Understanding Highly Configurable Storage for Diverse Workloads

Olga Kogiou
Dept. of Computer Science
Florida State University
Tallahassee, FL, USA
ok22b@fsu.edu

Hariharan Devarajan
Lawrence Livermore
National Laboratory
Livermore, CA, USA
hariharandev1@llnl.gov

Chen Wang
Lawrence Livermore
National Laboratory
Livermore, CA, USA
wang116@llnl.gov

Weikuan Yu
Dept. of Computer Science
Florida State University
Tallahassee, FL, USA
yuw@cs.fsu.edu

Kathryn Mohror
Lawrence Livermore
National Laboratory
Livermore, CA, USA
kathryn@llnl.gov

***Abstract*—Highly configurable storage solutions such as VAST DataStore recently have emerged and are now being deployed in many High Performance Computing facilities. However, these state-of-the-art storage systems have yet to be evaluated for their abilities to serve diverse I/O patterns. Indeed, the evaluation of a new storage system can prove to be challenging, as the system needs to be tested across different platforms using various storage system configurations and workloads. To address this, we test different configurations and deployments of the VAST DataStore with diverse workloads against GPFS, Lustre, and NVMe on machines located across different sites. To simulate diverse workloads, we use the benchmarks IOR and Deep Learning I/O, and test VAST with scientific, data analytics, and AI-driven workloads. Our findings conclude that the deployment of VAST with RDMA can achieve up to 8× higher bandwidths as compared to TCP-based instances of VAST. Furthermore, VAST can viably serve applications with low I/O requirements, such as ResNet-50 trained with small datasets.**

I. Introduction

High-Performance Computing (HPC) workloads are becoming increasingly data-intensive, generating large amounts of data [1] and consisting of individual applications [2]. Traditionally, HPC workloads were dominated by scientific simulations that conduct bulk synchronous I/O [3]. However, modern HPC workloads are evolving to include data analytic applications and AI, which can have different I/O requirements from the storage system compared to scientific simulations [4]. To respond to such diverse workloads, highly configurable file systems are starting to gain attention [5]. Examples include VAST DataStore [6] and UnifyFS [7] which allows users to configure the data management policy, such as the number of dedicated I/O servers and the data placement strategy [8].

The performance of traditional parallel file systems such as Lustre and GPFS [9] has been a subject of extensive scrutiny. Firstly, the storage systems have been evaluated using diverse workloads including Deep Learning (DL) and data analytic applications [10]–[13]. Other efforts have also evaluated different deployments of these systems across different sites [14]–[16]. These three dimensions; the use of *diverse workloads*, different *storage system configurations* and *deployment methods*, can be used to allow for informed decision-making in selecting storage solutions for specific computing environments. While extensive evaluations have been conducted for other parallel file systems, not all three dimensions have been explored for VAST DataStore.

Fully understanding the performance of a new storage technology such as VAST is not a trivial task. Firstly, the anticipated performance output of VAST can be impacted from differences in the *storage system configuration*. This includes, the number of its building components such as storage servers and Solid-state drives (SSDs) and the use of different interconnects. The performance implications of *different deployments* such as different connection protocols between the compute nodes of a cluster and VAST file system need also to be considered during evaluation. Lastly, the performance of VAST with *diverse workloads* needs to be examined to investigate how the file system performs under various I/O requirements to provide meaningful insights.

To investigate the impact of storage system deployment methods we explore the performance characteristics of VAST and compare it against other storage solutions such as GPFS, Lustre and node-local NVMe on several Livermore Computing (LC) and Oak Ridge Leadership Computing Facility (OLCF) machines. We evaluate different storage system configurations and cluster deployments of VAST by performing scalability tests and single node tests with synchronization on write. We use scientific simulations, data analytic and Machine Learning (ML) applications that are simulated with the IOR benchmark [17]. Preliminary results of this work are presented in [18]. In addition, we test VAST using two real-time DL applications; ResNet50 [19] and Cosmoflow [20] with the Deep Learning I/O benchmark [21] and conduct an in-depth analysis of the results using the tracing tool DFTracer [22], [23].

In summary, we have made the following contributions.

- We evaluate different storage system configurations and deployments of VAST to test the impact of its main building components and TCP and RDMA protocols on LC and OLCF clusters.
- We perform an extensive set of experiments using diverse workloads and analyze the I/O time results for real-time DL applications to identify bottlenecks.
- We conclude that an RDMA-based instance of VAST can provide up to 8× higher bandwidths as compared to TCP-based instances of VAST. Moreover, VAST can viably serve DL applications with low I/O requirements

to reduce the contention of parallel file systems such as GPFS which is more commonly used in the LC clusters.

II. Related Work

Many storage solutions have been evaluated with the use of benchmarks that can simulate diverse I/O access patterns. Scalability and single-node tests have been conducted for BurstFS [24], GekkoFS [25], IME [26] and Ceph [11], [13], [27] using IOR and MDTest [28]. However, none of the aforementioned works have tested the file system of interest with all groups of diverse workloads to identify the applications best suited to the storage system's performance, as in our work.

Several studies have tested different storage system configurations of Lustre [29]–[32]. The file system has been configured with SSDs [32] at the Intel CRT-DC and over Quadrics [33]. Other works have evaluated different configurations of GPFS such as GPFS-SNC [34] and GPFS-WAN deployed at Indiana University [35].

