IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 August 2023, accepted 8 October 2023, date of publication 16 October 2023, date of current version 25 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324959

RESEARCH ARTICLE

Performance Analysis of Distributed File System Based on RAID Storage for Tapeless Storage

JUNGBIN KIM[1], HYEON-JIN YU[1], HYEONGBIN KANG[1], JAE-HYUCK SHIN[1], HEESEOK JEONG[2], AND SEO-YOUNG NOH[1]

[1]Department of Computer Science, Chungbuk National University, Cheongju 28644, South Korea
[2]Global Science Experimental Data Hub Center, Korea Institute of Science and Technology Information, Daejeon 34141, South Korea

Corresponding author: Seo-Young Noh (rsyoung@cbnu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2008-00458.

ABSTRACT Tape storage is used as a long-term data storage solution in institutions and research centers around the world, contributing to the advancement of business and scientific research. Despite the existence of other storage media, tape storage has gained popularity owing to its low cost per storage capacity, making it an affordable option for initial implementation. However, recently, the strengths of tape storage have become ambiguous, and their drawbacks have become more evident. In the current storage system, tape storage requires complex management compared to other storage media, which incurs additional human resource costs, making it less cost effective. On the other side, other storage media continue to become cheaper and have larger storage capacities. Therefore, companies and research centers are no longer required to continuously purchase tape storage. With this trend, distributed file systems can be an alternative to tape storage. Distributed file systems combine storage nodes into a network through Reliable Array of Independent Nodes (RAIN). Erasure coding is used to store data in this file system, which makes the system cost-effective and highly reliable. This allows the construction of a safe and efficient storage system. In this study, we constructed and evaluated a distributed file system based on Redundant Array of Independent Disks (RAID) storage. We configured and evaluated distributed file systems that use Reed-Solomon in the same environment, providing the performance characteristics of each distributed file system.

INDEX TERMS Tapeless storage, distributed file system, RAID, RAIN, performance, EOS, GlusterFS.

I. INTRODUCTION

Tape storage has been used as traditional storage for data backup and archiving [1]. Compared to other storage media, tape storage has lower initial costs and is energy efficient, resulting in very low operating costs. Therefore, many institutions that need to store large amounts of data have adopted tape storage as a cost-effective option [2]. However, recent technological advancements have highlighted the disadvantages of tape storage [3]. First, tape storage does not provide fast access speeds, such as hard disks or solid-state memory, making it unsuitable for the cloud era where data mobility is important. Additionally, restoring archived data from tape storage is very complex and time-consuming, often takes several days and results in higher personnel costs for management compared to other storage media. Despite these drawbacks, tape storage has been used because it is very cheap in terms of storage capacity compared to other media. However, as the price of disks continues to decrease, their advantages have become less clear. Although tape storage is still cheaper in terms of storage capacity, it cannot be considered cheap when considering personnel expenses required to manage it compared to other storage media. The readily available cost metrics often fail to highlight these drawbacks due to the overwhelmingly cost-effective nature of tape storage in terms of storage capacity. However, beyond these disadvantages, tape storage also harbors long-term issues. The current tape market has potential risks from single vendor

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Abid.

lock-in issues, so potential risks must be taken into account when introducing tape storage [4].

In this situation, companies and institutions no longer need to stick to tape storage when purchasing new data storage. Therefore, the trend in the storage industry is moving towards tapeless storage that is faster, more reliable, scalable, and accessible anywhere and anytime [3].

As a replacement for tape storage, which has lost its advantages, a distributed file storage system using cheaper disks is an alternative [5]. A distributed file system stores data on multiple computers at physically different locations. These nodes look like a local system by using a network. Distributed file systems have data backup capabilities through erasure coding like RAID storage and allow multiple disks or storage to be combined into one storage through a network. This is an attractive solution for large-scale data centers and institutions owing to its high scalability and reliability.

In practice, there are many available options for configuring distributed file system storage [6], [7], [8]. Adopting an appropriate distributed file system can have a significant impact on storage performance, scalability, and resource utilization. However, because each user has different available resources and service workloads, it is difficult to select a suitable distributed file system for each situation. To identify a suitable distributed file system for each user's situation, we conducted research on the performance characteristics of a RAID [9] storage-based distributed file system. Characterizing the performance of a distributed file system is not new; however, we focused on an intensive RAID storage environment because the observed performance can vary depending on the type of underlying storage.

For our research, we constructed a storage system by connecting RAID storage through a distributed file system. The RAID storage-based distributed file system leverages both RAID technology, which combines multiple disks into a single logical disk, and distributed file system technology that enables file sharing and management across multiple nodes through a network. This amalgamation yields superior data access speed, fault tolerance, and availability compared to traditional tape storage.

A RAID storage-based distributed file system has the following advantages. First, RAID provides redundancy by storing data on multiple disks, which helps to prevent data loss in the case of disk failure [10]. When the distributed file system and RAID are integrated together, data can be distributed across multiple cluster nodes, enhancing redundancy and fault tolerance. Second, RAID enables access to multiple disks in parallel, which provides improved performance by reducing disk I/O bottlenecks and overall increased system throughput. Integrating RAID into a distributed file system can support distributed data saving, parallelism increases, load balancing, and performance improvements. Third, in many cases, institutions and research facilities that are looking to adopt tapeless storage already possess RAID storage. From a cost perspective, introducing RAID storage is expensive. However, if you consider using existing RAID storage to create tapeless storage, there are no additional costs. In this study, we evaluated a distributed file system storage based on RAID storage to identify a suitable Distributed File System (DFS) for each work situation and provided performance characteristics for each DFS. DFSs were constructed using the same hardware configuration. We contribute to helping users choose a more suitable DFS for their environment by providing them with the performance characteristics of the DFS.

The main contributions of this study are as follows:

- Empirical evaluation of a distributed file storage system based on RAID storage.
- Analysis of factors affecting performance, such as workload, in an intensive RAID DFS storage environment.
- In-depth comparison and performance characterization of two DFSs based on RAID storage.

In the remainder of this paper, we cover the following topics. Section II presents the background and related work. Section III discusses the experimental design and implementation. Section IV presents experimental results and analysis. Finally, in Section V, we summarize and conclude the study.

II. BACKGROUND AND RELATED WORK

Tapeless storage is a data storage method that does not use a magnetic tape. With the advancements in SSD and HDD, it has become possible to create large-capacity storage systems that can store data quickly and reliably. Recently, owing to the rapid decline in HDD prices, it has become a viable alternative to tape storage. Compared to tape storage, tapeless storage is much faster and has higher reliability and flexibility, making it preferred as a data storage solution in many industries.

RAID storage-based distributed file system refers to a system that utilizes RAID technology as a foundation for performing data storage and management through a DFS. The primary features of a RAID storage based DFS are enhancing data reliability and availability, supporting efficient distributed storage and access of large-scale data. By employing RAID technology, it prevents data loss and provides fault-tolerant capabilities, while DFS technology facilitates the sharing and efficient management of data among multiple users and systems. Building upon these attributes, a RAID storage-based distributed file system is gaining attention as a tapeless storage solution that can replace conventional tape storage.

DFS is a file system that allows data to be stored and managed on different computers or nodes connected over a network. The purpose of DFS is to enable users of physically distributed computers to share data and storage resources by using a common file system [11]. Although not all DFSs do so, some provide high availability and high-level data protection through technologies such as RAIN and erasure coding, making them a commonly used solution for large-scale storage systems such as cloud storage.

RAIN [12] is a data storage architecture designed as a joint project between Caltech Parallel, Distributed Computing Group, and NASA's Jet Propulsion Laboratory. Developed to build a reliable distributed system using inexpensive off-the-shelf components, the RAIN architecture distributes data to each independent node to provide fault tolerance and availability through a network.

Erasure coding is a data protection technique that restores data using parity equations and is commonly used in data storage systems. In traditional storage systems, data replication is used to protect data by storing copies of the data in different locations. This method is simple and effective, but highly inefficient in terms of storage space. Erasure coding has been introduced to solve this problem [13]. Instead of simply replicating the original data, erasure coding generates parity data that contain information from the original data using mathematical algorithms to recover the lost data. Parity data are distributed and stored on each node, providing data recovery for limited damage even if random data nodes are corrupted. Erasure coding is much more space-efficient and faster than traditional data replication; therefore, it is widely used in most storage systems.

Reed-Solomon is a type of erasure coding that is commonly used and has a high-level resilience against data loss [14]. Although there are several types of algorithms for erasure coding, Reed-Solomon is highly suitable for data storage systems that require high reliability and data restoration capabilities. For this reason, most DFSs currently adopt the Reed-Solomon code to provide efficient data redundancy.

In our study, we adopted CERN EOS and GlusterFS, which use the Reed-Solomon algorithm among many DFSs for performance characteristic analysis.

CERN EOS [15] is a DFS developed to provide stable storage technology for Large Hadron Collider (LHC) experiments. The EOS consists of three essential daemons: the Management Server (MGM), which performs metadata management, batch and access scheduling, and pool configuration management; the File Storage Server (FST), which stores files; and the Message Queue (MQ), which asynchronously delivers messages between the two daemons. The EOS creates a network RAID storage system that supports erasure coding through a RAIN layout. It combines storage over the network, stores data through erasure coding, and enables cheap, efficient, and reliable storage creation.

GlusterFS [16] is a DFS designed to provide scalable and highly available storage solutions for large-scale distributed storage configurations. This provides a unified namespace that allows the clustering of multiple servers over a network to store and access data. GlusterFS can be used with off-the-shelf hardware and has built-in fault tolerance using erasure coding, as well as support for automatic load balancing. It is a powerful and flexible DFS that is suitable for environments such as cloud and big data, which require scalability, availability, and fault tolerance.

There are various studies that have been conducted on DFSs.

Lee et al. [17] implemented disk-based DFSs for performance evaluation in scientific big-data environments. They mounted file systems using FUSE clients and identified the characteristics of each file system through equivalent evaluations.

Zhang et al. [18] evaluated the performance of Ceph in an OpenStack cloud environment by using well-known benchmark tools. They confirmed the excellence and scalability of Ceph using the Bonnie++, RADOS, and Iperf benchmark tools.

Donvito et al. [19] set up and tested several Disk-based DFSs for HEP experimental analysis. They conducted performance evaluations using various benchmarks. The DFSs used for evaluation were the HDFS, GlusterFS, and CephFS. The benchmark tools used were iozone and dd.

Leite and Solis [20] configured a hyperconverged system using a GlusterFS and a VMware ESXi hypervisor. By hosting Docker containers, they evaluated the performance of the configured system compared to the traditional method, where data are stored directly on the server's local disk. The results demonstrate the superiority of the configured system in handling large volumes of write operations.

