Performance Benchmarking of File Systems

Nikhil Pavan Kanaka
Department of Computer Science and Engineering
The Ohio State University
Columbus, Ohio
kanaka.3@osu.edu

***Abstract*—The proposed project aims to evaluate and compare various file systems, analyze their performance, and determine their applicability in different scenarios. File systems are crucial components of computer systems, and their performance and features significantly affect the overall functionality of the system. The project begins with a survey of popular file systems to gain insights into their key features and limitations. The popular file systems, including `ext4`, `xfs`, and `btrfs`, are then benchmarked to analyze their performance differences. Different generations of the ext file-systems are then investigated and compared to identify their strengths, weaknesses, and trade-offs. The goal is to optimize the properties of commonly used file systems to enhance their performance in specific use cases. The results of the analysis show that the `xfs` file-system outperforms other file systems in terms of read and write throughput in the tested scenario. Overall, the project provides valuable insights into the performance and applicability of different file systems in various scenarios, enabling users to make informed decisions while choosing a file system for their specific needs.**

***Index Terms*—File systems, IO operations, Benchmarking**

I. Introduction

Managing data storage, retrieval, and access is critical in computer systems, and file systems play a vital role in this process. Evaluating and comparing different file systems available in the market is essential to identify the best fit for various use cases and scenarios.

Currently, the most popular file systems used in operating systems are `ext4` for Linux, `ntfs` for Windows, and `hfs+` for macOS. `ext4` is a highly scalable and portable journaling file system. On the other hand, `ntfs` and `hfs+` offer better access control, quotas, and encryption. `hfs+` supports snapshots, compression, and deduplication. Other file systems, such as `xfs`, are widely utilized in enterprise-level environments for large-scale data storage and high-performance computing. `zfs` is also used in enterprise-level environments for data storage and backup. `fat32` is best suited for removable storage devices, while `exFAT` is more suitable for larger files on removable storage devices.

The differences in file systems are vast, including the features offered and the way they manage files. Study of various features such as Journaling, Snapshots, Compression, Encryption, Deduplication, Quotas, Access Control, Scalability, and Portability is necessary as these can have a significant impact on the performance of file systems and need to be carefully considered when selecting a file system for a specific workload. The performance of a file system can be affected by the speed and capacity of the storage devices used, as well as the CPU and memory configuration of the system. Understanding these factors is critical to optimizing the performance of file systems for different types of workloads.

We have chosen `ext4`, `xfs`, and `btrfs` for benchmarking, as they are some of the most widely used file systems in the Linux operating system. `ext4` is the default file system for many Linux distributions, while `xfs` and `btrfs` are commonly used for specific purposes, such as high-performance computing and data storage. Different generations of `ext` file systems will be studied and benchmarked to understand the trade-off between features and performance, as some features might cause an overhead or performance difference in general.

The benchmarking will be conducted on a CentOS 7 Linux environment using `filebench`, shell scripts, and tools like `fio` and RAID controllers. This study will provide valuable insights into the performance and suitability of different file systems for specific use cases and scenarios.

II. Related work

The field of file systems has been widely studied by researchers and engineers, and their features and limitations are well-known. However, with the increasing demands of data storage and processing, there is a need to evaluate file system performance and optimize them for various types of workloads.

Several projects have been conducted in the past to evaluate different file systems, In a project by authors [15], `btrfs`, `xfs`, and `ext4` were compared on a single SATA drive with a 6 Gbps link using the Linux kernel make tool. `btrfs` was found to be slightly better in average seek count, while `ext4` had higher throughput, little less than twice as much compared to `btrfs` and `xfs`. Also, `ext4` slightly outperformed in average I/O operations per second. However, since this test was meta-data intensive, the authors concluded that, in general, these file systems had similar performance. The same authors [15] performed the Flexible File System Benchmark (FFSB) test, simulating a mail server that created 500,000 files in 1,000 directories with file size varying from 1 KB to 1 MB. This test showed that `ext4` had better throughput in both read and write scenarios, while `btrfs` was slightly behind. The worse results were with `xfs`. The final test in [15] was about write performance, using Tiobench, which wrote to a 2,000 MB file with 1, 2, 4, 8, and 16 threads. This test showed that `btrfs` ran faster than the other two file systems. In [16], `ext3`, `ext4`, `ReiserFS`, `jfs`, `xfs`, and `btrfs` were

tested on three advanced format hard disk drives (the older 1 TB Western Digital HDD and the newer 2 TB Seagate and 3 TB Toshiba HDDs) to benchmark file system performance against partition alignment. The authors created a fresh file system and performed several writes and reads of files that were 365 MB and 451 MB in size. On the Western Digital disk, `btrfs` was found to be the best in reading performance, while `ext4` slightly outperformed in writing performance. The situation was reversed on the Toshiba HDD. However, on the Seagate HDD, `btrfs` was better in both aspects. In [6], `btrfs` and `ext4` were tested on a single disk with several benchmarking tools. The Iozone tool showed that `btrfs` was better in general for both reading and writing large size files. The Seekwatcher tool showed that `btrfs` did many more seeks than ext in terms of read performance for large files, but `btrfs` had better write performance. The Seekwatcher tool showed that `btrfs` did more seeks than `ext4`.

