RAC: Revista Angolana de Ciências
E-ISSN. 2664-259X
Vol. 7, No. 2. e070209. July-December, 2025 (Continuous Flow Publication)

Comparative Performance Analysis of NFS and SMB/CIFS File Systems Integrated into Windows Server DFS

Análise comparativa de desempenho dos sistemas de arquivos NFS e SMB/CIFS integrados ao DFS do windows server

Análisis comparativo del desempeño de los sistemas de archivos NFS y SMB/CIFS integrados al DFS de Windows Server

Adilson José da Silva Silvério[1]
https://orcid.org/0009-0004-4305-2449

Lázaro Emílio Makili[2]
https://orcid.org/0000-0002-7371-1407

RECEIVED: 17 February, 2025 | **ACCEPTED**: 01 September, 2025 | **PUBLISHED**: 25 November, 2025

How to cite Silvério, A., Makili, L. (2025). Análise comparativa de desempenho dos sistemas de arquivos NFS e SMB/CIFS integrados ao DFS do windows server. *RAC: Revista Angolana de Ciências, 7(*2), e070209. https://doi.org/10.54580/R0702.09

ABSTRACT

Distributed file systems (DFS) are technological resources that enable secure and efficient information sharing and access within a network infrastructure. Since the adoption of these types of systems is vital for an institution, and considering the importance of choosing the right protocol for infrastructure reliability and performance, this study aims to comparatively analyze the operation and performance of Network File System (NFS) and Server Message Block (SMB)/Common Internet File System (CIFS) file systems integrated with the Windows Server DFS service, analyzing variables such as transfer rate, CPU (Central Processing Unit) utilization, and threads under different workloads. The test environment was designed based on the network infrastructure of Universidade Katyavala Bwila, located in Benguela province, Angola, which uses a client-server architecture. Based on the results presented, the system's responsiveness was determined for different operations (write, rewrite, read, reread, and others) on files and records of different sizes. Similar performance was observed between the protocols in intensive load tests, with minor variations in throughput for specific operations. This information provides relevant information for network administrators, experts, and the scientific community in defining file sharing policies and choosing the most appropriate protocol for deployments in DFS environments.

Keywords: Distributed File System (DFS); Performance; Analysis; Windows Server; NFS; SMB/CIFS.

[1]Instituto Politécnico da Universidade Katyavala Bwila. Department of Teaching and Research in Computer Science, Angola. p.eng.ajs.silverio@gmail.com

[2] Instituto Politécnico da Universidade Katyavala Bwila. Department of Teaching and Research in Computer Science, Angola.makili_le@yahoo.com

RESUMO

Os sistemas de arquivos distribuídos (DFS) são recursos tecnológicos que permitem a partilha e acesso a informações dentro de uma infra-estrutura de rede de forma segura e eficiente. Na medida em que a adopção destes tipos de sistema é de vital importância para uma instituição e considerando a importância da escolha adequada do protocolo para a confiabilidade e o desempenho da infra-estrutura, o estudo em causa tem como propósito analisar comparativamente o funcionamento e desempenho dos sistemas de arquivos Network File System (NFS) e Server Message Block (SMB)/Common Internet File System (CIFS) integrados ao serviço DFS do Windows Server, analisando variáveis como taxa de transferência, utilização de CPU (Central Processing Unit) e threads sob diferentes cargas de trabalho. O desenho do ambiente de teste teve como base a infra-estrutura de rede da Universidade Katyavala Bwila situada na província de Benguela-Angola, assente na arquitectura cliente-servidor. Com base nos resultados apresentados, permitiu-se conhecer a capacidade de resposta do sistema de acordo às diferentes operações realizadas (escrita, reescrita, leitura, releitura e outras) em arquivos e registos de diferentes tamanhos, onde indicaram um desempenho semelhante entre os protocolos em testes de carga intensiva, com pequenas variações na taxa de transferência em operações específicas. Estas informações fornecem subsídios relevantes para administradores de redes, especialistas e a comunidade científica, na definição de políticas de compartilhamento de arquivos e na escolha do protocolo mais adequado para implantações em ambientes DFS.

