Performance Evaluation of Object Storages (NHR2022)

Nellie Marie Lackschewitz, Sebastian Krey, Hendrik Nolte,
Steffen Christgau, Sebastian Oeste, Julian Kunkel

October 2022

1 Introduction

POSIX-IO semantics, which describes all guarantees when a call to the POSIX-IO API is done, is a well known problem for the scalability of the storage systems of future HPC systems. The strong consistency required by the POSIX-IO semantics lead to a statefullness with extensive metadata. This has side-effects like a decreasing performance when many IO requests are coming from multiple nodes.
This has already led to a softening of the string consistency model or to other trade-offs. For instance, DataWarp supports strong consistency, but dropped the usage of a node-local page cache to offer a lock-free data access. Similarly, the on HPC systems commonly utilized parallel file system, like Lustre and GPFS, utilize complicated, distributed locking mechanisms to ensure dirty pages are always flushed before another node reads. On the other hand, NFS has relaxed the consistency, where it only guarantees that a file is consistent between the time a file is closed and consecutively opened.

Since enforcing strong consistency is associated with a large performance penalty [1], especially for small file IO, there is an open quest to circumvent this issue altogether. One promising option is the use object storage. Here, the strong consistency is dropped in favor of eventual consistency, thus reducing the statefullness and therefor the overhead drastically. Another advantage of object storage is the native support for flat namespaces, which occur when a large amount of files are stored within a single directory. The problem with those on POSIX filesystems are, that in order to model those flat namepaces, a binary tree of indirect inodes are created, which increases the number of hops necessary to find the final address on the storage target. These scenarios are often observed in the machine learning community. However, there is also often a write-once-read-many workload employed, which is also not taken advantage of by the strict POSIX-IO semantics. There exists already preliminary work on the utilization of direct access to data storage in HPC systems using RADOS [2].

In this report, we have analyzed and compared different POSIX and S3 storage systems. S3 was used since there is already a lot of native support by often used software and libraries, particularly within the ML community. This rich ecosystem was in this first step more advantageous then the possible performance penalty by the http overhead. In addition, S3 also enables secure data sharing between data centers and is more favorable then copying data with scp. In the following, this report first presents the examined storage systems in Section 2, then introduces the used HPC systems in Section 3, and presents the used benchamrks in Section 4.1. In the following sections the results for the different systems are presented which are then in Section 13 concluded.

2 Examined Storage Systems

We briefly introduce the storage systems evaluated.

2.1 Ceph

Ceph is an object-based parallel distributed file system with network storage that provides excellent performance, reliability, and scalability. Additionally, Ceph provides its Cephx authentication system to authenticate users, so that it can identify users and realize authentication [3]. Ceph is an open source software system designed to enable block-, file- and object-storage to quickly adapt to the changing requirements of an organization. It is also the oldest open source block storage system. The architecture of Ceph is shown in figure 1 and visualizes the relations between the components of the Ceph storage platform. The main advantage of Ceph is that it provides interfaces for multiple storage types within a single cluster, eliminating the need for multiple storage solutions or any specialized hardware, thus reducing management overheads[1]. According to [4], Ceph delivers the promised scalability with a significant increase in the write throughput when data are stored in a faster location (in memory). Moreover, the performance degrades slightly while the system scales with an increasing number of clients accessing the cluster, after the saturation point. The setup for Ceph is rather complicated, however there is well established enterprise support. The use-cases for Ceph are cloud infrastructure, private/public cloud storage (both hyper-converged and disaggregated), big data analytics, and rich media.

2.2 MinIO

MinIO[2] is an object storage system similar to Ceph, but it mainly focuses on being an open source S3 storage with support for unstructured data such as images, videos and more with a single object limit of 5 TB. However, individual

[1] `https://ubuntu.com/ceph/what-is-ceph`
[2] `https://min.io`

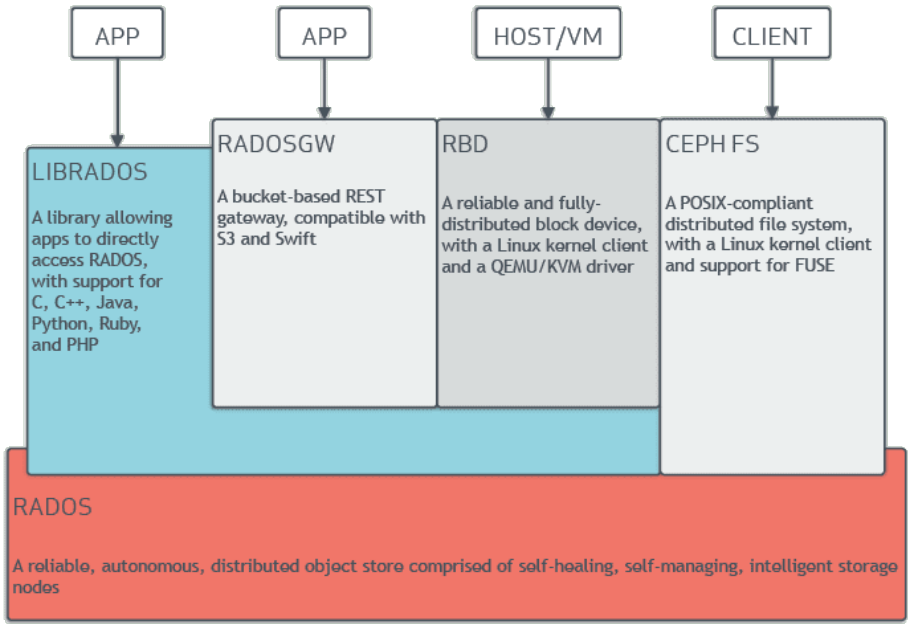


Figure 1: Ceph Stack - Architecture Overview
Source: `https://ceph.io/en/discover/technology/`

upload operations via `PUT` are limited to 5 GB per object such that objects bigger than 5 GB have to be uploaded in multiple parts (details are described in [3]. It is Kubernetes-Native and is considered an open source counterpart to Amazon's S3 while still having compatibility with AWS S3. It is an enterprise level product and its features include active-active-replication for data availability, encryption for data security, automated data and identity management interfaces with data life settings, and is available with extreme scalability. According to [5], the performance analysis results between MinIO, BigchainDB, and IPFS, in the context of object storage systems for edge computing, show that MinIO has the best overall performance regarding: a) query response times, b) RAM consumption, c) disk IO time, and d) transaction rate. MinIO is easy to set up, requiring only a single binary and mountpoints for the storage drives.

2.3 Distributed Asynchronous Object Storage (DAOS)

DAOS[4] is an object storage system designed for massively distributed Non-Volatile Memory (NVM). DAOS take advantage of next-generation NVM technologies, such as Intel© Optane™ Persistent Memory and NVM express (NVMe). It presents a key-value storage interface on top of commodity hardware provid-

[3] `https://min.io/docs/minio/container/operations/checklists/thresholds.html`
[4] `https://docs.daos.io/v2.2/`

ing features including transactional non-blocking I/O, advanced data protection with self-healing, end-to-end data integrity, fine-grained data control, and elastic storage. DAOS is suitable for highly scalable HPC systems supporting structured, semi-structured, and unstructured data models by overcoming the limitations of traditional POSIX-based parallel filesystems [6]. Moreover, for HPC workloads, DAOS provides direct MPI-IO and HDF5 support as well as POSIX access for applications. As per [7], DAOS can provide the required performance, with bandwidth scaling linearly and additional Storage Class Memory (SCM) nodes in most cases, although choices in configuration and application design can impact the resulting throughput.

