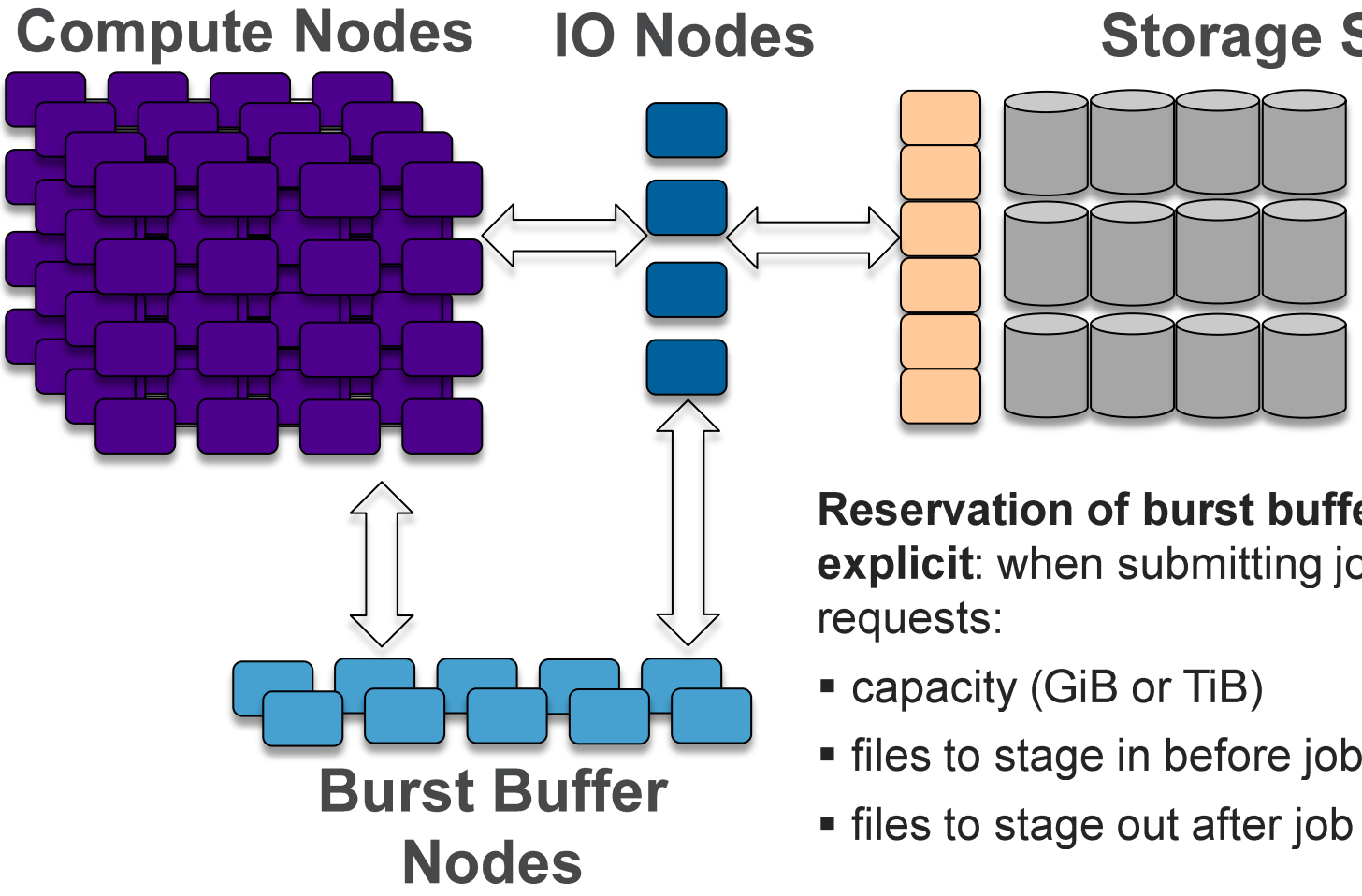
BB nodes scattered throughout HSN fabric
2 BB blades/chassis (12 nodes/cabinet) in Phase I



CORI BURST BUFFER SYSTEM: THE USAGE MODEL

THANK YOU TO KATIE ANTYPAS, GLENN LOCKWOOD, AND WAHID BHIMJI OF NERSC FOR CONTRIBUTING MATERIAL TO THIS SECTION!

