E-ISSN: 2964-6219, P-ISSN: 2964-6200
Volume 4 Number 2 August 2026.
pp: 81-92. DOI: https://doi.org/10.58477/cj.v4i2.476.

Computer Journal

RESEARCH ARTICLE Open Access

Performance and Efficiency Analysis of NTFS, exFAT, and EXT4 on SSD Using a Multi-Workload Approach

Wahyudin [1*], **Agung Permana** [2], **Jan Everhard** [3]

[1*,2,3] Department of Informatics Engineering, Universitas Budi Luhur, South Jakarta City, Special Capital Region of Jakarta, Indonesia.

*Correspondence email: wahyudinalzabir@gmail.com.

Received: 6 July 2026.
Revised: 2 August 2026.
Accepted: 10 August 2026.
Published: 30 August 2026.

Full list of author information is available at the end of the article.

Abstract

The increasing adoption of Solid State Drives (SSDs) as primary storage media requires careful selection of file systems that can deliver optimal performance and efficiency. Although numerous studies have compared file system performance, most focus on only two file systems or employ limited testing scenarios. This study evaluated the performance and efficiency of NTFS, exFAT, and EXT4 file systems on SSD using a multi-workload benchmarking approach in an Ubuntu Server 22.04 environment running on Oracle VirtualBox. Testing was conducted across five workload scenarios: Sequential Read, Sequential Write, Random Read, Random Write, and Small File Management, with throughput, latency, IOPS, CPU utilization, and storage overhead as measured parameters. Each scenario was repeated three times and analyzed using descriptive statistics. Results indicated that exFAT tended to produce higher throughput on sequential workloads (544 MB/s on Sequential Read; 495 MB/s on Sequential Write), although differences with EXT4 were marginal. EXT4 showed clearer advantages on random workloads (44,700 IOPS on Random Read; 34,500 IOPS on Random Write) and achieved the fastest completion time of 101.2 seconds on Small File Management. exFAT recorded the lowest storage overhead (0.2%) and CPU utilization (4.6%). The study concluded that no single file system outperformed all others across every workload; therefore, file system selection should be aligned with the dominant workload characteristics of the target system.

Keywords: SSD; NTFS; exFAT; EXT4; Benchmarking; Multi-Workload; File System; Performance.

Abstrak

Meningkatnya penggunaan *Solid State Drive* (SSD) sebagai media penyimpanan utama menuntut pemilihan *file system* yang tepat agar dapat memberikan performa dan efisiensi yang optimal. Meskipun banyak penelitian telah membandingkan performa *file system*, sebagian besar hanya melibatkan dua *file system* atau menggunakan skenario pengujian yang terbatas. Penelitian ini mengevaluasi performa dan efisiensi *file system* NTFS, exFAT, dan EXT4 pada SSD menggunakan pendekatan *benchmarking* multi-beban kerja dalam lingkungan Ubuntu Server 22.04 di atas Oracle VirtualBox. Pengujian dilakukan pada lima skenario beban kerja: *Sequential Read*, *Sequential Write*, *Random Read*, *Random Write*, dan *Small File Management*, dengan parameter *throughput*, *latency*, IOPS, utilisasi CPU, dan *overhead* penyimpanan. Setiap skenario diulang tiga kali dan dianalisis menggunakan statistik deskriptif. Hasil menunjukkan bahwa exFAT cenderung menghasilkan *throughput* lebih tinggi pada beban kerja sekuensial (544 MB/s pada *Sequential Read*; 495 MB/s pada *Sequential Write*), meskipun perbedaannya dengan EXT4 tergolong kecil. EXT4 menunjukkan keunggulan lebih jelas pada beban kerja acak (44.700 IOPS pada *Random Read*; 34.500 IOPS pada *Random Write*) dan mencatat waktu penyelesaian tercepat sebesar 101,2 detik pada *Small File Management*. exFAT mencatat *overhead* penyimpanan terendah (0,2%) dan utilisasi CPU terendah (4,6%). Penelitian ini menyimpulkan bahwa tidak ada satu *file system* yang unggul di semua skenario; oleh karena itu, pemilihan *file system* sebaiknya disesuaikan dengan karakteristik beban kerja dominan pada sistem yang dituju.

Kata Kunci: SSD; NTFS; exFAT; EXT4; *Benchmarking*; *Multi-Workload*; *File System*; Performa.

1. Introduction

Data storage technology has undergone significant evolution in recent years, driven by the rapid and widespread adoption of Solid State Drives (SSDs) as primary storage media across modern computing devices. Unlike Hard Disk Drives (HDDs), which rely on mechanical components for data retrieval, SSDs operate entirely on flash memory, offering substantially faster data access speeds, lower latency, more efficient power consumption, and greater resistance to physical shock (Jaffer *et al.*, 2019). These characteristics have established SSDs as the preferred storage medium across a broad range of platforms, including personal computers, enterprise servers, mobile devices, virtualization environments, and cloud computing infrastructure.

Despite the superior hardware capabilities of SSDs, storage system performance is not solely determined by the underlying physical medium. *File systems*—which govern directory structure, storage allocation, metadata management, and data access mechanisms—also significantly affect overall system performance and efficiency (Djordjevic & Timcenko, 2011). The choice of *file system* therefore has direct implications for throughput, latency, and resource utilization, particularly as SSDs continue to push the boundaries of I/O bandwidth. Zhan *et al*. (2025) demonstrated that conventional *file systems* are often unable to fully exploit the bandwidth of modern high-performance SSDs due to structural inefficiencies in the request-to-I/O transformation process, reinforcing the need to evaluate *file system* behavior under varied and realistic workload conditions.

