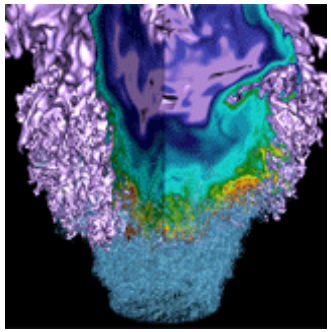
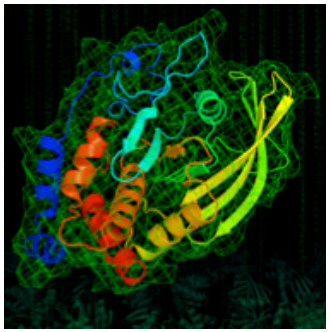
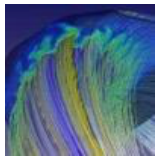
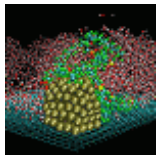
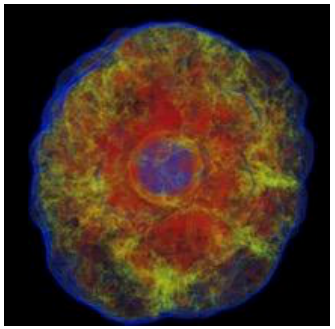
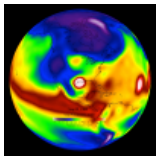
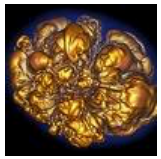
Accelerating Science with the NERSC Burst Buffer Early User Program

Wahid Bhimji, Debbie Bard, Melissa Romanus, David Paul, Andrey Ovsyannikov, Brian Friesen, Matt Bryson, Joaquin Correa, Glenn K. Lockwood, Vakho Tsulaia, Suren Byna, Steve Farrell,
Doga Gursoy,Chris Daley,Vince Beckner,Brian Van Straalen, Nicholas J Wright, Katie Antypas, Prabhat

NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Outline

- **Introduction to I/O Hierarchy and Burst Buffers**
- **Architecture and Software of NERSCs Burst Buffer**
- **Overview of Early User Program**
- **Science Use Cases**
  - –Nyx/Boxlib
  - –Chombo-Crunch and VisIt
  - –VPIC I/O
  - –TomoPy
  - –ATLAS
- **Challenges and Lessons Learned**

Why a Burst Buffer?

- **HDD performance not increasing sufficiently**
  - More and more capacity to get required bandwidth
  - The bandwidth demand comes in 'spikes'
- **Huge POSIX parallel filesystems don't scale**
- **For bandwidth HDD/PFS is more expensive than SSD**
- **Some applications have challenging I/O patterns**
  - High IOPS – better match for SSD than spinning disk
- **Use NVRAM-based storage 'Burst Buffer'**
  - Handle I/O bandwidth spikes without increasing size of PFS
  - Underlying Media supports challenging I/O patterns
  - Filesystems on demand scale better than large POSIX PFS
  - Staging to PFS asynchronously

HPC memory hierarchy

Past

On Chip

Off Chip

CPU

Memory (DRAM)

Storage (HDD)

Future

On Chip

Off Chip

CPU

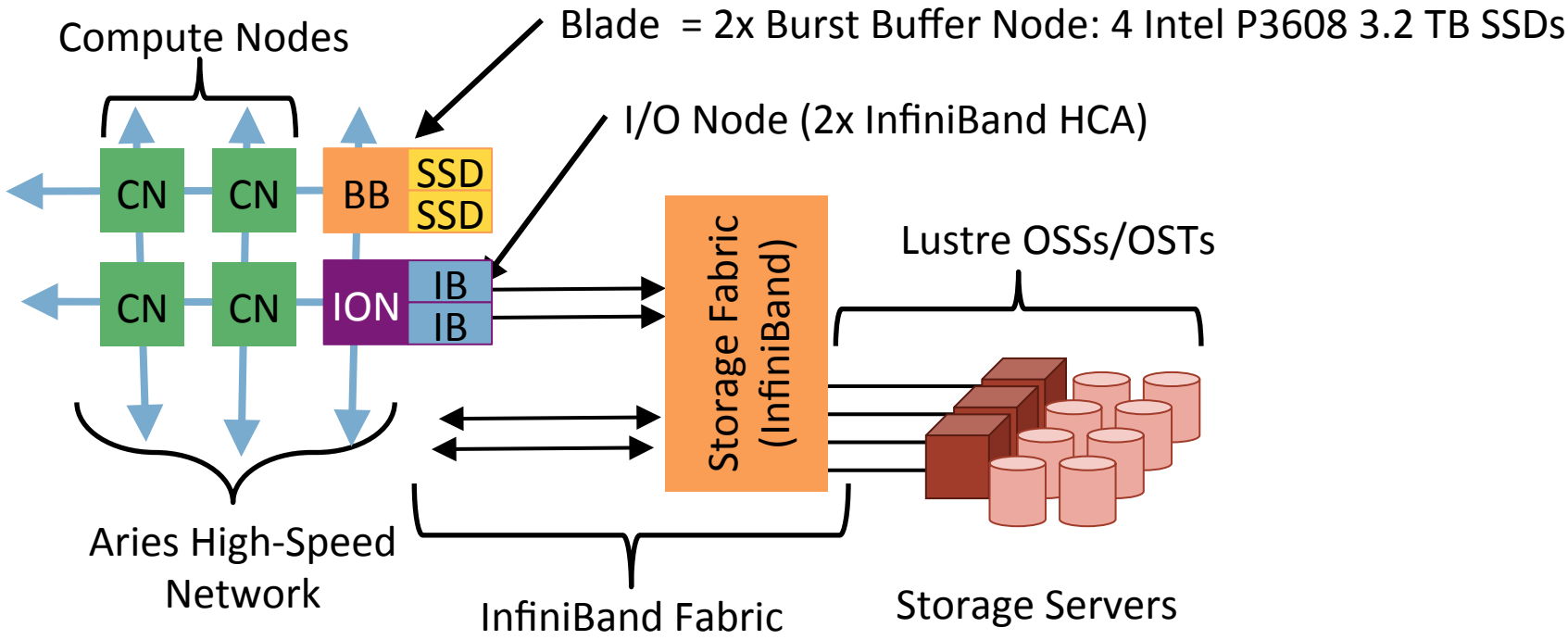
Near Memory (HBM)

Far Memory (DRAM)

Near Storage (SSD)

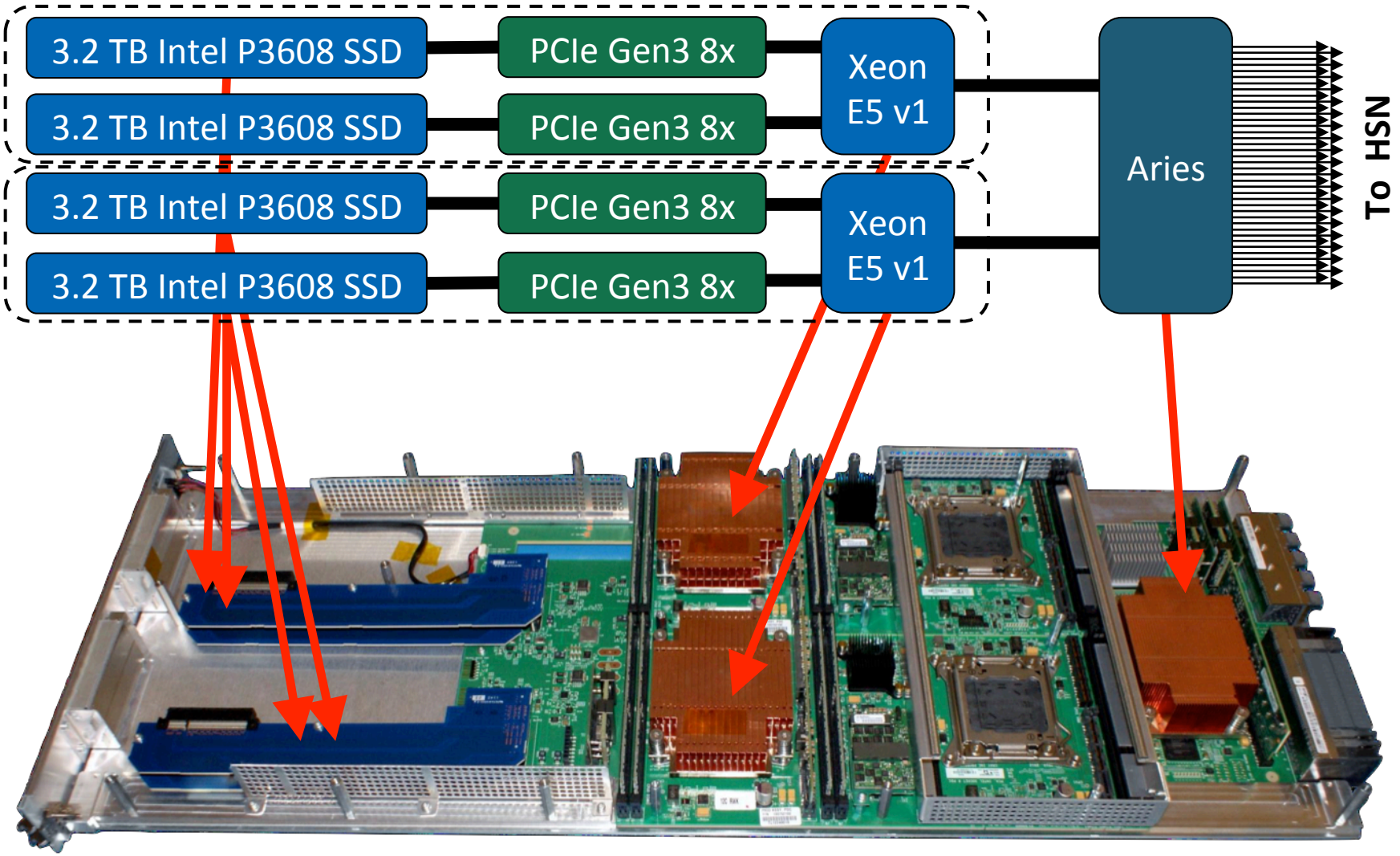
Far Storage (HDD)