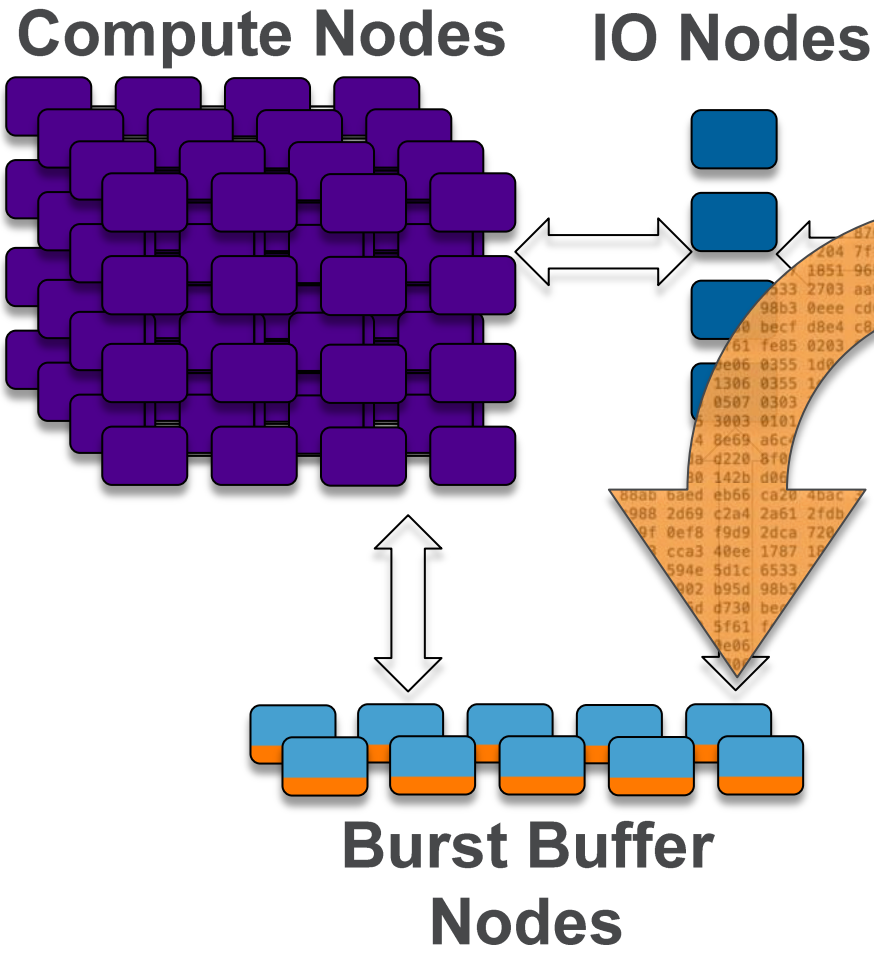
CORI'S DATA PATHS



Reservation of burst buffer capacity is explicit: when submitting job, the ***user*** requests:

- capacity (GiB or TiB)
- files to stage in before job starts
- files to stage out after job finishes

