# Performance Characterization of Scientific Workflows for the Optimal Use of Burst Buffers

Christopher S. Daley, Devarshi Ghoshal, Glenn K. Lockwood, Sudip Dosanjh,
Lavanya Ramakrishnan, Nicholas J. Wright.
Lawrence Berkeley National Laboratory
1 Cyclotron Rd
Berkeley, CA
[csdaley,dghoshal,glock,sudip,lramakrishnan,njwright]@lbl.gov

## ABSTRACT

Scientific discoveries are increasingly dependent upon the analysis of large volumes of data from observations and simulations of complex phenomena. Scientists compose the complex analyses as workflows and execute them on large-scale HPC systems. The workflow structures are in contrast with monolithic single simulations that have often been the primary use case on HPC systems. Simultaneously, new storage paradigms such as Burst Buffers are also becoming available on HPC platforms. In order to maximize the performance of data analyses workflows today it is critical to determine the characteristics of the workflows. Obtaining a deeper understanding of the workflows helps us identify opportunities to leverage the capabilities of the Burst Buffer. In this paper, we analyze the performance characteristics of the Burst Buffer and two representative scientific workflows. We measure the performance of these workflows using the Burst Buffer, allowing us to make recommendations for future optimal usage of workflows using Burst Buffer.

## Keywords

Burst Buffer; DataWarp; Workflow; HPC

## 1. INTRODUCTION

The science drivers for high-performance computing (HPC) are broadening with the proliferation of high-resolution observational instruments and emergence of completely new data-intensive scientific domains. Scientific workflows that chain the processing and data are becoming critical to manage these on HPC systems. Thus, while providers of supercomputing resources must continue to support the extreme bandwidth requirements of traditional supercomputing applications, centers must now also deploy resources that are capable of supporting the requirements of these emerging data-intensive workflows. In sharp contrast to the highly coherent, sequential, large-transaction reads and writes that are characteristic of traditional HPC checkpoint-restart workloads [11], data-intensive workflows have been shown to often utilize non-sequential, metadata-intensive, and small-transaction reads and writes [13, 23]. Parallel file systems in today's supercomputers have been optimized for more traditional HPC workloads [12]. The rapid growth in I/O demands coming from data-intensive workflows are demanding new performance and optimization requirements of future HPC I/O subsystems [13]. It is therefore essential to develop methods to quantitatively characterize the I/O needs of data-intensive workflows to ensure that correct resources can be deployed with the correct balance of performance characteristics.

The emergence of data-intensive workflows has coincided with the emergence of flash devices being integrated into the HPC I/O subsystem as a "Burst Buffer", a performance-optimized storage tier that resides between compute nodes and the high-capacity parallel file system (PFS). The Burst Buffer was originally conceived for massive bandwidth requirements of checkpoint-restart workloads for extreme-scale simulation [19]. The tier *buffers* bursts of I/O traffic to enable the PFS to service a lower bandwidth load spread over a longer time period. However, the flash-based storage media underlying Burst Buffers are also substantially faster than spinning disk for the non-sequential and small-transaction I/O workloads of data-intensive workflows. This motivates using the media for use cases beyond buffering of I/O requests, such as providing a temporary scratch space, coupling workflow stages, and in-transit processing [4].

Today's commercially available Burst Buffer solutions [17] expose their flash through the POSIX API which enables workflows to easily leverage the technology's capabilities. We need to understand and optimize the use of Burst Buffers to serve the needs of data-intensive workflows. Thus, it is essential to understand workflows' specific I/O requirements in the context of both flash-based storage media and the I/O stack through which applications utilize the Burst Buffer.

In this paper, we characterize two of the production data analytics workflows used at the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory, and we present an analysis of their performance on the production Burst Buffer resource deployed as a part of NERSC's Cori system. The paper is organized as follows. Section 2 presents the background for the paper - related work and the details of the NERSC Burst Buffer Architecture. Section 3 details our approach to scalable I/O characterization for both workflows and Section 4 presents a detailed analysis of the I/O requirements of these workflows. We discuss efficient use of Burst Buffers in Section 5 and provide conclusions in Section 6.

## 2. BACKGROUND

In this section we describe related work and the NERSC Burst Buffer architecture.

