Characterizing Machine Learning I/O Workloads on Leadership Scale HPC Systems

Arnab K. Paul†, Ahmad Maroof Karimi†, Feiyi Wang
Oak Ridge National Laboratory, USA
{*paula, karimiahmad, fwang2*}*@ornl.gov*

***Abstract*—High performance computing (HPC) is no longer solely limited to traditional workloads such as simulation and modeling. With the increase in the popularity of machine learning (ML) and deep learning (DL) technologies, we are observing that an increasing number of HPC users are incorporating ML methods into their workflow and scientific discovery processes, across a wide spectrum of science domains such as biology, earth science, and physics. This gives rise to a diverse set of I/O patterns than the traditional checkpoint/restart-based HPC I/O behavior. The details of the I/O characteristics of such ML I/O workloads have not been studied extensively for large-scale leadership HPC systems. This paper aims to fill that gap by providing an in-depth analysis to gain an understanding of the I/O behavior of ML I/O workloads using darshan - an I/O characterization tool designed for lightweight tracing and profiling. We study the darshan logs of more than $23,000$ HPC ML I/O jobs over a time period of one year running on Summit - the second-fastest supercomputer in the world. This paper provides a systematic I/O characterization of ML I/O jobs running on a leadership scale supercomputer to understand how the I/O behavior differs across science domains and the scale of workloads, and analyze the usage of parallel file system and burst buffer by ML I/O workloads.**

***Keywords*-Burst Buffer, Darshan, High Performance Computing, HPC Storage, IBM Spectrum Scale, I/O Characterization, Machine Learning, Parallel File System**

I. Introduction

The I/O needs of high performance computing (HPC) workloads have been historically dominated by write-intensive modeling and simulation workflows. However, the increasing popularity of machine learning (ML) methods to solve complex problems in various areas, such as biology, astrophysics, and chemistry, has given rise to varied I/O patterns in large-scale data analysis and ML/deep learning (DL) workloads [1]. This diversity of I/O needs by HPC workloads warrants the need to characterize the requirements of I/O in modern HPC centers.

There has been a multitude of studies [2]–[5] dedicated to the characterization of write-heavy checkpoint/restart simulation workloads. Recent studies [6]–[9] have also focused on analyzing the I/O requirements of popular ML methods, like deep learning. It has been typically seen that ML workloads have small read and write access patterns. However, there lacks a holistic understanding of the I/O characteristics of typical ML workloads based on different domain sciences and the scale of ML jobs on leadership scale HPC systems.

Application users typically use various I/O profiling tools to characterize their I/O workloads. These tools are valuable to provide insights into potential performance-tuning efforts and enable evaluation of I/O trends for the entire HPC cluster. Darshan [10]–[12] is one of the most popular HPC I/O characterization tools. It is designed to capture an accurate picture of application I/O behavior, including properties such as patterns of access within files, with minimum overhead. 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 in characterizing 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 [13] will provide better I/O performance for ML workloads than a parallel file system, like IBM Spectrum Scale (GPFS) [14], 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.

In this paper, we aim to fill the gap in the 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 [15] - the second-fastest supercomputer in the world according to the latest top-500 list [16]. IBM Spectrum Scale (previously known as

† Made equal contribution to this work.

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

Table I: List of 42 ML keywords used to annotate ML jobs from darshan logs.

GPFS) [14] 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.

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.
- 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.

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.

II. Background

A. Summit Supercomputer

The Summit supercomputer is based upon IBM AC922 system and deployed at the Oak Ridge Leadership Computing Facility (OLCF). It consists of 4,608 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. An InfiniBand EDR network with a fat-tree topology connects the nodes. 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.

B. Darshan - HPC I/O Characterization Tool

Figure 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. Therefore, the STDIO module characterizes the *stdio.h* family of functions, such as $fopen()$, $fprintf()$, and $fscanf()$.

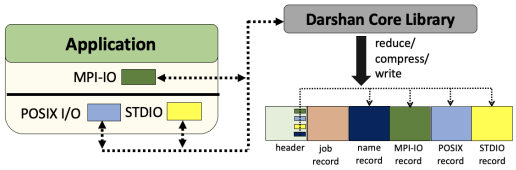


Figure 1: High-level overview of darshan's architecture.

