Sheila Navarro Carrasco

Machine Learning performance and behaviour of HPC storage systems

Newest Trends in High-Performance Data Analytics

Table of contents

Table of Contents

Introduction

Motivation

- Popularity of Machine Learning (ML)
- Differences with traditional workflows in HPC systems
- Data sizes rapidly increasing
- Bottlenecks using traditional HPC storage

Storage

- Crucial role in HPC
- Efficient and reliable data access
- HPC systems must be able to:
  - Accommodate big data volumes
  - Rapid access
  - Parallel I/O
  - Data integrity and reliability

Types of HPC storage

- **Parallel File Systems (PFS)**
  - GPFS, Lustre, BeeGFS
  - Efficient concurrent access to data
- **Object Storage Systems**
  - Ceph, Amazon S3
  - Highly scalable and durable storage
- **Block Storage**
  - SSDs, NVMe drives
  - High-performance I/O for critical data and temporary storage needs
- **Intermediate layer**
  - Burst Buffer (BB)
  - Catch frequently accessed data -> reduce I/O latency

Performance

- Determines efficiency and effectiveness of a system
- Measures:
  - Speed of the system - measured in FLOPS for HPC
  - Latency - measured in milliseconds
  - Efficiency of the system - ratio of useful work to energy consumed
  - Scalability - that it does not compromise performance

How to evaluate performance in HPC systems?

- Analyze: I/O performance, computation efficiency, workloads...

How to evaluate performance in HPC systems?

- Analyze: I/O performance, computation efficiency, workloads...
- **I/O Performance**
  - I/O monitoring tools: Darshan, Recorder...
  - Presence of I/O bottlenecks

How to evaluate performance in HPC systems?

- Analyze: I/O performance, computation efficiency, workloads...
- **I/O Performance**
  - I/O monitoring tools: Darshan, Recorder...
  - Presence of I/O bottlenecks
- **Computation efficiency**
  - Utilization of processor units -> CPUs and GPUs
  - Memory management and access patterns
  - Design and implementation of algorithms

How to evaluate performance in HPC systems?

- Analyze: I/O performance, computation efficiency, workloads...
- **I/O Performance**
  - I/O monitoring tools: Darshan, Recorder...
  - Presence of I/O bottlenecks
- **Computation efficiency**
  - Utilization of processor units -> CPUs and GPUs
  - Memory management and access patterns
  - Design and implementation of algorithms
- **Workloads**
  - Classification in science domains
  - Help characterize behaviour and performance

Table of Contents

Traditional Workloads in HPC

- Typically large-scale simulations numerical analysis and modelling tasks
- Traditional checkpoint/restart-based HPC I/O behaviour
  - Can be expensive and inefficient
- Use parallel file systems
- Other characteristics: parallelism, high memory requirements, large scale, distributed computing...

ML workloads in HPC

- Focus on training and deploying ML models
- Solving problems from image recognition, predictive analytics, NLP...
- Often data-driven
- Pushed for the utilization of GPUs or a combination of CPUs and GPUs
- Different I/O behaviours depending on the domain
- **Large number of small reads and writes**

Table of Contents

Types of HPC storage

- **Parallel File Systems (PFS)**
  - GPFS, Lustre, BeeGFS
  - Efficient concurrent access to data
- **Object Storage Systems**
  - Ceph, Amazon S3
  - Highly scalable and durable storage
- **Block Storage**
  - SSDs, NVMe drives
  - High-performance I/O for critical data and temporary storage needs
- **Intermediate layer**
  - Burst Buffer (BB)
  - Catch frequently accessed data -> reduce I/O latency

Types of HPC storage

- **Parallel File Systems (PFS)**
  - GPFS, Lustre, BeeGFS
  - Efficient concurrent access to data
- **Object Storage Systems**
  - Ceph, Amazon S3
  - Highly scalable and durable storage
- **Block Storage**
  - SSDs, NVMe drives
  - High-performance I/O for critical data and temporary storage needs
- **Intermediate layer**
  - **Burst Buffer (BB)**
  - Catch frequently accessed data -> reduce I/O latency

Paralell File Systems (PFS)

- Data distribution among nodes
- Routing process for I/O request
- Parallelization for data transfer
- Data placement updates
- Robust fault tolerance
- Communication optimization (network catching, TSO, RDMA)
- Easily scalable
- Often used in traditional workloads