### 2.1 Related Work

**Scientific Workflows.** Data-intensive scientific workflows have been shown to process large amounts of data with varied I/O characteristics [16, 21, 9, 7]. Deelman et al. [14] highlights several challenges in data management for data-intensive scientific workflows. Several strategies have been proposed to optimize data management for scientific workflows in HPC environments [28, 20, 8]. However, Burst Buffers add another layer in the storage hierarchy, adding to the data management challenges for scientific workflows. Hence, it is important to characterize scientific workflows to optimally use Burst Buffers based on their I/O characteristics. In this paper, we evaluate and characterize multiple workflows with different I/O profiles to understand the optimal use of Burst Buffers.

**Burst Buffers.** Several uses of Burst Buffers have been shown in order to mitigate the I/O bottlenecks of data-intensive workloads [19, 6, 22, 25]. Most studies surrounding the design and use of Burst Buffers have so far focused on the I/O characteristics of individual applications [26] or small components within workflows [23]. However, research into optimizing scientific workflows with diverse I/O and storage requirements for Burst Buffers is still in its infancy, and a limited body of work presently exists [13, 5]. Beyond single applications and workflows, researchers are investigating I/O-aware scheduling on systems with a Burst Buffer. Herbein et al. [18] demonstrate that system utilization can be improved by using application drain bandwidth between the Burst Buffer and PFS as a scheduling constraint. Thapaliya et al. [24] show how different Burst Buffer allocation policies and the order of servicing I/O requests affects total application throughput on a system with a shared Burst Buffer.

**DataWarp.** DataWarp is Cray's implementation of a Burst Buffer, and few guidelines exist for how to use it optimally for scientific workflows. Bhimji et. al show performance results for a collection of applications selected as part of NERSC's Early User Program [10]. The results focus on application I/O bandwidth on DataWarp and the PFS. The NERSC website provides a list of known issues and overall guidelines for achieving high performance, but does not show when, why and how to use DataWarp for specific workflow use cases [1]. Our work has analyzed two data analytics workflows and identified I/O signatures along with the specific workflow requirements to advise how to use DataWarp.

## 2.2 The NERSC Burst Buffer Architecture

NERSC's Cori system features a Burst Buffer based on Cray DataWarp [17]. This architecture is built upon discrete *Burst Buffer nodes* (BB nodes), each containing two Intel P3608 SSDs that deliver 6.4 TiB of usable capacity and 5.7 GiB/s of bandwidth. Currently, Cori has a total of 144 BB nodes, over 900 TiB of usable capacity, and over 800 GiB/sec of peak performance.

Cray's DataWarp middleware aggregates the SSDs on each of the BB nodes and provides user jobs with dynamically provisioned private parallel file systems. Users can request a certain capacity of Burst Buffer in 200 GiB increments (which we call *fragments*) when submitting jobs. Each fragment is allocated on a different BB node to allow the aggregate performance of the BB allocation to scale with the requested capacity. DataWarp also designates one of the BB nodes as the metadata server for the allocation. This allocation is mounted on the job nodes when the job is launched, and it is typically torn down upon job completion. However, users may also request a *persistent mode* allocation, which allows a BB allocation to persist across multiple jobs.

DataWarp also offers *private mode* reservations where each compute node gets its own metadata server within the Burst Buffer allocation and, by extension, its own private namespace. This enables higher aggregate metadata performance since each compute node's metadata is serviced by a unique BB node.

## 3. METHODOLOGY

In this section, we detail our performance analysis methodology and workloads used for our analyses.

## 3.1 Workflows

The two workflows studied in the paper were selected because they stress the I/O subsystem in very different ways: CAMP is limited by metadata performance and SWarp is limited by data transfer performance. When discussing the workflows, we use the term "workflow pipeline" to refer to a single unit of the larger workflow.