The evaluation of different deployment methods of the same file system is an aspect that is not popularly explored among the HPC community. Collectively, studies have tested storage systems across different sites. For example, evaluation works of DAOS [36] have been conducted [14]–[16] in the German Computing Center, Sandia National Laboratory and NEXTGenIO research HPC system [37] where DAOS was compared against traditional storage solutions such as OrangeFS [38], BeeGFS and Lustre.

Despite all the aforementioned studies, few are the works focused on the evaluation of file systems across all dimensions; using diverse workloads, different storage system configurations and different deployment methods. Notable is the work by Chowdhury et. al [10] where different deployments of BeeGFS as a shared and node-local storage and different storage system configurations by tuning the stripe size in write bandwidth have been evaluated with IOR, MDTest and real-time DL applications. In 2021, Glenn K. Lockwood, Alberto Chiusole and Nicholas J. Wright [39] published an evaluation work on VAST where the main focus was the effects of the two different types of SSDs that VAST uses.

III. Background

In this section, we briefly discuss the architecture of VAST. We then discuss the diversity of I/O requirements of different workloads.

A. VAST DataStore

VAST DataStore is an all-flash storage system. The main building blocks of VAST are its two types of servers, CNodes and DNodes. The CNodes are able to access the data, metadata and system state directly in a shared-everything model, where everything is stored on NVMe SSDs contained in enclosures called the DBoxes.

1) The VAST Servers (CNodes): During the boot time, the CNodes stage all the Storage Class Memory (SCM) SSDs and hyperscale Quad-level cell (QLC) flash SSDs in the cluster via NVMe-over-Fabrics (NVMe-oF) or other interconnects. Therefore, all the I/O requests fall on the CNodes. When a read request arrives, the CNode accesses the file's metadata from the SCM SSD. Each node can complete such read requests independently and does not need to consult or communicate with other CNodes.

2) Stateless Containers: The VAST system state is firstly written into multiple SSDs, then acknowledged and finally committed and thus the containers (which host the CNodes) are considered stateless.

3) High Availability Enclosure (DBoxes): Each DBox contains two or more DNodes tasked with directing NVMe-oF requests from their fabric ports to the enclosure's SSDs via a network of PCIe switch chips.

4) Storage Class Memory: VAST uses SCM SSDs as an intermediate fast layer between the storage backbone and a global metadata store. In fact, SCMs are known for their ultra-low latency (in the range of 100 nanoseconds to 30 microseconds for random access) and therefore promise optimization in write requests.

5) Hyperscale Quad-level Cell Flash: Hyperscale QLC flashes are used as the backbone of the storage where data are eventually persisted.

B. Diverse workloads

Scientific workloads have previously been used for the evaluation of storage systems [11], [40], [41]. Examples of such workloads include CM1 [42], an atmospheric-simulation model that generates more than 750 files each of 16 MB in size, and HACC-I/O [43], an I/O kernel for hardware/hybrid accelerated cosmology which emaulates checkpoint/restart on simulation data.

On the other hand, modern high-performance data-analytics tasks often access data using embarrassingly parallel algorithms [44]. A common I/O pattern in these workloads involves iteratively traversing data to merge to a solution. Examples of these workloads are BD-CATS [45], which operates on a shared HDF5 file using MPI-IO, and KMeans [46], which reads points from files with divisions based on algorithmic tasks.

Lastly, ML workloads conduct I/O by consuming different kinds of large datasets. Some examples of these applications are linear regression [47] and decision tree [48] models. DL workloads typically consume tabular data, perform data shuffling and splitting, and leverage Stochastic Gradient Descent (SGD) [49] for model training. Examples include Cosmoflow [20], which consumes HDF5 files with a size of 32 MB each, and Cosmic Tagger with UNet (Cosmic Tagger) [50] which also uses HDF5 files with h5py APIs and stripes the file in memory.

All these different types of workloads can have different I/O requirements and investigating into the performance ad-

vantages of arising storage solutions such as VAST can lead to a better mapping between specific workloads and file systems.

IV. Test Methodology

In this section, we describe the system configuration and software tools we use for the experimental evaluation.

A. Hardware

For our experiments we use supercomputers Lassen [51], Ruby [52], and Quartz [53], all located at Lawrence Livermore National Laboratory. We have also tested VAST on Wombat [54] machine located at Oak Ridge National Laboratory. The specifications of each cluster are described in Table I.

TABLE I: Clusters used for experiments

Name	Nodes	Node characteristics				Network
		CPU	GPU	RAM	Arch	
Lassen [51]	795	44	4	256	IBM Power9	IB EDR
Ruby [52]	1,512	56	0	192	Intel Xeon	Omni-Path
Quartz [53]	3,018	36	0	128	Intel Xeon	Omni-Path
Wombat [54]	8	48	2	512	ARM Fujitsu A64fx	IB EDR

B. File system Deployments and Configurations

The compute nodes of Lassen are connected with the VAST CNodes over a single gateway node with a 2×100Gb Ethernet over a single TCP link, as shown in Figure 1a. On Ruby there is a 1×40Gb Ethernet link on eight gateway nodes. On Quartz, there is a 2×1Gb Ethernet link on 32 gateway nodes. The instance of VAST on the LC clusters consists of ten DNodes and 16 CNodes that are exposed over the Network File System (NFS). VAST has five DBoxes and each DBox contains two DNodes with 22 QLC and 6 SCM SSDs. The CBoxes and DBoxes are connected with the use of EDR InfiniBand with NVMe-oF protocol.

