Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

I/O performance analysis of machine learning workloads on leadership scale supercomputer[☆]

Ahmad Maroof Karimi[a,1,*], Arnab K. Paul[b,1], Feiyi Wang[a]

[a] Oak Ridge National Laboratory, USA
[b] BITS Pilani, K K Birla Goa Campus, India

ARTICLE INFO

Article history:
Available online 5 September 2022

Keywords:
Burst buffer
Darshan
I/O characterization
Machine learning
Parallel file system
Processing units

ABSTRACT

The popularity of machine learning technologies and frameworks has led to an increasingly large number of machine learning workloads running on high-performance computing (HPC) clusters. The ML workflows are readily being adopted in diverse computational fields such as Biology, Physics, Materials, and Computer Science. The I/O behavior of the emerging ML workloads distinctly differs from the traditional HPC workloads, such as simulation or checkpoint/restart-based HPC I/O behavior. Additionally, the ML workloads have also pushed for the utilization of GPUs or a combination of CPUs and GPUs in addition to using only CPUs for computational tasks. The diverse and complex I/O behavior of ML workloads requires extensive study and is critical for the efficient performance of various layers of the I/O stack and the overall performance of HPC workloads. This work aims to fill the gap in understanding the I/O behavior of emerging ML workloads by providing an in-depth analysis of ML jobs running on large-scale leadership HPC systems. In particular, we have analyzed the behavior of jobs based on the scale of the jobs, the science domains, and the processing units used by the ML jobs. The analysis was performed on 23,000 ML jobs collected from one year of Darshan logs running on Summit, which is one of the fastest supercomputers. We also collect the CPU and GPU usage of 15,165 ML jobs by merging the Darshan dataset with the power usage of the processing units on Summit. Therefore, this paper is able to provide a systematic I/O characterization of ML workloads on a leadership scale HPC machine to understand how the I/O behavior differs for workloads across various science domains, the scale of workloads, and processing units and analyze the usage of parallel file system and burst buffer by ML I/O workloads. We have made several observations regarding I/O performances and access patterns through various analytical studies and discuss the important lessons learnt from the perspective of a ML user and a storage architect for emerging ML workloads running on large-scale supercomputers.

☆ This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

* Corresponding author.
E-mail addresses: karimiahmad@ornl.gov (A.M. Karimi), arnabp@goa.bits-pilani.ac.in (A.K. Paul), fwang2@ornl.gov (F. Wang).
URL: https://arnabkrpaul.github.io/ (A.K. Paul).

1 Both authors made equal contributions.

https://doi.org/10.1016/j.peva.2022.102318
0166-5316/© 2022 Published by Elsevier B.V.

1. Introduction

As machine learning (ML) and deep learning (DL) technologies have gained more attention, particularly for solving large and complex modeling problems, the I/O usage patterns of high performance computing (HPC) systems are diverging from the patterns exhibited by traditional workloads. I/O behavior observed from conventional workloads like simulation, visualization, or checkpoint/restart-based is different from the complex I/O of ML and DL jobs which involve multiple sub-tasks applying a large number of read I/O calls for training on a numerous number of batches and across a large number of epochs. The datasets involved in these tasks are also heterogeneous, such as images, text, and time-series vectors. Various science domains such as biology, physics, and chemistry, have contributed to even more diverse I/O patterns in large-scale data analysis and ML/DL workloads [1]. This diversity of I/O needs by HPC workloads warrants the need to characterize the requirements of I/O in modern and future HPC centers.

There has been a number of works published [2–5] dedicated to the characterization of traditional workloads, such as simulation involving write-heavy checkpoint/restart. Recent studies [6–9] have also focused on analyzing the I/O requirements of popular ML methods, like deep learning. However, there is a lack of research on a system-wide cross-sectional analysis of HPC systems that can provide a holistic understanding of the typical I/O characteristics of ML workloads. The ML workloads generally follow several standard steps, such as reading data from a file system, processing and organizing the data, and passing it to a ML training algorithm. It has been typically observed that ML workloads have small read and write access patterns. The read calls are usually made in small batches during training for a large number of epochs on entire datasets. However, based on different domain sciences, applications, the scale of ML jobs, and CPU/GPU utilization, the I/O calls can become very complex, especially on large-scale HPC systems having thousands of nodes.

There are various I/O monitoring tools to record the performance of HPC storage systems. One such tool is Recorder [10], which can capture I/O function calls at multiple layers of the parallel file system I/O stack without modifying an application's source code. An I/O toolkit called RIOT [11] provides a robust framework to analyze parallel file system I/O behavior. Darshan [12–14] is one of the most widely used I/O profiling tools for system-wide analysis of HPC workloads performance. It is designed to capture an accurate picture of application I/O behavior, including properties such as access patterns, with minimum overhead, and provides I/O metrics at the individual files level. It is lightweight and is efficiently deployed on an HPC system. It provides valuable insights into potential performance-tuning efforts and enables evaluating I/O trends for the entire HPC cluster. Darshan logs have been used in many studies to characterize the I/O needs of HPC workloads. However, Darshan does not annotate the logs into ML and non-ML workloads. Therefore, it is a non-trivial task to classify the two kinds of HPC workloads (ML and non-ML), which would help characterize the different HPC I/O behavior.

Most ML jobs are perceived to be read-intensive with a lot of small reads while a few ML jobs also perform small writes. This kind of I/O behavior suggests that an in-system storage, like a burst buffer [15] will provide better I/O performance for ML workloads than a parallel file system, like IBM Spectrum Scale (GPFS) [16], where the performance is limited by a large number of metadata requests. On the other hand, a burst buffer is a fast and intermediate storage layer between the non-persistent memory of the compute nodes and persistent storage — the parallel file system. The burst buffer layer is configured to take a burst of read or write I/O at a very high rate. However, there has been no prior study on the usage of burst buffer and parallel file system by large-scale ML I/O workloads.

ML scientists and engineers generally focus more on the computation portions of the jobs, thereby overlooking the performance of the I/O component. Therefore, it is essential to study the effect of I/O when ML jobs use different kinds of processing units — CPUs and GPUs. Do ML jobs which use only GPUs fare better in terms of I/O performance than their counterparts that use only CPUs? How do the different science domains use the processing units and the underlying storage systems?

In this paper, we aim to fill the gap in literature by characterizing the I/O behavior of ML workloads based on different science domains, the scale of ML I/O job runs, and temporal trends over one year. We also analyze the usage of the parallel file system and burst buffer by ML I/O workloads. To this end, we study the Darshan logs of 23,389 HPC ML I/O jobs spanning over one year (January 2020–December 2020) on Summit [17] - the second-fastest supercomputer in the world according to the latest top-500 list [18]. IBM Spectrum Scale (previously known as GPFS) [16] forms the parallel file system in Summit. In the remainder of the paper, we use the term 'GPFS' for the parallel file system and 'BB' for burst buffer.

This article extends our previous work [19] by merging the Darshan logs of the ML jobs with the power data of the CPUs and GPUs for the jobs running on Summit. The Darshan dataset itself does not provide enough information about the usage of CPU and GPU layers. Merging Darshan data with power consumption logs allows us to analyze the effects of I/O on the compute nodes' processing units (CPU and GPU). Upon combining the two datasets, we are able to get the CPU and GPU usage of 15,165 ML jobs.

Specifically, we make the following contributions with regards to the I/O behavior of ML workloads on a leadership scale HPC systems.

- Develop a technique to annotate ML workloads from Darshan logs.
- Analyze the I/O behavior of large-scale ML workloads classified by different science domains and the scale of job runs.

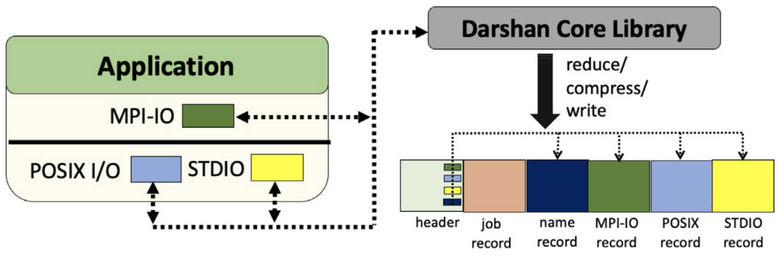


Fig. 1. High-level overview of Darshan's architecture.

- Study the usage of the parallel file system and burst buffer by ML I/O jobs and understand the scope of improvement in I/O performance.
- Estimate the I/O effects caused on different layers of the storage subsystem by ML jobs submitted by various science domains classified on the basis of CPU and GPU usage.
- Understand the I/O behavior and potential bottleneck caused during the execution of the workloads running on GPUs.

From our study, we observe that ML workloads generate a large number of small file reads and writes, which is ideal for burst buffer. However only few science domains use burst buffer, and out of those, only some use the burst buffer efficiently. The temporal trend of ML workloads on Summit also indicates an exponential increase in the I/O activity from ML workloads by different science domains which is indicative of the future which will be dominated by ML. We also find that GPU only jobs cause a more significant I/O bottleneck than jobs running on only CPUs. Based on our observations, we also list down key discussion points for the benefit of the ML scientists and the storage architects.