Adde et al. [21] describe the open-source distributed file system EOS developed and used at CERN. EOS provides low latency, high availability, strong authentication, multiple replication schemas, and multiple access protocols and features. The paper provides a brief overview of EOS's architecture and introduces its main and latest features.

Purandare et al. [22] investigated the workload characteristics of CERN EOS DFS. By analyzing and observing trends in over 450PB of read and write operation events, they recommended a standardized set of tracing collection and analysis tools that can aid researchers in designing systems.

Riasetiawan and Amien [23] analyzed various data storage models such as GlusterFS and MooseFS to solve the high latency problem in big data storage methods. They conducted experiments on parameters such as transfer rate, IOPS, and CPU load to identify the segments in which each model performs best, and confirmed that the performance of the models is proportional to the size of the block.

Selvaganesan and Liazudeen [24] propose GlusterFS as an easy solution for efficient data storage. They point out that some existing distributed file systems have excellent scalability and low cost, but are difficult to deploy and maintain. They explain that GlusterFS can solve this problem. By describing the main architecture structure of GlusterFS, they provide readers with insight into GlusterFS.

Macko and Hennessey [25] describe the various choices that should be considered when designing a distributed file system. Using several existing distributed file systems as examples, the paper introduces the most commonly used choices in each aspect of distributed file systems. This provides an introductory guide to distributed file systems.

Ahn et al. [4] discussed the potential issues with tape storage and proposed alternatives. They suggested a disk-based archival storage solution with low error rates and high

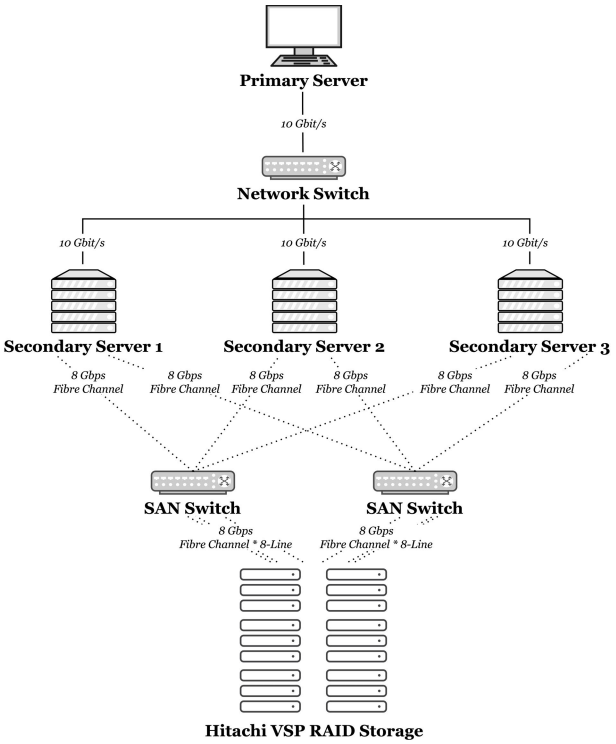


FIGURE 1. Experimental network configuration.

reliability by configuring the CERN EOS DFS based on Just-Bunch-Of-Disks (JBOD).

Through related research, we have addressed the issues with tape storage. We explored alternative solutions that could replace tape storage and evaluated distributed file systems based on RAID storage as an alternative. Our objective was to provide users with the I/O characteristics of each DFS through equivalent comparisons. Therefore, we mounted the systems in the same user space using FUSE clients in each file system and measured their performance using the FIO benchmark tool.

III. EXPERIMENTAL DESIGN AND IMPLEMENTION

We configured a DFS storage cluster using RAID storage, as shown in Figure 1, to verify the performance chracteristics of each DFS. We set up the same four physical servers in a primary-secondary structure and mounted the RAID storage volume on each secondary server. The RAID volumes mounted on the secondary servers were configured as a single storage unit through the DFS. Subsequently, we mounted the configured storage on the primary server system using a FUSE [26] client and measured the performance using the FIO benchmark tool. This measurement method enabled a fair performance comparison of different DFSs under the same conditions. A notable aspect of this experimental setup is that the network bandwidth of the primary server is limited to 10Gbit/s. Consequently, in the bandwidth experiment results, the DFS bandwidth performance could not exceed the primary server's Iperf [27] measurement value of 1175 MB/s.

TABLE 1. Server specification.

Server Specification	
OS	CentOS 7
CPU	Intel(R) Xeon(R) CPU E5-2680 2.70GHz
RAM	96GB
Disk	600GB HDD(Boot), 2.7 TB RAID Volume * 2
Network	10Gbit/s

TABLE 2. RAID storage specification.

Hitachi VSP RAID Storage	
Disk	1.2 TB SAS HDD (Model: DKR5E-J1R2SS)
RAID	RAID6 (6D + 2P)
Network	8 Gbps Fiber Channel

TABLE 3. Distributed file system layout.

DFS Layout	Distributed	Replication	RAID6 RAIN
EOS	Plain	Replica	RAID6, stripe=6
GlusterFS	Distributed Layout	Replicated	Dispersed Volume, k=4, m=2

Table 1 lists the server specifications used in the experiments. CentOS 7.9 was installed and used identically on all servers. Each server has one physically mounted boot disk. For the actual storage configuration, two RAID storage volumes are mounted on each server. The final configured DFS storage consists of six 2.7 TB RAID volumes, totaling 16.2 TB in size.

Table 2 lists the specifications of the RAID storage used in the experiment. We used the Hitachi VSP RAID Storage and configured RAID 6 with 1.2 TB SAS HDDs. The detailed RAID configuration was set to six data and two parity drives. Each server was connected to the storage with an 8Gbps FC 2-Line connection.

In DFSs, the layout refers to the manner in which data are stored in the system. Depending on the layout, the structure and arrangement of files and directories and the data allocation and placement across multiple servers and storage devices on the network are determined [28], [29]. DFSs can flexibly apply data protection as needed using these layouts. In this study, we evaluated three layouts for each DFS. Table 3 lists the options used for each DFS layout in this experiment.

Distributed layout is a method that stores data linearly across all storage devices. This method utilizes all storage space, but does not provide data redundancy, which entails the potential for data loss. In the EOS, the Plain option was used for the layout, whereas the Distributed Layout option was applied in GlusterFS.

TABLE 4. FIO benchmark option.

FIO Benchmark Option	
Block Size	4K, 16K, 32K, 64K, 128K, 256K, 512K, 1024K
Work Size	6GB
Num of Thread	1, 2, 4, 8, 16, 32, 64, 128
I/O Depth	32
I/O Type	Read, Write, Rand Read, Rand Write
Runtime	180 sec

Replication layout stores data copies on all the storage nodes. This method can restore data with up to N-1 node failures but reduces storage space utilization to 1/N. In the EOS, the Replica option was used, and the Replicated option was applied in the GlusterFS.

RAID 6 RAIN layout uses erasure coding and parity data for data redundancy. Similar to conventional RAID 6, it guarantees data recovery for a certain level of node failure. Unlike physical RAID, all connections are made through the network, and parity data can be flexibly applied. In the experiment, the EOS was designed with RAID6 and stripe = 6 options, whereas GlusterFS was designed with the Dispersed Volume, k = 4, and m = 2 options, allowing for resilience against up to two node failures. The versions of each DFS used were EOS 4.7.1 and GlusterFS 9.6.

Table 4 lists the options used for the FIO [30] benchmark tool in performance evaluation. The test was designed to measure eight block sizes, allowing the observation of performance trends with varying block sizes. A single work size was set to 6GB, assuming that one data-set was stored at 6GB. Measurements were taken for eight thread entries, allowing for an examination of latency and performance in relation to varying workload levels. Four different types of I/O patterns were measured, allowing the examination of the results in each I/O pattern. The FIO version 3.7 was used in this study.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we describe the experimental results and analysis. The results are grouped according to I/O patterns, and we provide an overall bandwidth results graph as well as trend graphs for bandwidth, IOPS, and latency. Bandwidth refers to the amount of data that can be transferred through a network or storage system within a given timeframe. In this study, denotes the data transfer rate per second in MB/s. IOPS represents the number of read and write operations that a storage system can perform in one second. Latency refers to the time required for an I/O operation to be completed. In this study, it denotes the latency in ns (nanoseconds). All three metrics are important for understanding the storage performance. Bandwidth and IOPS show similar trends, as they are influenced by similar factors, such as the speed of the storage medium and the efficiency of the data transfer protocol. In general, when a high workload is applied, the latency increases, but the bandwidth and IOPS remain the same or even decrease.

The results of the experiments depict performance metrics of DFS in each aspect. As the main proposition of the paper is to present the RAID storage-based DFS system as an alternative to tape storage, the significant indicators in tape storage are also of importance in the RAID storage-based DFS system. In typical tape storage scenarios, sequential read operations, performed to retrieve backup data or archived information, are the most frequent tasks. Following that, sequential write operations, used to record new data onto tape storage, occur with the next highest frequency. Random read operations, involving retrieving data recorded at arbitrary locations, are less frequent than the preceding tasks. Finally, random write operations, which involve recording data at random locations, are the least frequent operations conducted. These frequency patterns are significant in the context of typical tape storage usage. However, when constructing large-scale storage systems for organizations or enterprises, each entity possesses distinct workloads, requirements, and characteristics. Therefore, for the configuration of such systems, it is advisable to individually assess the specific demands and workloads to ensure optimal system setup.

A. READ RESULTS

The following presents the experimental results for the read benchmarks according to the layout. In the read benchmarks, the DFSs generally demonstrated consistent performance. We can observe that most items reach the bandwidth limit.

1) RAID 6 RAIN LAYOUT READ

Figure 2 displays the overall bandwidth results for the RAID6 layout read benchmarks of the EOS and GlusterFS. In the 1, 2, and 4 thread items, GlusterFS generally exhibits superior performance compared to EOS. In items with 8 threads or more, the EOS consistently outperformed GlusterFS because of the improved throughput increase as the block size increased. Graphs for items after 8 threads exhibit the same pattern, implying EOS performance saturation from 8 threads onwards. GlusterFS shows little difference in bandwidth from 1 to 128 threads.

Figure 3 illustrates the trends in bandwidth, IOPS, and latency for the RAID6 layout read benchmarks of EOS and GlusterFS. As shown in Figure 2, the EOS graph demonstrates an increasing trend in bandwidth as the workload increases. In contrast, GlusterFS consistently displays a stable performance regardless of the number of threads. In terms of latency, both DFSs experienced a sharp increase after 8 threads. Notably, GlusterFS exhibited substantially higher latency than EOS at 64 and 128 threads.

In summary, adopting EOS for the RAID6 layout read may be a more advantageous choice in terms of performance when expecting massive workloads. In contrast, GlusterFS appears to be more favorable if consistent performance is desired regardless of the situation.