Nersc/Cray Architecture



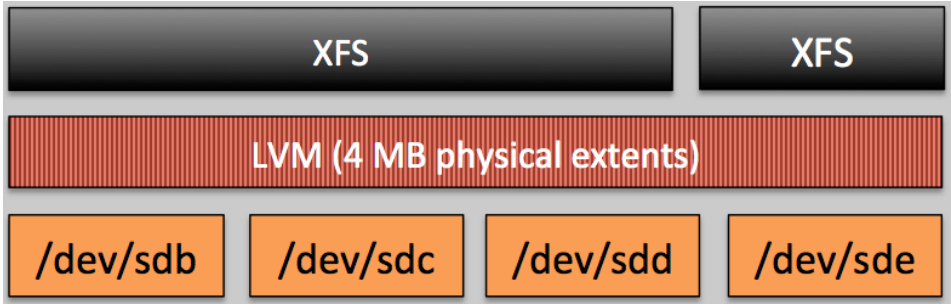
- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or 'persistent').
- Users see a POSIX filesystem
- Filesystem can be striped across multiple nodes (depending on allocation size requested)

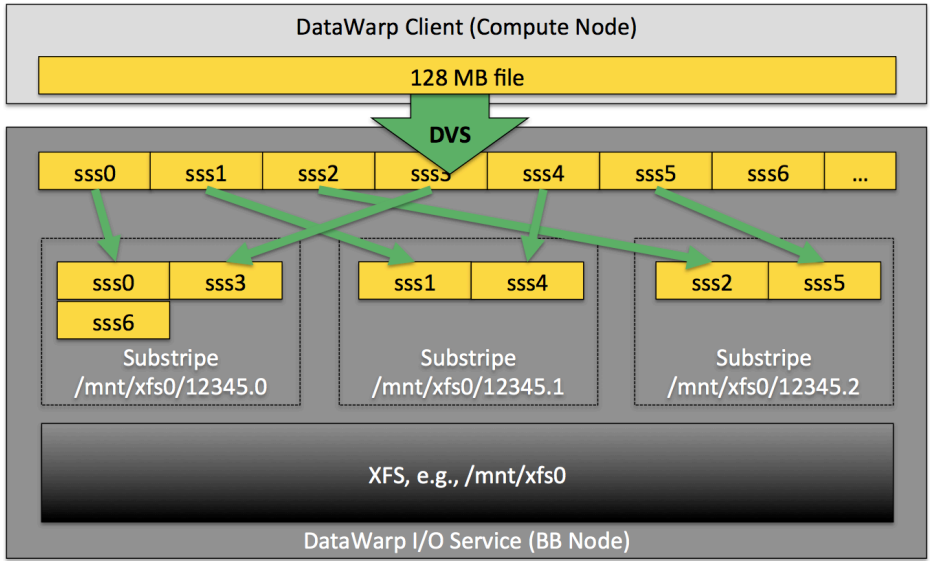
Burst Buffer Blade = 2xNodes



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,
- The DataWarp File System (DWFS), stacked file system provides namespaces (e.g. striped access)
- 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) substripes on the Burst Buffer node.





