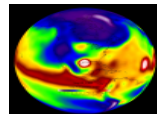
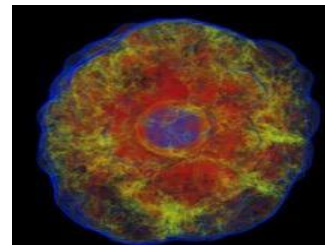
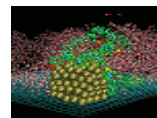
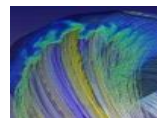
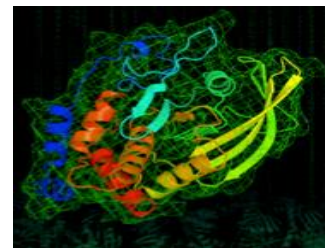
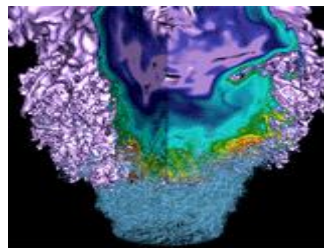


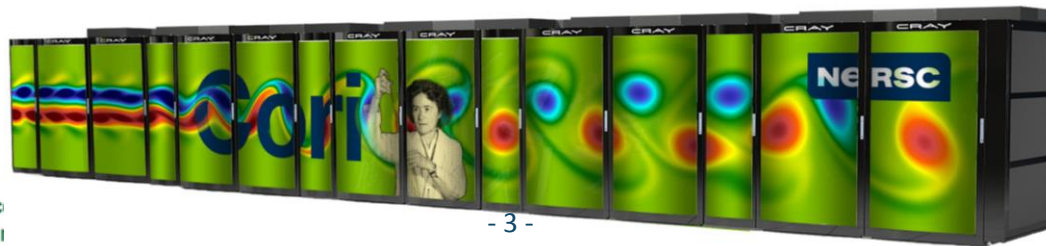
Extreme I/O on HPC for HEP using the Burst Buffer at NERSC



Wahid Bhimji, Debbie Bard, Kaylan Burleigh, Chris Daley, Steve Farrell, Markus Fasel, Brian Friesen, Lisa Gerhardt, Dave Paul, Jeff Porter, Peter Nugent, Jialin Liu, Vakho Tsulaia

- **Introduction to NERSC and our ‘Burst Buffer’**
 - Hardware and Software Architecture
- **HEP Use Cases**
 - ALICE
 - ATLAS
 - Tractor (DESI)
 - H5Boss
- **Conclusions**

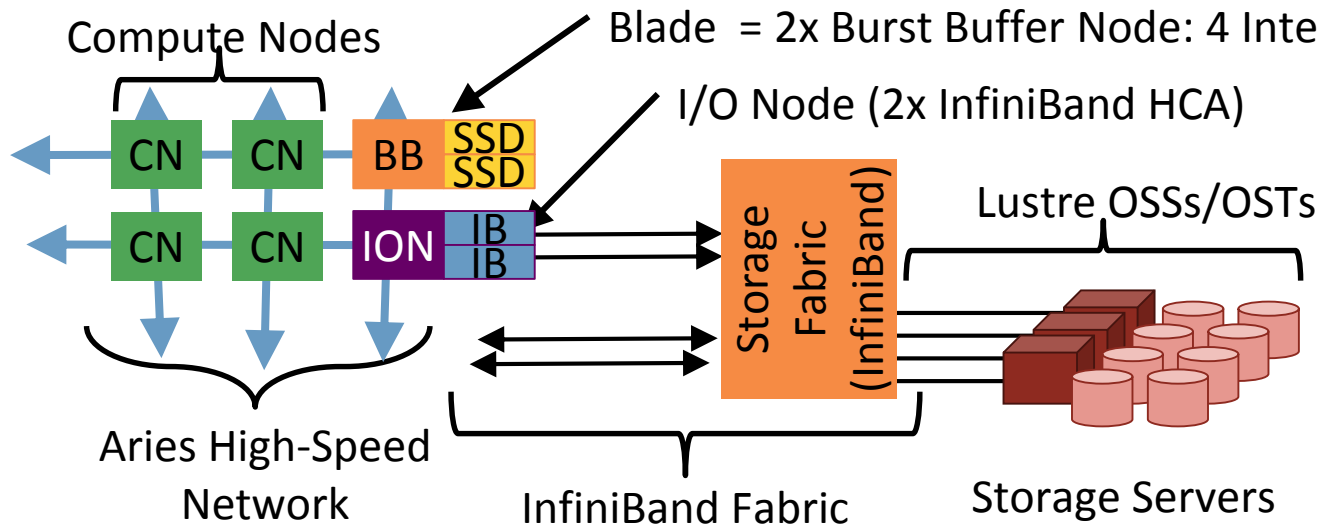
- **NERSC at LBL, production HPC center for US Dept. of Energy**
 - >7000 diverse users across science domains including many outside HEP
- **Cori – NERSCs Newest Supercomputer – Cray XC40**
 - Phase 1 (available Nov 2015): 1630 Intel Haswell dual 16-core nodes 128 GB
 - Phase 2 (becoming available now): >9,300 Intel Knights Landing nodes
- **Cray Aries high-speed “dragonfly” topology interconnect**
- **Lustre Filesystem: 27 PB ; 248 OSTs; 700 GB/s peak performance.**