2. Background

2.1. Summit supercomputer

The Summit supercomputer is based upon IBM AC922 system and deployed at the Oak Ridge Leadership Computing Facility (OLCF). It consists of 4,626 compute nodes. Each node is equipped with 2 IBM POWER9 (P9) processors and 6 NVIDIA Tesla V100 (Volta) GPUs. Also, each node has 512 GB of DDR4 CPU memory, and each GPU has 16 GB of HBM2 memory. An NVLink 2.0 bus connects each P9 CPU to 3 V100 GPUs. Mellanox InfiniBand Enhanced Data Rate (EDR) network with a fat-tree topology connects the nodes. Summit's peak power consumption is 13 MW and is currently ranked No. 11 on the Green500 list with 14.719 GFlops/watts, supported by a 20MW facility that provides the necessary power and cooling to the compute cluster floor. A 1.6 TB NVMe device is present on each compute node to be used as node-local storage — burst buffer (BB). Summit is connected to Alpine, a 250 PB IBM Spectrum Scale (GPFS) file system. Summit can access Alpine at 2.5 TB/s in aggregate under a large, and sequential write I/O access pattern. Alpine is a center-wide file system and directly accessed by all other OLCF resources.

2.2. Darshan - HPC I/O characterization tool

Fig. 1 provides an overview of the Darshan architecture. As an application executes, the Darshan instrumentation module for MPI, POSIX, and STDIO generates data records characterizing the application's I/O workload within different components of the I/O stack. The instrumentation modules for the various components are registered with the Darshan core library. During application shutdown, each module organizes its records, compresses it and writes those collectively to the log file. MPI-IO is recorded for every *MPI_File_read*() and *MPI_File_write*() calls. POSIX module records each *read*() and *write*() call. Many applications rely on text-based I/O in leadership class computing facilities. The STDIO module characterizes the *stdio.h* family of functions, such as *fopen*(), *fprintf*(), and *fscanf*(). We share a month's worth of Darshan logs collected from Summit supercomputer.[2]

3. Methodology

3.1. Annotating ML jobs

Darshan provides *darshan − util*, a collection of tools for parsing and summarizing log files produced by the Darshan instrumentation module. We use one of its tools *darshan − parser*, to parse the raw Darshan logs of one year (January to December 2020) and get the file-wise I/O statistics in each module (MPI-IO, POSIX, or STDIO) accessed by the application. The metadata of each parsed Darshan log consists of *jobid*, *userid*, *start_time*, *end_time*, *executable*, *no_of_processes*, and *runtime*. These parsed logs are used to annotate ML jobs and study their I/O behavior.

[2] Dataset: https://doi.ccs.ornl.gov/ui/doi/384, DOI: 10.13139/OLCF/1865904.

Table 1
List of 42 ML keywords used to annotate ML jobs from Darshan logs.

List of ML Keywords					
keras	*training*	*solr_keras*	*candle_keras*	*cosmoFlow_cnn1*	*Large_Batch_Training*
train	*tf_cnn*	*stemdl*	*randomForest*	*STEMDL-Benchmark*	*network_FCDenseNet_custom*
imagenet	*.tfrecord*	*ppi-cnn-gpu*	*ppi-3d-cnn*	*tensorboardX*	*pytorch_synthetic_benchmark*
DeepBench	*epoch*	*regression*	*prediction*	*batch_size*	*federated-learning*
nt_train	*FC-DenseNet*	*iRF*	*tensorflow*	*genomicPredictionRF*	*genomicPredictioniRF*
sklearn	*horovod*	*cnn.pickle*	*convolutional.py*	*spDNN_data*	*sp1vfast3d_regression_ibm*
interpolate	*pytorch*	*tensorflow*	*DNN*	*train_tfrecord*	*tf_cnn_benchmarks*

To identify ML I/O workloads from the parsed Darshan logs, we create a list of ML keywords from the most common ML libraries used on leadership scale HPC systems. By browsing through the executable and file names present in the Darshan logs from the first two months, we find that most HPC ML workloads use libraries from either R or Python. Combining the knowledge of ML terminologies found in the Darshan logs of the first two months and searching for the top ML/DL libraries used in R and Python, a list containing 42 ML keywords is created, which is shown in Table 1. The list of keywords may not be comprehensive, but it is sufficient to provide enough samples representing the ML jobs on Summit.

Summit's scheduler logs are merged with the Darshan logs to get the science domain (Physics, Chemistry, Computer Science, or others) and the number of nodes on which the ML jobs ran. A total of 845,036 jobs ran on Summit in 2020. Out of this, Darshan logs were generated for 279,642 jobs. The executable and filenames present in Darshan logs for the entire year are checked against the list of ML keywords shown in Table 1, and the final dataset of Darshan logs for 23,389 ML jobs running on Summit in 2020 is obtained, which is used for the analysis in this paper.

The final number of ML jobs in the dataset is close to 9% of the entire number of jobs running on Summit for the year 2020. This is believed to be a sufficiently large number of jobs to provide logical observations as there have been prior studies on I/O analysis which have been done on one month of supercomputing workloads, where the number of jobs were less than 10,000 [20]. Moreover, the above described process of filtering ML jobs ensures that the dataset does not contain any false positives. However, there is no way to determine the true negatives, or the ML workloads that were missed in the above process. This is an existing challenge in the community. Even manually scavenging workloads coming for the ML domain suggests that not all workloads coming from even the ML domain can be classified as ML. Therefore, the only option is to resort to keywords in the filenames or in the executable. This ensures that our observations are not biased towards traditional simulation based HPC workloads.

3.2. Per-node processing unit power consumption and error management

We calculated the job-wise power measurement used in this work by aggregating the values recorded for the duration of job run-time at each node on which the job was executed. The power data is collected at each node at a frequency of 1 Hz based on the sampling from 500 μs instantaneous power measurement. The recorded power data has separate fields for CPU power, GPU power, and total power. We validated the aggregated values against the measurements from the main switchboards (MSBs) that distribute power to the compute cabinets. Mainly, we compared the summation at the per-node 10-second mean input power that belongs under each MSB. Overall, our method using the aggregation of the 10-second mean power of each node was about 11% from the actual physical MSB measurement [21]. After that, we calculated the mean values across all the nodes on which the job was running and for the duration of the job to get the job-wise average power values.

4. Analysis of ML I/O workloads

4.1. Classification based on science domains

The science domain is a high-level classification of the workloads based on the project codes from which the job is executed. We mapped every project with one science domain depending on the code, and several projects can belong to one science domain. Analyzing ML workloads based on the science domain is necessary because the nature of applications under different science domains exhibit significantly different I/O behavior.

4.1.1. Distribution of ML I/O jobs

ML I/O jobs are analyzed to identify the science domains that make the most use of ML techniques in their HPC applications. Fig. 2 shows the distribution of ML jobs in different science domains along with the number of users and unique applications which make use of ML techniques in their HPC jobs. As explained in Section 3, a total of 23,389 ML jobs were analyzed.

Observation: Biology constitutes the maximum proportion of ML jobs on Summit over a year. However, Computer Science has the maximum number of users that use ML approaches in their jobs.

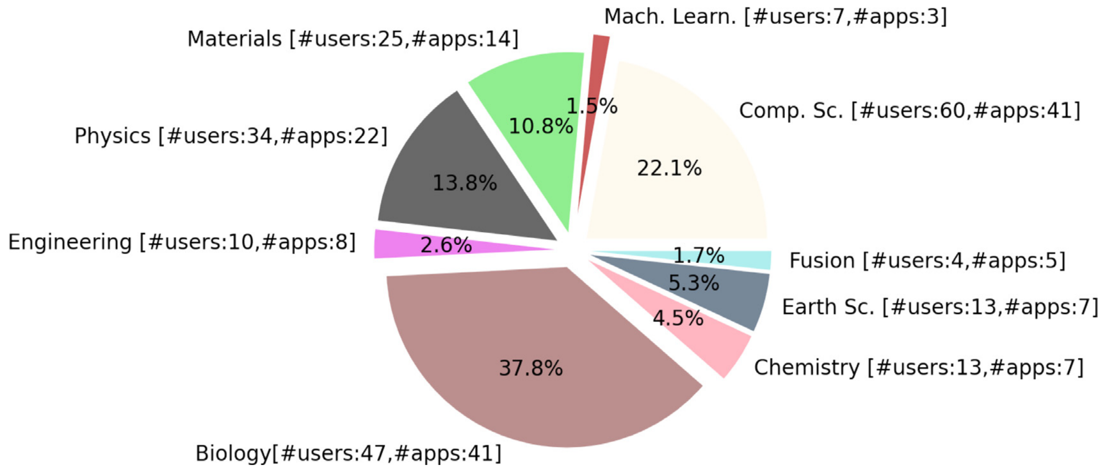


Fig. 2. Classification of 23,389 ML jobs by science domains. (**Note:** #users specifies the number of unique users submitting ML jobs, and #apps is the number of unique applications in each science domain).

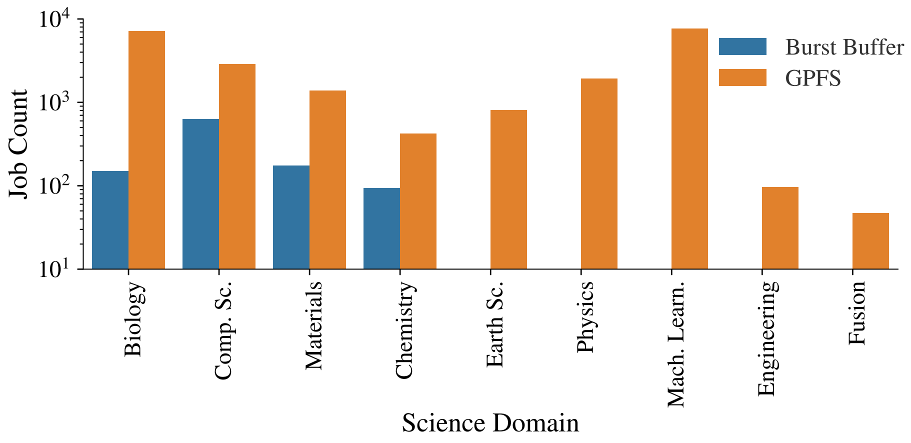


Fig. 3. The number of ML jobs using burst buffer and GPFS classified by different science domains on Summit.