2.4 OceanStor

OceanStor[5] is a flash-based storage system solution sold by Huawei as a combined hardware and software product. The most recent OceanStor product series saw the release of the OceanStor Pacific[6] storage systems. OceanStor Pacific is an all-flash system with technologies in place to provide high reliability and performance via intelligent frameworks that accelerate hardware performance. It provides balanced read/write performance, but the performance is sometimes inconsistent and needs to be re-checked on site. One downside during setup is the complicated backend network cabling.

2.5 VAST Data

VAST Data[7] is a US private company founded in 2016 with over 140 employees that offers proprietary data storage solutions focusing on flash memory. Its vision for modern storage systems consists of simplifying existing solutions by providing a single monolithic storage system backed by flash memory and a container framework, which is also extendable in terms of storage capacity and performance. VAST's systems use allflash with internal tiering via storage class memory on top of QLC SSDs that can be accessed via S3, NFS and SMB. The storage is commonly optimized for read intensive usage and for optimal performance, NFS over RDMA with a multipath extension should be used.

Furthermore, the storage systems support data compression, similarity based deduplication without losing performance and data replication between storage systems. As part of the software solution, a CLI, GUI and REST API are provided to manage the systems and monitor performance per user and per client. Finally, VAST's systems support wide erasure coding with locally decodable ECs (up to 144+4).

Customers need to purchase hardware and license software from VAST Data with license fees incrementing in 100 TB steps. The smallest storage building block with JBOF that can be purchased from VAST brings 300 TB of usable storage (22 QLC SSDs and 8 SCM, 1U).

[5] `https://e.huawei.com/en/products/storage/all-flash-storage`
[6] `https://e.huawei.com/en/material/storage/3706ec6458964f74aee551a4f2c5059a`
[7] `https://vastdata.com/`

2.6 SeaWeedFS

SeaWeedFS[8][9] is an open source distributed storage system written in Go, initially released in 2015, with support for blob-, object- and file-storage, as well as, data warehouses via Hadoop. It is optimized for large amounts of small files and handles disk seek for files on volumes in O(1) by storing the offset in a volume for each file. Volumes on volume servers are managed by a master node, while metadata is stored separately on filer servers that run SQL or no-SQL databases. Furthermore, SeaWeedFS supports mounting of cloud storage, metadata change subscription via gRPC, strong consistency for writes, POSIX compatible FUSE user clients and integration with Kubernetes via the SeaWeedFS operator and CSI.

Additional features of SeaWeedFS include support for data replication between storage systems, cloud tiering and frontend caching when using external object stores as well as internal tiering with replication for hot and cold data.

On GitHub, SeaWeedFS is relatively popular as it has over 100 contributors and releases new updates regularly with the newest release[10] 3.36 published on the 5th of December 2022.

2.7 WekaFS

WekaFS is a multiprotocol (POSIX, NFS, SMB, S3) low-latency, high IOPS, high-bandwidth file storage platform that is ideal for CPU and GPU accelerated workloads such as AI, machine learning, finance, high performance data analytics, K8s containers, genomics, VFX, DevOps and software builds. Starter clusters deliver 550,000 4K IOPS and 10 GB/second bandwidth, as well as seamless scaling to millions of IOPS and hundreds of GB/second throughput with no downtime. Furthermore, WekaFS on AWS was ranked #1 file system by the Virtual Institute for I/O at SC 2019[11].

To achieve optimal performance in terms of bandwidth and IOPS, the DPDK client should be used as it provides better throughput and scalability compared to the TCP and NFS clients. Nevertheless, the DPDK client requires at least 32 processes per client to reach said performance. It also supports EC for secure and efficient storage.

Weka's clients and server are deployed via containers and can be managed via CLI, GUI and REST API. The installation and management is relatively easy compared to other storage systems thanks to the provided CLI and web GUI. WekaFS must be licensed per TB and can be deployed in the cloud or on owner hardware.

[8] `https://github.com/seaweedfs/seaweedfs`

[9] `https://github.com/seaweedfs/seaweedfs/wiki/SeaweedFS_Architecture.pdf`

[10] As of the time of this writing 11th December 2022.

[11] `https://aws.amazon.com/marketplace/pp/prodview-p57dbxzyqzovo`

2.8 Lustre

The Lustre[12] file system is an open source parallel file system that supports many requirements of leadership class HPC simulation environments. The Lustre file system provides a POSIX compliant file system interface, can scale to thousands of clients, petabytes of storage and hundreds of gigabytes per second of I/O bandwidth. The key components of the Lustre file system are its Metadata Servers (MDS), the Metadata Targets (MDT), Object Storage Servers (OSS), Object Server Targets (OST) and the Lustre clients. Lustre supports internal redundancy via file level mirroring while relying on RAID on the storage sever side.

Due to the size of their machines and I/O requirements, early adopters of Lustre were the Department of Energy National Laboratories including Lawrence Livermore, Sandia, Oak Ridge and, more recently, Los Alamos' Cielo supercomputer is supported by the Lustre file system.

2.9 BeeGFS

BeeGFS[13] is a hardware-independent POSIX parallel file system developed with a focus on performance and designed for ease of use, simple installation, and management. No object storage API is provided, only a POSIX kernel module client. Storage servers run in user-space and support all major enterprise distributions and most currently supported kernel versions. It was created on an Available Source development model, offering a self-supported Community Edition and a fully supported Enterprise Edition with additional features and functionalities. BeeGFS is designed for all performance-oriented environments including HPC, AI and Deep Learning as well as Media & Entertainment, Oil & Gas, and Life Sciences.
Remarkable points:

- Performance: Well-balanced from small to large files
- Scalability: Increase file system performance and capacity, seamlessly and non-disruptive
- Ease of Use: Easy to deploy and integrate with existing infrastructure
- Robust: High availability design enabling continuous operations

BeeGFS achieves internal redundancy only via file level mirroring and relying on RAID storage on the server side.

3 NHR Systems

It is important to understand the scope of the cluster systems that we try to drive using the storage systems. Therefore, we describe here the NHR systems

[12] `https://www.lustre.org/`
[13] `https://www.beegfs.io/`

at the sites for which the storage systems are evaluated.

3.1 NHR@Göttingen

The NHR center Göttingen offers two systems, one CPU-based System called Emmy which was built in 2 phases, and in future an GPU-based System called Grete. Emmy has 1470 CPU nodes (448 nodes with 40 Skylake SP cores, 1022 nodes with 96 Cascadelake-AP cores), with different memory sizes ranging from 192GB to 1536GB. Although Emmy is mostly used for CPU depended tasks, it also has 3 GPU nodes (4xV100 32GB). This will be extended by the 36 nodes of Grete (4xA100 40GB). Emmy has a 8.5 PiB Lustre Scratch file system, with a 130TiB SSD pool built from 2 DDN SFA14KXE, 2 DDN SFA200NVX (each with 2 frontend servers) and a DDN SFA7700X with 4 frontendservers for Metadata. It has 100 8+2 HDD RAID6 targets, 8 declustered RAID NVME SSD targets, and 4 RAID10 metadata targets. In addition, a 340 TiB GPFS/Spectrum Scale provides the Home and Software storage. It is built from one DDN SFA7700X with 2 frontend servers, 12 8+2 HDD RAID6 data NSDs and 4 RAID1 metadata NSDs.