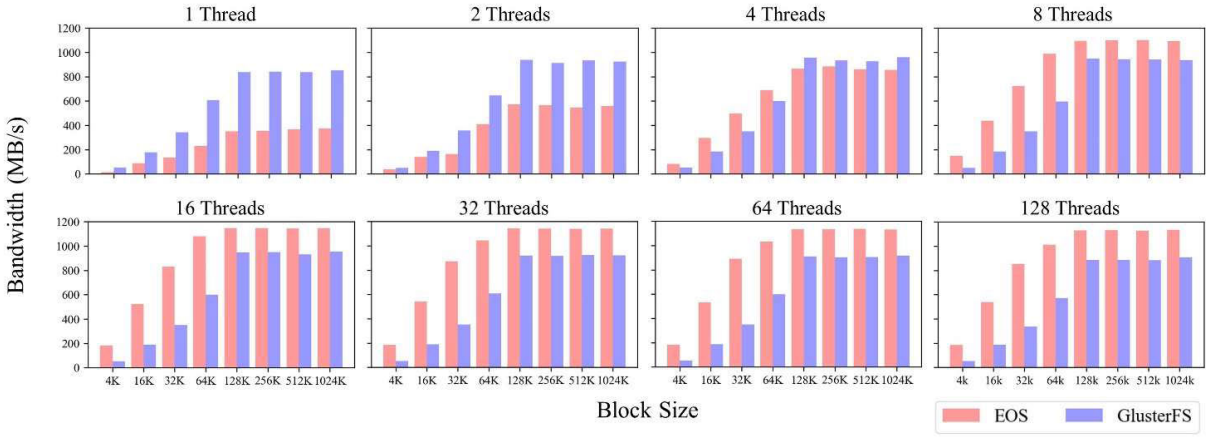


FIGURE 2. The overall bandwidth results for the RAID6 layout Read benchmarks of EOS and GlusterFS.

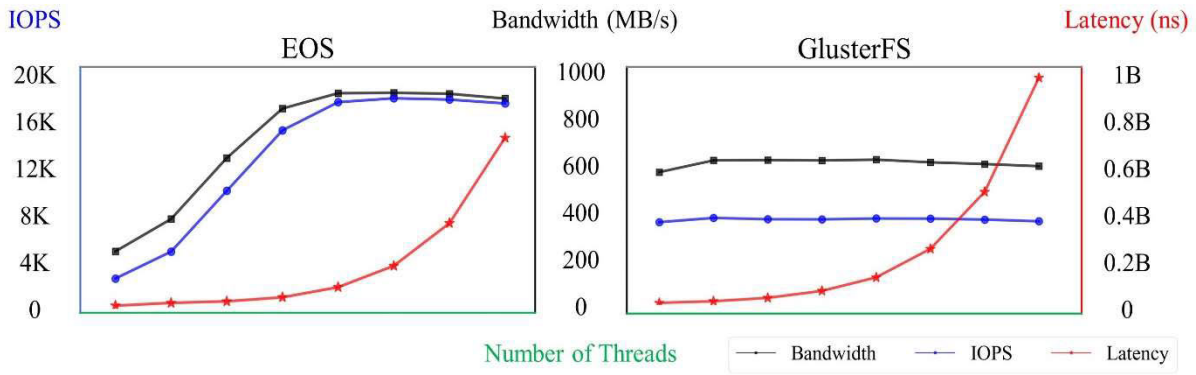


FIGURE 3. Bandwidth, IOPS and latency for the RAID6 layout Read benchmarks of EOS and GlusterFS.

2) REPLICATION LAYOUT READ

Figure 4 presents the overall bandwidth results for the replication layout read benchmarks of EOS and GlusterFS. At 1, 2, and 4 threads, GlusterFS exhibited better performance than EOS. From 8 threads onwards, EOS outperforms GlusterFS in items with smaller block sizes, whereas GlusterFS excels in those with larger block sizes. The graph reveals that the EOS approaches its maximum performance at 16 threads, whereas GlusterFS does so at 2 threads. While EOS demonstrates incremental performance improvement from 1 to 16 threads, GlusterFS experiences a sharp performance enhancement at 2 threads, reaching its maximum value with no significant bandwidth changes observed up to 128 threads.

Figure 5 displays the trends of bandwidth, IOPS, and latency for the replication layout read benchmarks of EOS and GlusterFS. Overall, GlusterFS was found to be the higher-performing option for items with a smaller number of threads. The EOS is marginally better for 16 to 128 thread items. In particular, EOS outperformed GlusterFS, especially for items with smaller block sizes at higher thread counts.

3) DISTRIBUTED LAYOUT READ

Figure 6 presents the overall bandwidth results for the distributed layout read benchmarks of EOS and GlusterFS. In the 1, 2, and 4 thread items, GlusterFS significantly outperformed the EOS. Although EOS gains a slight advantage in the 8-thread 16 K block size item, GlusterFS demonstrates a higher performance in the remaining thread and block size items.

Figure 7 displays the trends in bandwidth, IOPS, and latency for the distributed layout read benchmarks of EOS and GlusterFS. GlusterFS is observed to have a performance advantage for nearly all evaluation items. In contrast to EOS, which exhibits a bandwidth of 200MB/s for 1 thread, GlusterFS shows a bandwidth of 600MB/s for 1 thread. While EOS approaches GlusterFS from the 8-thread items onwards, GlusterFS maintains a slight advantage in the remaining thread items. No significant differences in latency were observed; thus, GlusterFS appeared to be generally superior in distributed read items.

B. WRITE RESULTS

The following section presents the experimental results for layout-specific write benchmarks. In write benchmarks, clear maximum bandwidth differences are observed across layouts, resulting from the distinct ways in which each layout stores data. Notably, in this section, the RAID 6 RAIN layout does not exceed 800MB/s, replication does not surpass 200MB/s, and the distributed does not go beyond 1200MB/s. This is

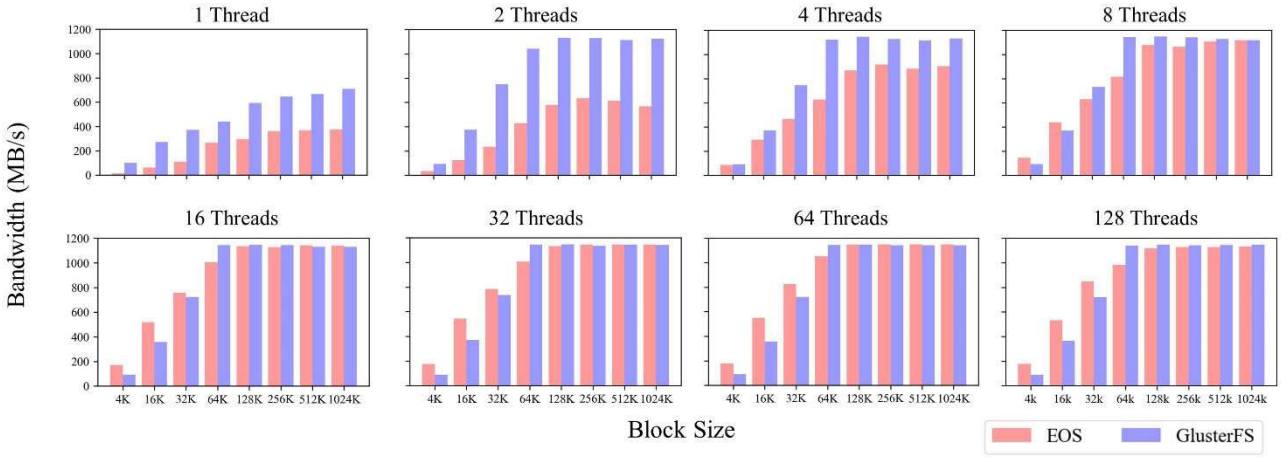


FIGURE 4. The overall bandwidth results for the replication layout Read benchmarks of EOS and GlusterFS.

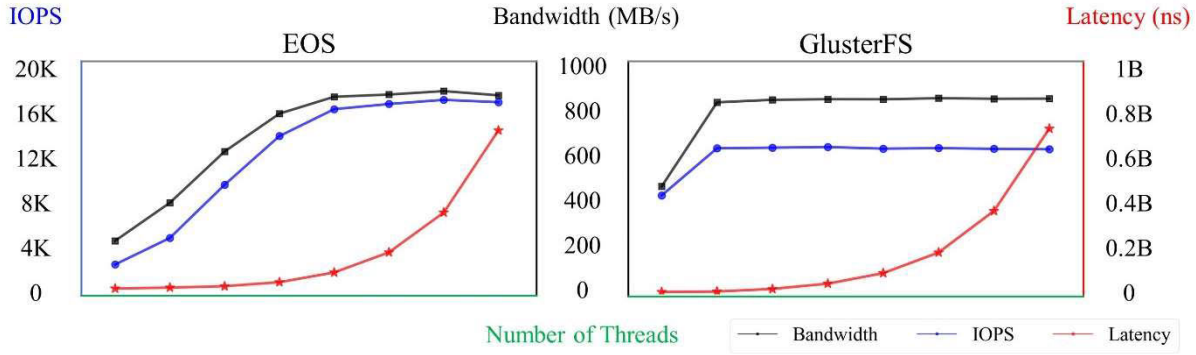


FIGURE 5. Bandwidth, IOPS and latency for the replication layout Read benchmarks of EOS and GlusterFS.

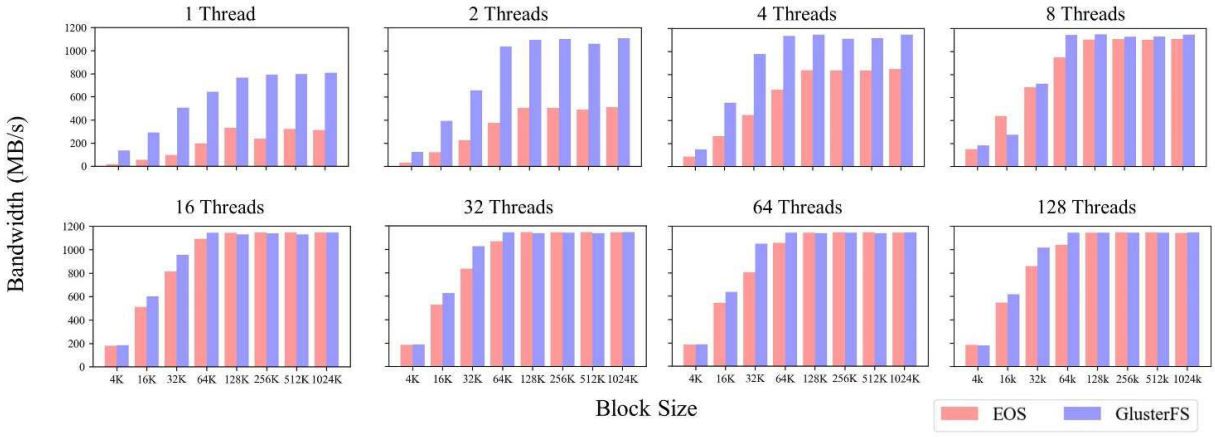


FIGURE 6. The overall bandwidth results for the Distributed layout Read benchmarks of EOS and GlusterFS.

because the write benchmark bandwidth measurement only counts the original data; For RAID 6 RAIN with four data nodes and two parity nodes, only 4/6 of 1200MB/s as the maximum bandwidth. Thus, this results in 800MB/s. In the case of replication with one data node and five replica nodes, the maximum bandwidth is 1/6 of 1200MB/s, which is 200MB/s. However, graph values should be carefully checked adjusting the graph range according to the mentioned bandwidth decrease.