Burst Buffer Concepts

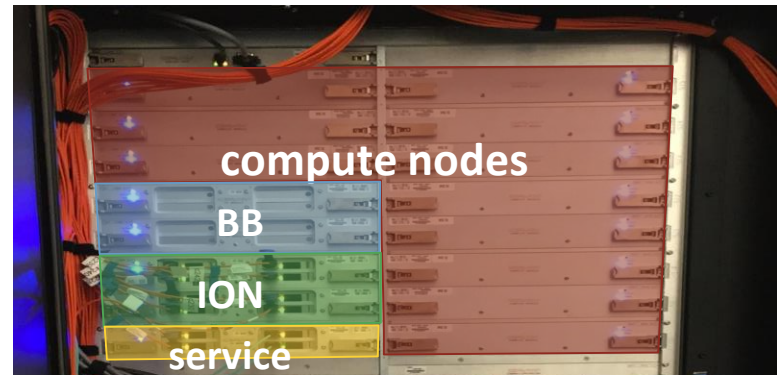
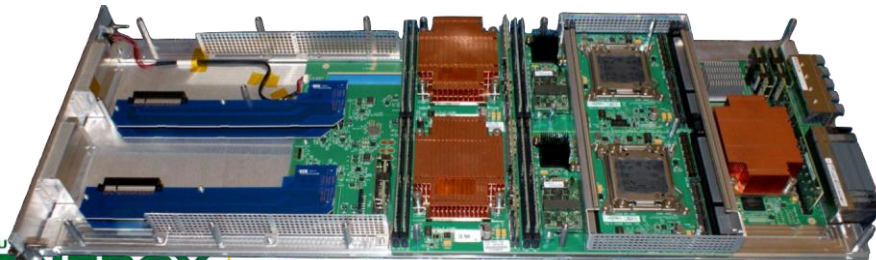
- **HDD performance not increasing sufficiently**
 - HPC centers buying large capacity parallel filesystems to get bandwidth
 - Huge POSIX filesystems don't scale
 - Actual bandwidth demands comes in 'bursts'
 - For bandwidth SSD is cheaper than HDD
- **Some applications (including experimental HEP) have I/O patterns that better match SSD than disk**
- **Use NVRAM-based 'Burst Buffer' (BB) as intermediate layer**
 - Handle I/O spikes without a huge PFS (stage to PFS asynchronously)
 - Underlying media supports challenging I/O
 - Software for filesystems- on-demand - scales better than large POSIX PFS

Nersc Burst Buffer Architecture



Phase 1:
920TB on 144
BB nodes
Now being
doubled
for Phase 2

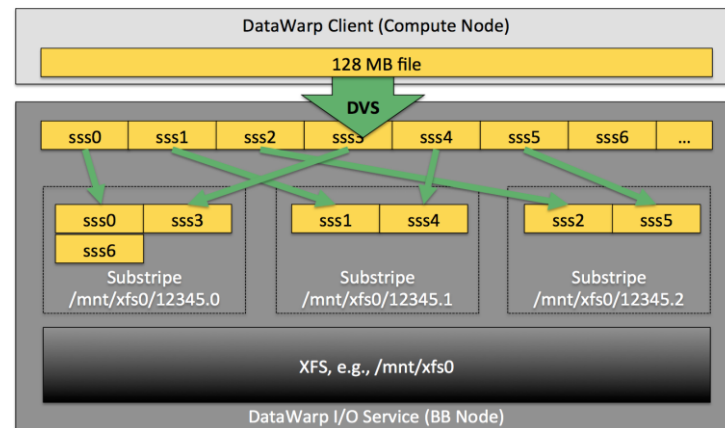
Burst Buffer Blade:



Software

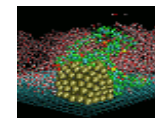
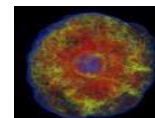
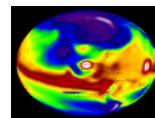
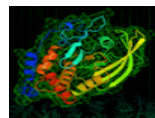
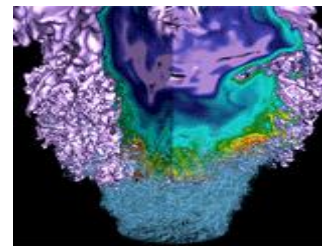


- Cray *DataWarp* software (integrated with SLURM WLM) allocates storage to users per-job (or ‘persistent’ for multiple jobs)
- Users see a POSIX filesystem on-demand, striped across nodes
- Can specify data to stage in/out from Lustre while job is in queue
- Benchmark Results with IOR - 140 BB Nodes
 - 1120 Compute nodes: 4 ranks/node
 - Bandwidth: 8 GB block-size 1MB transfers
 - IOPS tests: 1M blocks 4k transfer