In another study by authors [13], the performances of `ext4`, `btrfs`, and `xfs` were compared as guest file systems under a Linux environment. The computational chemistry test showed that `ext4` had higher throughput and made fewer I/O operations than `btrfs`.

File systems have been extensively studied, and their features and limitations are well-known in the field. However, with the rapid growth of data storage and processing demands, there is a need to re-evaluate the performance of file systems and optimize them.

Several projects have evaluated different file systems based on their generalized performance. As suggested by earlier studies, different file systems may perform better for specific workloads, block sizes, journaling modes, and file system options. Therefore, it is crucial to analyze, evaluate and compare different file systems based on their feature properties and hardware configurations to study the impact on the performance for different types of workloads and hardware configurations.

III. XFS, BTRFS AND EXT FILESYSTEMS

`ext4` is considered the standard for Linux systems, while `xfs` is known for its excellent scalability and fast file creation and deletion. `btrfs` may not be as fast as the other two, but its advanced features make it an attractive option for certain use cases.

Feature	EXT4	XFS	BTRFS
Architecture	Hashed B-tree	B+ tree	Extent based
Introduced	2006	1994	2009
Max volume size	1 Ebytes	8 Ebytes	16 Ebytes
Max file size	16 Tbytes	8 Ebytes	16 Ebytes
Max number of files	4 billion	2^{64}	2^{64}
Max file name size	255 bytes	255 bytes	255 bytes
Attributes	Yes	Yes	Yes
Transparent compression	No	No	Yes
Transparent encryption	Yes	No	Planned
Copy-on-Write (COW)	No	Planned	Yes
Snapshots	No	Planned	Yes

Fig. 1. Characteristics of xfs, btrfs, and ext4 compared

The filesystems `xfs`, `btrfs`, and `ext4` demonstrate unique and discernible traits, as exemplified by the graphical representation depicted in Fig. 1. Each of these filesystems possesses its own set of advantages and disadvantages, making them suitable for different use cases and environments. By examining their distinctive characteristics, one can gain a better understanding of which filesystem is best suited for a particular application or scenario.

A. *xfs*

`xfs` is a file system that was originally developed at Silicon Graphics, Inc. with the purpose of being a highly scalable and high-performance file system. [14] The main aim of `xfs` is to provide high parallel I/O performance, high scalability of I/O threads, and high file system bandwidth [11]. In order to achieve these objectives, `xfs` incorporates a number of key characteristics, which are as follows:

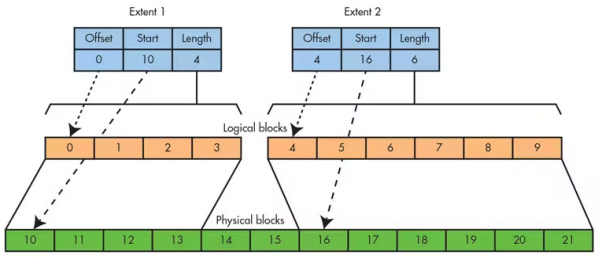

Fig. 2. File allocation in XFS

- Maximum file system size and maximum file size of nearly 8 EB.
- Journaling for metadata operations, which helps to improve data consistency and ensure fast recovery in the event of a crash.
- Partitioning into allocation groups, which are fixed-size virtual storage regions that enable `xfs` to balance the load across the file system.
- Based on extents (Fig. 2), where file blocks can have extents of variable length, allowing for better space utilization and improved performance.
- Delayed allocation for minimizing fragmentation and increasing performance by avoiding unnecessary disk writes.
- Implemented direct I/O for high throughput, and non-cached I/O for DMA devices, which improves I/O performance for large files and databases. Snapshot capabilities, which allow for the creation of point-in-time copies of the file system.
- Support for user, group, and project disk quotas on block and inode, which help to ensure fair usage of disk resources.

Overall, `xfs` is a powerful file system that is ideal for large-scale storage applications where high performance and scalability are critical. Its unique combination of features makes it a popular choice for data centers and other demanding computing environments.

B. btrfs

Introduced in 2007, btrfs is a modern copy-on-write (COW) file system that is based on B-trees [15]. It was designed to address the need for improved scalability of large storage subsystems, while also providing advanced features such as fault tolerance, repair and easy administration [10] [17].