1) RAID 6 RAIN LAYOUT WRITE

Figure 8 presents the overall bandwidth results for the RAID 6 RAIN layout write benchmarks of the EOS and GlusterFS. As previously mentioned, because the writing of

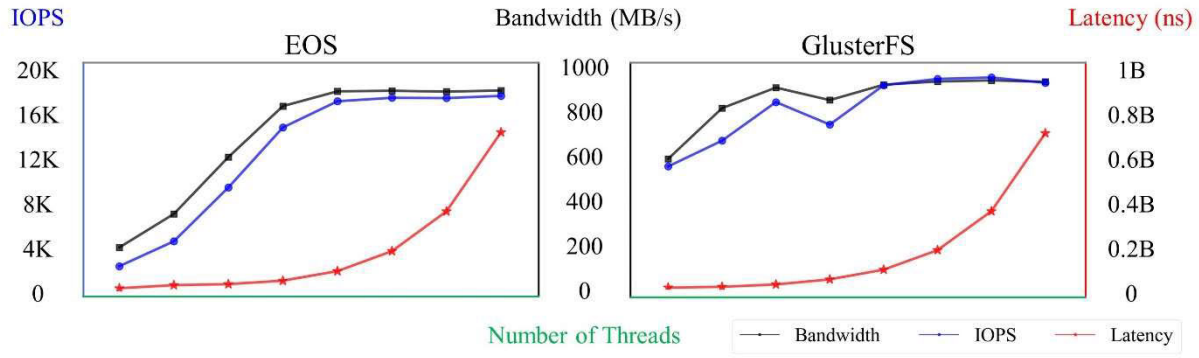


FIGURE 7. Bandwidth, IOPS and latency for the distributed layout read benchmarks of EOS and GlusterFS.

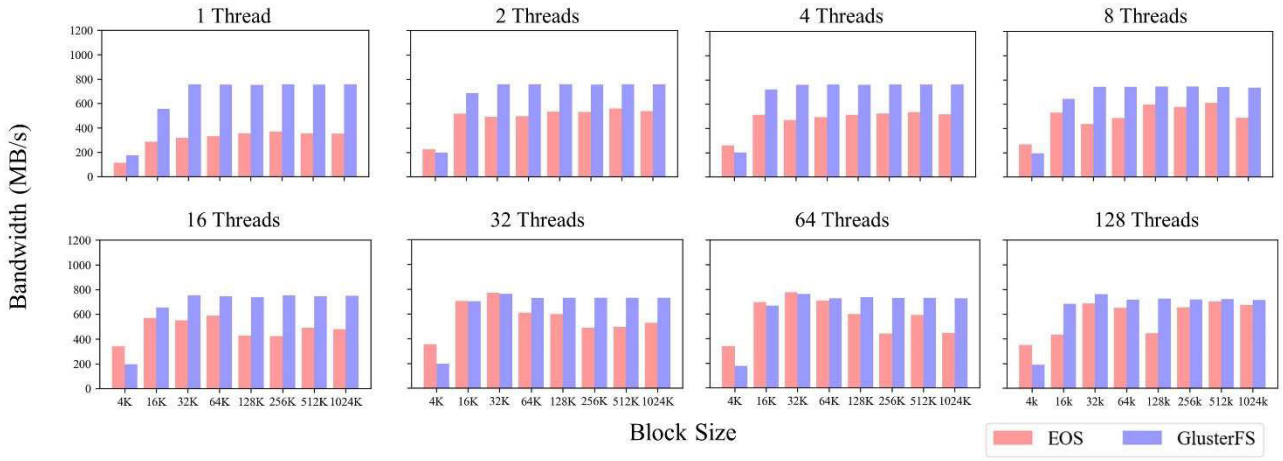


FIGURE 8. The overall bandwidth results for the RAID 6 RAIN layout Write benchmarks of EOS and GlusterFS.

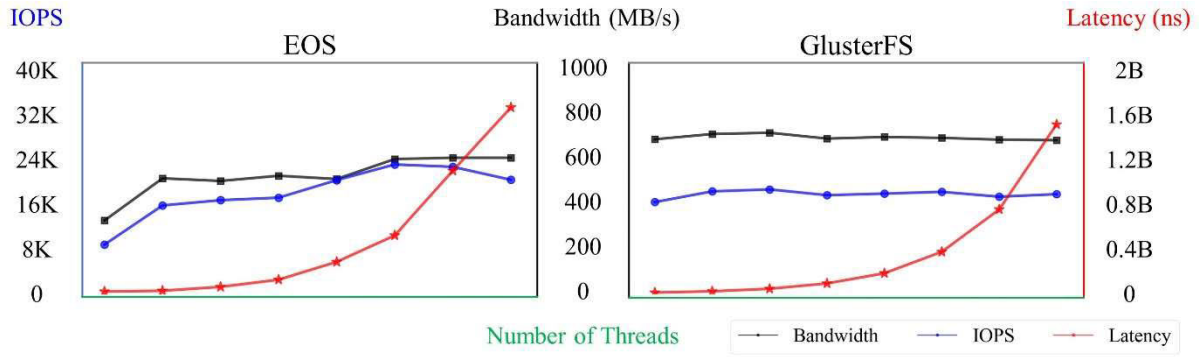


FIGURE 9. Bandwidth, IOPS and latency for the RAID 6 RAIN layout write benchmarks of EOS and GlusterFS.

parity data was not measured, the bandwidth performance was shown to be up to 800MB/s. The performance of GlusterFS saturates at a 32 K block size, maintaining or decreasing in the subsequent block sizes. In contrast, the EOS's performance saturates at a 16 K block size, more frequently showing a decreasing trend in the following block sizes rather than maintaining performance. Except for the 1-thread item, the 4 K block size bandwidth exhibits a higher performance in EOS than GlusterFS in the remaining thread items. However, GlusterFS outperformed EOS in most other items.

Figure 9 displays the trends of bandwidth, IOPS, and latency for the RAID 6 RAIN layout write benchmarks of EOS and GlusterFS. Similar to the Read items, GlusterFS demonstrates a nearly constant performance regardless of the thread count. For the EOS, it is still evident that the throughput increases as the number of threads increases. Considering the results of all items, the EOS performs better than GlusterFS for items with a large number of threads and small block size workloads. However, the GlusterFS exhibited superior performance for the remaining items.

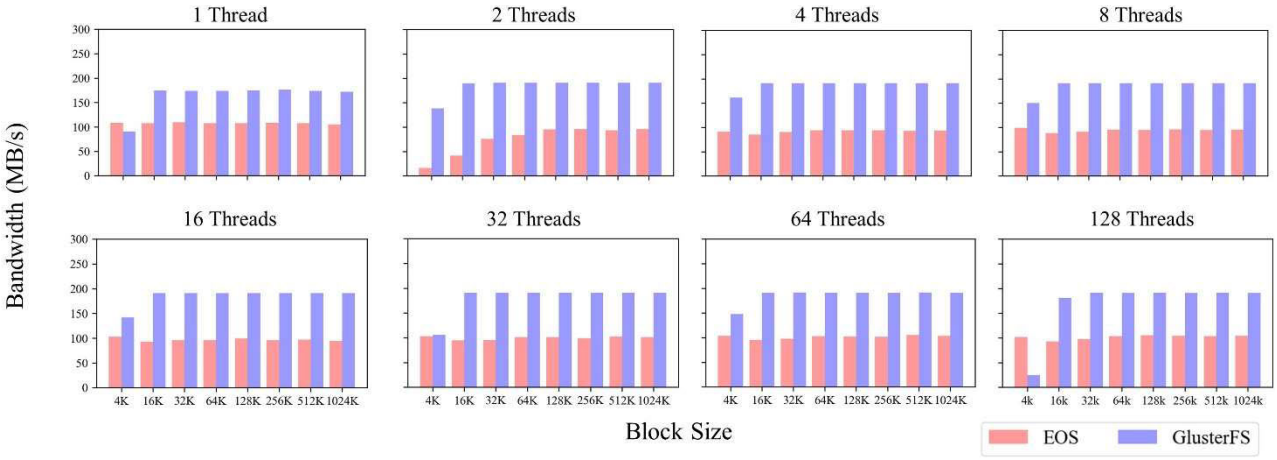


FIGURE 10. The overall bandwidth results for the replication layout write benchmarks of EOS and GlusterFS.

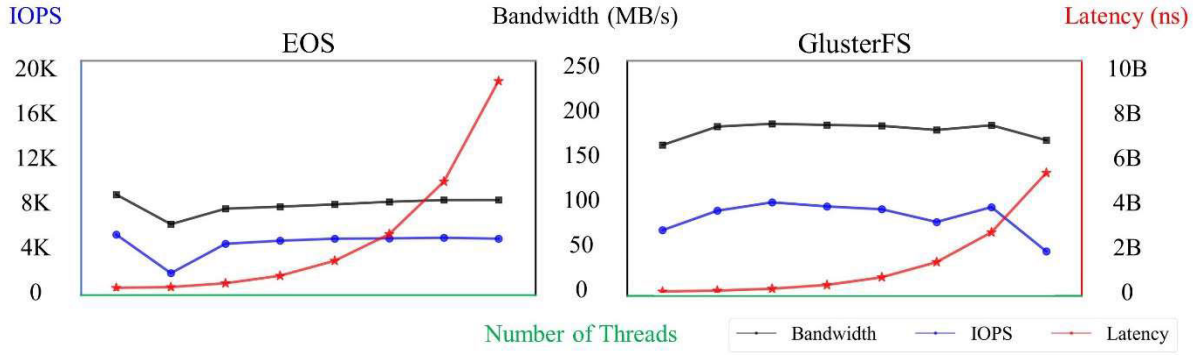


FIGURE 11. Bandwidth, IOPS and latency for the replication layout write benchmarks of EOS and GlusterFS.

2) REPLICATION LAYOUT WRITE

Figure 10 presents the overall bandwidth results for the replication layout write benchmarks of the EOS and GlusterFS. In the Replication Write, the bandwidth is measured at a sixth of the other item's levels because the data replication to the five replica nodes is not counted.

