Baltic J. Modern Computing, Vol. 13 (2025), No. 1, pp. 177–199
https://doi.org/10.22364/bjmc.2025.13.1.10

Performance-Driven and Cost-Efficient Convergence of Cloud and HPC: Evaluating MinIO and LustreFS

Hrachya ASTSATRYAN[1], Hovhannes BAGHDASARYAN[1,2], Ruben ABAGYAN[3], Hovakim GRABSKI[3,4], Siranuysh GRABSKA[3,4]

[1] Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia
[2] Department of Informatics and Computer Engineering, International Scientific-Educational Center of NAS RA, Yerevan, Armenia
[3] Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, San Diego, CA, USA
[4] L.A. Orbeli Institute of Physiology of NAS RA, Yerevan, Armenia

baghdasaryan@ieee.org, abagyan@health.ucsd.edu, hgrabski@health.ucsd.edu, sgrabska@health.ucsd.edu, hrach@sci.am

ORCID 0000-0001-8872-6620, ORCID 0009-0008-3242-5464, ORCID 0000-0001-9309-2976, ORCID 0000-0001-6115-9339, ORCID 0000-0001-9291-3357

Abstract. Integrating cloud technologies into high-performance computing (HPC) systems addresses the rapidly increasing demand for data growth in HPC. The HPC over cloud solutions may simplify complicated workflow, enhance scalability, and improve the end-user experience. Reliable file systems in HPC over cloud environments may efficiently manage vast amounts of data. This study evaluates various approaches to distributed storage and cloud object storage performance. A cost-effectiveness and performance analysis prove that cloud-integrated HPC can offer a scalable alternative to traditional storage solutions.

Keywords: cloud, distributed file system, object storage, Lustre, S3 API

1 Introduction and background

Parallel file systems are fundamental components of high-performance computational (HPC) infrastructure, providing the necessary storage, organization, sharing, protection, and performance capabilities to support the complex computational workflows and data-intensive applications prevalent in the HPC domain (Lockwood et al., 2018). The parallel file systems effectiveness in HPC encompasses scalability, cost-effectiveness, reliability, performance, and manageability. The end of Dennard scaling and Moore's

Law has made it challenging to scale HPC systems within a given performance range, especially in large systems such as supercomputers (Milojicic et al., 2021). Many vendors have deployed scalable cloud object stores to accommodate the continued growth of unstructured data and simplify access to HPC.

Object stores offer faster resource access and cost reduction than traditional file systems. The object stores scale performance and capacity efficiently without hierarchical structures, ensuring seamless growth with demand. Object stores' "key-value" data format enables robust data protection through replication and erasure coding, enhancing durability and reliability. Object stores like Amazon S3 and Google Cloud Storage are widely used for their efficient resource access and scalability without hierarchical structures (Palankar et al., 2008). Leveraging a "key-value" data format, the stores ensure data protection through replication and erasure coding, bolstering durability and reliability. Amazon S3 is a highly reliable and widely used object storage service that offers seamless scalability and comprehensive data protection features. Numerous cloud storage providers offer a RESTful gateway broadly compatible with the S3 interface.

Many libraries face limitations in harnessing the full potential of object storage due to their dependence on traditional file system interfaces like POSIX (Portable Operating System Interface). The POSIX programming API defines a set of operations for interacting with files, directories, and entire file systems. This integration challenge often results in storage sprawl, as object stores are frequently deployed alongside file systems, resulting in ad hoc data access and management across both systems. Storage sprawl causes numerous problems, including over-provisioning, reduced backup efficiency, and cost inflation (Smith, 2016). Managing data complexity across multiple storage systems increases administrative overhead and operational costs. Additionally, the need for redundant storage resources and backup infrastructure further escalates operational expenses. In addition, the coexistence of object storage and traditional file systems often leads to redundant provisioning of storage resources to accommodate both systems' requirements, resulting in wasted resources and increased costs.

The convergence of cloud and HPC storage architectures can effectively address modern storage system challenges to seamlessly integrate object and file storage, consolidating data onto a unified platform (Lofstead et al., 2016). It seeks to enhance storage capabilities while preserving established semantics and interfaces, ensuring consistent and efficient data access across diverse computing environments. Various approaches exist to realize this convergence. One involves integrating S3-compatible gateways with parallel file systems at the storage level (Gadban and Kunkel, 2021). Another approach extends distributed file systems by combining them with cloud object storage (Luettgau et al., 2023). Additionally, efforts enhance the performance of existing cloud-native distributed file systems (Jeong et al., 2019). These strategies bridge the gap between traditional file systems and object storage in converged cloud and HPC environments.

The article studies several cloud and HPC systems' converging concepts and efficient solutions, focusing on cost-effectiveness and performance metrics. Specifically, we evaluate MinIO and LustreFS (Schwan et al., 2003) against S3 object storage to ascertain their suitability in achieving this convergence. The structure of this paper is as follows. Section 2 presents state-of-the-art and related work. The methodology and

experimental environment are described in Section 3. Section 4 contains a performance analysis results. Finally, the last Section 6 provides the conclusion of our study.

2 Related work

In recent years, numerous publications have delved into the convergence of cloud object storage and high-performance file systems, including those by Chen, Li and Ke (2017), Lackschewitz et al. (2022), Huang et al. (2015), Durner et al. (2023), and Liu et al. (2018). These studies shed light on the strengths and limitations of various approaches. Jones et al. (2017) evaluated the high-performance parallel file system Lustre and Amazon's S3, highlighting S3's suitability for sharing vast amounts of data over the Internet. The results show that with proper implementation of parallel I/O, full network bandwidth performance can be attained, ranging from 10 gigabits/s over a 10 GigE S3 connection to 0.35 terabits/s using Lustre on a 1200 port 10 GigE switch. Lustre excels in processing large datasets locally. However, their study did not assess the performance of a file system compatible with the S3 API for HPC environments. In Gadban et al. (2020) study, they examined the RESTful API protocol via HTTP for high-performance file storage, contrasting it with the HPC-native communication protocol Message Passing Interface (MPI) in object storage operations. The authors showed that REST often delivers comparable latency and throughput to MPI implementations. Still, their study did not assess its suitability, efficiency, and performance in handling large file accesses. Notably, their research did not explicitly explore the performance implications of integrating a cloud storage system like S3 within an HPC environment.

Several studies propose implementations aimed at achieving such convergence, such as MarFS (Inman et al., 2017; Chen, Grider and Montoya, 2017) and ArkFS (Cho et al., 2023) scalable distributed file systems designed to be near-POSIX compliant, operating on top of object storage systems. They support S3-compatible platforms with the aid of suitable API translation modules. However, they do not fully adhere to POSIX standards. Many lack support for essential POSIX features like symlinks, hardlinks, or file attributes (chmod), leading to suboptimal performance for random writes (Lillaney et al., 2019).