### 3.1.1 CAMP

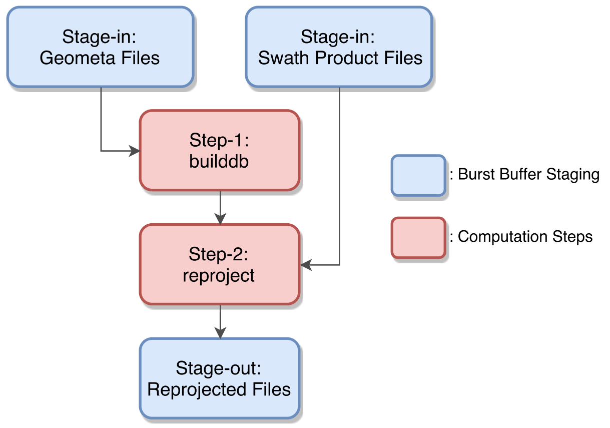

Figure 1: CAMP workflow: i) staging operations move the data from the parallel file system to the Burst Buffer and vice-versa, ii) builddb and reproject transform the swath products to a sinusoidal tiling system.

The CAMP (Community Access MODIS Pipeline) workflow processes Earth's land and atmospheric data obtained from MODIS satellite data [3, 27, 16]. It transforms the MODIS data from a swath space and time coordinate system (latitude and longitude) into a sinusoidal tiling system (tiles using sinusoidal projection). The MODIS data for CAMP consists of small geometa files in plain text format and swath products as Hierarchical Data Format (HDF) files. Each geometa file is only a few KBs and is used by all the swath products from a particular satellite. Each swath product has several files per day, each of which is approx. 1.1 MB in size and contains the product data in swath space and time coordinate system.

The CAMP workflow consists of two processing steps – a) **builddb**, that assembles and maps swaths to their corresponding sinusoidal tiles and b) **reproject**, that converts the MODIS products from a swath coordinate system to a sinusoidal tiling system. Figure 1 shows the high-level representation of the CAMP workflow that includes the data staging operations to and from the Burst Buffer. The workflow pipeline in this paper transforms one MODIS product's swath coordinates for one day into one specific tile. CAMP is written in Python and generates an intermediate SQLite database to provide the mapping for the reproject stage. We use Conda, which uses the Anaconda Python distribution, to install CAMP on DataWarp.

### *3.1.2 SWarp*

The SWarp workflow combines overlapping raw images of the night sky into high quality reference images. It is used in the Dark Energy Camera Legacy Survey (DECaLS) to produce high quality images of 14,000 deg$^2$ of northern hemisphere sky. In this survey, each SWarp workflow pipeline produces an image for a 0.25 deg$^2$ "brick" of sky. The average input to each workflow pipeline is 16 × 32 MiB input images and 16 × 16 MiB input weight maps.

The SWarp workflow pipeline consists of a data resampling stage and a data combination stage. The data resampling stage interpolates the raw images and creates resampled images which can be trivially stacked. The data combination stage reads back the resampled images and then performs a reduction over the pixels to produce a single stacked image. The raw, resampled and stacked images are all in Flexible Image Transport System (FITS) file format. The DAG when using a Burst Buffer is similar to CAMP: input images and weight map files are staged-in prior to the data resampling stage and the combined image is staged-out after the data combination stage. SWarp is written in C and multithreaded with POSIX threads.

## 3.2 Workload Configuration

The workflow pipelines are run in their production configuration on Cori and all I/O is directed to DataWarp mount points. The DataWarp reservation is configured to use a shared namespace and one fragment of capacity. A job reservation is used for SWarp and a persistent reservation is used for CAMP (in order to retain the CAMP Python software environment between jobs). The Integrated Performance Monitoring (IPM) profiling tool [2] is used to collect run time, memory usage and time in different I/O calls for each workflow stage. The workflow pipelines are then replicated on 1 to 64 compute nodes (with 1 workflow pipeline per compute node) and I/O is directed to a fixed storage reservation of 1 DataWarp fragment. This allows us to study how run time is affected by the saturation of the storage resource.

## 4. RESULTS

The high-level characteristics of the stages in a single workflow pipeline are shown in Table 1. The workflow stages are found to spend 10 - 30 % of time in I/O. This is the best achievable I/O time and can only get worse as more workflow pipelines contend for the same storage resource.

Figures 2 and 3 show how I/O time changes with concurrency for the most time-consuming stage of each workflow. I/O time is divided into time spent in metadata operations and data operations. The experiments are repeated three times at each node count and the plots show the mean time per workflow pipeline stage. The error bars simply show the range of mean times over the three experiments.