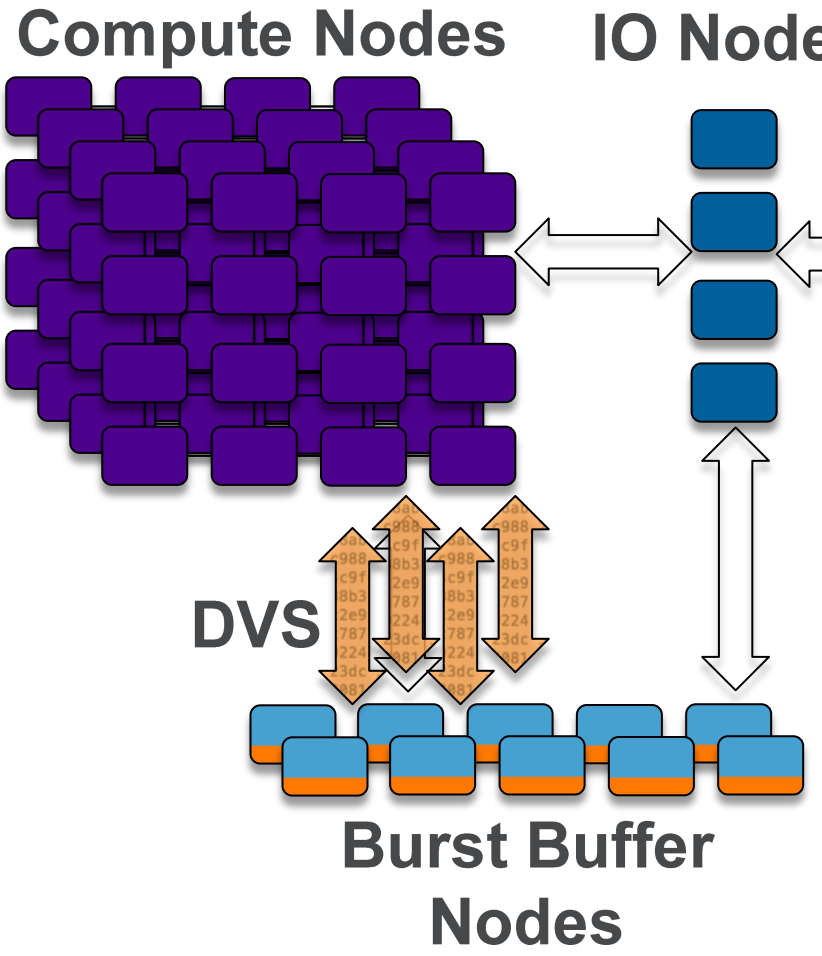
CORI'S DATA PATHS



Before job start, the ***system*** does this automatically:

- Create private parallel file system (DWFS) across parts of multiple BB nodes
- Pre-load user data into this DWFS

