An Evaluation of DAOS for Simulation and Deep Learning HPC Workloads

Luke Logan
Illinois Institute of Technology
Chicago, Illinois, United States
llogan@hawk.iit.edu

Jay Lofstead
Sandia National Labs
Albequerque, New Mexico, United States
gflofst@sandia.gov

Xian-He Sun
Illinois Institute of Technology
Chicago, Illinois, United States
sun@iit.edu

Anthony Kougkas
Illinois Institute of Technology
Chicago, Illinois, United States
akougkas@iit.edu

Abstract

Traditionally, distributed storage systems have relied upon the interfaces provided by OS kernels to interact with storage hardware. However, much research has shown that OSes impose serious overheads on every I/O operation, especially on high-performance storage and networking hardware (e.g., PMEM and 200GBe). Thus, distributed storage stacks are being re-designed to take advantage of this modern hardware by utilizing new hardware interfaces which bypass the kernel entirely. However, the impact of these optimizations have not been well-studied for real HPC workloads on real hardware. In this work, we provide a comprehensive evaluation of DAOS: a state-of-the-art distributed storage system which re-architects the storage stack from scratch for modern hardware. We compare DAOS against traditional storage stacks and demonstrate that by utilizing optimal interfaces to hardware, performance improvements of up to 6x can be observed in real scientific applications.

CCS Concepts: • **Information systems** → **Distributed storage**; **Flash memory**; **Phase change memory**; • **Computing methodologies** → **Parallel computing methodologies**; **Machine learning**; **Distributed computing methodologies**.

ACM Reference Format:
Luke Logan, Jay Lofstead, Xian-He Sun, and Anthony Kougkas. 2023. An Evaluation of DAOS for Simulation and Deep Learning HPC Workloads. In *3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS '23), May 8, 2023, Rome, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3578353.3589542

1 Introduction

HPC systems have traditionally suffered from an I/O bottleneck due to the gap between storage and CPU performance. To lower this gap, HPC centers are adopting high-performance storage (e.g., PMEM) and networking (e.g., 200GBe) hardware. This hardware is typically organized in a hierarchy [8, 10, 18–20], where data is initially buffered in a high-performance tier and then flushed to a high-capacity tier. This improvement in hardware provides substantial benefits to overall application performance. However, while performance benefits can be observed, many storage systems were designed under the assumption that I/O is significantly slower than compute. This assumption is no longer true, with many works noting significant performance loss caused by software overheads [16, 27, 34] and context switching [17, 23] under various workloads. To fully take advantage of the low latency and high bandwidth provided by this hardware, the design and implementation of storage systems are being thoroughly reconsidered.

One source of performance degradation is caused by the storage and networking stacks provided by OS kernels. The Linux I/O stack, for example, was designed primarily for reliability and correctness, while maintaining acceptable performance for general users. However, many works have demonstrated that the Linux I/O stack causes significant performance loss due to its lengthy I/O path [15, 26, 31], interrupts [17, 23], and context switches [17, 23]. To improve this, new kernel-bypass storage stacks [26, 32, 33] have been developed to harness the characteristics of modern hardware. One such work is the SPDK [32], which is an NVMe driver which utilizes userspace function calls and polling instead of system calls and interrupts to interact with hardware. Many single-node storage systems (e.g., filesystems [15, 31] and key-value stores [26, 27]) have been developed on top of these kernel-bypass stacks, which have demonstrated significant improvements to I/O performance in terms of latency. However, the evaluations of these works are only at the single-node.

Due to the significant performance impacts of kernel-bypass and hardware optimization on single-node storage stacks, distributed storage stacks are also emerging which are designed specifically for modern hardware. Many well-established storage systems are being patched to better support modern hardware. One such system is CephFS [5], which is a distributed filesystem which recently replaced kernel-level filesystems for storing metadata in NVMe and PMEM with kernel-bypass technologies [2]. However, since existing systems were already largely designed towards slower storage mediums, other works have re-architected the storage stack completely [3, 7, 24]. For example, the Distributed Asynchronous Object Store (DAOS) [24] by Intel is a state-of-the-art storage system which re-architects the I/O path entirely to account for the high performance of modern hardware.

While many distributed storage stacks have been proposed to take advantage of modern hardware, the impact of these different optimizations have not been well-evaluated for a variety of workloads. For the most part, evaluations of storage systems which incorporate kernel-bypass technologies are limited to single-node cases, and many of these evaluations are over emulated hardware [14, 15, 22, 31] using only synthetic workloads [13]. Overall, from the existing work, it is not clear the extent to which re-desigining storage stacks for modern hardware has affected the performance of real distributed scientific applications.