4.1.2. Jobs using GPFS and BB

Fig. 3 shows the number of ML jobs in each science domain that either has at least one file access on BB (BB Jobs) or all file accesses exclusively on GPFS (GPFS jobs).

Observation: Computer Science has the most number of jobs that use BB compared to the other science domains. Only four science domains (Biology, Computer Science, Materials, and Chemistry) use BB for their ML workloads. Contrary to the common viewpoint, the plot shows the usage of BB is limited to users only from selected domain sciences and may reflect a limited understanding of the BB techniques to improve I/O performance by various science domains.

4.1.3. Types of ML I/O jobs

For every science domain that uses BB, we classify each job into one of the three categories: Read-Intensive (RI), Write-Intensive (WI), and Read–Write (RW).

$$result = \frac{ReadBytes - WriteBytes}{ReadBytes + WriteBytes} \tag{1}$$

Eq. (1) specifies the job type based on the following:

- $-1 \leq result \leq -0.5$: *Write-Intensive (WI)*
- $0.5 \leq result \leq 1$: *Read-Intensive (RI)*
- $-0.5 < result < 0.5$: *Read–Write (RW)*

Table 2 shows the percentage of ML jobs classified into the job type (RI, WI, RW) that either use exclusively GPFS or at least one of the file is in BB.

Table 2
Comparison of the percentage of read-intensive (RI) vs write-intensive (WI) vs read–write (RW) ML jobs using GPFS or Burst Buffer classified by the four science domains that use BB.

Job Size	GPFS			Burst Buffer		
	RI	WI	RW	RI	WI	RW
Comp. Sc.	82.21	9.41	8.38	31.47	67.41	1.12
Biology	43.29	24.59	32.12	97.28	1.36	1.36
Materials	21.15	18.82	60.03	100.0	0	0
Chemistry	76.78	9.72	13.50	0	100.0	0

Table 3
The mean number of read and write calls per job made in each group of file access sizes classified by science domains. The file access sizes are grouped into < 1MB, 1MB–10MB, 10MB–100MB, 100MB–1GB, and > 1GB bins. ∼99% of the read and write calls are less than 10MB calls.

Sc. Domains	Number of read calls				
	<1M	1M-10M	10M-100M	100M-1G	>1G
Biology	2.92e+7	922.12	678.61	70.07	2.48
Chemistry	8.63e+5	1135.22	21.2	0	0.02
Comp. Sc.	5.14e+6	421151.22	69558.12	1.45	4.91
Earth Sc.	5.57e+5	24435.34	382.81	0	0
Engineering	4.67e+5	12.99	104971.99	0.74	0.24
Fusion	3.05e+7	80.81	87.57	83.66	0
Mach. Learn.	3.90e+5	28126.52	6484.93	0.59	0
Materials	5.33e+6	7037.46	103.03	0.29	0.16
Physics	1.5e+7	1004.92	6644.35	25.76	31.34

(a) Mean number of read calls.

Sc. Domains	Number of write calls				
	<1M	1M-10M	10M-100M	100M-1G	>1G
Biology	5.41e+7	79.57	11.62	0.29	0.03
Chemistry	1.77e+7	117.08	305.52	0	0
Comp. Sc.	1.98e+6	406.57	5883.53	0.26	0.01
Earth Sc.	6.93e+4	48.97	7.49	0.10	0
Engineering	5.49e+5	1192.63	241313.12	0	0
Fusion	2.05e+3	325.89	959.40	239.85	0.17
Mach. Learn.	1.62e+3	89.91	1.48	0	0
Materials	4.49e+6	2.34	17.58	0.26	0.23
Physics	3.59e+6	959.99	44.69	1.50	0

(b) Mean number of write calls.

Observation: Computer Science and Chemistry have a high percentage of RI ML jobs which have all files in GPFS. Therefore, a large percentage of read-heavy files from Computer Science and Chemistry can be migrated from GPFS to BB to improve the I/O performance of the ML workloads.

4.1.4. I/O activity of ML I/O jobs in GPFS and BB

Fig. 4 shows the density distribution of I/O behavior (x-axis: bytes written, y-axis: bytes read) by ML jobs in both GPFS and BB classified by the four science domains that use BB. Based on the job types (RI, WI, RW) discussed above, the plot suggests that jobs which are nearer to the *x*-axis and further away from zero are WI, the jobs which are close to *y*-axis and far away from zero are RI, and the jobs in the middle are RW. Fig. 4 is consistent with Table 2 which shows that Computer Science and Chemistry ML jobs which use BB are mostly WI, while the ML jobs using BB from Materials and Biology are mostly RI.

Observation 1: ML jobs that use BB can either be classified as read-intensive or write-intensive, while a large majority of jobs use exclusively GPFS are read–write. This warrants further investigation into the read and write access sizes of the ML workloads, which is discussed in Section 4.1.5.

Observation 2: A large number of ML jobs performing fewer reads and writes (closer to zero along the y-axis) exclusively use GPFS. This shows that many ML users believe jobs performing less I/O will incur a much higher overhead in copying files from GPFS to BB than the gain in I/O performance by doing I/O on BB.

4.1.5. Read and write access sizes

Table 3(a) and Table 3(b) show the mean number of read and write calls used per ML job in different access sizes respectively. The various file access sizes are grouped into five bins: *< 1MB, 1MB–10MB, 10MB–100MB, 100MB–1GB,* and *> 1 GB.* This analysis is important because large sequential read and write (higher sized bins) gives a higher performance from GPFS, while small reads and writes (lower sized bins) are more efficient in BB. The classification by different science

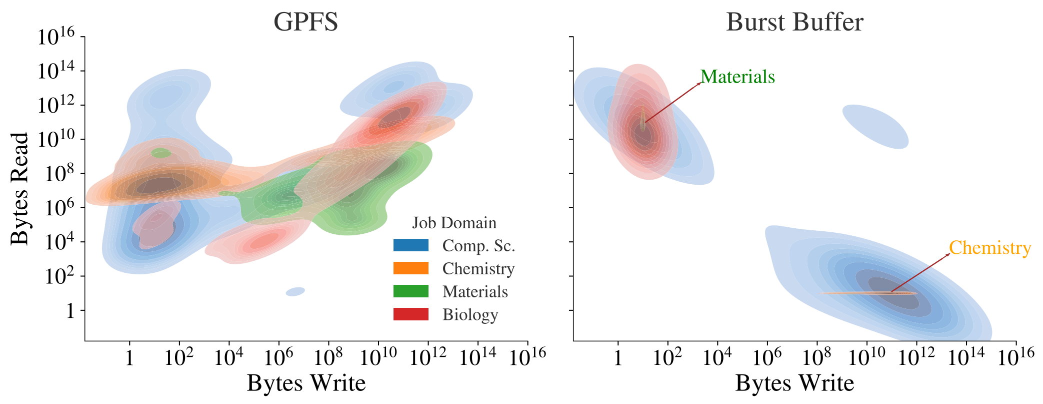
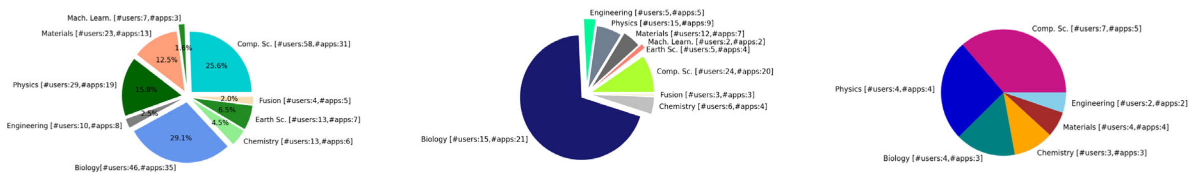


Fig. 4. Density distribution plots of I/O activity from ML jobs using either GPFS or BB classified by science domains.



(a) Small-scale ML jobs. **(b)** Medium-scale ML jobs. **(c)** Flagship-scale ML jobs.

Fig. 5. Classification of Small-Scale (18,269), Medium-Scale (5043), and Flagship-Scale (77) ML jobs. (**Note:** #users is the number of unique users submitting ML jobs, and #apps specifies the number of unique applications in each science domain).

Table 4
Summit scheduling groups by job node count.

Category	Number of Nodes	Max Walltime (hours)
Flagship	922–4608	24
Medium	46–921	6 or 12
Small	1–45	2

domains shows that the amount of calls across different bins depends on the science domain. At the same time, a large number of I/O calls in all the science domains fall under the smallest bin of $< 1MB$.

Observation: Almost 99% of the read and write calls for ML workloads are less than 10MB. This implies that burst buffer is an excellent candidate to improve I/O performance as a large number of small read and write requests overloads the parallel file system. There is a massive scope in I/O performance improvement, especially in the Physics domain, which comprises a healthy portion of ML I/O jobs on Summit as shown in Fig. 2. Physics has a large number of small file accesses but does not utilize burst buffer as previously seen in Fig. 3.

