UC Davis

UC Davis Previously Published Works

Title

Accelerating Science with the NERSC Burst Buffer Early User Program

Permalink

https://escholarship.org/uc/item/9wv6k14t

Authors

Bhimji, W
Bard, D
Romanus, M
et al.

Publication Date

2016

Peer reviewed

eScholarship.org

Powered by the California Digital Library
University of California

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*, David Trebotich*, Craig Tull*, Gunther Weber*,
Nicholas J. Wright*, Katie Antypas*, Prabhat*
* Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA, Email: wbhimji@lbl.gov
† Rutgers Discovery Informatics Institute, Rutgers University, Piscataway, NJ, USA
‡Advanced Photon Source, Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL 60439, USA

Abstract—**NVRAM-based Burst Buffers are an important part of the emerging HPC storage landscape. The National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory recently installed one of the first Burst Buffer systems as part of its new Cori supercomputer, collaborating with Cray on the development of the DataWarp software. NERSC has a diverse user base comprised of over 6500 users in 700 different projects spanning a wide variety of scientific computing applications. The use-cases of the Burst Buffer at NERSC are therefore also considerable and diverse. We describe here performance measurements and lessons learned from the Burst Buffer Early User Program at NERSC, which selected a number of research projects to gain early access to the Burst Buffer and exercise its capability to enable new scientific advancements. To the best of our knowledge this is the first time a Burst Buffer has been stressed at scale by diverse, real user workloads and therefore these lessons will be of considerable benefit to shaping the developing use of Burst Buffers at HPC centers.**

Index Terms—**Nonvolatile memory, Data storage systems, Burst Buffer, Parallel I/O, High Performance Computing**

I. Introduction

HPC faces a growing I/O challenge. One path forward is a fast storage layer, close to the compute, termed a Burst Buffer [1]. Such a layer was deployed with the first phase of the Cori Cray XC40 System at NERSC in the later half of 2015, providing around 900 TB of NVRAM-based storage. This system not only employs state-of-the-art SSD hardware, but also a new approach to on-demand filesystems through Cray's DataWarp software. In order to enable scientific applications to utilize this new layer in the storage hierarchy, NERSC is running the Burst Buffer Early User Program, focused on real science applications and workflows that can benefit from the accelerated I/O the system provides. The program is providing a means to test and debug the new technology as well as drive new science results.

In this paper we first briefly review the motivation for Burst Buffers and the range of potential use-cases for NERSC's diverse scientific workload. We then provide a brief overview of the architecture deployed at NERSC in Section II-B before outlining the Early User Program and the projects selected. We then focus on five specific projects and describe in detail their workflow, initial results and performance measurements. We conclude with several important lessons learned from this first application of Burst Buffers at scale for HPC.

A. The I/O Hierarchy

Recent hardware advancements in HPC systems have enabled scientific simulations and experimental workflows to tackle larger problems than ever before. The increase in scale and complexity of the applications and scientific instruments has led to corresponding increase in data exchange, interaction, and communication. The efficient management of I/O has become one of the biggest challenges in accelerating the time-to-discovery for science.

Historically, the memory architecture of HPC machines has involved compute nodes with on-node memory (DRAM), a limited number of I/O subsystem nodes for handling I/O requests, and a disk-based storage appliance exposed as a parallel file system. DRAM node-memory is an expensive commodity with limited capacity, but fast read/write access, while disk-based storage systems provide a relatively inexpensive way to store and persist large amounts of data, but with considerably lower bandwidth and higher latency.

This traditional HPC architecture is often unable to meet the I/O coordination and communication needs of the applications that run on it, particularly at extreme scale. In order to address this I/O bottleneck system architects have explored ways of offering cost-effective memory and filesystem solutions that can offer faster performance than parallel filesystems on disk-based storage. A natural extension of this work has been to explore ways of deepening the memory hierarchies on HPC machines to include multiple storage layers in-between DRAM and disk. These proposed solutions leverage technology advancements like solid-state devices (SSDs), as well as other flash-based and/or NVRAM offerings.

Therefore, some state-of-the-art HPC systems now include a new tier of 'intermediate' storage between the compute nodes and the hard disk storage, known as a 'Burst Buffer'. This layer is slower (but higher capacity) than on-node memory, but faster (and lower capacity) than HDD-based storage.

B. The Burst Buffer

Burst Buffers can be realized in a variety of ways. In the case of the Cray DataWarp implementation [2], it is achieved through SSDs in I/O nodes that are directly connected to the high-speed network, rather than in compute nodes. The DataWarp software presents to the application a POSIX filesystem interface built on these SSDs. However, unlike a traditional parallel filesystem, this mount is only available to the compute nodes using it, and for limited duration. The software also allows for 'persistent' Burst Buffer reservations so data can be re-used by subsequent (or simultaneous) compute jobs without re-staging. Further details of the software involved are provided in Section II-C.

The Burst Buffer therefore has the potential for considerably higher performance than the underlying Parallel File System (PFS) for a variety of reasons, including the underlying storage medium and the high-performance network connection, as well as the possibility to limit metadata load by exposing only the namespace required for the job or workflow.

C. Burst Buffer use-cases

Burst Buffer technology was primarily conceived as a high-bandwidth solution for a checkpoint-and-restart application, but at NERSC there are a variety of additional purposes for which this technology can be valuable [3]. Example use cases include:

- IO improvements for:
  - High-bandwidth streaming reads and writes, e.g. for checkpoint-and-restart
  - Complex I/O patterns, including those with high IO operations per second (IOPs), e.g. non-sequential table lookup
  - Out-of-core applications
- Workflow performance improvements (either within one compute job or across many jobs using a persistent Burst Buffer reservation) for:
  - Coupling applications, using the Burst Buffer as interim storage between for example simulation and analysis codes.
  - Optimizing node usage by changing node concurrency partway through a workflow
  - Analysis and visualization: including in-situ, in-transit and interactive

The early-user projects selected for the NERSC program stress many of these different use-cases as shown in Section III.

II. NERSC Burst Buffer

A. Cori

Cori is NERSC's newest supercomputer system. The Cori system will be delivered in two phases with the first phase online now and the second expected in mid-2016. Phase 2 will be based on the second generation of the Intel Xeon Phi family of products, called the Knights Landing (KNL) Architecture. The Phase 1 system (also known as the "Cori Data Partition") is a Cray XC40 System with 1632 dual-socket compute nodes with two 2.3 GHz 16-core Haswell processors and 128 GB of DRAM per node, a Cray Aries high speed "dragonfly" topology interconnect and a number of new features that will benefit data-intensive science, including the Burst Buffer. The Cori system also has a very high performance Lustre scratch filesystem with 27 PB of storage served by 248 OSTs, providing over 700 GB/s peak performance.

B. Burst Buffer Architecture

The current Cori Phase 1 system has 144 Burst Buffer nodes. Each NERSC Burst Buffer node contains two Intel P3608 3.2 TB NAND flash SSD modules attached over two PCIe gen3 interfaces. A single SSD appears as 2 block devices. These are packaged two to a blade (shown in Figure 1), and attached directly to the Cray Aries network interconnect of the Cori system (Figure 2).

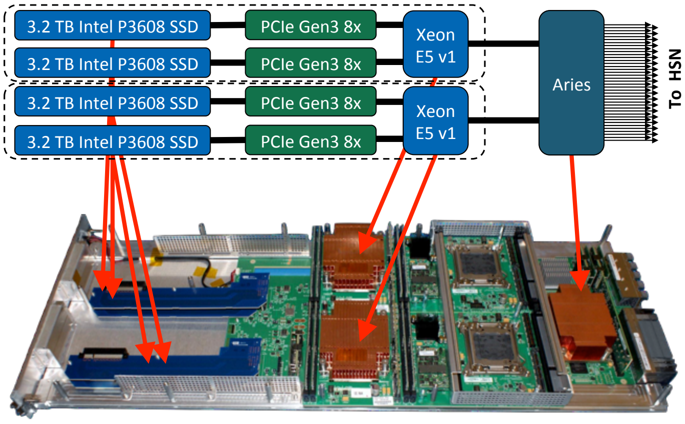


Fig. 1. A Cori Burst Buffer Blade

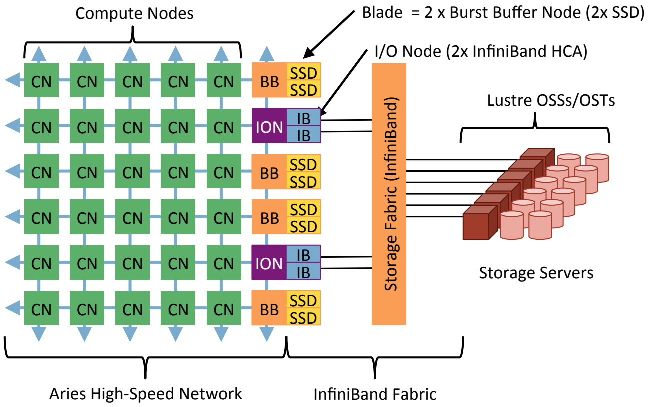


Fig. 2. Placement of Burst Buffer nodes in the Cori System

C. Software Environment

In addition to the hardware, NERSC has invested in software through Non-Recurring Engineering (NRE) projects with Cray and SchedMD, supporting the Cray DataWarp software, and integration with the SLURM workload manager (WLM). This is being delivered in a series of stages. Stage 1 (currently deployed) supports a range of features allowing users to automatically setup filesystems on Burst Buffer nodes.

Users can allocate Burst Buffer resources via the SLURM WLM. Resources can be striped across different Burst Buffer nodes, or used in a ‘private’ mode whereby each compute node gets its own namespace which potentially offers improved metadata handling. These are documented in the Cray and Slurm User Guides [2] [4] as well as numerous practical examples provided on the NERSC website [5].