Among the most widely used *file systems* today are New Technology File System (NTFS), Extended File Allocation Table (exFAT), and Fourth Extended File System (EXT4). NTFS, the standard *file system* for Windows operating systems, provides robust security features, journaling support, and compatibility with large files (Oyewole & Joseph, 2024). exFAT was designed as a lightweight, cross-platform *file system* optimized for flash memory, trading advanced features such as journaling for simplicity and minimal metadata overhead (Heeger *et al.*, 2022). EXT4, the primary *file system* in Linux environments, is recognized for high performance and storage optimization through extent-based allocation and delayed allocation mechanisms (Djordjevic & Timcenko, 2011). Each of these *file systems* carries distinct architectural characteristics that produce different behaviors when handling various types of storage workloads.

In modern computing environments, data access patterns are no longer homogeneous. Activities such as transferring large multimedia files, booting operating systems, managing databases, running virtual machines, and manipulating thousands of small files each produce fundamentally different I/O characteristics (Yadgar *et al.*, 2021). A *file system* that performs well on large sequential transfers may degrade significantly when subjected to random I/O or intensive metadata operations. This workload diversity means that evaluating a *file system* based on a single scenario or a limited set of parameters is insufficient to characterize its real-world behavior. Performance evaluation must therefore employ multiple workload scenarios that collectively represent the range of conditions encountered in actual computing systems (Jo, 2024).

Several prior studies have examined *file system* performance from various perspectives, yet each carries notable limitations. Sterniczuk (2022) compared NTFS and EXT4 on SSD and found that EXT4 outperformed NTFS across several scenarios, but the study was confined to two *file systems* and a narrow workload scope. Hines *et al*. (2023) evaluated NTFS and EXT4 in the context of embedded key-value database operations, which does not adequately represent general-purpose operating system or application workloads. Oyewole and Joseph (2024) analyzed EXT4 and NTFS from performance, security, and reliability perspectives, but their work relied primarily on conceptual analysis rather than controlled empirical experimentation. Fawwauzy *et al*. (2025) similarly compared the technical structures of NTFS and Linux *file systems* through literature review without conducting benchmarking tests. Rahman *et al*. (2025) compared NTFS, FAT32, and exFAT in a digital forensics context using file carving methods, which addresses data recovery rather than storage performance under workload variation. Collectively, these studies share a common pattern: most compare only two *file systems*, restrict testing to specific use cases, and rarely integrate resource efficiency metrics alongside performance measurements.

Studies that simultaneously compare NTFS, exFAT, and EXT4 on SSD using diverse, representative workloads while also accounting for CPU utilization and storage overhead remain scarce. This gap limits the availability of practical, evidence-based guidance for *file system* selection in systems where workload characteristics vary considerably. This study addresses those gaps by evaluating the performance and efficiency of NTFS, exFAT, and EXT4 on SSD using a multi-workload benchmarking approach covering five scenarios: Sequential Read, Sequential Write, Random Read, Random Write, and Small File Management.

In addition to throughput, latency, and IOPS, the study measures CPU utilization and storage overhead to provide a more complete characterization of each *file system* across varied operational conditions. The findings are intended to serve as a practical reference for users, system administrators, and researchers seeking to make informed *file system* selection decisions based on workload requirements.

2. Related Work

2.1 File System Performance Studies

Research comparing *file system* performance on SSD-based storage systems has grown considerably over the past decade. One of the most directly relevant studies was conducted by Sterniczuk (2022), who compared EXT4 and NTFS on SSD using file copy operations measured through a custom shell script. The results demonstrated that EXT4 consistently outperformed NTFS, particularly in scenarios involving random I/O. While this study provides a useful empirical baseline, it was limited to two *file systems* and did not include exFAT, nor did it measure resource efficiency metrics such as CPU utilization or storage overhead. Djordjevic and Timcenko (2011) analyzed EXT4 performance characteristics in Linux environments and identified key relationships between *file system* parameters and I/O throughput, offering foundational data that remains relevant to understanding EXT4 behavior on modern storage hardware. Hines *et al*. (2023) compared NTFS and EXT4 in the context of embedded key-value database storage operations and demonstrated that *file system* architecture exerts measurable influence on storage latency even for specialized workloads. However, this work does not address general-purpose workloads representative of typical operating system or application behavior, limiting its applicability to broader performance evaluation. Oyewole and Joseph (2024) provided a conceptual comparison of EXT4 and NTFS covering performance, security, and reliability dimensions, concluding that EXT4 exhibits advantages in data management efficiency. The absence of controlled empirical experimentation, however, limits the generalizability of their conclusions. Taken together, these studies establish a useful but incomplete picture of *file system* behavior on SSD — none simultaneously evaluates more than two *file systems* or integrates resource efficiency measurements alongside performance metrics.

2.2 exFAT and Cross-Platform File Systems

The exFAT *file system* has received comparatively less attention in performance-oriented research, with most existing work focusing on its structural characteristics or forensic properties. Heeger *et al*. (2022) investigated the internal organization and metadata management mechanisms of exFAT, confirming that it minimizes metadata overhead by design. This is achieved by omitting journaling and advanced allocation features, trading data integrity mechanisms for simplicity and broad cross-platform compatibility across Windows, macOS, and Linux. While this design makes exFAT well-suited for flash-based removable storage, it raises questions about its behavior under workload-intensive conditions — a gap that the present study directly addresses. Fawwauzy *et al*. (2025) compared the technical structures of NTFS and Linux *file systems* through literature review, highlighting architectural differences that contribute to divergent performance behaviors across workloads. Their analysis reinforces the argument that structural design choices — such as the presence or absence of journaling — have direct implications for I/O efficiency under different access patterns. Rahman *et al*. (2025) examined NTFS, FAT32, and exFAT in digital forensics contexts using file carving methods, demonstrating that distinct *file system* designs produce substantially different data management behaviors. Although their focus was on data recovery rather than storage performance, the finding that *file system* architecture fundamentally shapes data handling behavior is relevant to understanding the workload-dependent performance differences examined in the present study.