4.2. *Classification based on scale of jobs*

We define the scale of the jobs based on the number of nodes on which the job is executed. The need to classify jobs based on the number of nodes is because the bigger jobs that usually run on a large number of nodes affect the performance of the HPC system differently compared to smaller jobs that may have little impact on the HPC system performance.

4.2.1. *Distribution of ML I/O jobs*

Based on Summit's scheduling policy [22], the ML jobs are classified into three categories: small, medium, and flagship. The description for each category is shown in Table 4.

Fig. 5 shows the classification of small, medium and flagship scale ML jobs by science domains.

Observation: More than 78% of ML jobs run on less than 45 nodes with Biology and Computer Science having the maximum proportion of jobs. Biology has more than two-third share of the medium-scale ML jobs while Computer Science and Physics have the maximum share among the 77 ML jobs which run on flagship scale.

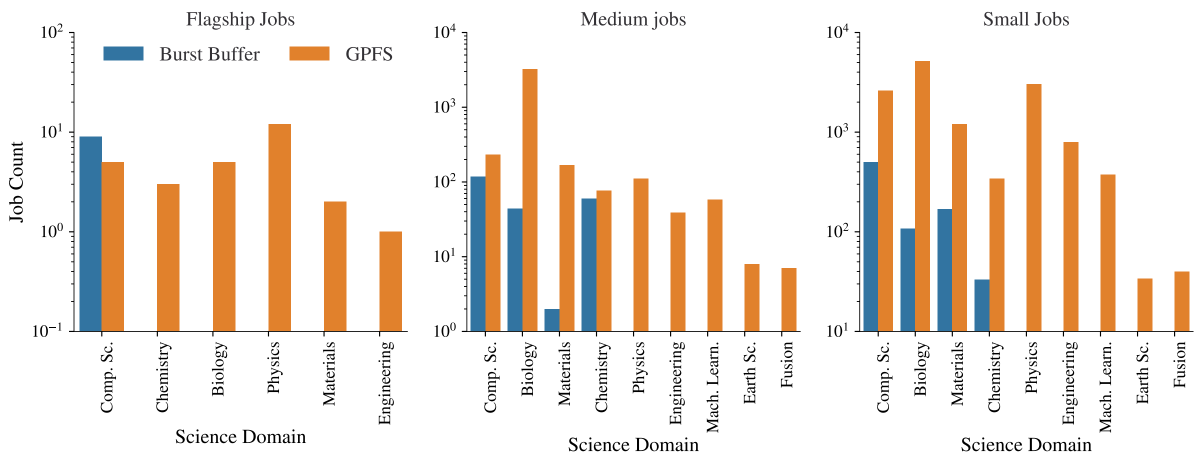


Fig. 6. Number of ML jobs which either access at least one file on BB or all files exclusively on GPFS classified on the basis of the scale of jobs and science domains.

Table 5
Comparison of percentage of read-intensive (RI), write-intensive (WI), and read–write (RW) jobs for GPFS and Burst Buffer classified by the scale of job runs.

Job Size	GPFS			Burst Buffer		
	RI	WI	RW	RI	WI	RW
Flagship	78.57	21.43	0	88.89	11.11	0
Medium	55.94	14.29	29.77	37.04	62.50	0.46
Small	56.43	30.71	12.86	52.24	46.76	0.99

4.2.2. Jobs using GPFS and BB

Fig. 6 shows the number of ML jobs that either has at least one file access on BB or all file accesses on GPFS.

Observation 1: Only ML users from Computer Science use BB for flagship jobs, while the ML users from other domains using BB; Biology, Materials, and Chemistry use it for jobs running on less than 922 nodes. The reason might be the multiple legacy codebases from the three science domains are in the process of being scaled up to include BB and improve I/O performance.

Observation 2: Typically, the number of jobs having at least one file on BB is lesser than the number of jobs using GPFS exclusively. Contrary to the pattern, the number of Computer Science jobs at the flagship scale, which uses BB, surpasses the number of jobs only using GPFS.

4.2.3. I/O activity for different types of jobs in GPFS and BB

The different job types: RI, WI, and RW, are described above in Section 4.1.3. Table 5 shows how the different types of jobs use GPFS and BB based on the scale of the ML jobs.

Observation 1: Flagship jobs which use exclusively GPFS are more read-intensive. Therefore, there is a scope of improvement in I/O performance if these RI ML jobs can migrate the read-heavy files from GPFS to BB.

Observation 2: Medium scale jobs which have at least one of their files on BB, use it more for WI jobs. This implies that ML users might not be confident of using BB for improved read performance and still prefer to use BB in a traditional manner, that is, for capturing periodic write bursts.

4.3. Temporal trend of ML I/O jobs

In this sub-section, we analyze the month-wise behavior of the ML jobs for one year period. The temporal analysis of ML workloads will reveal the I/O behavior of the emerging ML workloads.

4.3.1. High-level I/O trend

The evolution of the I/O characteristics of ML jobs over a period of one year is shown by Cumulative distribution function (CDF) plots for read and write bytes in Fig. 7.

Observation: More than 50% of the I/O happened in the later part of the year (starting mid-August). This suggests an exponential growth in the use of ML technologies in the HPC workloads and the importance of such a study to build better technologies to meet the future I/O needs of such ML applications.

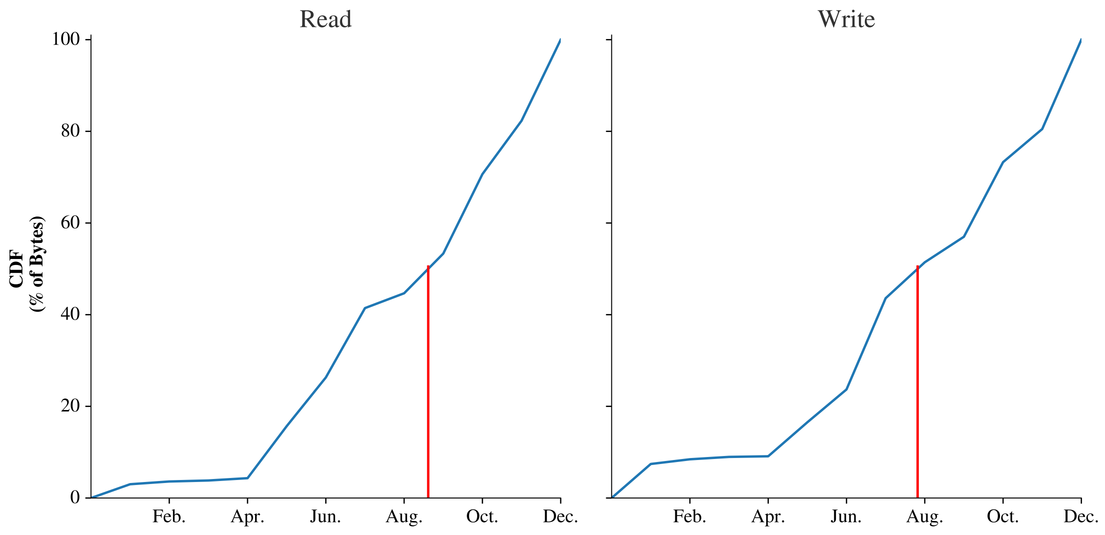


Fig. 7. Cumulative distribution function (CDF) plots for read bytes and write bytes for one year.

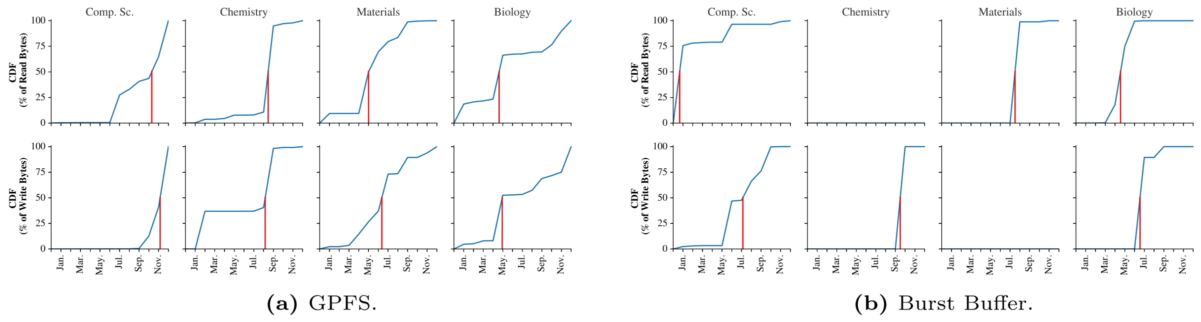


Fig. 8. CDF plots showing the temporal trend of bytes read and written by ML jobs on GPFS and BB over a period of one year classified by science domains.

4.3.2. ML I/O behavior trend of different science domains

Figs. 8(a) and 8(b) show the temporal trend of the percentage of bytes read and written by ML jobs on GPFS and BB over a period of one year classified by the four science domains that use BB.

Observation 1: Users from Computer Science started adopting BB for their ML jobs earlier than the other science domains. The steep jump in the usage of BB in Chemistry, Materials, and Biology suggests that only a few users used BB in a small time period. Fig. 8(b) follows the same trend as Fig. 7, where most of the ML jobs were run in the last few months of the year. Therefore, domain sciences outside Computer Science are starting to make use of BB in a more prevalent manner for their ML workloads, which will continually be on the rise. This means that better I/O optimization methods need to be developed to use burst buffer more efficiently.