Integrated with SLURM WLM – easy user interface

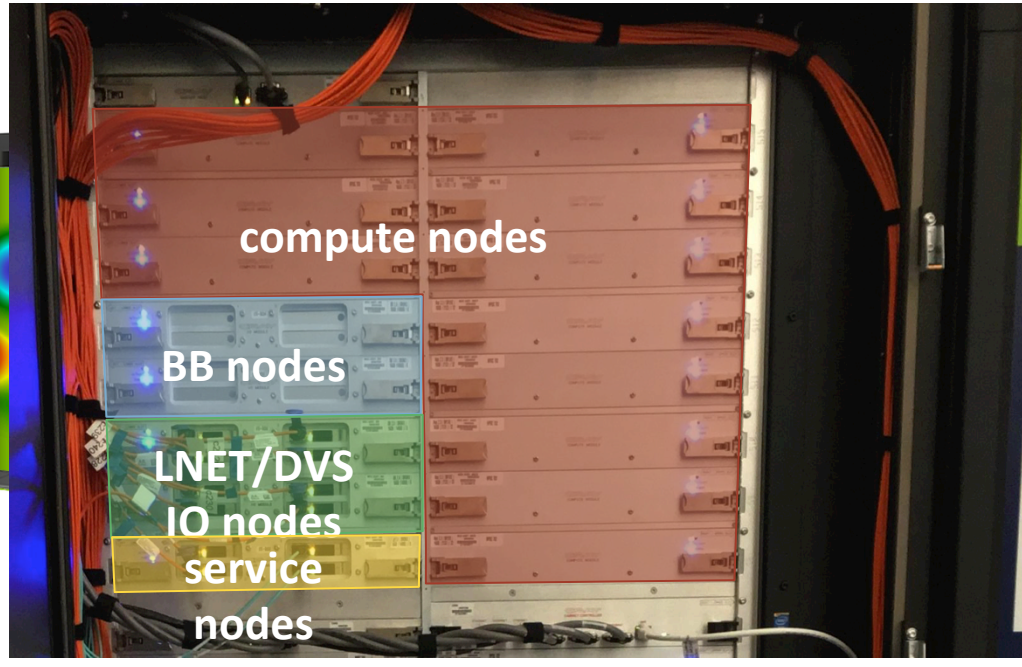
```
#!/bin/bash
#SBATCH –p regular –N 10 –t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs
type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs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

Cori, a Cray XC40 system

- **Cori Phase 1 – partition to support data intensive applications**
  - –1630 Intel Haswell nodes: Two 16-core processors and 128 GB DDR4 /node,
- **Cori Phase 2: >9,300 Intel Knights Landing compute nodes**
- **Lustre Filesystem: 27 PB ; 248 OSTs;700 GB/s peak performance.**
- **Cray Aries high-speed "dragonfly" topology interconnect**
- **Burst Buffer (Phase 1) 920TB on 144 BB nodes**
  - Doubled for Phase 2



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

3Q15 | 4Q15 | 1Q16 | 2Q16 | 3Q16 | 4Q16 | 1Q17

Benchmark Performance

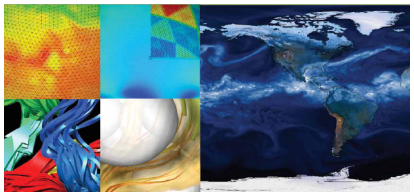
- **Burst Buffer is doing well against benchmark performance targets**
  - Work on-going to improve MPIO shared file write
  - Out-performs Lustre (and we have half the full Phase 2 Burst Buffer and only a fraction of the full Cori compute)

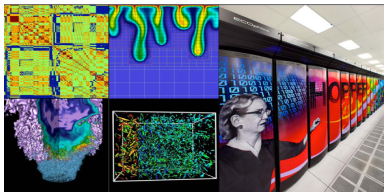
	IOR Posix FPP		IOR MPIO Shared File		IOPS	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Best Measured (140 Burst Buffer Nodes : 1120 Compute Nodes; 4 ranks/node)*	905 GB/s	873 GB/s	803 GB/s	351GB/s	12.6 M	12.5 M
Lustre (peak – 24 OSTs: 930 compute nodes, 4 ranks/ node; 4 MB transfer)	708 GB/s	751 GB/s	573 GB/s	223 GB/s	-	-

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

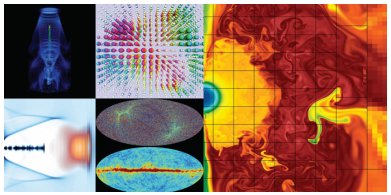
Burst Buffer Early User Program

- NERSC is production HPC Facility for US Dept. of Energy, Office of Science
  - Diverse Users across science domains
  - ~7000 users on more than 700 projects, running over 700 codes
- Call for proposals
  - Award of early use of BB on Cori P1, plus help of NERSC staff.
  - Selection criteria including Scientific merit; Computational challenges; Cover range of BB data features; Cover range of DoE Science Offices.
- Great interest from the community, ~30 proposals received: support 13 actively, others given early access