	IOR Posix FPP		IOPS	
	Read	Write	Read	Write
Peak	905 GB/s	873 GB/s	12.6 M	12.5 M

HEP Use Cases

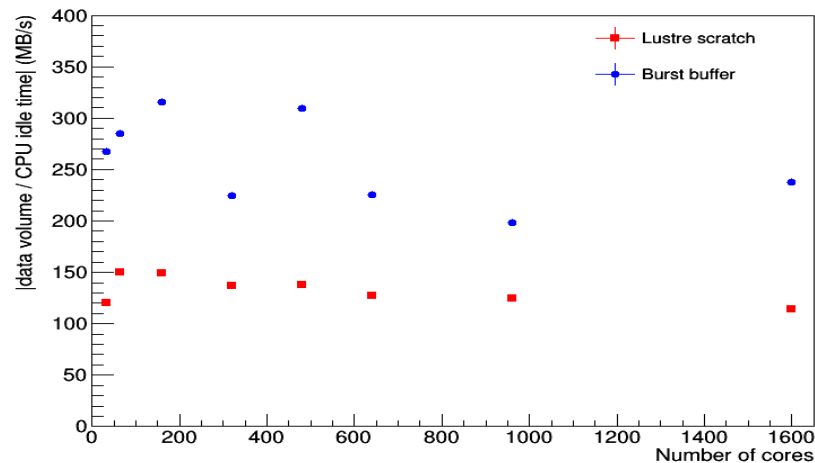


See also [Cray Users Group Paper](#) for more use-cases beyond HEP

Markus Fasel, Jeff Porter

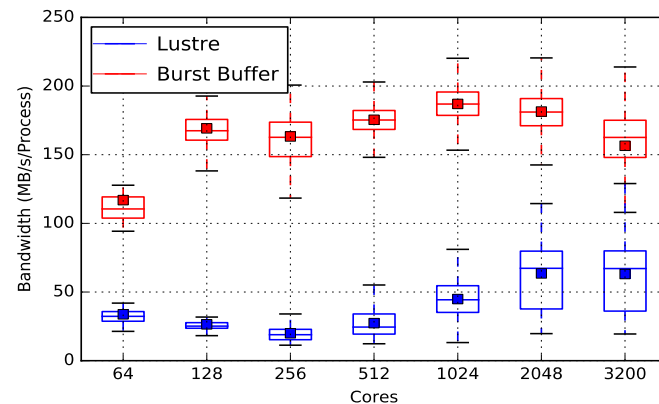
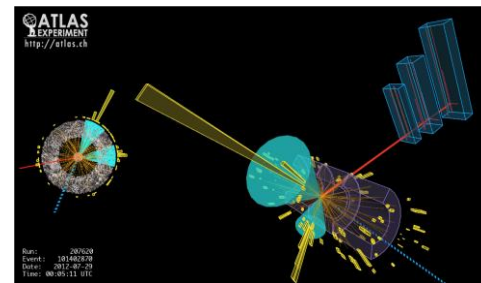
- I/O intensive analysis:
 - Running on ESDs: Events after reconstruction
 - Minimal compute –fill histograms
 - Serial application
- Data: “Pb-Pb 2015 full reco pass”, 1 run
 - ~10 TB data, 16k files
 - 10 files per compute core
- 26TB BB – 126 servers

- CPU time/wall time > 90%
- CPU ‘idle time’ = wall – cpu
- Effective average ‘bandwidth’ data read / idle time per process



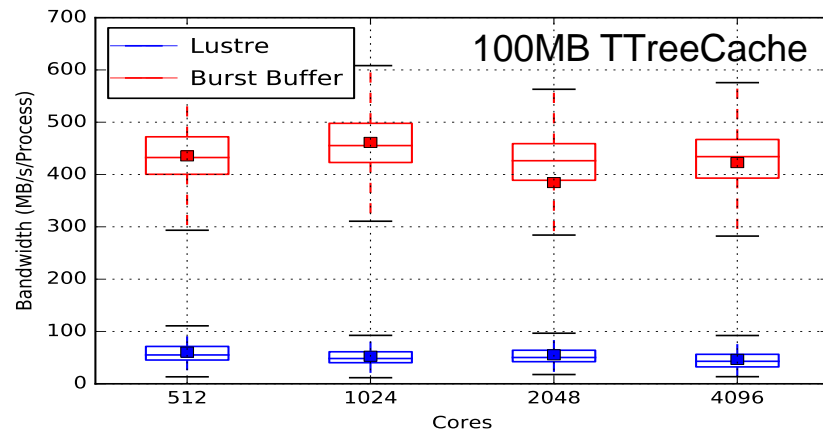
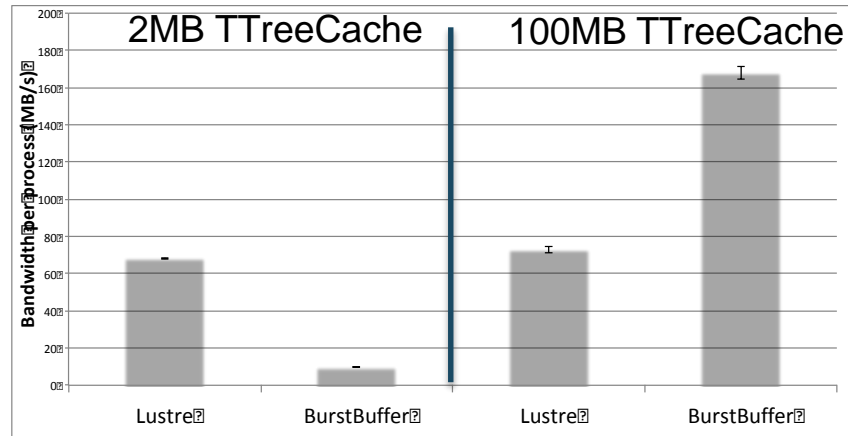
Vakho Tsulaia, Steve Farrell, WB