In this work, we aim to quantify the extent to which re-architecting storage stacks for modern hardware impacts the performance of real simulation and deep learning HPC workloads. To do this, we provide a comprehensive evaluation of DAOS 2.0: a state-of-the-art distributed storage stack which has been built specifically for modern hardware. We quantify the performance impacts of DAOS's diverse software stack optimizations compared to traditional kernel-based approaches over modern storage and networking hardware, including NVMe and PMEM. We demonstrate that DAOS outperforms traditional storage systems, such as OrangeFS, by as much as 15x under various synthetic and real workloads on high-performance storage hardware.

2 Background & Related Work

There has been growing interest in optimizing storage stacks to maximize the bandwidth and latency potential of storage. This effort has spanned from OS-level changes to device drivers to entire distributed storage system designs. Various works have proposed new storage stack designs, and some work has been done to evaluate the implications of these new designs at the distributed level.

2.1 Distributed Storage Stacks

While many changes have been made to the design of single-node storage stacks, the impacts of these changes must also be understood at the distributed level. Due to the success of single-node I/O stack optimizations, distributed storage stacks are also evolving to optimize for the high-performance of modern storage hardware. Some storage systems are being patched to utilize these new interfaces while making minimal changes to their overall design, whereas other systems have been developed entirely from scratch.

Traditional Storage Stacks: OrangeFS [4] and BeeGFS [9] are both traditional high-end computing storage systems. OrangeFS has shipped with the Linux kernel since 4.6 and provides a userspace FUSE plugin for performing I/O. BeeGFS comes with a custom kernel module which acts as a kernel-level filesystem. When performing I/O, OrangeFS and BeeGFS divide data into stripes and distributes them among storage servers. The locations of stripes are managed by the metadata servers, which are by default co-located with the storage servers to achieve scalability. Both these systems currently utilizes the I/O interfaces provided by default with the OS, and have not specifically been optimized to support high-performance modern hardware, outside of RDMA capabilities.

Patched Storage Stacks: CephFS [5] is a POSIX-compliant file system built on top of Ceph's distributed object store, RADOS. CephFS endeavors to provide a state-of-the-art, multi-use, highly available, and performant file store for a variety of applications, including traditional use-cases like shared home directories, HPC scratch space, and distributed workflow shared storage. CephFS has recently replaced kernel-level filesystems for storing metadata with kernel-bypass technologies [2], including the SPDK [32] and DAX [12]. The work demonstrated significant performance improvement of their kernel-bypass storage backend. The evaluation of this work was primarily over a 16-node cluster using RAM and HDD, and was not evaluated over PMEM. One synthetic experiment was conducted over NVMe.

Modular Storage Stacks: Recent efforts have been aiming to provide modularized storage stacks in order to combat software overheads caused by the OS kernel and provide workload- and hardware-specific customization. Mochi [30] provides a diverse set of building blocks, which can be composed by users to rapidly build highly customized distributed storage systems. LabStor [26] proposes an extensible platform to facilitate the rapid development of new, hardware-optimized, workload-specific storage stacks. Users can develop a wide variety of storage modules, ranging from per-node I/O schedulers to distributed storage systems, which can then be composed to form highly optimized I/O stacks. The evaluations of these works were primarily over emulated hardware and NVMe.

DAOS: The Distributed Asynchronous Object Store (DAOS) [24] is a state-of-the-art storage system developed by Intel to take full advantage of modern hardware, such as NVMe and PMEM. DAOS requires the existence of either NVMe or PMEM for acting as a persistent cache to metadata. For

NVMe, the SPDK can be used for interacting directly with NVMe while bypassing the kernel. For PMEM, DAX can be used for mapping PMEM directly into DAOS's address space, minimizing kernel overheads.

DAOS has two storage concepts: pools and containers. A DAOS pool abstracts over the storage hardware for a subset of storage nodes. A pool can contain a variety of different storage hardware. The storage in a DAOS pool is tiered based off of the type of storage (e.g., NVMe vs PMEM). There can be up to a few hundred pools. A DAOS container, which is a transactional object store, can be allocated from a DAOS pool. A DAOS container represent many different storage systems. For example, a container can be exposed as a filesystem or a key-value store. There can be up to a few hundred containers. Each container can store billions of objects (e.g., files). By default, DAOS recommends 6% of a container's allocated space be PMEM/NVMe while 94% be allocated to mass storage (e.g., HDD). This ensures that all metadata will be stored in high-performance tiers, while all data operations are stored in slower, high-capacity tiers. Metadata is required to be stored in either PMEM or DRAM, and it is assumed that a user's pool will contain a sufficient amount of space to store metadata.

DAOS provides a variety of interfaces for interacting with containers, including POSIX, MPI-IO, HDF5, Spark and Hadoop. For POSIX, both a FUSE filesystem and an interceptor (which can be loaded using LD_PRELOAD) are provided. The filesystem interceptor is intended to reduce the overheads imposed by FUSE. Unlike a typical parallel filesystem such as OrangeFS, DAOS stores variable-sized blobs of information and does not use fixed-sized striping. This avoids high-latency accesses for I/O operations smaller than the stripe size.