According to the market survey, supporting POSIX completeness is extremely important for HPC, as most datacenters need to support legacy applications that rely on POSIX semantics (Inman et al., 2017). Therefore, there is the most significant interest in convergence with POSIX-complete file systems. This study addresses the limitation by examining well-established parallel file systems and object storage solutions supporting the S3 API. Among the parallel file systems commonly employed in HPC environments, three widely recognized Free and open-source software systems — Lustre, BeeGFS (formerly known as FhGFS), and DAOS (Distributed Asynchronous Object Storage) were studied. These systems frequently appear in submissions to the IO500 ranking, a semi-annual performance evaluation of HPC storage systems. IO500 evaluates the storage system performance based on bandwidth and metadata performance.

The studies (Lackschewitz et al., 2022; Manubens et al., 2022; Hennecke, 2020) show that DAOS outperforms other parallel file systems in most metrics. At the time of

Table 1. The number of DAOS and Lustre file systems in ISC24 List

ISC24 List	Lustre	DAOS
TOP-25	1	9
TOP-50	7	16
TOP-100	23	20

writing, DAOS-based systems occupy nine positions in the top 25 in IO500 Research ISC24 list[5]. Table 1 shows the number of installations with DAOS and Lustre.

DAOS requires large NVMe (non-volatile memory express) and NVRAM (non-volatile random access memory) devices, making it unsuitable for all environments. In the near future, the relatively high prices of these storage devices will limit the use of DAOS in datacenters and especially in research. With this in mind, the most promising avenues lie in the convergence of cloud storage systems with Lustre (Gadban, 2022). It is worth mentioning that Ceph offers greater redundancy than Lustre, but Lustre is faster in an HPC environment. However, Lustre can be problematic due to its single point of failure.

Table 2. Storage Solution Comparison Matrix

	LustreFS	BeeGFS	JuiceFS	Ceph	MinIO	DAOS
File storage support	yes	yes	yes	yes	no	no
Block storage support	no	no	no	yes	no	no
S3 support	no	no	yes	yes	yes	no
POSIX compliance	high	high	high	acceptable	low	relaxed
Scalability	high	high	low	medium	low	high
High availability	yes	yes	yes	yes	yes	yes
License	GPLv2	GPLv2	Apache License 2.0	LGPLv3	AGPLv3	BSD-2

Comparative characteristics based on preliminary research and available documentation for all considered file systems are presented in Table 2.

[5] IO500 ISC24 Research List - `https://io500.org/list/isc24/io500`

3 Methodology

The methodology section provides an overview of the parallel distributed file systems under examination, describes the benchmarking tools employed, and explains the approach used for cost-effectiveness analysis.

3.1 Overview of parallel distributed file systems

A selection was made of Lustre in combination with the MinIO object store to explore further the convergence approach of integrating S3-compatible gateways with parallel file systems at the storage level. The lightweight architecture of MinIO and its full S3 compatibility make it an optimal candidate for deployment in gateway mode or as a complement to Lustre in hybrid storage environments. As an alternative approach, JuiceFS was considered due to its combination of POSIX compliance and native S3 compatibility. Additionally, Ceph was evaluated as a highly versatile storage system that is effectively used for data redundancy and fault tolerance.

This section presents the overview of Ceph, Lustre, MinIO, and JuiceFS parallel distributed file systems.

3.1.1 Ceph Ceph is a distributed object storage and file system with excellent performance, reliability, and scalability (Weil, Brandt, Miller, Long and Maltzahn, 2006).

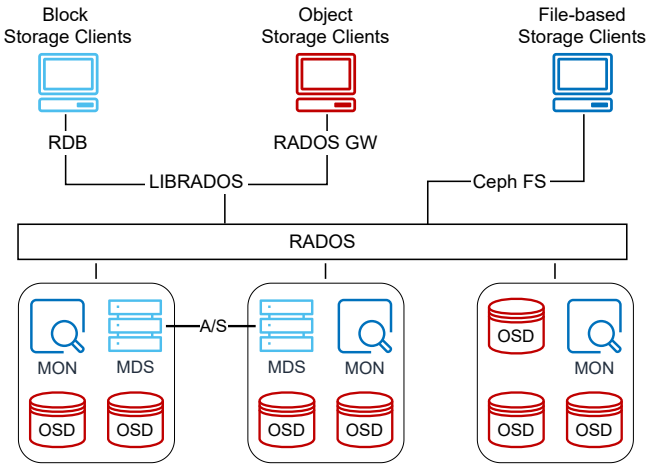

Fig. 1. Ceph architecture.

It operates on a cluster of commodity hardware, utilizing a scalable architecture for seamless expansion as storage requirements grow. Ceph organizes data into objects stored within logical pools. Each is managed independently to optimize performance and reliability. In a Ceph cluster, nodes fulfill three different roles:

- **MDS (Metadata Server)** - act as a metadata service required for the Ceph.
- **OSD (Object Storage Daemon)** - serves as storage resource provider responding to client requests and ensuring data synchronization with other OSD nodes.
- **MON (Monitor)** - responsible for monitoring the overall status of the entire Ceph cluster.

One of the critical components of Ceph is its RADOS (Reliable Autonomic Distributed Object Store) architecture, which ensures data redundancy and fault tolerance by replicating objects across multiple nodes in the cluster (Van der Ster and Wiebalck, 2014). This redundancy enhances data durability and enables high availability and resilience to node failures. In addition to its object storage capabilities, Ceph provides a POSIX-compliant distributed file system called CephFS. CephFS allows users to mount Ceph storage as a traditional file system, enabling seamless integration with existing applications and workflows that rely on standard file access protocols. The LibRados programming interface serves as a basic framework for various client interfaces.

As shown in Fig. 1, Ceph's object storage is exposed through the RADOS gateway, the block storage through the rados block device, and the file system is exposed through Ceph FS. All of these components rely on Librados interfaces for their operation. Ultimately, the data is stored as objects in the RADOS system. Ceph uses the CRUSH (i.e., Controlled Replication Under Scalable Hashing) algorithm (Weil, Brandt, Miller and Maltzahn, 2006) to ensure that data is distributed evenly across the cluster, allowing for easy retrieval by all cluster nodes.

Overall, Ceph could be considered an ideal cloud solution for HPC if there was no performance degradation when scaling the system (Gudu et al., 2014). There are very few studies (Jeong et al., 2019; Zhang et al., 2019; Li et al., 2020) on improving Ceph performance, although this could be a promising direction in the convergence of cloud and HPC.

3.1.2 Lustre Lustre has been one of supercomputers' most popular parallel distributed file systems for many years, offering scalability, high throughput, and low latency. Lustre's architecture was carefully designed to serve as a scalable storage platform for computer networks, using a distributed, object-based storage approach. A Lustre consists of three key components:

- **Metadata Servers (MDS)** - host metadata targets (MDT) per Lustre file system and manage namespace metadata such as file names, access permissions, etc.
- **Object Storage Servers (OSS**) - store file types on object storage target (OST) devices. A Lustre file system is the total capacity of its OSTs.
- **Clients** - access and use the data. Lustre offers a unified namespace for all files and enables standard POSIX semantics.