3.2 NHR@TUD

The ZIH as the NHR center for the TUD operates a high performance computing (HPC) system with more than 60.000 cores, 720 GPUs, and a flexible storage hierarchy with about 16 PB total capacity. The HPC system *taurus* consists of 612 nodes each with 2 Intel(R) Xeon(R) CPU E5-2680 v3 (12 cores) processors and main memory ranging from 64 up to 128 GB and local storage of 128 GB SSDs. 192 nodes each with 2 AMD EPYC CPU 7702 (64 cores) and 512 GB main memory and 200 GB local SSD storage. Further, a GPU partition with 34 nodes, each with 8 NVIDIA A100-SXM4 Tensor Core-GPUs and 2 AMD EPYC CPU 7352 (24 cores) and 1 TIB main memory and 3.5 TB local NVME storage. Furthermore 32 IBM Power9 nodes each with 2 IBM Power9 CPU (22 cores), 256 GB main memory and 6 NVIDIA VOLTA V100 with 32 GB HBM2 that provides an NVLINK bandwidth of 150 GB/s between GPU and Host. For shared memory jobs an HPE Superdome Flex system provides 32 Intel(R) Xeon(R) Platinum 8276M CPU (28 cores) and 47 TB main memory configured as a single node, which also provides 370 TB of local NVME storage. For storage an island with 90 NVME storage nodes, each with 8 Intel NVMe Datacenter SSD P4610, 3.2 TB SSDs and 2 Infiniband EDR links, Mellanox MT27800, 100 Gbit/s provides a total of 2 PB of fast storage and offers BeeGFS and WEKA file systems. Further *taurus* has a 4.5 PiB lustre file system on spinning disk and a 42.6 TiB system on SSDs. As a warm archive for storage during an HPC project an 10 PiB quobyte storage on spinning disks is available.

3.3 NHR@ZIB / DAOS Environments

NHR@ZIB operates two DAOS environemnts: An experimental testbed and one production system. The production system is connected by Omni-Path to the infrastructure of the HPC system *Lise*. It constists of 20 dual-socket Cascade Lake, dual 100G OmniPath HFI servers which equipped with first generation Optane persistent memory DIMMs (Apache Pass) and NVMe drives. In total, the production system provides about 0.5 PB of stoarge capacity. Contrary, the experimental testbed operates indepently from Lise and comprises two nodes with 3 and 6 TB of Apache Pass DIMMs and 8 and 16 TB of NVMe disks (Intel P4510). The nodes are connected back-to-back with each other using Omni-Path. In all environments, CentOS 7.9 is used as operating system.

4 Performance Analysis

4.1 Benchmarks

IO500 is a benchmark for POSIX storage, with adaptations for MPI-IO and S3 storage. It was designed to provide a reference benchmark with which to construct a list of fastest IO systems, similar to the Top500[14] list of computers with the greatest computing power. The total score is determined by the geometric mean of several individual benchmarks: The default benchmark includes multiple configurations of IOR and MDTest as well as a timing of find operations, an extended mode is available that adds IOR configurations with randomized access patterns and the MD-Workbench benchmark. One possible pitfall with this benchmark is that the storage access via POSIX API could be cached by the Linux kernel of the machine on which the test is run, therefore tests need to be sufficiently large to ensure they cannot be satisfied by the cache.

S3Warp, also known as MinIOWarp, is a benchmark for S3 storage designed by the MinIO Team. Unlike IO500 it is originally and exclusively designed for S3 storage rather than the POSIX storage API. Since it directly addresses S3 via HTTP requests, it cannot run into issues with the Linux kernel page cache. On the other hand, HTTP connections bring their own pitfalls, such as connection keepalive. TCP keepalive should not matter much in a benchmark, since it only serves to keep unused connections open, which should happen rarely if ever in a benchmark scenario, but HTTP keepalive (reusing existing TCP connections for new HTTP requests) could produce significantly better results than negotiating a new TCP connection for each request. This does apply to S3Warp, since it uses the Golang standard library's default setting for keepalive, which is to keep connections alive, and sets the timeout for closing a connection to 15 seconds of inactivity. S3Warp provides multiple benchmark methods, of which the Mixed mode and the Get mode where used for this evaluation. In Mixed mode, a number of PUT, STAT, GET and DELETE requests are made in arbitrary order

[14] `https://top500.org/`

to provide an insight into how various operations perform while the system is generally under load. It returns individual results for the performance of the different request type, but more attention is given here to the total bandwidth and IOPS measurements that are achieved by the system under this mode. Get Mode first issues PUT request to measure the maximum write speed, and then GET requests to measure the maximum read speed. The read and write speeds returned by Get will likely indicate a higher bandwidth than the Mixed mode test, this is by design since those speeds are achieved in isolation, while in a Mixed mode test the system simultaneously also has to complete STAT and DELETE requests. Also keep in mind that the S3 protocol introduces significant overhead since each operation necessitates an HTTP request, so the results, particularly the IOPS performance via S3, is not comparable to that of a POSIX file system interface.

DAOS Benchmarking DAOS provides an optimized POSIX compatibility layer (DFUSE/dfs/liboil) that enables running traditional workloads on the object store. In functional tests carried out in the experimental testbed at ZIB, the POSIX interface proofed to work as intented. Thus, it is principle possible to run the IO500 benchmarks, IOR and MDTest. However, it turned out that the employed Omni-Path interconnect and DAOS software stack are built with assumptions that contradict each other leading to incompatibilities. Therefore it is not possible to successfully run DAOS in combination with native Omni-Path support. While the involved vendors (which happened to be the same in the past) are working on a solution and collaborate with ZIB to test possible solutions, it was not possible to run the benchmarks, or any applications in general, over the employed high performance interconnect using RDMA-features. It was, however, possible to run applications using TCP/IP, but without any RDMA support. As a result, IOR benchmarks results will experience significantly lower performance than in a fully supported environment.

Since DAOS intentionally breaks with POSIX, a benchmark that makes use of DAOS' underlying key-value/object store paradigm was developed in addition to IOR. This benchmarks employs the key-value API of DAOS and allows to measure the time for performing a number of consecutive operations to put, enumerate, get, and delete key-value pairs to/from a DAOS system. For put operations, the value sizes are configurable. In case of get operation, it is possible to measure the time for retrieving values for existing keys, querying the size of the values for existing keys, and querying non-existing values.

5 Results for Ceph

5.1 Results at NHR@Göttingen

Describes briefly the setup used for testing. Show the results and discuss them.

Object size	Read [obj/s]	Write [obj/s]	Read [MiB/s]	Write [MiB/s]
4 KiB	98959.9	8663.98	377.5	33.1
16 MiB (cached)	1509.45		23032.35	
16 MiB	774.69		11820.85	

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	35779.63	81.89
64 MiB	338.6	12314.98
128 MiB	116.52	7999.01

Table 1: S3 Results for Ceph. Total IOPS and bandwidth based on Warp Mixed tests, read and write speed based on Warp Get tests. Excluding file sizes for which only one test was performed.

6 Results for MinIO

6.1 Results at NHR@Göttingen

MinIO was run on seven service nodes of the Emmy Cluster, each equipped with two Intel Xeon Silver 4110 processors, 192 GB of memory and two Intel P4510 2 TB NVMe SSDs for the SSD test, and six Seagate Exos 12 TB S-ATA HDDs for the HDD test.