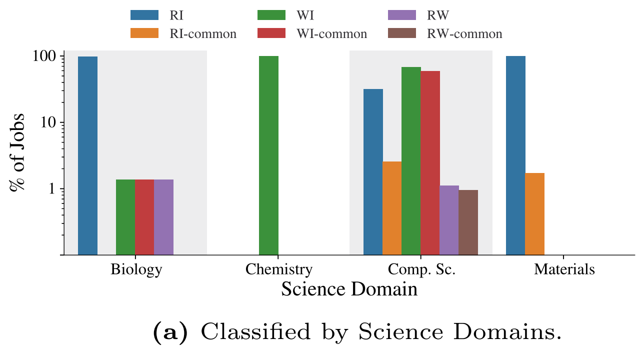
Observation 2: The temporal trend for reads and writes is similar on both GPFS and BB, except for Computer Science for reads on BB, which suggests that there might have been benchmark runs from the users in Computer Science to test the read performance of BB at the start of the year, before using BB in the ML applications.

4.4. Usage of burst buffer by ML I/O jobs

As seen in Section 4.1.5, a large number of small reads and writes constitute the I/O behavior of ML workloads which is more suitable for BB than GPFS. Therefore, in this section, we analyze the usage of BB by ML jobs. Out of a total of 23,389 ML jobs, only 1046 jobs make use of BB.

4.4.1. Jobs having common files in BB and GPFS

Write-intensive ML Jobs typically use BB for temporary writes and then persist those to GPFS. On the other hand, read-intensive ML jobs copy files from GPFS to BB, and then the files are read from BB to improve the overall read performance of the ML job. Therefore, from all the 1046 ML jobs that use BB, we analyze only the jobs with common files in both GPFS and BB and observe the distribution of bytes read and written by these common files.



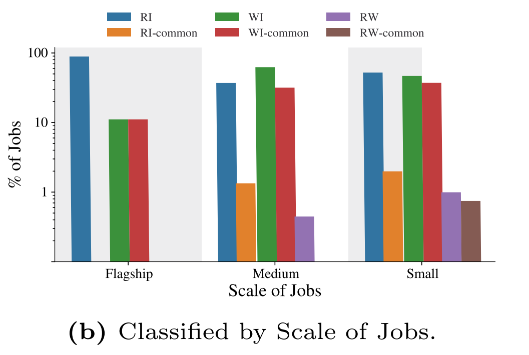


Fig. 9. Percentage of read-intensive (RI), write-intensive (WI), and read–write (RW) ML jobs which have common files across GPFS and burst buffer. (**Note:** *Common* for read-intensive jobs means that files were copied from GPFS to burst buffer and then read from burst buffer. *Common* for write-intensive jobs means that writes are persisted from burst buffer to GPFS).

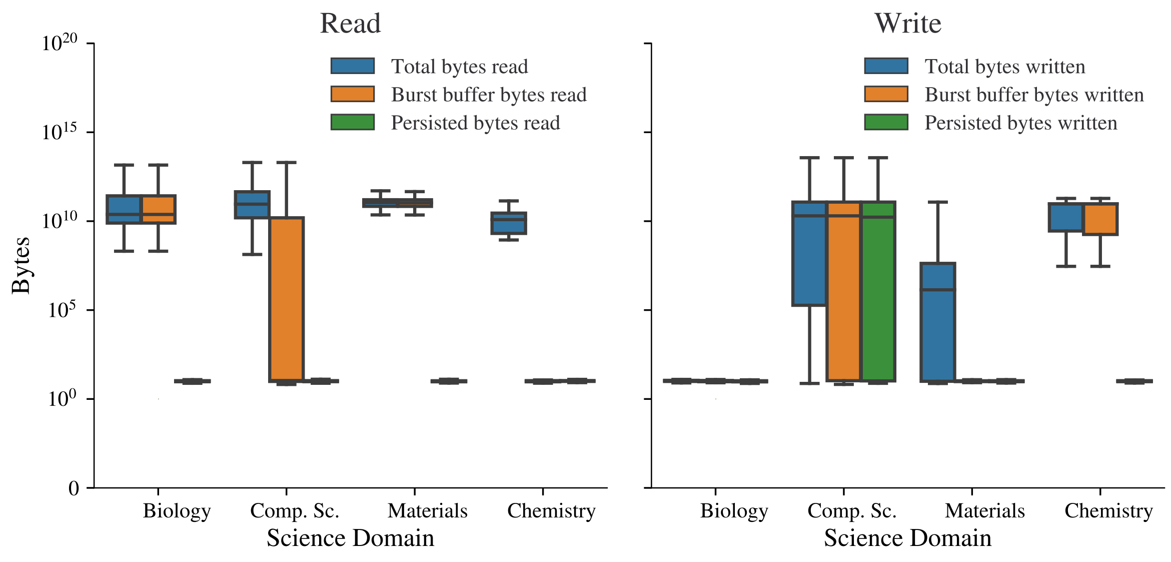


Fig. 10. Bytes persisted from burst buffer to GPFS classified based on reads and writes.

Figs. 9(a) and 9(b) show the percentage of RI, WI, and RW jobs classified based on the science domains and the scale of job runs that have common files on both GPFS and BB. We found a total of 396 jobs that have common files on GPFS and BB.

Observation 1: Fig. 9(a) shows that ML jobs from Biology performed persistent writes while the other types of jobs do not have any common files. This is completely opposite to the behavior exhibited by ML jobs from Chemistry, where the WI jobs do not have any files which are persisted. Jobs coming from Materials are only RI jobs, some of which use BB to improve read performance. ML users from Computer Science try to make optimal use of BB by having all three categories of jobs; RI, WI, and RW, use BB to either improve read performance or persist write-heavy files from BB to GPFS.

Observation 2: Fig. 9(b) shows that the trend of common files across all the scales of jobs is dominated by Computer Science shown in Fig. 9(a). However, as more ML technologies are adopted by science domains other than Computer Science, the usage of burst buffer will be skewed towards the sub-optimal BB usage. Therefore, for an optimal system-wide I/O performance, ML users need to be well educated on the benefits of BB as well as novel I/O optimization techniques similar to [23,24] should be developed which can transparently make use of BB without making changes to legacy code bases.

4.4.2. I/O activity of files persisted from BB to GPFS

Fig. 10 shows the I/O distribution of ML jobs which use BB to gauge the data size which are persisted from BB to GPFS. *Total bytes read/written* are the total bytes by the ML jobs which use BB. *Burst buffer bytes read/written* are the total bytes read or written from BB. *Persisted bytes read/written* shows the size of files which are persisted from BB to GPFS after they are read or written.

Table 6
Comparison of I/O performance for files in the ML jobs on GPFS and Burst Buffer.

I/O Rate	GPFS			Burst Buffer		
(MBps)	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Read	721	390	967	3576	2994	2518
Write	782	257	1285	2721	2807	1792

Table 7
I/O performance comparison for ML jobs using either exclusively GPFS or at least one file on Burst Buffer classified by science domains; (a) read rate, (b) write rate.

Read Rate	GPFS		Burst Buffer	
(MBps)	Mean	Median	Mean	Median
Biology	658.79	652.38	3319.09	2366.32
Chemistry	365.01	50.90	0	0
Comp. Sc.	724.03	399.09	4617.62	4455.59
Materials	709.75	38.39	5465.60	5535.77

(a)

Write Rate	GPFS		Burst Buffer	
(MBps)	Mean	Median	Mean	Median
Biology	220.71	85.30	3838.12	4560.81
Chemistry	281.58	280.39	2560.34	2753.97
Comp. Sc.	1216.89	826.57	2844.06	4041.16
Materials	124.30	2.89	0	0

(b)

Observation: Files which perform read bytes from BB are not persisted back to the GPFS. Also, as expected almost all the write bytes on BB from Computer Science are persisted to GPFS. However, this behavior does not hold true for Chemistry which do not persist the write bytes that were performed on BB. This might imply that ML jobs from Chemistry might write into a lot of temporary files which need not be persisted.

4.5. ML I/O job performance on GPFS and burst buffer

In this section, we analyze the performance benefit that can be observed by using the BB more efficiently by ML I/O jobs.

4.5.1. Read vs write performance

Table 6 compares the mean, median and standard deviation of read and write performance by files in the ML jobs on GPFS and BB.

Observation 1: Both read and write performance on BB outperforms GPFS. BB gives a better improvement on read performance (4.95x) when compared to the write performance (3.48x) than GPFS.

Observation 2: Based on the mean and standard deviation on BB, we can conclude that 66% of the reads and writes on BB get I/O performance between 1GBps and 6GBps. This is consistent with the theoretical peak that can be obtained on BB from a single node on Summit – 2.1 GBps for writing and 5.5 GBps for reading [25].

4.5.2. I/O performance by different science domains

Table 7(a) and Table 7(b) show the read and write performance of ML jobs classified by science domains when they either exclusively use GPFS or have at least one file access from BB. ML jobs from Chemistry do not perform any read on BB, and the jobs from Materials do not have writes on BB, therefore the corresponding values are zero.

Observation 1: The performance of both reads and writes on BB surpass those on GPFS for all the domains. This suggests that there is a huge scope of performance improvement by using BB more efficiently.