Existing Benchmarks: Recently published works have aimed to measure the performance implications of DAOS over real hardware at scale. Most of these works are an analysis of the DAOS components themselves [13], comparing DAOS only against itself. Some work has been done to compare the performance of DAOS against other storage systems for IOR-based workloads. The IO500, for example [25], utilizes the IOR and mdtest benchmark suites to generate synthetic workloads to stress the boundaries of storage systems. In 2019, DAOS received top marks compared to other submissions in the 10-node challenge. Other works use a similar approach to compare DAOS against Lustre using IOR [28]. While these works are informative on the best ways to tune DAOS and providing an indication of the types of workloads DAOS will work well for, the existing benchmarks do not demonstrate the implications of DAOS underneath real workloads.

2.2 Motivation

While many proposals have been made to change the way storage systems are designed to incorporate the high-performance of modern storage hardware, it is not well-understood how these design choices have impacted the performance of real scientific applications over real modern hardware. Evaluations of hardware-optimized storage stacks have primarily been in a single-node setting, where network costs are avoided entirely. Evaluations which show distributed impacts are typically conducted over emulated hardware or only NVMe. Lastly, the existing evaluations are based solely on synthetic workloads (e.g., IOR), and do not measure the performance impact on real simulation and deep learning HPC applications. To truly understand the performance impacts of optimizing storage stacks for modern hardware, an evaluation across storage systems using real hardware and applications is necessary.

3 Evaluations

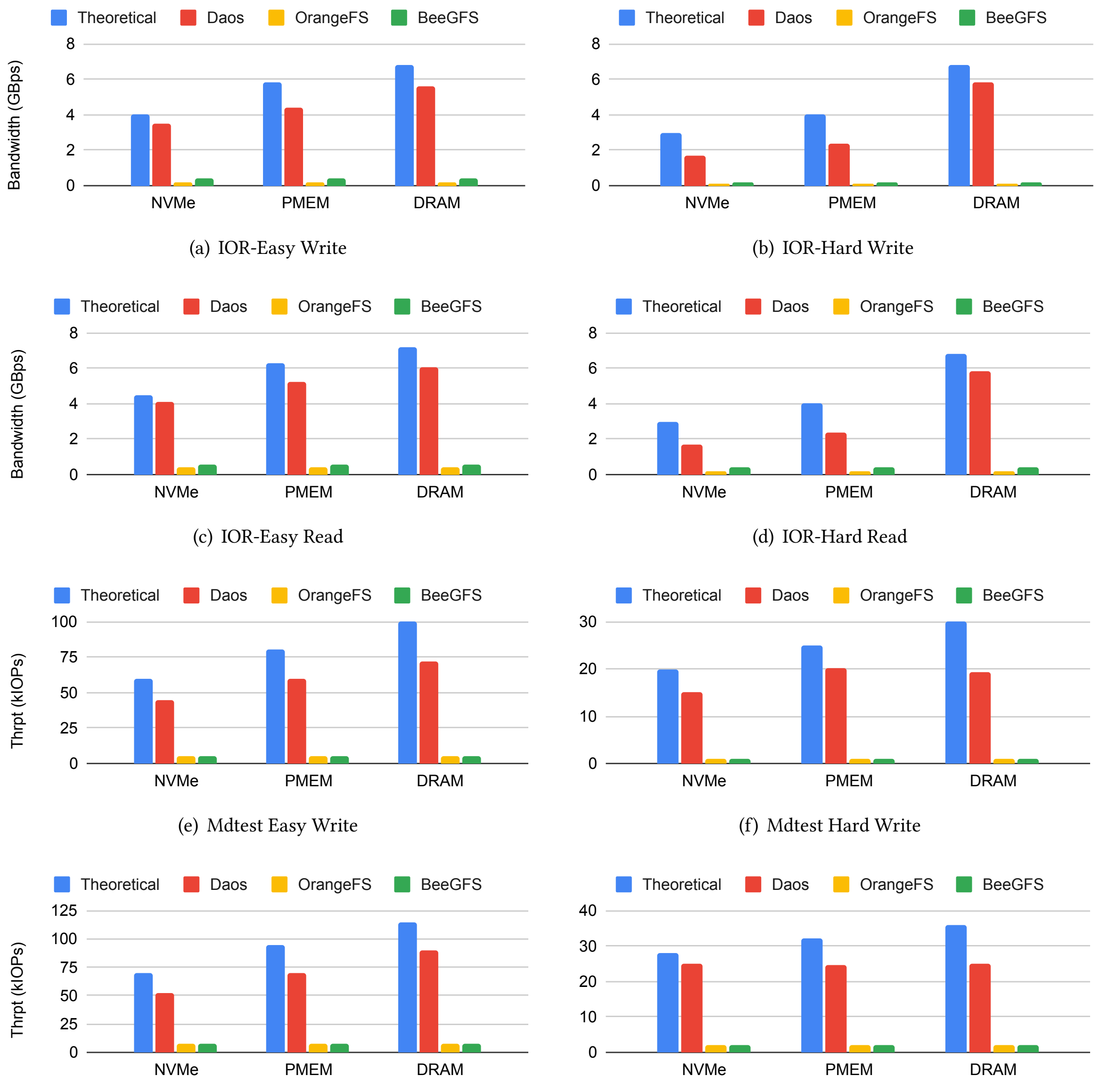
Hardware: We ran all our experiments on a 3-node cluster. Each node contains 512GB RAM, 8x 256GB of Intel Optane DC Persistent Memory, and 16x 4TB NVMes. Each node has 2x Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz, which is 48 cores and 96 threads per-node and 144 cores in total. The network interconnect is 100GBe.