The total performance of MinIO on HDD with its S3 intrface had a maximum of 695.88 MiB/s write speed and 2394.66 MiB/s read speed for 16 MB files, and a miced bandwidth of up to 1237.12 MiB/s. On SSD, the system is predictably more performant, reaching up to 15829.12 MiB/s on the same file size. The read and write speed is also considerably faster at a maximum of 23914.50 MiB/s and 7171.86 MiB/s respectively using 16 MB files, which represents more than 90% of the sum performance limit of the used SSDs. The maximum for IOPS was 9901.16 operations per second on 4 KiB files, which is likely limited mostly by MinIO's Erasure Coding[15] producing a high disk volume compared to the actual payload size for such small objects, as each object is split across an erasure set of between 4 and 16 drives (including parity blocks), with a data block and a sidecar file for metadata on each drive. Therefore, a single 4 KiB object can result in up to 32 inodes, and since each inode comprises 256 bytes of metadata (the actual inode) and a 4 KiB data block on a standard ext4 file system, $32 \cdot 4.25$ KiB = 136 KiB being written to the disks requiring 64 IOPs. Notable about the MinIO results is the difference between read and write speeds, with reading reaching speeds around three times as high as those for writing, while other systems are considerably more balanced in that respect.

[15] `https://min.io/docs/minio/linux/operations/concepts/erasure-coding.html`

Threads per Client	Bandwidth [MiB/s]		
	Mixed	Read	Write
5	1081.22	2394.66	695.88
10	1237.12		

Table 2: Results for 16 MB files accessed from 8 clients, on an HDD-backed 7 server MinIO system. Mixed bandwidth based on Warp Mixed tests, read and write speed based on Warp Get tests

Object size	Read [obj/s]	Write [obj/s]	Read [MiB/s]	Write [MiB/s]
4 KiB	13375	4660	53.50	18.64
16 MiB	1494.6	448.2	23914.50	7171.86
256 MiB	93.1	42.5	23833.04	10876.20

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	9901.16	23.20
16 MiB	1649.47	15829.12

Table 3: Results for SSD-backed 7 server MinIO system with 8 clients and 40 threads per client. Total IOPS and mixed bandwidth based on Warp Mixed tests, read and write speed based on Warp Get tests

6.2 Results at NHR@ZIB

7 Results for BeeGFS

7.1 Results at NHR@Göttingen

Like MinIO, the benchmarks for BeeGFS were run on the same seven Emmy nodes, but here a POSIX and not a S3 interface was used; for the exact specification of those see section Section 6.1. Since there are multiple different interfaces for BeeGFS, the presented benchmarks were performed both via RDMA and via TCP connections over a 100 Gb/s RoCE fabric. When using RDMA, BeeGFS was able to read up to 23.43 GiB/s and write up to 17.83 GiB/s, whereas the TCP connection managed to get a slightly higher read speed at 33.66 GiB/s, retaining a similar writing speed of 17.11 GiB/s. This is nearly the theoretical limit of the used SSDs. The IOPS performance of BeeGFS is also respectable, handling up to 214.95k stat operations per second via RDMA, and up to 280.56k via TCP.

7.2 Results at NHR@TUD

The ZIH at TUD operates serveral BeeGFS file systems in production. The following tests were run on an empty BeeGFS file system deployed on 10 tau-

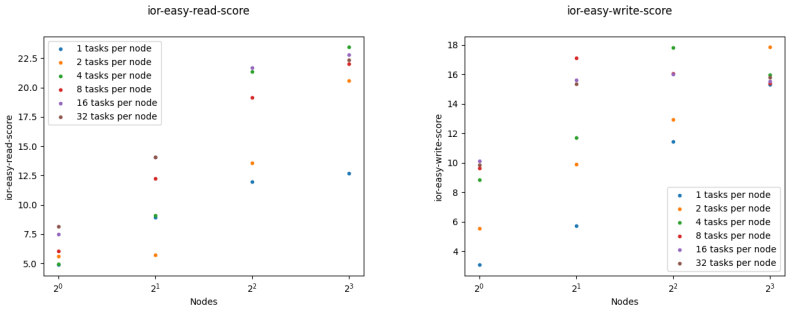


Figure 2: IOR Easy scores for BeeGFS via RDMA. Score corresponds to GiB/s, IOR Easy as part of IO500 accesses 2 MiB per IO call.

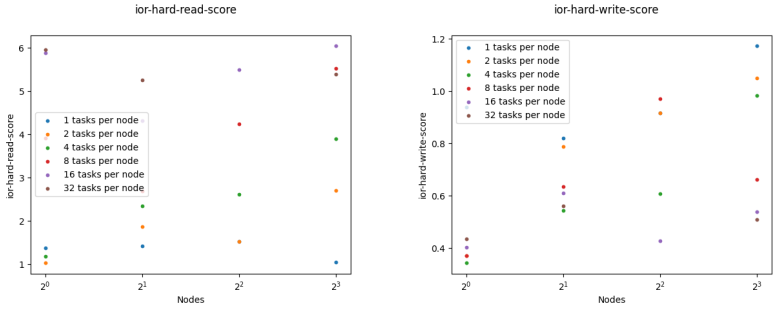


Figure 3: IOR Hard scores for BeeGFS via RDMA. Score corresponds to GiB/s, IOR Hard as part of IO500 accesses 47008 bytes per IO call.

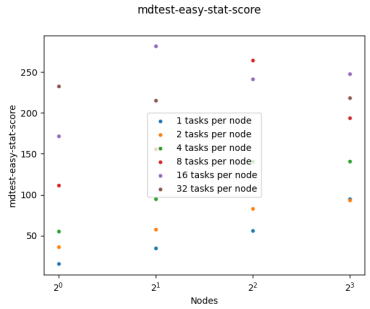


Figure 4: MDTest Stat scores for BeeGFS via RDMA. Score corresponds to kIOPS, MDTest Easy as part of IO500 simply creates and stats empty files.

rusnvme nodes[16], each with 8 3.2 TB NVME SSDs and 2 Infiniband EDR interconnects on the server side. The BeeGFS setup consists of 1 metadata service and 20 storage services running on the 10 server nodes. Each of the

[16] `https://doc.zih.tu-dresden.de/jobs_and_resources/nvme_storage/`

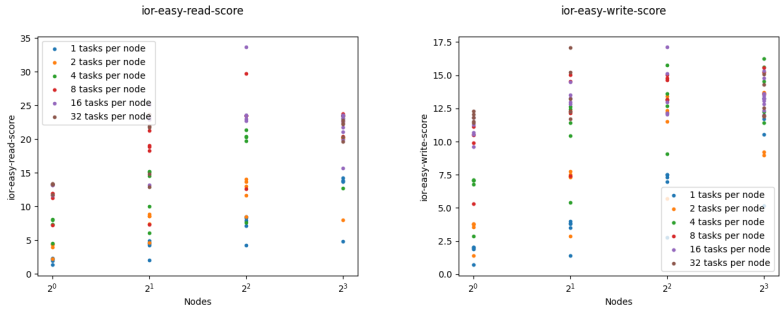


Figure 5: IOR Easy scores for BeeGFS via TCP. Score corresponds to GiB/s, IOR Easy as part of IO500 accesses 2 MiB per IO call.