Details of the entire DataWarp software stack, including the services that create the mount points, can be found in the DataWarp admin guide [6]. We note here that a single DataWarp filesystem mount involves several layers:

- Logical Volume Manger (LVM) is used to group the multiple SSD block devices on a single node into one logical block device.
- An XFS file system that is created for every Burst Buffer allocation, comprising the allocated Burst Buffer space. The Burst Buffer allocation therefore appears to the user as an isolated filesystem.
- The DataWarp File System (DWFS), which is a stacked file system based on wrapfs [7]. It handles coordination between the Burst Buffer nodes and staging data in/out of the Burst Buffer allocation, and provides the namespaces (e.g. striped access type) described above.
- Cray Data Virtualization Service (DVS) [8], used for communication between DWFS and the compute nodes.

These layers are shown schematically in Figure 3, and their interaction when striping a file between Burst Buffer nodes is shown in Figure 4.

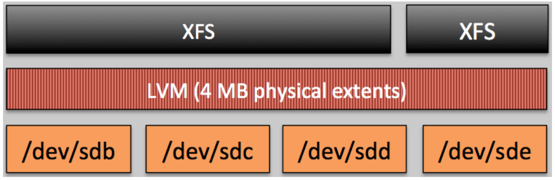


Fig. 3. Schematic of the different layers of the DataWarp software. The Burst Buffer node is logically 4 SSD block devices, which are aggregated by LVM. When a Burst Buffer allocation is requested by the user, an XFS filesystem is created by the DataWarp software on that allocation. In this example, two different allocations coexist on the same Burst Buffer node

To illustrate the normal user-level interaction with the Burst Buffer, example batch script directives are given below:

```
#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
```

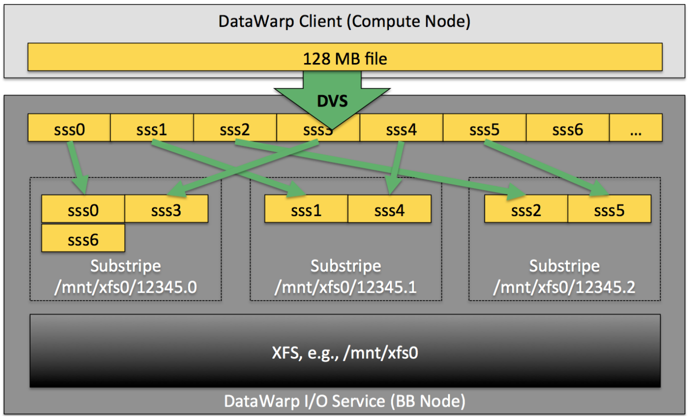


Fig. 4. Schematic of how a file is striped over the DataWarp software on one Burst Buffer node. DVS serves the file to DataWarp in (configurable) 8MB chunks, which are laid out across the three (configurable) substripes on the Burst Buffer node.

This example requests a Burst Buffer allocation for the duration of the compute job (“type=scratch”) and that would be visible from all compute nodes (“access_mode=striped”). The allocation is distributed across Burst Buffer nodes in units of ‘granularity’ (currently 218 GB on the NERSC system). So a request of 1000 GB will normally be placed over 5 separate Burst Buffer nodes (though this is not guaranteed). A file and a directory are staged in using the “stage_in” directive, and a directory is staged out at the end of the job using “stage_out”. Since the path to the Burst Buffer allocation is unknown when the job is submitted, the user can configure their executable to read in the *$DW_JOB_STRIPED* variable at runtime.

An API in the C language is also provided, which allows DataWarp users to control the flow of data to/from the Burst Buffer from within their application code, including staging data in/out asynchronously with compute operations.

Future planned stages of DataWarp software will add implicit caching (Stage 2) and in-transit analysis capabilities (Stage 3). Implicit caching will allow data to be transferred between the PFS and the Burst Buffer without the user specifying how, when or where the data is moved. Staging data in and out of the Burst Buffer and caching temporary files will be performed by the DataWarp software. In-transit analysis in Stage 3 will make use of the Intel Xeon CPU on the Burst Buffer node to do fast analysis of data stored on the local Burst Buffer SSD.

D. Benchmark Performance

The performance of the Cori Phase 1 Burst Buffer was measured on installation using the IOR benchmark. Bandwidth tests were performed with 8 GB block size and 1 MB transfer size, using total file sizes greater than 1.5x the compute node’s memory. IOPS benchmark tests were performed with random 4KB-sized transfers. 1120 compute nodes were used with 4 processes per node. At the time 140 Burst Buffer nodes were in service. Results are given in Table I. The total Cori Phase 1 Burst Buffer system provides approximately 900 GB/second of peak I/O performance and over 12.5M IOPS. The MPI-IO shared file bandwidth is not at the desired performance level for the system, however this should be improved in future versions of the DataWarp software where it will be possible to use more sub-stripes on the underlying Burst Buffer nodes (and therefore increased parallelism). All benchmark bandwidth numbers (including MPI-IO Shared file) outperform the Lustre scratch filesystem which achieves around 700-750 GB/s peak (POSIX File-Per-Process) bandwidth and 573/223 GB/s Read/Write performance for MPI-IO shared file.

TABLE I
I/O PERFORMANCE TO THE CORI BURST BUFFER AS MEASURED WITH THE IOR BENCHMARK (DETAILS GIVEN IN THE TEXT)

Posix File-Per-Process		MPI-IO Shared File		IOPS	
Read	Write	Read	Write	Read	Write
905 GB/s	873 GB/s	803 GB/s	351 GB/s	12.6 M	12.5 M

III. BURST BUFFER EARLY USER PROGRAM OVERVIEW

NERSC has the most diverse user base of all the DOE computing facilities, with over 6500 users working on more than 700 projects, running over 700 codes [9]. This makes NERSC an ideal environment to comprehensively test the use of Burst Buffers as a new layer in the storage hierarchy. The diversity of the user base ensures that the hardware and software is challenged in unforeseen ways, as highlighted in Section V. To facilitate the testing and configuration of the Cori Phase 1 Burst Buffer, and to give interested users a first chance to run their code on cutting-edge hardware, NERSC initiated an Early User Program for the Burst Buffer.

In August 2015, NERSC put out a call for proposals, asking the entire user-base of NERSC for projects that could benefit from the Burst Buffer. NERSC received 30 high-quality responses spanning the six DOE Offices of Science and representing the full range of Burst Buffer use cases given in Section I-C. Submissions were evaluated according to the relevance and diversity of their science and computing goals. Proposals that demonstrated a significant potential benefit from the Burst Buffer, and that spanned a representative range of Burst Buffer use cases, were selected to receive individual attention from NERSC computing and data experts. Other proposals were granted early access to the Burst Buffer, without the active support. A compete list of projects selected can be found in Appendix A.

In this paper we focus on detailed results from the following representative projects:

- Nyx/BoxLib: cosmology simulation code
- Chombo-Crunch + VisIt: simulation and visualization of carbon sequestration processes
- VPIC-IO: simulation and analysis of plasma physics simulations
- TomoPy and SPOT: real-time image reconstruction of Advanced Light Source (ALS) and Advanced Photon Source (APS) data
- ATLAS/Yoda: simulation and data analysis for the LHC ATLAS detector

Table II shows that these applications exercise various different use-cases of the Burst Buffer as introduced above. It is also worth noting that we provide examples from both the simulation and experimental science community. The later community provides interesting use-cases for the Burst Buffer in having both challenging I/O requirements and workflows that extend beyond the compute facility.

TABLE II
A TABLE OF THE BURST BUFFER USE-CASES DEMONSTRATED BY THE FEATURED APPLICATIONS (USE-CASES HAVE BEEN MERGED INTO MORE GENERAL HEADINGS FROM THOSE SHOWN IN THE TEXT).

	IO High Bandwidth	Complex IO patterns	Workflow coupling Visualization
Nyx/Boxlib	X		X
Chombo-Crunch+VisIt	X	X	X
VPIC-IO	X		
TomoPy		X	X
ATLAS		X	X

IV. SCIENCE USE CASES

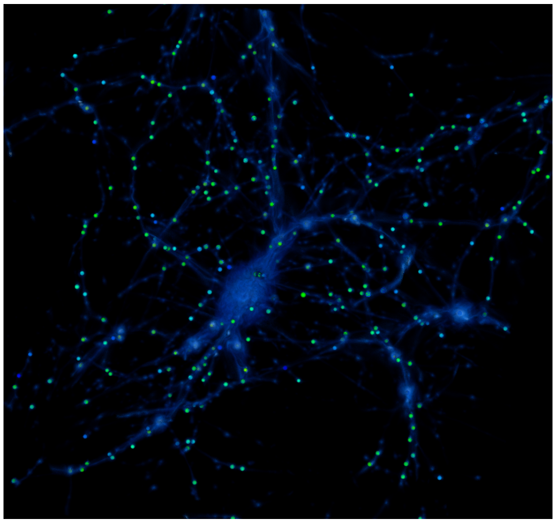
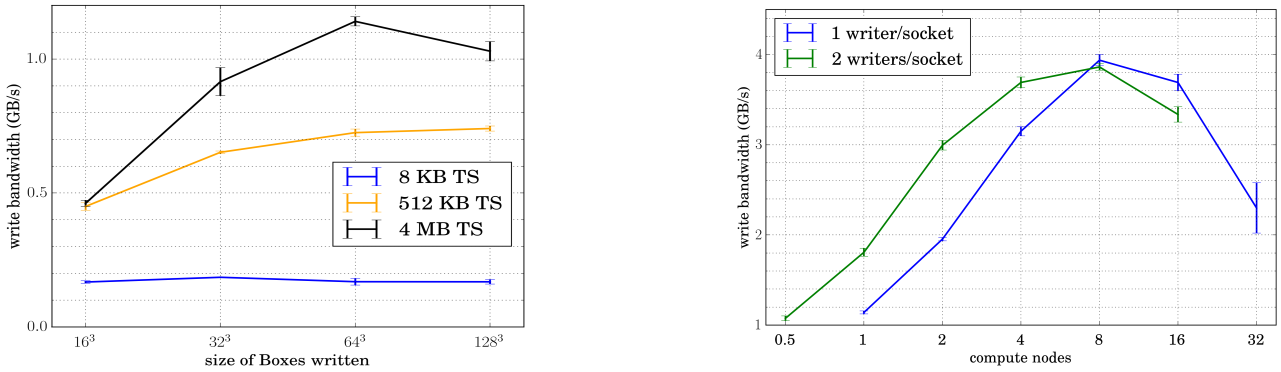


Fig. 5. Volume rendering of a Nyx calculation at redshift z=3. The problem domain spans 10 Mpc per side, discretized on a grid of size 512^3 points. The baryonic matter collapses into filaments, shown in blue. The green spheres indicate the centroids of dark matter halos which surround the filaments.