Observation 2: ML jobs from Computer Science have a better mean performance combining file accesses on both GPFS and BB, compared to other domains. This implies that Computer Science makes better use of BB and GPFS, for example, not using BB for ML jobs doing small overall I/O, copying read-heavy files from GPFS to BB before doing the reads on BB, capturing burst of writes for write-heavy files on BB before persisting it on GPFS. This again suggests that there should be better optimization technologies that can transparently migrate files between GPFS and BB for the domain sciences who are less informed about BB.

4.6. ML I/O job performance based on processing unit usage

From the power dataset, we get the mean power consumption on the CPU and GPU per job. Combining the power data with the Darshan dataset, we are able to analyze the I/O performance of ML jobs based on CPU and GPU usage by jobs.

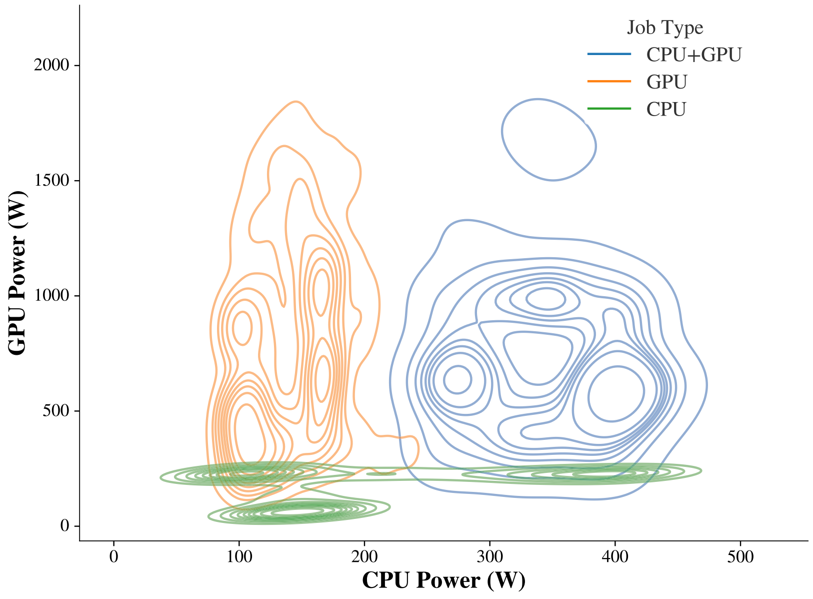


Fig. 11. Density distribution plot based on the classification of 15,165 ML jobs by processing units. The jobs are classified into CPU only, GPU only, and CPU and GPU jobs.

Table 8
Classification of ML jobs into RI, WI and RW for CPU and GPU usage.

Job Type	CPU	GPU	CPU+GPU
Read-Intensive (RI)	4297	4588	112
Write-Intensive (WI)	2421	1170	221
Read–Write (RW)	1670	401	241

4.6.1. Classification of ML jobs

We augment the Darshan dataset that we used for the I/O pattern analysis with the power dataset to classify workloads into three classes that are GPU only jobs, CPU+GPU jobs that run on both CPU and GPU, and remaining jobs which are CPU only jobs. The classification of jobs is based on comparing the mean power consumption of CPUs and GPUs per node. If the mean power consumption of CPUs is greater than 250 W and the mean power consumption of GPUs greater than 260 W, we label the jobs as CPU + GPU jobs (jobs that use both CPU and GPU). If the mean power consumption of GPUs is more than 260 W and the mean CPU power is less than 250 W, we label the jobs as GPU only jobs. The remaining jobs that consume GPU power less than or equal to 260 W are labeled CPU jobs, including jobs that use low CPU power, i.e., CPU power consumption is less than 250 W. The equations below summarize the classification of jobs into three categories.

- *CPU Jobs: GPU Power $\leq$ 260 W*
- *GPU Jobs: (GPU Power $>$ 260 W) and (CPU Power $\leq$ 250 W)*
- *CPU+GPU Jobs: (GPU Power $>$ 260 W) and (CPU Power $>$ 250 W)*

Fig. 11 shows GPU power vs. CPU power consumption, kernel density estimate plot (KDE) of the ML jobs in the three categories, i.e., CPU only, GPU only, and jobs that uses both CPU and GPU (CPU+GPU) jobs. A KDE plot [26] is a visualization technique to show the continuous probability distribution of observations in a dataset. It is analogous to a histogram, but a histogram bins the data points in discrete bins. In contrast, KDE smooths the observation with a Gaussian kernel resulting in a continuous density estimate or continuous probability density curve.

Observation: Based on the classification, the number of ML jobs using only CPU is 8294. The number of ML jobs utilizing only GPUs for computation and processing is 6291, while a smaller number of 580 ML jobs use both CPUs and GPUs for processing and computation tasks.

4.6.2. Classification based on I/O activity

Table 8 shows the further classification of each group of ML job into read-intensive, write-intensive and read–write jobs.

Observation: For CPU only or GPU only jobs, the number of read-intensive jobs largely surpasses the number of write-intensive or read–write jobs. This concurs with the popular opinion that ML jobs have more reads than writes. However,

Table 9
Classification of ML jobs based on science domains and processing units.

Science Domains	CPU	GPU	CPU+GPU
Biology	4947	1977	301
Chemistry	155	150	98
Comp. Sc.	1121	1505	38
Earth Sc.	209	388	4
Engineering	52	14	0
Fusion	3	0	44
Mach. Learn.	93	208	0
Materials	729	543	38
Physics	985	1506	57

Table 10
Classification of ML jobs into GPFS and BB jobs based on science domains and processing units.

Science Domains	CPU		GPU		CPU+GPU	
	BB	GPFS	BB	GPFS	BB	GPFS
Biology	81	4866	26	1951	11	290
Chemistry	2	153	37	113	44	54
Comp. Sc.	113	1008	306	1199	11	27
Materials	3	726	161	382	0	38

the ML jobs that use both CPUs and GPUs for their processing, do not have a clear distinction in the read–write intensity and are therefore difficult to optimize.

4.6.3. Classification based on science domains

Table 9 shows the classification of ML jobs based on different science domains and how they use the processing units. Jobs classified under the Machine Learning domain are jobs from new proposals which are specifically labeled as ML. The jobs in other domains have more traditional HPC workloads. Therefore, this section provides the comparison of jobs from the ML domain with jobs from other domains.

Observation: ML jobs originating from Biology majorly use CPUs than GPUs. However, ML jobs from Computer Science and Physics domains use more GPUs. Jobs which label themselves as machine learning (ML domain) do not use a combination of CPUs and GPUs. They majorly use GPUs which suggests that the emerging ML workloads are shifting towards using more GPUs than CPUs or a combination of both.

4.6.4. Classification based on usage of GPFS and BB

As seen in the previous sections, there are only four science domains, namely, Biology, Chemistry, Computer Science, and Materials, which use BB for their ML jobs. Table 10 shows the number of BB jobs based on the usage of processing units.

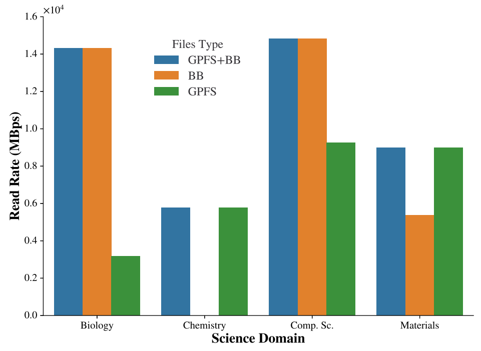
Observation: ML jobs which use only GPUs have a higher tendency of also using BB for their file accesses. The explanation for this usage behavior is due to the increased processing power provided by GPUs, which, if used purely on GPFS for file accesses, will lead to a much higher I/O bottleneck.

4.6.5. I/O performance of BB jobs

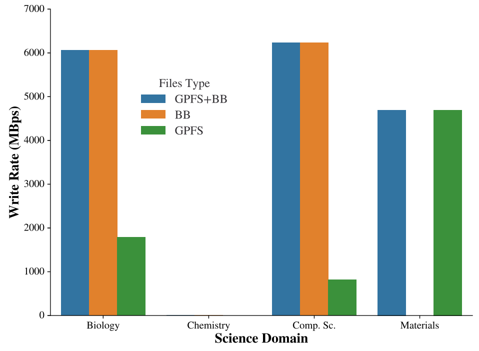
We find the I/O performance of BB ML jobs based on their processing unit usage. Fig. 12 shows the I/O performance of BB jobs classified according to science domains and the usage on processing units. For every science domain, we plot the overall performance of the jobs, followed by the performance of files accessed on BB and GPFS, respectively. Fig. 12(a–b) shows the read and write performance of CPU only BB jobs, Fig. 12(c–d) demonstrates the read and write performance of GPU only BB jobs, and similarly, read and write performance of jobs that utilized both CPU and GPU during their execution is shown in Fig. 12(e–f).

Observation 1: The mean I/O performance of BB jobs originating from Biology and Computer Science have a much greater dependency on the BB performance than the jobs which originate from Chemistry and Materials, irrespective of the usage of processing units. For example, BB Jobs from Biology in Fig. 12(c) has the overall read performance very similar to the performance of files accessed on BB. In contrast, the BB jobs in Chemistry in Fig. 12(f) has the overall mean performance similar to the files accessed from GPFS. This discrepancy in the I/O performance might be related to the read/write pattern on BB as discussed in the previous sections.