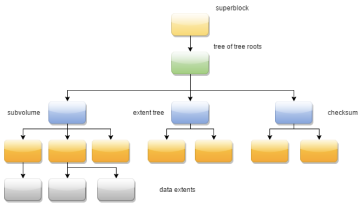

Fig. 3. Organization of btrfs file-system

The key features of btrfs include support for:

- Based on extents, which allows for more efficient use of disk space.
- Supports a maximum file size of 16 exabytes, making it suitable for large-scale storage needs.
- Allows for dynamic allocation of inodes, which allows for more efficient use of disk space.
- Uses a space-efficient method of packing small files and directories, which helps to minimize wasted space.
- Supports snapshots, subvolumes, and compression, which allows for better data management and improved performance.
- Optimized for use with flash storage, which allows for better performance and longer lifespan of flash-based storage devices.
- Fix errors on redundant files in the background, without disrupting ongoing operations.
- Online defragmentation and offline file system checks, which helps to maintain optimal performance and data integrity.
- Supports quotas on the subvolume level, which allows for better management of disk space usage.
- Built-in RAID functionality and checksums [12], which helps to ensure data integrity and provides better fault tolerance.
- Organized as a forest of B-trees, which allows for faster lookups and more efficient use of disk space.
- Uses a copy-on-write update method, which provides faster and more efficient updates to the file system. [4]

btrfs is a file system that is based on a unique type of B-trees known as B+ trees. These B+ trees employ basic B-tree construction, but also use a top-down update procedure, remove leaf linking, and utilize lazy reference-counting for space management [15]. By using the Copy-On-Write (COW) technique, btrfs can write modified blocks to a new location on the disk, which prevents the overwriting of old data. Additionally, COW provides data protection and improves performance. To accumulate updates in memory and write them all at once to new blocks on the disk, btrfs uses a transaction mechanism.

C. ext2

ext2 (Extended File System 2) is a second extended file system for the Linux operating system, developed as a replacement for the original extended file system (Ext) in 1993. ext2 is an open source file system that uses a traditional block-based allocation scheme, and it is still used on some Linux distributions today. [19] Here are some of the main features and characteristics of ext2:

- ext2 is a block-based file system, where files are stored in fixed-size blocks of data on the disk. The file system tracks the allocation of blocks to files using an inode structure, which stores metadata about each file, such as ownership, permissions, and timestamps.
- Maximum file size supported by ext2 is 2 terabytes (TB), and the maximum file system size is 32 terabytes.
- ext2 does not have journaling support, which means that in the event of a system crash or power failure, there is a risk of data loss or file system corruption. However, some tools like e2fsprogs can be used to repair the file system in case of errors.
- Pre-allocation to avoid fragmentation of files. When a file is created, the file system pre-allocates contiguous blocks of data on the disk to store the file, which helps to minimize fragmentation.
- Does not support file compression or encryption natively. However, these features can be added through third-party tools or using an encrypted file system like dm-crypt.
- Fast access times and low overhead, making it suitable for use on systems with limited resources.

ext2 is a simple and reliable file system that has been used for many years in the Linux community. Although it lacks some of the advanced features of newer file systems like ext4 and btrfs, it is still a popular choice for certain use cases, such as embedded systems or systems with limited resources.

D. ext3

ext3, also known as the third extended filesystem, is a journaled file system for Linux operating systems. It was first introduced in 2001 and is an improvement over the original ext2 filesystem. [3] Here are some detailed characteristics:

- Uses journaling to provide data consistency in case of a system crash or power failure. The journal records metadata changes before they are committed to the main file system, which allows for quicker recovery times.
- Fully compatible with ext2, which means that ext2 partitions can be mounted as ext3 and vice versa. This allows for easy migration between the two filesystems.
- Supports a maximum file system size of 32 TB and a maximum file size of 2 TB.
- Inode-based filesystem, which means that each file is represented by an inode structure that contains information about the file's ownership, permissions, and other attributes.

- Supports the use of extents, which improves performance by allowing for more efficient allocation of disk space.
- Uses delayed allocation to reduce fragmentation and improve performance. This means that blocks are not immediately allocated when a file is created, but are instead allocated when the data is actually written to disk.
- Soft updates, a technique that improves performance by allowing multiple disk writes to be combined into a single operation.
- Supports online resizing, which allows for the resizing of the filesystem while it is still mounted and in use.
- Checksums to ensure the integrity of data on the filesystem.

Overall, `ext3` is a reliable and stable filesystem that provides improved data consistency and performance over its predecessor, `ext2`.

E. `ext4`

`ext4`, short for fourth extended file system, is a widely used Linux file system that was introduced in 2008 as an improvement over its predecessor, `ext3`. It offers several new features and improvements to enhance the performance, scalability, and reliability of the file system. [2] [9]