Overall, GlusterFS demonstrated a higher performance than EOS. For all items except for the 1-thread and 128-thread 4 K block sizes, GlusterFS appears to outperform EOS.

Figure 11 displays the trends of bandwidth, IOPS, and latency for the replication layout write benchmarks of EOS and GlusterFS. As GlusterFS experiences a sharp decrease in IOPS when it transitions from 64 threads to 128 threads and severe bandwidth degradation in the 4 K item, it implies that the performance degradation is due to an excessive workload. EOS exhibits consistent performance from 4 threads to 128 threads; however, the measured latency is considerably higher than that of GlusterFS, making it difficult to describe EOS as more stable than GlusterFS.

3) DISTRIBUTED LAYOUT WRITE

Figure 12 shows the overall bandwidth results of the distributed layout write benchmark for EOS and GlusterFS. In the case of the distributed layout, because each node stores only one data, there is no maximum bandwidth reduction in write unlike the RAID 6 RAIN layout and replication layout. For all block size items with 1 thread, GlusterFS showed better performance than EOS. From 2 threads to 128 threads, EOS shows a higher performance than GlusterFS in the 4 K and 16 K block size items. EOS and GlusterFS showed similar performances for the remaining block size items.

Figure 13 shows the trends of write bandwidth, IOPS, and latency for the distributed layout of the EOS and GlusterFS. Overall, GlusterFS still shows consistently good performance regardless of the number of threads, whereas EOS shows an increasing throughput trend with an increase in the number of threads. The latency of the two DFS was measured similarly, and as the number of threads increased and the block size decreased, EOS showed higher performance than GlusterFS; therefore, if a large amount of work is expected, EOS is a good choice. If a good performance is expected at low thread counts, GlusterFS may be a better option.

C. RAND READ RESULTS

The following are the experimental results of the random read benchmark for each layout. Random reads are similar to read operations; however, they involve accessing data randomly

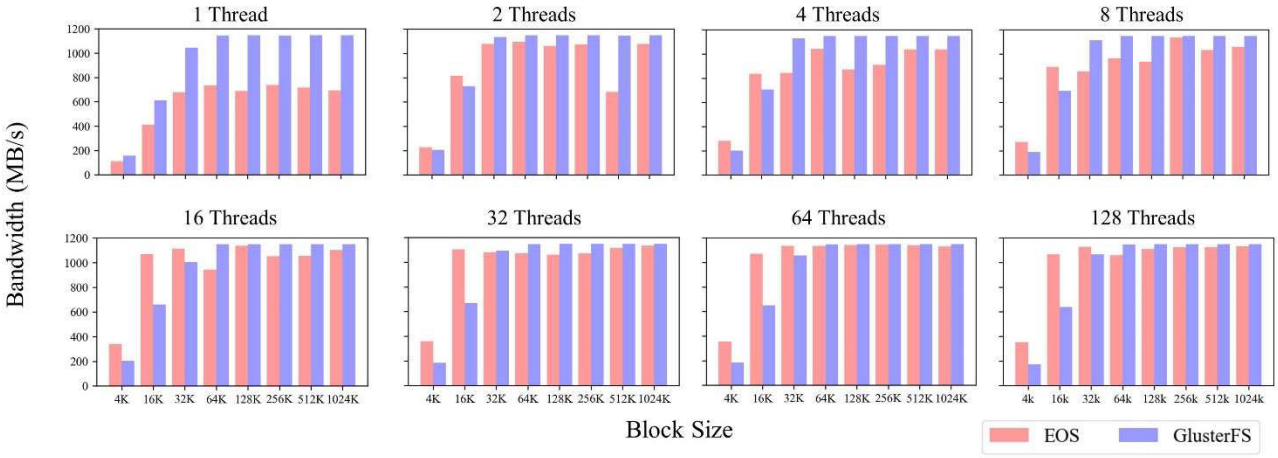


FIGURE 12. The overall bandwidth results for the distributed layout write benchmarks of EOS and GlusterFS.

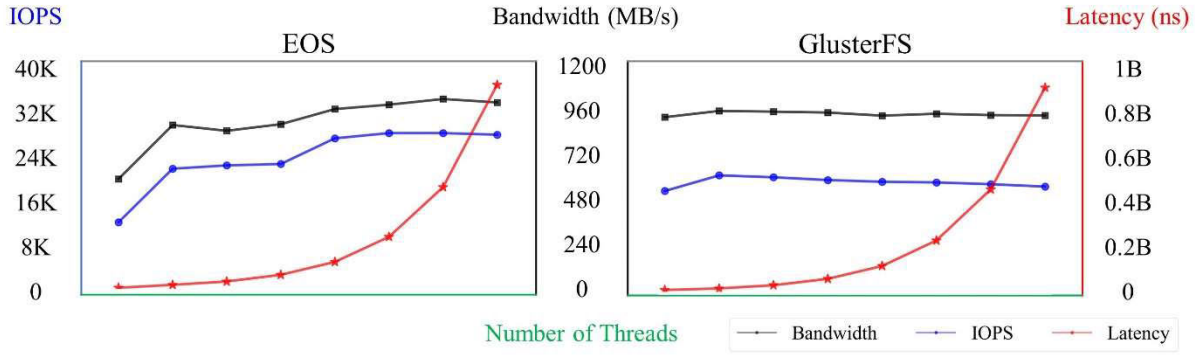


FIGURE 13. Bandwidth, IOPS and latency for the distributed layout write benchmarks of EOS and GlusterFS.

from different locations in the storage device. Therefore, read requests are not sequential, and more time is required for processing tasks. Overall, the bandwidth results showed an uneven trend. In particular, a very low bandwidth is measured in the 4 K item because if the data size is small, the storage device spends more time finding the read head and specifying the location than actually reading the data.

When looking at the results of the entire item, GlusterFS showed much better performance than EOS in Rand Read.

1) RAID 6 RAIN LAYOUT RAND READ

Figure 14 shows the overall bandwidth results of the RAID 6 RAIN layout rand read benchmark for the EOS and GlusterFS. The EOS showed an increase in throughput with an increase in block size for all thread items. GlusterFS shows somewhat uneven results, increasing the performance up to 128 K block size and then decreasing in subsequent block sizes. As the number of threads increases, GlusterFS exhibits bandwidth performance proportional to the block size. From 32 to 128 threads, The EOS exhibited very low performance.

Figure 15 shows the trends of the RAID 6 RAIN layout rand read bandwidth, IOPS, and latency for EOS and GlusterFS. The EOS shows saturation in performance at 8 threads, followed by a sharp decrease in bandwidth and a sharp increase in latency. GlusterFS exhibits a high bandwidth at 1 thread and then gradually decreases in performance. Compared with EOS, GlusterFS appears to have a less significant latency increase with an increase in thread count. Overall, GlusterFS was considered a better choice in this category.

2) REPLICATION LAYOUT RAND READ

Figure 16 displays the overall bandwidth results of the replication layout rand read benchmark for EOS and GlusterFS. The EOS exhibits highly uneven performance for each block size item, whereas GlusterFS shows a somewhat stable performance. Except for the 4 K block item with 4 threads, GlusterFS outperformed EOS for all items. Similar to RAID 6 RAIN Layout Rand Read, GlusterFS shows an uneven increase in bandwidth with block size at low thread counts, which becomes more uniform with an increase in the number of threads.

Figure 17 shows the trends of the replication layout rand read bandwidth, IOPS, and latency for the EOS and GlusterFS. EOS shows an increase in throughput of up to

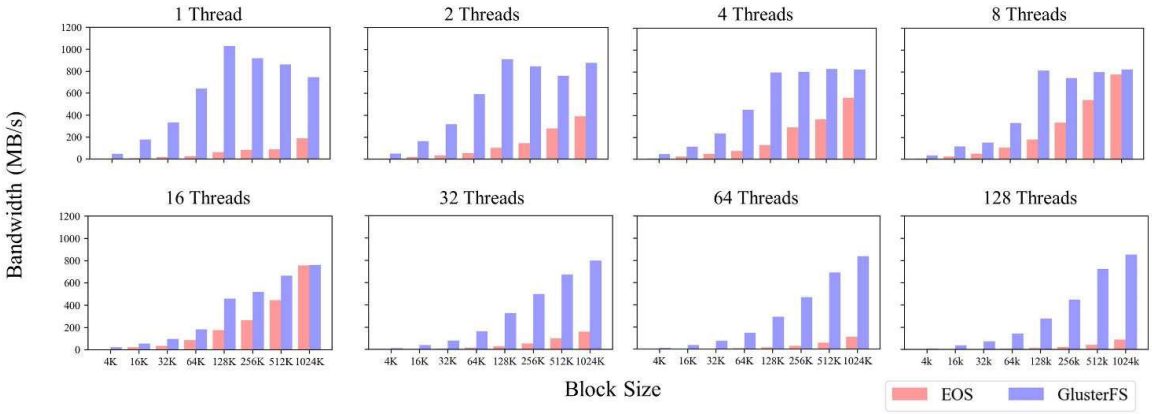


FIGURE 14. The overall bandwidth results for the RAID 6 RAIN layout rand read benchmarks of EOS and GlusterFS.

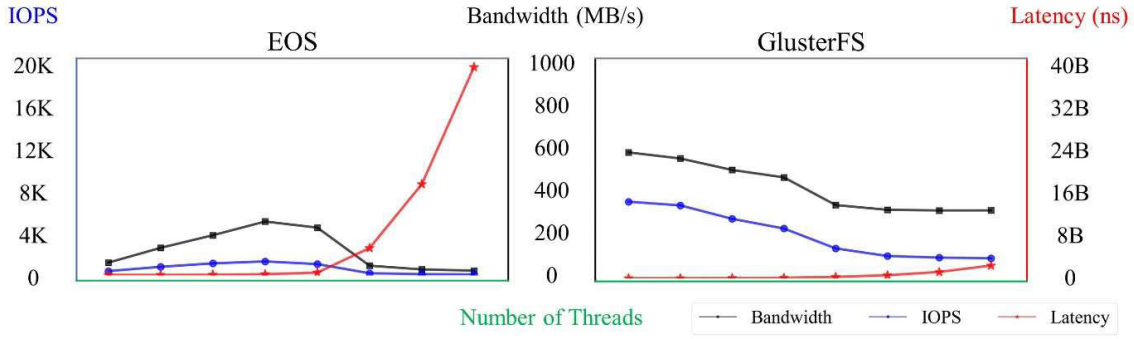


FIGURE 15. Bandwidth, IOPS and latency for the RAID 6 RAIN layout rand read benchmarks of EOS and GlusterFS.