Figure 2 shows the scaling of SWarp-resample. The results show that wall clock time remains relatively constant until about 16 workflow pipelines and that I/O time is dominated by data rather than metadata operations. Figure 3 shows the scaling of CAMP-builddb is limited by metadata performance. One source of these metadata operations is from the startup of Python applications, which is known to be a scalability issue in Python HPC applications [15]. It happens because Python searches for files providing a package in every directory in the Python path. In spite of this, the dominant source of metadata load in CAMP-builddb are the transactions to the SQLite database.

| | **SWarp rsmpl** | **SWarp coadd** | **CAMP db** | **CAMP reprj** |
|---|---|---|---|---|
| Compute threads | 16 | 16 | 1 | 1 |
| I/O threads | 1 | 1 | 1 | 1 |
| Wall time (s) | 10.7 | 4.7 | 15.3 | 9.2 |
| I/O time (s) | 2.2 | 1.2 | 2.1 | 1.5 |
| I/O time (%) | 20.3 | 26.0 | 13.5 | 16.6 |
| Peak mem. (MiB) | 108.8 | 1064.7 | 96.1 | 93.0 |
| Total file size (MiB) | 1686.5 | 1016.8 | 74.1 | 77.5 |

Table 1: Time and memory measurements achieved with 1 compute node and 1 DataWarp fragment

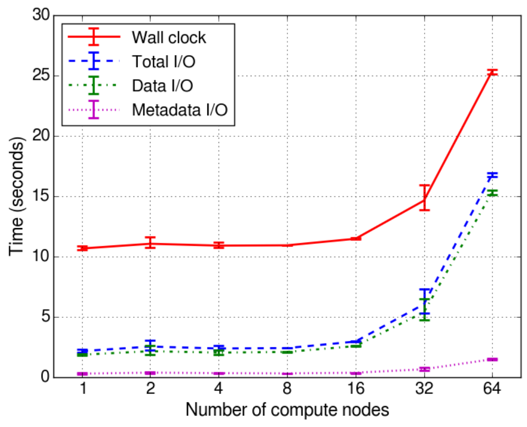

Figure 2: Scaling of SWarp-resample with number of workflow pipelines

## 5. DISCUSSION

In this section, a) we discuss the key characteristics of the workflows analyzed and use the information to highlight the effective use of Burst Buffers and, b) we apply this knowledge to explain how to achieve the optimum performance with the DataWarp implementation of a Burst Buffer.

## 5.1 Efficient use of Burst Buffers

The key findings from our experimental analyses are:

1. **A single workflow pipeline does not provide the I/O parallelism needed to make efficient use of Burst Buffers.** The data analytics workflows studied in this paper consist of single-process applications which perform I/O with a single thread of execution. This is poorly matched with the need to have

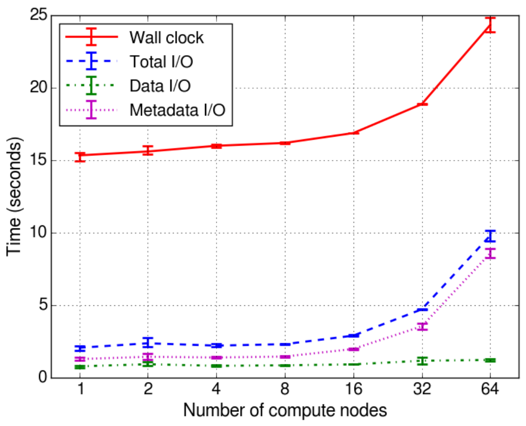

Figure 3: Scaling of CAMP-builddb with number of workflow pipelines

multiple I/O streams to obtain the peak performance from Burst Buffer Flash storage. Unfortunately, single I/O stream workflow pipelines are a common feature of high throughput data analytics workflows. We show that better utilization of Burst Buffer resources is possible by executing multiple concurrent workflow pipelines against the same unit of Burst Buffer storage. Our results indicate that a single unit of DataWarp storage on Cori can sustain the I/O requests from approximately 16 concurrent workflow pipelines before there is any slow down.