GPFS on Lassen consists of 16 PowerPC64 storage nodes with 1.4PB Network Shared Disk (NSD) each using GPFS RAID interconnected with InfiniBand. A high-level architecture of GPFS on Lassen can be found on Figure 1b. Lustre consists of 16 Metadata Servers (MDSs) with six Serial Attached SCSI (SAS) SSD Zettabyte File System (ZFS) mirrors, 36 Object Storage Server (OSSs) with 80 SAS Hard-Disk Drive (HDD) raidz2 groups, leveraging an EDR InfiniBand SAN with 100Gb OmniPath.

VAST on Wombat is deployed using RDMA with `nconnect=16` and `multipathing` enabled. VAST on Wombat consists of eight DNodes, which are BlueField DPUs, and eight CNodes, which are exposed over the NFS. There are 11 SSDs and four NVRAMs hosted by a pair of DPUs of VAST. The CBoxes and DBoxes are connected via 2×50Gbps Ethernet links through NVMe-oF and RDMA over Converged Ethernet (RoCE).

The NVMe used to compare VAST with on Wombat is the local storage physically attached to the compute nodes of Wombat. A mount point to each node is available for interacting with the NVMe of each node. The NVMe consists of three Samsung 970 PRO SSDs on each compute node, connected via PCIe Gen3x4.

C. Software tools and diverse workloads

In our experiments, we used I/O benchmarks, namely, the IOR and DLIO benchmarks. This section describes the workloads and the benchmark configurations for our tests.

1) Interleaved-Or-Random (IOR): We use IOR-4.1.0 to measure the I/O bandwidth. Sequential write requests were used to simulate *scientific applications*, sequential reads were used for *data analytic applications* and random read requests for *ML algorithms* [10]. We chose to evaluate the file systems using POSIX API since it is a lower-level pattern that is used for initially profiling storage subsystems and N-N (file-per-process) as it is a common pattern seen in most applications [40], instead of N-1 (shared-file) as the contention, file locking and metadata overhead it introduces can make the isolation of the storage system behavior challenging. In an attempt to minimize client read caches, a different client read the requests than the one who generated the writes.

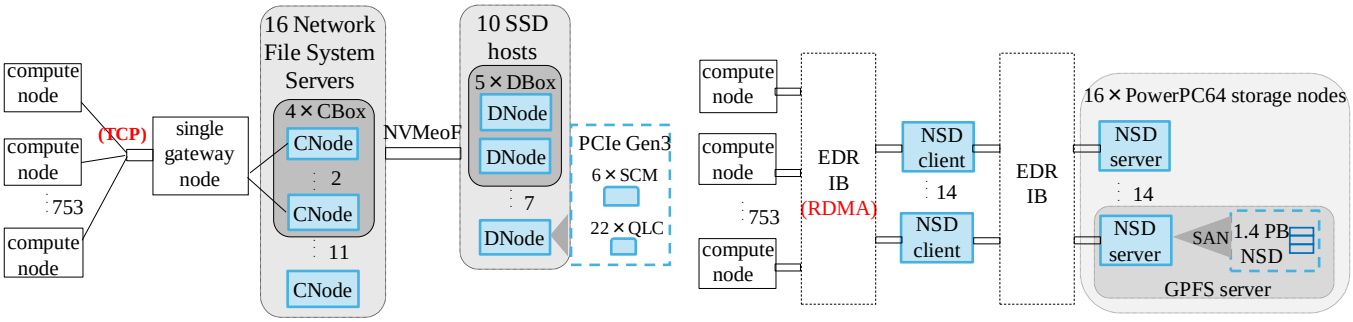
2) Deep Learning I/O (DLIO): We use DLIO-1.1.0 that aims to emulate the I/O behavior of DL applications. With the use of the benchmark, we tested VAST against GPFS on Lassen with the DL applications ResNet50 with weak scaling and Cosmoflow with strong scaling due to the larger size of this application's dataset. For our I/O time result analysis we used the DLIO Profiler tool.

Our experiments are not performed in an isolated environment and all file systems, including VAST, are shared (typically GPFS and Lustre are more commonly used and they might experience contention effects). To test performance consistency in the shared environment we repeated our tests 10 times.

V. I/O Characterization of VAST

This section presents the evaluation results after testing different configurations and deployment methods of VAST against GPFS, Lustre and node-local NVMe storage with diverse workloads simulated with IOR. We first test VAST's scalability on full nodes (44 processes per node on Lassen and 48 processes per node on Wombat) by scaling up to 128 nodes on Lassen and all eight nodes of Wombat. This scale allows us to draw conclusions without wasting resources and node-hours as the I/O pattern stays the same. The scalability results presented using node-local NVMe were conducted by copying the data from the writing node to reading nodes using round-robin (to avoid caching) as NVMe SSDs cannot access data from a remote node directly. Operating System cache write-back is allowed on this test to replicate a realistic user scenario. To prevent process-local caching in reads, we use task reordering, where we offset tasks issued by the processes by number of processes per node. In addition, we have kept the total size of I/O large enough, at approximately 120 GB per node, in order to outgrow the block size of GPFS's and Lustre's cache to avoid misleading results. To achieve this, we set the segment number to 3,000 and the block and transfer size to 1 MB.