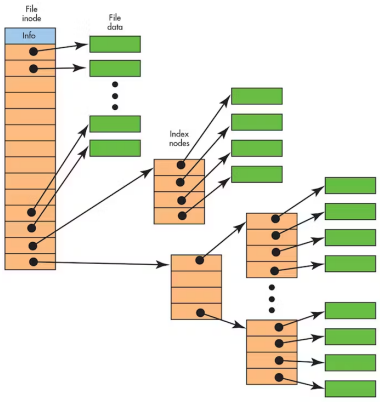

Fig. 4. File allocation in ext

- Support file systems up to 1 exabyte (EB) in size and individual files up to 16 terabytes (TB) in size, making it suitable for high-capacity storage needs.
- Uses a journal to record file system changes, making it less susceptible to corruption and data loss in case of sudden power loss or system crashes.
- Several performance enhancements over `ext3`, including faster file system checks and improved disk allocation algorithms. It also supports delayed allocation, which helps minimize fragmentation and improve write performance. [1]
- Backward compatible with `ext3` and can be easily upgraded from `ext3` to `ext4` without losing data.
- Allows users to resize the file system while it is still mounted and in use, without the need to unmount and remount it.
- Multi-block allocator to improve performance when writing small files, which can reduce fragmentation and improve read performance.
- Checksums to ensure the integrity of data stored on the file system, which helps prevent data loss due to disk errors or hardware failures.
- Extents to improve performance and reduce metadata overhead, allowing it to handle large files more efficiently.

Overall, `ext4` is a robust and reliable file system that offers significant improvements over its predecessors, making it an ideal choice for high-capacity storage needs and applications that require high performance and reliability.

IV. Methodology and Benchmarking Setup

To achieve the objectives of this project, we employed a variety of tools and techniques, such as filebench and shell-based tools, to assess and contrast the efficiency and appropriateness of different file systems in various situations. Filebench [18] is a versatile file system and storage benchmarking tool that provides a comprehensive environment for measuring and analyzing the performance of file systems, storage devices, and applications. It is a flexible benchmarking tool that can emulate complex applications using various workloads. It contains several predefined macro workloads that make it a popular framework for file system benchmarking [5] [8]. In addition to these workloads, Filebench also utilizes Workload Model Language (WML), a powerful language that can specify application behavior by encoding new workloads.

Firstly, we conducted a benchmarking survey of popular file systems in linux, including `ext4`, `xfs`, `btrfs`, and different workloads were run on appropriate partitions to record results for I/O throughput and latency. The benchmarking process involved disabling both the RAID controller cache and all the HDD caches to ensure accurate results without any unwanted optimizations by buffers or caches on the test system. Before each test, all I/O operations were serialized to prevent any optimizations by the Linux kernel through buffers and caches through a tool called `fio`. This tool is highly configurable and allows for fine-grained control over I/O operations, making it suitable for various use cases.

TABLE I
Hardware Profile for Benchmark Testing

Hardware	**Specification**
Processor	Intel Core i7-10750H Processor
Number of CPU Cores	6
Processor Speed	2.6 GHz up to 5.0 GHz
Memory (RAM)	16 GB DDR4 2933 MHz
Hard Disk Drive	1TB 15000 RPM 2.5” SATA HDD
Operating System	CentOS 7.5

Testing was performed on hardware system with specification presented in Table II. The Core i7-10750H Processor with 6 cores and a speed of 2.6 GHz up to 5.0 GHz is a high-performance processor suitable for running resource-intensive applications. The 16 GB DDR4 2933 MHz RAM allows for multitasking and faster data transfer. The 15000

RPM 2.5" SATA HDD (1TB) provides ample storage space and faster data access. The operating system, CentOS 7.5, is a widely used Linux distribution suitable for benchmarking file systems. Overall, these specifications provide a good baseline for evaluating the performance of file systems on the system.

TABLE II
HARD DISK DRIVE SPECIFICATION

Hardware	Specification
Hard Disk Drive	WD Blue 1TB SATA 6 Gb/s 15000 RPM (WD10EZEX)
Capacity	1 TB
Interface	SATA 6 Gb/s
Transfer Rate to Host	Up to 6 Gb/s
Cache	64 MB
Rotation Speed	15000 RPM
Average Latency	4.2 ms
Seek Time Average Read/Write	8.9 ms / 10.9 ms

The hard drive utilizes the Serial Advanced Technology Attachment (SATA) interface, providing a transfer rate of up to 6 Gb/s to the host. It has a cache size of 64MB, which allows for frequently accessed data to be stored in cache to enhance access times. The disk rotates at 15000 RPM, providing a high-speed read/write capability. The average latency of the hard drive is less than 4.2ms, representing the time it takes for the disk to rotate to the correct sector for reading or writing data. The average seek time of the hard drive is 8.9ms, representing the time it takes for the disk head to move to the correct track for reading or writing data.

V. RESULTS

To evaluate the performance of various file systems, scripts were written using Filebench and the system was then executed by using a 64MB file with block size 512KB and 2048KB for sequential and random reads and writes, as well as mixed workloads. To avoid any unwanted optimization by the buffers or caches on the test system, and to add different levels of variance based on the filesystem, both the RAID controller cache and all HDD caches were disabled. Prior to each test, it is ensured that the Linux kernel did not perform any optimizations by serializing I/O operations through buffers and caches, accomplished by running

```
sync && echo 3 > /proc/sys/vm/drop_caches
```

or

```
systemd-sysctl vm.drop_caches=3
```

and doubly deleting the page cache, inodes, and dentries.