The typical architecture of Lustre is shown in Fig. 2.

In most scenarios, the MDT, OST, and client components are distributed across different nodes within a Lustre file system and connected over a network. The Lustre Network (LNet) offers compatibility with various network connection options, including InfiniBand connections, Omni-Path, or TCP/IP over Ethernet.

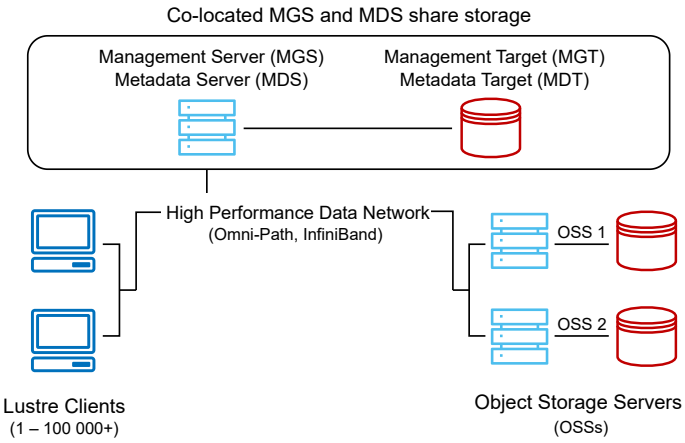

Fig. 2. Lustre architecture.

Lustre provides a coherent, global POSIX-compliant namespace for large-scale computer infrastructure, including the world's largest supercomputer platforms. It can support hundreds of petabytes of data storage and tens of terabytes per second in simultaneous, aggregate throughput (Panda et al., 2022). Such convergence would be promising if the S3 gateway adapts to HPC (Gadban and Kunkel, 2021).

3.1.3 MinIO and JuiceFS MinIO is an object storage solution compatible with the Amazon Web Services S3 API and encompasses the full range of S3 core functionality. MinIO achieves horizontal scalability using a concept known as Server Pools. Server pools integrate technology components, each representing a self-contained group of nodes with computing, networking, and storage resources. In addition, MinIO provides extensive functionality for working with metadata, which is valuable from the end user's perspective (Spiga et al., 2022). Given the S3 compliance, running MinIO in the gateway mode in front of Lustre is one of the convergence scenarios.

Another approach is to implement high-performance POSIX-compliant distributed file systems. One such solution is JuiceFS, an open-source distributed file system compatible with POSIX, Hadoop distributed file system (HDFS) and S3 protocols. Full POSIX compatibility allows almost all kinds of object storage, such as Ceph or MinIO. JuiceFS offers data management, analysis, archiving, and backup APIs (Luettgau et al., 2023). Therefore, JuiceFS integration with MinIO is interesting for further exploration.

JuiceFS is compatible with POSIX, HDFS, and S3 protocols, making it highly versatile and suitable for various use cases. The JuiceFS architecture (Fig. 3) consists of three key components: the metadata engine, the object storage backend, and the client layer. The metadata engine implemented with Redis manages file system metadata such as file hierarchy, permissions, and attributes. The speed and efficiency of Redis make it the ideal choice for this purpose, ensuring that metadata operations do not remain a bottleneck even in large-scale deployments. The object storage backend is where JuiceFS

stores data blocks. JuiceFS, supporting various backends, is flexible in choosing suitable storage solutions. The client layer provides access to the file system through a POSIX-compliant interface. JuiceFS also supports additional protocols, such as NFS and SMB, further improving compatibility.

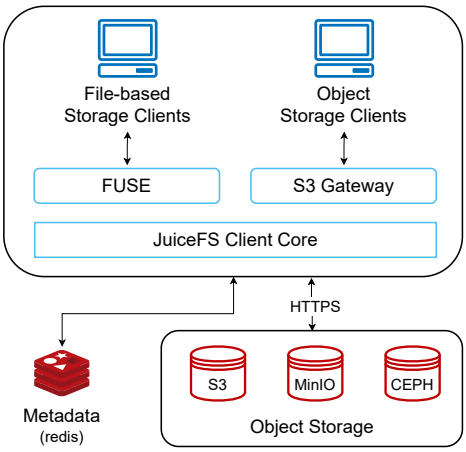

Fig. 3. JuiceFS Architecture.

The integration of JuiceFS and MinIO creates a robust storage solution that combines the scalability and cost-effectiveness of object storage of MinIO with the ease of use and performance of JuiceFS. JuiceFS is an interface layer in this architecture that connects applications to the underlying MinIO storage. The Redis manages the metadata, while MinIO stores and splits data into blocks. The integration is beneficial for optimizing performance through client-side caching. JuiceFS clients cache frequently accessed files locally, significantly reducing latency and increasing throughput for read-intensive workloads. This feature is particularly beneficial in machine learning applications, where large data sets are frequently accessed repeatedly (Luettgau et al., 2023). MinIO's horizontal scalability also allows for handling increasing amounts of data without sacrificing performance. This scalability is achieved by dynamically adding nodes to server pools.

JuiceFS's full POSIX compliance makes it an excellent choice for organizations moving from legacy systems to modern, cloud-native storage. POSIX compliance ensures that applications requiring a traditional hierarchical file system can run seamlessly on JuiceFS without significant changes. This compatibility also extends to MinIO, as JuiceFS translates traditional file operations into object storage operations, enabling efficient use of MinIO as a backend. This design allows organizations to leverage the cost benefits of object storage while maintaining the ease of use of traditional file systems. Additionally, the modular nature of this architecture allows the metadata and data storage components to scale independently, enabling optimal resource utilization.

3.2 Benchmarking tools

COSBench (Cloud Object Storage Benchmark) and MDtest are used to evaluate the performance of storage systems. Each tool offers unique capabilities and focuses on different aspects of storage performance assessment (Zheng et al., 2012).

3.2.1 COSBench COSBench tool developed by Intel evaluates the performance of cloud object storage services, such as Amazon S3, OpenStack Swift, and Ceph RADOS Gateway. The tool simulates workloads and measures key performance metrics, including throughput, bandwidth, latency, and scalability. COSBench evaluates cloud object storage's read and write performance as a system compatible with the S3 protocol. The COSBench interface includes three core operations (create, get, and delete an object) for identifying bottlenecks and measuring capacity.

3.2.2 MDtest (IOR) The IOR, developed by Lawrence Livermore National Laboratory, generates various I/O patterns to evaluate the throughput and latency of storage systems, such as sequential and random reads/writes. IOR supports configurable parameters such as block or file sizes or several processes to tailor the benchmark to specific use cases. It provides detailed performance metrics, including bandwidth, IOPS (I/O operations per second), and access latencies. MDtest is part of the IOR suite that evaluates metadata performance in parallel file systems. It focuses on measuring the performance of metadata operations such as file creation, deletion, and listing. MDtest allows users to generate metadata workloads, including small and large file counts, directory hierarchies, and metadata access patterns. It provides detailed metrics on metadata throughput, latency, and scalability.