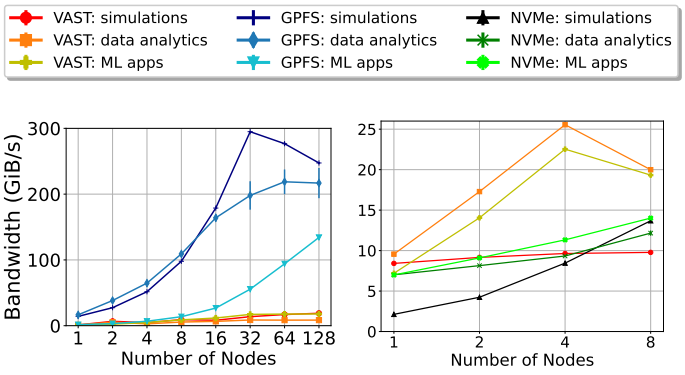
We also conduct a single client test, where we scale the number of processes to 32 on Lassen, Quartz, Ruby



(a) High-level Architecture of VAST on Lassen.

(b) High-level Architecture of GPFS on Lassen.

Fig. 1: The differences between VAST and GPFS on Lassen.



(a) Lassen

(b) Wombat

Fig. 2: Scalability test results for scientific simulations, data analytics and ML applications.

and Wombat and utilized synchronization on writes. Write synchronization or `fsync` flushes the file to the storage server's device after each write. Our purpose is to test the raw performance of the file systems. Because I/O bottlenecks were easily detectable using single node tests for Quartz and Ruby, we did not use these machines for scalability tests.

A. Performance evaluation of scientific applications

A common practice when testing a file system is to first test the access pattern of sequential accesses, as many real-world scientific applications follow this data access pattern [42], [55]. The scalability results on VAST against GPFS and NVMe for scientific simulations are provided in Figure 2.

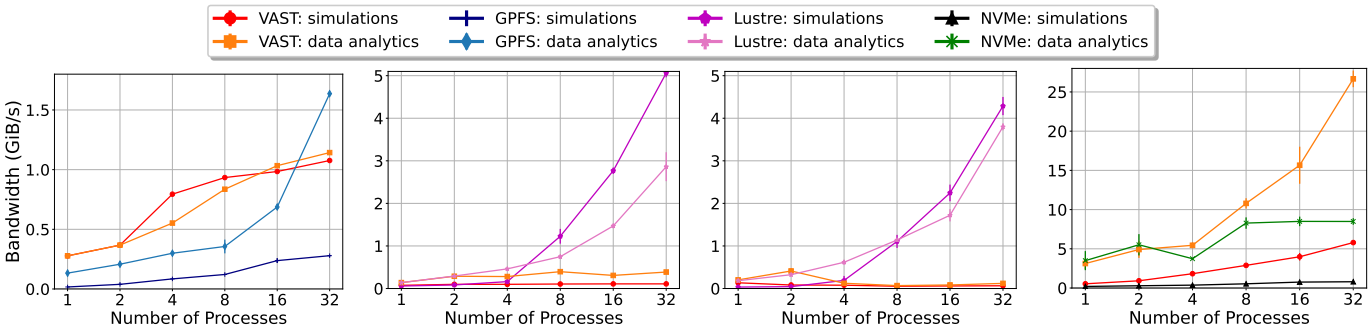
As seen from Figure 2a, VAST does not scale linearly on Lassen as opposed to GPFS. The bandwidth for VAST is similar to the maximum available bandwidth on the network. This observation leads us to believe that there is a network bottleneck relevant to VAST's deployment on Lassen, where the CNodes (that are NFS servers) communicate with the compute nodes over a single TCP link. In contrast, VAST on Wombat (Figure 2b) performs better than on Lassen, leveraging the RDMA link with the compute nodes. However, its scalability is still limited, probably due to the 2×50Gb Ethernet links used to connect its CNodes and DNodes. As this is a difficult hypothesis to prove we plan on deploying a custom VAST configuration on cloud-like resources, such as Chameleon Cloud to test this.

The single node results on VAST against GPFS, Lustre and NVMe for scientific simulations are depicted in Figure 3. Lustre behaves similarly on Quartz and Ruby (Figure 3b-3c) with almost linear increase in bandwidth, while VAST on Quartz and Ruby (Figure 3b-3c) shows weak performance. The main reason for this is the network bottleneck created by these clusters' small Ethernet links with the gateway nodes described in section IV. On the contrary, the results of VAST on Lassen shown in Figure 3a are promising, due to the better deployment of VAST on Lassen as compared to Ruby and Quartz. VAST performs almost 5× better for a single node on Wombat than the NVMe (Figure 3d). That is because VAST can leverage its configuration with RDMA, `multipath` and `nconnect` to its advantage, resolving the network challenge with the compute node communication. This time, its maximum performance is reached at **5.8 GB/s** when using 32 processes per node where its saturation point lies.

B. Performance evaluation of data analytic applications

As most data analytic applications require high read availability, we have evaluated VAST's ability to handle sequential read requests. The scalability results can be found in Figure 2.

As shown in Figure 2a, GPFS on Lassen demonstrates high bandwidths for sequential read accesses, reaching its maximum for 32 nodes where it saturates. Sequential read bandwidths on VAST are higher than sequential writes, as during write operations the CNodes are burdened with similarity-based data arrangement and compression which does not happen during reads [6]. However, because most of these requests are served by GPFS' caches, VAST continues to perform poorly in comparison. Here, it is worth mentioning that the instance of GPFS is much larger on Lassen (total capacity of 24 PB) compared to VAST (total capacity of 5.2 PB), with multiple levels of caches and several disks that make it an ideal HPC file system. Moreover, the network



(a) Lassen (b) Quartz (c) Ruby (d) Wombat

Fig. 3: Single node test with `fsync` results for scientific simulations and data analytics.

bottleneck due to the deployment of VAST on Lassen with the use of NFS and a single TCP link for the connection of VAST with Lassen's compute nodes is a limiting factor for its performance. VAST's performance appears improved on Wombat (Figure 2b) leveraging RDMA, `multipathing` and `nconnect` which allow the use of multiple network links between client and server and parallel data transfers despite the use of NFS. Interestingly, VAST is able to outperform the NVMe in smaller scales. However, its scalability is still limited in this design, as mentioned before.