Palavras-chave: Sistema de arquivo distribuído (DFS); Desempenho; Analise; Windows Server; NFS; SMB/CIFS.

RESUMEN

Los sistemas de archivos distribuidos (DFS) son recursos tecnológicos que permiten compartir y acceder a información dentro de una infraestructura de red de forma segura y eficiente. A medida que la adopción de este tipo de sistemas es de vital importancia para una institución, y considerando la relevancia de elegir el protocolo adecuado para garantizar la confiabilidad y el rendimiento de la infraestructura, el presente estudio tiene como propósito analizar comparativamente el funcionamiento y desempeño de los sistemas de archivos Network File System (NFS) y Server Message Block (SMB)/Common Internet File System (CIFS) integrados al servicio DFS de Windows Server, analizando variables como la tasa de transferencia, la utilización de la CPU (Central Processing Unit) y los threads bajo diferentes cargas de trabajo. El diseño del entorno de prueba se basó en la infraestructura de red de la Universidade Katyavala Bwila, ubicada en la provincia de Benguela (Angola), sustentada en una arquitectura cliente-servidor. Con base en los resultados presentados, fue posible conocer la capacidad de respuesta del sistema de acuerdo con las distintas operaciones realizadas (escritura, reescritura, lectura, relectura y otras) en archivos y registros de diferentes tamaños, donde se indicó un rendimiento similar entre los protocolos en pruebas de carga intensiva, con pequeñas variaciones en la tasa de transferencia en operaciones específicas. Esta información proporciona aportes relevantes para administradores de redes, especialistas y la comunidad científica, en la definición de políticas de compartición de archivos y en la elección del protocolo más adecuado para implementaciones en entornos DFS.

Palabras clave: Sistema de archivos distribuido (DFS); Rendimiento; Análisis; Windows Server; NFS; SMB/CIFS.

INTRODUCTION

The acceleration of digitalisation and computerisation of services means that many public or private institutions resort to the implementation of applications (web, mobile, desktop) or network and service infrastructure to meet their needs. It is in this context that the importance or necessity of implementing Distributed File

Systems (DFS) in institutions arises, with the objective of improving file sharing and storage systems.

The need to share resources within a computational system arises due to the economy or nature of certain applications. In such cases, it is necessary to facilitate the sharing of long-term devices and their data. This can be achieved through distributed file systems. “A distributed file system allows users of physically distributed computers to share data and storage resources using a common file system” (Alangeh, 2021). According to Coulouris et al. (2013), a distributed file system enables programs to store and access remote files exactly as if they were local, allowing users to access files from any computer on a network. The performance and security in accessing files stored on a server should be comparable to files stored on local disks.

Rani et al. (2014) assert that the primary objective of a distributed file system is to provide a common view of the centralised file system, even when it has a distributed implementation, as can be observed in Figure 1.

DFS may offer several advantages, including:

- Enabling multiple users to access or store data;
- Permitting remote data sharing;
- Improving the capacity to resize data and enhancing data exchange capabilities;
- Providing data transparency even in the event of server or disk failure;
- Improving file availability, access time, and network efficiency.

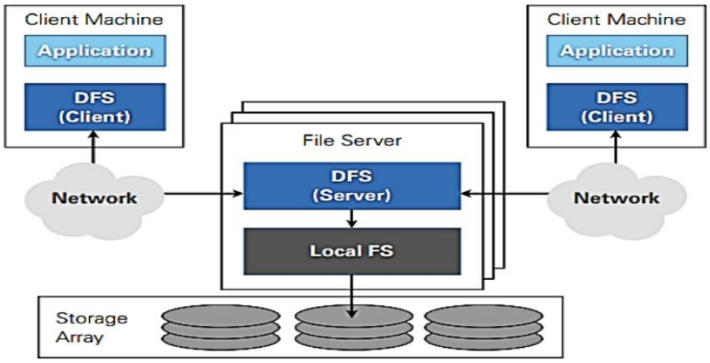


Figure 1 – Generic architecture of distributed file systems.