Software: For our experiments, we use Centos8 with kernel 4.18. We install DAOS 2.1.104-tb, OrangeFS 2.9.8, and BeeGFS 3.7.1. For benchmarks, we use IO500 (isc'22 branch) and DLIO (commit: 2a5ed47). We use mpich 3.3.2 for all experiments. Each experiment is executed 3 times and the average is reported.

Experiment Setup: In each experiment, we run the workload generator with 128 processes. In each test, we co-locate the application with the storage/metadata servers. Caches are cleared before every experiment. We use the default configuration for OrangeFS and BeeGFS in all experiments. For OrangeFS, a 64KB stripe size is used and libaio is used as the I/O backend. BeeGFS is also configured with a 64KB stripe size, but automatically scales I/O threads depending on the current workload. We use only one NVMe per-node for these tests. More detail about experimental setup is in each evaluation.

3.1 IO500

In this evaluation, we perform a stress test of the different storage stacks to understand the baseline performance characteristics of each system. To do this, we use the IO500 [21], which is a community benchmark designed to stress storage systems. We compare DAOS, OrangeFS [4], and BeeGFS [9] running over RAM, PMEM and NVMe, each using RDMA-capable networks. OrangeFS and BeeGFS are traditional parallel filesystems which are designed to be used in high-end-computing environments. Neither system has been specifically optimized for modern storage hardware, outside of RDMA capabilities. For OrangeFS and BeeGFS, we use EXT4 as the filesystem for interacting with storage (for PMEM, DAX is enabled). A stripe size of 64KB is used and data is



(a) IOR-Easy Write

(b) IOR-Hard Write

(c) IOR-Easy Read

(d) IOR-Hard Read

(e) Mdtest Easy Write

(f) Mdtest Hard Write

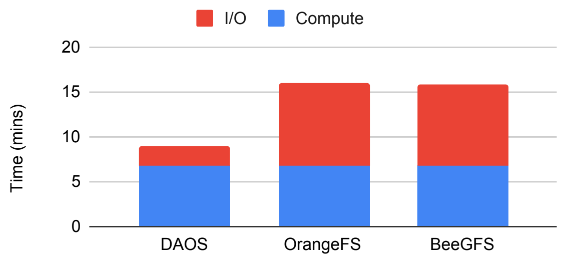
(g) Mdtest Easy Read

(h) Mdtest Hard Read

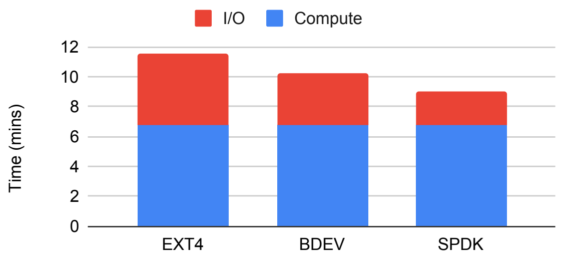
Figure 1. IO500 over various hardware and storage systems. DAOS outperforms OrangeFS and BeeGFS by at least 10x in every workload.

distributed among the metadata servers in a round-robin fashion. Metadata and data servers are co-located. For DAOS, we use the SPDK for storing data on NVMe and DAX for storing data on PMEM. For the NVMe case, DAOS is configured with 50GB of PMEM and 5TB of NVMe; the majority of I/O will be to the NVMe instead of the PMEM. We run the IO500 for 5 minutes for each test. We measure I/O bandwidth and metadata throughput for the various workloads executed by the IO500. We also measure the theoretical bandwidth and throughput of the underlying hardware using the dd tool over the device file for the PMEM and NVMe devices per-node. The measurement of theoretical bandwidth does not account for network impacts.