The tests excluded numerous possibilities for optimization (such as `btrfs` -o, which can increase throughput by up to 30% in some cases). Prior to running the actual benchmark, an extensive series of tests with common file system operations such as mkdir, touch, echo, cat, dd, rm, and rmdir. These operations were repeated multiple times with different input values and over long loops to generate a significant amount of data. The collected data was analyzed to ascertain the ideal benchmarking setup for filebench.

A. Sequential Read

Sequential read throughput is the rate at which data can be read from a storage device in a continuous manner.

TABLE III
THROUGHPUT IN MBPS FOR SEQUENTIAL READ

FileSystem	512KB block	2048KB block
`ext2`	303.24	447.18
`ext3`	428.4	518.52
`ext4`	353.64	560.28
`btrfs`	197.4	566.37
`xfs`	372.96	512.43

The results of Sequential Read are shown in Table III. Based on the data, we can analyze the sequential read-throughput performance of different file systems. The table provides the throughput measurements in megabytes per second (MBps) for different file systems using 512 KB and 2048 KB block sizes.

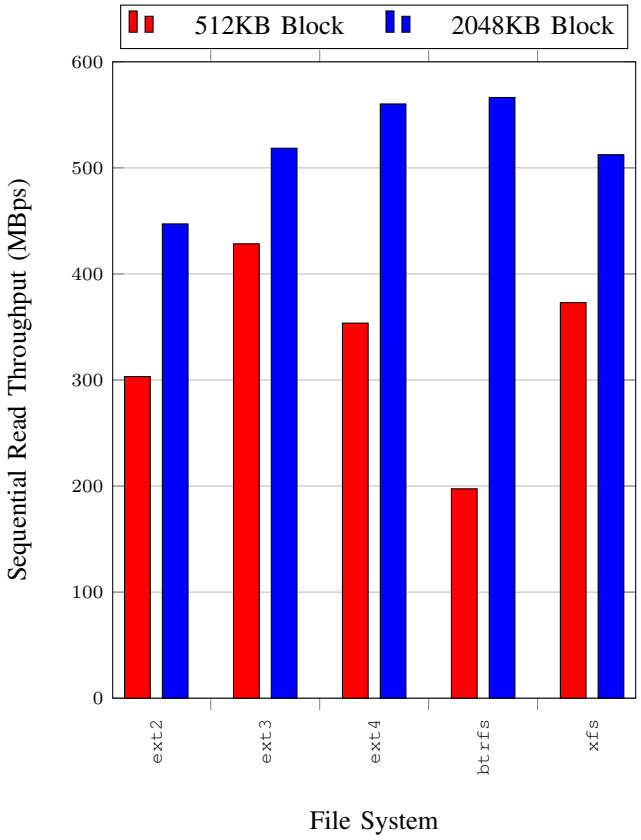

Fig. 5. Sequential Read Throughput of File Systems

The graph shown in Figure 5 illustrates the sequential write throughput of different file systems, including `ext2`, `ext3`, `ext4`, `btrfs`, and `xfs`. The data for both 512KB block and 2048KB block sizes are shown.

The results indicate that `xfs` has the highest sequential read throughput for both block sizes, with 372.96 MBps for 512KB block and 512.43 MBps for 2048KB block. `ext3` and `ext4` file systems also show good performance for both block sizes,

while btrfs has the lowest throughput for 512KB block size, but the highest for 2048KB block size. ext2 has the lowest throughput for both block sizes, indicating that it might not be the best option for scenarios that require high sequential read throughput.

B. Sequential Write

Sequential write throughput is the rate at which data can be written to a storage device in a continuous manner.

TABLE IV
THROUGHPUT IN MBPS FOR SEQUENTIAL WRITE

FileSystem	512KB Block	2048KB Block
ext2	12.6	83.52
ext3	12.8	100.05
ext4	17.64	133.11
btrfs	15.12	77.43
xfs	18.48	159.21

The results of Sequential Write are shown in Table IV. Based on the data, we can analyze the sequential write-throughput performance of different file systems. The table provides the throughput measurements in megabytes per second (MBps) for different file systems using 512 KB and 2048 KB block sizes.