Source: Alangeh (2021).

Sie et al. (2019) state that according to how file storage is managed, there are two concepts of distributed file systems: client-server model and peer-to-peer (P2P) model.

There are various types of distributed file systems, including the classic NFS and others such as SMB/CIFS, Hadoop Distributed File System (HDFS), Andrew File System (AFS), Google File System (GFS), GlusterFS, and others (Bžoch, 2012). Among these, some are OpenSource and others proprietary, and they all allow for the implementation of systems that can better handle files. Seth (2023) states that among the numerous protocols designed for this purpose, SMB/CIFS and NFS are two of the most widely implemented.

In particular, the network infrastructure under study utilises Linux and Windows operating systems for the implementation of file management services under a client-server architecture. In this context, the focus of the research will be on the NFS and SMB/CIFS file systems as they are the predominant file sharing systems in these operating systems.

To meet the requirements of the network infrastructure restructure of the institution under study, the growth in data volume, the increase in the number of users, and the need for fast and reliable access to files, it becomes essential to have detailed information obtained through testing or performance analysis. This information will allow responses to the question of how the NFS and SMB/NFS systems behave when subjected to different operations, under various conditions related to workload, CPU utilisation, and concurrency.

The possible responses will enable the optimisation of network resources and an understanding of the behaviour of the aforementioned systems, evaluating their performance under different operations (write, rewrite, read, reread, random write and read) under varying workloads, CPU (Central Processing Unit) utilisation, and concurrency, with transfer rate as the reference.
In light of this, it is proposed to conduct a comparative analysis between the NFS (version 4.1) and SMB/CIFS (version 3.1.1) file systems, in order to assist in defining file sharing policies and selecting the most appropriate protocol for deployments in DFS environments.

Related work

Dakic et al. (2024) state that file systems play a critical role in optimising performance, as they directly determine read and write speed, aligning with the specific requirements of the environment in question. In this context, Honwadkar (2011) emphasises that the use of technologies such as Redundant Array of Independent Disks (RAID), Storage Area Networks (SANs), and Network Attached Storage (NAS) can make the file sharing, storage, and access process more efficient.

The study conducted by Sie et al. (2019) presents a practical implementation and performance analysis of a distributed file system (DFS) using the NFS protocol in a Windows environment, with a server on Windows Server 2008 and a client on Windows Vista/7, connected via an ad hoc network. The IOzone benchmarking tool provided quantitative data resulting from an evaluative process on read, write, rewrite, and reread operations on files ranging from 1 MB to 2 GB. The study identified that performance is better with smaller files, degrades with larger files due to cache limitations, and can be optimised with 32 KB transfer blocks. The study presents a limited scope, opening the door for a study that aims to compare NFS with other protocols such as SMB/CIFS, as proposed in the present work. Lee et al. (2021), in their study, investigate and compare various distributed file systems, such as Ceph, GlusterFS, Luster, and EOS for data-intensive environments, starting from two layouts: distributed and RAIN (Reliable Array of Independent Nodes). The distributed layout can use all disk capabilities, but does not provide parity calculations, which leads to data loss in case of disk failure, while RAIN cannot fully utilise disk capacity, but parity calculations provide an additional layer of protection.

Although the distributed layout, based on horizontal storage, does not provide parity mechanisms, the replication strategy can be adopted as a complementary approach to reinforce data reliability and integrity. On the other hand, Kumar (2019) argues that optimising a file system for a specific workload can compromise its performance in different scenarios, highlighting the importance of parameters such as network, disk configurations, number of servers and clients, and workload characteristics, due to their critical impact on final performance.

Zhao and Yuan (2014) propose a model based on the Analytic Hierarchy Process (AHP) for selecting the most relevant performance factors in distributed file systems. Among the factors identified as most influential are: the number of servers, the type of storage disks, the number of threads, and the type of storage connection. Although the model was originally applied to the Lustre file system, the results obtained can be adapted for a comparative analysis between other systems, such as NFS and SMB/CIFS.