From Figure 1, it can be observed that DAOS provides performance benefits across the different benchmarks. In terms of bandwidth reported in the IO500-Easy experiment, DAOS outperforms OrangeFS by 10x over NVMe and by 15x over PMEM. IO500-Easy performs a workload which is optimal towards parallel filesystems such as OrangeFS, making large, sequential, and aligned I/O. However, although it is the best-case scenario for OrangeFS, DAOS's leaner I/O stack still provides significant performance improvements. This

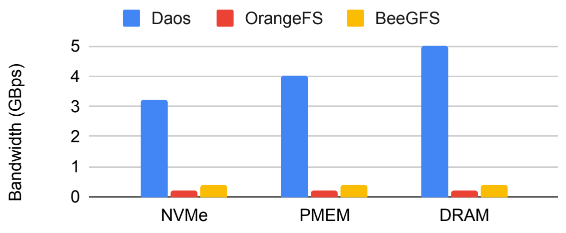


(a) CosmicTagger for various storage systems

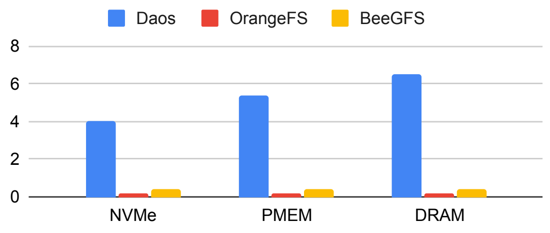


(b) CosmicTagger for various DAOS storage backends

Figure 2. CosmicTagger for various storage systems and storage backend configurations of DAOS. For (a), DAOS performs 2x faster than the traditional storage stacks. For (b), The storage backend varies between SPDK, Linux block layer, and the EXT4 filesystem. SPDK performs up to 55% faster than others as the storage backend. Much of the performance gained in (a) is due to the benefits of SPDK.



(a) VPIC



(b) BD-CATS

Figure 3. HPC workloads over different hardware and storage systems. DAOS improves performance significantly over other other storage system types.

is likely due to the performance of OrangeFS and BeeGFS metadata management services for handling the stripes. Each I/O is divided into 64KB stripes, each of which have to be registered with the metadata servers.

IO500-Hard performs a less optimal workload, which generates small and unaligned I/O. In this case, DAOS outperforms both BeeGFS and OrangeFS by 8x on NVMe and 10x over PMEM. There are two reasons for performance differences. First, IO500-Hard stresses metadata and small-I/O performance significantly more, which accrues overheads due to the kernel I/O stack. Second, BeeGFS and OrangeFS perform I/O in units of stripes (64KB). For I/O which is smaller than this and when boundaries are misaligned, an increased amount of I/O and metadata management occurs.

For the mdtest-easy and mdtest-hard workloads, DAOS performs at least 18x faster than OrangeFS on both NVMe and PMEM. This is because DAOS uses a minimalistic I/O path for storing and querying metadata. Both BeeGFS and OrangeFS rely on the kernel's I/O stack. OrangeFS is a FUSE filesystem running atop EXT4, and BeeGFS is a kernel-level filesystem also running atop EXT4. While metadata queries don't (typically) go directly to disk, they must travel through multiple levels of software and network in order to complete. This leads to a long, expensive I/O path for every metadata access.

In the basic synthetic stress tests, the leaner software stack provided by DAOS significantly outperforms the traditional storage stacks in each category of metadata throughput, latency, and bandwidth. While this knowledge gives indication that DAOS will perform well underneath real applications, we demonstrate the performance impacts in a few real workloads.

3.2 Machine Learning

In this evaluation, we measure the performance impact of DAOS compared to traditional storage stacks for a real deep learning HPC workload. For this, we run CosmicTagger [1] through the DLIO [11] benchmark, which is a convolutional neural network for separating cosmic pixels, background pixels, and neutrino pixels. The training dataset contains 430,000 samples, where each sample contains three images of size 1280x2048. The samples are stored in an HDF5 dataset sparsely. At each iteration, 32 images are read from the dataset and preprocessed. Most I/Os are between 20 and 50KB in size. The total size of the dataset is 450GB. DAOS is configured to have 50GB of PMEM and 5TB of NVMe. DAOS, OrangeFS, and BeeGFS are all configured to use POSIX sockets for their networking operations. DAOS uses SPDK for its storage backend. OrangeFS and BeeGFS use EXT4 for their storage backend.

Overall, we found that CosmicTagger completed its training 2x faster over DAOS compared to the traditional storage stacks. CosmicTagger spends 6.5 minutes of its runtime in computation, which involves analyzing in-memory samples of data for the training. However, for OrangeFS and BeeGFS, CosmicTagger spends 9 minutes of its runtime in I/O. For DAOS, this only takes 2.5 minutes. There are three main reasons for this performance difference. First, DAOS uses the SPDK, which is much lighter than the EXT4 filesystem used by OrangeFS and BeeGFS. Second, the implementation of DAOS was designed around the knowledge that storage hardware is fast, so software overheads were minimized on the critical I/O path. Third, the storage systems have variations in metadata management, caching, and data distribution policies, which can result in unnecessary duplication of data and increased networking overheads.