C. 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.

To identify ML I/O workloads from the parsed darshan logs, we created 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 I. 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 I, 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.

III. Analysis of ML I/O Workloads

A. Classification Based on Science Domains

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. Figure 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 II-C, a total of $23,389$ ML jobs were analyzed.

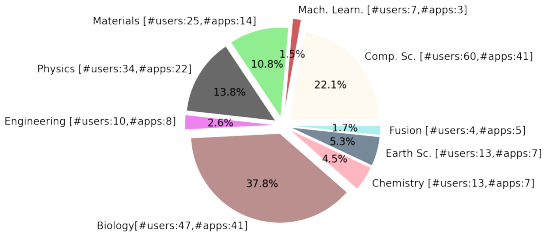


Figure 2: Classification of $23,389$ ML jobs by science domains. (**Note:** #users specifies the number of unique users submitting ML jobs, and #apps is the number of unique applications in each science domain).

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.

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

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.

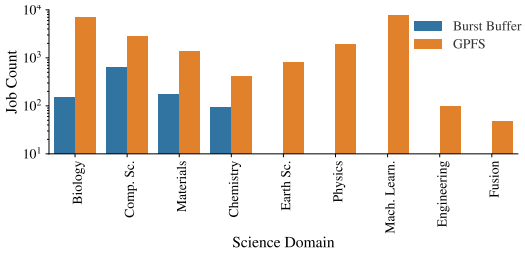


Figure 3: The number of ML jobs using burst buffer and GPFS classified by different science domains on Summit. (**Note:** BB jobs access at least one file from the BB. GPFS jobs access all the files exclusively from GPFS).

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).

Equation 1 gives the job type based on the following conditions, where *result* is the answer from Equation 1.

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

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

Table II 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.

	GPFS			Burst Buffer		
Job Size	**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 II: 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.

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) I/O Activity of ML I/O Jobs in GPFS and BB: Figure 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. Therefore, Figure 4 is consistent with Table II which

Sc. Domains	Number of Read Calls					Number of Write Calls				
	<1M	1M - 10M	10M - 100M	100M - 1G	>1G	<1M	1M - 10M	10M - 100M	100M - 1G	>1G
Biology	$2.92e+7$	922.12	678.61	70.07	2.48	$5.41e+7$	79.57	11.62	0.29	0.03
Chemistry	$8.63e+5$	1135.22	21.2	0	0.02	$1.77e+7$	117.08	305.52	0	0
Comp. Sc.	$5.14e+6$	421151.22	69558.12	1.45	4.91	$1.98e+6$	406.57	5883.53	0.26	0.01
Earth Sc.	$5.57e+5$	24435.34	382.81	0	0	$6.93e+4$	48.97	7.49	0.10	0
Engineering	$4.67e+5$	12.99	104971.99	0.74	0.24	$5.49e+5$	1192.63	241313.12	0	0
Fusion	$3.05e+7$	80.81	87.57	83.66	0	$2.05e+3$	325.89	959.40	239.85	0.17
Mach. Learn.	$3.90e+5$	28126.52	6484.93	0.59	0	$1.62e+3$	89.91	1.48	0	0
Materials	$5.33e+6$	7037.46	103.03	0.29	0.16	$4.49e+6$	2.34	17.58	0.26	0.23
Physics	$1.5e+7$	1004.92	6644.35	25.76	31.34	$3.59e+6$	959.99	44.69	1.50	0

Table III: 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.

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.

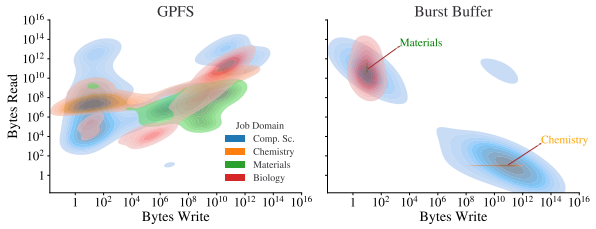


Figure 4: Density distribution plots of I/O activity from ML jobs using either GPFS or BB classified by science domains.