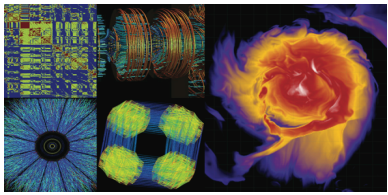




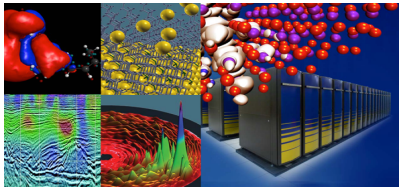
Bio Energy, Environment Computing



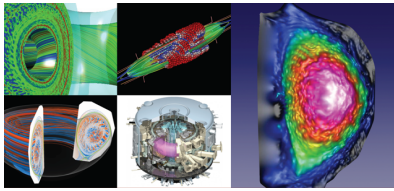
Particle Physics, Astrophysics



Nuclear Physics



Materials, Chemistry, Geophysics



Fusion Energy, Plasma Physics

Use-cases and examples here

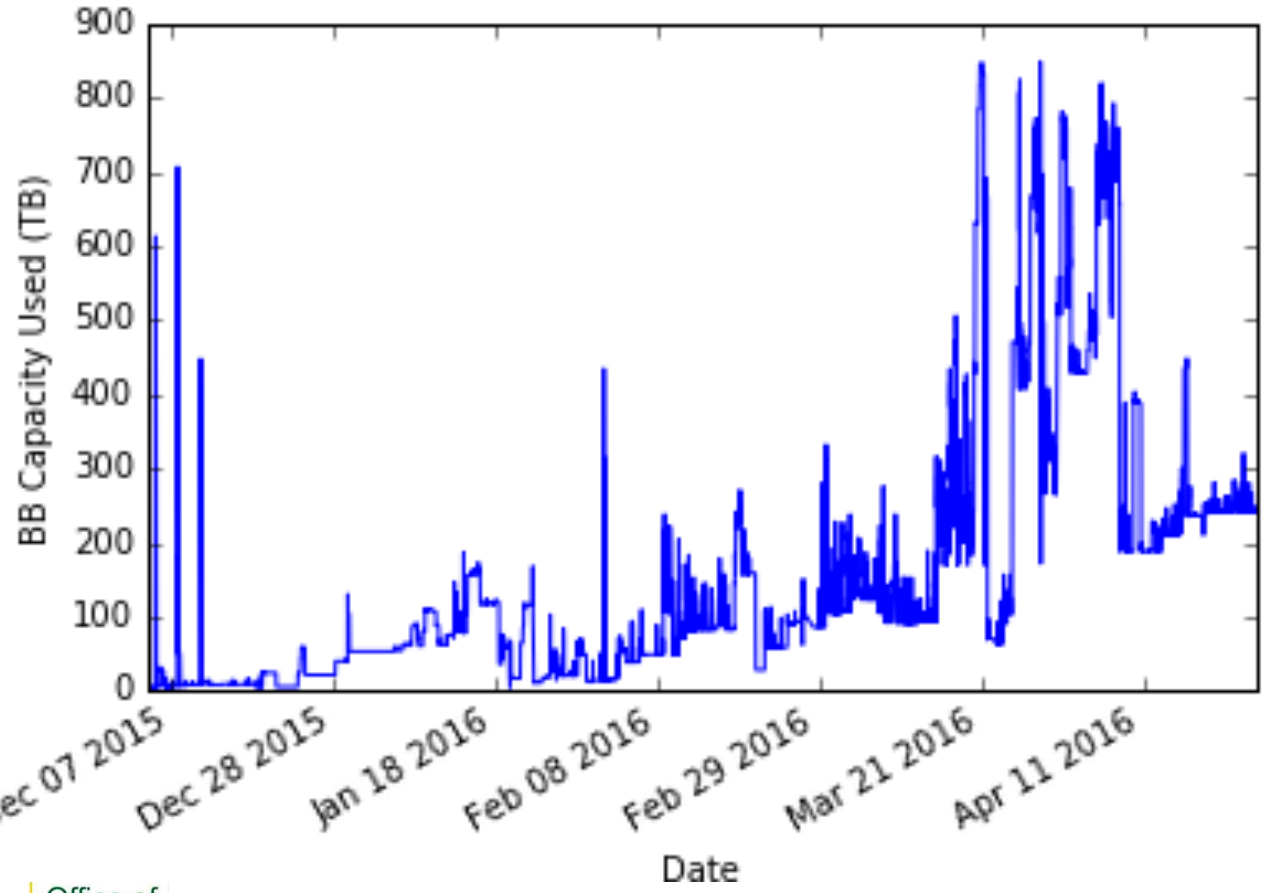
Burst Buffer Use-Case
IO High Bandwidth: Reads/ Writes
Data-intensive Experimental Science - “Challenging” IO patterns, eg. high IOPs
Workflow coupling and visualization: in transit / in-situ analysis
Staging experimental data

Use-cases and examples here

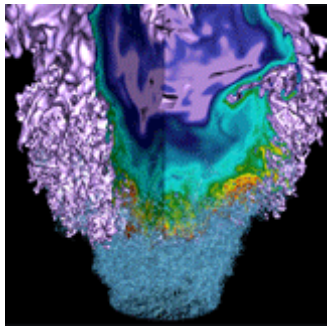
Burst Buffer Use-Case	Early Users Covered Here
IO High Bandwidth: Reads/ Writes	• Nyx/BoxLib • VPIC IO
Data-intensive Experimental Science - “Challenging” IO patterns, eg. high IOPs	• ATLAS experiment • TomoPy for ALS and APS
Workflow coupling and visualization: in transit / in-situ analysis	• Chombo-Crunch / VisIt carbon sequestration simulation
Staging experimental data	• ATLAS and ALS SPOT Suite

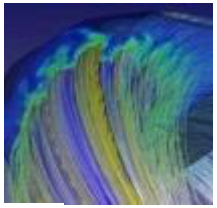
Usage

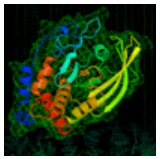
Many other projects not described here
~50 active users though not yet opened for general access

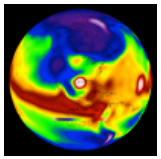


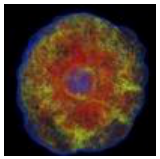
Science Use Cases

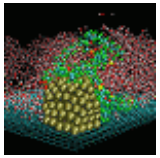








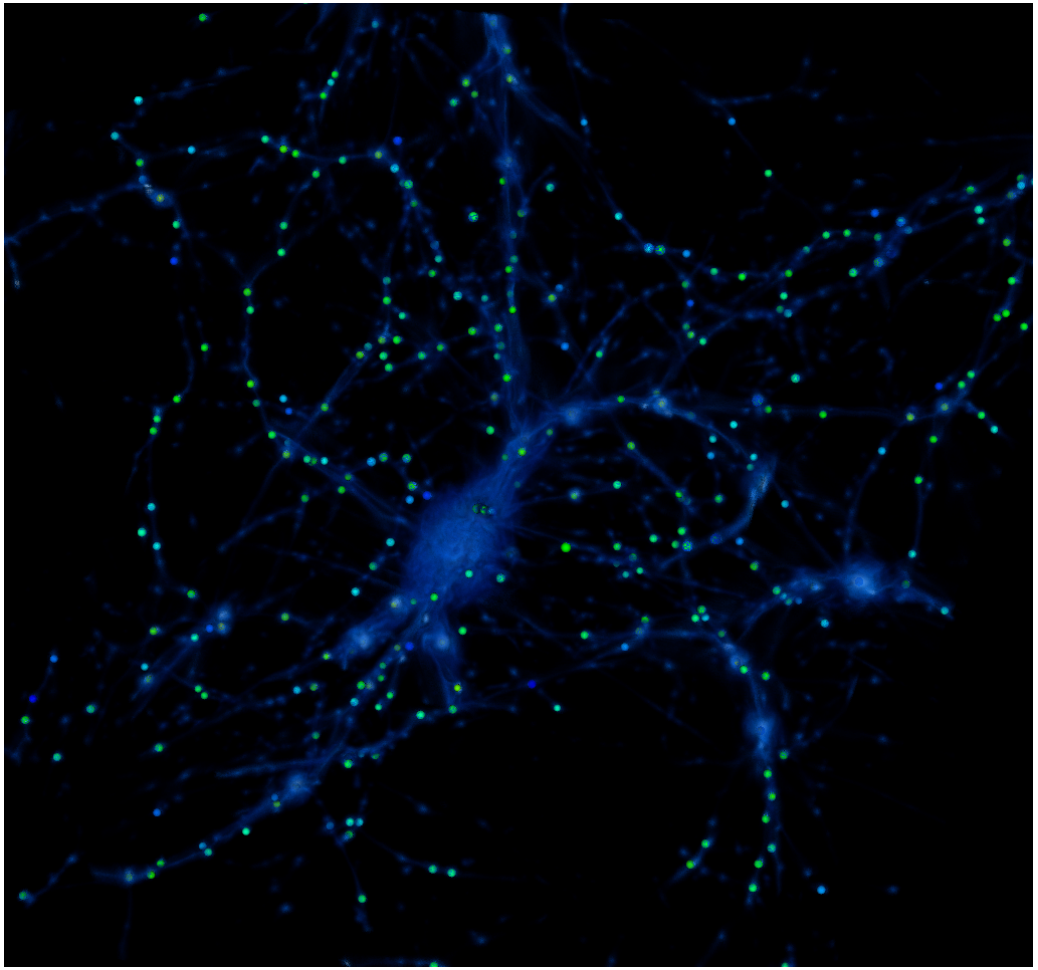




Nyx/BoxLib

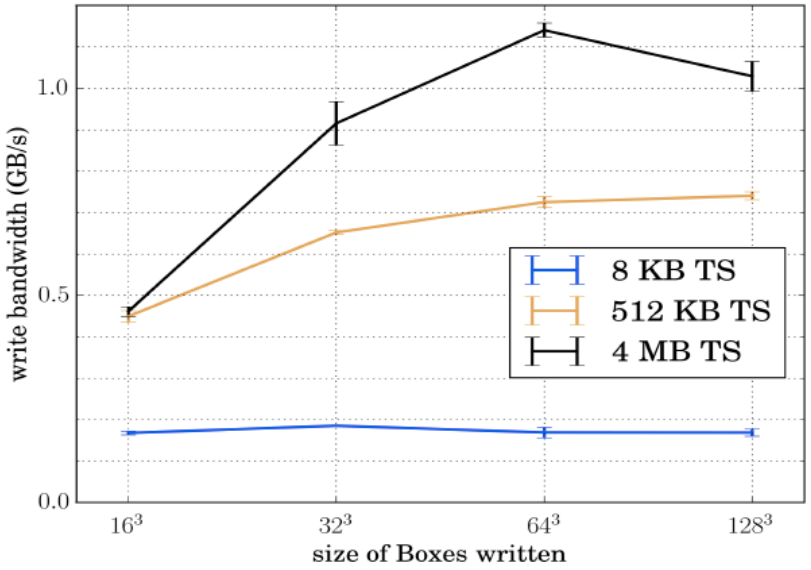
Brian Friesen and Vince Beckner

- 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
- Burst Buffer offers I/O time savings and potential for in-transit analysis



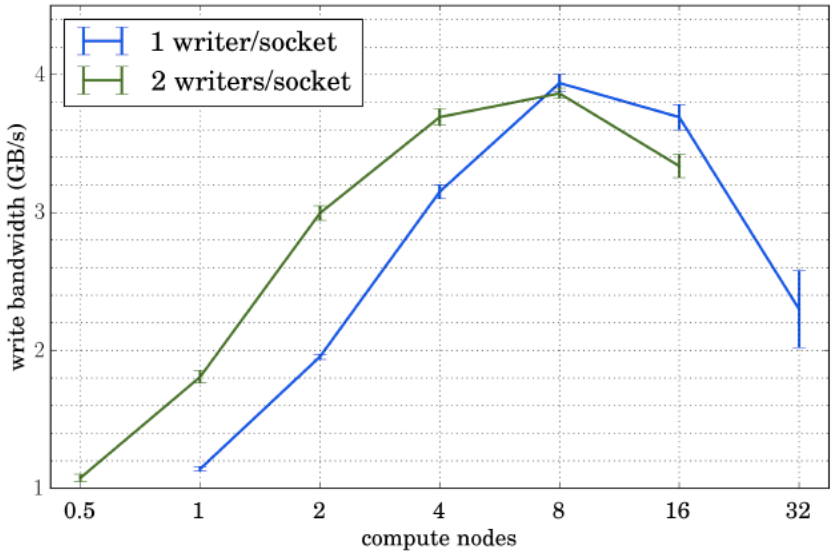
Nyx/Boxlib – Single BB Node

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



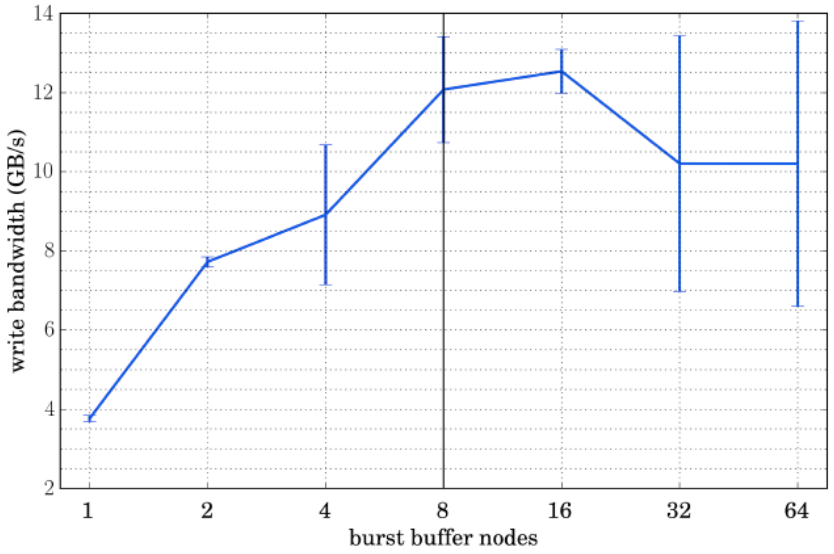
Nyx/Boxlib – Single BB Node

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



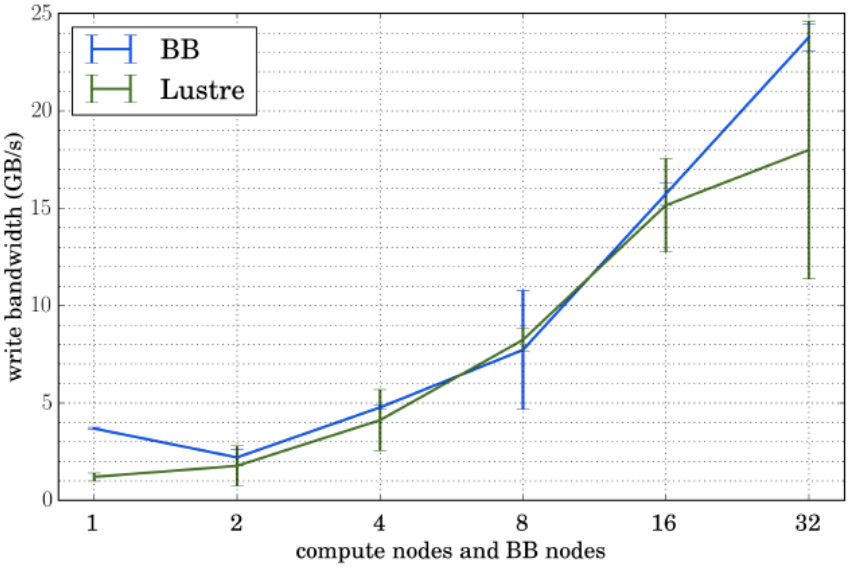
Nyx/Boxlib – Scaling up

- 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



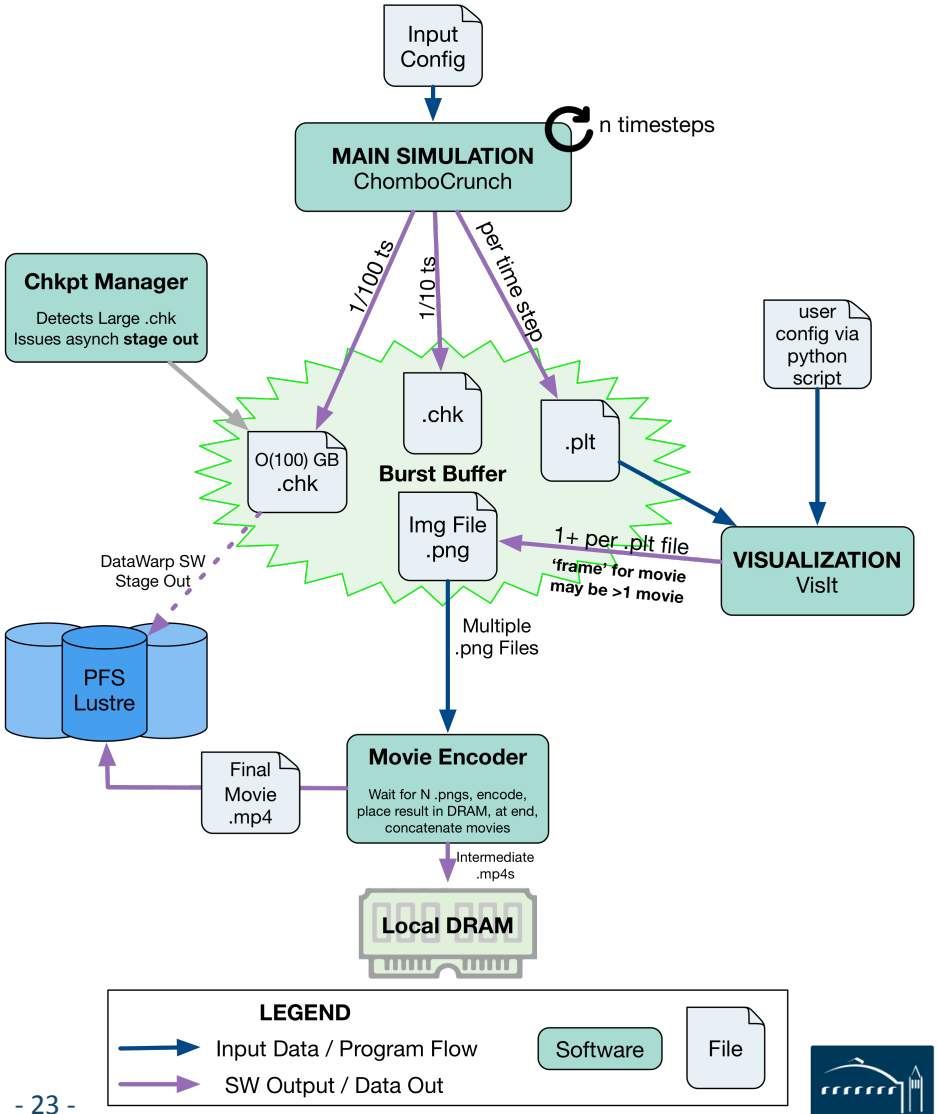
ChomboCrunch and VisIT

Andrey Ovsyannikov, Melissa Romanus, Brian Van Straalen