- **ATLAS now making use of HPC resources**
 - Running simulation – not I/O intensive workloads
 - Test here I/O heavy ‘derivation’ and analysis
- ‘Derivation’: filtering ‘xAOD’ ROOT-based format to analysis-specific files
 - Run in ATLAS ‘AthenaMP’ framework
- BB allocation 2TB to 20TB
- Faster read times on BB
 - ‘Bandwidth’ = data read / IO time measured via `xAOD::ReadStats` ([TVirtualPerfStats](#))



Atlas Analysis

- ‘QuickAna’ ROOT/python app
- Initial study to improve read I/O
 - Poor initial BB performance
 - Increase ROOT ‘TTreeCache’ - 100M
 - **Less reads → 17x boost on BB**
 - Also use ‘branch access’ and ‘learning pre-fill’
- Scales well to larger job sizes
 - 50TB dataset: 143 node BB allocation



Tractor and DESI

- **Tractor: make 3D map of distribution of galaxies**

- Optical images $\sim 1/3$ of night sky.
Telescopes at Kitt Peak, AZ and Cerro Tololo, Chile
- Astro sources classified χ^2 -fit between models and images
- Combined with source spectra from DESI to infer redshift and distance (2D \rightarrow 3D Map (2D image + distance using spectra))



- Python application
- Files in FITS format
- Runs 1 process / node with 32 threads
- 32 TB BB used in tests shown here

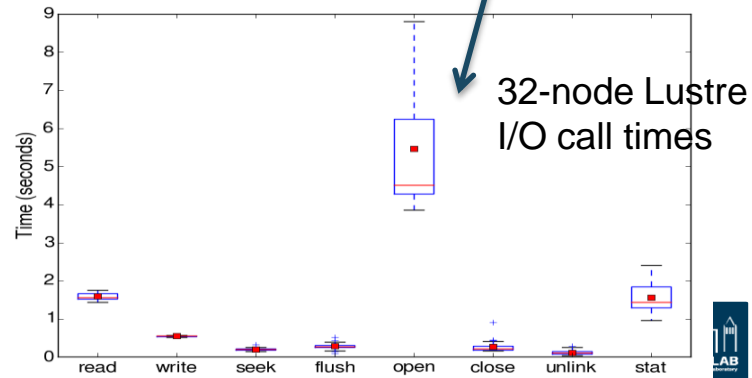
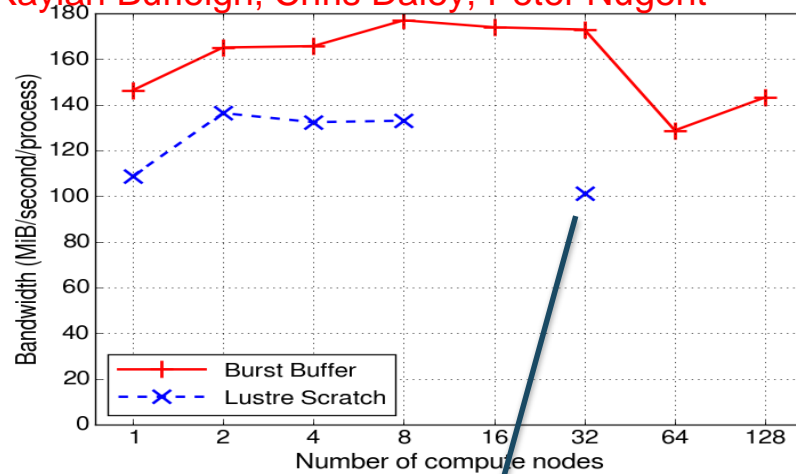
- **Detailed profiling:**

- strace, ipm and custom code
- Plot here data moved by application via freads and fwrites
- Actual I/O is greater due to prefetching by glibc

- **I/O time dominated by metadata (inc. python loads)**

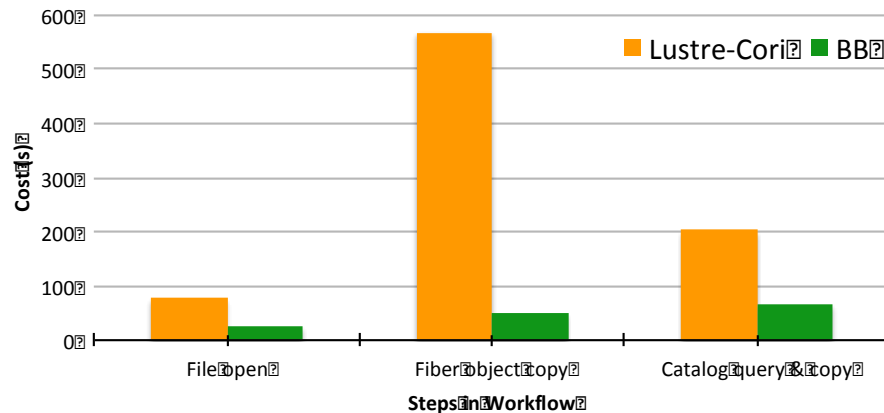
- **Variation on Lustre - maybe due to other user impact**