Seth (2023), in his analysis, emphasises that the choice between SMB and NFS for file sharing in Linux environments depends on a series of factors, including performance, security, compatibility, and administrative preferences. SMB offers a rich feature set, is secure, and is compatible with Windows, which stands out in mixed operating system networks and business environments. NFS, with its optimised performance and native integration with Linux, is more suitable for homogeneous Linux implementations and high-performance applications.

Technical Approach

The evaluation of systems such as DFS requires critical consideration of the addressing scheme and server specifications, particularly random access memory (RAM), processor, and storage capacity (HDD). These components, whose functions are described below, prove essential for performance analysis, as demonstrated in Tables I and II:

- Server 1 – this server will have Windows Server 2019 installed, with configuration of DNS (*Domain Name System*), DFS Namespace, and DFS Replication services. According to Microsoft Learn (2025a), namespaces are a service that allows grouping shared folders located on different servers into one or more logically structured namespaces. This makes it possible to give users a virtual view of shared folders, where a single path leads to files located on multiple servers, and with the *replication* service, replication will occur.

- Servers 2 and 3 – these servers will have *Windows Server* 2019 installed and DFS Replication service configured; they will function exclusively for data replication, acting as support for the namespace server.

- PCs 1, 2, and 3 – these devices will function as test platforms, where the IOZone tool will be configured and used to run the tests. IOZone is a benchmarking tool for file systems, allowing testing of file I/O performance for the following operations: *read, write, reread, rewrite, read backwards, read strided, fread, fwrite, random read, pread, mmap, aio_read, aio_write*.

Table I – Software and addressing of the test environment.

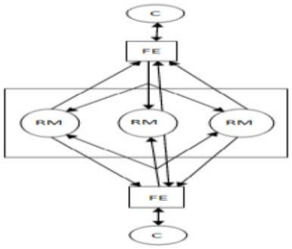
Device (service)	**Softwares**	**IP**	***Subnet-mask***	**DNS**
Server 1 (DNS e Namespace)	Windows Server 2019	10.10.0.1	255.255.255.0	10.10.0.1
Server 2 (Replica 1)	Windows Server 2019	10.10.0.2	255.255.255.0	10.10.0.1
Server 3 (Replica 2)	Windows Server 2019	10.10.0.3	255.255.255.0	10.10.0.1
PC 1 (Cliente)	Windows 10+IOZone	10.10.0.10	255.255.255.0	10.10.0.1
PC 2 (Client)	Windows 10+IOZone	10.10.0.11	255.255.255.0	10.10.0.1
PC 3 (Client)	Windows+IOZone	10.10.0.12	255.255.255.0	10.10.0.1

Table II - Characteristics of the computers that form part of the test environment.

Computer	**RAM**	**Processor**	**HDD**
Namespace Server	8G	Core i5; 2.80 GHz; 6 core; 6 logical processors.	1TB
Replica server	8G	Core i5; 2.80 GHz; 6 core; 6 logical processors.	1TB
Client	8G	Core i5; 2.80 GHz; 6 core; 6 logical processors.	1TB

For the evaluation of the proposed system, it was necessary to define a test infrastructure based on the client-server architecture and the TCP/IP model along with the protocols, techniques, type of replication, and horizontal storage mechanism, i.e., using the local capacity of the servers to store files.

Replication consists of two types of solutions, namely synchronous and asynchronous. Noor et al. (2019) state that “synchronous or active solution will update two replicas at the same time and revert if one of them fails. Some of the benefits of this type of solution are high availability, automatic failover, and minimal data loss” (p. 1299). Dave and Raghuvanshi (2012), on the other hand, state that in active replication, each client request is processed by all servers. This requires the process to be deterministic. Deterministic means that, given the same initial state and sequence of requests, all processes will produce the same sequence of responses and end in the same final state. For this reason, synchronous replication was adopted for the test infrastructure, thereby increasing the system's fault tolerance level.



C: *Client.*

FE: *Front-End.*

Figure 2 – Active replication.
Source: Dave & Raghuvanshi (2012, p. 128).