A. Nyx/Boxlib

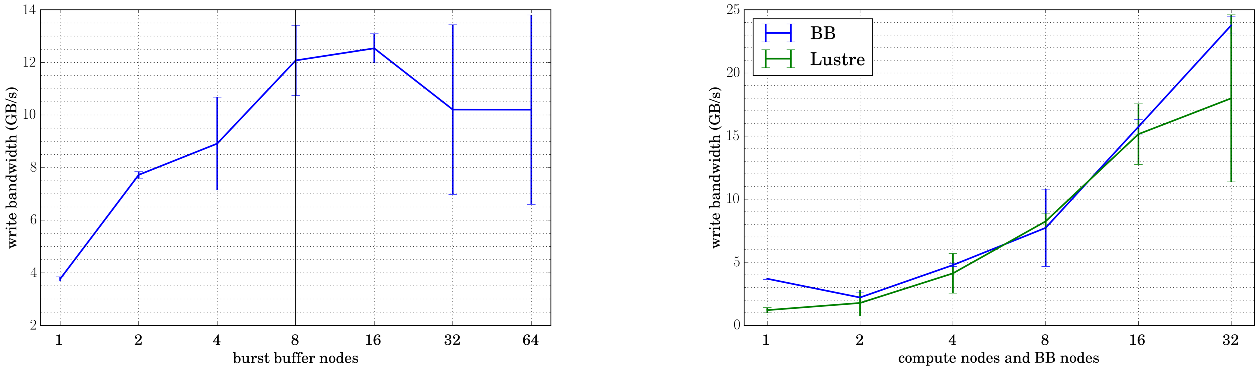
1) Science Motivation: Cosmology is the study of the universe as a whole - its contents, history and expansion. Cosmological theory makes predictions about the state of the universe under certain conditions - the amount of regular (baryonic) matter; the amount of dark (non-baryonic) matter; and the amount of dark energy (the as-yet unknown force that is driving the accelerated expansion of the universe). These theories are tested in simulations of miniature universes, which can be used to make predictions about observables that can



(a) IO performance of Nyx using different transfer sizes, for different Box size. This simulation used 2 MPI ranks per compute node, and one Burst Buffer node. Blue: small (8KB); Yellow: mid-size (512KB) ; Black: large (4MB) transfer sizes (TS).

(b) IO performance using either one or two MPI writers per socket driving the IO on a single Burst Buffer node, for different numbers of compute nodes running the simulation. 0.5 compute nodes refers to use of 1 socket on a compute node.

Fig. 6. Maximizing write bandwidth to a single Burst Buffer node using Nyx.



(a) IO performance of Nyx for different numbers of Burst Buffer nodes, for a simulation using 8 compute nodes, each with 2 MPI writers. The vertical black line denotes 1 Burst Buffer node for each compute node.

(b) IO performance of Nyx on the Burst Buffer and on the Lustre PFS, fixing the ratio of compute to Burst Buffer nodes to 1:1. Individual files on Lustre were not striped across OSTs, which is appropriate for the relatively small files, but they will be distributed across OSTs.

Fig. 7. Scaling up Nyx performance to more Burst Buffer nodes, and more compute nodes.

be measured from the real universe we see around us. For example, the statistical distribution of galaxies predicted by theoretical simulations can be compared to that observed by astronomical survey telescopes.

Nyx [10] is a cosmological simulation code based on a widely-used adaptive mesh refinement (AMR) library, BoxLib [11]. Many theoretical models in cosmology run simulations of a universe that contains only dark matter, neglecting the baryonic matter contained in gas, dust, stars and galaxies. This makes the simulations easier to run, since dark matter can be assumed to interact only via the gravitational force, whereas baryonic matter also interacts via complex electromagnetic processes. Modeling the behavior of baryonic matter in codes such as Nyx allows cosmologists to simulate observable phenomena accurately, such as the Lyman-α Forest [12]

The largest cosmology simulations evolve from the early universe (redshift z=150, roughly 900,000 years after the Big Bang) to the present day, which takes thousands of time steps. The data files (”plotfiles”) written at certain time steps can be large, O(TB), and the checkpoint files even larger. I/O time can therefore consume a significant fraction of run time in these large simulations - so writing the plotfiles and checkpoint files to the Burst Buffer offers a large potential compute time savings.

2) Project Workflow: The Nyx simulation code starts by reading either a checkpoint file or configuration file containing the initial conditions of the matter (both baryonic and dark). This model universe is then evolved using a hybrid N-body/compressible gas dynamics algorithm. The properties of the baryonic matter is calculated at the AMR mesh points, whereas the dark matter particles are treated individually. The input files and the Nyx executable are staged in to the Burst

Buffer. The compute job then runs entirely off the Burst Buffer, writing output plotfiles and checkpoint files to the space allocated.

Nyx is generally run on Cori in a configuration of two MPI processes per compute node with 16 threads each - i.e. one MPI process per socket. We vary the number of writers per MPI process to determine how to best drive the Burst Buffer bandwidth. For the studies described here, each MPI process writes separate plotfiles. For these tests mesh data was written but we disabled writing particle data to avoid potential load imbalance, since particles can coalesce on different compute nodes. The simulation size used was 256^3 mesh points, decomposed into sub-volumes known as Boxes. Each plotfile was around 1.8GB; future work will investigate the optimal configuration for writing out checkpoint files.

3) Results: Our first test consisted of a single compute node writing plotfiles at each of six consecutive time steps to a single Burst Buffer node, with two concurrent file streams issued to one Burst Buffer node. The results are shown in Figure 6(a). The performance achieved strongly depends on the transfer size used for the write - larger transfer sizes (at least 512 KB) are required to achieve good IO bandwidth. At large Box sizes Nyx can achieve 1.1 GiB/s for large transfer sizes.

However, this is still below the peak write bandwidth of a single Burst Buffer node, which is around 6 GiB/s (see Section II-D). Two write instruction streams coming from a compute node is not enough work to saturate the Burst Buffer node bandwidth. This is confirmed in Figure 6(b), which shows the IO bandwidth to a single Burst Buffer node when the IO is driven by different numbers of MPI writers. Using more than 16 MPI writers (i.e. one writer per socket on eight compute nodes, or two writers per socket on four compute nodes) gets close to the peak theoretical write bandwidth.

We next scale up the number of Burst Buffer nodes used in this configuration, using 16 MPI writers over 8 compute nodes, but increasing the number of Burst Buffer nodes. This assesses the benefit of striping files over multiple Burst Buffer nodes to attain better bandwidth. Figure 7(a) shows application I/O bandwidth approximately doubles when using two Burst Buffer nodes, but the scaling does not continue above this point, indicating that the limited size of the simulation is not providing enough data to fully drive the Burst Buffer IO bandwidth. The best performance is achieved when using one Burst Buffer node for every compute node, however, using this ratio for the described Nyx configuration leaves significant Burst Buffer performance potential, as well as capacity, unused.

Finally, we scale up the simulation, matching the number of compute nodes with the number of Burst Buffer nodes. Figure 7(b) shows that performance scales almost linearly. This figure also shows a comparison of performance when writing to the Lustre PFS, and demonstrates that the Burst Buffer performs as well as Lustre, and indicates that it will scale better than Lustre when the number of compute nodes is increased further.

These tests highlight competing interests in the overall performance of Nyx. On one hand, the floating-point performance of Nyx is optimal when we use a small number of MPI processes (ideally one per NUMA domain) and a large number of threads. However, this leads to a small number of I/O instructions streaming from compute nodes to Burst Buffer nodes, which, as we have seen, underutilizes the available Burst Buffer bandwidth. This can be compensated by using a large ratio of compute nodes to Burst Buffer nodes, i.e. 16 MPI writers per Burst Buffer node, which can drive the maximum available bandwidth.

4) Future Work: Analysis of the plotfiles that result from the Nyx simulations also require significant computing time. This could be reduced by running the analysis code over the plot file as they are written to the Burst Buffer, simultaneously with the simulation code that creates the plotfiles. This example of using the Burst Buffer to enable a coupled workflow is being actively developed. Future work will also include tuning Nyx configuration for writing checkpoint files.

B. Chombo-Crunch + VisIt

1) Science Motivation: CO_2 is one of the largest contributors to global climate change. One proposed method to reduce the amount of CO_2 entering the atmosphere is to inject the CO_2 into the Earths subsurface and trap it there. However, this process affects the porous soil and rock that comprise Earths surface and subsurface, so more insight must be gained into the short and long term effects of this process, via experimentation and computational simulation, before it can be considered a safe and viable solution for the future.