Kaylan Burleigh, Chris Daley, Peter Nugent



Jialin Liu, Debbie Bard

- BOSS Baryon Oscillation Spectroscopic Survey – from SDSS
- Perform typical randomly generated query to extract small amount of stars/galaxies from millions
- Workflows involve 1000s of file open/close and random and small read/write I/O
- Run on final release of [SDSS-III](#) complete BOSS dataset
 - 2393 HDF5 files - total ~3.2TB

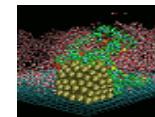
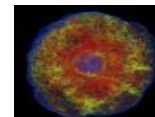
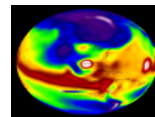
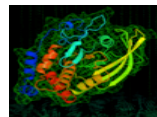
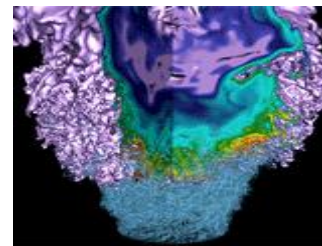


- 4.4 TB Burst Buffer - 22 nodes
- Lower I/O times on Burst Buffer
- 5.5x speedup for entire workflow

Conclusions

- Burst Buffer – A new approach to dynamically allocate filesystems striped across high-performance SSDs
 - Now in production at NERSC
- Demonstrated here for experimental HEP Workflows
 - Substantially improves I/O over comparable Lustre filesystem
 - Variety of use-cases from NP, HEP, Cosmology
- I/O is not (now) a significant barrier to these projects
 - Can run their most I/O intensive workflows on Cori
- Further improvements to performance and usability of NERSC
Burst Buffer coming soon...

Extra slides: for those who just can't get enough



Coming performance improvements



1. DVS client-side caching

- Lustre has client-side caching, currently DVS (used for BB) does not
- Essential for small sequential Read/Write transfers and re-reads
- Expected later this year

2. Smaller granularity

- Amount of space allocated on each BB node
 - Previously couldn't be less than 200G
 - Users had to request larger space to get striped performance
 - Now have the ability to reduce this – testing lower values

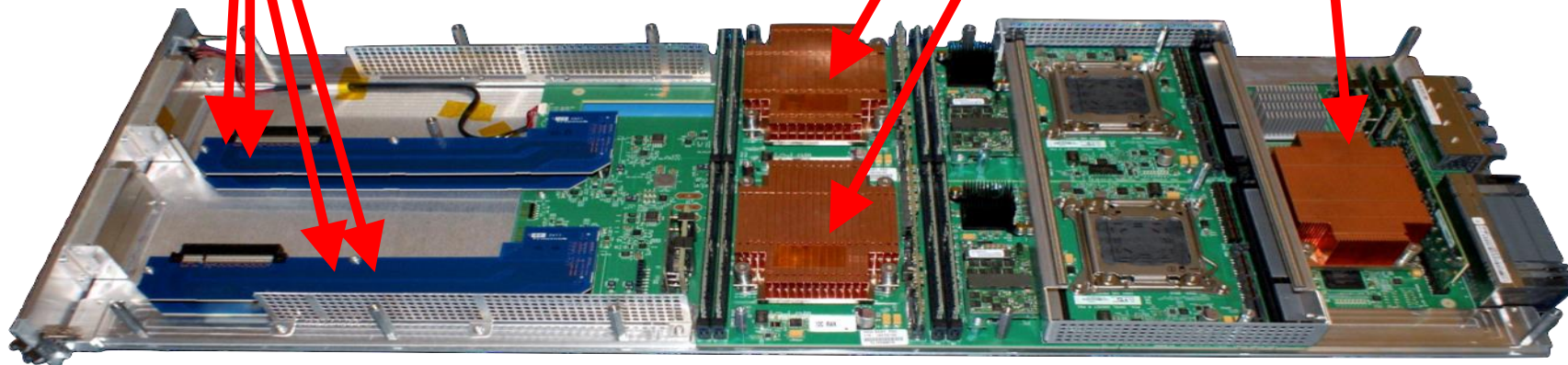
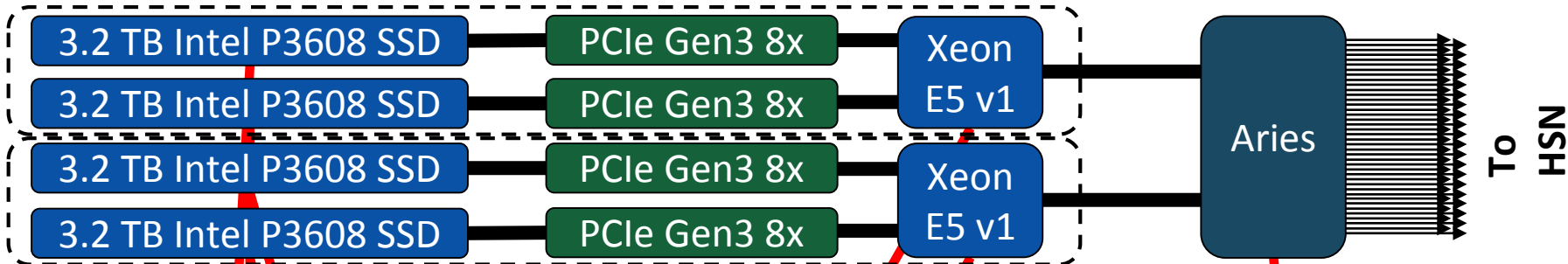
3. Transparent caching