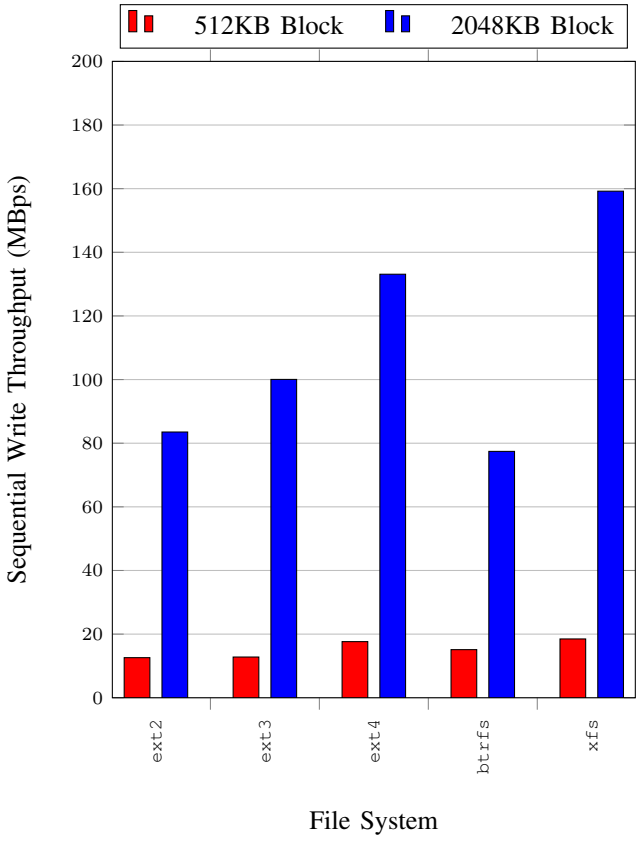

Fig. 6. Sequential Write Throughput of Various File Systems

The graph shown in Figure 6 illustrates the sequential write throughput of different file systems, including ext2, ext3, ext4, btrfs, and xfs. The data for both 512KB block and 2048KB block sizes are shown.

The results indicate that xfs has the highest sequential write throughput for both block sizes, with 18.48 MBps for 512KB block and 159.21 MBps for 2048KB block. ext4 file system also shows good performance for both block sizes, while btrfs has the lowest throughput for 512KB block size, but shows a moderate performance for 2048KB block size. ext2 and ext3 file systems have lower sequential write throughput for both block sizes, indicating that they might not be the best option for scenarios that require high sequential write throughput.

C. Random Read

Random read throughput is the rate at which data can be randomly accessed and read from a storage device.

TABLE V
THROUGHPUT IN MBPS FOR RANDOM READ

FileSystem	512KB Block	2048KB Block
ext2	75.6	198.24
ext3	84	265.44
ext4	88.2	283.08
btrfs	71.4	239.4
xfs	110.88	251.16

The results of random read throughput are presented in Table V. The table provides the throughput measurements in megabytes per second (MBps) for different file systems using 512 KB and 2048 KB block sizes. Based on the data, we can analyze the random read-throughput performance of different file systems.

The graph shown in Figure 7 illustrates the sequential write throughput of different file systems, including ext2, ext3, ext4, btrfs, and xfs. The data for both 512KB block and 2048KB block sizes are shown.

The results indicate that xfs has the highest random read throughput for 512KB block size with 110.88 MBps, while for 2048KB block size, ext4 has the highest throughput with 283.08 MBps. ext3 and ext4 file systems also show good performance for both block sizes, while btrfs has the lowest throughput for 512KB block size, but it performs better for 2048KB block size. Interestingly, ext2 has the lowest throughput for both block sizes, indicating that it might not be the best option for scenarios that require high random read throughput.

D. Random Write

Random write throughput measures the rate at which data can be written to a storage device in a random manner.

The results of random write throughput are presented in Table III. The table shows the throughput measurements in megabytes per second (MBps) for different file systems using 512 KB and 2048 KB block sizes.

The graph shown in Figure 8 illustrates the sequential write throughput of different file systems, including ext2, ext3,