The disruption caused by carbon injection forces the surface/subsurface system to come out of equilibrium both chemically and mechanically. This results in a nonlinear dynamical regime in which emergent behavior develops at several different scales. These nonlinear interactions between multiphase flow, solute transport, mineral dissolution, and precipitation have been addressed at the larger scale where soil grains and pores are not resolved, but have been largely neglected at the pore scale where the chemical and physical environments undergo strong variations locally in space and time. The key fluid-fluid (H_2O and CO_2) and fluid-solid interfaces need to be resolved in order to understand how to control CO_2 injection in the subsurface. By carefully modeling processes at the pore scale, the overall goal of this research is to bring such knowledge to bear on the macroscopic scale of a CO_2 reservoir.

2) Project Workflow: Chombo-Crunch [13], [14] is an MPI-based simulation software for modeling subsurface flow and reactive transport processes associated with carbon sequestration over time. The simulation generates data in the form of a single `.plt` file per time step, i.e., all MPI ranks contribute the data they computed to a single shared file. The file management over the MPI ranks is handled by HDF5 [15], however, it is important to note that these `.plt` files can vary from O(10) GB up to O(1) TB in size depending on the resolution of the simulation. Depending on the number of time steps that the simulation runs for, the aggregate data size associated with the simulation can easily exceed hundreds of terabytes.

The other component application for this workflow is VisIt [16], a visualization and analysis tool for scientific data. In this workflow, the role of VisIt is to read each `.plt` file as it is generated by Chombo-Crunch and begin performing data analysis on it. The result is a `.png` file representing a movie frame. These frame files can then be compiled into a movie using third-party encoding software, such as `ffmpeg`. The combined Chombo-Crunch and Visit workflow, incorporating the Burst Buffer is shown in Figure 8.

This workflow is too IO-intensive for traditional disk-based storage. Traditionally, Chombo-Crunch would run independently of VisIt, writing each file to the Lustre PFS - VisIt would then run on the stored files. There are two major bottlenecks to scalability of the workflow. First, the Chombo-Crunch processes cannot progress to the next step until the `.plt` file is finished being written. Checkpoint files written by the Chombo-Crunch simulation also blocked the simulation from progressing until they were finished writing. Second, VisIt then reads these file from the PFS into node-local memory. It is worth noting that the end-result movie is of scientific interest, and the intermediate `.plt` or `.png` files need not be retained.

By using the Burst Buffer, Chombo-Crunch and VisIt can write and read their files to/from this intermediate storage layer for faster data exchange. In the modified Burst Buffer workflow, VisIt runs in conjunction with Chombo-Crunch and can read `.plt` files directly from the burst buffer as soon as they are generated. In addition, VisIt can write the resultant `.png` files to the Burst Buffer. These image files are staged out to the PFS at the end of the job.

3) Results: Figure 9 shows the bandwidth and scaling achieved by Chombo-Crunch in writing data to the Burst Buffer and to the Cori Lustre PFS. The number of compute nodes is doubled in each case starting with 16 compute nodes writing to a single Burst Buffer node, up to 1024 compute nodes writing to 64 Burst Buffer nodes. This corresponds to an increase in simulation resolution. Although the bandwidth achieved is around a quarter of the peak bandwidth per Burst Buffer node (6GB/s), the application is using HDF5 collective I/O [17] which is currently unable to reach peak file-per-process I/O rates (as discussed also in more depth for the VPIC-IO case in Section IV-C). The Lustre results used a 1MB stripe size and a stripe count of 72 OSTs in all cases. The Burst Buffer significantly out performs Lustre for this application at all resolution levels and the bandwidth scales exceptionally well.

We then ran the full Chombo-Crunch and VisIt workflow for a ‘packed cylinder’ problem [14]. This Chombo-Crunch simulation ran on 8192 cores over 256 nodes with 8 further nodes used for VisIt. It used the full Burst Buffer available at the time, around 140 nodes and achieved a bandwidth of 90.7GB/s. Figure 10 shows the movie assembled from this workflow, demonstrating a coupled science workflow using the Burst Buffer for the first time.

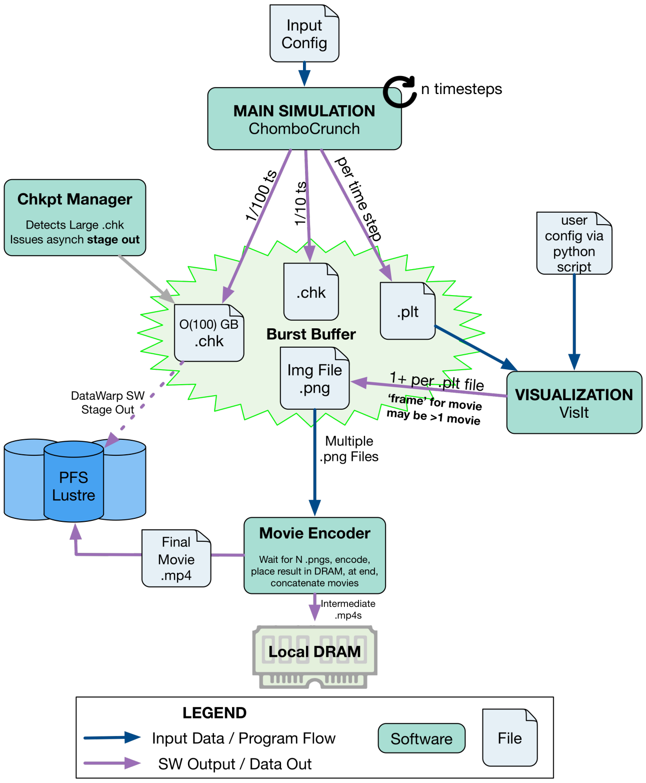


Fig. 8. Chombo-Crunch + VisIt workflow

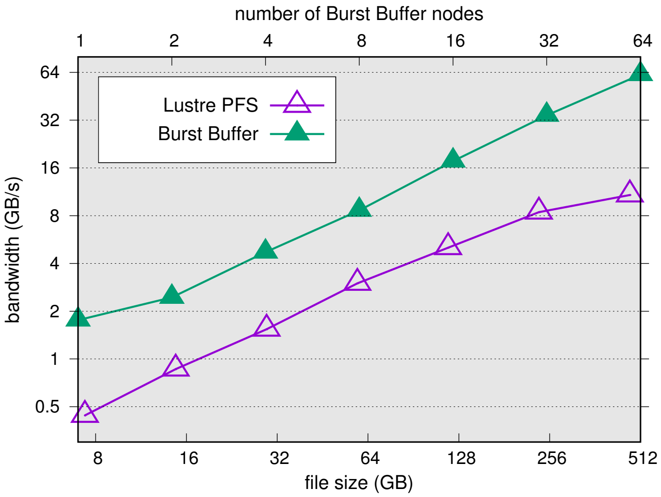


Fig. 9. Chombo-Crunch I/O bandwidth scaling.The compute node to Burst Buffer node ratio is fixed at 16:1

4) Future Work: As shown in Figure 8, checkpoint files are written out every 10 timesteps. This can be done by the main Chombo-Crunch application, however in-order to further improve the run time, the DataWarp C API will be used to stage out the files to the PFS asynchronously without interrupting the workflow. Furthermore the movie produced by the workflow will also be produced in a coupled application running on the input files in place.

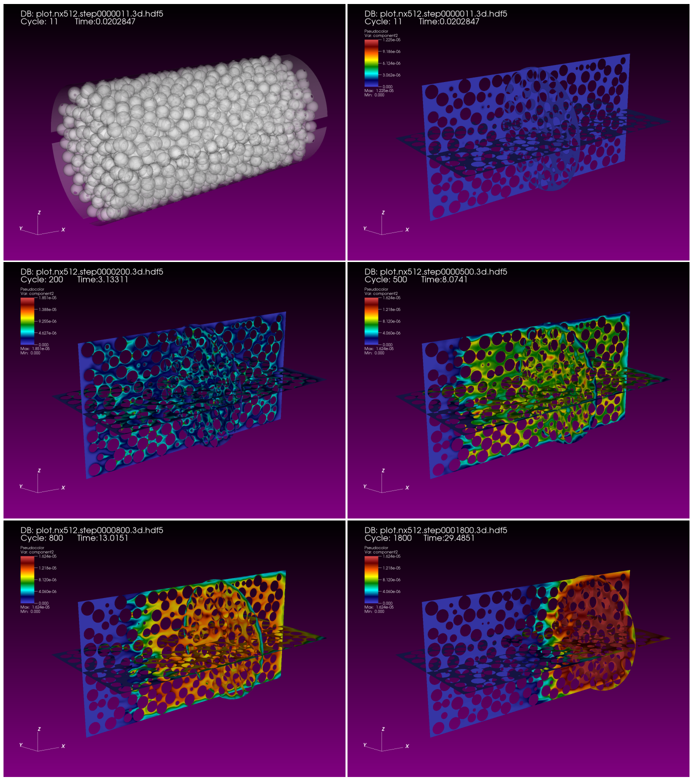

DB: plot.nx512.step0000011.3d.hdf5
Cycle: 11 Time:0.0202847
DB: plot.nx512.step0000011.3d.hdf5
Cycle: 11 Time:0.0202847
DB: plot.nx512.step0000200.3d.hdf5
Cycle: 200 Time:3.13311
DB: plot.nx512.step0000500.3d.hdf5
Cycle: 500 Time:8.0741
DB: plot.nx512.step0000800.3d.hdf5
Cycle: 800 Time:13.0151
DB: plot.nx512.step0001800.3d.hdf5
Cycle: 1800 Time:29.4851

Fig. 10. Snapshots from movie produced by a 'packed cylinder' [14] Chombo-Crunch and VisIt workflow. Full movie is available from http://portal.nersc.gov/project/mpccc/wbhimji/BurstBuffer/packedCylinder.mp4

C. VPIC-IO

1) Science Motivation: Plasmas are clouds of unbound positively and negatively charged particles, which play an important role in many astrophysical phenomena, from stars to supernovae to the interstellar medium. Because plasmas contain many charge carriers, they are particularly sensitive to electromagnetic processes. Magnetic reconnection is one such process, and occurs when the magnetic field within a plasma rearranges itself - which can be accompanied by a huge release of energy. For example, magnetic reconnection is thought to be the cause of solar flares from the Sun, and magnetic reconnection in the Earth's magnetosphere produces the aurora.