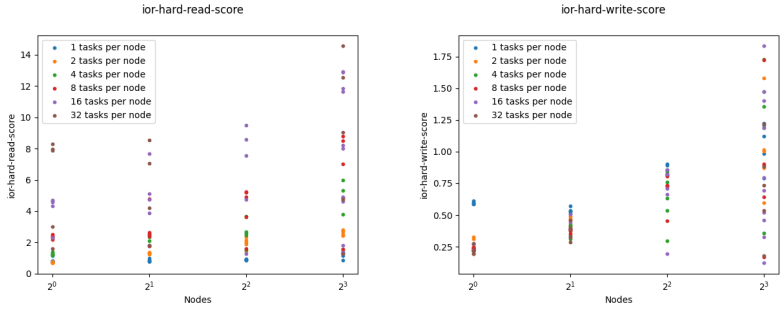


Figure 6: IOR Hard scores for BeeGFS via TCP. Score corresponds to GiB/s, IOR Hard as part of IO500 accesses 47008 bytes per IO call.

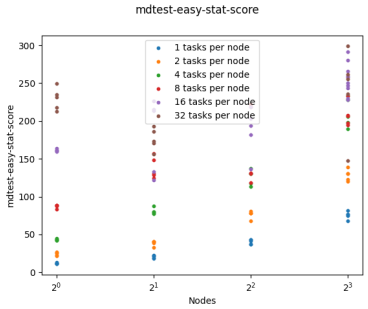
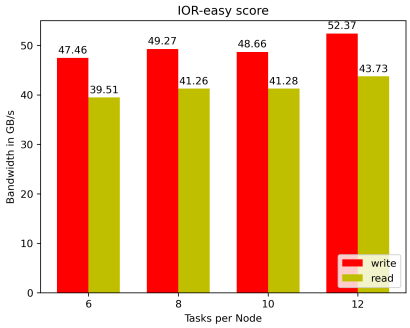


Figure 7: MDTest Stat scores for BeeGFS via TCP. Score corresponds to kIOPS, MDTest Easy as part of IO500 simply creates and stats empty files.

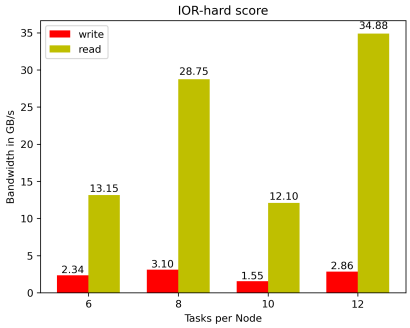
storage services manages 4 NVME SSDs that are connected to the same EDR interconnect in respect to the NUMA affinity. As clients 10 taurus haswell [17]

[17] `https://doc.zih.tu-dresden.de/jobs_and_resources/hardware_overview/#island-6-intel-haswell-cpus`

compute nodes were used. Figure 8 shows the results for the IOR runs of the IO500. Figure 8a shows that BeeGFS delivers good performance for large files. The achived bandwidth is close to the maximum of the underlying network that is limited by the FDR Infiniband interconnects of the client nodes. For the IOR-hard case in figure 8b BeeGFS shows a reasonable read performance with up to 34 GB/s for small random files, the write performance suffers from the challenging access pattern of IOR-hard, varying at 2-3 GB/s. In Figure 9 the MDTest results are shown. Figure 9a shows similar results for *mdtest-easy-write* and *mdtest-easy-stat* over all task configurations. For *mdtest-easy-delete* the number of kIOP/s double with 12 tasks per node. In Figure 9b the results for the mdtest-hard cases show for each operation similar results except for the 12 tasks per node configuration, where the kIOP/s of *write* and *delete* operations grow by a factor of 4, *read* operations grow by factor 5. The *stat* operations show not a clear pattern the results vary between ∼70 and ∼100 kIOP/s. As a conclusion BeeGFS shows stable bandwidth results that are more dependent on the access pattern than on the number of tasks as soon as the threads on the client nodes can saturate the network device. For metadata performance more tasks on the client nodes reach higher kIOP/s.



(a) The achived bandwidth of IOR-easy is close to the maximum of the underlying network.



(b) IOR hard shows reasonable read performance for small files.

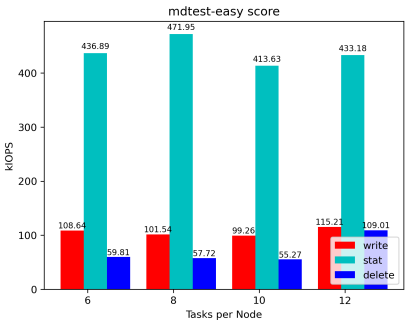
Figure 8: IOR results for BeeGFS at TU Dresden.

8 Results for Vast

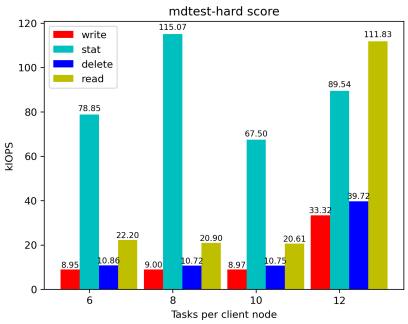
8.1 Results at NHR@Göttingen

8.1.1 S3Warp results

The tests were conducted on a setup of one VAST dBOX (storage enclosure) with 600 TB capacity, and 8 cBOXes (server enclosures), running the actual tests from 8 clients based on MinIO servers connected via RDMA over Ethernet



(a) MDTest-easy shows ∼400 kIOP/s for *stat* ∼100 kIOP/s for *write* and up to 100 kIOP/s for *delete* operations.



(b) MDTest-hard shows better results when using 12 tasks per node except for *stat* operation.

Figure 9: Mdtest results for BeeGFS at TU Dresden.

and TCP. Vast's bandwidth when operating with large files is more or less average (maximum of 2926.22 MiB/s reading and 766.23 MiB/s writing), and while the bandwidth significantly degraded under non-optimal conditions (e.g. a minimum bandwidth of 0.65 MiB/s for writing 1 KiB files), Vast excelled at IOPS for small files, as could be expected since this is one of their main advertising points (up to 4620.12 operations per second for 16 KiB files). Worth noting is that Vast uses both Intel Optane and QLC-SSDs for storage, but writing always goes to Optane, and reading directly after writing, when the data is likely still on Optane storage, provided no significantly different results compared to reading only after forcing the system to flush all data to QLC storage.

Object size	Read [obj/s]	Write [obj/s]	Read [MiB/s]	Write [MiB/s]
4 KiB	95165.34	22007.79	363.03	83.95
256 MiB	87.46	21.89	21841.94	5344.47

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	36897.36	84.45
256 MiB	141.44	19844.07

Table 4: S3 Results for Vast. Total IOPS and bandwidth based on Warp Mixed tests, read and write speed based on Warp Get tests. Excluding file sizes for which only one test was performed.

8.1.2 IO500 results

Tests were also performed against the VAST client for POSIX file system access, using the same infrastructure as above and the IO500 test suite. Four

different modes of connectivity were sufficiently tested to compare them here: The Linux kernel implementation of RDMA (Remote Direct Memory Access), RDMA via NVIDIA Mellanox ConnectX-4 and ConnectX-5 adapters, and TCP connections. In the results we can see that the performance scales rather nicely with the amount of parallel clients, and we see the slight performance differences based on the connection mode, with RDMA using Mellanox ConnectX-5 being the fasted option (but of course also more expensive in terms of hardware cost).