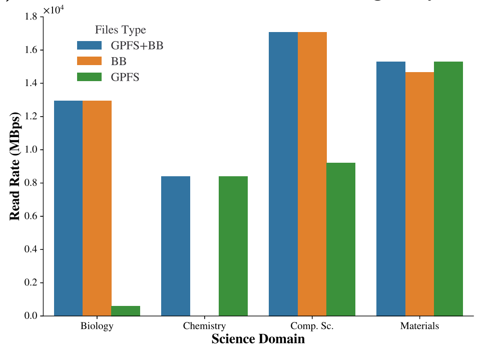
Observation 2: GPU only BB jobs from Materials and Computer Science yield a much higher performance for both reads and writes compared to BB jobs which use only CPU or both CPU and GPU. The I/O performance of BB jobs from other science domains (Biology and Chemistry) are similar. This suggests that jobs which use BB for file accesses and GPUs for processing have a much higher chances of achieving a better I/O performance.



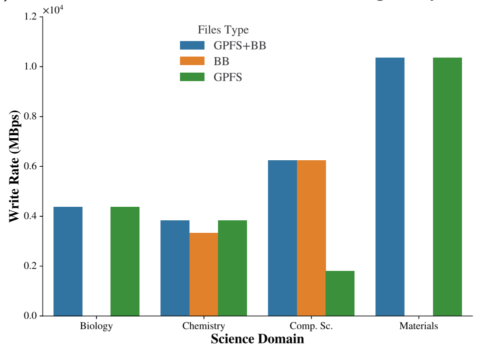
(a) Read Performance of BB Jobs Using Only CPU.



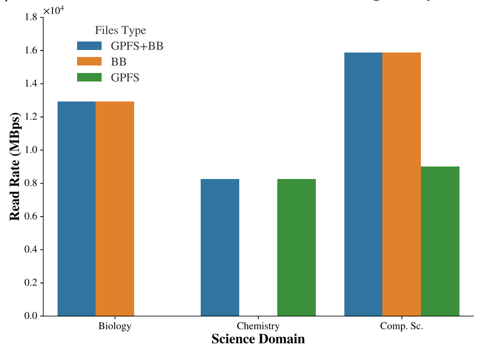
(b) Write Performance of BB Jobs Using Only CPU.



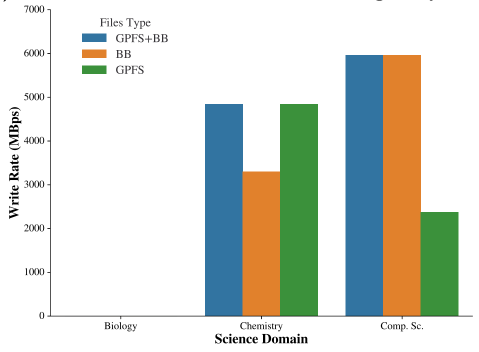
(c) Read Performance of BB Jobs Using Only GPU.



(d) Write Performance of BB Jobs Using Only GPU.



(e) Read Performance of BB Jobs Using CPU and GPU.



(f) Write Performance of BB Jobs Using CPU and GPU.

Fig. 12. I/O Performance of BB ML jobs based on usage of processing units.

4.6.6. I/O performance of GPFS jobs

The read and write performance of ML jobs which do not use BB but only use GPFS to access files are shown in Table 11.

Observation 1: ML jobs which use only GPUs in association with GPFS tend to get much better read and write performance than the jobs which use only CPUs or a combination of CPUs and GPUs. It might be because of prefetching data into the GPU memory which is not possible for jobs using CPUs.

Observation 2: Read performance in general surpasses the write performance from GPFS. However, the trend is not true for ML jobs in Fusion and Physics. This might be because of the file sizes that the jobs from these two domains access. Large number of small reads vs a big write can explain such behavior which is synonymous to the behavior of ML jobs which are not optimized for I/O.

4.6.7. Is I/O really the bottleneck for ML applications?

In this section, we aim to check if I/O is the real bottleneck for ML applications. First we see the percentage of I/O time that contributes to the application total runtime. The following is the mean I/O percentage for the different types of ML

Table 11
Read and write performance (MBps) of ML jobs which access files only from GPFS classified based on science domains and processing unit usage.

Science Domains	CPU		GPU		CPU+GPU	
	Read	Write	Read	Write	Read	Write
Biology	19779	11303	20400	12302	19809	8966
Chemistry	9144	5090	9113	4923	8550	10306
Comp. Sc.	12570	11386	20356	10669	11477	3113
Earth Sc.	40179	4127	40394	10774	8438	1836
Engineering	6433	11917	5086	36	0	0
Fusion	2879	10048	0	0	2550	11501
Mach. Learn.	4919	2749	10893	10554	0	0
Materials	22830	10240	16635	2992	7298	889
Physics	9464	11686	8072	9888	6372	7960

Table 12
Dependency of job runtime on the I/O time for different kinds of jobs for two applications — train_cvae, and bert_horovod.

Job Type	#Nodes	Data Transfer Size (bytes)	Job Runtime (s)	I/O Time (s)
GPU Job	32	3.78e+07	597	9.08
1(train_cvae)	32	3.78e+07	595	5.15
GPU Job	32	9.8e+10	478	138.31
2(train_cvae)	32	9.8e+10	455	65.26
CPU Job	2	2.69e+10	187	8.64
1(train_cvae)	2	2.88e+10	176	7.86
CPU Job	6	1.58e+11	4851	33.39
2(bert_horovod)	6	1.58e+11	3930	34.92

jobs based on the processing unit usage.

- CPU jobs: 1.35%
- GPU jobs: 16.47%
- CPU+GPU jobs: 16.12%

Observation 1: ML jobs that use GPUs generally have a more significant I/O percentage, which indicates that the workloads that run on GPUs perform computation part quickly and are likely to have higher chances of I/O bottleneck.

Next, we see if such an I/O percentage causes an effect on the job runtime. To answer this question, we consider an application from the Biology science domain. The application is *train_cvae*. We chose this application because this was run on only CPUs as well as only GPUs. We also choose another application - *bert_horovod* to showcase the behavior for CPU only ML job. The result is shown in Table 12. We also observed a similar pattern for I/O time and job runtime for other applications and used *train_cvae* and *bert_horovod* as a representative of those applications. In this paper, we use the two applications as examples. However, this was a trend that was observed in most of the ML applications.

Observation 2: For the same or similar data transfer size and running on the same number of nodes, ML jobs which run on GPUs show that job runtime is directly proportional to the I/O time. However, this is not true for CPU only jobs. This means that I/O performance deeply impacts the job runtime for jobs which use only GPUs rather than jobs which use only CPUs. This contradicts the popular opinion that having more GPUs would improve the ML job runtime. For more GPUs, the I/O bottleneck might have a larger impact which means that I/O optimization should go hand-in-hand with increasing processing power.

5. Discussion

The analysis of I/O behavior of 23,389 ML jobs (15,165 ML jobs from power usage on processing units) provides valuable insights into the future of HPC storage systems. This study is focused on Summit — the world's second-fastest supercomputer. However, the ML jobs studied in this paper represent other leadership scale HPC systems as well.

Recently, a study of the I/O behavior for all of Summit workloads (that is, traditional HPC and emerging ML) for the year 2020 [27] points out the BB usage pattern. On comparing that study with our study of ML jobs running on Summit for the year 2020, there is an observation that many domain sciences, such as Physics and Earth Science use BB for the traditional HPC workloads but not for ML workloads. However, the number of read and write calls which are below 1MB is of the same magnitude. This shows the difference in the usage of the storage system by traditional HPC workloads and the scope for improvement in the emerging ML workloads.

5.1. Lessons for domain scientists

- ML workloads from all science domains generate a large number of small file reads and writes, which is better suited for BB. However only few science domains use BB for their ML workloads. Therefore, domain scientists need to be trained to use BB for their ML workloads.
- HPC ML users from Computer Science makes use of BB more efficiently which results in the much better I/O performance compared to other science domains which also use BB. It is observed that ML workloads from Computer Science copies read-intensive files from GPFS to the BB and performs most reads from BB, and burst of small writes also happen on BB which is later persisted to the GPFS. This means that other domain scientists should also be trained to use BB more efficiently to yield better I/O performance.
- Irrespective of using CPUs or GPUs, few science domains like Fusion and Physics have write performance better than read performance when the reads and writes happen from the parallel file system. This suggests that the reads of such ML jobs are not optimized, and the I/O performance of such applications can be improved by optimizing the reads, either by coalescing the reads or by using BB.

5.2. Lessons for storage architects

- The temporal trend of ML workloads shows that there is an exponential increase in the I/O activity from ML workloads which is indicative of the future which will be dominated by ML. Therefore better storage solutions need to be designed that can handle the diverse I/O patterns from future HPC ML I/O workloads.
- It can sometimes be difficult to modify legacy code bases from various science domains to include the usage of burst buffer. I/O optimization techniques must be developed that can help use the burst buffer transparently without modifying the application code.
- The trend for ML jobs is to shift the processing from CPUs to GPUs to make use of the increased processing power of GPUs. However, our study indicates that GPU only ML jobs are much highly affected by I/O bottlenecks. Therefore, storage architects should focus on developing I/O optimization techniques for GPU only jobs rather than pushing for more GPUs in the HPC clusters.
- This study also provides guidance on the capacity of future HPC storage systems, which will be dominated by ML workloads.