- Allows user to specify directory in Lustre and blocks are cached as used
- Software now available - but undergoing testing

4. Twice as much Burst Buffer! (and therefore ~2x bandwidth!)

Working with Cray to improve BB performance out-of-the-box for all use-cases

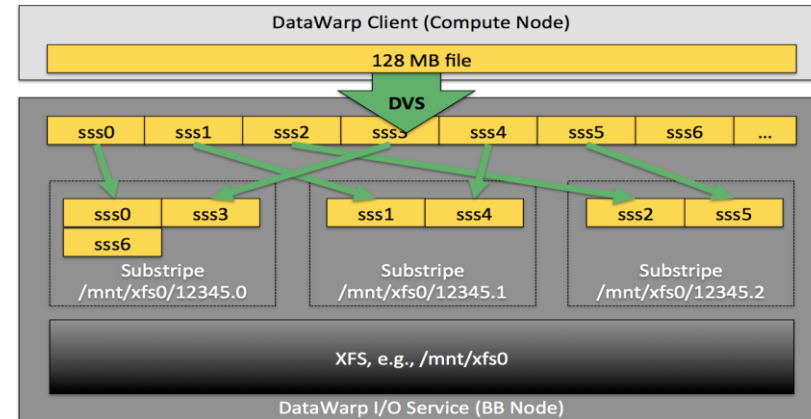
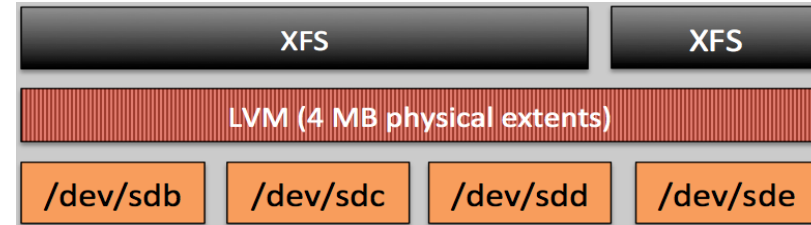
Burst Buffer Blade = 2xNodes



DataWarp Filesystem layers



- Logical Volume Manger (LVM) group SSDs into one block device
- An XFS file system that is created for every Burst Buffer allocation
- Cray Data Virtualization Service (DVS), for communication between DWFS and the compute nodes.
- File written from compute node ends up as (configurable) 8MB chunks, laid out across the three (configurable) substrips on the Burst Buffer node.



Integrated with SLURM WLM



```
#!/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
```

Example illustrates:

- ‘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
 - Actual distribution across BB Nodes is in units of (configurable) granularity (currently 218 GB at NERSC so 1000 GB would normally be placed on 5 BB nodes)
- Data stage_in before job start and out after