To quantify the impact of these differences, we re-run the same CosmicTagger workload over various configurations of DAOS. Since the networking backends were the same between DAOS, OrangeFS, and BeeGFS, we focus primarily on quantifying the impact of the storage backend. We configure DAOS to use either SPDK, Linux block layer, or the EXT4 filesystem as the storage backend.

From Figure 2, it is clear there are significant variations in overall performance due to the choice of local storage backend. It can be seen that SPDK performs 35% faster than the Linux kernel block layer and 55% faster than the EXT4 filesystem. Both kernel-level filesystems and the Linux block layer have been noted to impose several overheads [26], such as memory allocations, interrupts, and context switches. When using EXT4, CosmicTagger spends 5 minutes in I/O, compared to the 9 minutes spent by OrangeFS and BeeGFS. This explains about half of the performance lost by the traditional storage stacks in this workload. The remaining time is due to metadata performance from handling the metadata for stripes. Overall, by using the I/O stack optimizations provided in DAOS, a performance benefit of up to 2x can be observed in a real deep learning workload when running over modern hardware.

3.3 Simulation

In this evaluation, we quantify the benefit of an optimized storage stack on common checkpoint-restart simulation workloads experienced in HPC. To do this, we run two common HPC workloads: VPIC [6] and BD-CATS [29]. VPIC is a particle simulation code where each process produces particle data and writes them at each time step. VPIC writes 8 million particles where each particle is a vector of 8 floating point values per-process. We run VPIC for 16 time steps. BD-CATS reads the data generated by VPIC to perform a parallel clustering algorithm. Both VPIC and BD-CATS primarily perform large, aligned I/Os to HDF5 files. The total I/O performed was 500GB for both VPIC and BD-CATS.

From Figure 3, it can be observed that, for both VPIC and BD-CATS, DAOS is about 6x faster than both OrangeFS and BeeGFS over NVMe and PMEM. VPIC and BD-CATS spend roughly 20% of their time in computation, so the majority of the bottleneck is in I/O. The I/O performance benefit is mainly due to the fact BeeGFS and OrangeFS have a significant metadata cost to keep track of stripes, which are distributed and queried among servers during reads and writes. Overall, OrangeFS and BeeGFS service 8 million metadata operations (1 per 64KB stripe). As shown in the IO500 results, metadata operations are in the order of roughly 5-6 kIOPS for OrangeFS and BeeGFS. Their lengthy, kernel-based I/O path causes significant software overhead on the metadata servers, which is where performance degradation arises. Overall, even high-bandwidth HPC workloads can experience great performance benefits by using hardware-optimized storage stacks.

3.4 Discussion

DAOS achieves remarkable performance on PMEM and state-of-the-art NVMes when compared to traditional, kernel-reliant storage stacks such as OrangeFS and BeeGFS. This can be attributed to a number of factors. DAOS was designed specifically for modern hardware characteristics and reduces the software overhead on the critical I/O and metadata path significantly. Hardware-specific I/O interfaces (e.g., SPDK, DAX) are used to avoid the overheads of the kernel stack and avoid unnecessary duplication of data. This accelerates metadata and small I/O operations, which also causes performance impacts in bandwidth-sensitive workloads. Performance is also gained due to differences in metadata management, data placement, and caching. DAOS, for example, doesn't rely on the OS for caching, and provides its own specific cache which supports PMEM, increasing cache capacity and reducing data duplication. This makes DAOS suit the access characteristics of deep learning I/O workloads better, and can result in significant performance benefits due to the reduced metadata cost and software overhead.

However, while DAOS achieves high performance in a small-scale cluster with state-of-the-art hardware, these evaluations do not necessarily reflect the same reality as scale reaches thousands of nodes and as storage hardware becomes more diverse. When increasing scale, network topology and network speed will likely be more impactful and result in a smaller performance gap between DAOS and the traditional storage stacks. In addition, many clusters employ HDDs as the primary storage of data due to capacity constraints, and use NVMe and PMEM only for intermediate buffering of data. Our evaluations assume the datasets fit entirely within NVMe and PMEM and do not go into the complexity of storage tiering across PMEM, NVMe, and HDDs. If the dataset is primarily stored on HDDs, the effects of improved metadata performance and software overheads will likely reduce significantly. For traditional clusters which are based primarily on DRAM and HDDs, the performance of DAOS and traditional storage stacks will likely be much more similar.