The single node results on VAST against GPFS, Lustre and NVMe for data analytics are depicted in Figure 3. Lustre's bandwidth for data analytics in Figure 3b-3c seems to grow exponentially as the request size grows and the number of processes increases. Similarly to the scientific workload results, VAST performs better on Lassen (Figure 3a), as compared to Ruby and Quartz for data analytics. This results from the faster Ethernet link (2×100Gb of Ethernet on the single gateway node) used to connect VAST with the compute nodes on Lassen compared to the rest of the clusters (section IV:A). VAST is able to outperform GPFS leveraging the SSDs in its design. The performance of VAST seems even more improved on Wombat (Figure 3b), where for 32 processes per node it serves approximately **26.6 GB/s** outperforming the NVMe by leveraging the DNode caches as well as the RDMA deployment. It is not uncommon for file systems to outperform node-local solutions as many times they parallelize requests better [56].

C. Performance evaluation of machine learning applications

The most common applications that use random file access are out-of-core sorting and data processing in database-like files where the offset indicates the location of each entry. Since such types of applications would not require `fsync` for consistency, we have decided to benchmark by only performing scalability tests. We present the scalability results in Figure 2.

The performance of VAST on Lassen for ML applications (Figure 2a) is similar to that of data analytics. The abrupt stagnation of VAST after 32 nodes further highlights the network bottleneck while GPFS increases exponentially without saturating all 128 nodes. VAST on Wombat (Figure 2b) is able to outperform the NVMe on small scales, following the same trend as that of data analytics. Despite having a smaller number of CNodes and DNodes hosting fewer SSDs, VAST is able to achieve acceptable bandwidths. The use of the DNode caches and the use of RDMA in its deployment allow VAST to reach a global maximum bandwidth of **22.5GB/s** with just four nodes. However, VAST saturates on eight nodes, likely due to its configuration with eight CNodes on Wombat.

Interestingly, GPFS demonstrates significantly lower bandwidths for random reads compared to sequential reads (Figure 2a). This is expected as its caching mechanisms are optimized for sequential reads where the spatial locality can be exploited, but get thrashed more in random access patterns. However, the same does not occur for VAST, where read bandwidths remain almost the same for the two access patterns, as seen in Figure 2b.

VI. Performance Evaluation of Deep Learning Applications

To test VAST in a real-time environment against other storage solutions, we decided to evaluate VAST against GPFS with the use of the DLIO benchmark using two deep learning applications, ResNet-50 and Cosmoflow, on the Lassen supercomputer. Testing on Lassen allows us to have a large enough node scale to test the full potential of the file systems. The two applications we chose have key differences and satisfy different user cases.

A. Deep learning application runtime analysis and profiling

DL applications train large datasets in epochs, processing data in batches to avoid biased learning and to optimize system utilization. Data loaders, such as TensorFlow [57], create a task graph to fetch these batches from storage to memory before the training begins. This process ensures that the required batch is readily available before the computation begins and happens with the use of system calls that are translated into "events".

However, these read events can introduce overhead to the application runtime as they need to stall the GPU from proceeding with the computations to read the next batch. Therefore, AI workloads allow the input pipeline to execute

asynchronously in conjunction with the compute to minimize GPU stalls. As a result, part of the I/O overhead can be hidden as it overlaps with the computation time of the application and the runtime can be sub-categorized into three main groups: the I/O that does not overlap with the computation (called the *non-overlapping I/O*), the I/O time which happens in parallel with the computation (called the *overlapping I/O*) and the time which is dedicated only in computing. In our experiments, 97% of the overall application runtime consists of only GPU computation, however, I/O is still not completely hidden. as data need to be fetched before the computation begins. For this work, we have decided to exclude the compute time from our results in order to focus on the I/O aspect.

To further understand the time analysis results, we divide the throughput presented for the two applications into two categories: the *application throughput* and the *system throughput*. The application only has the ability to perceive as I/O, the time that the application actually stalls its computation and, therefore, depends only on the non-overlapping I/O. In contrast, the system throughput depends on the total I/O time as the system resources are occupied to read the input.

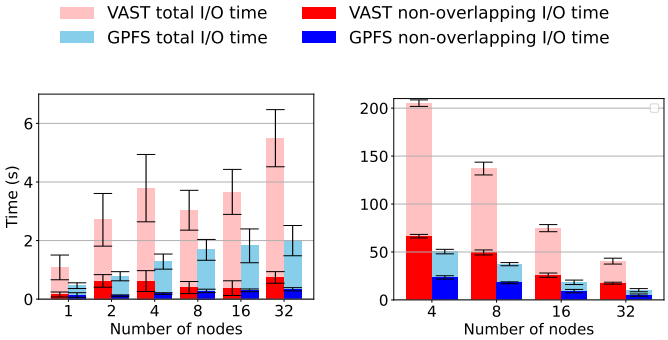
We used the DLIO benchmark to simulate the two applications, ResNet50 and Cosmoflow. First, we generate a real dataset that we can customize and scale depending on the number of nodes used for testing, and we then train it while using a different set of nodes to read the dataset than the one that generated it to avoid Operating System write-back caching. The profiling was performed using the DFTracer, which captures system-level calls and stores them into log trace files which consist of "read" and "compute" events.

