GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

HPS

Seminar Report

Machine Learning performance and behaviour of HPC storage systems

Sheila Navarro Carrasco

MatrNr: 12613836

Supervisor: Patrick Höhn

Georg-August-Universität Göttingen
Institute of Computer Science

March 30, 2024

Abstract

High-performance computing (HPC) plays a main role in modern scientific research, spanning various fields and facilitating advancements in areas such as drug discovery and artificial intelligence (AI) development. However, the emergence of machine learning (ML) workloads presents challenges to traditional HPC tasks, particularly in terms of managing input/output (I/O) operations effectively.

While Parallel File Systems (PFS) have been effective for traditional HPC tasks, they may struggle to meet the I/O requirements of ML workloads, necessitating the exploration of alternative solutions such as Burst Buffers (BB).

This report explores the benefits of incorporating burst buffers as a layer in HPC systems to address the challenges posed by ML workloads, offering improved performance and efficiency in managing I/O tasks. Through an in-depth analysis of storage systems in HPC environments, including PFS and burst buffers, alongside a performance evaluation, the effectiveness of burst buffers in enhancing the performance of ML workloads is demonstrated, providing valuable insights for optimizing computational resources in diverse scientific applications.

Statement on the usage of ChatGPT and similar tools in the context of examinations

In this work I have used ChatGPT or a similar AI-system as follows:

- ☐ Not at all
- ☐ In brainstorming
- ☐ In the creation of the outline
- ☐ To create individual passages, altogether to the extent of 0% of the whole text
- ☑ For proofreading
- ☐ Other, namely: -

I assure that I have stated all uses in full.
Missing or incorrect information will be considered as an attempt to cheat.

Contents

List of Tables

List of Figures

List of Abbreviations

ML Machine Learning

HPC High-Performance Computing

BB Burst Buffer

PFS Paralell File System

GPFS IBM Spectrum Scale

1 Introduction

Modern scientific research heavily relies on High Performance Computing (HPC) to perform computations and simulations, across fields. It addresses massive data analysis in science, finance, and more. HPC boosts advancements in drug discovery and AI development and also gives businesses a competitive edge and helps humans address global challenges like climate change. However, the emergence of Machine Learning (ML) workloads introduces new challenges to traditional HPC tasks, necessitating a closer examination of classical storage systems such as Parallel File Systems (PFS), used in supercomputers like Summit.

While exploring possibilities to enhance the performance of these storage systems for ML tasks, one may consider the option of Burst Buffers (BB). BB serves as an in-system storage layer positioned between non-persistent memory and persistent storage, designed to handle a burst of read or write I/O at a high rate, making it an excellent candidate for improving I/O performance for ML workloads.

Consequently, this report explores the realm of HPC workloads, highlighting the differences between traditional tasks and those involving ML. It delves into how these differences can pose challenges depending on the storage system used, emphasizing the crucial role played by storage systems, with Parallel File Systems being a well-known storage system in supercomputers like Summit, and Burst Buffer emerging as a potential solution for the I/O performance issues that may arise in PFS when performing ML tasks.

In HPC, distinguishing between workloads and ML workloads reflects how computational practices are evolving. Traditional tasks involve computations such as large-scale simulations, numerical analyses and modelling while ML workloads focus on training models for activities like image recognition and predictive analysis.

One key challenge lies in understanding the I/O behaviours of traditional versus ML workloads. Traditional tasks typically follow checkpoint/restart-based I/O patterns while ML workloads introduce frequent I/O operations that pose unique storage system challenges.

Various solutions exist to address these challenges with PFS being a choice for managing data in HPC environments. Though effective for tasks PFS may face difficulties in handling the I/O requirements of ML workloads prompting a search for alternative options such, as burst buffers. This report delves into the advantages of incorporating burst buffers as a layer, in HPC systems to tackle the issues presented by ML workloads. Burst buffers provide a solution, for handling regular I/O tasks ultimately boosting performance and efficiency.