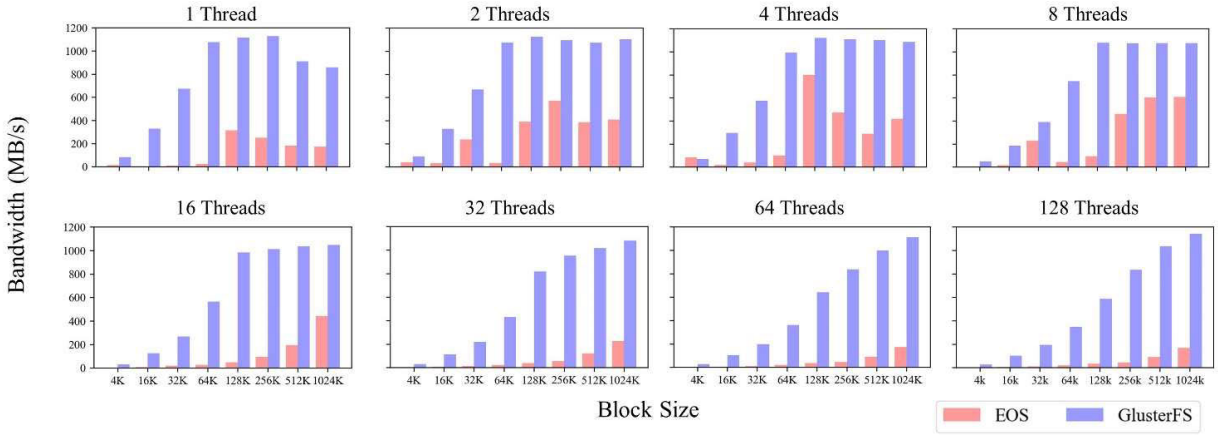


FIGURE 16. The overall bandwidth results for the replication layout rand read benchmarks of EOS and GlusterFS.

4 threads, but a sharp decrease in throughput occurs after 8 threads. Very low bandwidth was measured after 16 threads. It showed a significantly higher latency than GlusterFS for high thread counts. GlusterFS shows a decrease in throughput after 2 threads, but overall, it performs better than the EOS for all threads.

3) DISTRIBUTED LAYOUT RAND READ

Figure 18 displays the overall bandwidth results of the distributed layout rand read benchmark for EOS and GlusterFS. The EOS exhibits extremely low performance, which is difficult to observe in the graph, in the 4 K block size for all thread items. EOS shows a higher performance than GlusterFS at 8 threads and 16 threads, but its performance drops sharply after 32 threads. GlusterFS exhibits relatively consistent performance depending on the block size.

Figure 19 shows the trends of distributed layout rand read bandwidth, IOPS, and latency for the EOS and GlusterFS. The EOS shows an increase in throughput of up to 16 threads, followed by a sharp decrease in performance at 32 threads.

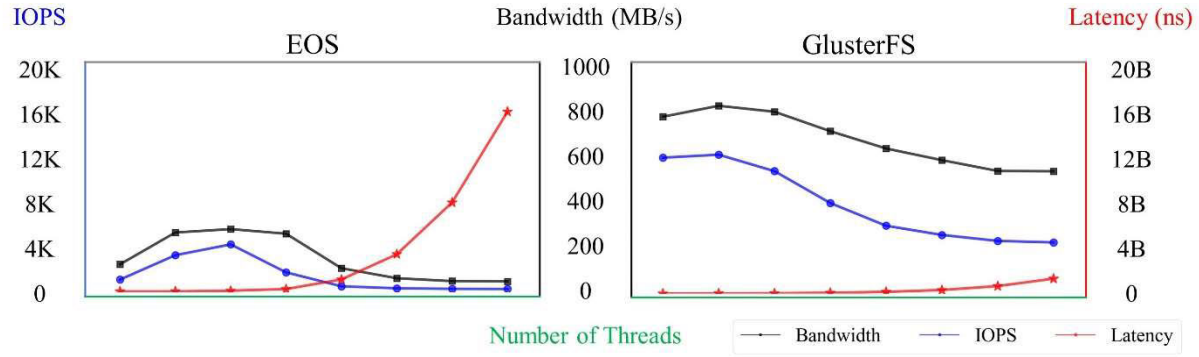


FIGURE 17. Bandwidth, IOPS and latency for the replication layout rand read benchmarks of EOS and GlusterFS.

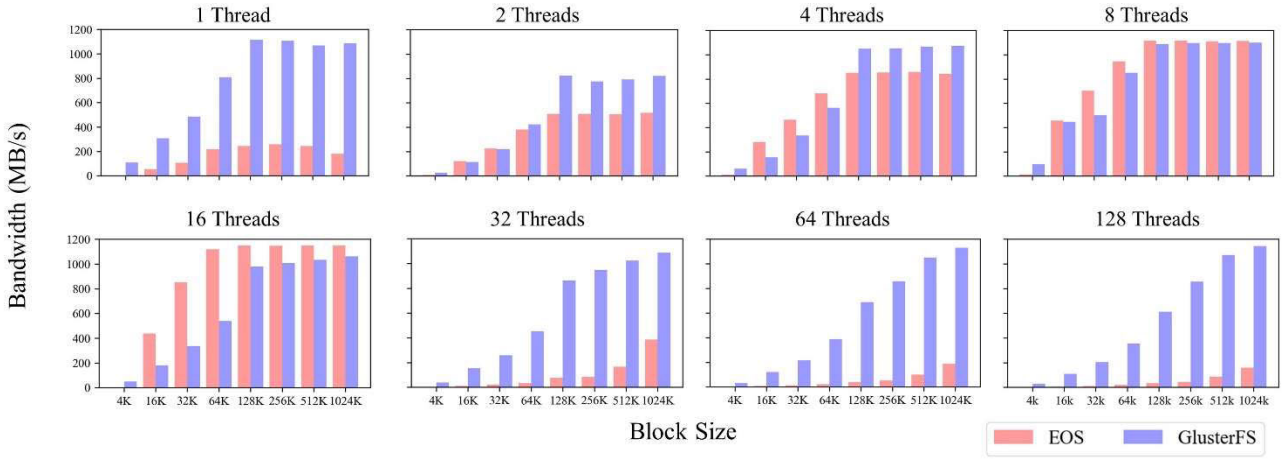


FIGURE 18. The overall bandwidth results for the distributed layout rand read benchmarks of EOS and GlusterFS.

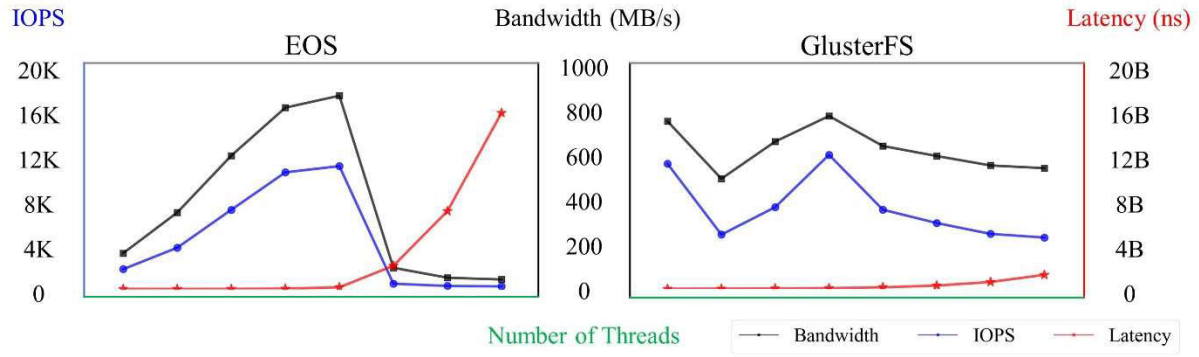


FIGURE 19. Bandwidth, IOPS and latency for the distributed layout rand read benchmarks of EOS and GlusterFS.

GlusterFS shows a varying trend in throughput depending on the number of threads; however, overall, it exhibits a stable throughput.

D. RAND WRITE RESULTS

The following is the experimental result for layout-specific random write benchmarks. Rand Write is an I/O type in which data are written randomly to different locations on the storage device, rather than sequentially to a specific location. As with write benchmarks, it can be observed that the bandwidth range measured by layout varies. In comparison to other measurement items, a bandwidth of approximately 6/4 was measured for the RAID 6 RAIN layout, and approximately 6/1 for replication. It should be noted that, unlike write, because it is recorded in random order, there are items that exceed the maximum of 800MB/s and 200MB/s in Write. In the Rand Write item, EOS generally performed better at high thread counts, whereas GlusterFS exhibited higher performance at low thread counts.

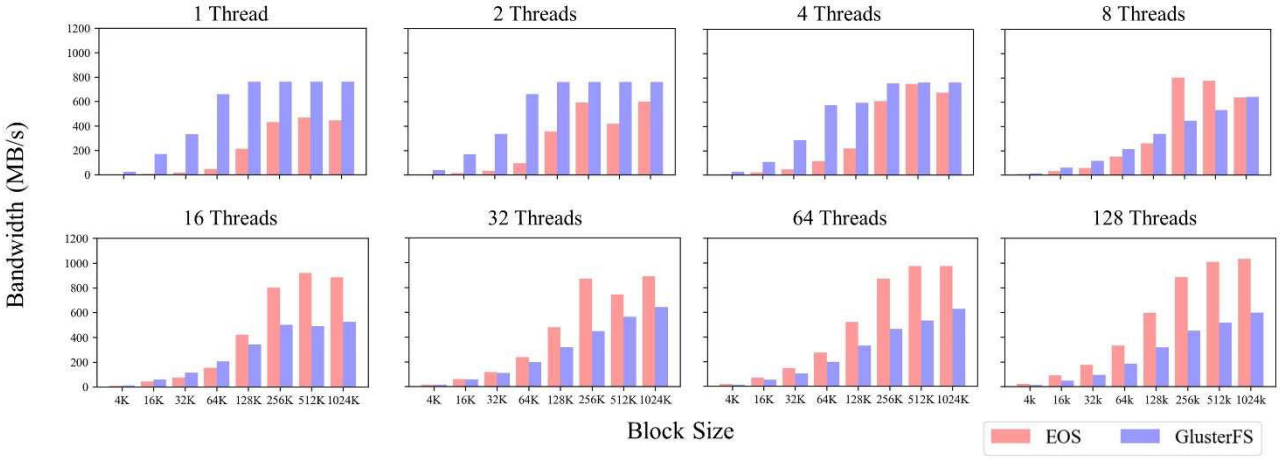


FIGURE 20. The overall bandwidth results for the RAID 6 RAIN layout Rand Write benchmarks of EOS and GlusterFS.