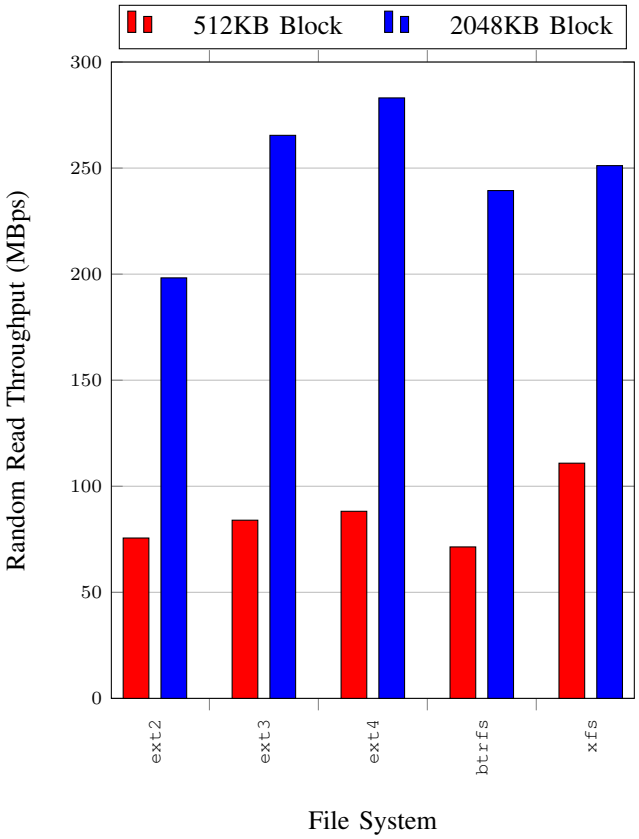

Fig. 7. Random Read Throughput of Various File Systems

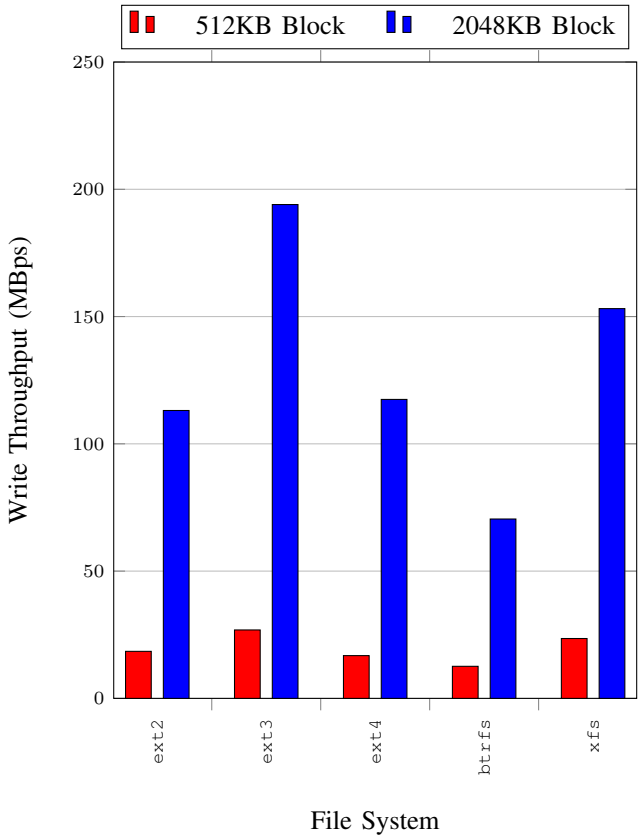

Fig. 8. Random Write Throughput of Various File Systems

TABLE VI
THROUGHPUT IN MBPS FOR RANDOM WRITE

FileSystem	512KB Block	2048KB Block
ext2	18.48	113.1
ext3	26.88	194.01
ext4	16.8	117.45
btrfs	12.6	70.47
xfs	23.52	153.12

ext4, btrfs, and xfs. The data for both 512KB block and 2048KB block sizes are shown.

Based on the data, we can analyze the random write throughput performance of different file systems. The table indicates that xfs has the highest random write throughput for both block sizes, with 23.52 MBps for 512KB block and 153.12 MBps for 2048KB block. ext3 file system shows good performance for both block sizes, while ext2 and ext4 file systems have lower throughput for both block sizes. btrfs has the lowest throughput for both block sizes, indicating that it might not be the best option for scenarios that require high random write throughput.

VI. DISCUSSION

We analyzed the performance of different file systems for IO operations such as sequential read and write, and random read and write for two block sizes: 512KB and 2048KB. Our results indicate that the performance of file systems varies significantly based on the workload and configuration used.

For sequential read throughput, xfs performs the best for both block sizes, while btrfs performs the worst. For sequential write throughput, ext4 performs the best for both block sizes, while ext2 performs the worst. For random read throughput, xfs performs the best for both block sizes, while ext2 performs the worst. For random write throughput, ext3 performs the best for both block sizes, while btrfs performs the worst.

Overall, the results suggest that xfs and ext4 are the best-performing file systems for sequential and random read and write operations, while ext2 and btrfs [7] are the least performing.

Looking at the numbers for ext2, ext3, and ext4, we can see that there are some differences between the three file systems in terms of their performance for different types of operations. For sequential read throughput, ext3 and ext4 both perform better than ext2, with ext4 having the highest numbers. For sequential write throughput, ext4 performs better than ext2 and ext3, with the highest numbers in the 2048KB block size.

For random read throughput, ext4 has the highest numbers in both block sizes, followed closely by xfs. For random write throughput, ext3 has the highest numbers in both block sizes, followed by xfs. It's interesting to note that ext3 performs

better than `ext4` in random write throughput, which could be due to `ext3` having a more stable and mature codebase compared to `ext4`, which is still being actively developed.

Overall, as expected, `ext4` appears to be the best performer among the three file systems, with high numbers in sequential read and write throughput, as well as random read throughput. However, it's important to consider the specific use case and workload requirements when selecting a file system, as performance can vary depending on the application and hardware configuration.

VII. Limitations

Throughout the project, we encountered some assumptions, limitations, and constraints. One significant limitation was the limited scope of the study, as we only focused on a few file systems commonly used in the Linux operation system. The study's results may not be generalized for other file systems not included in the study. Another limitation was the hardware configurations, which may not represent all possible hardware configurations used in different scenarios. Additionally, the workload used in the study may not represent all possible workloads in different scenarios.

Despite these limitations, we believe that this project provides valuable insights into the performance of different file systems and their applicability in different scenarios. Moreover, our methodology can be extended to evaluate the performance of other file systems not included in this study, providing a foundation for future research on file systems' performance and optimization. In addition, we experimented with different configurations based on the workload and hardware used to optimize file system performance. Our results show that the optimal configuration for a file system depends on the specific use case and workload and that tweaking the configuration can result in significant performance improvements.