4 Conclusion

In this work, we quantified the value of re-architecting distributed storage systems to support the high-performance and low-latency of modern hardware. We performed a comprehensive benchmark of a state-of-the-art storage system (DAOS) and traditional storage systems (OrangeFS + BeeGFS) using a variety of workloads over both persistent memory and NVMe. We quantified the extent to which the software stack optimizations proposed in DAOS affected overall performance of deep learning and simulation workloads. We found that by using SPDK, performance benefits of up to 55% can be observed due to improving small I/O performance. We also found significant performance increases (up to 6x) in both simulation and deep learning workloads due to performance differences in metadata management for querying data stripes during large and small I/O operations. Overall, we showed that real deep learning and HPC workloads can experience performance improvements of up to 6x by using a storage stack designed specifically optimized for modern storage hardware.

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] C Adams, M Alrashed, R An, J Anthony, J Asaadi, A Ashkenazi, M Auger, S Balasubramanian, B Baller, C Barnes, et al. 2019. Design and construction of the MicroBooNE Cosmic Ray Tagger system. *Journal of instrumentation* 14, 04 (2019), P04004.

[2] Abutalib Aghayev, Sage Weil, Michael Kuchnik, Mark Nelson, Gregory R. Ganger, and George Amvrosiadis. 2019. File Systems Unfit as Distributed Storage Backends: Lessons from 10 Years of Ceph Evolution. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. Association for Computing Machinery, New York, NY, USA, 353–369. https://doi.org/10.1145/3341301.3359656

[3] Thomas E Anderson, Marco Canini, Jongyul Kim, Dejan Kostić, Youngjin Kwon, Simon Peter, Waleed Reda, Henry N Schuh, and Emmett Witchel. 2020. Assise: Performance and Availability via Client-local {NVM} in a Distributed File System. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 1011–1027.

[4] Michael Moore David Bonnie, Becky Ligon, Mike Marshall, Walt Ligon, Nicholas Mills, Elaine Quarles Sam Sampson, Shuangyang Yang, and Boyd Wilson. 2011. OrangeFS: Advancing PVFS. In *USENIX Conference on File and Storage Technologies (FAST)*.

[5] Goncalo Borges, Sean Crosby, and Lucien Boland. 2017. CephFS: a new generation storage platform for Australian high energy physics. In *Journal of Physics: Conference Series*, Vol. 898. IOP Publishing, 062015.

[6] Surendra Byna, Jerry Chou, Oliver Rubel, Homa Karimabadi, William S Daughter, Vadim Roytershteyn, E Wes Bethel, Mark Howison, Ke-Jou Hsu, Kuan-Wu Lin, et al. 2012. Parallel I/O, analysis, and visualization of a trillion particle simulation. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[7] Suren Byna, Quincey Koziol, Venkat Vishwanath, Jerome Soumagne, Houjun Tang, Jingqing Mu, Bin Dong, Richard A Warren, François Tessier, Teng Wang, et al. 2018. Proactive Data Containers (PDC): An Object-centric Data Store for Large-scale Computing Systems. In *AGU Fall Meeting Abstracts*, Vol. 2018. IN34B–09.

[8] Jaime Cernuda, Hariharan Devarajan, Luke Logan, Keith Bateman, Neeraj Rajesh, Jie Ye, Anthony Kougkas, and Xian-He Sun. 2021. HFlow: A Dynamic and Elastic Multi-Layered I/O Forwarder. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 114–124.

[9] Fahim Chowdhury, Yue Zhu, Todd Heer, Saul Paredes, Adam Moody, Robin Goldstone, Kathryn Mohror, and Weikuan Yu. 2019. I/o characterization and performance evaluation of beegfs for deep learning. In *Proceedings of the 48th International Conference on Parallel Processing*. 1–10.

[10] Hariharan Devarajan, Anthony Kougkas, Luke Logan, and Xian-He Sun. 2020. Hcompress: Hierarchical data compression for multi-tiered storage environments. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 557–566.

[11] Hariharan Devarajan, Huihuo Zheng, Anthony Kougkas, Xian-He Sun, and Venkatram Vishwanath. 2021. 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, 81–91.

[12] Direct Access for files 2014. Direct Access for files. https://www.kernel.org/doc/Documentation/filesystems/dax.txt

[13] Michael Hennecke. 2023. Understanding DAOS Storage Performance Scalability. In *Proceedings of the HPC Asia 2023 Workshops*. 1–14.

[14] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R Dulloor, et al. 2019. Basic performance measurements of the intel optane DC persistent memory module. *arXiv preprint arXiv:1903.05714* (2019).

[15] Rohan Kadekodi, Se Kwon Lee, Sanidhya Kashyap, Taesoo Kim, Aasheesh Kolli, and Vijay Chidambaram. 2019. SplitFS: Reducing software overhead in file systems for persistent memory. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 494–508.