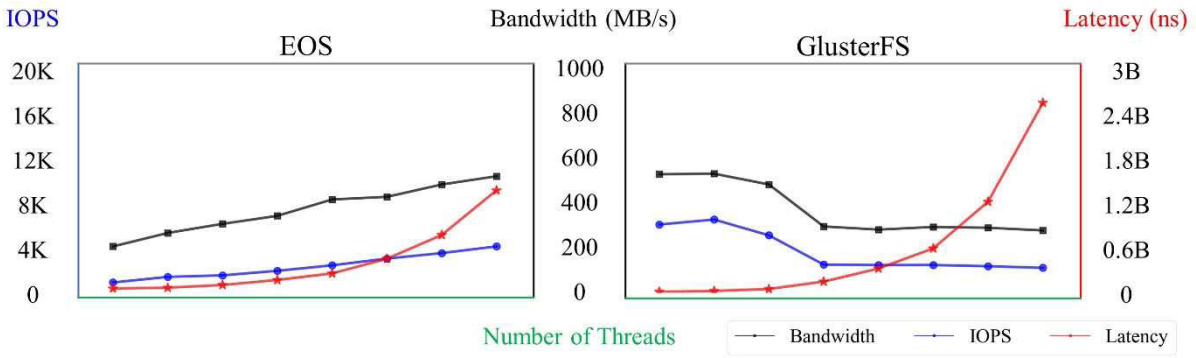


FIGURE 21. Bandwidth, IOPS and latency for the RAID 6 RAIN layout Rand Write benchmarks of EOS and GlusterFS.

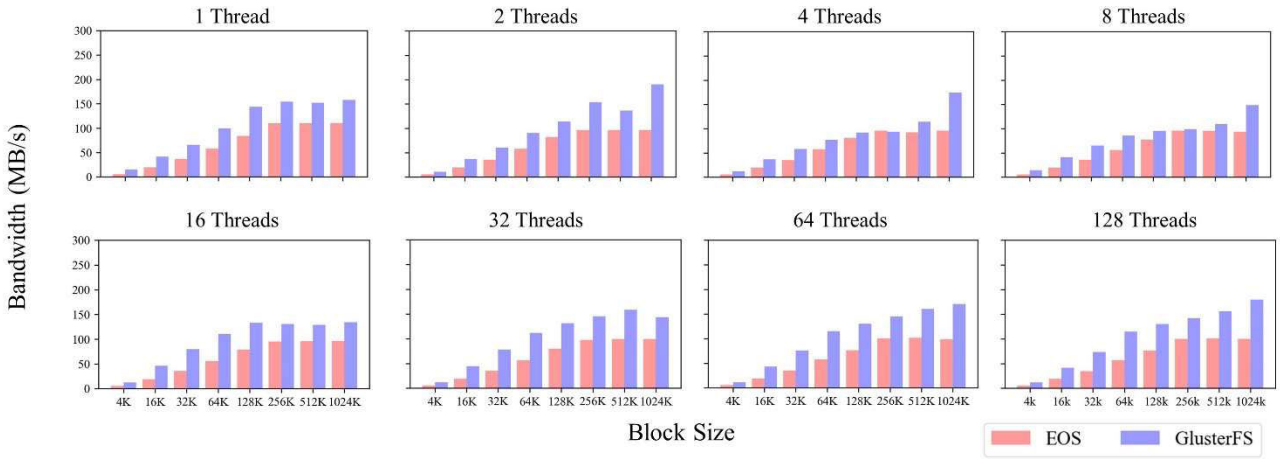


FIGURE 22. The overall bandwidth results for the replication layout rand write benchmarks of EOS and GlusterFS.

1) RAID 6 RAIN LAYOUT RAND WRITE

Figure 20 shows the overall bandwidth results of the RAID 6 RAIN layout rand write benchmark for the EOS and GlusterFS. It can be observed that the performance of EOS is slightly lower than that of GlusterFS at 1, 2, and 4 threads, but the performance is reversed from 8 threads. Generally, both EOS and GlusterFS exhibit a type of performance in which the throughput saturates early with block size at low thread counts, and the throughput increases up to 1024K block size as the number of threads increases.

Figure 21 shows the trends of RAID 6 RAIN layout rand write bandwidth, IOPS, and latency for EOS and GlusterFS.

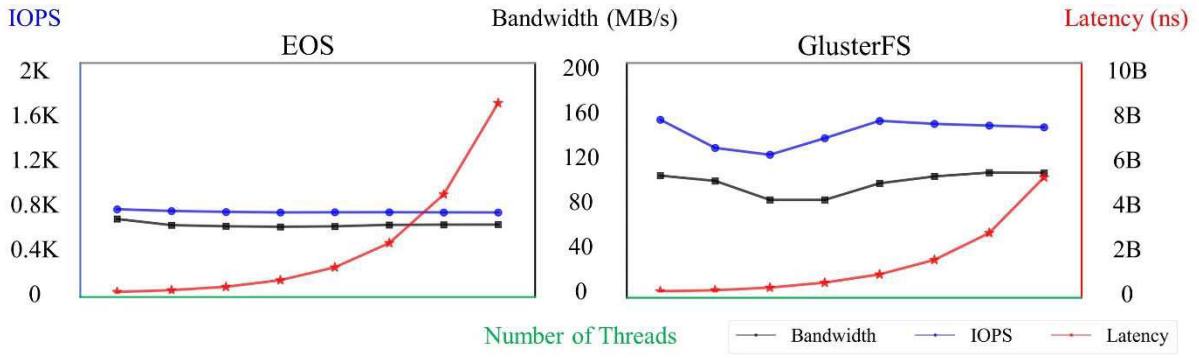


FIGURE 23. Bandwidth, IOPS and latency for the replication layout rand write benchmarks of EOS and GlusterFS.

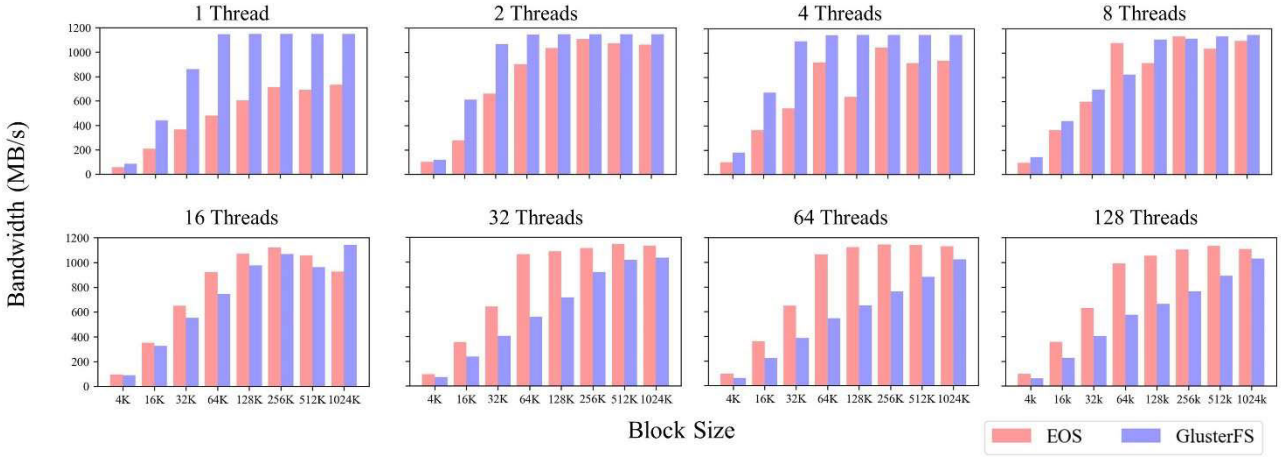


FIGURE 24. The overall bandwidth results for the distributed layout rand write benchmarks of EOS and GlusterFS.

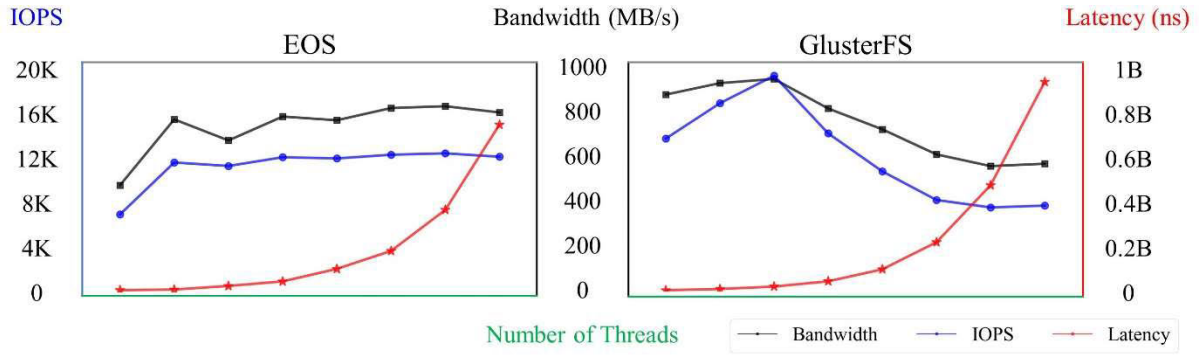


FIGURE 25. Bandwidth, IOPS and latency for the distributed layout rand write benchmarks of EOS and GlusterFS.

The EOS showed a consistent increase in throughput with an increase in the number of threads. GlusterFS exhibits the maximum performance at 1 thread, followed by a decrease in throughput. It shows a slightly significant decrease in throughput at 8 threads and maintains a stable performance thereafter. GlusterFS exhibited a significantly higher latency, and lower performance than EOS at high thread counts, indicating that EOS is advantageous for large workloads. GlusterFS records higher bandwidth and IOPS than EOS for 1, 2, and 4 thread items, indicating that GlusterFS performs better at low thread counts.

2) REPLICATION LAYOUT RAND WRITE

Figure 22 displays the overall bandwidth results of the replication layout rand write benchmark for EOS and GlusterFS. EOS shows performance saturation from a 256 K block size at all thread counts, whereas GlusterFS generally exhibits an increasing trend in performance up to a 1024 K block size. For all items, the GlusterFS performed slightly better than the EOS.

Figure 23 shows the trends of the replication layout rand write bandwidth, IOPS, and latency for the EOS and GlusterFS. The EOS showed almost no variation in throughput

depending on the number of threads. GlusterFS also exhibits a similar trend, but overall, it performs better than EOS with some variations. Notably the EOS showed a very high latency at 128 threads.

3) DISTRIBUTED LAYOUT RAND WRITE

Figure 24 shows the overall bandwidth results of the distributed layout rand write benchmark for EOS and GlusterFS. Owing to differences in data storage methods, a wider bandwidth range was measured than for other Rand Write items. It can be observed that the block size where performance saturation occurs varies depending on the number of threads. Similar to RAID 6 RAIN, GlusterFS's performance saturation point shifted to a larger block size as the number of threads increased. This means that the workload is relatively small when the block size unit is large as the number of threads increases.