3.3 Experimental environment

All experiments were conducted utilizing the CloudLab (Duplyakin et al., 2019), a collaborative initiative involving five US universities and US Ignite, offering robust testbeds tailored for the computer science research community.

Servers within the University of Wisconsin cluster with the c220g2 configuration were selected for the experiments. The configuration details are provided in Table 3.

Table 3. Node configuration of c220g2

CPU	Two Intel E5-2660 v3 10-core CPUs at 2.60 GHz (Haswell EP)
RAM	160 GB ECC Memory (10x 16GB DDR4 2133 MHz dual rank RDIMMs)
Disk 1	One Intel DC S3500 480 GB 6G SATA SSD
Disk 2	Two 1.2 TB 10K RPM 6G SAS SFF HDDs
NIC	Dual-port Intel X520 10GB NIC (PCIe v3.0, 8 lanes)
NIC	Onboard Intel i350 1GB

3.4 Cost evaluation

In this study, cost predictions were made using the AWS Pricing Calculator to estimate expenses associated with various AWS services. The calculations are based on the widely adopted Pay-as-you-go (PAYG) model, wherein users are billed monthly for the specific services they utilize. This approach ensures transparency and accuracy in forecasting expenses, allowing researchers to plan and manage their budgets effectively.

The total cost model can be represented by equation (1), where:
I – total number of connected services.
N – total number of consumed resources from the service i.
p_{ni} – unit price for consumed resource n from the service i.
q_{ni} – quantity of units for consumed resource n from the service i.

$$C_{\text{cloud}} = \sum_{i=1}^{I} \left(\sum_{n=1}^{N} p_{ni} \cdot q_{ni} \right). \tag{1}$$

Such a model is beneficial for small or short-term projects. However, the costs can be excessive for large HPC research projects dealing with large volumes of data, even if the vendor offers significant discounts and cost optimization tools. Therefore, most research studies (Smith et al., 2019; Emeras et al., 2016) on this topic concluded that running scientific workloads on-premises is more cost-effective than running them in the cloud. The main cost factors of on-premises implementation of HPC are shown in Table 4.

Table 4. Main Cost Factors

Capital expenditures (CapEx)	Operating expenditures (OpEx)
Servers	Electricity
Network	Staff
Storage	Maintenance
Facilities	Depreciation
Licenses	Recurring licenses

The equation (2) for total spending over M months can be expressed as follows:

$$C_{\text{on-premises}} = \sum_{m=1}^{M} \left(\frac{C_{CapEx}}{m} + C_{OpEx} \right). \tag{2}$$

The calculations in (Gadban, 2022) show that for $M \geqslant 12$, using an on-premise solution is more advantageous than using the cloud. Furthermore, this inequality becomes even more significant in regions with lower operating costs (including space rental, salaries, and wages).

4 Performance analysis

This section describes the configuration of benchmarking tools and the experimental environment for studying the throughput and average response time for the main operations with the file systems. It presents the results of the experiments carried out.

4.1 Configuration of benchmarking tools and environment

4.1.1 COSBench configuration The COSBench testing procedure uses a workload configuration file to simulate different usage patterns. A workload is represented as an XML file specifying important testing parameters. Key components of this configuration include Special Work blocks and Operation blocks:

Special Work blocks:

```
<workstage name="<name>">
  <work
    type="init|prepare|cleanup|dispose"
    workers="<num>"
    config="<key>=<value>;..." />
</workstage>
```

Operation blocks:

```
<work name="<name>"
  workers="<num>"
  runtime="<num>">
    <operation type="read|write|delete"
    ratio="<1-100>"
    config="<key>=<value>;..." />
</work>
```

Among the parameters that define the configuration, the following should be highlighted:

- **workers**: number of threads to conduct the work in parallel
- **runtime**: duration of the work
- **sizes**: object size with unit (B/KB/MB/GB)

Over 120 configuration files were created for the test environment to simulate various usage patterns, comprehensively analyzing system performance under different conditions.

COSBench evaluates performance using several crucial metrics: average response time (Avg-ResTime), operation count (Op-Count), and throughput. Avg-ResTime measures the average time to complete an operation from request initiation to response. This metric indicates system latency, with lower values signifying higher responsiveness. It helps identify and reduce performance bottlenecks. Op-Count is the total number of operations (read, write, delete) executed during testing. Throughput is the amount of data processed per unit of time, typically measured in bytes per second (B/s). This metric

gauges data transfer efficiency and helps determine optimal configurations for maximizing data processing speed. Analyzing Avg-ResTime, Op-Count, and Throughput helps evaluate the system's performance.

4.1.2 MDTest configuration The first step is to clone the repository and build the application by executing the following commands:

```
$ git clone https://github.com/hpc/ior.git
$ cd ior
$ ./bootstrap
$ ./configure
$ make
```

Then to run MDtest, it is performed the following command:

```
mpirun -n <...> mdtest -n 10000000 -w 3901 -e 3901
```

The main parameters for MDTest configuration include:

- **mpirun -n**: initiates MPI processes
- **-n 1000000**: each process will create 1 000 000 files and directories
- **-w 3901**: writes 3901 bytes to each file after its creation
- **-e 3901**: reads 3901 bytes from each file

These configuration settings evaluate the system's performance under different conditions.

4.1.3 Environment configuration Clusters were deployed using disk images based on Linux kernel version 4.18.x for the experiments, widely adopted by Rocky Linux 8.x and CentOS distributions. This approach ensures compatibility with existing software and hardware configurations. The Ceph setup included an MDS, a MON, and various OSDs, while the Lustre cluster consisted of an MDS and a number of OSSes. The JuiceFS cluster was configured, ensuring that the setup mirrored Ceph and Lustre for evolution (Dai et al., 2019).

The selected configurations are critical for understanding the performance characteristics by considering key performance metrics and behaviors to evaluate various loads and operational scenarios. The evaluation results highlight the capabilities and limitations of each storage solution, guiding future improvements and optimizations.

4.2 Results

4.2.1 COSBench The results are shown in Fig. 4. The highest throughput for Lustre+MinIO is 488.1 – 510.32 ops/s. In addition, Lustre+MinIO has the best average response time performance. Notably, the average response time changes insignificantly as the workload increases, indicating the high scalability of this solution.