In the next subsections, we present the applications used and the time analysis results where we focus on the I/O time.

B. ResNet-50

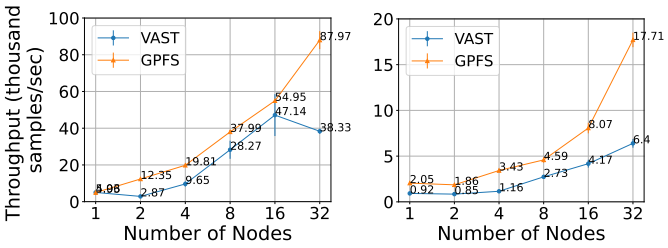
ResNet50 is an application commonly used for JPEG image classification and is a supervised machine learning model consisting of 50 deep convolution neural network layers. For this work, we have used the one batch-sized PyTorch [50] version of ResNet-50 created by DLIO where the whole dataset consists of 1024 JPEG samples, each of size 150 KB. We performed a weak scaling test by increasing the number of nodes to 32 and trained the dataset for one full epoch. This scale allows us to identify curve trends while being large enough to train the model in just a few minutes.

We present the I/O time analysis results in Figure 4a. As demonstrated in our previous results with IOR in section V, the I/O performance of VAST on Lassen is throttled by its deployment, which reduces the overall I/O throughput achieved by the DL workload and results in increased I/O time. In addition, due to the smaller size of this dataset, we expect that the requests are majorly served by GPFS's caches, which have proved from our initial testing in section V to perform better than VAST's SSDs. However, despite the fact that VAST spends more time on I/O than GPFS, most of it overlaps with the computation. From figures 5a-5b, we can observe that although the system throughput looks very different for the



(a) ResNet-50 (b) Cosmoflow

Fig. 4: I/O time analysis.



(a) Application Throughput (b) System Throughput

Fig. 5: ResNet-50 Throughput.

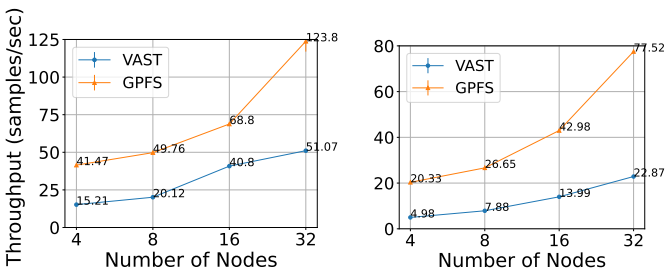
two file systems, the throughput that the application perceives is only slightly higher for GPFS compared to that of VAST, with the difference becoming more apparent only for larger scales.

C. Cosmoflow

Cosmoflow is a TensorFlow [57] application used for studying the features in the distribution of dark matter. For this work, we use a version of Cosmoflow, which consists of 1024 TFRecord samples, and the transfer size for the I/O requests remains constant at 256 KB throughout the training process. To run the application, we use four full epochs and batch size one. There are eight threads per process for computation and four threads for the I/O data pipeline. The smaller number of I/O threads in Cosmoflow can provide a contrasting scenario to ResNet50 and demonstarte the file system capabilities under limited resources.

We present the I/O time analysis results in Figure 4b. Cosmoflow trains a larger dataset for four epochs and therefore spends minutes in I/O, as compared to ResNet-50 which spends seconds. Consequently, the non-overlapping I/O in Cosmoflow is dramatically increased for VAST, as many dataset samples do not allow the I/O to be hidden from the compute time.

Unsurprisingly, GPFS serves Cosmoflow better than VAST (Figure 6a). A reason behind this big difference in throughput for the two file systems could be the use of only four threads in this version of Cosmoflow as opposed to ResNet-50 where



(a) Application Throughput (b) System Throughput

Fig. 6: Cosmoflow Throughput.

eight threads were responsible for the I/O pipeline. The system throughput of VAST (Figure 6b) is also lower than that of GPFS which confirms our initial findings in section V.

VII. Conclusion and Main Takeaways

With this work, we aim to overcome the challenges of testing new storage systems and provide a useful guide for the HPC community to follow when benchmarking emerging storage solutions. For our future work, we wish to help Livermore Computing administrators improve the interconnection used with VAST and perform further testing.

Our final takeaways from this work are summarized as follows:

- **Takeaway for application users:** VAST can viably serve workloads with low I/O requirements to reduce the contention effect of GPFS, which all users on the Livermore Computing clusters more commonly use. An example of such an application is ResNet-50, which has a small dataset and runs for a limited number of epochs.
- **Takeaway for system administrator:** An RDMA-based deployment of VAST, with `multipathing` and `nconnect` is expected to provide up to 8x higher bandwidths per node as compared to TCP-based deployments of VAST when using the Network File System. The anticipated bandwidths for RDMA-deployed VAST are approximately 8 GB/s per node for write and read, while the TCP-deployed VAST can serve around 1 GB/s per node and lacks scalability.
- **Takeaway for I/O researchers:** The use of SCM/QLC SSDs on the VAST DNodes re-configures the data according to access patterns and allows for comparable results between sequential and random accesses as compared to the HDD-configuration of GPFS on Lassen (Figure 2b). GPFS can serve approximately 14.5 GB/s per node for sequential reads but experiences a 90% performance drop, providing 1.4 GB/s for random reads. In contrast, RDMA-based VAST stays consistent, with its expected bandwidths being 9 GB/s and 7 GB/s for sequential reads and random reads, respectively.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 23-ERD-053. LLNL-CONF-862956. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under the DOE Early Career Research Program. This work is supported in part by the National Science Foundation award 1763547, and has used the NoleLand facility that is funded by the U.S. National Science Foundation grant CNS-1822737. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] M. Mercier, D. Glesser, Y. Georgiou, and O. Richard, "Big data and hpc collocation: Using hpc idle resources for big data analytics," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 347–352.