2.3 Virtualized Testing Environments

The use of virtualized environments for *file system* benchmarking has been established as a valid and reproducible methodological approach in prior research. Farizy and Gunawan (2020) evaluated EXT4 performance on virtualized infrastructure using Unix-specific benchmarking tools and demonstrated that a virtualized environment can produce consistent and comparable results across repeated test runs, provided that hardware specifications are fixed and background processes are controlled. Their work provides a direct methodological precedent for the approach adopted in this study. Djordjevic and Timcenko (2011) further analyzed EXT4 performance under Linux kernel environments and offered performance models that help explain the overhead behavior observed in virtualized storage benchmarking. The use of Oracle VirtualBox as the hypervisor in the present study is consistent with these prior approaches, as it enables a controllable testing environment with reproducible virtual hardware specifications across all three *file systems* under

evaluation.

2.4 Resource Efficiency and Large-Scale File Management

Resource efficiency has received comparatively less attention in *file system* benchmarking literature, despite its practical importance in production and server environments. Ramadan (2021) evaluated Docker storage performance across multiple backing *file systems* — including EXT4, XFS, and Btrfs — under four representative workload types: web server, mail server, file server, and random file access. Results consistently identified EXT4 as the most performant option for containerized environments involving random I/O, while also noting that Btrfs performed poorly under sequential write-dominated workloads. These findings align with the results of the present study and reinforce the argument that workload type is a decisive factor in *file system* performance outcomes. Shaikh (2024) extended this line of inquiry by benchmarking EXT4 against modern alternatives including XFS, Btrfs, ZFS, and F2FS at the scale of one billion files, demonstrating that metadata management efficiency becomes the decisive performance factor when file counts are extremely large — a finding that directly informs the Small File Management scenario employed in this study. On the hardware side, Dakic *et al*. (2024) evaluated storage performance and design patterns for high-performance computing environments, comparing Btrfs and ZFS across multiple workload scenarios on SSD. Their study demonstrated that workload type significantly influences which *file system* configuration delivers optimal performance, and that no single *file system* is universally superior across all use cases. This conclusion is consistent with the multi-workload evaluation approach adopted in the present study. Oyewole and Joseph (2024) further described the architectural characteristics of NTFS — including its journaling mechanism and Master File Table (MFT) structure — and noted that these features influence not only performance but also CPU resource consumption during intensive I/O operations. These findings collectively support the argument that resource efficiency metrics, particularly CPU utilization and storage overhead, should be treated as integral components of any comprehensive *file system* evaluation rather than secondary considerations.

2.5 State of the Art Analysis

Tabel 1. Comparison of previous studies and research positioning

Study	NTFS	exFAT	EXT4	SSD	Multi-Workload	CPU Utilization	Storage Overhead
Sterniczuk (2022)	✓	✗	✓	✓	✗	✗	✗
Hines *et al*. (2023)	✓	✗	✓	✗	✗	✗	✗
Oyewole & Joseph (2024)	✓	✗	✓	✗	✗	✗	✗
Heeger *et al*. (2022)	✗	✓	✗	✗	✗	✗	✗
Dakić *et al*. (2024)	✗	✗	✗	✓	✓	✗	✗
This Study	✓	✓	✓	✓	✓	✓	✓

Table 1 shows that most previous studies focused on comparing only two *file systems* or evaluating limited performance aspects. Sterniczuk (2022) and Hines *et al*. (2023) both restricted their comparisons to NTFS and EXT4, while Heeger *et al*. (2022) focused exclusively on exFAT without performance benchmarking. Dakić *et al*. (2024) is the only prior study that employed both SSD-based testing and a multi-workload approach, though their focus was on Btrfs and ZFS rather than the NTFS, exFAT, and EXT4 combination examined here. None of the reviewed studies simultaneously measured CPU utilization and storage overhead alongside throughput and IOPS metrics. This study addresses those gaps by combining performance and efficiency evaluation within a unified multi-workload testing approach covering all three *file systems*.

2.6 Research Gap and Positioning

A review of existing literature reveals several recurring limitations in prior *file system* research. Most studies compare only two *file systems*, restricting the scope of conclusions that can be drawn about relative performance across a broader set of options. Testing scenarios are often narrow, focusing on a single workload type or a specific application context such as database operations or digital forensics, rather than representing the diverse range of I/O patterns encountered in real computing systems. Resource efficiency metrics — particularly CPU utilization and storage overhead — are rarely integrated with performance measurements, despite their relevance to system design decisions. Most critically, the specific combination of NTFS, exFAT, and EXT4 evaluated simultaneously on SSD using a comprehensive multi-workload benchmarking approach has not been previously reported in the literature. The present study addresses these gaps by measuring throughput, IOPS, latency, CPU utilization, and storage overhead across five representative workload scenarios —

Sequential Read, Sequential Write, Random Read, Random Write, and Small File Management — for all three *file systems* within a uniform virtualized testing environment. This approach provides a more complete and practically applicable basis for workload-specific *file system* selection decisions in modern computing systems.