2. **A scaled out workflow pipeline is often limited by metadata performance.** Our analysis has found significant metadata costs originating from database transactions, Python initialization and opening many small files. The aggregated metadata operations from multiple workflow pipelines can easily saturate a single metadata server, as shown in the CAMP-builddb workflow stage.
3. **It is valuable to explicitly control the data in the Burst Buffer tier.** The workflows read input data sets and produce a number of intermediate files which can be discarded once there are final results, e.g. the resampled images in SWarp and the SQLite database in CAMP. Therefore, we do not expect automatic file movement between the Burst Buffer and the PFS to benefit these two data analytics workflows. This is because the one-time cost of staging the input data at access time may not be hidden by significant data reuse. Automatic file movement would also transfer the intermediate files to the PFS unnecessarily.
4. **It is valuable to leave data in the Burst Buffer tier for longer than a single batch job.** We have found that input files and software environments are reused across workflow pipelines.
    - The input data for data analytics workflows are generally Write Once Read Many times (WORM). In the SWarp workflow a single input image often contributes to multiple regions of the sky. Therefore it is wasteful to re-stage the same input file multiple times for each workflow pipeline.
    - The software environment is reused in every single workflow pipeline. In the CAMP workflow the Python environment is responsible for some of the I/O. The role of “support I/O” (e.g. Python packages) is rarely mentioned in the context of Burst Buffers. It is useful to stage the software environment once to avoid the overhead and wear of repeatedly staging the software environment.

    Long-term data residency is not a good fit for today’s Burst Buffers because they do not provide data redundancy. This imposes a data management burden upon the developer.

## 5.2 Efficient use of DataWarp

DataWarp storage reservations on Cori consist of multiple storage fragments of size 200 GiB. The scaling studies show that both SWarp and CAMP are limited by DataWarp performance rather than capacity. SWarp and CAMP have an aggregate capacity requirement of up to 2.6 GiB and 150 MiB per workflow pipeline, respectively (Table 1). However, the performance saturates before fully utilizing the 200 GiB of capacity at approximately 16 workflow pipelines per DataWarp fragment. This means that excess capacity must be reserved to sustain performance in a scaled out workflow. Metadata bottlenecks, such as seen in CAMP-builddb, can be addressed by combining the reservation of excess capacity with the private mode feature of DataWarp.

## 6. CONCLUSION

In this paper we analyzed the performance of two scientific workflows running on the Cori supercomputer with the DataWarp Burst Buffer. We show that a single workflow pipeline does not have the parallelism to utilize the capabilities of the Flash storage hardware. We also show that the workflows have different I/O performance characteristics: SWarp is bound by data transfer performance and CAMP (specifically CAMP-builddb) is bound by metadata performance as the workflows are scaled out. The results are used to give general advice about using Burst Buffers more efficiently and to provide specific advice for DataWarp.

## Acknowledgments

This work was supported by Laboratory Directed Research and Development (LDRD) funding from Berkeley Lab, provided by the Director, Office of Science and Office of Science, Office of Advanced Scientific Computing Research (ASCR) of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. 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. The authors would also like to thank Rollin Thomas for help with installing the CAMP Python software environment on DataWarp.

## 7. REFERENCES

[1] Burst Buffer. NERSC website: http://www.nersc.gov/users/computational-systems/cori/burst-buffer/; accessed 31 August 2016.
[2] IPM. https://github.com/nerscadmin/IPM; accessed 13 July 2016.

[3] NASA MODIS Website. http://modis.gsfc.nasa.gov/.

[4] Trinity / NERSC-8 Use Case Scenarios. Technical Report SAND 2013-2941 P, Los Alamos National Laboratory, Sandia National Laboratories, NERSC, Apr. 2013. https://www.nersc.gov/assets/Trinity--NERSC-8-RFP/Documents/trinity-NERSC8-use-case-v1.2a.pdf; accessed 4 October 2016.

[5] APEX Workflows. Technical report, Los Alamos National Laboratory, NERSC, and Sandia National Laboratories, Los Alamos, NM, 2016.

[6] J. Bent, G. Grider, B. Kettering, A. Manzanares, M. McClelland, A. Torres, and A. Torrez. Storage challenges at los alamos national lab. In *IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–5, April 2012.

[7] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, 2004.

[8] S. Bharathi and A. Chervenak. Scheduling data-intensive workflows on storage constrained resources. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, pages 3:1–3:10, New York, NY, USA, 2009. ACM.

[9] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, and K. Vahi. Characterization of scientific workflows. In *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, Nov 2008.