Anomalous results were observed for block sizes of 4KB and 4MB, likely due to the minimum chunk size requirement of 5MB for S3. A correlation was noted between

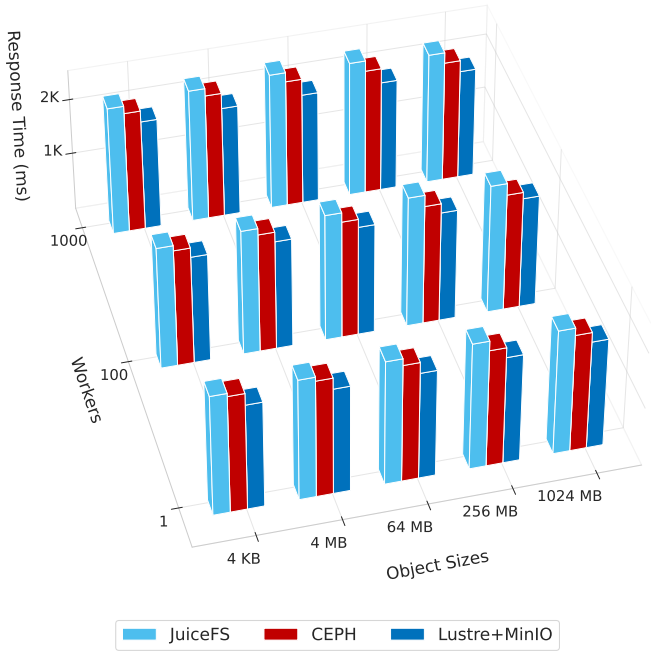

Fig. 4. Average Response Time for the write operation

the variables under consideration for larger chunk sizes. In the case of Ceph, as block sizes increase, the average response time tends to rise, indicating longer durations for write operations. Conversely, throughput increases with larger block sizes and a more significant number of workers. This suggests enhanced efficiency under conditions of higher parallelism and more extensive data sizes.

Similarly, JuiceFS exhibits comparable trends, with the average response time increasing as block sizes and the number of workers rise. However, JuiceFS generally shows a longer average response time than Ceph under similar configurations, highlighting potential performance differences between the two storage systems. In the Lustre with MinIO setup, average response time, and throughput vary with block sizes and worker counts. This configuration demonstrates competitive performance metrics, particularly notable for their lower average response time and higher throughput in specific scenarios compared to Ceph and JuiceFS (see Fig. 5).

4.2.2 MDTest During test iteration, each MPI task generates, parses, and deletes a specified number of directories and files while measuring operation performance per second (ops/s).

The test configuration was designed following the IO500 benchmark guidelines, specifically utilizing variations of mdtest-hard.

The main statistical performance metrics for storage operations are presented in Table 5.

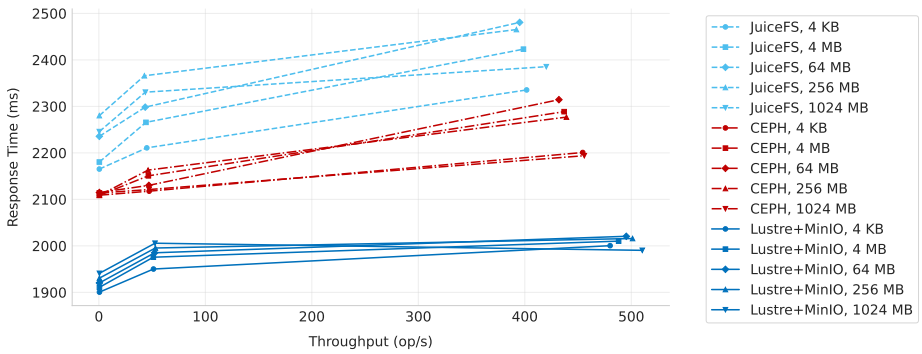

Fig. 5. Throughput and average response time.

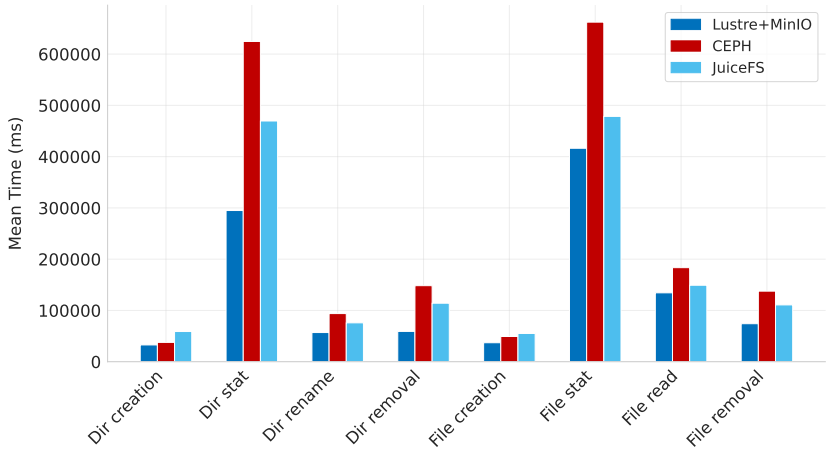

Fig. 6. Mean Performance Comparison of MDTest by Operations

MDTest allows assessment of the maximum, minimum, and average time values for operations such as creation, statistics, rename, and removal for both files and directories. For clarity, Fig. 6 shows averaged metrics across all test iterations on each node for each operation.

As expected, Lustre significantly outperforms its counterparts across all operations.

5 Cost analysis

5.1 Cloud costs

This section develops a cost model to evaluate the costs associated with using Amazon AWS for data processing and cloud storage. The proposed model considers core AWS services, which include Amazon FSx for Lustre, Amazon S3, and Elastic Load Balancing (ELB). These services provide scalable solutions for file storage, object storage,

Table 5. Performance metrics for storage operations, mean time (ms)

Operation	Storage	Max	Min	Mean	Median
Directory creation	Lustre	32744.976	14104.531	23109.016	24385.488
	Ceph	37694.494	28103.387	30661.326	28103.387
	JuiceFS	58899.664	36306.540	44500.488	38396.262
Directory stat	Lustre	294895.873	2815347.026	1471021.501	2815347.026
	Ceph	624617.126	5208374.519	2004635.763	5208374.519
	JuiceFS	469476.606	4139251.949	1543666.445	4139251.949
Directory rename	Lustre	56912.046	1916.461	29414.924	1916.461
	Ceph	93897.423	19557.603	37198.115	19557.603
	JuiceFS	75820.315	43232.581	59567.710	43232.581
Directory removal	Lustre	59122.169	6621.153	32821.761	6621.153
	Ceph	148323.927	98359.250	83700.888	98359.250
	JuiceFS	114115.195	70459.051	91608.189	70459.051
File creation	Lustre	72868.254	1055.402	39469.875	1055.402
	Ceph	49197.739	89360.672	40968.084	89360.672
	JuiceFS	55114.241	79455.935	66175.250	79455.935
File stat	Lustre	4422505.272	6423818.786	2187457.716	6423818.786
	Ceph	662293.384	3094057.244	1222599.920	3094057.244
	JuiceFS	478474.104	4103614.128	1591060.817	4103614.128
File read	Lustre	134355.308	9864.991	62187.950	9864.991
	Ceph	183445.766	897273.291	290572.401	897273.291
	JuiceFS	149146.718	1084472.024	451627.622	1084472.024
File removal	Lustre	125090.710	5420.850	63696.992	5420.850
	Ceph	137649.043	177107.122	115788.872	177107.122
	JuiceFS	110802.135	137756.189	124121.737	137756.189

and load balancing. AWS follows a PAYG pricing model, allowing users to pay only for the consumed resources, with no upfront costs. Discounts of up to 50% are available through reserved or spot instances; however, these options are not considered in this model due to the general unpredictability of resource requirements in most workflows.