Magnetic reconnection is inherently a multi-scale problem. It is initiated at small scale around individual electrons in the plasma but eventually leads to a large-scale reconfiguration of the full magnetic field. Recent simulations have revealed that electron kinetic physics is not only important in triggering reconnection [18], but also in its subsequent evolution. Modeling the detailed electron motion requires 3D simulations of reconnection in plasmas consisting of large numbers of particles, which poses severe computational challenges. The combination of advances in particle simulations and large-scale supercomputers are now enabling simulations of trillions of particles. Fueled by the new capabilities of these highly-optimized simulations, supercomputers have been providing the first glimpses of collisionless reconnection in 3D space.

2) Project Workflow: The VPIC code performs shared file I/O using the HDF5 library. Simulations of trillions of particles involve massive amounts of data. For example, a recent simulation of magnetic reconnection involving two trillion particles (one trillion electrons and one trillion ions) using the VPIC code [19] produced data files in the range of 32TB to 40TB in size per time step [20]. The simulation runs for tens of thousands of timesteps to understand the evolution of electron motion and energy.

The fast I/O offered by the Burst Buffer promises to accelerate writing-out of data for each time step. The data can then be moved asynchronously to the PFS. In addition, while the data is in the Burst Buffer, in-transit analysis can be performed efficiently avoiding data reads from the PFS. Figure 11 gives an example of this kind of visualization, showing the mapped magnetic field for highly energetic particles in physical space.

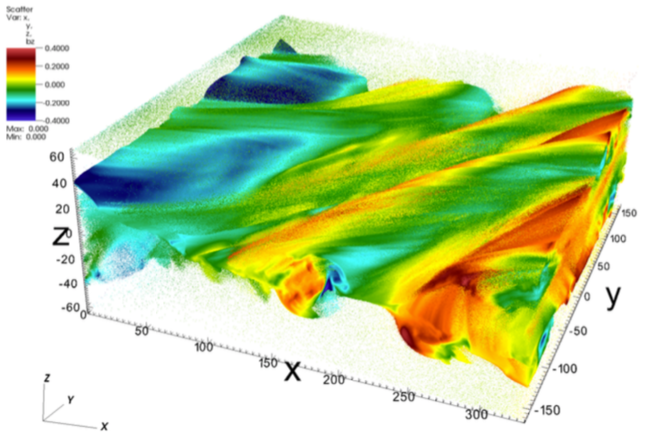


Fig. 11. Mapped magnetic field data in the z dimension (bz) for all highly energetic particles (Energy>1.5) in physical space (x, y, and z dimensions). The plots use a linear color scale. The range is restricted to [0.4, 0.4] for bz.

3) Results: We compared running VPIC's I/O kernel in various configurations accessing the Burst Buffer and Cori's Lustre PFS, shown in Figure 12, over a range of compute scales. The I/O time includes the times for opening, writing, and closing the HDF5 file. Typically, the times for opening and closing are negligible.

We have compared the I/O rate of VPIC-IO kernel for writing data to the Lustre file system on Cori with three configurations of MPI-IO in writing data to the burst buffer. In the first configuration, 65 Burst Buffer nodes were used which did not match the MPI-IO aggregators. In MPI-IO, in order to reduce the number of writers, aggregator processes are used to collect the data and to initiate writing only from those aggregated processes. This is called *two-phase I/O* or *collective buffering* in the MPI-IO implementation.

In the second configuration, we have selected 64 Burst Buffer nodes to match the number of aggregators. This is similar to the MPI-IO collective buffering in CB2 mode, where

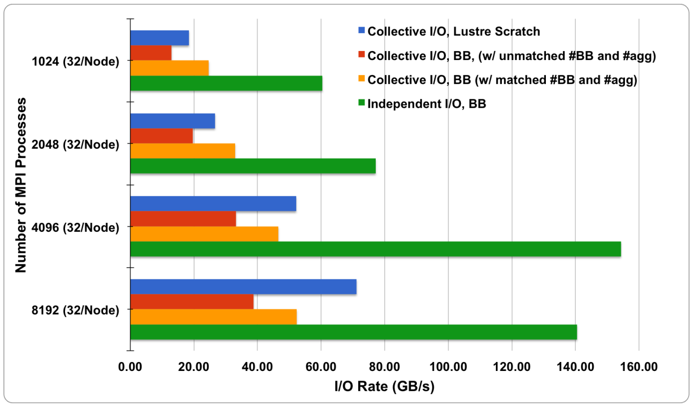


Fig. 12. Performance of the VPIC-IO kernel, comparing different scales and configurations of MPI-IO. Blue: I/O bandwidth for MPI collective I/O using the Lustre PFS; Red: MPI collective I/O using the Burst Buffer and the number of aggregators not matching the number of Burst Buffer nodes; Yellow: matching the number of MPI aggregators; Green: using independent I/O on the Burst Buffer.

the number of aggregators is an integer multiple of the number of storage devices.

In the third configuration, we have disabled collective buffering, i.e., each process initiates writing data, called *independent I/O* mode in MPI-IO.

Our evaluation demonstrates that the imbalance caused by having a number of Burst Buffer nodes that were not matched to the number of MPI aggregators deteriorates I/O performance (shown in the red bar in figure 12). By matching the number of aggregators to the number of the Burst Buffer nodes, we observed that the I/O rate is 60% faster on average, which demonstrates the importance of balancing the load on the DataWarp servers - if two compute nodes are accessing the same Burst Buffer node then their accesses will be competing for bandwidth, resulting in significantly degraded performance for the entire compute job. This imbalance was confirmed by implementing monitoring of I/O counters for the DVS service on the compute node.

Despite the improvement with matching aggregators, we observe that performance of VPIC-IO with MPI-IO collective buffering in writing data to the Burst Buffer is not always superior compared to the Lustre file system. Previous work has shown that collective buffering performs better than independent I/O on disk-based file systems as a few MPI aggregator processes write large chunks of contiguous data, reducing the number of seeks [21], [22]. However, collective I/O has an overhead on the compute nodes as the aggregators have to synchronize write operations. The green bar in Fig. 12 shows a a significant performance gain using independent I/O with the Burst Buffer, since SSDs perform better for random seeks than the disks. Compared to the collective I/O performance on Burst Buffer with matched number of aggregators, the independent I/O mode performs 4× better. This I/O rate with Burst Buffer is also up to 3.5× better than that of the Lustre PFS. In order to explore the overhead of collective I/O on Burst Buffer in more detail, we profiled VPIC-IO using the Darshan I/O tool [23] and determined that for a run over 20 Burst Buffer nodes using collective I/O, the underlying POSIX I/O rate was 27 GiB/sec (around 1.3 GiB/sec/node), but the MPI-IO rate was 10 GiB/sec. Around two-thirds of performance was lost in the MPI-IO layer.

This was further confirmed by mimicking VPIC-IO using the IOR benchmark with three APIs: HDF5, MPI-IO, and POSIX. This configuration used 64 compute nodes and 64 Burst Buffer Nodes with collective I/O enabled, and achieved the I/O rates shown in Table III, again showing a significant overhead for MPI I/O.

TABLE III
BANDWIDTH FOR AN IOR RUN CONFIGURED TO BROADLY MATCH VPIC-IO AND SHOW MPI I/O OVERHEAD, USING 64 COMPUTE NODES AND 64 BURST BUFFER NODES.

API	Max(GiB/s)	Min(GiB/s)	Mean(GiB/s)
HDF5	14.8	14.6	14.7
MPIIO	15.7	15.0	15.4
POSIX	66.9	66.1	66.5

4) Future work: We are working with Cray developers to understand and improve MPI collective I/O as well as considering if the layout of HDF5 files on the Burst Buffer can yield further improvements.

D. TomoPy and the ALS Spot Suite

1) Science Motivation: Tomographic reconstruction (as used with CT or computed tomography scanners in hospitals) creates a three-dimensional image of an object based on a series of two-dimensional projection images taken from multiple directions. Getting results from tomographic reconstruction very quickly is essential for researchers in order to monitor data quality as well as to inform the next experimental steps when performing time-resolved experiments. In most cases, the raw 2D data is not sufficient to give the information needed for this. Lightsource science, such as that conducted at the Advanced Light Source (ALS) and Advanced Photon Source (APS), puts heavy value on time-to-knowledge.

TomoPy [24] is an open-sourced Python toolbox to perform tomographic data processing and image reconstruction tasks. TomoPy could be used by multiple beamlines at the ALS.

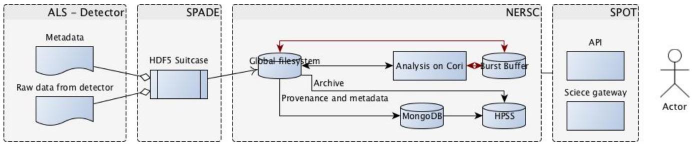


Fig. 13. SPOT suite used for processing data from the ALS at NERSC.