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 being read-write. This warrants further investigation into the read and write access sizes of the ML workloads, which is discussed in Section III-A5.

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.

5) Read and Write Access Sizes: Table III shows the mean number of read and write calls used per ML job in different access sizes classified by different science domains. The various file access sizes are grouped into five bins: $<$ *1MB, 1MB - 10MB, 10MB - 100MB, 100MB - 1GB,* and $>$ *1GB.* 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.

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 Figure 2. Physics has a large number of small file accesses but does not utilize burst buffer as previously seen in Figure 3.

B. Classification Based on Scale of Jobs

1) Distribution of ML I/O Jobs: Based on Summit's scheduling policy [17], the ML jobs are classified into three categories: small, medium, and flagship. The description for each category is shown in Table IV.

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

Table IV: Summit scheduling groups by job node count.

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

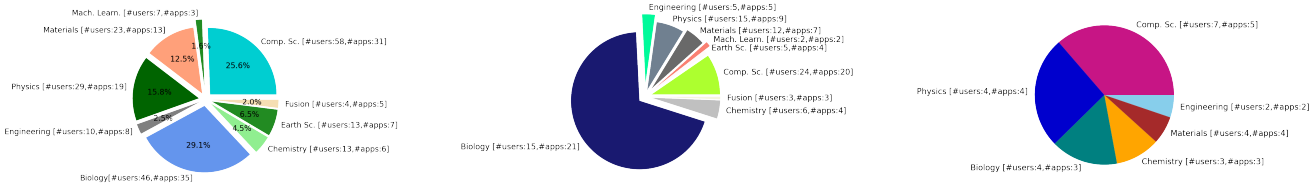
Observation: More than 78% of ML jobs run on less than 45 nodes with Biology and Computer Science having the maximum proportion of jobs. Biology has more than two-third share of the medium-scale ML jobs while Computer Science and Physics have the maximum share among the 77 ML jobs which run on flagship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

2) Jobs using GPFS and BB: Figure 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.

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 III-A3. Table V shows how the different



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

Figure 5: Classification of Small-Scale $(18,269)$, Medium-Scale $(5,043)$, 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).

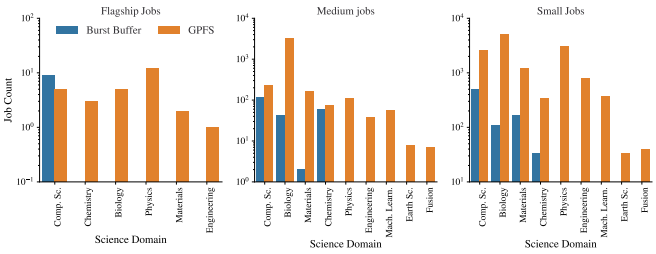


Figure 6: Number of ML jobs which either access at least one file on BB or all files exclusively on GPFS classified on the basis of the scale of jobs and science domains.

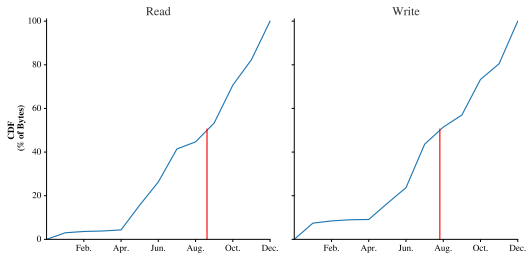


Figure 7: Cumulative distribution function (CDF) plots for read bytes and write bytes for one year.

types of jobs use GPFS and BB based on the scale of the ML jobs.

	GPFS			Burst Buffer		
Job Size	**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

Table V: 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.

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.

C. Temporal Trend of ML I/O Jobs

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 Figure 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.