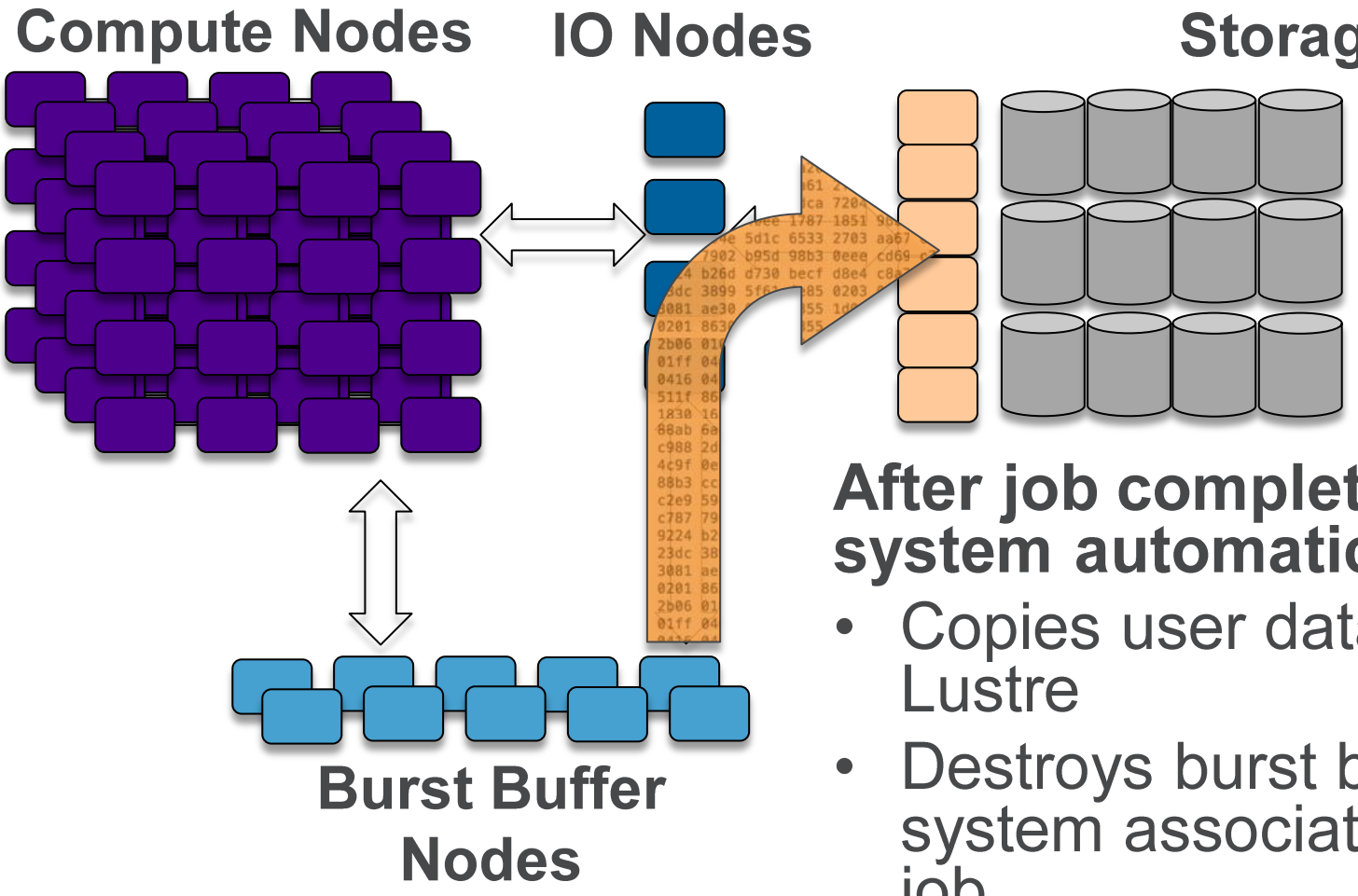
CORI'S DATA PATHS



At job runtime:

- The system automatically mounts the burst buffer file system on the job's compute nodes
- The user's application interacts with DWFS via standard POSIX I/O or other libraries