2) Project Workflow: TomoPy is run on data collected at the ALS and the APS. These datasets are packaged in HDF5 form [25] and transferred from the beamline experiments to the filesystems at NERSC for processing. This can be performed in an integrated workflow using the SPOT suite [26] (see Figure 13) which runs analysis code on compute resources at NERSC such as Cori. The results of the analysis can then be explored using web portals at NERSC (as in figure 15).

In order to ensure prompt turnaround for compute resources, use is made of the ’realtime’ queue at NERSC [27] which uses the SLURM workload manager on Cori to provide access to a pool of dedicated resources and priority on the compute nodes. Use of the Burst Buffer can further improve time to insight for these workflows. In this project the Burst Buffer is integrated into this workflow and TomoPy is run on images staged into the Burst Buffer rather than the Lustre PFS.

The tomographic reconstruction part of the pipeline can be run in different ways and current approaches store intermediate results to disk which can be IO intensive. TomoPy has been designed to avoid writing intermediate data to disk and also to optimize handling of operations on data in memory. Nonetheless more than 30% of the application runtime is spent in I/O.

3) Results: In Figure 14 we show a comparison of the total time for various I/O operations performed by TomoPy on files stored on the Burst Buffer and the Lustre filesystem. The analysis is run on 4 nodes of the ‘realtime’ queue at NERSC using 9.9GB of input data. 500 GB of scratch Burst Buffer space in striped access mode was requested. For this run it can be clearly seen that running the analysis on data stored on the Burst Buffer outperforms that on Lustre, however other runs on production data show more variation in results that still require further investigation.

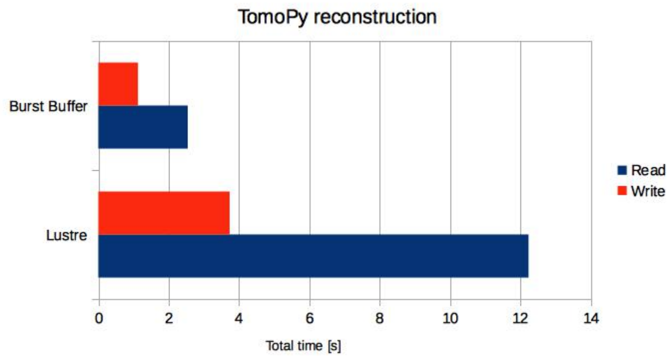
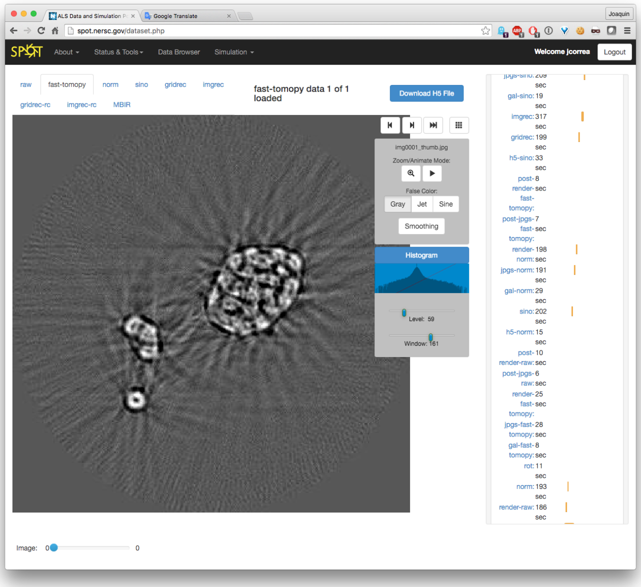


Fig. 14. Total time for read and write operations by TomoPy running on 9.9 GB of input data on 500 GB Burst Buffer allocation.

Figure 15 shows a snapshot of one SPOT interface showing the experimental data pipeline incorporating a TomoPy analysis on the Burst Buffer. This is the first coupling of Burst Buffer technology into an experimental science workflow.

E. Further work

We have integrated the Burst Buffer with TomoPy and experimental ALS workflows, seeing clear I/O time advantages in some cases. This study is being repeated for other datasets and analyses to ensure the Burst Buffer is optimal in all cases. In addition high availability of the Burst Buffer system will need to be demonstrated (see also section V) to make use of it by default in production.



fast-tomopy data 1 of 1 loaded

img0001_thumb.jpg

Level: 59

Window: 161

Image: 0 0

jpgs-sino: 209 sec
gal-sino: 19 sec
imgrec: 317 sec
gridrec: 199 sec
h5-sino: 33 sec
post-render-fast-tomopy: 8 sec
post-jpgs-fast-tomopy: 7 sec
render-norm: 198 sec
jpgs-norm: 191 sec
gal-norm: 29 sec
sino: 202 sec
h5-norm: 15 sec
post-render-raw: 10 sec
post-jpgs-raw: 6 sec
render-fast-tomopy: 25 sec
jpgs-fast-tomopy: 28 sec
gal-fast-tomopy: 8 sec
rot: 11 sec
norm: 193 sec
render-raw: 186 sec

Fig. 15. A screenshot of the SPOT webportal at NERSC, illustrating both the output of TomoPy and how it is running in the production pipeline using the Burst Buffer.

F. ATLAS: simulations and data analysis for the ATLAS detector at the LHC

1) Science Motivation: The ATLAS detector at the Large Hadron Collider (LHC) at CERN is designed to search for new fundamental particles and forces of nature at the highest energies ever produced by mankind. The LHC collides together protons at the centre of the ATLAS detector 40 million times a second. ATLAS is a complex instrument with over 100m channels of electronics, so this results in a considerable data rate (initially PB/s) which must be filtered by fast electronics on the detector followed by considerable computing offline. New discoveries, such as the recently discovered Higgs Boson, are often rare signals within a large amount of background events. An example of a filtered signal candidate event is shown in figure 16. Furthermore both signal and background physics must be simulated with a computationally expensive simulation of the underlying physics and the response of the detector. Collected and simulated data, even after initial filtering, amounts to hundreds of petabytes. To process this, ATLAS makes use of distributed computing resources in the Worldwide LHC Computing Grid. Until recently however we were not able to make use of large supercomputers at facilities such as those at NERSC.

Use of Cray machines at NERSC is now being made possible by Yoda [28]. Yoda packages ATLAS computing tasks into highly-parallel workloads suitable for running on supercomputers; it is now used in regular production for ATLAS. However due to the I/O limitations of previous machines, it has been restricted only to running the least I/O intensive of ATLAS workloads. The Burst Buffer will enable ATLAS to run all workloads, including the most I/O intensive ones, on

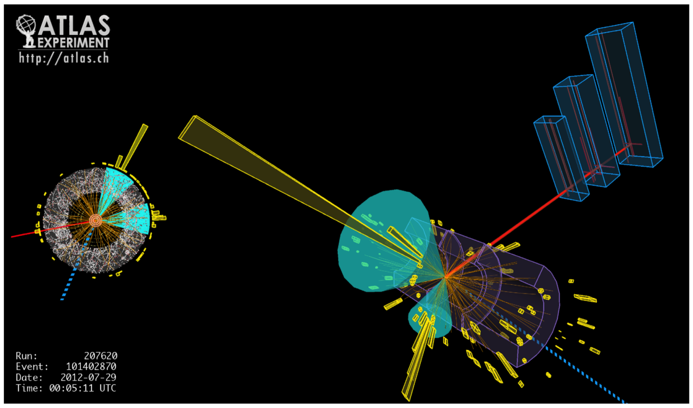


Fig. 16. Example Higgs (decaying to b-quarks) signal 'event' in the ATLAS detector.

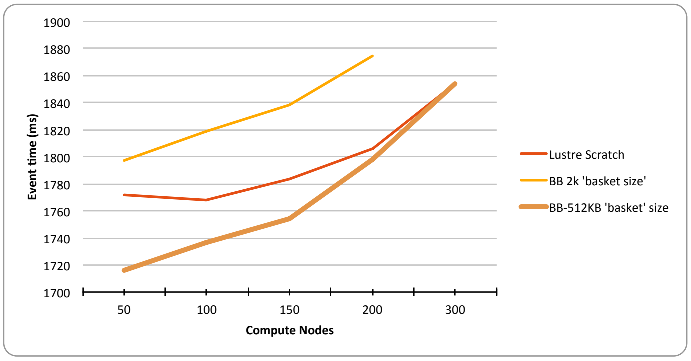


Fig. 17. Scaling of total time per 'event' for ATLAS simulation for the Lustre Scratch filesystem and the Burst Buffer with both the default 2k 'basket' (transfer) size and a 512k basket size. This time also includes compute time but this is constant for the different storage backends.

Cori. These workloads are currently run on a separate compute cluster at NERSC (called PDSF), so enabling these to be run on the large Cray machines will be an important achievement for the direction of NERSC computing, and more generally for the use of HPC facilities by experimental science.

2) Project Workflow: Yoda runs one MPI-rank per compute node. The responsibility of Rank 0 (the master) is to dynamically distribute workload (events) between other ranks (the slaves). Each of the slaves occupies the entire compute node by running a parallel version of the ATLAS event framework, which actually processes the events and writes out separate output files to disk. Currently, as mentioned above, Yoda is used primarily for running event simulation and that is the first application we consider here. This application is relatively computationally expensive.