Figure 25 shows the trends of distributed layout rand write bandwidth, IOPS, and latency for the EOS and GlusterFS. GlusterFS exhibits an increasing trend in throughput for up to 4 threads, but then shows a sharp decrease in throughput. The EOS shows a relatively consistent performance from 2 threads onwards. Latency was slightly higher for GlusterFS at high thread counts.

V. SUMMARY AND CONCLUSION

In this section, we summarize our performance analysis study, which provides the following results and implications through performance measurements and observations:

- A significant performance difference between the two DFSs is found, even though the hardware environment and the method for each file system are the same. Therefore, it is necessary to determine an appropriate DFS according to the situation.
- Bandwidth, IOPS, and latency vary significantly depending on various factors, such as layout, thread load, and block size. Therefore, it is important to understand the characteristics of the workload and predict how these factors affect the performance.
- When configuring a DFS in a RAID storage environment, the EOS is generally a good choice in situations where a large amount of work is expected. GlusterFS guarantees a somewhat consistent performance regardless of the amount of work and is more advantageous for random workloads.

In this study, we configured and compared DFSs based on RAID storage. Our results demonstrate the need to accurately understand the characteristics of the workload and select an appropriate DFS for the situation when constructing a storage system. Therefore, we provide insights into the performance characteristics of each RAID storage-based DFS that can be used for tapeless storage. As our study shows, the performance of DFS varies significantly depending on the nature of the workload, so it is important to carefully consider solutions that are suitable for individual cases.

ACKNOWLEDGMENT

The authors would like to extend their sincere thanks to the Global Science Experimental Data Hub Center (GSDC) at the Korea Institute of Science Technology Information (KISTI) for their support of their research.

REFERENCES

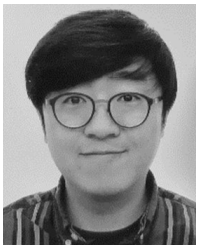
[1] R. H. Dee, "Magnetic tape for data storage: An enduring technology," *Proc. IEEE*, vol. 96, no. 11, pp. 1775–1785, Nov. 2008, doi: 10.1109/JPROC.2008.2004311.
[2] R. L. Moore, J. D'Aoust, R. H. McDonald, and D. Minor, "Disk and tape storage cost models," *Archiving Conf.*, vol. 4, no. 1, pp. 29–32, Jan. 2007, doi: 10.2352/issn.2168-3204.2007.4.1.art00008.
[3] *Going Tapeless—BCNET*. Accessed: Aug. 9, 2023. [Online]. Available: https://www.bc.net/sites/www.bc.net/files/bcnet/NewsEvents/Conference Archives/2017Conference/Presentations/2017Conference-Presentation-GoingTapeless.pdf
[4] S. U. Ahn, L. Betev, E. Bonfillou, H. Han, J. Kim, S. H. Lee, B. Panzer-Steindel, A. J. Peters, and H. Yoon, "Seeking an alternative to tape-based custodial storage," in *Proc. EPJ Web Conf.*, vol. 245, 2020, p. 04001, doi: 10.1051/epjconf/202024504001.
[5] S. U. Ahn, L. Betev, E. Bonfillou, H. Han, J. Kim, S. H. Lee, B. Panzer-Steindel, A. J. Peters, and H. Yoon, "A disk-based archival storage system using the EOS erasure coding implementation for the ALICE experiment at the CERN LHC," *J. Inf. Sci. Theory Pract.*, vol. 10, pp. 56–65, Jun. 2022, doi: 10.1633/JISTaP.2022.10.S.6.
[6] J. Blomer, "A survey on distributed file system technology," *J. Phys., Conf.*, vol. 608, May 2015, Art. no. 012039, doi: 10.1088/1742-6596/608/1/012039.
[7] *Analysis of Six Distributed File Systems*. Accessed: Aug. 9, 2023. [Online]. Available: https://inria.hal.science/hal-00789086/document
[8] S. Bai and H. Wu, "The performance study on several distributed file systems," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery*, Oct. 2011, pp. 226–229, doi: 10.1109/cyberc.2011.45.
[9] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *ACM SIGMOD Rec.*, vol. 17, no. 3, pp. 109–116, Jun. 1988, doi: 10.1145/971701.50214.
[10] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, reliable secondary storage," *ACM Comput. Surv.*, vol. 26, no. 2, pp. 145–185, Jun. 1994, doi: 10.1145/176979.176981.
[11] E. Levy and A. Silberschatz, "Distributed file systems: Concepts and examples," *ACM Comput. Surveys*, vol. 22, no. 4, pp. 321–374, Dec. 1990, doi: 10.1145/98163.98169.
[12] V. Bohossian, C. C. Fan, P. S. LeMahieu, M. D. Riedel, L. Xu, and J. Bruck, "Computing in the RAIN: A reliable array of independent nodes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 2, pp. 99–114, Feb. 2001, doi: 10.1109/71.910866.
[13] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding Vs. replication: A quantitative comparison," in *Peer-to-Peer Systems* (Lecture Notes in Computer Science), vol. 2429, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Germany: Springer, 2002, doi: 10.1007/3-540-45748-8_31.
[14] S. B. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, "Erasure coding for distributed storage: An overview," *Sci. China Inf. Sci.*, vol. 61, no. 10, pp. 1–45, 2018, doi: 10.1007/s11432-018-9482-6.
[15] *Introduction—EOS DIOPSIDE Documentation*. Accessed: Aug. 11, 2023. [Online]. Available: https://eos-docs.web.cern.ch/diopside/introduction/index.html
[16] *What is Gluster?* Introduction—Gluster Docs. Accessed: Aug. 10, 2023. [Online]. Available: https://docs.gluster.org/en/latest/Administrator-Guide/GlusterFS-Introduction/
[17] J.-Y. Lee, M.-H. Kim, S. A. Raza Shah, S.-U. Ahn, H. Yoon, and S.-Y. Noh, "Performance evaluations of distributed file systems for scientific big data in FUSE environment," *Electronics*, vol. 10, no. 12, p. 1471, Jun. 2021, doi: 10.3390/electronics10121471.
[18] X. Zhang, S. Gaddam, and A. T. Chronopoulos, "Ceph distributed file system benchmarks on an openstack cloud," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Nov. 2015, pp. 113–120, doi: 10.1109/CCEM.2015.12.
[19] G. Donvito, G. Marzulli, and D. Diacono, "Testing of several distributed file-systems (HDFS, ceph and GlusterFS) for supporting the HEP experiments analysis," *J. Phys., Conf.*, vol. 513, no. 4, Jun. 2014, Art. no. 042014, doi: 10.1088/1742-6596/513/4/042014.

[20] R. Leite and P. Solis, "Performance analysis of data storage in a hyperconverged infrastructure using Docker and GlusterFS," in *Proc. XLV Latin Amer. Comput. Conf. (CLEI)*, Sep. 2019, pp. 1–10, doi: 10.1109/clei47609.2019.235108.
[21] G. Adde, B. Chan, D. Duellmann, X. Espinal, A. Fiorot, J. Iven, L. Janyst, M. Lamanna, L. Mascetti, J. M. P. Rocha, A. J. Peters, and E. A. Sindrilaru, "Latest evolution of EOS filesystem," *J. Phys., Conf.*, vol. 608, May 2015, Art. no. 012009, doi: 10.1088/1742-6596/608/1/012009.
[22] D. R. Purandare, D. Bittman, and E. L. Miller, "Analysis and workload characterization of the CERN EOS storage system," in *Proc. Workshop Challenges Opportunities Efficient Performant Storage Syst.*, Apr. 2022, pp. 1–7, doi: 10.1145/3503646.3524293.
[23] M. Riasetiawan and N. Amien, "Random and sequence workload for web-scale architecture for NFS, GlusterFS and MooseFS performance enhancement," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 3, pp. 1–9, 2022, doi: 10.14569/ijacsa.2022.0130354.
[24] M. Selvaganesan and M. A. Liazudeen, "An insight about GlusterFS and its enforcement techniques," in *Proc. Int. Conf. Cloud Comput. Res. Innov. (ICCCRI)*, May 2016, pp. 120–127, doi: 10.1109/iccri.2016.26.
[25] P. Macko and J. Hennessey, "Survey of distributed file system design choices," *ACM Trans. Storage*, vol. 18, no. 1, pp. 1–34, Feb. 2022, doi: 10.1145/3465405.
[26] *Libfuse API Documentation*. Libfuse. Accessed: Aug. 11, 2023. [Online]. Available: http://libfuse.github.io/doxygen/
[27] V. Gueant. *IPERF*. Accessed: Aug. 11, 2023. [Online]. Available: https://iperf.fr/iperf-doc.php
[28] *Rain*. RAIN—EOS CITRINE Documentation. Accessed: Aug. 11, 2023. [Online]. Available: https://eos-docs.web.cern.ch/using/rain.html
[29] *Architecture—Gluster Docs*. Accessed: Aug. 11, 2023. [Online]. Available: https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/
[30] Axboe. *AXBOE/FIO: Flexible I/O Tester*. GitHub. Accessed: Aug. 11, 2023 [Online]. Available: https://github.com/axboe/fio

JUNGBIN KIM received the B.S. degree in computer science from Chungbuk National University, South Korea, in February 2022, where he is currently pursuing the M.S. degree with the Computer Science Department. His research interests include storage systems, cloud computing, and file systems.

HYEON-JIN YU received the B.S. degree in computer science from Chungbuk National University, South Korea, in February 2023, where she is currently pursuing the Ph.D. degree with the Computer Science Department. Her research interests include cloud and scientific computing, job processing, and containerization.

HYEONGBIN KANG is currently pursuing the B.S. degree in computer science with Chungbuk National University, South Korea. His research interest includes cloud and resource optimization.



JAE-HYUCK SHIN received the B.S. and M.S. degrees from Chungbuk National University, South Korea, where he is currently pursuing the Ph.D. degree with the Department of Computer Science. He previously worked at Samsung Research. His research interests include distributed computing and embedded systems.

HEESEOK JEONG received the B.S. and M.S. degrees in computer engineering from Chungnam National University, Daejeon, South Korea, in 2002 and 2004, respectively. He is currently a Senior Researcher with the Global Science Experimental Data Hub Center (GSDC), Korea Institute of Science and Technology Information. His research interests include distributed computing systems and artificial intelligence.

SEO-YOUNG NOH received the B.E. and M.E. degrees in computer engineering from Chungbuk National University, South Korea, and the M.S. and Ph.D. degrees in computer science from Iowa State University. He is currently an Associate Professor with the School of Computer Science, Chungbuk National University. Prior to joining Chungbuk National University, he worked as a Principal Researcher at the National Supercomputing Division, Korea Institute of Science and Technology Information, and as a Chief Research Engineer at LG Electronics. His research interests include scientific data management, cloud and scientific computing, Linux platforms, and database management systems.