Furthermore, the devices involved in the replication process, regardless of the form or solution adopted, need to communicate. For this to happen, the adoption of specific protocols is crucial. For the configuration of the test infrastructure, the Remote Differential Compression (RDC) protocol was used; according to Microsoft Learn (2025b), the server management operating systems of the Windows family provide the DFS service that allows the creation of a virtual management and replication space for distributed files as an efficient multiple master mechanism, enabling the synchronisation of folders between servers, using the replication algorithm or protocol known as RDC.

RDC detects changes in data within a file and allows distributed file system replication to replicate only the changed file blocks instead of the entire file, which generates efficiency in larger folders or files. As can be observed in Figure 3, which illustrates the mode of operation between a replication group, group members, and replicated folders within a DFS that uses RDC.

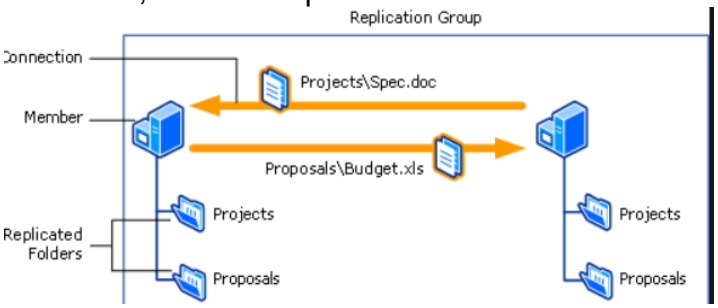


Figure 3 – Relationship between replication group, members, and replicated folders.
Source: Microsoft Learn (2025b).

According to Noor et al. (2019), replication techniques have been successfully implemented for distributed computing systems and allow these systems to remain distributed while increasing their availability and performance on a large scale, where the system is capable of operating in the presence of failures without user intervention, tolerating failures that may occur in the distributed computing environment. The proposed system will make use of this technique, the Two-Replica Distribution Technique (TRDT), to better replicate data, allowing two replica nodes with the same capacity as the namespace server. Noor et al. (2019, p. 1301) state that for TRDT to function “each node must have equal storage capacity and all data must have two replicas. For N sets of nodes (N = 2n), where each set consists of two nodes.” As can be observed in the figure below.

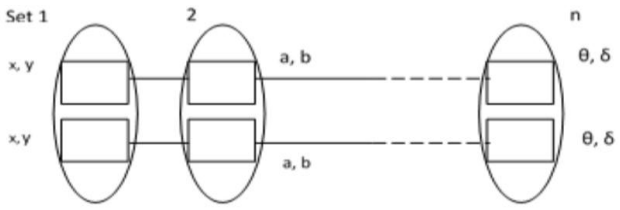


Figure 4 - Data replica distribution technique when N=2n.
Source: Noor et al. (2019, p. 1301).