As well as simulation workflows, the Burst Buffer also allows the most I/O intensive 'analysis' applications to run on Cori. These run filtering of data to find rare events and would require the ability to read their highly compressed input data at rates of around 500 GB/s to run at full machine scale on Cori. The read pattern can also be somewhat complex depending on the different types of analyses run. For this study we run on a 475G dataset of Atlas data in compressed 'xAOD' ROOT format [29] [30] starting with a single node fully occupied with 32 processes and using a 2 TB Burst Buffer allocation.

3) Results: Figure 17 shows the average time taken for simulating an event of the ATLAS simulation described above using the Yoda framework and scaling up to 300 compute nodes (9600 cores). This time includes both the compute time and the I/O time. As mentioned above this step is relatively compute intensive but the particular physics was chosen to provide the more relative I/O than most runs. The I/O corresponds to around 7% of the run time on Lustre. As shown Lustre initially outperforms the Burst Buffer (smaller total time per event). However the default configuration of the run was to use a 'basket' size for the underlying ROOT I/O layer of 2 KB. This corresponds to the amount kept in memory before compression and flushing to disk. Increasing this to 512 KB, as seen for other projects, improves the write performance to the Burst Buffer which then outperforms Lustre at low node concurrences.

Figure 17 also shows that this workflow can scale well to 300 nodes. However initial runs used the Burst Buffer as a working directory for all files including many small log files. When this was done it was repeatedly seen that the at scales of 300 (or greater) nodes the event time for the Burst Buffer considerably increased to around 2500 ms even with the larger basket size. This scaling issue was not seen with the working directory held on Lustre. This was resolved for the results shown above by moving the log files to Lustre and only using the Burst Buffer for the data output. However it illustrates possible scaling issues for small files on the Burst Buffer.

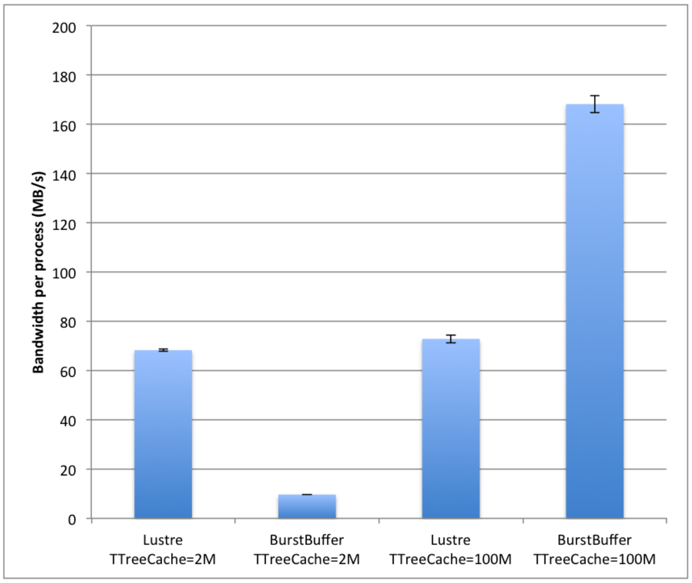


Fig. 18. Bandwidth per process for the ATLAS analysis application with the default 'TTreeCache' size of 2M and an increased cache of 100M.

In Figure 18 we show the average bandwidth per process for reading data in the ATLAS 'analysis' application. The initial run showed significantly worse performance for the Burst Buffer than the Lustre filesystem. Further investigation showed that the application was making over 2 million read calls. ROOT allows for use of a configurable 'TTreeCache' that 'learns' what data 'branches' are being used and prefetches these into a memory cache [30]. By default this cache size was set to a very low value in the application (2.5 MB). Increasing the cache size to 100 MB significantly improves the Burst Buffer performance which then out-performs Lustre. This clearly illustrates again that client-side caching in Lustre masks the challenge of small transactions but that Burst Buffer can perform well for larger transactions.

G. Further work

We have demonstrated that the Burst Buffer can be used at scale for production ATLAS simulation workflows and also that, with the use of an application memory cache, we can also run the more challenging analysis workflows with better performance on the Burst Buffer than on Lustre. The analysis workflows will be integrated into the Yoda framework in order to run these in production at at the larger scales that has already been demonstrated here for simulation.

Additionally, a second step of the simulation workflow involves merging of the output files and is largely I/O bound. Currently this is not performed at NERSC but instead output files produced by the first step are staged out to a distributed external storage (object store) for final merging at compute resources elsewhere on the Grid. However the Bust Buffer should allow this part of the workflow to also run as a separate coupled application with the data in place.

Once all ATLAS workflows, including those with heavy I/O, are demonstrated at scale with the Burst Buffer it will be possible to enable the bulk of ATLAS workflows on the Cori supercomputer instead of relying on separate computational resources for this community.

V. Challenges

The NERSC Burst Buffer Early User program was very effective in immediately revealing bugs and limitations in the available software. This was crucial given it is the first time that the Cray Burst Buffer and DataWarp software has been used at large scale and for a wide variety of applications. Many of these issues were quickly rectified by the Cray and SchedMD developers, benefiting from the close collaboration of the NRE arrangement. Others will require longer term development which is being folded into Stage 2 of the DataWarp software. These issues included:

- posix_fadvise calls against DataWarp mounts induced compute node kernel panic: This was determined in the first week of the early-user program by a user employing 'tar -x'. A patch was promptly provided by Cray.
- stat/fstat system calls on the DataWarp mounts, returned a sub-optimal value for st_blksize: This led to 'cp' (and other applications) using very small transfer sizes and therefore performing poorly. This was again discovered very early on by users and various workarounds could be employed until a patch was installed.
- Use of the DVS_MAXNODES environment variable (used for GPFS DVS mounts at NERSC) prevented DataWarp from striping across more than one BurstBuffer node: This was resolved by preventing this variable from being passed through in the central scripts executed for DataWarp mounts.
- Stage-in failed for large numbers of files ($\gtrsim$7000 files, depending also on file sizes): This was due to timeout values in the REST API communication. Currently we are employing a workaround of increasing timeout values. Re-engineering is underway by Cray to improve performance.
- Limits to the number of open files on the underlying filesystem: This led to, for example, jobs that loaded very large numbers of shared libraries failing with 'No such file or directory' errors. A workaround is in place of increasing the sysctl 'fs.file-max' parameter. Re-engineering is underway to reduce number of files particularly once the number of substripes is increased.
- The 'granularity' size which governs how large an allocation must be before it is striped across multiple nodes, cannot be lowered below around 200 GB: This means that users who do not require more space either do not get striped file performance, or must occupy more space then needed which will increasingly become a problem as more users are permitted on the system. This will be fixed in future releases of the DataWarp software.
- Lack of client side caching for Burst Buffer mounts: This means that in the cases of small (sequential) transactions and re-reads of data recently written, the Burst Buffer significantly under-performs relative to Lustre which does have such caching. DVS client-side caching is expected in future releases of DataWarp software available in late-2016.
- MPI-IO Shared file performance: As noted in Section II-D MPI-IO shared file performance does not meet expectations. Cray engineers have demonstrated that this can be improved with increased sub-stripe counts. However metadata handling improvements are also required to allow this increase. This limitation will be addressed in future software releases.
- Kernel panics on Burst Buffer nodes: Two issues that caused kernel panics on Burst Buffer nodes were discovered: the first has been patched by Cray; the second is a "Null pointer" exception that in understood by Cray and fixed, but not yet backported to NERSC's version of the software.
- DVS Client hang: This causes compute nodes to be held in a "completing" state in the SLURM WLM while Burst Buffer nodes cannot unmount the DataWarp filesystem due to a "busy" thread. This has been identified as an XFS deadlock and is under investigation by Cray.

VI. Lessons Learned

As well as the challenges noted in the last section, a range of valuable lessons have been learned by the science applications discussed here. Here we summarize some of these and also note some general themes.

1) **Early User programs are a very useful way to debug complex new technologies.** As shown in the last section, a large number of performance improvements and fixes to scaling limits, operational problems and usability issues have occurred because of the dedicated early user program. This would have been difficult to achieve with synthetic tests or with large numbers of generic users.
2) **The Burst Buffer provides a high-performance solution for large streaming I/O** . We show here, particularly in the Nyx and Chombo-Crunch applications, that large block transfers can work well with the Burst Buffer.
3) **The Burst Buffer enables coupled workflows**. The Chombo-Crunch plus VisIt workflow illustrates this for visualization. The SPOT suite and TomoPy workflow shows this can also be done for experimental science workflows. This activity is just beginning, but now we have demonstrated that there is considerable potential for exploring new approaches to workflows.
4) **More challenging I/O patterns currently have a mixed performance** . We can see here in the TomoPy and tuned ATLAS analysis applications that there are benefits for some more challenging I/O patterns. This is also seen in the random IOPS benchmark. However small files and varied I/O patterns are also still performing poorly, for example in the default ATLAS use case or with smaller transfer sizes for the Nyx application.
5) **MPI-IO with Burst Buffers will require further tuning to perform well.** As shown particularly in the VPIC-IO study, there is considerable work that could be done to improve performance to the level the system is capable of. The Cray MPI used here did not contain any MPI-IO optimizations for DataWarp, whereas it did contain several years worth of optimizations for a Lustre PFS. A complicating factor is that MPI-IO does contain optimizations for DVS which are active for these tests, but these do not necessarily improve DataWarp performance, so the interplay between these factors will need to be understood. Much of this work is underway and the use of real applications like those shown here should be an important part of this development.
6) **Tuning of transfer size and number of parallel writers is needed with the Burst Buffer, more so than with Lustre.** Even the use of the Burst Buffer for checkpointing has required tuning of applications in some cases to see performance benefits. Larger transfer sizes have been generally seen to be better (for example in the Nyx application), though 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.
7) **The NERSC Cray DataWarp system now functions quite well**. Having resolved many of the challenges mentioned in the last section we now have a functional system that can run with large scale applications as demonstrated here. This is the culmination of considerable efforts by Cray, SchedMD, NERSC systems and user staff and the early users.