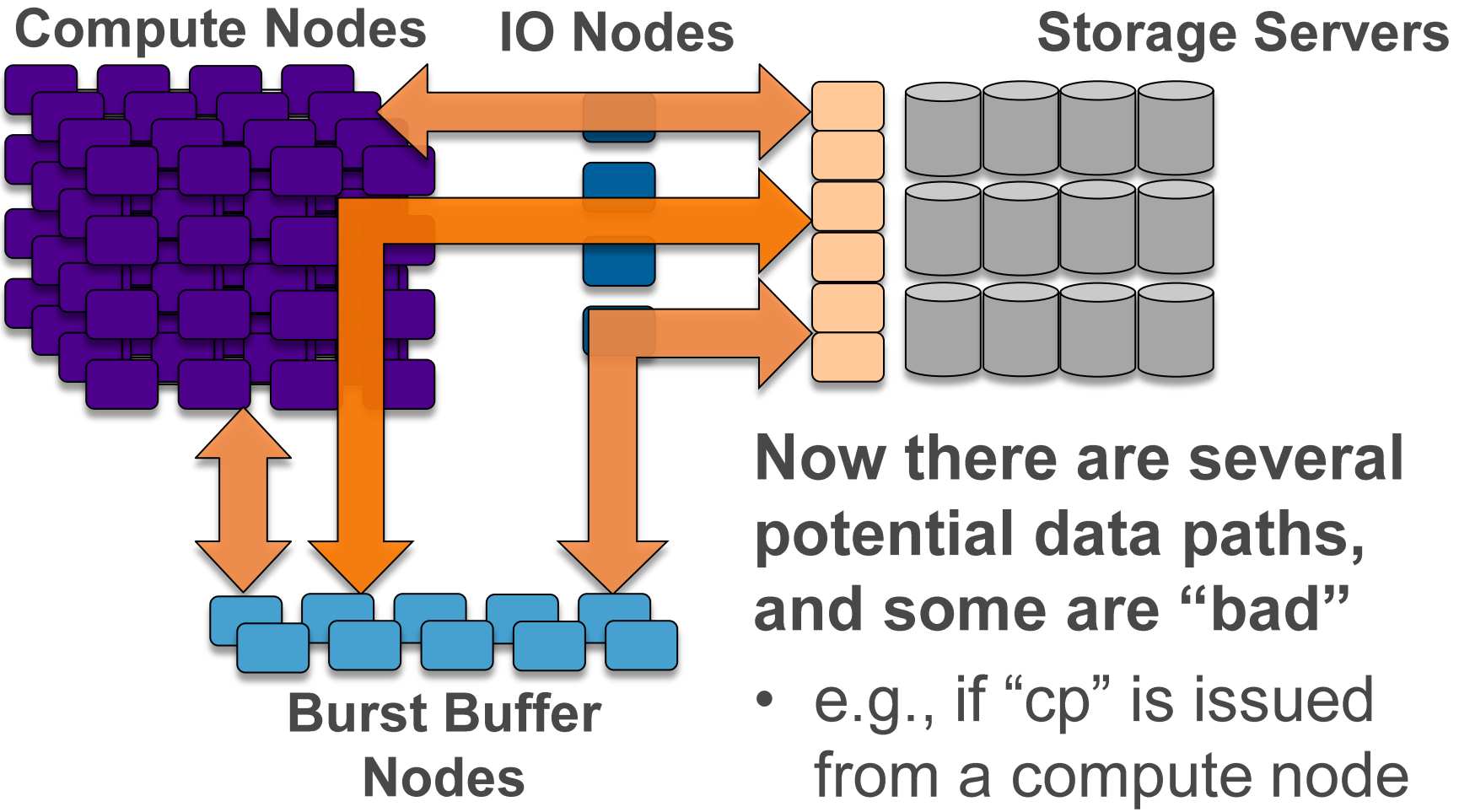
CORI'S DATA PATHS



After job completes, the system automatically:

- Copies user data back to Lustre
- Destroys burst buffer file system associated with job

CORI'S DATA PATHS



Now there are several potential data paths, and some are "bad"

- e.g., if "cp" is issued from a compute node

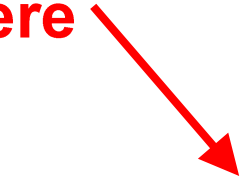
CORI BURST BUFFER SYSTEM: HOW TO USE IT IN PRACTICE

THANK YOU TO KATIE ANTYPAS, GLENN LOCKWOOD, AND WAHID BHIMJI OF NERSC FOR CONTRIBUTING MATERIAL TO THIS SECTION!

Burst Buffer Software

Non-recurring Engineering (NRE) arrangement with Cray (and SchedMD for SLURM WLM integration). Software in Stages:

we are here



Stage 0

Static mapping of compute to BB node, manual data migration

Stage 1

Striping, per-job and persistent allocations; staging; WLM Integration

Stage 2

Transparent caching mode

Stage 3

In-transit processing and filtering

SUPPOSE I HAVE ACCESS - NOW WHAT?

- Burst buffers will eventually be general access, but for now you have to get permission to use them
- Add parameters to your existing job script to request burst buffer capacity and configuration
- Adjust your application parameters to us a different directory than usual for I/O
- Fun and profit?

JOB SCRIPT EXAMPLE

```
#!/bin/bash
#SBATCH –p regular –N 10 –t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs
type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
  --outdir=$DW_JOB_STRIPED/outputs
```

This is a normal Cori job script except for #DW primitives:

- ‘type=scratch’ duration just for compute job (not ‘persistent’)
- ‘access_mode=striped’ – visible to all compute nodes (not ‘private’) and striped across multiple BB nodes
- stage_in and stage_out cause the system to copy data in and out of burst buffer automatically in job setup and tear down

THANK YOU!

THIS CONCLUDES “BURST BUFFERS: A NERSC CASE STUDY”

ON DECK: BREAK TIME!

THEN: “BUILDING AN I/O API”