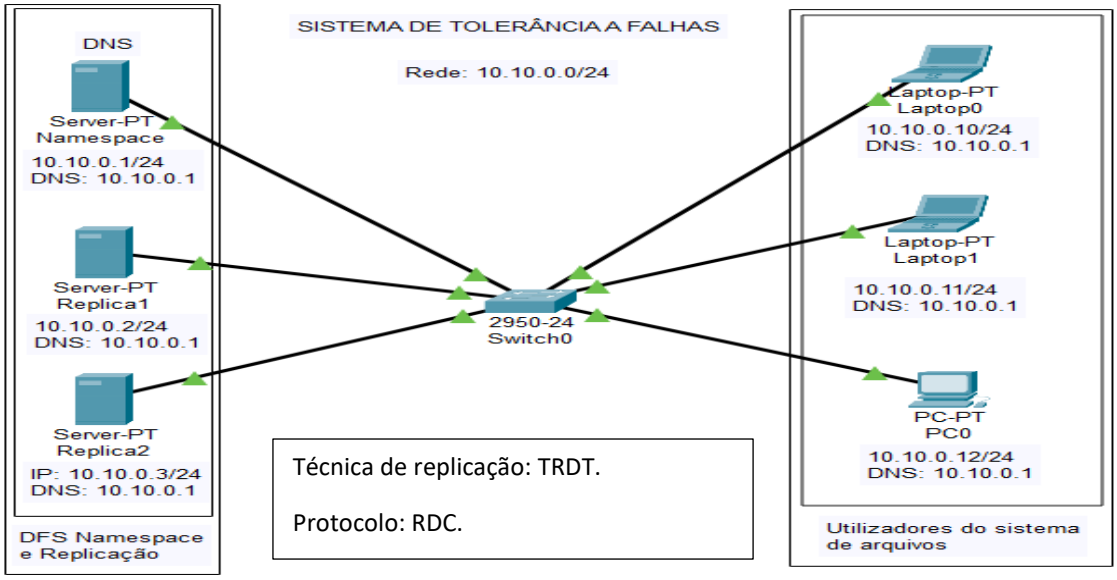


Figure *5* – Design of the test environment.

After configuring DFS, the following figure shows the structure created on the namespace server (\\ukb.ed\dfs) and the shared folders within it (nfs-sharing and smb-sharing). In this figure, it is also possible to observe the replication folders that are located on the replica server, i.e., for the nfs-sharing and smb-sharing folders

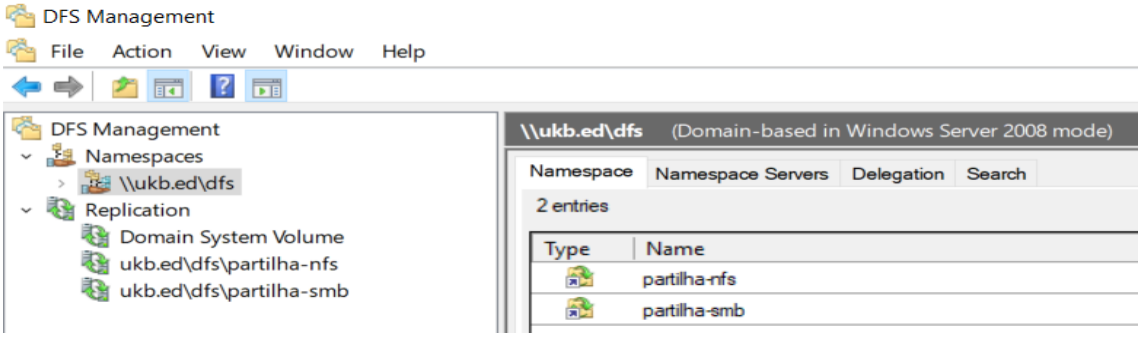


Figure *6* – DFS structure on the namespace server.

In Figure 7, it is possible to observe the DFS structure on the replica server, which functions as a secondary server in case of failure of the master/main server, containing the replication folders.

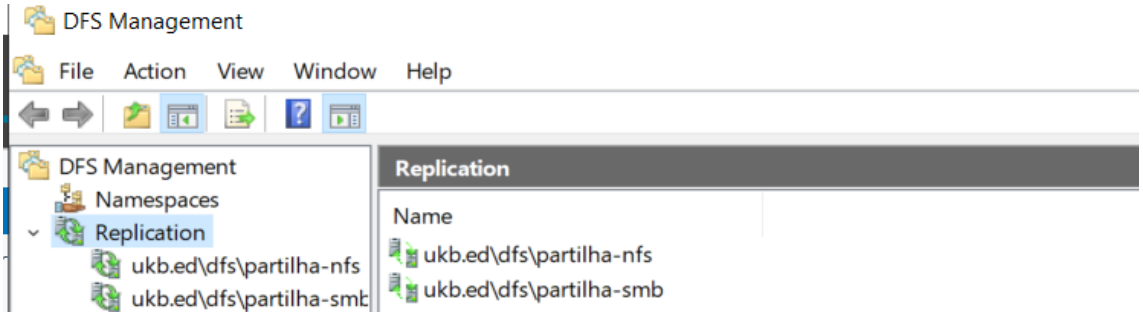


Figure 7 – *DFS structure on the replica server*.