[2] H. Luu, B. Behzad, R. Aydt, and M. Winslett, "A multi-level approach for understanding i/o activity in hpc applications," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2013, pp. 1–5.

[3] S. Kuo, M. Winslett, Y. Cho, J. Lee, and Y. Chen, "Efficient input and output for scientific simulations," in *Proceedings of the sixth workshop on I/O in parallel and distributed systems*, 1999, pp. 33–44.

[4] D. Milojicic, P. Faraboschi, N. Dube, and D. Roweth, "Future of hpc: Diversifying heterogeneity," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 276–281.

[5] B. Behzad, H. V. T. Luu, J. Huchette, S. Byna, Prabhat, R. Aydt, Q. Koziol, and M. Snir, "Taming parallel i/o complexity with auto-tuning," in *Proceedings of the international conference on high performance computing, networking, storage and analysis*, 2013, pp. 1–12.

[6] VASTData, "The vast datastore." [Online]. Available: https://vastdata.com/whitepaper/#ThePromiseofAI-EnabledDiscovery

[7] A. Moody, D. Sikich, N. Bass, M. J. Brim, C. Stanavige, H. Sim, J. Moore, T. Hutter, S. Boehm, K. Mohror *et al.*, "Unifyfs: A distributed burst buffer file system-0.1. 0," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2017.

[8] M. J. Brim, A. T. Moody, S.-H. Lim, R. Miller, S. Boehm, C. Stanavige, K. M. Mohror, and S. Oral, "Unifyfs: A user-level shared file system for unified access to distributed local storage," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023, pp. 290–300.

[9] F. Schmuck and R. Haskin, "{GPFS}: A {Shared-Disk} file system for large computing clusters," in *Conference on file and storage technologies (FAST 02)*, 2002.

[10] 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 *Proceedings of the 48th International Conference on Parallel Processing*, 2019, pp. 1–10.

[11] S. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06)*, 2006, pp. 307–320.

[12] W. Schenck, S. El Sayed, M. Foszczynski, W. Homberg, and D. Pleiter, "Early evaluation of the "infinite memory engine" burst buffer solution," in *High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P^ 3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19–23, 2016, Revised Selected Papers 31*. Springer, 2016, pp. 604–615.

[13] M. Oh, J. Eom, J. Yoon, J. Y. Yun, S. Kim, and H. Y. Yeom, "Performance optimization for all flash scale-out storage," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2016, pp. 316–325.

[14] L. Logan, J. Lofstead, X.-H. Sun, and A. Kougkas, "An evaluation of daos for simulation and deep learning hpc workloads," in *Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems*, 2023, pp. 9–16.

[15] T. Krabbe and R. Felgenhauer, "Evaluation of daos for existing scientific software."

[16] N. Manubens, S. D. Smart, T. Quintino, and A. Jackson, "Performance comparison of daos and lustre for object data storage approaches," in *2022 IEEE/ACM International Parallel Data Systems Workshop (PDSW)*. IEEE, 2022, pp. 7–12.

[17] Lawrence Livermore National Laboratory (LLNL), "IOR." [Online]. Available: https://github.com/hpc/ior

[18] O. Kogiou, H. Devarajan, C. Wang, W. Yu, and K. Mohror, "I/o characterization and performance evaluation of large-scale storage architectures for heterogeneous workloads," in *2023 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops)*. IEEE, 2023, pp. 44–45.

[19] B. Koonce and B. Koonce, "Resnet 50," *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pp. 63–72, 2021.

[20] A. Mathuriya, D. Bard, P. Mendygral, L. Meadows, J. Arnemann, L. Shao, S. He, T. Kärnä, D. Moise, S. J. Pennycook *et al.*, "Cosmoflow: Using deep learning to learn the universe at scale," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 819–829.

[21] H. Devarajan, H. Zheng, A. Kougkas, X.-H. Sun, and V. Vishwanath, "Dlio: A data-centric benchmark for scientific deep learning applications," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 81–91.

[22] H. Devarajan, L. Pottier, K. Velusamy, H. Zheng, I. Yildirim, O. Kogiou, W. Yu, A. Kougkas, X.-H. Sun, J. S. Yeom, and K. Mohror, "DFTracer: An Analysis-Friendly Data Flow Tracer for AI-Driven Workflows," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. Atlanta, GA: IEEE, Jun. 2024.

[23] Hariharan Devarajan, "DFTracer." [Online]. Available: https://github.com/hariharan-devarajan/dlio-profiler/tree/dev

[24] T. Wang, W. Yu, K. Sato, A. Moody, and K. Mohror, "Burstfs: A distributed burst buffer file system for scientific applications," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2016.