The resources included in the model are evaluated based on key pricing components: storage capacity, throughput capacity, backup storage, data transfer, and processing costs. Specific pricing details for each service, excluding compute nodes, which would also be deployed in the cloud environment, as outlined in Table 6.

Table 6. AWS Services Pricing Overview

AWS Service	Pricing Details (USD/GB/month)
FSx for Lustre	
HDD Storage	0.088
SSD Storage	0.794
Backup Storage	0.054
Elastic Load Balancing	
Application Load Balancer	0.008
Network Load Balancer	0.006
Data Processing	0.008
Amazon S3	
Standard Storage (First 50 TB)	0.0245
Standard Storage (Over 50 TB)	0.0235
Glacier Deep Archive	0.00099

Each service contributes specific cost elements based on usage metrics, consolidated into an overall cost model for a comprehensive assessment.

Amazon FSx for Lustre, a service designed for high-performance workloads, incurs costs for HDD and SSD storage, throughput capacity, and backup storage. These costs are monthly TB for storage and megabytes per second per tebibyte (MBps/TiB) for throughput. Therefore, the cost of Amazon FSx for Lustre, costs includes storage, throughput, and backup, expressed as:

$$C_{\text{FSx}}(m) = \left(p_{\text{HDD}} \cdot q_{\text{HDD}} + p_{\text{SSD}} \cdot q_{\text{SSD}} + p_{\text{Throughput}} \cdot q_{\text{Throughput}} + p_{\text{Backup}} \cdot q_{\text{Backup}}\right) \cdot m, \quad (3)$$

where p represents unit pricing, and q represents quantities.

For Amazon S3, costs include inbound storage and data transfer fees, calculated as follows:

$$C_{\text{S3}}(m) = \left(p_{\text{S3}} \cdot q_{\text{S3}} + p_{\text{DT}} \cdot q_{\text{DT}}\right) \cdot m. \quad (4)$$

ELB bases its cost on the volume of data the load balancer processes, typically measured in GB or TB per month. The cost of ELB is determined by the volume of data processed, represented as:

$$C_{\text{ELB}}(m) = p_{\text{ELB}} \cdot q_{\text{ELB}} \cdot m. \quad (5)$$

The total cost of operating in the cloud, C_{cloud}, over m months, is the sum of the expenses from Amazon FSx for Lustre, Amazon S3, and ELB, as defined in Equation (6).

$$C_{\text{cloud}}(m) = C_{\text{FSx}}(m) + C_{\text{S3}}(m) + C_{\text{ELB}}(m). \quad (6)$$

5.2 On-premises HPC costs

The cost evaluation of on-premises solutions focuses on the total expenses required for deploying and operating computing infrastructure locally. These expenses represent the total cost of ownership, which comprises fixed costs, capital expenditures (CapEx), and variable costs, which account for operational expenditures (OpEx). The total costs are expressed as follows:

$$C_{\text{on-premises}} = C_{\text{CapEx}} + C_{\text{OpEx}} \tag{7}$$

Fixed costs (CapEx) are required to build the infrastructure and do not depend on its use over time. These include acquiring hardware components, networking equipment, storage systems, and facility preparation costs. Mathematically, fixed costs can be expressed as:

$$C_{\text{CapEx}} = N \cdot C_{\text{Servers}} + C_{\text{Network}} + C_{\text{Storage}} + C_{\text{Facilities}} \tag{8}$$

where N is the number of hardware units, C_{Servers}, C_{Network}, and C_{Storage} are the costs associated with individual hardware components, network equipment, and storage systems, respectively, and $C_{\text{Facilities}}$ represents the expenses for facility preparation. Facility costs can be excluded if the existing infrastructure can sufficiently host the equipment without modifications.

Variable costs (OpEx) reflect the operating costs incurred over the lifecycle of the system and scale with usage over time. This includes electricity consumption, maintenance, depreciation, software licenses, and personnel costs. Assuming consistent monthly usage, variable costs are modeled as follows:

$$C_{\text{OpEx}} = m \cdot (C_{\text{Electricity}} + C_{\text{Staff}} + C_{\text{Licenses}} + C_{\text{Maintenance}} + C_{\text{Depreciation}}) \tag{9}$$

where m denotes the system's operating time in months. Electricity consumption contributes significantly to variable costs, which are calculated as follows:

$$C_{\text{Electricity}}(h) = \frac{24 \cdot 365}{12 \cdot 1000} \cdot c_{\text{Electricity}} \cdot (N \cdot P_{\text{Servers}} + P_{\text{Network}} + P_{\text{Cooling}}) \tag{10}$$

Here, P_{Servers}, P_{Network}, and P_{Cooling} denote the power consumption of the respective components in Watts, and $c_{\text{Electricity}}$ is the electricity cost per kWh. For systems with power-saving modes such as idle or hibernate, the calculation can be refined to account for the proportion of time spent in each mode.

Maintenance costs, which include hardware repairs and replacements, are often modeled as a percentage of the total initial hardware investment:

$$C_{\text{Maintenance}} = r \cdot (N \cdot C_{\text{Servers}} + C_{\text{Network}} + C_{\text{Storage}}) \tag{11}$$

where r represents the annual maintenance rate. Personnel costs depend on the total working hours required for operation and maintenance, expressed as:

$$C_{\text{Staff}}(h) = h \cdot C_{\text{Staff/hour}} \tag{12}$$

where h is the total monthly working hours and $C_{\text{Staff/hour}}$ is the hourly wage. Software licensing costs C_{Licenses} account for recurring software expenses required for operation. This cost model provides a detailed breakdown of the financial components of deploying and maintaining on-premise solutions. It allows adaptation to specific local conditions such as hardware prices, electricity tariffs, and labor costs and provides a robust framework for evaluating cost-effectiveness compared to alternative computing solutions.

5.3 On-premises and cloud costs comparison

Cost models assess computing infrastructure's financial viability for comparing on-premises solutions and cloud services. These models perform detailed analyses and determine economical approaches for specific computational tasks. Budget, standard, and high-performance on-premise solutions are considered with different complexities and configurations. Table 7 shows the main parameters for calculating the cost of an on-premise solution.

As a cloud service provider, AWS was selected with the nearest data center regarding latency, located in Frankfurt, Germany. The cost analysis for the cloud solution was conducted using formulas (3)–(6) based on data obtained from the AWS calculator[6]. The cost estimation for the on-premises solution was performed using formulas (7)–(12), leveraging publicly available data on equipment specifications, facility expenses, electricity rates[7], and staff salaries[8]. The comparison was visualized through a month-by-month analysis following formulas (1), (2) and is shown in Fig. 7.