VIII. Conclusion

By conducting a comprehensive evaluation and comparison of popular file systems, analyzing their key features and limitations, and identifying the best-performing file systems for different types of IO operations, this project has provided valuable insights into selecting the optimal file system for their specific use case and workload.

While this project has identified several strengths, such as the comprehensive analysis of popular file systems, there are also limitations to acknowledge, such as the limited set of hardware and file systems evaluated. However, this project suggests several avenues for future research, such as expanding the benchmarking survey to include more file systems and workload types, exploring the impact of different file system options on performance in more detail, and evaluating the impact of different hardware configurations on file system performance.

Overall, this project provides a foundation for future research to improve system functionality and performance. By understanding the factors that influence the performance of file systems, users can make informed decisions when selecting the optimal file system for their specific use case and workload.

IX. Future Scope

The findings from this project have important implications for future studies in two key ways. Firstly, this work sheds light on the performance characteristics of various file systems and their suitability in diverse settings. Secondly, the project serves as a fundamental basis for future investigations on the effects of different file systems on data processing performance, encompassing the distinction between CPU and GPU processing. This investigation may lead to the discovery of novel methods to optimize file system performance and enhance the overall functionality of computer systems.

References

[1] Ext4 disk layout. https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout. Accessed: 10.4.2016.

[2] Ext4, red hat enterprise linux 6 storage administration guide. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/ch-ext4.html. Accessed: 10.4.2016.

[3] Mingming Cao, Theodore Y Tso, Badari Pulavarty, Suparna Bhattacharya, Andreas Dilger, and Alex Tomas. State of the art: Where we are with the ext3 filesystem. In *Proceedings of the Ottawa Linux Symposium (OLS)*, pages 69–96. Citeseer, 2005.

[4] Jianping Chen, Jiahua Wang, Zhiqiang Tan, and Changsheng Xie. Recursive updates in copy-on-write file systems - modeling and analysis. *Journal of Computers*, 9(10):2342–2351, oct 2014.

[5] Tobias Hirt. KVM - The Kernel-Based Virtual Machine. http://www.cs.hs-rm.de/~linn/fachsem0910/hirt/KVM.pdf, 2010.

[6] Mesfin Tesfaye Kebede. Performance comparison of btrfs and ext4 filesystems. Master's thesis, Network and System Administration Oslo And Akersus University College Of Applied Science, 2012.

[7] Jelena Kljaji'c, Nikola Bogdanovi'c, Milan Nankovski, Mihailo Tončev, and Branislav Djordjevi'c. Performance analysis of 64-bit ext4, xfs and btrfs filesystems on the solid-state disk technology. *INFOTEH-JAHORINA*, 15:563–566, 2016.

[8] Duc Le, Haibo Huang, and Hai Wang1. Understanding performance implications of nested file systems. In *FAST'12 Proceedings of the 10th USENIX conference on File and Storage Technologies*, pages 1–13. USENIX Association, 2012.

[9] Andreas Mathor, Mingming Cao, Suparna Bhattacharya, Andreas Dilger, Alexander Tomas, and Laurent Vivier. The new ext4 filesystem: current status and future plans. In *Proceedings of the 2007 Ottawa Linux Symposium*, volume 2, pages 21–33, 2007.

[10] Oracle. All About btrfs. https://docs.oracle.com/cd/E37670_01/E37355/html/ol_about_btrfs.html, 2016. [Accessed: 10-Apr-2016].

[11] Oracle. All About xfs. https://docs.oracle.com/cd/E37670_01/E37355/html/ol_about_xfs.html, 2016. [Accessed: 10-Apr-2016].

[12] Oracle. Btrfs, Getting Started. http://www.oracle.com/technetwork/articles/servers-storage-admin/gettingstarted-btrfs-1695246.html, 2016. [Accessed: 10-Apr-2016].

[13] Alexey Vladimirovich Ostroukh and Andrey Salniy. Research of performance linux kernel file systems. *International Journal of Advanced Studies (iJAS)*, 5(2):12–17, 2015.

[14] Red Hat. Red Hat Enterprise Linux 6 Storage Administration Guide. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/ch-xfs.html, 2016. [Accessed: 10-Apr-2016].

[15] Ohad Rodeh, Josef Bacik, and Chris Mason. The linux b-tree filesystem. *Trans. Storage*, 9(3):9, Aug 2013.

[16] Roderick W. Smith. Linux on 4kb-sector disks: Practical advice. http://www.ibm.com/developerworks/linux/library/l-4kb-sector-disks/, 2016. Accessed: 10.4.2016.

[17] The Btrfs project. Btrfs. https://btrfs.wiki.kernel.org/index.php/Main_Page, 2016. [Accessed: 10-Apr-2016].

[18] The Filebench project. Filebench. http://filebench.sourceforge.net/wiki/index.php/Filebench, 2016. [Accessed: 05-Apr-2016].

[19] KC Wang and KC Wang. Ext2 file system. *Systems Programming in Unix/Linux*, pages 301–356, 2018.