[25] M.-A. Vef, N. Moti, T. Süß, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann, "Gekkofs-a temporary distributed file system for hpc applications," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 319–324.

[26] DDN, "Ime burst buffers documentation." [Online]. Available: https://www.ddn.com/products/ime-flash-native-data-cache/

[27] W. Schenck, S. El Sayed, M. Foszczynski, W. Homberg, and D. Pleiter, "Early evaluation of the "infinite memory engine" burst buffer solution," in *High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P^ 3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19–23, 2016, Revised Selected Papers 31*. Springer, 2016, pp. 604–615.

[28] Lawrence Livermore National Laboratory (LLNL), "MDTest." [Online]. Available: https://github.com/LLNL/mdtest

[29] T. Zhao, V. March, S. Dong, and S. See, "Evaluation of a performance model of lustre file system," in *2010 Fifth Annual ChinaGrid Conference*. IEEE, 2010, pp. 191–196.

[30] J. Piernas, J. Nieplocha, and E. J. Felix, "Evaluation of active storage strategies for the lustre parallel file system," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2007, pp. 1–10.

[31] Y. Wang, Y. Lu, C. Qiu, P. Gao, and J. Wang, "Performance evaluation of a infiniband-based lustre parallel file system," *Procedia Environmental Sciences*, vol. 11, pp. 316–321, 2011.

[32] M. Hebenstreit, "Performance evaluation of intel® ssd-based lustre* cluster file systems at the intel® crt-dc," Tech. rep., Intel, Tech. Rep., 2014.

[33] W. Yu, R. Noronha, S. Liang, and D. K. Panda, "Benefits of high speed interconnects to cluster file systems: a case study with lustre," in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2006, pp. 8–pp.

[34] R. Jain, P. Sarkar, and D. Subhraveti, "Gpfs-snc: An enterprise cluster file system for big data," *IBM Journal of Research and Development*, vol. 57, no. 3/4, pp. 5–1, 2013.

[35] K.-Y. Cheng, H.-S. Chen, and C.-Y. Liu, "Performance evaluation of gfarm and gpfs-wan in data grid environment," 2010.

[36] M. Hennecke, "Daos: A scale-out high performance storage stack for storage class memory," *Supercomputing frontiers*, p. 40, 2020.

[37] "Next generation i/o for the exascale." [Online]. Available: http://www.nextgenio.eu/about-nextgenio

[38] M. M. D. Bonnie, B. Ligon, M. Marshall, W. Ligon, N. Mills, E. Q. S. Sampson, S. Yang, and B. Wilson, "Orangefs: Advancing pvfs," in *USENIX Conference on File and Storage Technologies (FAST)*, 2011.

[39] G. K. Lockwood, A. Chiusole, and N. J. Wright, "New challenges of benchmarking all-flash storage for hpc," in *2021 IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW)*. IEEE, 2021, pp. 1–8.

[40] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate, "Plfs: a checkpoint filesystem for parallel applications," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–12.

[41] A. Ovsyannikov, M. Romanus, B. Van Straalen, G. H. Weber, and D. Trebotich, "Scientific workflows at datawarp-speed: Accelerated data-intensive science using nersc's burst buffer," in *2016 1st Joint International Workshop on Parallel Data Storage and data Intensive Scalable Computing Systems (PDSW-DISCS)*. IEEE, 2016, pp. 1–6.

[42] H. Rahman, B. Pinty, and M. M. Verstraete, "Coupled surface-atmosphere reflectance (csar) model: 2. semiempirical surface model usable with noaa advanced very high resolution radiometer data," *Journal of Geophysical Research: Atmospheres*, vol. 98, no. D11, pp. 20 791–20 801, 1993.

[43] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel *et al.*, "The universe at extreme scale: multi-petaflop sky simulation on the bg/q," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[44] J. Kwon, N. L. Kim, M. Kang, and J. WonKim, "Design and prototyping of container-enabled cluster for high performance data analytics," in *2019 International Conference on Information Networking (ICOIN)*. IEEE, 2019, pp. 436–438.

[45] M. M. A. Patwary, S. Byna, N. R. Satish, N. Sundaram, Z. Lukić, V. Roytershteyn, M. J. Anderson, Y. Yao, Prabhat, and P. Dubey, "Bd-cats: big data clustering at trillion particle scale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015, pp. 1–12.

[46] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: a review," in *Computational Science and Its Applications–ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30–July 3, 2014, Proceedings, Part V 14*. Springer, 2014, pp. 707–720.

[47] P. Xenopoulos, J. Daniel, M. Matheson, and S. Sukumar, "Big data analytics on hpc architectures: Performance and cost," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2286–2295.

[48] Y.-Y. Song and L. Ying, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[49] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[51] HPC@LLNL, "Lassen." [Online]. Available: https://hpc.llnl.gov/hardware/compute-platforms/lassen

[52] ——, "Ruby." [Online]. Available: https://hpc.llnl.gov/hardware/compute-platforms/ruby

[53] ——, "Quartz." [Online]. Available: https://hpc.llnl.gov/hardware/compute-platforms/quartz

[54] OLCF, "Wombat." [Online]. Available: https://www.olcf.ornl.gov/tag/wombat/

[55] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *2008 IEEE international symposium on parallel and distributed processing*. IEEE, 2008, pp. 1–9.

[56] A. Moody, G. Bronevetsky, K. Mohror, and B. R. De Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2010, pp. 1–11.

[57] "TensorFlow," https://www.tensorflow.org/.