The evaluation shows that the on-premises solution incurs significantly higher costs in the initial months due to substantial CapEx. After the first year, the monthly on-premises solution costs exhibit a noticeable increase. This rise reflects the inclusion of annual depreciation and maintenance provisions, accounting for the lifecycle and potential replacement of hardware components. While this adjustment increases monthly costs, the on-premises solution remains more cost-effective than the cloud alternative beyond the 12-month.

In contrast, the cloud-based solution demonstrates a consistent and predictable cost structure, reflecting the subscription-based pricing model of providers such as AWS. This flat monthly cost remains advantageous for short-term projects or applications requiring rapid scalability and minimal initial investment. However, this fixed-cost model becomes less favorable for long-term high-performance workloads due to its inability to benefit from economies of scale or decreasing marginal costs.

The results confirm that on-premises solutions offer clear financial advantages for computationally intensive projects exceeding one year, particularly in regions like Armenia where operational costs—such as electricity, real estate, and labor—are relatively

[6] `https://calculator.aws`

[7] `https://energyagency.am/page_pdf/sakagner`

[8] `https://www.paylab.com/am/salaryinfo/information-technology/systems-administrator`

Table 7. Cost Parameters for On-Premises Configurations

Characteristics	**Budget**	**Standard**	**High**
Configuration Details	10 nodes, basic cooling	50 nodes, air cooling, 150TB storage	150 nodes, CRAH cooling, 600TB storage
Node Specifications	Intel i7 12700, 16GB ECC RAM, 480GB SSD, 1.2TB HDD	Intel Xeon E5-2660 v3, 64GB ECC RAM, 480GB SSD, 1.2TB HDD	Intel Xeon E5-2660 v3, 160GB ECC RAM, 2x480GB SSD, 2x1.2TB HDD, Intel X520 10GbE NIC
Cooling System	Basic air cooling	Air-cooled chiller	CRAH with VFD chiller/tower
C_{Node} (USD)	800	1,500	2,100
C_{Servers} (USD)	8,000	75,000	315,000
C_{Storage} (USD)	—	4,500	18,000
$C_{\text{Facilities}}$ (USD)	—	—	140,000
C_{Network} (USD)	—	—	31,500
$C_{\text{Maintenance}}$ (%)	—	—	5
$C_{\text{Depreciation}}$ (%)	—	—	10
C_{Staff} (USD/month)	—	—	$800 \times 3 = 2,400$
P_{Servers} (W)	200	320	495
P_{Cooling} (W)	2,400	26,250	110,000
P_{Network} (W)	—	—	2,250

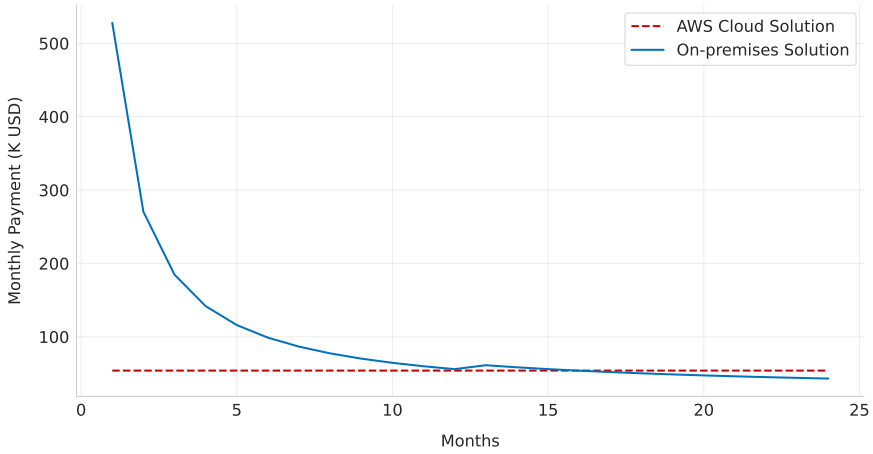

Fig. 7. On-Premises and Cloud Comparison

low. These results corroborate earlier studies, emphasizing the cost-efficiency of on-premises infrastructure in similar economic contexts. The lower costs of electricity and salaries in Armenia further amplify the financial feasibility of this approach and make it a more attractive option for local organizations or research institutions.

In summary, while cloud-based solutions remain suitable for short-term and highly variable computational needs, on-premises infrastructure demonstrates superior cost-effectiveness for long-term, resource-intensive projects. These results provide a tradeoff for decision-making in selecting computational infrastructure, particularly for institutions in regions with favorable economic conditions for on-premises deployment.

6 Conclusion and future work

As demand for cloud-based HPC continues to increase, existing market solutions often cannot meet the needs of large research projects due to high costs. As shown, on-premise solutions are more cost-effective than PAYG models. Despite the topic's relevance and extensive research, there are still hardly any viable alternatives. We have focused on three development strategies for such systems by comprehensively exploring various approaches.

Our research, which examines systems from both a file and object perspective, shows the convergence of Lustre parallel file systems with MinIO object storage as a promising solution. Specifically, as a COSBench-evaluated cloud storage solution, this converged approach performs better than the popular Ceph system, achieving over 20% better average response time and throughput metrics. As an HPC file system, Lustre delivers an average of 30% faster performance across all major operations. This performance advantage becomes much more noticeable with higher workloads.

The convergence of cloud technologies with HPC systems offers significant benefits such as improved scalability, improved data availability, and simplified workflow management. Although the initial setup is more complex compared to Ceph or JuiceFS, the convergence of Lustre with MinIO appears more promising.

Therefore, the convergence of Lustre with MinIO is the most compelling option for further investigation. Further exploration of this solution will contribute to developing an open-source HPC system with a fully integrated S3 gateway. The planned experiments will cover a broader range of workloads and compare the results with alternative object storage solutions, and utilize memory pooling technologies. Additionally, we will extend our experiments to various infrastructures, including serverless computing environments, containerized applications, virtual machines, traditional HPC setups, and clustered systems (Petrosyan and Astsatryan, 2022; Astsatryan et al., 2017, 2004).

Acknowledgements

The research was supported by the Science Committee of the Republic of Armenia by the project entitled "Self-organized Swarm of UAVs Smart Cloud Platform Equipped with Multi-agent Algorithms and Systems" (Nr. 21AG-1B052).

References

Astsatryan, H., Narsisian, W., Kocharyan, A., Da Costa, G., Hankel, A., Oleksiak, A. (2017). Energy optimization methodology for e-infrastructure providers, *Concurrency and Computation: Practice and Experience* **29**(10), e4073.

Astsatryan, H., Shoukourian, Y., Sahakyan, V. (2004). The armcluster project: brief introduction, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'04*, pp. 1291–1295.

Chen, H.-B., Grider, G., Montoya, D. R. (2017). An early functional and performance experiment of the marfs hybrid storage ecosystem, *2017 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, pp. 59–66.