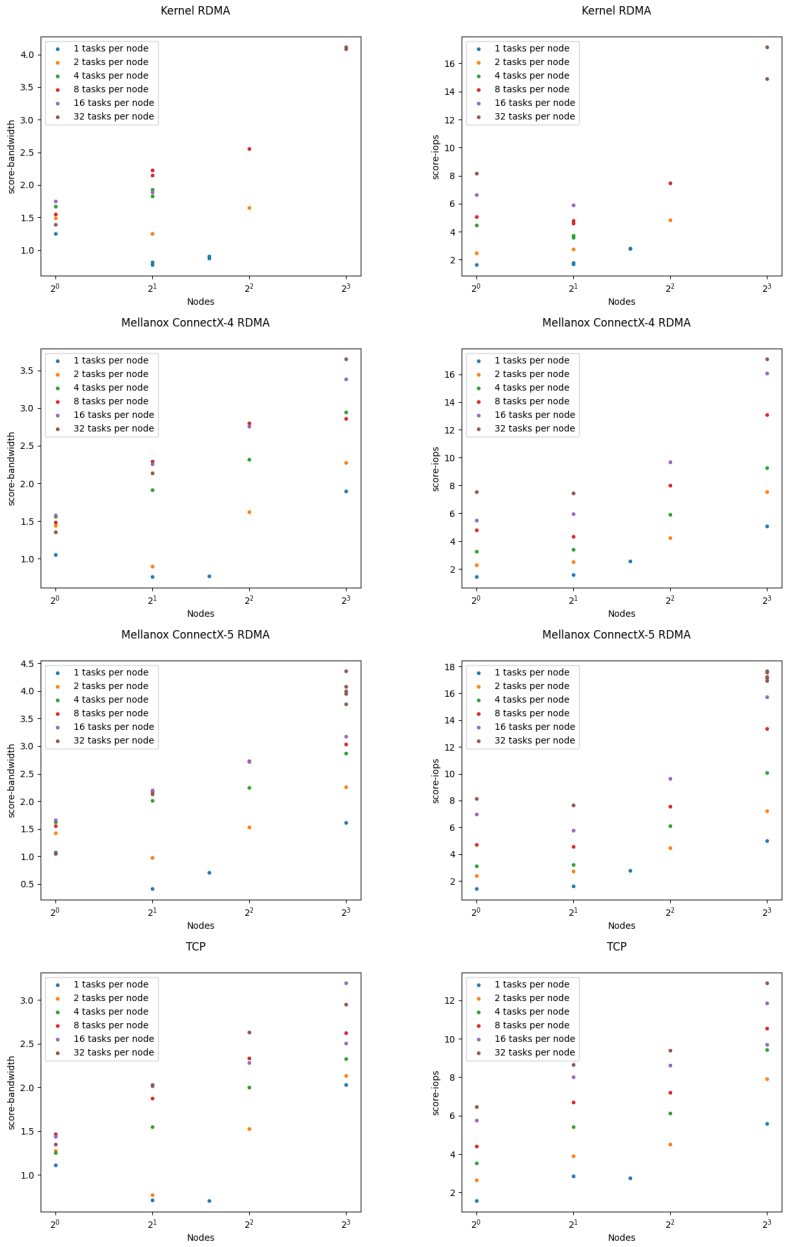


Figure 10: Vast IO500 results. Bandwidth given in GiB/s, IOPS given in 1000 operations per second

9 Results for WekaFS

9.1 Results at NHR@Göttingen

The test environment for Weka was the same as for MinIO and BeeGFS, see section 6.1 for details. Weka's maximum bandwidth on large files (ior-easy and warp with large objects) is similar to that of BeeGFS at a maximum of 27.92 GiB/s reading and 13.99 GiB/s writing reaching nearly the theoretical limit of the installed SSDs (EC overhead has to be added to these numbers, BeeGFS has no EC). The metadata IOPS of up to 82k obj/s for file creation or 913k obj/s for stat in the IO500 benchmarks are very high. Weka's S3 interface was not thoroughly benchmarked in the scope of this evaluation, as it simply uses MinIO as an S3 frontend to access the Weka POSIX file system, and initial benchmarks showed that Weka's performance on S3 is near identical to that of MinIO. The IOPs for the small object S3 benchmarks is similar to MinIO (around 9.9k obj/s with 4k object size in the warp mixed benchmark), and the same applies to the bandwidth benchmarks. The IO500 metadata IOPs are not comparable to S3 due to the difference in overhead between S3, which needs to make an HTTP request for each operation, and POSIX file systems.

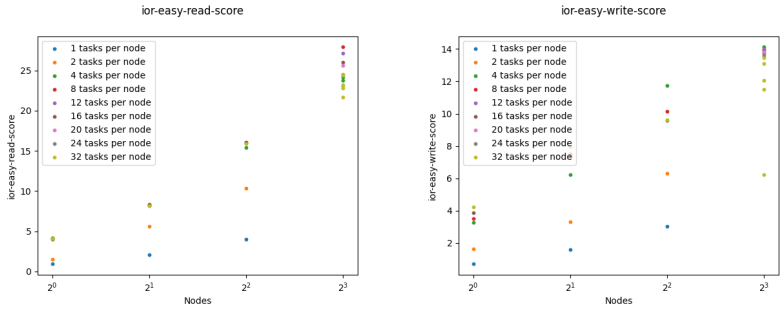


Figure 11: IOR Easy scores for WekaFS. Score corresponds to GiB/s, IOR Easy as part of IO500 accesses 2 MiB per IO call.

9.2 Results at NHR@TUD

The Weka installation of TUD consists of 16 taurusnvme nodes[18], each with 8 3.2 TB NVME SSDs and 2 Infiniband EDR interconnects, for the server side. The file system is mounted on 10 client nodes of the taurus AMD (romeo) partition. The maximum bandwidth is capped by the FDR Infiniband switches between storage and compute nodes. As depicted in Figure 13 Weka shows good bandwidth results for writing large files (ior-easy) 13a and reasonable bandwidth for writing small files (ior-hard) 13b. The IOPS for metadata operations are shown in figure 14. Figure 14a shows that Weka achived up to 131 kIOP/s for file

[18] `https://doc.zih.tu-dresden.de/jobs_and_resources/nvme_storage/`

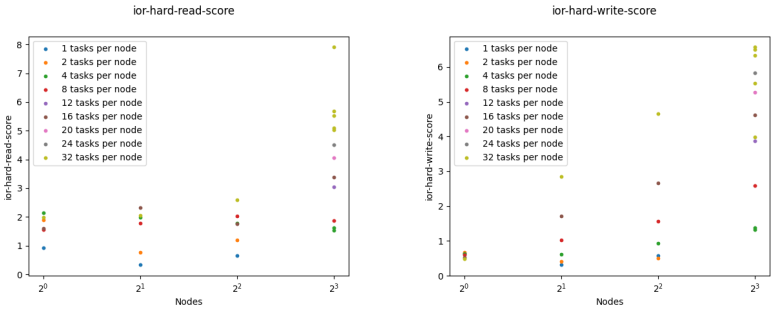
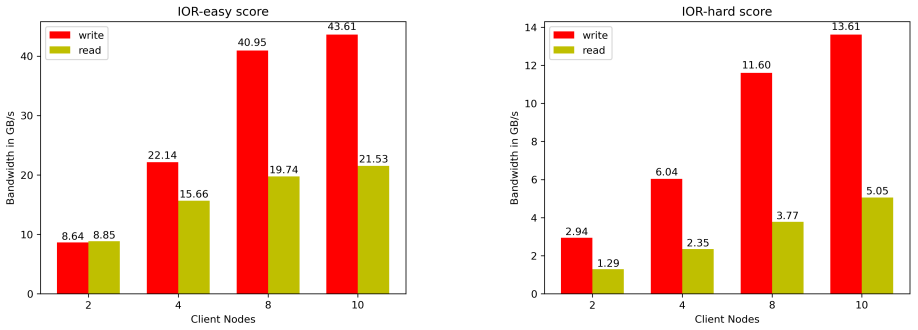


Figure 12: IOR Hard scores for WekaFS. Score corresponds to GiB/s, IOR Hard as part of IO500 accesses 47008 bytes per IO call.

creation and 672 kIOP/s for stat operations also delete operations are a factor of 2 higher than for BeeGFS for the same number of client nodes. Figure 14b shows a scalable behavior of the metadata operations for the different numbers of client nodes in the mdtest-hard case. In conclusion WekaFS is able to deliver very high IOP/s for metadata operations and bandwidths that are comparable with traditional HPC file systems.