For this purpose, this report proceeds with an in-depth analysis of storage systems in HPC environments, focusing on PFS and BB. This analysis encompasses their architecture, functionality, and performance characteristics. Subsequently, a performance analysis section evaluates the effectiveness of BB compared to PFS in enhancing the performance of ML workloads, drawing insights from real-world data and empirical studies. Finally, the report concludes with key findings and recommendations for leveraging burst buffers effectively in HPC environments to meet the growing computational demands of diverse scientific applications. [Dar]

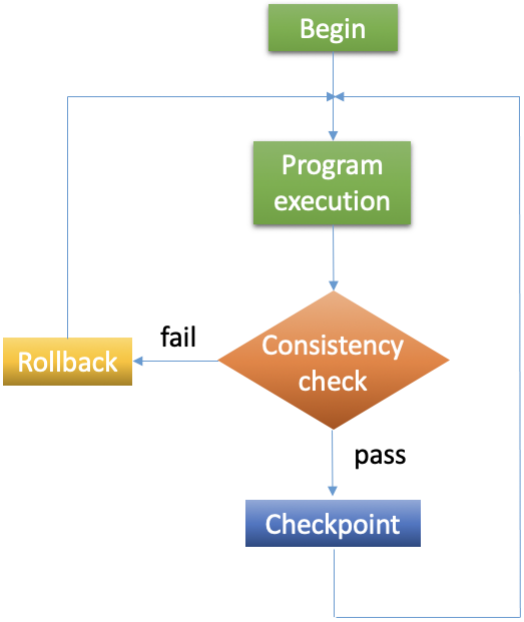

Figure 1: Checkpoint/restart-based HPC I/O behaviour logic. [Source: [Res23]]

2 Workloads in HPC

Regarding workloads in HPC, traditional workloads and machine learning workloads can be differentiated, as they have slight but relevant differences for the topic that attaches to this report.

Starting with the traditional workloads in HPC, they typically involve computationally intensive tasks that require high-performance computing resources. Some examples of these tasks are the following:

- Large-scale simulations: This can involve simulating complex physical, chemical, or biological processes, such as weather forecasting, climate modelling, or drug discovery. These simulations typically involve solving large sets of equations that require significant computational power.
- Numerical analysis: This involves solving complex mathematical problems that are too difficult or time-consuming to solve analytically. Examples include financial modelling, risk analysis, and engineering design optimization.
- Modeling: This refers to creating mathematical models of real-world systems or processes to study their behaviour. Examples include designing aircraft wings, optimizing traffic flow, and understanding the behaviour of galaxies.

It's important to note that these categories of tasks are not mutually exclusive, and many traditional HPC workloads involve elements of all three.

Regarding the I/O behaviour, these workflows follow a traditional checkpoint/restart-based HPC behaviour. The logic behind this pattern is to create checkpoint files periodically during the execution of the HPC application, after passing the consistency check.

These files contain a snapshot of the state of the application at the specific moment of the creation of the checkpoint. With these files, if the consistency check of the application fails, it is possible to restart from the most recent checkpoint file and avoid having to execute the whole application from the beginning again. This behaviour can be observed in Figure 1. This pattern allows the minimisation of the amount of work that needs to be redone if an application fails. Still, the downside is that it can be expensive and inefficient under certain circumstances or workflows, to store these periodic checkpoint files. Some examples of circumstances where this pattern could become expensive and inefficient are when there are too frequent checkpoints, which can lead to an excessive I/O overhead due to the constant writing in the storage, slowing down the overall execution. Another possibility is when it is a huge data set, typical from Machine Learning (ML) workflows. Under these circumstances the checkpoint files can become very large, leading to significant storage requirements and slowing down writing and reading checkpoints during restarts.

Another characteristic is the typical use of parallel file systems, and naturally, all the other features typical from HPC systems such as parallelism, high memory requirements, large scale, distributed computing, etc.

Regarding machine learning workloads in HPC, slight differences can be observed from the traditional ones. The first difference is the type of task that they solve. Machine learning workloads focus on training and deploying ML, which pursue different goals like solving problems of image recognition, predictive analysis, natural language processing, etc. These tasks are often data-driven meaning that the success of the model hinges on the quality and analysis of large amounts of data [Syd23].

In a data-driven approach, decisions and actions are based on insights extracted from data, rather than intuition or gut feeling. This requires collecting, storing, processing, and analyzing vast amounts of information to identify patterns, trends, and relationships that can inform model development and optimization [Syd23]. The huge amounts of data and speed needed for processing these kinds of tasks pushed to the utilization of GPUs or combinations of GPUs and CPUs. Regarding I/O behaviour, it can be observed that the ML workloads present different behaviours in this regard depending on the domain in which they are being developed. Nevertheless, a common characteristic in the I/O behaviour is the presence of a large number of small writes and reads, which is a key point for the subject that concerns this report.