3. Method

3.1 Research Design

This study employed a quantitative experimental method with a laboratory benchmarking approach to evaluate and compare the performance and efficiency of NTFS, exFAT, and EXT4 *file systems* on SSD storage media (Sterniczuk, 2022). The experimental approach was selected to allow all variables affecting testing conditions to be rigorously controlled. Experiments were conducted by applying multi-workload stimuli representing real-world data storage activities in modern computing ecosystems (Yadgar *et al.*, 2021). Empirical data obtained from testing were analyzed comparatively to identify the performance characteristics, advantages, and limitations of each *file system*. Figure 1 illustrates the overall research workflow used in this study. The research began with a literature review focusing on SSD technology, NTFS, exFAT, EXT4, and benchmarking methodologies. Subsequently, a controlled experimental environment was prepared using Oracle VirtualBox and Ubuntu Server 22.04. Each *file system* was configured separately on the same SSD-based virtual storage and evaluated using five workload scenarios, namely Sequential Read, Sequential Write, Random Read, Random Write, and Small File Management (Sterniczuk, 2022). The benchmarking process generated performance and efficiency metrics including throughput, latency, IOPS, CPU utilization, and storage overhead (Sterniczuk, 2022). The collected data were then analyzed using descriptive statistical methods to identify the performance characteristics of each *file system* and formulate final conclusions and recommendations.

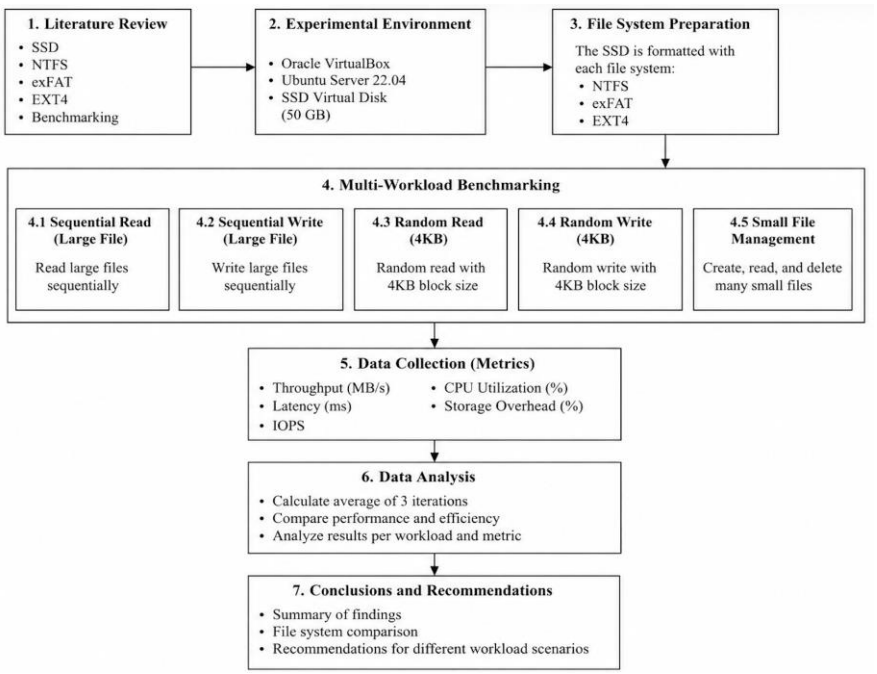

Figure 1. Research Workflow of Multi-Workload Benchmarking

3.2 Testing Environment

To ensure fairness and consistency, all *file system* formatting and testing procedures were conducted within an isolated virtualization environment using Oracle VirtualBox as the hypervisor (Sterniczuk, 2022). This approach ensured that all three *file systems* were tested under identical virtual hardware specifications. The virtual disk was configured with SSD passthrough emulation enabled to preserve the physical SSD characteristics of the host system. This virtualized testing approach is consistent with previous studies conducted by Farizy and Gunawan (2020), who evaluated EXT4 performance in a virtualized environment, and Djordjevic and Timcenko (2011), who demonstrated the suitability of Oracle VirtualBox for repeatable *file system* benchmarking.

Table 2. Physical Hardware Specifications

Component	Specification
Processor	Intel Core i7 / AMD Ryzen 7
RAM	16 GB
Host Storage	SSD 512 GB
Host OS	Windows 11 Pro 64-bit

Table 3. Virtual Machine Specifications

Component	Specification
Hypervisor	Oracle VirtualBox
vCPU	2 Cores
RAM	4 GB
Virtual Disk	50 GB (Fixed Size)
Guest OS	Ubuntu Server 22.04 LTS
File Systems	NTFS, exFAT, EXT4

3.3 Research Objects

The study focused on three *file system* architectures with distinct metadata handling characteristics:

1) NTFS *(New Technology File System)*: Represents Microsoft Windows' proprietary *file system*, employing a complex Master File Table (MFT) architecture with support for metadata journaling and security access control lists (ACL) (Oyewole & Joseph, 2024).
2) exFAT *(Extended File Allocation Table)*: A lightweight cross-platform *file system* (Windows, macOS, Linux) designed for flash memory optimization by minimizing metadata structure overhead and omitting journaling features (Heeger *et al.*, 2022).
3) EXT4 *(Fourth Extended File System)*: The standard open-source *file system* in Linux ecosystems, featuring transactional journaling and modern storage optimization technologies including delayed allocation and extent-based allocation (Sterniczuk, 2022).