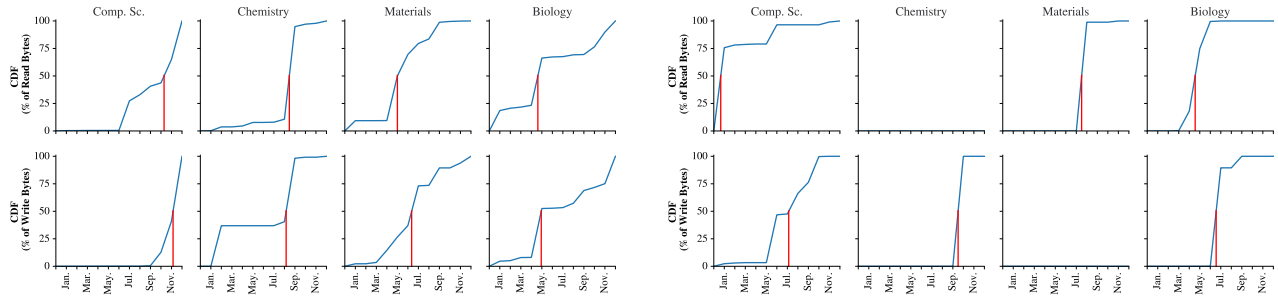
2) ML I/O Behavior Trend of Different Science Domains: Figures 8a and 8b show the temporal trend of the percentage of bytes read and written by ML jobs on GPFS and BB over a period of one year classified by the four science domains that use BB.

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. Figure 8b follows the same trend as Figure 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.

D. Usage of Burst Buffer by ML I/O Jobs

As seen in Section III-A5, 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



(a) GPFS. **(b)** Burst Buffer.

Figure 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.

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.

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.

Figures 9a and 9b 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: Figure 9a 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: Figure 9b shows that the trend of common files across all the scales of jobs is dominated by Computer Science shown in Figure 9a. 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 [18], [19] should be developed which can transparently make use of BB without making changes to legacy code bases.

2) I/O Activity of Files Persisted from BB to GPFS: Figure 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.

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.

E. 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.

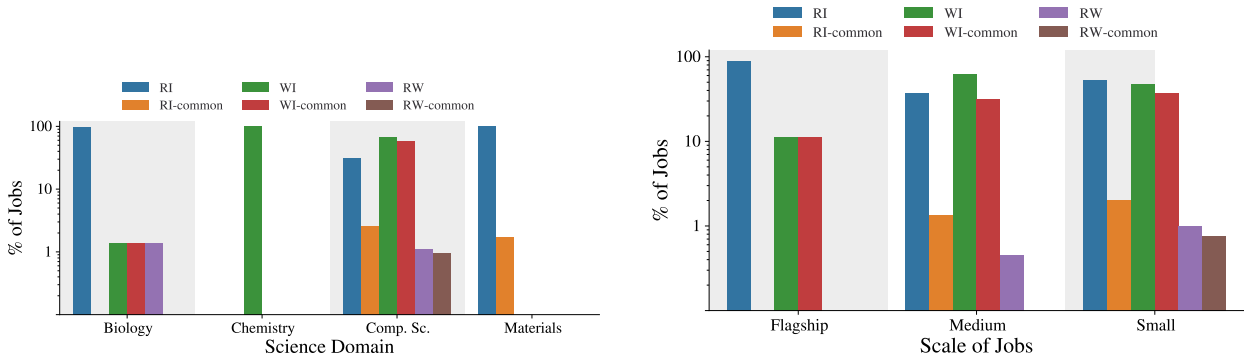
1) Read vs Write Performance: Table VI compares the mean, median and standard deviation of read and write performance by files in the ML jobs on GPFS and BB.

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

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



(a) Classified by Science Domains.

(b) Classified by Scale of Jobs.

Figure 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).

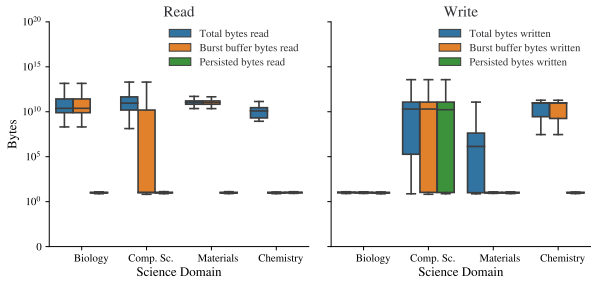


Figure 10: Bytes persisted from burst buffer to GPFS classified based on reads and writes.