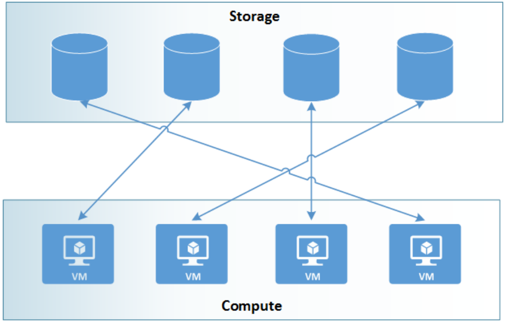


Figure: Paralell File System HPC

Ed, *Parallel File Systems for HPC Storage on Azure*

Burst Buffer (BB)

- Intermediate storage area
- Designed to handle I/O traffic bursts
- Solid-state drives (SSDs) or high-speed memory (RAM)
- Avoid I/O bottleneck-> suitable for ML

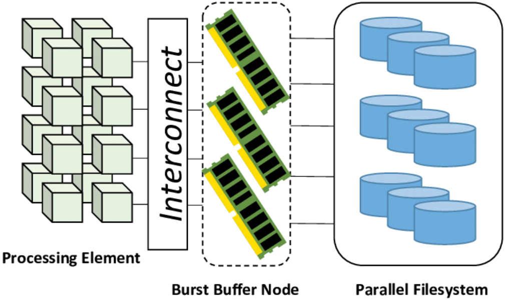


Figure: Burst Buffer Architecture

Funk, *The What And Why Of Burst Buffers*
T. Wang et al., “Development of a Burst Buffer System for Data-Intensive Applications”

Table of Contents

Classification based in science domains

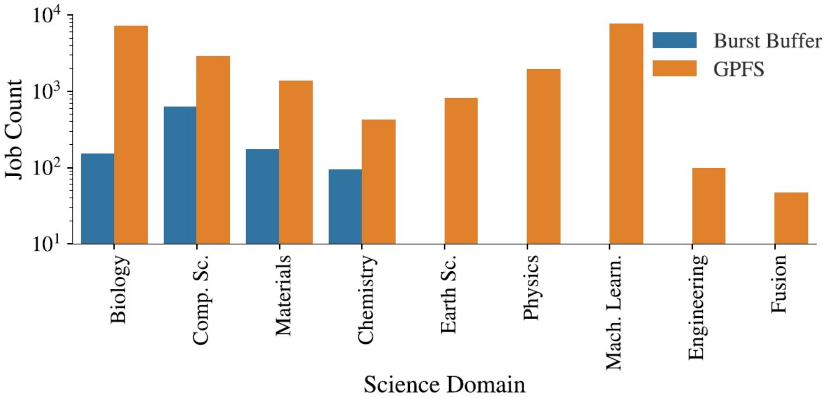


Figure: ML jobs using BB and GPFS classified in science domains

Image Source: Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Types of ML jobs

- Write Intensive (WI)
- Read Intensive (RI)
- Read-Write (RW)

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

Figure: RI vs. WI vs. RW ML jobs using BB or GPFS classified by science domain

Karimi, Paul, and Wang, “I/O performance analysis of machine learning workloads on leadership scale supercomputer”

Types of ML jobs

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

Figure: RI vs. WI vs. RW ML jobs using BB or GPFS classified by science domain

Karimi, Paul, and Wang, “I/O performance analysis of machine learning workloads on leadership scale supercomputer”

Types of ML jobs

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

Figure: RI vs. WI vs. RW ML jobs using BB or GPFS classified by science domain

Possible performance improvement:

A large percentage of read-heavy files from Computer Science and Chemistry can be migrated from GPFS to BB

Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Read and Write access sizes

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: Mean number of read and write calls per job classified by science domain

Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Read and Write access sizes

Possible performance improvement:

Almost 99% of the read and write calls are less than 10MB -> Burst Buffer

Common files in BB and GPFS

- **Write-Intensive** writes temporaly in BB and then persistent in GPFS
- **Read-intensive** copy data from GPFS to BB and read from BB

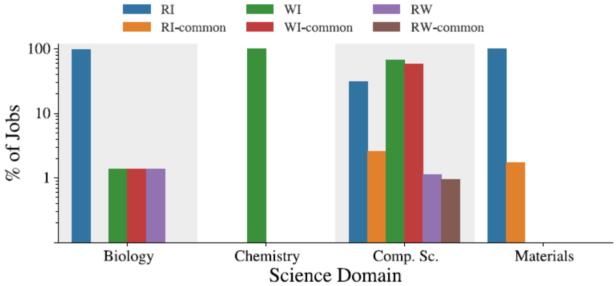


Figure: RI, WI, RW ML jobs with common files in GPFS and BB. (**NOTE:** *Common* for RI jobs means that files were copied from GPFS to BB and then read from burst BB. Equivalent for WI.)

Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Common files in BB and GPFS

Possible performance improvement:

ML users need to be well educated on the benefits of BB as well as I/O optimization techniques

Performance comparation

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: Burst Buffer and GPFS I/O performance comparation

Observation

BB outperforms GPFS in both read and write

Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Performance comparation by domain

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: Burst Buffer and GPFS I/O performance comparation by science domain

Observation

BB outperforms GPFS in both read and write -> huge scope of improvement

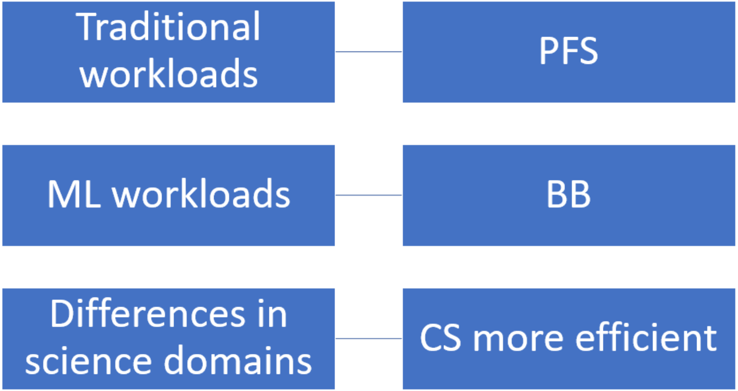
Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Table of Contents

Conclusions

- There exist different options of storage for HPC
- ML and traditional HPC workloads mainly differ in their I/O behaviour
- GPFS -> Bottlenecks which affect the performance
- BB is better suited for small file reads and writes of ML workloads
- Only few science domains use BB
- CS users makes much more efficient use of BB
- There is room for improvement of HPC performance in ML jobs

Sum up


Traditional workloads
PFS
ML workloads
BB
Differences in science domains
CS more efficient

Types of I/O jobs: Formula

- $$result = \left(\frac{ReadBytes - WriteBytes}{ReadBytes + WriteBytes} \right)$$
- $-1 \leq result \leq -0.5$: Write-Intensive (WI)
- $0.5 \leq result \leq 1$: Read-Intensive (RI)
- $-0.5 < result < 0.5$: Read–Write (RW)

Case where apparently BB is not a better option than PFS

- Nearer to the x-axis and further away from zero are WI, the ones closer to y-axis and far away from zero are RI, and the jobs in the middle are RW.
- 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

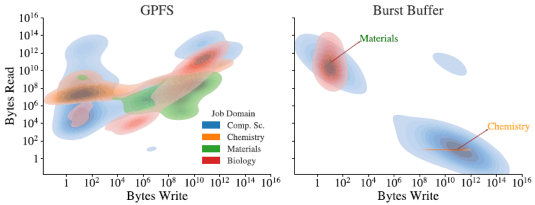


Figure: Density distribution plots of I/O activity from ML jobs using GPFS or BB by science domains.

Image Source: Karimi, Paul, and Wang, "I/O performance analysis of machine learning workloads on leadership scale supercomputer"

Traditional checkpoint/restart-based HPC I/O behaviour

- Checkpoint files are created periodically during the execution of an HPC application.
- Contain a snapshot of the application's state at a specific point in time.
- If the application fails, it can be restarted from the most recent checkpoint file
- This minimizes the amount of work that needs to be redone.

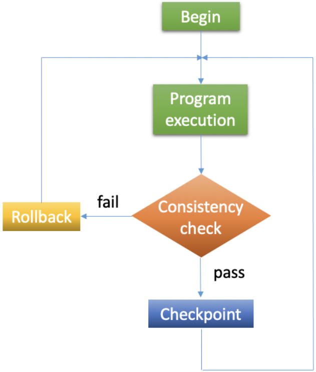


Figure: Checkpoint/restart point logic

Image Source: Research Computing, *Checkpoint/Restart Discovery Jobs*

PFS: Communication optimization

- Network Caching:
  - 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 (TSO):
  - 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.
- RDMA (Remote Direct Memory Access):
  - 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.

References

Ed, Price. *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.

Funk, Mark. *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.

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.

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

Wang, Teng et al. "Development of a Burst Buffer System for Data-Intensive Applications". In: (May 2015).