3 Storage Systems

Storage systems in HPC systems play a crucial role as they are essential for delivering optimal performance. Adequate storage management reduces data access latency, optimizes storage utilization, and increases data availability. Meanwhile, inadequate storage can lead to bottlenecks, delays, or, in the worst cases, job failures.

Multiple types of storage systems exist in HPC systems. Among them are Paralell File System (PFS), object storage systems, block storage, and intermediate storage layers. Parallel file systems such as GPFS, Lustre, and BeeGFS are designed to manage and distribute large datasets across multiple nodes in a cluster. They enable efficient concurrent access to data, facilitating parallel data processing for HPC applications. However, PFS can experience bottlenecks which lead to a reduced I/O performance if they face the problem of having a large number of files with small sizes (the "*small file issue*").

In the case of object storage systems, examples include Amazon S3 and Ceph. These systems offer highly scalable and durable storage for large-scale unstructured data.

Block storage devices, such as SSDs and NVMe drives, provide high-performance I/O for critical data and temporary storage needs. They are frequently employed as scratch disks for HPC applications.

Another storage option is Hadoop [Apa], which is a framework created by Apache that is designed for large-scale data processing and analysis. It has its own distributed file system, HDFS, and can be implemented in HPC systems. However, this distributed file system present the same problem as PFS: the "*small file issue*". However, Zongwei Zhu et al. propose a solution for this problem PHDFS [Zhu+20]. This solution solves the "*small file issue*" by introducing a file aggregation method called Pile, which merges a group of small files based on their correlation.

Finally, an intermediate layer, one popular option being the burst buffer, plays a crucial role in enhancing the performance of HPC systems. It acts as a high-performance intermediate layer positioned between the compute nodes and the main storage system. By caching frequently accessed data on the burst buffer, it can significantly reduce I/O latency and improve the overall performance of HPC applications.

This report focuses on two of the previously presented storage systems, PFS and BB. Although PHDFS could also be a suitable storage system approach for ML workloads, as it addresses the "small file issue" introduced by ML in HPC, it has been decided for this report to only compare the two previously mentioned storage systems due to their different purposes in terms of I/O performance when talking about ML workloads. In the one hand is PFS, an efficient storage system used in well known supercomputers such as Summit [Ene18], but it may experience reduced performance when running ML jobs due to its handling of small read and write tasks. When exploring different solutions for this problem an excellent candidate could be BB, BB serves as an in-system storage layer positioned between non-persistent memory and persistent storage, designed to handle a burst of read or write I/O at a high rate. By comparing both options, this report aims to determine if a burst buffer is indeed a viable option that could be much more suitable than parallel file systems for enhancing the performance of ML workloads.

3.1 Parallel File Systems (PFS)

HPC applications generate and process massive amounts of data, necessitating efficient data distribution across multiple nodes in the cluster. PFS handle this process by fragmenting data into smaller chunks and distributing them across the nodes, as illustrated in Figure 2. When HPC applications request data, PFS intercept these requests and route them to the appropriate nodes where the requested data is stored. This routing process ensures that data is accessed from the nodes where it resides, minimizing unnecessary data movement and improving I/O performance. During these data requests, parallel data transfer is ensured by utilizing parallel I/O techniques to enable multiple processes or threads to read or write data simultaneously. This parallelization significantly reduces I/O latency and improves overall throughput.

As data is accessed and modified, PFS update their internal data placement maps to reflect the changes in data distribution across the cluster. This update process helps maintain data locality and optimize data movement for improved performance, considering factors like load balancing and access patterns. [Usm+23]