BB from a single node on Summit – 2.1 GBps for writing and 5.5 GBps for reading [20].

2) I/O Performance by Different Science Domains: Table VII shows 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, when compared to other science 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 the advantages of BB.

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)

Table VII: 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.

IV. Discussion

The analysis of I/O behavior of 23,389 ML jobs 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.

A. 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.

B. 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. Therefore I/O optimization techniques must be developed that can help use the burst buffer transparently without modifying the application code.
- This study also provides guidance on the capacity of future HPC storage systems which will be dominated by ML workloads.

V. 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. 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 applications from only 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. In future, we will use this study to build a tool that can characterize ML workloads from darshan without the need of ML keywords.

Acknowledgment

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, and S. L. Scott, "Reliability-aware approach: An incremental checkpoint/restart model in hpc environments," in *CCGRID*, pp. 783–788, IEEE, 2008.

[2] F. Pan, Y. Yue, J. Xiong, and D. Hao, "I/o characterization of big data workloads in data centers," in *Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, pp. 85–97, Springer, 2014.

[3] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, "Understanding and improving computational science storage access through continuous characterization," *ACM TOS*, vol. 7, no. 3, pp. 1–26, 2011.

[4] B. K. Pasquale and G. C. Polyzos, "Dynamic i/o characterization of i/o intensive scientific applications," in *SC*, pp. 660–669, IEEE, 1994.

[5] S. Narayan and J. A. Chandy, "I/o characterization on a parallel file system," in *SPECTS*, pp. 133–140, IEEE, 2010.

[6] F. Chowdhury, Y. Zhu, T. Heer, S. Paredes, A. Moody, R. Goldstone, K. Mohror, and W. Yu, "I/o characterization and performance evaluation of beegfs for deep learning," in *ICPP*, pp. 1–10, 2019.

[7] S. W. Chien, A. Podobas, I. B. Peng, and S. Markidis, "tf-darshan: Understanding fine-grained i/o performance in machine learning workloads," in *IEEE CLUSTER*, pp. 359–370, IEEE, 2020.

[8] G. K. Lockwood, S. Snyder, S. Byna, P. Carns, and N. J. Wright, "Understanding data motion in the modern hpc data center," in *IEEE/ACM PDSW*, pp. 74–83, IEEE, 2019.

[9] A. K. Paul, O. Faaland, A. Moody, E. Gonsiorowski, K. Mohror, and A. R. Butt, "Understanding hpc application i/o behavior using system level statistics," in *HiPC*, pp. 202–211, IEEE, 2020.

[10] "Darshan - HPC I/O Characterization Tool." https://www.mcs.anl.gov/research/projects/darshan/. Accessed: July 17 2021.

[11] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, "Modular hpc i/o characterization with darshan," in *ESPT*, pp. 9–17, IEEE, 2016.

[12] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 characterization of petascale i/o workloads," in *Cluster Workshop*, pp. 1–10, IEEE, 2009.

[13] T. Wang, K. Mohror, A. Moody, K. Sato, and W. Yu, "An ephemeral burst-buffer file system for scientific applications," in *SC*, pp. 807–818, IEEE, 2016.

[14] "IBM Spectrum Scale (GPFS)." ibm.com/products/spectrum-scale. Accessed: July 17 2021.

[15] "Summit." https://www.olcf.ornl.gov/summit/. Accessed: July 17 2021.

[16] "Top 500 - November 2020." https://www.top500.org/lists/top500/2020/11/. Accessed: July 17 2021.

[17] "Summit Scheduling Policy." https://docs.olcf.ornl.gov/systems/summit_user_guide.html#scheduling-policy. Accessed: July 16 2021.

[18] A. Kougkas, H. Devarajan, and X.-H. Sun, "Hermes: a heterogeneous-aware multi-tiered distributed i/o buffering system," in *HPDC*, pp. 219–230, 2018.

[19] A. Kougkas, H. Devarajan, and X.-H. Sun, "I/o acceleration via multi-tiered data buffering and prefetching," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 92–120, 2020.

[20] "Burst Buffer on Summit." https://docs.olcf.ornl.gov/systems/summit_user_guide.html#burst-buffer. Accessed: July 16 2021.