Burst Buffer Software



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

Stage 2

Transparent caching mode

Stage 1

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

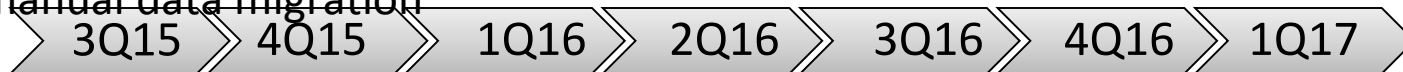
Stage 0

Static mapping of compute to BB node, manual data migration

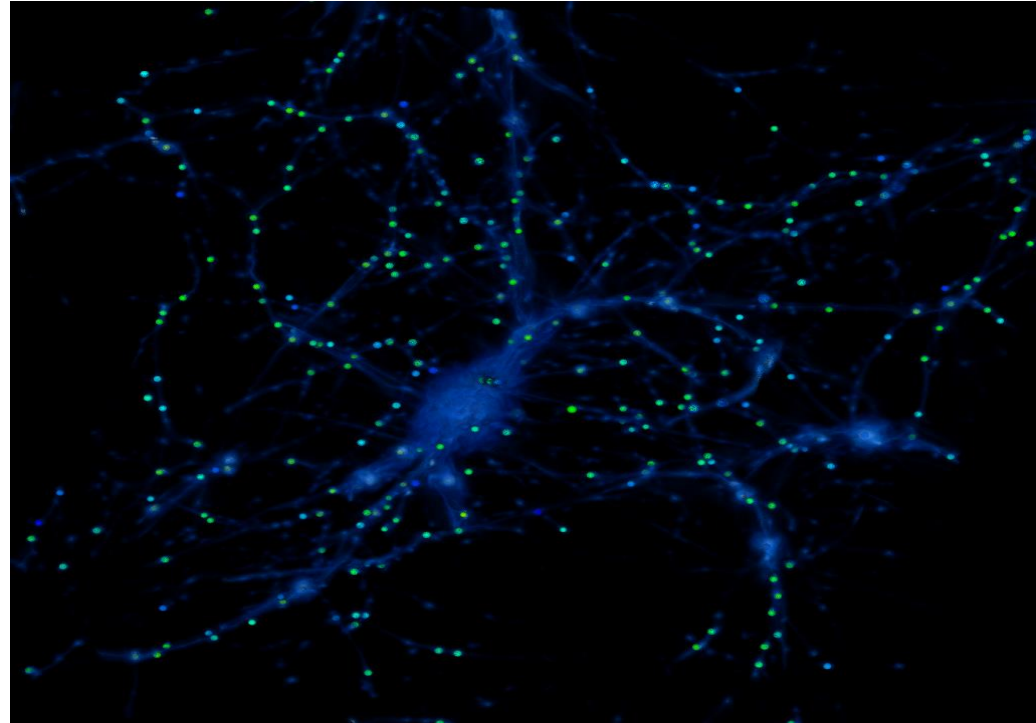
we are here

Stage 3

In-transit processing and filtering

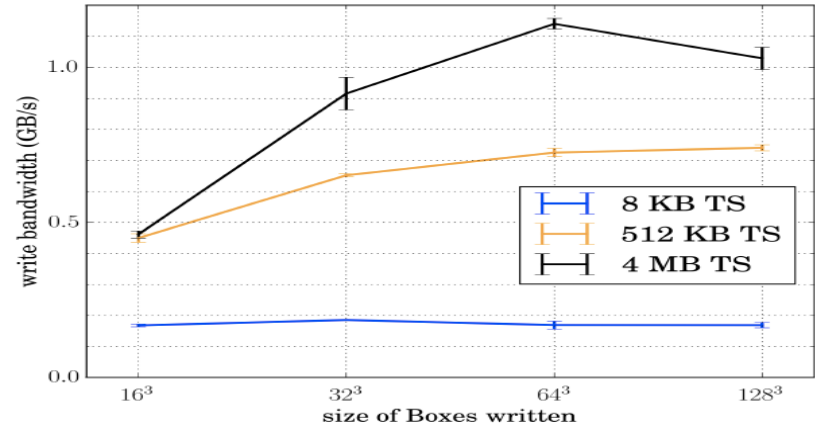


- Nyx cosmological simulation code based on a widely-used adaptive mesh refinement (AMR) library, BoxLib
- Large data files (“plotfiles”) written at certain time steps; checkpoint files also written



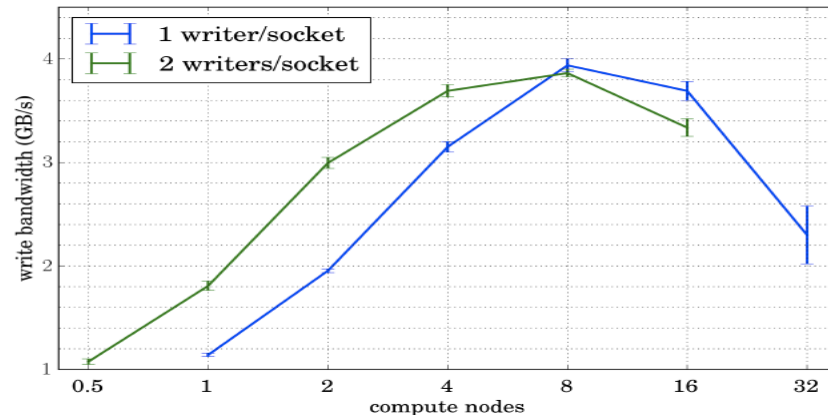
Nyx/Boxlib – Single BB Node

- **Need larger transfer size for good performance**
 - Less of an issue on Lustre
 - No client-side cache in DVS/DataWarp



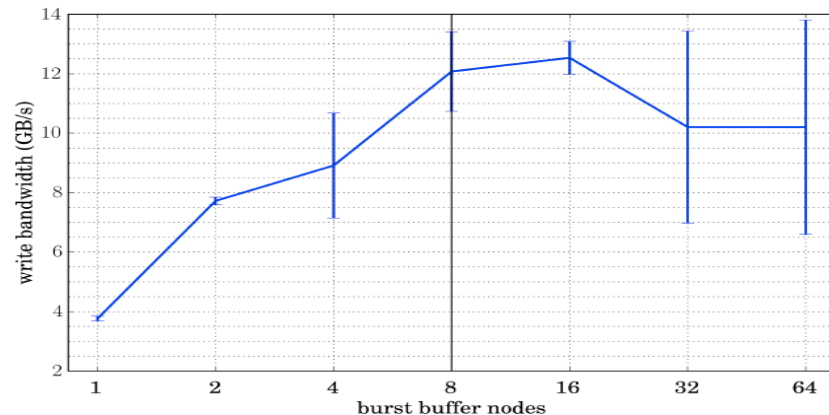
Nyx/Boxlib – Single BB Node

- Need larger transfer size for good performance
- More MPI writers (~16) to approach 4 GB/s