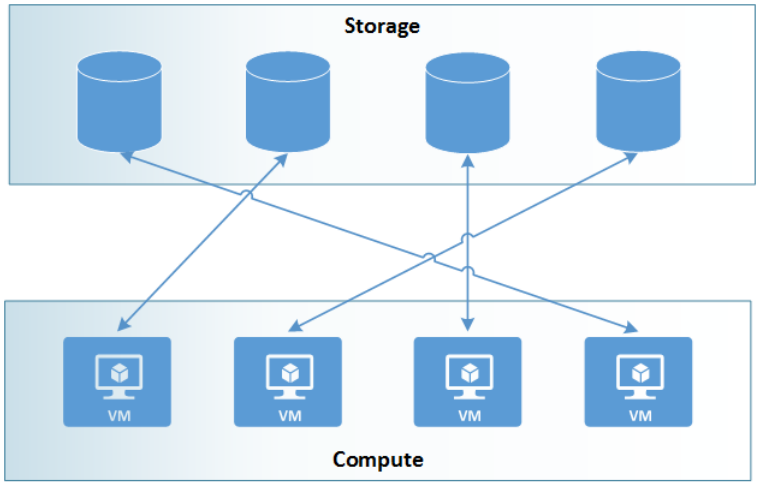

Figure 2: Parallel file system architecture in HPC. [Source: [Ed18]]

Other important characteristics of these kinds of storage systems are related to fault tolerance and other optimizations, such as data movement and communication between nodes. They implement robust fault tolerance mechanisms to ensure data availability even in the event of node failures. These mechanisms consist of storing the same data in more than one node (redundancy). Looking at Figure 2, it can be observed that piece 1 of data is stored in the first cluster, but from that first piece, the first half is stored in the first cluster and could also be at the end of the third one, for example. These data replication and redundancy strategies guarantee that data can be accessed and recovered even if individual nodes (the first node in our example) experience hardware or software issues.

Regarding optimization, PFS employ various techniques to optimize data movement and communication between nodes, minimizing data congestion and improving overall system performance. This includes techniques such as network caching, TCP segmentation offload (TSO), and RDMA (Remote Direct Memory Access). Network caching is a technique that stores frequently accessed data in a cache located at the network interface card (NIC). This can reduce the need to read data from slower storage media and improve the performance of ML applications. TCP segmentation offload is a technique that offloads the task of TCP segmentation and reassembly from the operating system to the NIC. This can improve the performance of ML applications by reducing the amount of CPU overhead associated with data transfers. Finally, RDMA is a technique that allows applications to directly access memory on another node over the network. This can significantly reduce the I/O latency of ML applications, as it eliminates the need for data to be copied between the network buffers and the application's memory. [Dub19; IBM21]

3.2 Burst Buffer (BB)

Burst buffer is an intermediate layer, located between the processing nodes and a storage system (such as a parallel file system in HPC), as shown in Figure 3. It is specifically designed to handle bursts of I/O traffic, such as those that occur during scientific simulations or machine learning training. It acts as a temporary staging area for data that needs to be accessed quickly, offloading it from slower storage like hard disk drives (HDDs) or network-attached storage (NAS) devices. Concerning this paper, more specifically, burst

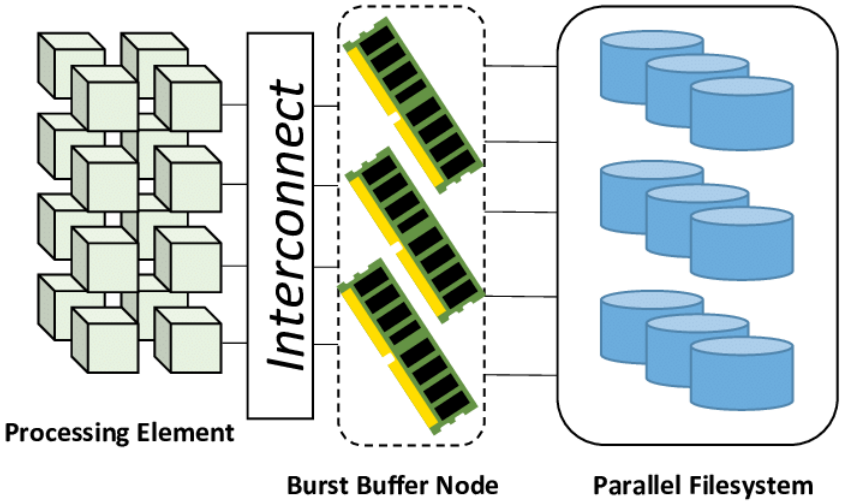

Figure 3: Architecture of a burst buffer system. [Source: [Wan+15]]

buffer acts as temporary staging area between the processing area and the parallel file system, taking as reference of HPC system the super computer Summit [Ene18; KPW22] By storing frequently accessed data in the burst buffer, it can significantly improve the performance of applications that rely on high I/O rates.