- Chombo-Crunch high-performance computational fluid dynamics and reactive transport code simulates pore-scale reactive transport processes associated with carbon sequestration
  - –All MPI ranks write to single shared HDF5 '.plt' file.
  - –Output varies on resolution – can be 100s TBs.
- VisIT – visualisation and analysis tool for scientific data
  - –Reads '.plt' files, produces '.png' for encoding into movie
- Traditionally have to do on Lustre as separate processes, ".plt" and ".png" files need not be retained

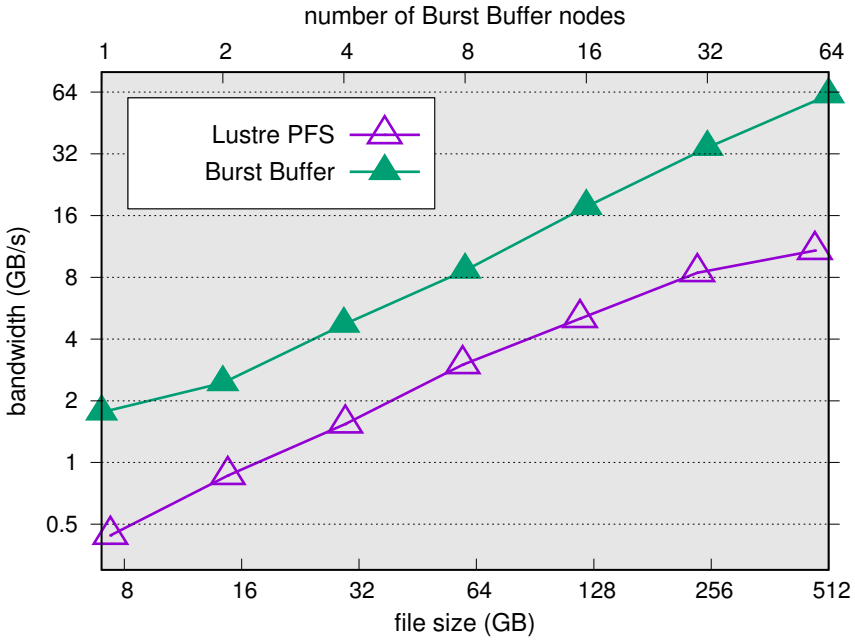
Workflow

- Use coupled processes storing '.plt' files on Burst Buffer
- This study ran encoding offline but both that and bleeding of 1/10 checkpoints to PFS can now also be done in place



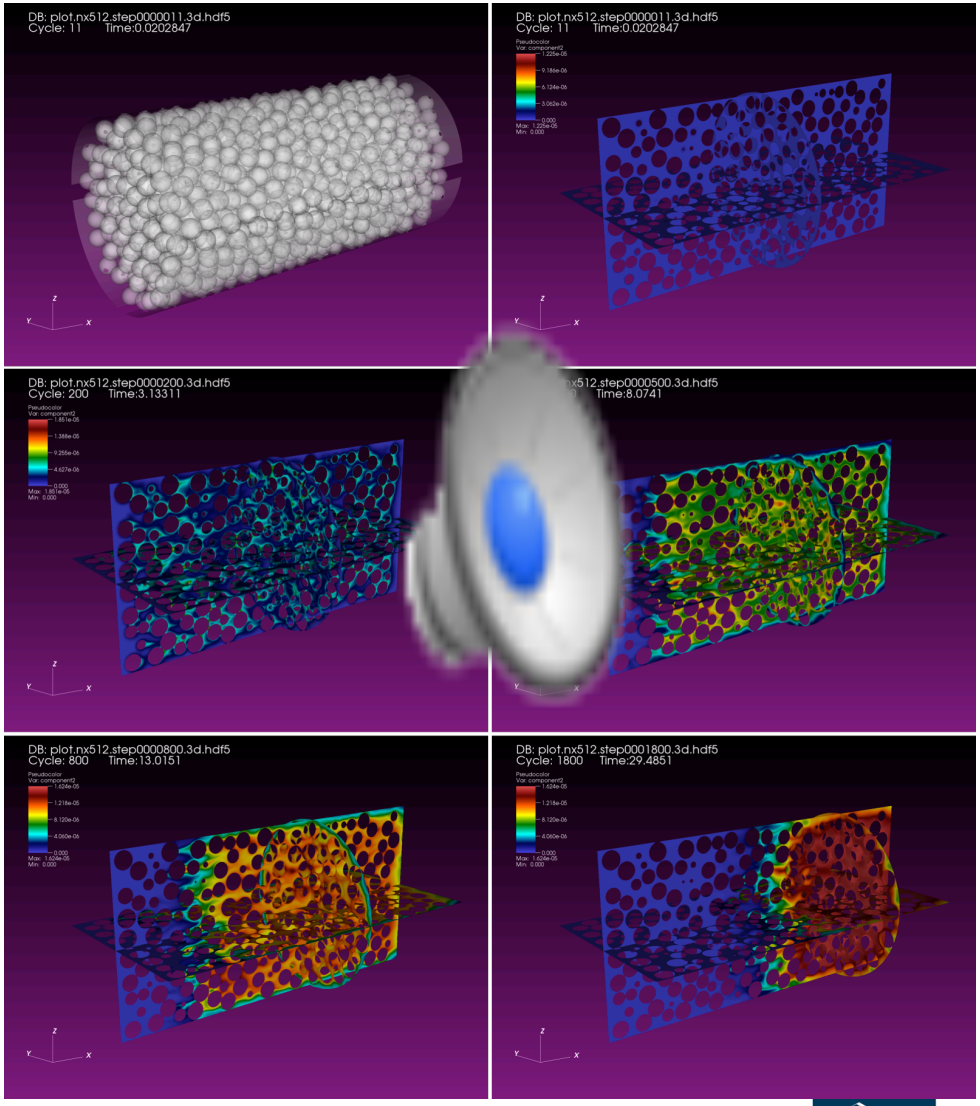
Scaling

- Compute node / BB node scaled from 16/1 to 1024/ 64
- Bandwidth achieved is around a quarter of peak
- Lustre results used a 1MB stripe size and a stripe count of 72 OSTs
- Burst Buffer significantly out performs Lustre for this application at all resolution levels and the bandwidth scales exceptionally well.



In-transit Movie

- 'packed cylinder' [1]
- Simulation on 8192 cores over 256 nodes with 8 further nodes used for VisIt.
- Full BB, 140 nodes:
  - –90.7GB/s obtained
- **A [first] coupled science workflow using the Burst Buffer**

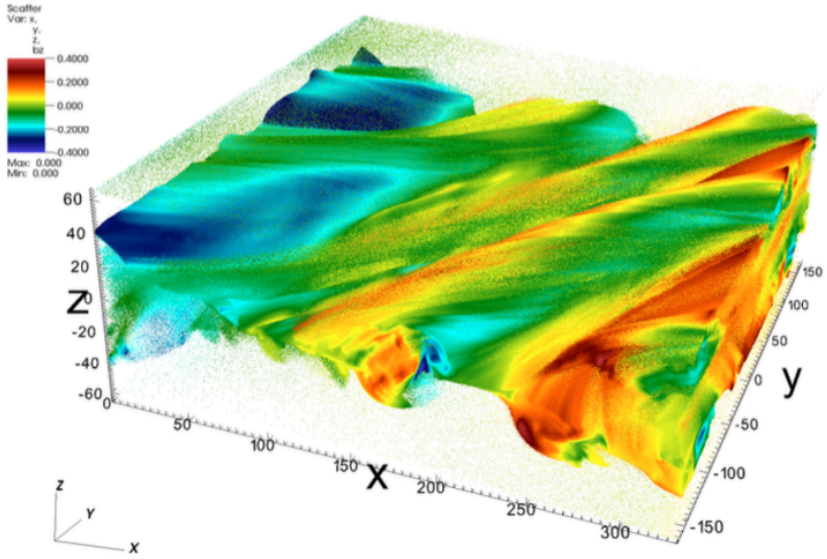


[1] D. Trebotich and D. Graves, "An adaptive finite volume method for the incompressible Navier–Stokes equations in complex geometries," Communications in Applied Mathematics and Computational Science, vol. 10, no. 1, pp. 43–82, 2015.

VPIC I/O

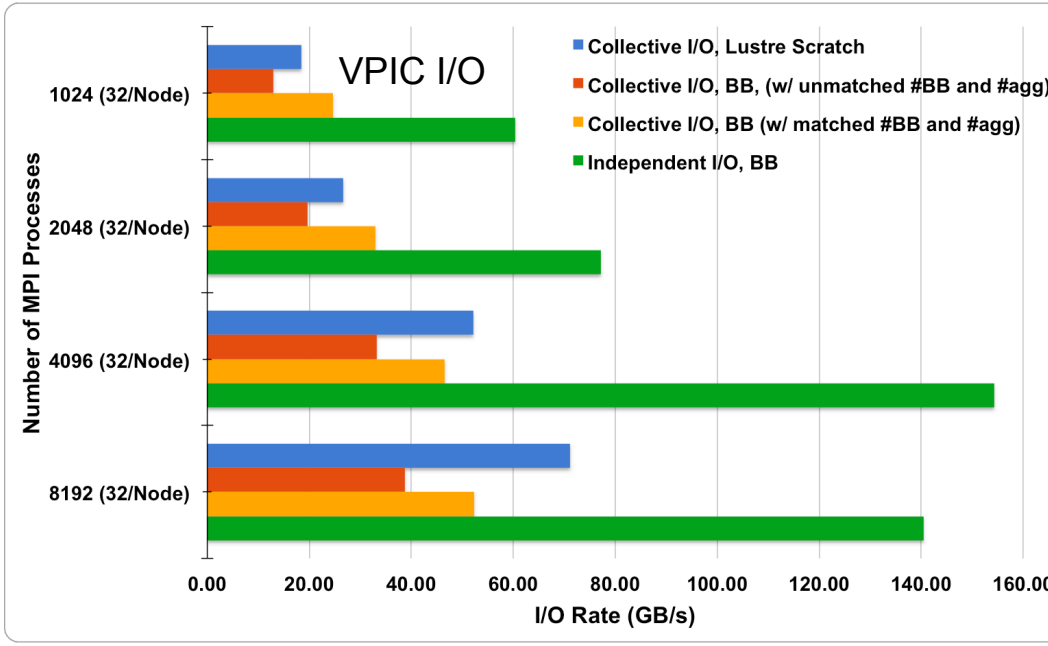
Matt Bryson, Suren Byna, Glenn K. Lockwood

- Plasma physics simulation
- Shared file I/O using HDF5
- Can be large amount of data e.g. magnetic reconnection with two trillion particles – 32-40 TB per time step
- Write out each time step to Burst Buffer with asynchronous copy to PFS
- Also potential for in-transit visualization



VPIC I/O: MPI-IO Collective

- Using 65 Burst Buffer nodes 'unmatched' with collective MPI aggregators – poor performance
- 64 BB nodes – 'matched' – significantly better
  - –Comparable with Lustre