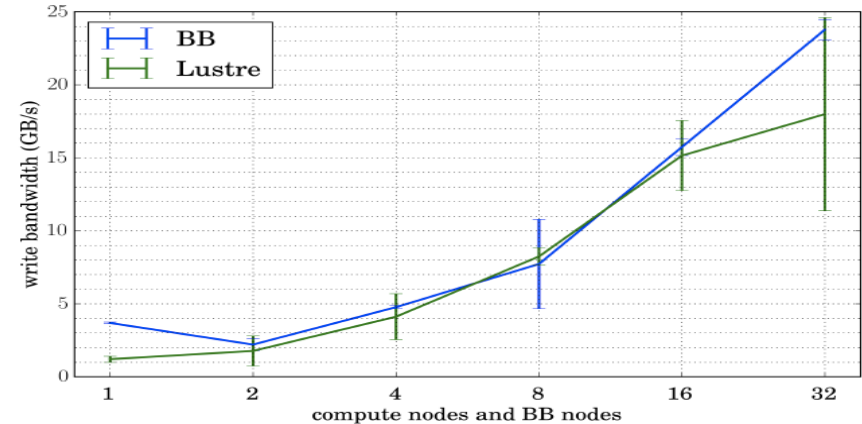
Nyx/Boxlib – Scaling up

- Need larger transfer size for good performance
- More MPI writers (~16) to approach 4 GB/s
- Maximize bandwidth per compute node by adding more BB nodes until 1/1 CN/BB ratio

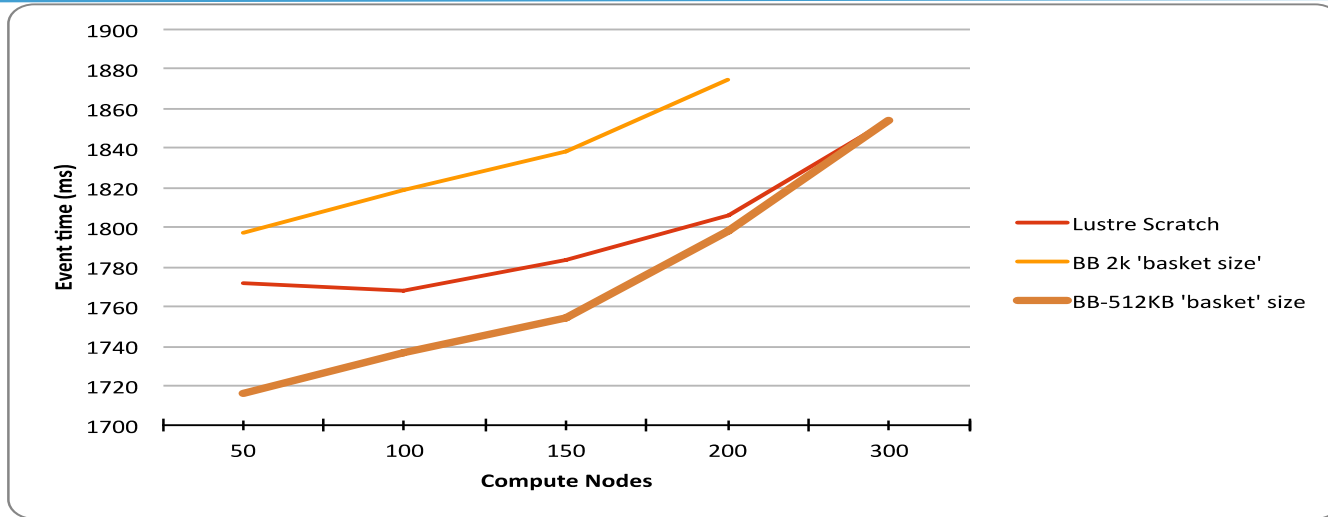


Nyx/Boxlib – Scaling up

- Need larger transfer size for good performance
- More MPI writers (~16) to approach 4 GB/s
- Maximize bandwidth per compute node by adding more BB nodes until 1/1 CN/BB ratio
- 32 CN nodes starts to outperform Lustre



ATLAS: Simulation Payload



- Initial scaling on BB poor
- Increase ROOT 'basket size' from 2k to 512k to
- increase transaction size
- Keep log files on Lustre
- Then scales to >300 nodes

Jialin Liu, Debbie Bard

- Typically produce a single FITS data file per object observed
 - Millions of files now -> Hundreds of millions in future
- This project uses instead a HDF5 data structure:
 - Improve small file metadata problems
 - Allow parallel access via HDF5 IO libraries.
- *H5boss* – python package: parses query, opens source HDF5 files, uses ‘subset’ function to create single shared output HDF5 file
 - Perform two independent copies – whole fiber including wavelength and exposures; only specific two in catalog