Burst buffers typically incorporate solid-state drives (SSDs) or high-speed memory (RAM) as the underlying storage medium. These devices offer much faster I/O performance compared to traditional HDDs, allowing them to handle bursts of data requests without bottlenecks. The burst buffer acts as a cache, storing copies of frequently accessed data from the slower storage tiers. When an application requests this data, the burst buffer can quickly retrieve it, reducing the need to access the slower storage devices.[Wan+15; Fun15]

4 Performance Analysis

By measuring the performance of a system, it can be assessed how efficiently, referring to speed, and effectively, referring to accuracy, the system accomplishes its tasks, ultimately reflecting the amount of valuable work it accomplishes. There exist countless measures to determine the performance of a system; among them, we can find the speed of a system, the latency, the efficiency, or the scalability. These measures have the following formal descriptions:

- Speed: The ability to complete tasks or computations promptly, measured in FLOPS (floating-point operations per second). This is particularly important for HPC applications that handle large datasets and require rapid processing.
- Latency: The time it takes for a system to respond to a request, measured in milliseconds. Low latency is crucial for real-time applications and those that involve tight communication between multiple components. In HPC, latency is important for minimizing delays and ensuring the smooth execution of parallel computations.
- Efficiency: The ratio of useful work performed by a system to the energy consumed.

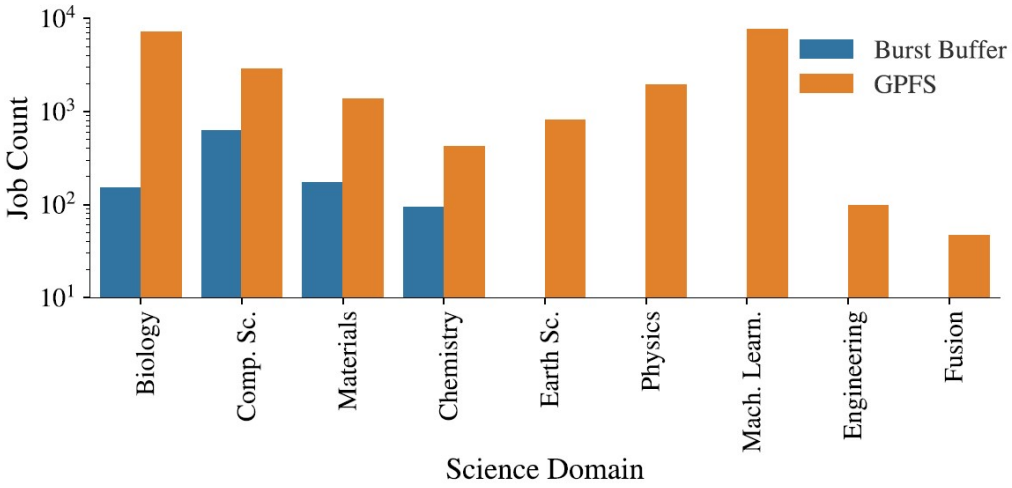

Figure 4: Number of ML jobs using burst buffer and GPFS classified by different scientific domains on Summit. [Source: [KPW22]]

This metric is becoming increasingly important in HPC, as energy efficiency is critical for reducing operating costs and environmental impact.

- Scalability: The ability of a system to handle increasing workloads or data volumes without compromising performance. Scalability is essential for HPC as applications and datasets continue to grow in size and complexity.

Getting more specific in the topic that concerns this report, we can ask how to evaluate the performance of HPC systems. For that, it can analyse, among other things, the I/O performance, the computation efficiency or the workloads. To measure the I/O performance, which is the key parameter in this report, they can be used multiple tools such as Darshan [Dar] or Recorder [Wan+20]. Karim et al. [KPW22] make use of Darshan which provides information on the executable and filenames of machine learning (ML) jobs running on Summit in 2020. Measuring this parameter allows us also to detect possible I/O bottlenecks.

Regarding computing efficiency, is closely related to the utilization of processor units such as GPUs and CPUs. Efficient memory management and data access patterns can improve computation efficiency. Moreover, the design and optimization of algorithms used in the computation process can affect efficiency.

Finally, the classification of workloads based on science domains provides insights into the different IO behaviours.