Test methodology

To analyse the figures and tables that follow as results of the tests conducted with IOZone, it is important to understand three elements, as per Farncomb (2015):

- File size: this is simply the total size of the file being written or read from the disk;
- Record size: the file is divided using a specific record size. A 64 MB file can be composed of 8 records of 8 MB or 4 records of 16 MB. Therefore, some of the graphs are displayed as completely zero; some of the file sizes tested are not large enough for all tested record lengths, for example, it is not possible to have a 512 KB file created from an 8 MB record;
- Data transfer units[3]: the speed at which a file of a certain size and record size was written or read from disk can be measured in KBps, MBps, or GBps, or simply transfer rate. [1 KBps = 1,024 bytes (or 8,192 bits). A low number of KBps or MBps (transfer rate) means that the operation will take much longer to complete.]

IOZone provides a set of commands that can be used to measure and analyse the performance of different file systems, hiding some results considering the file size defined during the test, showing only the most relevant data.

Tables III and IV present the commands used during the testing process, their functions, and also the specific commands used to measure the specific variables, according to the three types of tests performed, such as workload, performance by number of threads, and CPU utilisation, adopting KBps as the transfer rate unit.

The workload and CPU utilisation tests were conducted according to the following parameters: read and reread (read/re-read), write and rewrite (write/re-write), read backwards (read backwards), record rewrite (re-write-record), and random read and write (random read/write). For the present study, the evaluation of performance in read and write operations using temporary files from 64 KB to 2 GB (x-axis) and, in each file testing process, the records varied from 4 KB to 16 MB. For better data analysis, the files were classified according to size:

- Small files (4 KB - 512 KB);
- Medium files (1 MB - 16 MB);
- Large files (32 MB - 512 MB);
- Very large files (≥ 1 GB).

On the other hand, the thread tests were defined with the following analysis parameters: 4 processes, 512 KB files, and 4 KB records.

Tabela II – Commands selected for the tests in IOZone

Commands	Description
- a	Used to select the fully automatic mode. Produces output that covers all file operations tested for record sizes from 4k to 16M for file sizes from 64k to 512M.

[3] 1 KBps = 1,024 bytes (or 8,192 bits). A low KBps or MBps (transfer rate) means that the operation will take much longer to complete.

- i	Allows defining the type of test to be performed, varying from 1 to 12 (0=write/re-write 'write/rewrite', 1=read/re-read 'read/reread', 2=random-read/write 'random read/write', 3=read-backwards 'read backwards', 4=re-write-record 'read and write at a specific point', 5=stride-read 'read a file with accelerated access behaviour', 6=fwrite/re-fwrite 'user buffer usage for write and rewrite', 7=fread/re-fread 'user buffer usage for read and reread', 8=random_mix 'read and write of a file with accesses made to random locations within the file', 9=pwrite/re-pwrite, 10=pread/re-pread, 11=pwritev/re-pwritev, 12=preadv/re-preadv)
- n	Allows defining the minimum file size (in Kbytes) for automatic mode, ranging from 64K to 512M.
- g	Allows defining the maximum file size (in Kbytes) for automatic mode. Used when the computer has more than 512 MB of RAM.
- r	Used to specify the record size (4k to 16M), in Kbytes, to be tested. Can also be specified as -r #k (size in Kbytes) or -r #m (size in Mbytes) or -r #g (size in Gbytes)
- b	Allows creating a binary file in output format compatible with Excel.
- R	Generates Excel report. IOzone will generate a report compatible with Excel to standardise. This file can be imported with Microsoft Excel (space delimited) and used to create a graph of the file system performance.
- f	Used to specify the name of the temporary file for testing.
- F	Used to specify each of the names of the temporary files to be used in the transfer rate test. The number of names must equal the number of processes or threads specified.
- t	Allows the user to specify how many threads or processes should be active during the measurement.
- z	Used in conjunction with -a to test all possible record sizes.
- c	Includes close() in time calculations. It is used if the operating system's close() is not functioning during the test.
- e	Allows including flush (fsync, fflush) in time calculations
- w	Does not unmount the temporary files when finished using them. Leave them present in the file system.
-U	Allows defining the mount point to unmount and remount between tests. IOzone will unmount and remount this mount point before starting each test. This ensures that the buffer cache does not contain any test files.
-+ u	Will display CPU utilisation for each test performed.
-+ t	Activates network performance testing.
-+ m	Allows obtaining client configuration information for cluster testing.