- Independent I/O performs 4x better
- Profile with Darshan and VPIC-like IOR run confirms MPI collective overhead



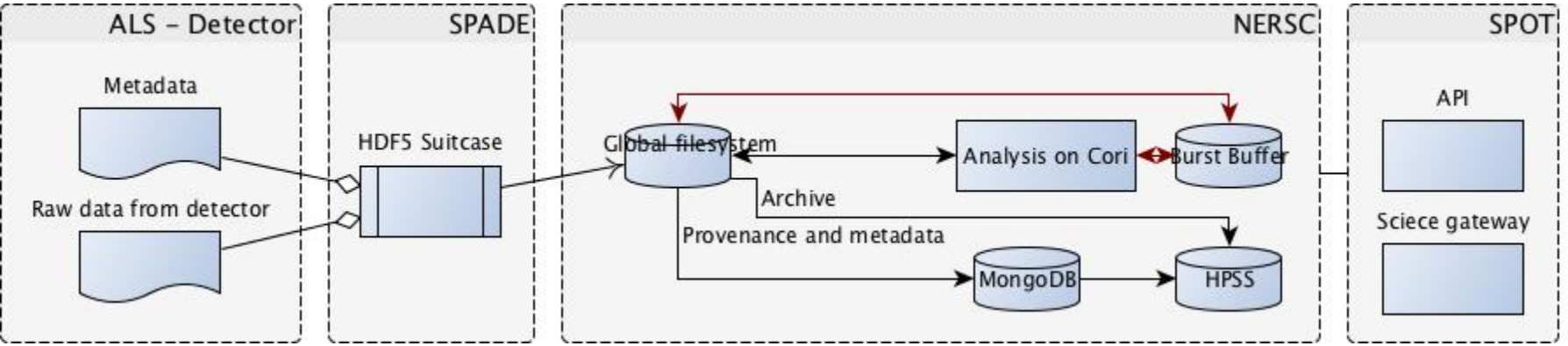
IOR based modeling of I/O pattern:

API	Mean B/W (GB/s)
HDF5	14.7
MPIIO	15.4
POSIX	66.5

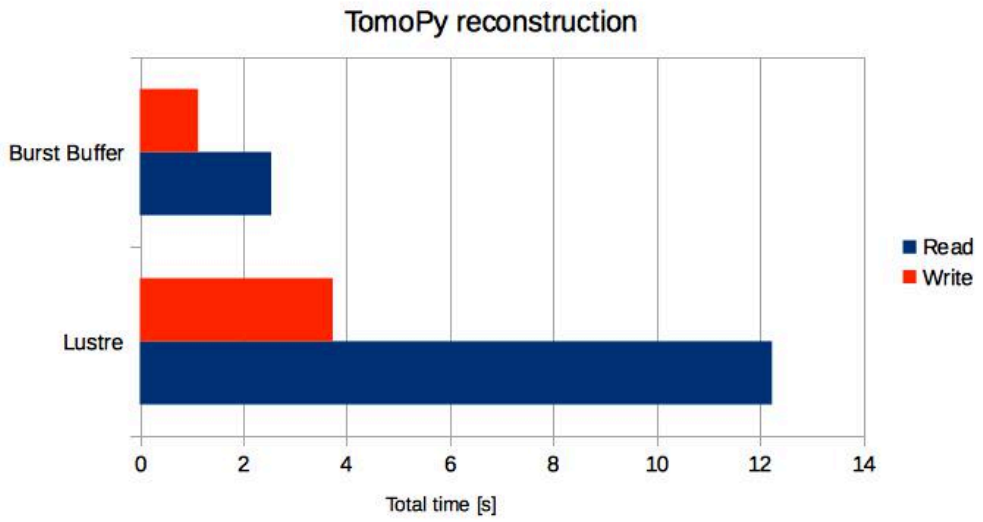
TomoPy and SPOT

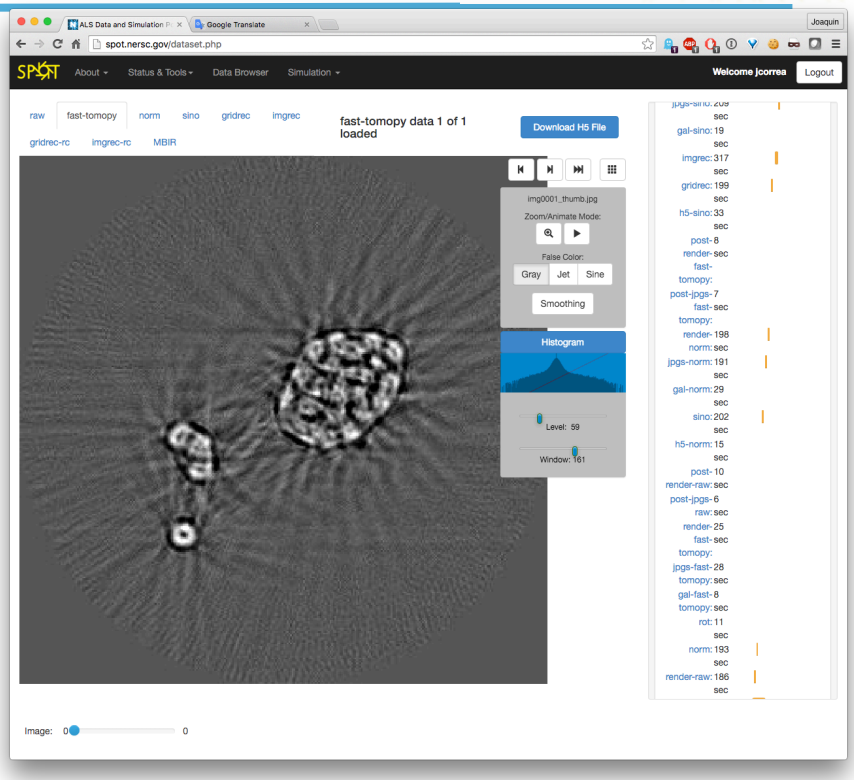
Joaquin Correa, Doga Gursoy

- Open-sourced Python toolbox for tomographic data processing and image reconstruction tasks
  - –Optimised to avoid intermediate data on disk – but still ~30% time in I/O
- Can be run with SPOT suite
  - –Workflow coupled to experimental beamline at ALS or APS



TomoPy Results



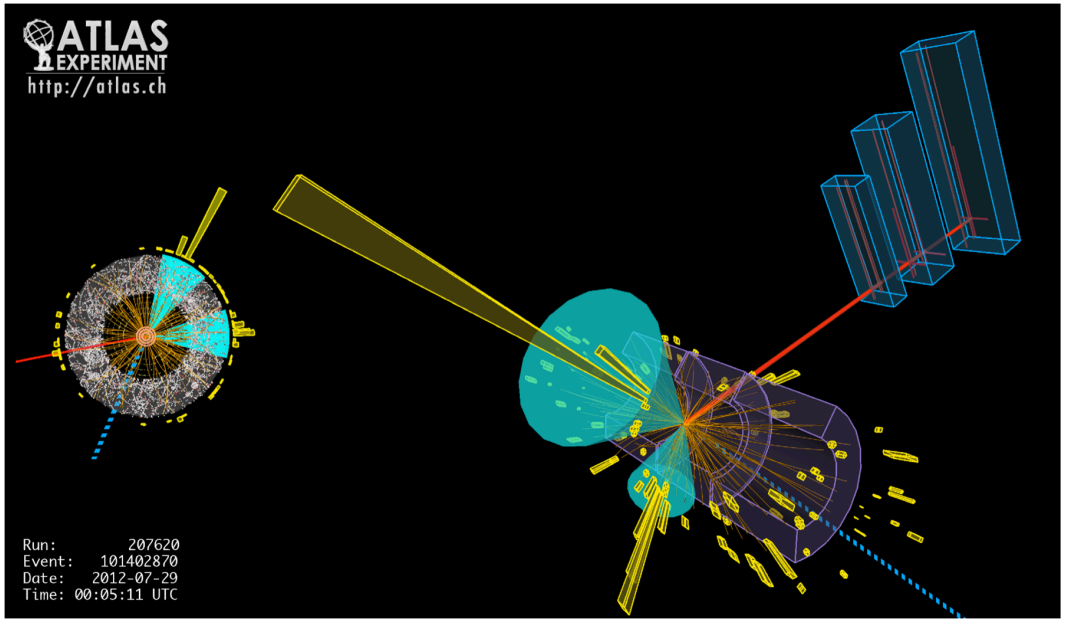


- 500 GB Burst Buffer Space
- 4 compute nodes Cori 'realtime' queue
  - –10GB input data
- Promising but production runs show variation in time

- SPOT interface of TomoPy analysis on the BB
- **[First] coupling of BB into an experimental science workflow**

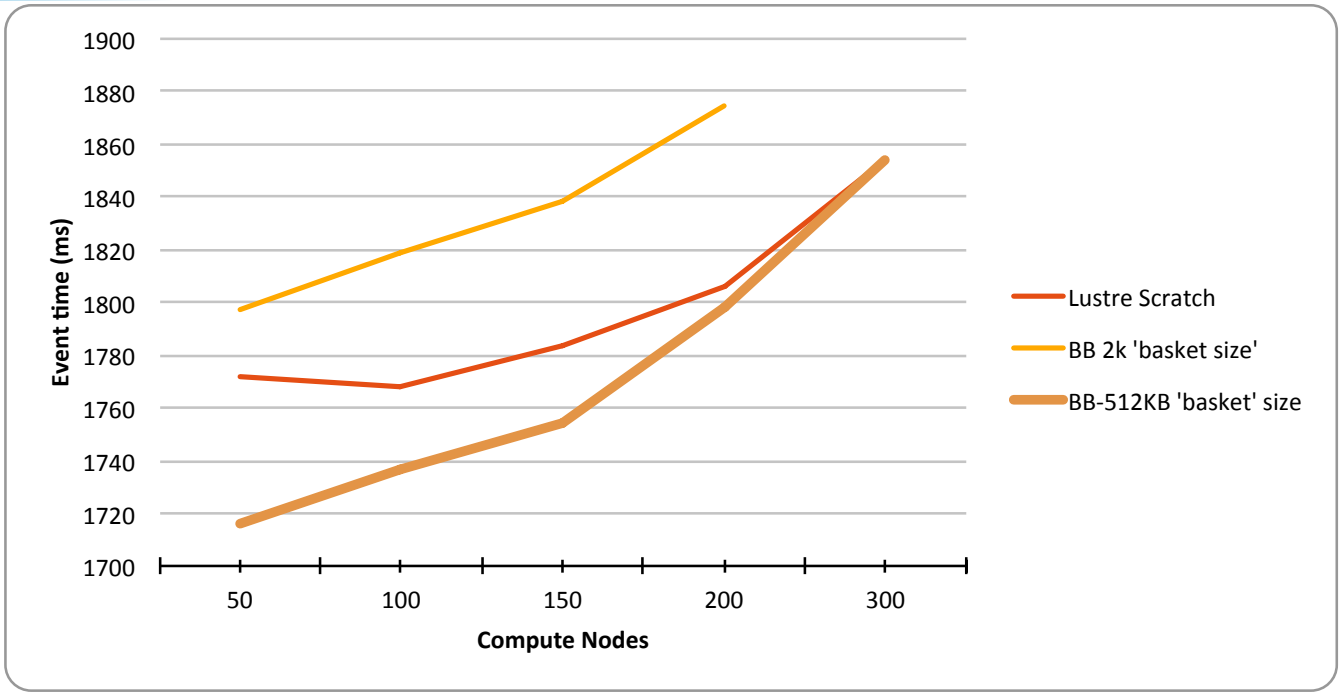
ATLAS/Yoda

Vakho Tsulaia, Steve Farrell, WB



- ATLAS LHC experiment – 100s of Petabytes of data processed worldwide - but little use of 'HPC' machines
- 'Yoda' packages ATLAS payloads for HPC
  - –Used in production but running least I/O intensive simulation
  - –Use Burst Buffer to run I/O intensive analysis

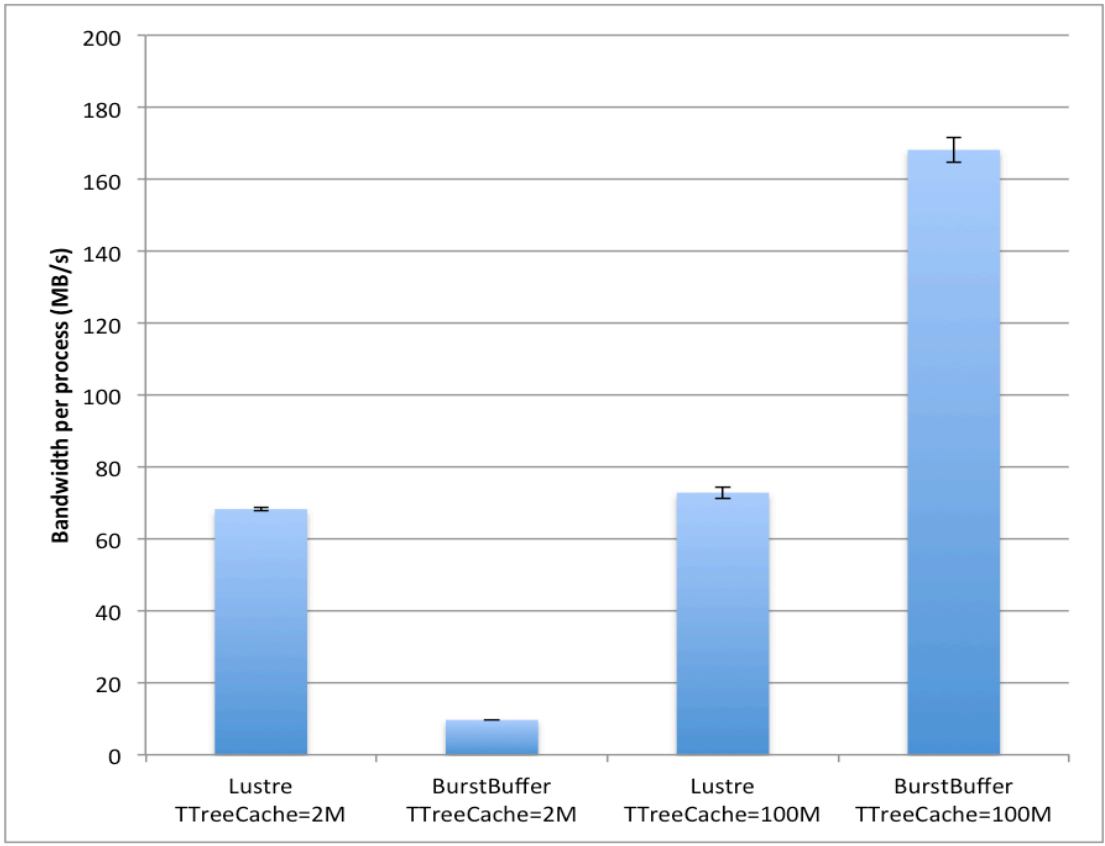
Scaling of Simulation Payload



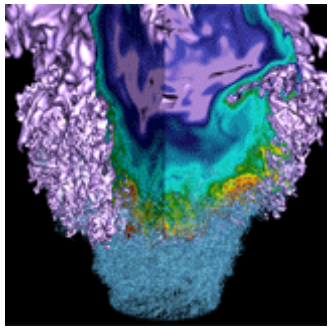
- Initial scaling on BB poor
- Increase ROOT 'basket size' from 2k to 512k to increase transaction size
- Keep log files on Lustre
- Then scales to >300 nodes
- But this is not most I/O intensive payload…

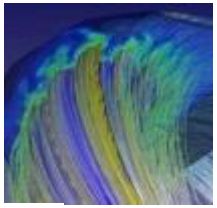
Atlas Analysis

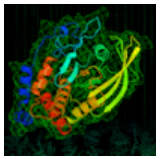
- Initial study of I/O intensive analysis
- Reading 475 GB dataset from a single node: custom ‘ROOT’-based format
- 32 processes per node
- 2TB BB
- Poor initial BB performance
- Increase application memory cache ‘TTreeCache’ to 100M
- **Less reads – > 17x performance boost on BB**

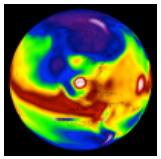


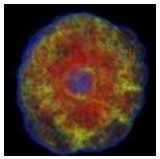
Lessons and future

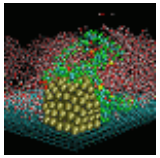