4.1 PFS vs. Burst Buffer

Karimi et al., in their article "I/O performance analysis of machine learning workloads on leadership scale supercomputer" [KPW22], conducted various comparisons to evaluate the performance of both storage options, parallel file system, and burst buffer, regarding different characteristics.

In their study, Karimi et al. [KPW22] analyzed the I/O behaviour of machine learning jobs on the Summit supercomputer, one of the world's fastest supercomputers [Ene18]. The study utilized the Darshan logs, which provide information on the executable and

filenames of machine learning (ML) jobs running on Summit in 2020. A dataset [1] of 23,389 ML jobs running on Summit in 2020 was obtained from the Darshan logs for analysis. It is noteworthy that the parallel file system on Summit is known as GPFS; henceforth, in this review, it will be referred to as the parallel file system.

Once the data were collected, the authors initially classified ML jobs into different scientific domains, including biology, computer science, materials, chemistry, earth sciences, physics, machine learning, engineering, and fusion, based on the number of ML jobs using burst buffer and those using GPFS, as depicted in Figure 4. It is evident from the figure that out of the various scientific domains available in the dataset, only four of them—biology, computer science, materials, and chemistry—utilized burst buffer as an intermediate storage system. Moreover, it is notable that the usage of GPFS significantly exceeded that of burst buffer in all four domains, with the highest number of burst buffer jobs being in computer science. This limited adoption of burst buffer by domains may suggest that its utilization could be hindered by a lack of understanding of how burst buffer could enhance I/O performance.

For the remainder of the analysis, the authors classified ML jobs into three categories: Read-Intensive (RI), Write-Intensive (WI), and Read-Write (RW).

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

The classification into these three categories was performed using the formula above (Equation 1), with the results categorized as follows:

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

This classification revealed, as shown in Table 1, that chemistry and computer science exhibited a high number of RI jobs using GPFS. This suggests that a significant portion of read-intensive jobs could potentially be transferred from GPFS to burst buffer, thereby enhancing I/O performance.

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 1: 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 scientific domains that use burst buffer. [Source: [KPW22]]

The subsequent analysis focuses on the sizes of accessed files. Tables 2 and 3 present the mean number of read and write calls used per machine learning job across different

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

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.

Figure 5: 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. $\approx$ 99 % of the read and write calls are less than 10MB calls. [Source: [KPW22]]

access sizes, categorized into bins: $<$ 1MB, 1MB–10MB, 10MB–100MB, 100MB–1GB, and $>$ 1 GB.

Analyzing the behaviour of read and write calls by file size is crucial, as larger access sizes, indicating sequential read and write operations, yield better performance when using GPFS, as previously demonstrated in this report. Conversely, small read and write operations, signified by small bin sizes, are more efficient when burst buffer is utilized, as shown in section 3.2.

It is evident that the majority of read and write calls fall into the smallest bins, under 1MB. Particularly noteworthy is that almost 99% of the read and write calls for machine learning workloads are less than 10 MB, implying that the burst buffer is an excellent candidate for improving I/O performance. This is because a significant number of small read and write requests overload the parallel file system, as previously observed. Focusing on domains that do not utilize the burst buffer at all, as depicted in Figure 4, the example of physics stands out. Physics represents a considerable opportunity for improving I/O performance, as it is the domain with the highest number of jobs among those that do not utilize the burst buffer at all and exhibits a very large number of small file accesses using only the parallel file system, which does not offer the most effective performance in this scenario.

If the jobs that use BB for ML are specifically examined, an important feature is that typically write-intensive jobs use BB to write the data temporarily and then write it persistently in GPFS. In the case of read-intensive jobs, the files are copied from GPFS to BB and then read from BB instead of from GPFS, thus improving the read performance.