3.4 Multi-Workload Testing Scenarios

Performance testing employed five workload scenarios representing real-world computer storage activities (Yadgar *et al.*, 2021; Shaikh, 2024):

1) Workload 1 — Sequential Read: Continuous large-data reading using a single 10 GB test file, measuring maximum data transfer throughput.
2) Workload 2 — Sequential Write: Continuous large-data writing with a 10 GB test file, evaluating maximum write speed and contiguous block allocation stability.
3) Workload 3 — Random Read: Non-sequential data reading with a constant 4 KB block size, simulating OS loading and application startup workloads.
4) Workload 4 — Random Write: Random data writing with a constant 4 KB block size, testing *file system* performance under simultaneous random I/O conditions.
5) Workload 5 — Small File Management: Manipulation of 10,000 small files (1 KB–100 KB) including create, read, and delete operations, evaluating metadata processing efficiency.

3.5 Testing Tools

Three CLI-based Linux tools were used for workload injection and metrics extraction:

1) FIO *(Flexible I/O Tester)*: Primary instrument for measuring throughput, latency, and IOPS on Workloads 1–4 using the *libaio* I/O engine with direct disk access to bypass OS buffer cache (Sterniczuk, 2022).
2) DD *(Data Duplicator)*: Secondary validation tool for large-file raw write speed testing, using *fdatasync* flags to force OS cache bypass (Ramadan, 2021).
3) IOSTAT: Background monitoring of storage device utilization and queue depth throughout benchmarking sessions (Ramadan, 2021).

3.6 Measured Parameters

Objective evaluation was based on five primary parameters (Sterniczuk, 2022):

1) Throughput (MB/s): Total data volume successfully transferred per second; higher values indicate better performance.
2) Latency (ms): System response time from I/O instruction issuance to execution completion; lower values indicate greater responsiveness.

3) IOPS: Total read or write operations handled per second, critical for random 4 KB workloads.
4) CPU Utilization (%): Processor workload consumed by the OS for *file* activity management, indicating computational efficiency of the *file system* architecture.
5) Storage Efficiency (%): Ratio of user-available storage after formatting relative to the 50 GB raw capacity, reflecting space consumed by system metadata.

3.7 Testing Procedure

Experiments were conducted systematically through four stages:

1) Stage 1 — Environment Preparation: Deploying the virtual machine on the hypervisor and allocating Fixed Size virtual disk storage.
2) Stage 2 — File System Configuration: Formatting partitions to NTFS, exFAT, and EXT4 in sequence and mounting each to the test directory.
3) Stage 3 — Benchmark Execution: Running FIO and DD test sequences with automated parameter extraction.
4) Stage 4 — Iteration and Averaging: Repeating each scenario three times to ensure data consistency and minimize thermal throttling artifacts.

3.8 Data Analysis

Table 4. Benchmark Repetition Results for Sequential Read Throughput

Iteration	NTFS (MB/s)	exFAT (MB/s)	EXT4 (MB/s)
1	509	543	535
2	514	545	538
3	513	544	538
Mean	512	544	537
Std. Dev	2.65	1.00	1.73

Table 4 presents the benchmark repetition results obtained for the Sequential Read workload. Each benchmark scenario was executed three times under identical conditions to ensure measurement consistency and reduce random variation. Similar repetition procedures were applied to all workload scenarios, and the performance values reported throughout this study represent the arithmetic mean of three independent measurements. Raw data collected from FIO, DD, and IOSTAT were processed using descriptive statistical methods. For each workload scenario, performance metrics were calculated as the arithmetic mean of three independent repetitions. Standard deviation values were also analyzed to evaluate measurement consistency (Sterniczuk, 2022). The resulting data were subsequently presented in comparative tables and graphical visualizations to facilitate performance and efficiency analysis across NTFS, exFAT, and EXT4 *file systems*.

4. Results and Discussion

4.1 Results

4.1.1 Overall Performance Results

Results from the multi-workload benchmarking experiments, averaged across three iterations per scenario, are summarized in Table 5.

Table 5. Benchmarking Results Summary

Workload	Parameter	NTFS	exFAT	EXT4	Best
Seq. Read	Throughput (MB/s)	512	544	537	exFAT
Seq. Write	Throughput (MB/s)	467	495	490	exFAT
Rand. Read	IOPS	41,800	39,200	44,700	EXT4
Rand. Read	Latency (ms)	0.24	0.25	0.21	EXT4
Rand. Write	IOPS	29,100	21,400	34,500	EXT4
Rand. Write	Latency (ms)	0.34	0.47	0.28	EXT4
Small Files	Time (s)	139.8	176.5	101.2	EXT4

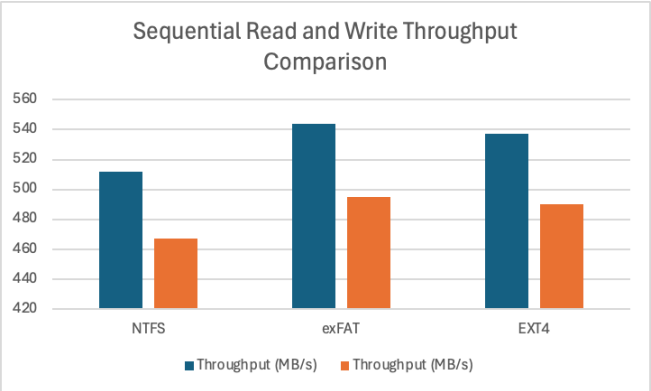

Figure 2. Sequential Throughput Comparison

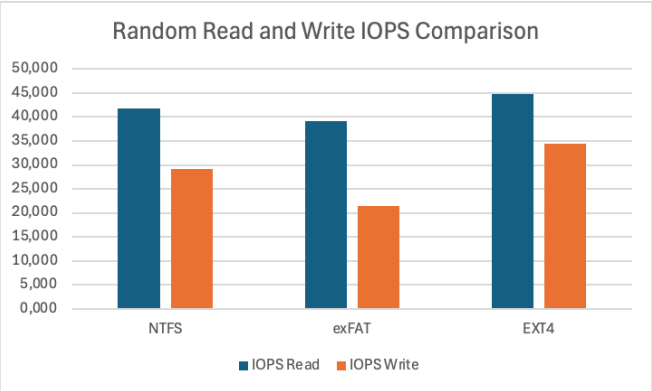

Figure 3. Random IOPS Comparison

Results indicate that no single *file system* outperformed all others across every workload category. Each *file system* exhibited distinct performance characteristics consistent with its architectural design and metadata management mechanisms.