[10] W. Bhimji et al. Accelerating Science with the NERSC Burst Buffer Early User Program. In *Cray User Group CUG*, May 2016.

[11] S. Byna, A. Uselton, D. Knaak, and Y. H. He. Lessons Learned from a Hero I/O Run on Hopper. In *2013 Cray User Group Meeting*, Napa, CA, 2013.

[12] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig. Small-file access in parallel file systems. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–11. IEEE, may 2009.

[13] C. S. Daley, L. Ramakrishnan, S. Dosanjh, and N. J. Wright. Analyses of Scientific Workflows for Effective Use of Future Architectures. In *Proceedings of the 6th International Workshop on Big Data Analytics: Challenges, and Opportunities (BDAC-15)*, Austin, TX, 2015.

[14] E. Deelman and A. Chervenak. Data management challenges of data-intensive scientific workflows. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 687–692, May 2008.

[15] J. Enkovaara, N. A. Romero, S. Shende, and J. J. Mortensen. Gpaw - massively parallel electronic structure calculations with python-based software. *Procedia Computer Science*, 4:17 – 25, 2011.

[16] V. Hendrix, L. Ramakrishnan, Y. Ryu, C. van Ingen, K. R. Jackson, and D. Agarwal. CAMP: Community Access MODIS Pipeline. *Future Generation Computer Systems*, 36:418 – 429, 2014.

[17] D. Henseler, B. Landsteiner, D. Petesch, C. Wright, and N. Wright. Architecture and Design of Cray DataWarp. In *Cray User Group CUG*, May 2016.

[18] S. Herbein, D. H. Ahn, D. Lipari, T. R. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer. Scalable I/O-Aware Job Scheduling for Burst Buffer Enabled HPC Clusters. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '16, pages 69–80, New York, NY, USA, 2016. ACM.

[19] 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 *IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–11, Apr. 2012.

[20] H. M. Monti, A. R. Butt, and S. S. Vazhkudai. On timely staging of hpc job input data. *IEEE Transactions on Parallel and Distributed Systems*, 24(9):1841–1851, 2013.

[21] L. Ramakrishnan and B. Plale. A multi-dimensional classification model for scientific workflow characteristics. In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science*, Wands '10, pages 4:1–4:12, New York, NY, USA, 2010. ACM.

[22] K. Sato, K. Mohror, A. Moody, T. Gamblin, B. R. d. Supinski, N. Maruyama, and S. Matsuoka. A user-level infiniband-based file system and checkpoint strategy for burst buffers. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 21–30, May 2014.

[23] K. A. Standish, T. M. Carland, G. K. Lockwood, W. Pfeiffer, M. Tatineni, C. C. Huang, S. Lamberth, Y. Cherkas, C. Brodmerkel, E. Jaeger, L. Smith, G. Rajagopal, M. E. Curran, and N. J. Schork. Group-based variant calling leveraging next-generation supercomputing for large-scale whole-genome sequencing studies. *BMC Bioinformatics*, 16(1):304, dec 2015.

[24] S. Thapaliya, P. Bangalore, J. Lofstead, K. Mohror, and A. Moody. Managing I/O Interference in a Shared Burst Buffer System. In *2016 45th International Conference on Parallel Processing (ICPP)*, pages 416–425, Aug. 2016.

[25] B. Van Essen, R. Pearce, S. Ames, and M. Gokhale. On the Role of NVRAM in Data-intensive Architectures: An Evaluation. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pages 703–714. IEEE, may 2012.

[26] T. Wang, S. Oral, M. Pritchard, K. Vasko, and W. Yu. Development of a burst buffer system for data-intensive applications. *CoRR*, abs/1505.01765, 2015.

[27] R. E. Wolfe, D. P. Roy, and E. Vermote. Modis land data storage, gridding, and compositing methodology: Level 2 grid. *IEEE Transactions on Geoscience and Remote Sensing*, 36(4):1324–1338, Jul 1998.

[28] Z. Zhang, C. Wang, S. S. Vazhkudai, X. Ma, G. G. Pike, J. W. Cobb, and F. Mueller. Optimizing center performance through coordinated data staging, scheduling and recovery. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC '07, pages 55:1–55:11, New York, NY, USA, 2007. ACM.