(a) IOR-easy results show write performance close to the theoretical maximum of the underlying network.

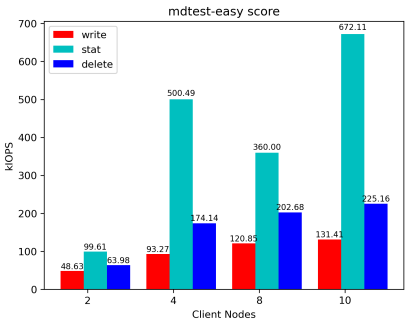
(b) IOR-hard show scalable performance of WekaFS for small random files.

Figure 13: IOR results for WekaFS at TU Dresden.

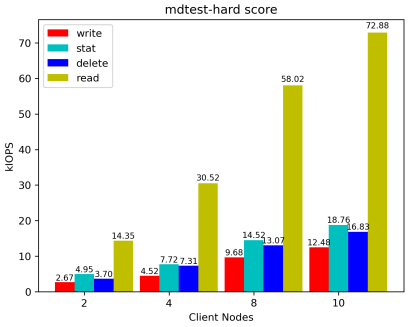
10 Results for Oceanstor

10.1 Results at NHR@Göttingen

For the purpose of these tests, Huawei provided an Oceanstor Pacific 9950 equipped with 4 servers and 10 NVMe SSDs per Server. Tests were performed from Broadwell Xeon based clients with a 100G Ethernet connection. Get-



(a) Mdtest-easy scores show high values ∼700 kIOP/s for *stat* operations and scalable kIOP/s for *write* and *delete* operations.



(b) Mdtest-hard scores show scalable metadata operation over the number of nodes.

Figure 14: MDTest results for WekaFS at TU Dresden.

ting close to the high bar Vast set for IOPS via S3, Oceanstor manages up to 32948.41 operations per second. At the same time however, it far outperforms Vast in terms of streaming bandwidth. It can reach a combined bandwidth of up to 13570.44 MiB/s, only beat by the SSD-backed MinIO system in this benchmark.

Object size	Read [obj/s]	Write [obj/s]	Read [MiB/s]	Write [MiB/s]
64 KiB	44049	21880	2753.06	1367.5
256 MiB	74,9	35,8	19170.38	9156.71

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	31023.44	72.71
64 KiB	32948.41	1235.56
16 MiB	1413.92	13570.44
256 MiB	99.13	15202.70

Table 5: Results for Oceanstor system. Total IOPS and bandwidth based on Warp Mixed tests, read and write speed based on Warp Get tests. Excluding file sizes for which only one test was performed.

11 SeaweedFS

This system was also tested on the same Emmy service nodes as e.g. MinIO. It provides some promising results, however one has to keep in mind that this is tested on hot data, SeaweedFS eventually compresses cold data, so accessing it would be slower. One possible use where this is less of a problem is the use

of SeaweedFS as a caching solution in front of another S3 object storage, this should be explored.

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	58836.57	137.89
4 MiB	4495.49	10785.84

Table 6: Results of Warp Mixed tests on SeaweedFS

11.1 Results at NHR@Göttingen

Object size	Mixed IOPS [obj/s]	Mixed Bandwidth [MiB/s]
4 KiB	58836.57	137.89
4 MiB	4495.49	10785.84

Table 7: Results for SeaweedFS. Total IOPS and bandwidth based on Warp Mixed tests.

12 Results for DAOS from NHR@ZIB

12.1 IOR

IOR benchmark were carried out on up to 16 regular compute nodes of the Lise system. Of the DAOS servers 16 where used. The servers span a DAOS storage pool of 8.5 TB SCM and the pool does not include any of the NVMe drives. IOR was built and used with native DAOS support and the S1 object class was employed while scaling the number of nodes which results in striping accross the DAOS storage targets.

	Read (MiB/s)		Write (MiB/s)	
Client Nodes	Min	Max	Min	Max
1 (peak, 96 PPN)	1350.55	1362.57	800.10	972.84
1 (ior-easy, 48 PPN)	–	–	1641.37	1706.56
1 (ior-hard, 48 PPN)	–	–	1202.46	1230.22
16 (peak, 32 PPN)	20466.60	29018.88	26350.04	27474.95

Table 8: Bandwidths for IOR benchmarks on the DAOS production environment at ZIB.

Results for the DAOS measurements are shown in Table 8. It has to be noticed that the experiements were affected by instabilities of the DAOS system. Therefore, no data for 16 client nodes and 96 PPN were obtained and lower thread counts had to be used. Similar, IOR's easy and hard benchmarks only returned data for writing.

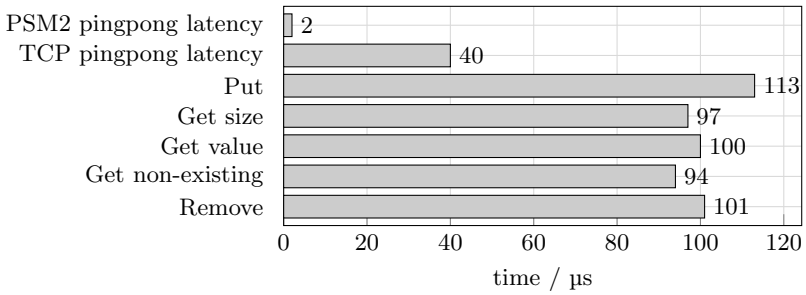


Figure 15: Timings for Key-Value Benchmarks for the DAOS testbed environment at ZIB.

12.2 Key-Value Benchmark

Figure 15 shows the median latency for individual KVS operations in the DAOS testbed environment at ZIB and using the specifically developed benchmark code. A single pool, spanning a single DAOS node, and a single client node where used in this experiment. As described above, the measurement had be conducted using TCP/IP which apparently to affect the performance of the fine-grained and small-sized KVS operations. For comparison, the pingpong latencies obtained from libfabric's `fi_pingpong` utility is shown as well.

The operations themself are quite smilar in performance, even when taking the small standard deviation of 3 µs into account. Nevertheless, insertion (put) of data into the KVS takes notably longer then retrieval (get) and removal. From the the data it can be assumed that network latency (half of the round-trip time) has a significant share on the performance of the individual operations. Additionally, the difference between the PSM2 and TCP libfabric providers suggests that the individual operations may benefit dramatically when native support for the Omni-Path network would be available.