4.1.2 Sequential Workload Performance

On Sequential Read testing, exFAT achieved the highest throughput at 544 MB/s, compared with EXT4 at 537 MB/s and NTFS at 512 MB/s. exFAT demonstrated a statistically notable advantage of approximately 6.3% over NTFS. However, the 1.3% margin between exFAT and EXT4 is marginal and may fall within measurement variation; it should therefore be interpreted as a tendency rather than a conclusive advantage. Similar results were observed for Sequential Write, where exFAT achieved 495 MB/s versus EXT4 at 490 MB/s and NTFS at 467 MB/s. exFAT showed a clearer advantage of approximately 6.0% over NTFS, while the 1.0% difference relative to EXT4 is too small to be claimed as conclusive without a larger number of test iterations. The exFAT advantage on sequential workloads is attributed to its simplified *file system* structure and lower metadata overhead compared to NTFS and EXT4. The absence of journaling mechanisms in exFAT eliminates transactional write overhead, enabling more efficient sequential data transfer. NTFS and EXT4 must allocate resources for metadata management and journaling to maintain data integrity, which introduces overhead that modestly reduces throughput on large sequential transfers.

4.1.3 Random I/O Performance

EXT4 dominated random workload testing with 44,700 IOPS on Random Read and 34,500 IOPS on Random Write, accompanied by the lowest latencies of 0.21 ms and 0.28 ms respectively. In contrast, exFAT exhibited the weakest random I/O performance, achieving only 21,400 IOPS on Random Write with a latency of 0.47 ms. EXT4's superiority on random workloads is attributable to its efficient block and metadata management mechanisms, including the *Delayed Allocation* (*delalloc*) feature that optimizes block allocation prior to writing, reducing fragmentation and improving I/O efficiency. These findings are consistent with Sterniczuk (2022) and Ramadan (2021), who both reported EXT4 as the superior *file system* for random I/O workloads. NTFS achieved competitive results at 41,800 IOPS (Random Read) and 29,100 IOPS (Random Write), demonstrating reasonable balance between compatibility and performance, though consistently below EXT4.

4.1.4 Small File Management

In the Small File Management scenario involving 10,000 files ranging from 1 KB to 100 KB, EXT4 completed all operations in the shortest time at 101.2 seconds, followed by NTFS at 139.8 seconds and exFAT at 176.5 seconds. This is consistent with findings reported by Oyewole and Joseph (2024) and aligns with large-scale evidence from Shaikh (2024), who demonstrated that metadata management efficiency is the primary performance determinant when handling very large numbers of files. EXT4's performance advantage in this scenario is associated with its *inode* structure and block allocation mechanisms, which optimize access to small files. exFAT's poor small-file performance confirms its design intent as a *file system* optimized for large sequential transfers rather than intensive metadata-heavy workloads.

4.1.5 Resource Efficiency

Table 6. Storage and CPU Efficiency Results

File System	Usable Capacity	Storage Overhead	Avg. CPU Utilization
NTFS	49.2 GB	0.8 GB (1.6%)	11.8%
exFAT	49.9 GB	0.1 GB (0.2%)	4.6%
EXT4	48.5 GB	1.5 GB (3.0%)	7.5%