When observing Figure 6, it is easily distinguishable how the domains Biology, Chemistry, and Materials (all of them using BB) have an outstanding percentage of Jobs

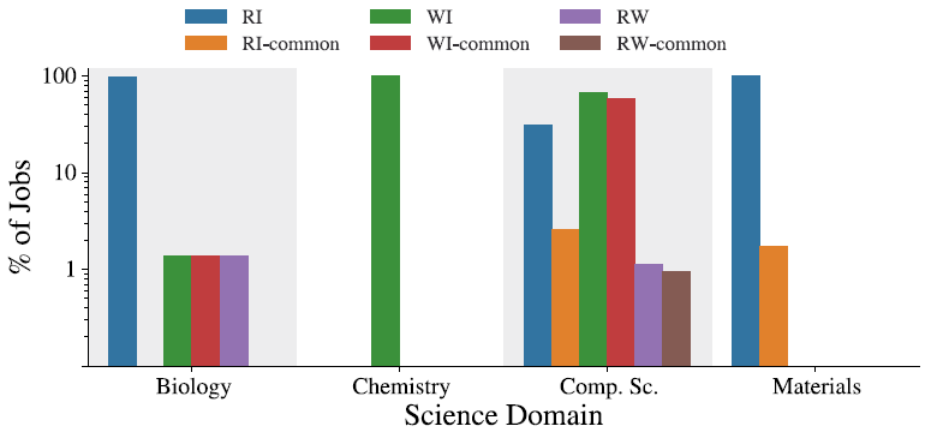

Figure 6: Percentage of read-intensive (RI), write-intensive (WI), and read–write (RW) ML jobs which have common files across GPFS and burst buffer classified by science domain. [Source: [KPW22]]

($\approx$ 100%) that do not use common files in BB and GPFS in their WI/RI jobs, which makes sub-optimal usage of BB. This is especially remarkable in Chemistry, where only write-intensive jobs are registered with no common files in BB and GPFS. The only domain following the trend of common files across all the jobs is Computer Science, which has been previously explained to improve performance.

This suggests that for optimal system-wide I/O performance, ML users of all domains need to be well educated on the benefits of BB, especially in novel I/O optimization techniques, as this behaviour suggests unawareness of users outside Computer Science about all the potential of using BB to improve I/O performance.

After examining the theoretical performance improvements that the use of BB could bring, by observing the different types of jobs submitted in different science domains and analyzing their use of BB and GPFS, let's dive into whether the burst buffer provides better real I/O performance.

In Figure 7, it is depicted how BB outperforms GPFS I/O rate in both writing and reading, in all three statistical measures: mean, median, and standard deviation, which is consistent with the theoretical peak. Upon closer examination, this trend is confirmed. In Figure 8, the performance of ML jobs classified by science domain can be observed, whether they exclusively use GPFS or have at least one file access from BB. Note that the zero values corresponding to Chemistry and Materials in the burst buffer are due to the lack of reads in BB, in the case of Chemistry, and the lack of BB writes in Materials. After these clarifications, it can be confirmed that the performance of both reads and writes in BB surpasses those in GPFS, aligning with the previous observations and conclusions. This confirms that there is significant potential for performance improvement by using BB.

Another observation that can be made in Figure 8 is that Computer Science has a better mean performance combining accesses on GPFS and BB. This confirms the suggestion that better-educated specialists in the topic of using BB efficiently would bring a significant performance improvement by having knowledge of the full potential of the burst buffer and the newest optimization techniques.

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

Figure 7: Burst Buffer and GPFS I/O performance comparison [Source: [KPW22]]

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)

Figure 8: Burst Buffer and GPFS I/O performance comparison by science domain [Source: [KPW22]]

5 Conclusion

Based on a comprehensive analysis of workloads in HPC systems, encompassing both conventional and machine learning workloads, as well as an in-depth examination of storage systems including parallel file systems and burst buffers, several key conclusions can be drawn.

Firstly, the distinction between conventional and machine learning workloads underscores the evolving computational requirements in HPC environments. Conventional workloads, characterized by large-scale simulations and modelling tasks, typically exhibit checkpoint/restart-based I/O behaviours and heavily rely on parallel file systems. In contrast, machine learning workloads present new challenges, particularly in terms of I/O behaviour characterized by a large number of small reads and writes.

Secondly, the role of storage systems, notably PFS and BB, is crucial in determining the overall performance of HPC systems. PFS, with its capability to distribute large datasets across multiple nodes and facilitate parallel data processing, remains a fundamental component of HPC environments. However, the limitations of PFS in handling small, frequent I/O operations highlight the importance of burst buffers as an intermediate layer to alleviate I/O bottlenecks and enhance overall performance, especially for machine learning workloads.