6. Conclusion

There is an exponential increase in the use of ML technologies in HPC I/O workloads by various science domains. Therefore, understanding the I/O characteristics of ML workloads is very important for system architects, file system developers and HPC users. This paper analyzed the Darshan logs of more than 23,000 ML workloads running on Summit. The Darshan logs were also merged with the dataset containing the processing units power usage to get the CPU and GPU usage of 15,165 ML jobs. Our study showed that ML workloads generate a large number of small file reads and writes which is ideal for a burst buffer compared to a parallel file system. However not many science domains use burst buffer efficiently for their ML I/O workloads. Even the different scale of jobs affected the I/O usage of ML workloads, where only applications from Computer Science used burst buffer for running jobs on more than 921 nodes. This is a lesson for both domain scientists to use burst buffer more efficiently for their ML workloads, and also for storage architects to build better storage solutions for future large-scale HPC storage systems. We also noticed that processing on GPUs and I/O on BB is the ideal combination for getting the best I/O performance for ML jobs. If ML jobs use only parallel file system, then the reads needs to be optimized. Also a much higher I/O bottleneck is observed for ML jobs running on only GPUs rather than only CPUs. This paper therefore provides critical insights to build better storage system optimization tools for the storage architects and gives ML users techniques to achieve better I/O performance when running emerging ML workloads. In the future, we will use this study as well as a few ML techniques developed for Darshan trace generation [28] to build a tool that can characterize ML workloads from Darshan without the need of ML keywords.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Hyogi Sim for his suggestions and inputs for the paper. This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at the Oak Ridge National Laboratory, which is supported by the Office of Science of the DOE under Contract DE-AC05-00OR22725.

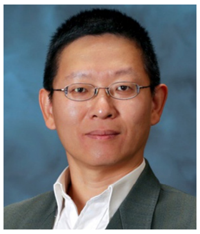
References

[1] N. Naksinehaboon, Y. Liu, C. Leangsuksun, R. Nassar, M. Paun, S.L. Scott, Reliability-aware approach: An incremental checkpoint/restart model in HPC environments, in: International Symposium on Cluster Computing and the Grid, CCGRID, IEEE, 2008, pp. 783–788.
[2] F. Pan, Y. Yue, J. Xiong, D. Hao, I/O characterization of big data workloads in data centers, in: Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware, Springer, 2014, pp. 85–97.
[3] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, R. Ross, Understanding and improving computational science storage access through continuous characterization, ACM Trans. Storage (TOS) 7 (3) (2011) 1–26.
[4] B.K. Pasquale, G.C. Polyzos, Dynamic I/O characterization of I/O intensive scientific applications, in: ACM/IEEE Conference on Supercomputing, 1994, pp. 660–669.
[5] S. Narayan, J.A. Chandy, I/O characterization on a parallel file system, in: International Symposium on Performance Evaluation of Computer & Telecommunication Systems, SPECTS, IEEE, 2010, pp. 133–140.
[6] F. Chowdhury, Y. Zhu, T. Heer, S. Paredes, A. Moody, R. Goldstone, K. Mohror, W. Yu, I/O characterization and performance evaluation of beegfs for deep learning, in: International Conference on Parallel Processing, ICPP, 2019, pp. 1–10.
[7] S.W. Chien, A. Podobas, I.B. Peng, S. Markidis, tf-Darshan: Understanding fine-grained I/O performance in machine learning workloads, in: International Conference on Cluster Computing, CLUSTER, IEEE, 2020, pp. 359–370.
[8] G.K. Lockwood, S. Snyder, S. Byna, P. Carns, N.J. Wright, Understanding data motion in the modern HPC data center, in: IEEE/ACM International Parallel Data Systems Workshop, PDSW, 2019, pp. 74–83.
[9] A.K. Paul, O. Faaland, A. Moody, E. Gonsiorowski, K. Mohror, A.R. Butt, Understanding HPC application I/O behavior using system level statistics, in: International Conference on High Performance Computing, Data, and Analytics, HiPC, IEEE, 2020, pp. 202–211.
[10] H. Luu, B. Behzad, R. Aydt, M. Winslett, A multi-level approach for understanding I/O activity in HPC applications, in: IEEE International Conference on Cluster Computing, CLUSTER, 2013, pp. 1–5.
[11] S. Wright, S. Hammond, S. Pennycook, R. Bird, J. Herdman, I. Miller, A. Vadgama, A. Bhalerao, S. Jarvis, Parallel file system analysis through application I/O tracing, Comput. J. (ISSN: 0010-4620) 56 (2) (2012) 141–155.
[12] Darshan - HPC I/O characterization tool, 2021, https://www.mcs.anl.gov/research/projects/darshan/. (Accessed July 17 2021).
[13] S. Snyder, P. Carns, K. Harms, R. Ross, G.K. Lockwood, N.J. Wright, Modular HPC I/O characterization with darshan, in: Workshop on Extreme-Scale Programming Tools, ESPT, IEEE, 2016, pp. 9–17.
[14] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, K. Riley, 24/7 characterization of petascale I/O workloads, in: IEEE International Conference on Cluster Computing and Workshop, 2009, pp. 1–10.
[15] T. Wang, K. Mohror, A. Moody, K. Sato, W. Yu, An ephemeral burst-buffer file system for scientific applications, in: The International Conference for High Performance Computing, Networking, Storage and Analysis, SC, IEEE, 2016, pp. 807–818.
[16] IBM spectrum scale (GPFS), 2021, https://ibm.com/products/spectrum-scale. (Accessed July 17 2021).
[17] Summit, 2021, https://www.olcf.ornl.gov/summit/. (Accessed July 17 2021).
[18] Top 500 - june 2021, 2021, https://www.top500.org/lists/top500/2021/06/. (Accessed July 17 2021).
[19] A.K. Paul, A.M. Karimi, F. Wang, Characterizing machine learning I/O workloads on leadership scale HPC systems, in: 2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, IEEE, 2021, pp. 1–8.
[20] T. Patel, S. Byna, G.K. Lockwood, N.J. Wright, P. Carns, R. Ross, D. Tiwari, Uncovering access, reuse, and sharing characteristics of I/O-intensive files on large-scale production HPC systems, in: 18th USENIX Conference on File and Storage Technologies, FAST 20, 2020, pp. 91–101.
[21] W. Shin, V. Oles, A.M. Karimi, J.A. Ellis, F. Wang, Revealing power, energy and thermal dynamics of a 200PF pre-exascale supercomputer, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC, 2021, pp. 1–14.
[22] Summit scheduling policy, 2021, https://docs.olcf.ornl.gov/systems/summit_user_guide.html#scheduling-policy. (Accessed July 16 2021).
[23] A. Kougkas, H. Devarajan, X.-H. Sun, Hermes: A heterogeneous-aware multi-tiered distributed I/O buffering system, in: International Symposium on High-Performance Parallel and Distributed Computing, HPDC, 2018, pp. 219–230.
[24] A. Kougkas, H. Devarajan, X.-H. Sun, I/O acceleration via multi-tiered data buffering and prefetching, J. Comput. Sci. Tech. 35 (1) (2020) 92–120.
[25] Burst buffer on summit, 2021, https://docs.olcf.ornl.gov/systems/summit_user_guide.html#burst-buffer. (Accessed July 16 2021).
[26] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Routledge, 2018.
[27] J.L. Bez, A.M. Karimi, A.K. Paul, B. Xie, S. Byna, P. Carns, S. Oral, F. Wang, J. Hanley, Access patterns and performance behaviors of multi-layer supercomputer I/O subsystems under production load, in: Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing, HPDC, 2022, pp. 43–55.
[28] A.K. Paul, J.Y. Choi, A.M. Karimi, F. Wang, Machine learning assisted HPC workload trace generation for leadership scale storage systems, in: Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing, HPDC, 2022, pp. 199–212.

Ahmad Maroof Karimi is an HPC Operational Data Scientist at Analytics and AI methods at Scale Group (AAIMS) at National Center for Computational Sciences at Oak Ridge National Laboratory (ORNL). His current research works are in the areas of large-scale distributed data analytics and machine learning for modeling and prediction of HPC operations, in particular, HPC storage systems and power facility operations.

Prior to joining ORNL, Dr. Karimi received his Ph.D. in the area of data mining and machine learning from Department of Computer and Data Sciences at Case Western Reserve University.

Arnab K. Paul is an Assistant Professor in the Department of Computer Science and Information Systems at BITS Pilani, K K Birla Goa Campus, India. Prior to joining BITS Pilani, he was a postdoctoral research associate in the AI, Analytics & Scalable Methods Group at the Oak Ridge National Laboratory, USA. He completed his Ph.D. in Computer Science at Virginia Tech, USA. His research interests include high performance computing, Internet of Things, distributed systems, parallel file systems, edge computing, and big data analysis.



Feiyi Wang received his Ph.D. in Computer Engineering from North Carolina State University (NCSU). Prior joining the Oak Ridge National Laboratory (ORNL), he was a principal research scientist at Microelectronic Center of North Carolina (MCNC) and the lead PI and Co-PI for several DARPA-funded projects. He is a currently a Senior Research Scientist and the Group Leader of Analytics and AI methods at Scale Group (AAIMS) at National Center for Computational Sciences of ORNL. His research interests include large-scale data analytics, distributed machine learning and scalable benchmarking, high performance storage system, parallel I/O and file systems. Dr. Wang held Joint Faculty Professor of ECE Department, Bredesen Center Faculty position at University of Tennessee. He is also a Senior Member of IEEE.