NERSC 40 YEARS at the FOREFRONT

Challenges ...

- **Initial instabilities resolved in early patches**
  - Early Users are the best testers of a new system!
  - New issues cropping up as use patterns extended at scale
- **Scaling limits**
  - Stage-in: tune time outs for REST API communication
  - Number of underlying open files (as each substripe is a new file on the xfs filesystem)
- **Usability**
  - Fussy syntax; error-handling etc.

Using a whole new multilayer filesystem approach for the first time at scale with a range of applications – close NRE relationship with Cray helped resolve quickly

Coming performance improvements

1. DVS client-side caching

- Lustre has client-side caching, currently DVS does not
- Will help small sequential Read/Write transfers and re-reads on BB
- Expected later this year

2. Smaller granularity

- Amount of space allocated on each BB node, currently cannot be configured lower than ~200GB
  - Users have to occupy more space than they need to get striped file performance
- Ability to lower this value coming ~June in Rhine/Redwood

3. MPI-IO shared file performance

- Can be improved with more substripes on Burst Buffer but requires improved metadata handling
- Coming ~June in Rhine/Redwood

We're working with Cray to improve BB performance out-of-the-box and for all use cases

Lessons Learned

- **Early User Program was crucial to our debugging of this complex new technology**
  - Led to performance improvements and fixes to scaling limits, operational problems and usability issues
  - Impossible to achieve with synthetic tests or with large numbers of generic users.
- **The Burst Buffer provides a high-performance solution for large streaming I/O**
  - E.g. Nyx large block transfers
- **The Burst Buffer enables coupled workflows**
  - Chombo-Crunch and VisIt Visulisation
  - Experimental workflows with TomoPy

Lessons Learned

- **Challenging I/O patterns - mixed performance**
  - Tuned ATLAS analysis and TomoPy run well
  - But many do not perform well out of the box – e.g. small transfer-sizes in Nyx case; default ATLAS analysis I/O
  - Hope that planned DVS client side caching and metadata improvements will help