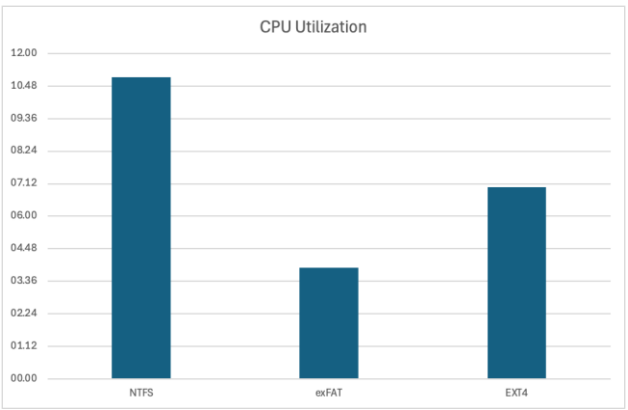

Figure 4. Average CPU Utilization

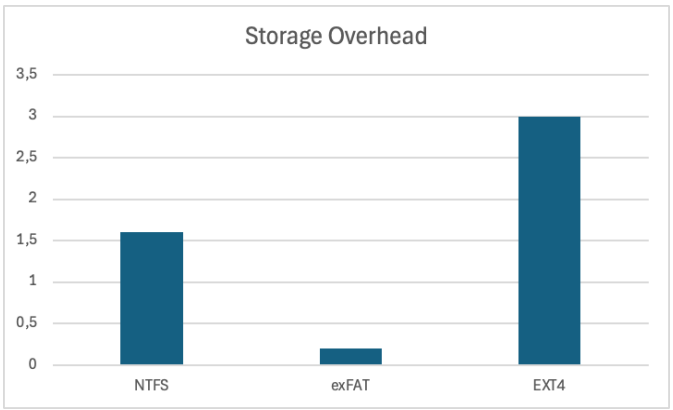

Figure 5. Storage Overhead

Figure 4 and Figure 5 show the resource efficiency characteristics of each *file system*. exFAT exhibited the lowest CPU utilization (4.6%) and storage overhead (0.2%), indicating a lightweight metadata structure. In contrast, EXT4 consumed the largest storage overhead (3.0%) due to journaling and metadata management mechanisms, while NTFS required the highest CPU utilization (11.8%) during workload execution. exFAT demonstrated the best resource efficiency with the lowest storage overhead (0.2%) and CPU utilization (4.6%). Its minimal metadata footprint allows nearly all storage capacity to be available for user data. EXT4 exhibited the highest storage overhead at 3.0%, reflecting the additional space required for journaling structures and metadata. However, this investment in metadata infrastructure directly enables EXT4's superior performance on random I/O and small file workloads. NTFS recorded the highest average CPU utilization at 11.8%, indicating that its complex MFT management and internal processing mechanisms require substantially greater computational resources than either exFAT or EXT4. These findings reveal a clear trade-off: exFAT maximizes storage and compute efficiency at the cost of random I/O performance, while EXT4 consumes more metadata space to achieve superior throughput across complex workloads.

4.2 Discussion

The results of this study are consistent with key findings from prior research. EXT4's advantage on random workloads corroborates Sterniczuk (2022) and Oyewole and Joseph (2024). The metadata efficiency advantage of EXT4 over exFAT and NTFS on small file workloads aligns with Ramadan (2021) and Shaikh (2024). The marginal sequential throughput differences between exFAT and EXT4 extend findings from Sterniczuk (2022) by introducing exFAT into the comparison and contextualizing it within a broader multi-workload framework. Critically, the multi-workload, multi-metric evaluation framework employed in this study extends beyond prior work in three key respects: (1) simultaneous evaluation of three *file systems* rather than pairs; (2) integration of resource efficiency metrics alongside traditional performance metrics; and (3) inclusion of small file management as a distinct workload category. These extensions provide a more complete characterization of *file system* behavior across practical computing scenarios.

The findings of this study carry direct implications for *file system* selection in real-world computing environments. The results suggest that workload type is the primary determinant of *file system* suitability, and that no single *file system* is universally optimal across all use cases. Table 7 summarizes the recommended *file system* selection based on workload characteristics.

Table 7. Recommended File System by Use Case

Use Case	Recommended File System	Rationale
Large sequential data transfer	exFAT	Lowest metadata overhead, highest sequential throughput
Database / random I/O workloads	EXT4	Superior IOPS and lowest latency on random access
Small file intensive operations	EXT4	Optimized *inode* structure and metadata management
Cross-platform removable storage	exFAT	Native support on Windows, macOS, and Linux
General-purpose Linux server	EXT4	Balanced performance with journaling and data integrity
Windows-native enterprise environments	NTFS	ACL support and MFT-based metadata management

Several limitations of this study should be acknowledged. First, all experiments were conducted within a virtualized environment using Oracle VirtualBox, which introduces a hypervisor abstraction layer that may attenuate peak I/O performance relative to bare-metal testing. Second, NTFS and exFAT were mounted on Ubuntu Server 22.04 using kernel-native drivers (*ntfs3* for NTFS and *exfat* kernel module for exFAT); performance characteristics may differ when these *file systems* are evaluated natively on Windows. Third, the use of three repetitions per scenario, while sufficient for consistency verification as demonstrated by the low standard deviation values in Table 4, does not constitute a statistically large sample. Future studies should consider a greater number of repetitions and the application of inferential statistical tests to validate performance differences, particularly for margins below 2%.

5. Conclusion and Recommendations

This study evaluated the performance and efficiency of NTFS, exFAT, and EXT4 *file systems* on SSD storage using a multi-workload benchmarking approach across five distinct scenarios: Sequential Read, Sequential Write, Random Read, Random Write, and Small File Management. The results demonstrate that no single *file system* is universally superior across all workload types. exFAT tended to produce higher throughput on sequential workloads (544 MB/s Sequential Read; 495 MB/s Sequential Write), though differences with EXT4 in these scenarios were marginal and should be interpreted cautiously. exFAT also demonstrated the best resource efficiency with the lowest storage overhead (0.2%) and CPU utilization (4.6%). EXT4 showed clear superiority on random workloads (44,700 IOPS Random Read; 34,500 IOPS Random Write) and small file management (101.2 seconds for 10,000 files). NTFS performed consistently across all scenarios, occupying a middle position in performance while recording the highest CPU utilization among the three *file systems* evaluated.

The primary contribution of this study is its integrated evaluation framework combining throughput, IOPS, latency, CPU utilization, and storage overhead across five workload types for all three *file systems* within a uniform virtualized environment. Unlike most prior studies that compared only two *file systems* or evaluated limited performance dimensions (Sterniczuk, 2022; Oyewole & Joseph, 2024), this study provides a comprehensive characterization enabling workload-specific *file system* selection decisions. The simultaneous inclusion of exFAT alongside NTFS and EXT4 — a combination not previously reported in the benchmarking literature — represents a further methodological contribution that extends the empirical basis for cross-platform *file system* evaluation.

Based on the findings, the following recommendations are offered for practitioners:

1) exFAT is recommended for use cases dominated by large sequential file transfers, such as multimedia storage, data archiving, and external storage media, particularly when cross-platform compatibility (Windows, macOS, Linux) and resource efficiency are priorities.
2) EXT4 is recommended for server environments, databases, virtual machine hosting, and operating systems where random I/O performance and metadata management efficiency are critical requirements.
3) NTFS remains appropriate for Windows-native environments requiring security features, journaling

reliability, and broad enterprise compatibility, particularly where cross-platform operation is not required.