[16] Sudarsun Kannan, Andrea C Arpaci-Dusseau, Remzi H Arpaci-Dusseau, Yuangang Wang, Jun Xu, and Gopinath Palani. 2018. Designing a True {Direct-Access} File System with {DevFS}. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*. 241–256.

[17] Hyeong-Jun Kim, Young-Sik Lee, and Jin-Soo Kim. 2016. {NVMeDirect}: A User-space {I/O} Framework for Application-specific Optimization on {NVMe} {SSDs}. In *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*.

[18] Anthony Kougkas, Hariharan Devarajan, Keith Bateman, Jaime Cernuda, Neeraj Rajesh, and Xian-He Sun. 2020. Chronolog: a distributed shared tiered log store with time-based data ordering. In *Proceedings of the 36th International Conference on Massive Storage Systems and Technology (MSST 2020)*.

[19] Anthony Kougkas, Hariharan Devarajan, Jay Lofstead, and Xian-He Sun. 2019. LABIOS: A distributed label-based I/O system. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*. 13–24.

[20] Anthony Kougkas, Hariharan Devarajan, and Xian-He Sun. 2018. Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*. 219–230.

[21] Julian Kunkel, Gerald Fredrick Lofstead, and John Bent. 2017. *The Virtual Institute for I/O and the IO-500.* Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[22] Youngjin Kwon, Henrique Fingler, Tyler Hunt, Simon Peter, Emmett Witchel, and Thomas Anderson. 2017. Strata: A cross media file system. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 460–477.

[23] Gyusun Lee, Seokha Shin, Wonsuk Song, Tae Jun Ham, Jae W Lee, and Jinkyu Jeong. 2019. Asynchronous I/O stack: A low-latency kernel I/O stack for ultra-low latency SSDs. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*. 603–616.

[24] Jay Lofstead, Ivo Jimenez, Carlos Maltzahn, Quincey Koziol, John Bent, and Eric Barton. 2016. DAOS and friends: a proposal for an exascale storage system. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 585–596.

[25] Jay Lofstead, Georgio Markomanolis, Julian Kunkel, and John Bent. 2019. *IO500 SC19 Lists*. https://doi.org/10.5281/zenodo.6462493

[26] Luke Logan, Jaime Cernuda Garcia, Jay Lofstead, Xian-He Sun, and Anthony Kougkas. 2022. LabStor: A Modular and Extensible Platform for Developing High-Performance, Customized I/O Stacks in Userspace. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'22), November 14–17, 2022.* IEEE.

[27] Luke Logan, Jay Lofstead, Scott Levy, Patrick Widener, Xian-He Sun, and Anthony Kougkas. 2021. pMEMCPY: a simple, lightweight, and portable I/O library for storing data in persistent memory. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 664–670.

[28] Nicolau Manubens, Simon D Smart, Tiago Quintino, and Adrian Jackson. 2022. Performance Comparison of DAOS and Lustre for Object Data Storage Approaches. *arXiv preprint arXiv:2211.09162* (2022).

[29] Md Mostofa Ali Patwary, Suren Byna, Nadathur Rajagopalan Satish, Narayanan Sundaram, Zarija Lukić, Vadim Roytershteyn, Michael J Anderson, Yushu Yao, Pradeep Dubey, et al. 2015. BD-CATS: big data clustering at trillion particle scale. In *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[30] Robert B Ross, George Amvrosiadis, Philip Carns, Charles D Cranor, Matthieu Dorier, Kevin Harms, Greg Ganger, Garth Gibson, Samuel K Gutierrez, Robert Latham, et al. 2020. Mochi: Composing data services for high-performance computing environments. *Journal of Computer Science and Technology* 35, 1 (2020), 121–144.

[31] Jian Xu and Steven Swanson. 2016. {NOVA}: A Log-structured File System for Hybrid {Volatile/Non-volatile} Main Memories. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*. 323–338.

[32] Ziye Yang, James R. Harris, Benjamin Walker, Daniel Verkamp, Changpeng Liu, Cunyin Chang, Gang Cao, Jonathan Stern, Vishal Verma, and Luse E. Paul. 2017. SPDK: A Development Kit to Build High Performance Storage Applications. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 154–161. https://doi.org/10.1109/CloudCom.2017.14

[33] Irene Zhang, Jing Liu, Amanda Austin, Michael Lowell Roberts, and Anirudh Badam. 2019. I'm Not Dead Yet! The Role of the Operating System in a Kernel-Bypass Era. In *Proceedings of the Workshop on Hot Topics in Operating Systems* (Bertinoro, Italy) *(HotOS '19)*. Association for Computing Machinery, New York, NY, USA, 73–80. https://doi.org/10.1145/3317550.3321422

[34] Shengan Zheng, Morteza Hoseinzadeh, and Steven Swanson. 2019. Ziggurat: A Tiered File System for {Non-Volatile} Main Memories and Disks. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*. 207–219.