- **MPI-IO with Burst Buffers will require further tuning to perform well.**
  - E.g. in VPIC-IO Collective MPI study
  - Several years of tuning for Lustre PFS, also DVS optimisations applied by default are not optimal for BB

Lessons Learned

- **Tuning of transfer size and number of parallel writers is needed with the Burst Buffer, more so than with Lustre**
  - –Larger transfer sizes have been generally seen to be better for example in Nyx. (This may be different on a heavily contended system where allocating large parts of Burst Buffer DRAM may not be possible).
  - –It is not possible to max out Burst Buffer bandwidth using a single process on a single node.
- **NERSC Cray DataWarp system now functions for users**
  - –Culmination of considerable efforts by Cray, SchedMD, NERSC systems and user staff and the early users.

Some Lessons and Conclusions

- **Successful Early User Program: crucial to our debugging of this complex new technology**
  - Exposed issues: led to performance improvements and fixes to scaling limits, operational problems and usability issues; impossible to achieve with synthetic tests or with large numbers of generic users.
- **The Burst Buffer provides a high-performance solution for large streaming I/O**
- **Other I/O patterns currently have mixed performance**
- **MPI-IO with Burst Buffers will require further tuning**
- **The Burst Buffer enables coupled workflows**
- **Tuning of transfer size /writers is needed with the Burst Buffer, more so than with Lustre**

Conclusions

- NERSC Cray DataWarp system now functions well
  - Culmination of considerable efforts by Cray, SchedMD, NERSC systems and user staff and the early users.
- Demonstrated here Burst Buffer use-cases for science:
  - **Systematic study** of transfer size/ writers / scaling to get **near peak bandwidth** for real science with **Nyx**
  - **Coupling of simulation and visualization** with the Burst Buffer in **Chombo-Crunch and VisIT**
  - **Significantly outperforming Lustre filesystem** for both **Chombo-Crunch and VPIC-IO**
  - **Accelerate experimental science** including analysis I/O patterns for **TomoPy** and tuned **ATLAS analysis**