The analysis of task classifications based on scientific domains reveals varying levels of adoption and utilization of burst buffers across different disciplines. While some domains, such as computer science, demonstrate a higher inclination for effectively employing burst buffers, others, like physics, present significant opportunities for improving I/O performance through better training and awareness of burst buffer advantages.

Furthermore, performance comparisons between PFS and burst buffers validate the

theoretical benefits of burst buffers in enhancing I/O rates for both reading and writing operations. These findings underscore the potential for significant performance enhancements by leveraging burst buffers, particularly when combined with novel optimization strategies and well-informed practitioners proficient in utilizing burst buffers effectively.

In conclusion, the study highlights the evolving landscape of HPC workloads and the crucial role of storage systems, particularly burst buffers, in addressing the specific challenges posed by machine learning workloads. By efficiently leveraging burst buffers and promoting awareness and education across scientific domains, HPC systems can achieve optimal performance and meet the growing computational demands of diverse scientific applications.

References

[Apa] Apache. *Apache Hadoop*. https://hadoop.apache.org/. [Accessed 27.03.2024].

[Dar] Darshan. *Darshan HPC I/O Characterization Tool*. https://www.mcs.anl.gov/research/projects/darshan/. [Accessed 27.03.2024].

[Dub19] Viacheslav Dubeyko. "Comparative Analysis of Distributed and Parallel File Systems' Internal Techniques". In: *CoRR* abs/1904.03997 (2019). arXiv: 1904.03997. URL: http://arxiv.org/abs/1904.03997.

[Ed18] Price Ed. *Parallel File Systems for HPC Storage on Azure*. https://techcommunity.microsoft.com/t5/azure-high-performance-computing/parallel-file-systems-for-hpc-storage-on-azure/ba-p/306223. [Accessed 3.12.2023]. 2018.

[Ene18] U.S Department of Energy. *Summit Supercomputer Ranked Fastest Computer in the World*. https://www.energy.gov/articles/summit-supercomputer-ranked-fastest-computer-world. [Accessed 9.01.2024]. 2018.

[Fun15] Mark Funk. *The What And Why Of Burst Buffers*. https://www.nextplatform.com/2015/05/19/the-what-and-why-of-burst-buffers/. [Accessed 3.12.2023]. 2015.

[IBM21] IBM. *Introducing General Parallel File System*. https://www.ibm.com/docs/en/gpfs/4.1.0.4?topic=guide-introducing-general-parallel-file-system. [Accessed 13.12.2023]. 2021.

[KPW22] Karimi, Paul, and Wang. "I/O performance analysis of machine learning workloads on leadership scale supercomputer". In: *Performance Evaluation* 157-158 (2022), p. 102318. ISSN: 0166-5316. DOI: https://doi.org/10.1016/j.peva.2022.102318. URL: https://www.sciencedirect.com/science/article/pii/S0166531622000268.

[Res23] Northeastern University Research Computing. *Checkpoint/Restart Discovery Jobs*. https://rc-docs.northeastern.edu/en/2.0.0/07_best-practices/01_checkpointing.html. [Accessed 3.12.2023]. 2023.

[Syd23] Sydle. *Data-Driven: What It Is and Why It's Important*. https://www.sydle.com/blog/data-driven-what-it-is-and-why-it-s-important-606c8a4e4b136c41e0e2c334. [Accessed 20.12.2023]. 2023.

[Usm+23] Sardar Usman et al. "Data Locality in High Performance Computing, Big Data, and Converged Systems: An Analysis of the Cutting Edge and a Future System Architecture". In: *Electronics* 12.1 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12010053. URL: https://www.mdpi.com/2079-9292/12/1/53.

[Wan+15] Teng Wang et al. "Development of a Burst Buffer System for Data-Intensive Applications". In: *ArXiv* abs/1505.01765 (2015). URL: https://api.semanticscholar.org/CorpusID:18324427.

[Wan+20] Chen Wang et al. "Recorder 2.0: Efficient Parallel I/O Tracing and Analysis". In: May 2020, pp. 1–8. DOI: 10.1109/IPDPSW50202.2020.00176.

[Zhu+20] Zongwei Zhu et al. "PHDFS: Optimizing I/O performance of HDFS in deep learning cloud computing platform". In: *Journal of Systems Architecture* 109 (2020), p. 101810. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2020.101810. URL: https://www.sciencedirect.com/science/article/pii/S1383762120301028.