13 Conclusion

13.1 ZIB

DAOS performance is currently not comparable to the other systems due to the lack of native support for the employed Omni-Path HPC network. However, the performance of the key-value store operations is reasonable fast given the high impact of the network baseline. It is assumed that the performance will improve with native support by the Omni-Path network.

13.2 Göttingen

In our testbed we could compare several storage systems with different access protocols on exactly the same hardware (BeeGFS via POSIX, WekaFS

via POISX and S3, MinIO via S3, SeaweedFS via S3). BeeGFS and WekFS with the DPDK based POSIX client could achieve very high bandwidths approaching the physical limits of the used Intel P4500 and P4510 2TB SSDs (3GB/s reading, 1.1/2 GB/s writing), if you take the used redundancy level (none for BeeGFS, EC 5+2 for WekaFS) into account. Both systems are in the range of 2.2-2.5GB/s mixed bandwidth. With large objects the S3 object storages MinIO and SeaweedFS can reach also very high bandwiths but there is a considerable gap (about 1.5GB/s mixed bandwidth). For small objects the image is quite different. While SeaweedFS even exceeds the number of objects per second it can handle compared to BeeGFS and WekaFS POSIX (normalized to the number of active data drives), the number of objects for MinIO is much lower as ist divides even small objects in x+4 (in our testbed 10+4) EC chunks and handles additional sidecar files for metadata. As WekaFS uses a MinIO gateway for S3 access the performance is similar to MinIO.

Additionally we could test Ceph (S3), VASTData (POSIX via parallel NFS, S3) and OceanFS (POSIX and S3) on dedicated hardware. The performance of the VASTData and OceanFS is decent compared on per data SSD. While the VASTData storage is optimized for read intensive workloads, the OceanFS S3 mode has a way more balanced performance profile between read and write. Ceph can provide decent read speeds for small objects, but the write speed and the bandwidth for large objects are really low.

Regarding the POSIX interface BeeGFS and WekaFS are reliable high performance storage systems. The performance of VASTData is lower but providing a very well devloped and enterprise ready management interface and a focus on high reliability. The OceanFS POSIX client has still a rather mixed performance profile. With high number of IO processes per node there is a large risk of load imbalances resulting in low average performance numbers. The shared file performance is very low. Bandwidth and easy metadata operations are very fast.

13.2.1 Comparing S3 to POSIX access

As mentioned earlier, S3 is at a distinct disadvantage in terms of IOPS due to protocol overhead, however good performance is certainly not impossible, as demonstrated for example by Vast. BeeGFS meanwhile is one of the systems whose performance suffers the most when using S3, going down to around 1/6 of the IOPS it delivered via POSIX. This comparison has one key problem though: Different benchmarks. IO500 has two main phases that test IOPS performance: mdtest-easy and mdtest-hard. While mdtest-easy is generally more comparable to the IOPS results of S3Warp because mdtest-hard keeps all files in one directory to test the locking system, mdtest-easy produces slightly better results than an exact match of S3Warp because it solely operates on empty files, while S3Warp also writes content to the files. Therefore, our best comparison is that between mdtest-easy and the Warp runs that use 4 KiB files.

Storage system	S3		POSIX	
	IOPs [obj/s]	BW [MiB/s]	IOPs [obj/s]	BW [MiB/s]
MinIO	9900	15830	-	-
BeeGFS	-	-	60000	30400
WekaFS	≈ 9900	≈ 16000	82000	25020
VAST	36897	19844	48475	23750

Table 9: Comparison between S3 and POSIX

14 Conclusions & Recommendations

Unsere Frage war: Welchen S3 Storage sollte man beschaffen/betreiben und können wir einen Unifieed Namespace bereits sinnvoll nutzen. Was haben wir getan?

Our observations are:

- The S3 based object stores all provide a consistent performance profile, but the suitability for HPC depends strongly on the data structure.
- Large objects can be served from MinIO, SeaweedFS, WekaFS and the appliances from VASTData and the Huawei Oceanstore Pacific (OceanFS) with high performance.
- VAST read performance is about 4x of the write performance and is therefor well suited for write-once-read-many workloads.
- The number of tasks per node heavily influences performance on unaligned or small accesses.
- The small object performance is only for SeaweedFS good, while the performance profile for the commercial VASTData and Hawei OceanFS systems is decent.
- MinIO based solutions provide here a very low performance, due to the sidecar file, and Ceph can only hold up in read speed.
- For POSIX IO, the tested systems BeeGFS and WekaFS provide highest performance and VASTData is a reliable enterprise solution with decent performance.
- Although DAOS is already employed on other HPC systems outside the NHR context, it is a quite new technology for which new workflows and technology have to mature. The usage in production is currently not recommended when used in combination with the Omni-Path interconnect.
- OceanFS still needs development work for providing a consistent performance profile but its development cycles are fast.

At this time, while convenient, we cannot advise to host one solution with a unified S3 + POSIX solution. Although Vast achieves reasonable performance using S3 compared to their POSIX performance, there are still data management challenges in practice when combining both semantics. As an S3 open-source solution, we recommend SeaweedFS, for POSIX BeeGFS. For machine learning workloads with read-mostly characteristics, VAST is useful but also WekaFS with its high metadata performance is worth mentioning.

14.1 Future Work

Comparing these vastly different storage systems, we could observe the limits of our approach. First of all, not all systems could be installed and tested in our own testbed, which makes an apple-to-apple comparison impossible. To accommodate benchmarks done in vendor labs, a more holistic approach in the valuation of results with respect to the underlying hardware is necessary. In addition, pushing and optimizing storage performance, in particular in a tiered, heterogeneous storage environment alone is not enough and will most like not improve the observed performance by the averga user. Instead these architectures need to be developed site-by-site with suitable data management systems to aid the users through a more challenging storage environment.

Acknowledgment

We gratefully acknowledge funding by "Nationales Hochleistungsrechnen" (NHR), a network of computing centers in Germany to provide computing capacity and promote methodological skills.

References

[1] D. Kimpe and R. Ross, "Storage models: Past, present, and future," *High Performance Parallel I/O*, pp. 335–345, 2014.

[2] P. Matri, Y. Alforov, A. Brandon, M. S. Pérez, A. Costan, G. Antoniu, M. Kuhn, P. Carns, and T. Ludwig, "Mission possible: Unify hpc and big data stacks towards application-defined blobs at the storage layer," *Future Generation Computer Systems*, vol. 109, pp. 668–677, 2020.

[3] R. LI, H. Zhu, and Y. Jiaqi, "Formalization and Analysis of Ceph Using Process Algebra," *IEICE Transactions on Information and Systems*, vol. E104.D, pp. 2154–2163, 2021.

[4] D. Gudu, M. Hardt, and A. Streit, "Evaluating the performance and scalability of the Ceph distributed storage system," *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, pp. 177–182, 2015.

[5] A. Makris, I. Kontopoulos, E. Psomakelis, S. Xyalis, T. Theodoropoulos, and K. Tserpes, "Performance Analysis of Storage Systems in Edge Computing Infrastructures," *Applied Sciences*, vol. 12, p. 8923, 2022.

[6] Z. Liang, J. Lombardi, M. Chaarawi, and M. Hennecke, "DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory," in *DAOS*, pp. 40–54, ResearchGet, June 2020.

[7] N. Manubens, T. Quintino, S. Smart, E. Danovaro, and A. Jackson, *DAOS as HPC Storage, a View from Numerical Weather Prediction*. ResearchGet, Aug. 2022.