VII. Conclusions

We have described here the Early User Program for the NERSC Burst Buffer. This program has brought real science applications to one of the first large scale 900 TB, on-demand, SSD-based filesystems provided by the Cray DataWarp software.

This program has pushed the frontier of Burst Buffer technology development by uncovering numerous stability, performance, usability and scaling issues. Many of these issues have been addressed, resulting in a fully-functional technology that can deliver I/O acceleration for science.

We have demonstrated that we can:

- Achieve near peak bandwidth for real science applications, such as Nyx.
- Couple simulation and visualization applications using the Burst Buffer as interim storage such as in Chombo-Crunch and VisIt.
- Significantly outperform the underlying Lustre fileystem for applications such as VPIC-IO.
- Accelerate challenging I/O patterns and use the Burst Buffer directly in production experimental data analysis workflows for TomoPy at the ALS and the ATLAS LHC experiment.

However we have also demonstrated that there are a number of remaining challenges, such as:

- Achieving the results above has required significant tuning in some cases. Untuned performance can still be worse than the underlying Lustre filesystem.
- MPI collective I/O significantly under-performs on the current Burst Buffer.

In order to address these, we are continuing to work with Cray on I/O profiling and optimization for these workflows and the other user projects. Considerable work is also underway at NERSC to provide multi-level I/O monitoring.

As these issues become resolved and as the stability of the DataWarp system improves, many of these projects are now moving into a phase where new science can be enabled from the increased I/O capabilities.

With the Phase 2 Cori system (mid-2016), an additional 900 TB of storage will be added to the Burst Buffer pool. Additionally, around this time, Stage 2 of the DataWarp software is also expected to be available, adding transparent caching to the underlying Lustre filesystem.

We expect that these improvements, together with the lessons learned during this program, will provide important direction for future HPC storage as well as accelerating science.

Acknowledgments

The authors would like to thank Tina Declerck and Doug Jacobsen from NERSC, and the Cray on-site staff, in particular Robert Johnson, for systems support of the Cori Burst Buffer; Rei Lee and Yun He at NERSC for being part of the initial project selection; Cray DataWarp developers Dave Henseler and Benjamin Landsteiner for rapid fixes and numerous clarifying discussions; Dilworth Parkinson, Beamline Scientist for the BL8.3.2 Tomography Beamline at the Advance Light Source for data used in the TomoPy study; and the ATLAS Collaboration for data used in the ATLAS study.

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*. IEEE, 2012, pp. 1–11.

[2] Cray. (2016) DataWarp User Guide. [Online]. Available: http://docs.cray.com/books/S-2558-5204/S-2558-5204.pdf

[3] Trinity nersc8 use cases. [Online]. Available: https://www.nersc.gov/assets/Trinity--NERSC-8-RFP/Documents/trinity-NERSC8-use-case-v1.2a.pdf

[4] SchedMD. (2016) SLURM Burst Buffer Guide. [Online]. Available: http://slurm.schedmd.com/burst_buffer.html

[5] Nersc burst buffer example batch scripts. [Online]. Available: https://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/

[6] Cray. (2016) DataWarp Administration Guide. [Online]. Available: http://docs.cray.com/books/S-2557-5204/S-2557-5204.pdf

[7] Wrapfs: A stackable passthru file system. [Online]. Available: http://wrapfs.filesystems.org

[8] S. Sugiyama and D. Wallace. (2008) Cray dvs: Data virtualization service. [Online]. Available: https://cug.org/5-publications/proceedings_attendee_lists/2008CD/S08_Proceedings/pages/Authors/16-19Thursday/Wallace-Thursday16B/Sugiyama-Wallace-Thursday16B-paper.pdf

[9] 2014 nersc workload analysis. [Online]. Available: http://portal.nersc.gov/project/mpccc/baustin/NERSC_2014_Workload_Analysis_v1.1.pdf

[10] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel, "Nyx: a massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, 2013.

[11] (2011) Boxlib. [Online]. Available: https://ccse.lbl.gov/BoxLib

[12] Z. Lukić, C. W. Stark, P. Nugent, M. White, A. A. Meiksin, and A. Almgren, "The lyman α forest in optically thin hydrodynamical simulations," *Monthly Notices of the Royal Astronomical Society*, vol. 446, no. 4, pp. 3697–3724, 2015.

[13] D. Trebotich, M. F. Adams, S. Molins, C. I. Steefel, and C. Shen, "High-resolution simulation of pore-scale reactive transport processes associated with carbon sequestration," *Computing in Science & Engineering*, vol. 16, no. 6, pp. 22–31, 2014.

[14] 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.

[15] The HDF Group. (2000-2010) Hierarchical data format version 5. [Online]. Available: http://www.hdfgroup.org/HDF5

[16] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," in *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, Oct 2012, pp. 357–372.

[17] R. Thakur, W. Gropp, and E. Lusk, "Data sieving and collective i/o in romio," *Proc. of the 7th Symposium on the Frontiers of Massively Parallel Computation*, p. 182189, 1999.

[18] W. Daughton, V. Roytershteyn, H. Karimabadi, L. Yin, B. J. Albright, B. Bergen, and K. J. Bowers, "Role of electron physics in the development of turbulent magnetic reconnection in collisionless plasmas," *Nature Physics*, vol. 7, no. 7, pp. 539–542, Jul. 2011.

[19] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan, "Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation," *Physics of Plasmas*, vol. 15, no. 5, p. 7, 2008.

[20] S. Byna, J. Chou, O. Rübel, Prabhat, H. Karimabadi, W. S. Daughton, V. Roytershteyn, E. W. Bethel, M. Howison, K.-J. Hsu, K.-W. Lin, A. Shoshani, A. Uselton, and K. Wu, "Parallel i/o, analysis, and visualization of a trillion particle simulation," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12, 2012.

[21] B. Behzad, H. V. T. Luu, J. Huchette, S. Byna, Prabhat, R. Aydt, Q. Koziol, and M. Snir, "Taming parallel i/o complexity with auto-tuning," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, 2013, pp. 1–12.

[22] M. Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf, "Tuning HDF5 for Lustre File Systems," in *Proceedings of 2010 Workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS10)*, Heraklion, Crete, Greece, Sep. 2010, lBNL-4803E.

[23] P. H. Carns, R. Latham, R. B. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 characterization of petascale i/o workloads." in *IEEE CLUSTER 2009*. IEEE Computer Society, 2009, pp. 1–10.

[24] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: a framework for the analysis of synchrotron tomographic data," *Journal of Synchrotron Radiation*, vol. 21, no. 5, pp. 1188–1193, Sep 2014.

[25] F. De Carlo, D. Gürsoy, F. Marone, M. Rivers, D. Y. Parkinson, F. Khan, N. Schwarz, D. J. Vine, S. Vogt, S.-C. Gleber *et al.*, "Scientific data exchange: a schema for hdf5-based storage of raw and analyzed data," *Journal of synchrotron radiation*, vol. 21, no. 6, pp. 1224–1230, 2014.

[26] The spot suite. [Online]. Available: http://spot.nersc.gov/index.php

[27] NERSC, "Cori - a system to support data-intensive computing," *CUG 2016: This Conference*, 2016.

[28] P. Calafiura, K. De, W. Guan, T. Maeno, P. Nilsson, D. Oleynik, S. Panitkin, V. Tsulaia, P. V. Gemmeren, and T. Wenaus, "Fine grained event processing on hpcs with the atlas yoda system," *Journal of Physics: Conference Series*, vol. 664, no. 9, p. 092025, 2015.

[29] A. Buckley, T. Eifert, M. Elsing, D. Gillberg, K. Koeneke, A. Krasznahorkay, E. Moyse, M. Nowak, S. Snyder, and P. van Gemmeren, "Implementation of the ATLAS Run 2 event data model," *J. Phys. Conf. Ser.*, vol. 664, no. 7, p. 072045, 2015.

[30] T. Maier, D. Benjamin, W. Bhimji, J. Elmsheuser, P. van Gemmeren, D. Malon, and N. Krumnack, "Atlas i/o performance optimization in as-deployed environments," *Journal of Physics: Conference Series*, vol. 664, no. 4, p. 042033, 2015.

Appendix A
List of NERSC Burst Buffer early user projects

NERSC-supported: New Efforts

- Nyx/BoxLib cosmology simulations
- Phoenix: 3D atmosphere simulator for supernovae
- Chombo-Crunch + VisIt for carbon sequestration
- Sigma/UniFam/Sipros bioinformatics codes
- XGC1 for plasma simulation
- PSANA for LCLS

NERSC-supported: Existing Engagements

- ALICE data analysis
- Tractor: Cosmological data analysis (DESI)
- VPIC-IO performance
- YODA: Geant4 simulations for the ATLAS detector
- Advanced Light Source SPOT Suite
- TomoPy for ALS and APS image reconstruction
- kitware: VPIC/Catalyst/ParaView

Early Access

- Image processing in cryo-microscopy/structural biology

- htslib for bioinformatics
- Falcon genome assembler
- Ray/HipMer genome assembly
- HipMer
- CESM Earth System model
- ACME/UV-CDAT for climate
- GVR with AMR, neutron transport, molecular dynamics
- XRootD for Open Science Grid
- OpenSpeedShop/component-based tool framework
- DL-POLY for material science
- CP2K for geoscience/physica chemistry
- ATLAS simulation of ITK with Geant4
- ATLAS data analysis
- Spark
- In-situ Analysis and I/O using Warp and VisIt