Several limitations of this study should be acknowledged. All testing was conducted within an Oracle VirtualBox virtualization environment; results may be influenced by hypervisor abstraction overhead and may not fully replicate the performance characteristics observed on physical hardware. Only one SSD capacity configuration (50 GB virtual disk) and one guest operating system (Ubuntu Server 22.04 LTS) were used, which may limit the generalizability of findings to other hardware and software configurations. Additionally, the use of three repetitions per scenario, while sufficient to demonstrate measurement consistency as shown in Table 4, does not constitute a statistically large sample for inferential analysis. Future research should address these limitations through the following directions:

1) Conducting experiments on physical SSD hardware to eliminate hypervisor overhead and validate the results obtained in this virtualized environment.
2) Including different SSD interface types (SATA vs. NVMe) to examine whether interface bandwidth constraints influence relative *file system* performance rankings.
3) Incorporating more complex and diverse workload scenarios, including mixed read/write workloads and application-level benchmarks representative of database, web server, and virtualization use cases.
4) Reporting inferential statistical measures — including standard deviation, confidence intervals, and significance tests — alongside mean values to enable more rigorous performance claims, particularly for margins below 2%.
5) Exploring advanced EXT4 reliability features such as copy-on-write snapshot mechanisms proposed by Dani *et al*. (2024) to evaluate their impact on both performance and storage efficiency in production-oriented environments.

References

Dakic, V., Kovac, M., & Videc, I. (2024). High-performance computing storage performance and design patterns—Btrfs and ZFS performance for different use cases. *Computers*, *13*(6), 139. https://doi.org/10.3390/computers13060139

Dani, A., Mangade, S., Nimbalkar, P., & Shirwadkar, H. (2024). *Next4: Snapshots in Ext4 file system* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2403.06790

Djordjevic, B., & Timcenko, V. (2011, August). *Ext4 file system performance analysis in Linux environment*. Paper presented at the 11th WSEAS International Conference on Applied Informatics and Communications (AIC'11, BEBI'11), Wisconsin, USA.

Farizy, S., & Gunawan, T. (2020). Analisa performa sistem berkas Ext4 pada kondisi tervirtualisasi. *Sainstech: Jurnal Penelitian dan Pengkajian Sains dan Teknologi*, *30*(1), 1–6. https://doi.org/10.37277/stch.v30i1.726

Fawwauzy, Z. R., Albani, T. E., & Rilvani, E. (2025). Perbandingan teknik pada struktur sistem file Windows & Linux. *Repeater: Publikasi Teknik Informatika dan Jaringan*, *3*(1), 69–79. https://doi.org/10.62951/repeater.v3i1.343

Heeger, J., Yannikos, Y., & Steinebach, M. (2022). An introduction to the exFAT file system and how to hide data within. *Journal of Cyber Security and Mobility*, *11*(2), 239–264. https://doi.org/10.13052/jcsm2245-1439.1125

Hines, J., Cunningham, N., & Alférez, G. H. (2023). Performance comparison of operations in the file system and in embedded key-value databases. In *Science and Information Conference* (pp. 386–400). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-37963-5_27

Jaffer, S., Maneas, S., Hwang, A., & Schroeder, B. (2019). Evaluating file system reliability on solid state drives. *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC '19)*, 783–798.

Jo, I. (2024). Toward ultra-low latency SSDs: Analyzing the impact on data-intensive workloads. *Electronics*, *13*(1), 174. https://doi.org/10.3390/electronics13010174

Oyewole, O. O., & Joseph, J. F. (2024). A comparative analysis of EXT4 and NTFS: Performance, security, and reliability features. *Multidisciplinary Journal of Engineering, Technology and Sciences*, *1*(1).

Rahman, M. A. K., Prayudi, Y., & Ramadhani, E. (2025). Analisis perbandingan jumlah file carving untuk format ekstensi NTFS, FAT32 dan exFAT. *JIIP: Jurnal Ilmiah Ilmu Pendidikan*, *8*(2), 1969–1973. https://doi.org/10.54371/jiip.v8i2.6994

Ramadan, A. (2021). A Docker's storage performance evaluation: Impact of backing file systems. *Journal of Intelligent Systems and Internet of Things*, *3*(1), 8–17. https://doi.org/10.54216/jisiot.030101

Shaikh, S. (2024). *Billion-files file systems (BfFS): A comparison* [Preprint]. arXiv. https://arxiv.org/abs/2408.01805

Sterniczuk, B. (2022). Comparison of EXT4 and NTFS filesystem performance. *Journal of Computer Sciences Institute*, *25*, 297–300. https://doi.org/10.35784/jcsi.3004

Yadgar, G., Gabel, M., Jaffer, S., & Schroeder, B. (2021). SSD-based workload characteristics and their performance implications. *ACM Transactions on Storage*, *17*(1), 1–26. https://doi.org/10.1145/3423137

Zhan, Y., Hu, H., Yang, X., Cao, Q., Jiang, H., Wang, S., & Yao, J. (2025). Rethinking the request-to-IO transformation process of file systems for full utilization of high-bandwidth SSDs. *Proceedings of the 23rd USENIX Conference on File and Storage Technologies (FAST '25)*, 69–86. https://www.usenix.org/conference/fast25/presentation/zhan.

How Cites

Wahyudin, W., Permana, A., & Everhard, J. (2026). Performance and Efficiency Analysis of NTFS, exFAT, and EXT4 on SSD Using a Multi-Workload Approach. *Computer Journal*, *4*(2), 81-92. https://doi.org/10.58477/cj.v4i2.476.

Publisher's Note