Chen, H.-M., Li, C.-J., Ke, B.-S. (2017). Designing a simple storage services (s3) compatible system based on ceph software-defined storage system, *Proceedings of the 2017 2nd International Conference on Multimedia Systems and Signal Processing*, pp. 6–10.

Cho, K.-J., Kang, I., Kim, J.-S. (2023). Arkfs: A distributed file system on object storage for archiving data in hpc environment, *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE, pp. 301–311.

Dai, D., Gatla, O. R., Zheng, M. (2019). A performance study of lustre file system checker: Bottlenecks and potentials, *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, IEEE, pp. 7–13.

Duplyakin, D., Ricci, R., Maricq, A., Wong, G., Duerig, J., Eide, E., Stoller, L., Hibler, M., Johnson, D., Webb, K. et al. (2019). The design and operation of {CloudLab}, *2019 USENIX annual technical conference (USENIX ATC 19)*, pp. 1–14.

Durner, D., Leis, V., Neumann, T. (2023). Exploiting cloud object storage for high-performance analytics, *Proceedings of the VLDB Endowment* **16**(11), 2769–2782.

Emeras, J., Besseron, X., Varrette, S., Bouvry, P., Peters, B. (2016). Hpc or the cloud: a cost study over an xdem simulation, *Proc. of the 7th International Supercomputing Conference in Mexico (ISUM 2016). Puebla, México*.

Gadban, F. (2022). *Analyzing Convergence Opportunities of HPC and Cloud for Data Intensive Science*, PhD thesis, Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky.

Gadban, F., Kunkel, J. (2021). Analyzing the performance of the s3 object storage api for hpc workloads, *Applied Sciences* **11**(18), 8540.

Gadban, F., Kunkel, J., Ludwig, T. (2020). Investigating the overhead of the rest protocol when using cloud services for hpc storage, *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers 35*, Springer, pp. 161–176.

Gudu, D., Hardt, M., Streit, A. (2014). Evaluating the performance and scalability of the ceph distributed storage system, *2014 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 177–182.

Hennecke, M. (2020). Daos: A scale-out high performance storage stack for storage class memory, *Supercomputing frontiers* **40**.

Huang, W.-C., Lai, C.-C., Lin, C.-A., Liu, C.-M. (2015). File system allocation in cloud storage services with glusterfs and lustre, *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pp. 1167–1170.

Inman, J. T., Vining, W. F., Ransom, G. W., Grider, G. A. (2017). Marfs, a near-posix interface to cloud objects, *; Login* **42**(LA-UR-16-28720; LA-UR-16-28952).

Jeong, K., Duffy, C., Kim, J.-S., Lee, J. (2019). Optimizing the ceph distributed file system for high performance computing, *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 446–451.

Jones, M., Kepner, J., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Michaleas, P., Prout, A. et al. (2017). Performance measurements of supercomputing and cloud storage solutions, *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, pp. 1–5.

Lackschewitz, N. M., Krey, S., Nolte, H., Christgau, S., Oeste, S., Kunkel, J. (2022). Performance evaluation of object storages, *NHR2022* .

Li, H., Zhang, S., Guo, Z., Huang, Z., Qian, L. (2020). Test and optimization of large-scale ceph system, *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, IEEE, pp. 237–241.

Lillaney, K., Tarasov, V., Pease, D., Burns, R. (2019). Towards marrying files to objects, *arXiv preprint arXiv:1908.11780* .

Liu, J., Koziol, Q., Butler, G. F., Fortner, N., Chaarawi, M., Tang, H., Byna, S., Lockwood, G. K., Cheema, R., Kallback-Rose, K. A., Hazen, D., Prabhat, M. (2018). Evaluation of hpc application i/o on object storage systems, *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*, pp. 24–34.

Lockwood, G. K., Snyder, S., Wang, T., Byna, S., Carns, P., Wright, N. J. (2018). A year in the life of a parallel file system, *SC18: International conference for high performance computing, networking, storage and analysis*, IEEE, pp. 931–943.

Lofstead, J., Jimenez, I., Maltzahn, C., Koziol, Q., Bent, J., Barton, E. (2016). Daos and friends: a proposal for an exascale storage system, *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, pp. 585–596.

Luettgau, J., Martinez, H., Olaya, P., Scorzelli, G., Tarcea, G., Lofstead, J., Kirkpatrick, C., Pascucci, V., Taufer, M. (2023). Nsdf-services: Integrating networking, storage, and computing services into a testbed for democratization of data delivery, *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, pp. 1–10.

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

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

Palankar, M. R., Iamnitchi, A., Ripeanu, M., Garfinkel, S. (2008). Amazon s3 for science grids: a viable solution?, *Proceedings of the 2008 international workshop on Data-aware distributed computing*, pp. 55–64.

Panda, D. K., Lu, X., Shankar, D. (2022). *High-performance big data computing*, MIT Press, London.

Petrosyan, D., Astsatryan, H. (2022). Serverless high-performance computing over cloud, *Cybernetics and Information Technologies* **22**(3), 82–92.

Schwan, P. et al. (2003). Lustre: Building a file system for 1000-node clusters, *Proceedings of the 2003 Linux symposium*, Vol. 2003, pp. 380–386.

Smith, H. (2016). *Data Center Storage: Cost-Effective Strategies, Implementation, and Management*, CRC Press, Boca Raton, London, New York.

Smith, P., Harrell, S. L., Younts, A., Zhu, X. (2019). Community clusters or the cloud: Continuing cost assessment of on-premises and cloud hpc in higher education, *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, Association for Computing Machinery, pp. 1–4.

Spiga, D., Ciangottini, D., Costantini, A., Cutini, S., Duma, C., Gasparetto, J., Lubrano, P., Martelli, B., Ronchieri, E., Salomoni, D. et al. (2022). Open-source and cloud-native solutions for managing and analyzing heterogeneous and sensitive clinical data, *International Symposium on Grids and Clouds 2022, ISGC 2022*.

Van der Ster, D., Wiebalck, A. (2014). Building an organic block storage service at cern with ceph, *Journal of Physics: Conference Series*, Vol. 513, IOP Publishing, p. 042047.

Weil, S. A., Brandt, S. A., Miller, E. L., Maltzahn, C. (2006). Crush: Controlled, scalable, decentralized placement of replicated data, *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, pp. 122–es.

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

Zhang, X., Wang, Y., Wang, Q., Zhao, X. (2019). A new approach to double i/o performance for ceph distributed file system in cloud computing, *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, IEEE, pp. 68–75.

Zheng, Q., Chen, H., Wang, Y., Duan, J., Huang, Z. (2012). Cosbench: A benchmark tool for cloud object storage services, *2012 IEEE Fifth International Conference on Cloud Computing*, IEEE, pp. 998–999.

Received December 9, 2024 , revised February 21, 2025, accepted February 21, 2025