Table IIIV - Metrics, file systems, and commands in IOZone.

Metrics	**File system**	**Device**	**Commands (executed in command line)**
Workload	SMB/CIFS	Client	iozone -a -i 0 -i 1 -i 2 -i 3 -i 4 -g 2G -R -b results-smb.xls \\ukb.ed\dfs\partilha-smb
	NFS	Client	iozone -a -i 0 -i 1 -i 2 -i 3 -i 4 -g 2G -R -b results-nfs.xls \\ukb.ed\dfs\partilha-nfs
Performance, specifying number of processes or *threads*	SMB/CIFS	Client	iozone -t 4 -i 0 -i 1 -i 2 -i 3 -i 4 -R -b performance-results-smb.xls \\ukb.ed\dfs\partilha-smb
	NFS	Client	iozone -t 4 -i 0 -i 1 -i 2 -i 3 -i 4 -R -b performance-results-nfs.xls \\ukb.ed\dfs\partilha-nfs
CPU utilisation	SMB/CIFS	Client	iozone -a -+u -i 0 -i 1 -i 2 -i 3 -i 4 -g 2G -R -b results-cpu-smb.xls \\ukb.ed\dfs\partilha-smb
	NFS	Client	iozone -a -+u -i 0 -i 1 -i 2 -i 3 -i 4 -g 2G -R -b results-cpu-nfs.xls \\ukb.ed\dfs\partilha-nfs

Results and Discussion
Workload Tests

As can be observed in Figure 8, which illustrates the application of the specific command for testing and the use of memory *cache.*

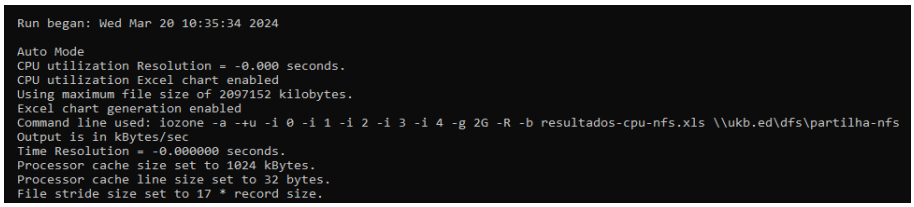


Figure 8 - Parameters for workload and CPU utilisation tests in IOZone.

The analysis of the performance data in the figures below (9 and 10) reveals significant differences between the NFS and SMB/CIFS protocols in write operations, varying according to file size and record size.

For small files, NFS shows relatively stable performance, but varies significantly between record sizes. For 64 KB files with a 64 KB record, it achieves up to 477,968 KBps, but at 256 KB it already exceeds 1.4 × 10^6 KBps, reaching 2,650,000 KBps for a 512 KB file with a 256 KB record. SMB/CIFS shows more irregular performance, with significant drops in 64 KB files with a 4 KB record, reaching 254,270 KBps, but at 128 KB there are fluctuations (max. ~1,170,000 KBps) in all small files. Notably, there are also interesting peaks in 256 KB records (1,820,000 KBps) and 512 KB records (2,450,000 KBps), but less consistent than NFS. For small files, NFS is more consistent and scalable, while SMB/CIFS has sharp variations, with good peaks but less predictability, which is because it has higher overhead per operation, explaining some instability with small records, but in some cases it exceeds NFS.

For medium files, NFS shows generally high and stable values, with peaks close to 4.6 × 10^6 KBps (16 MB file with 1 MB record) and very good performance with 64 KB to 1 MB records. SMB shows good performance with 1 to 16 MB records, with values in the 3.3-4.9 × 10^6 KBps range (4,800,000 KBps for a 16 MB file with a 1 MB record), despite a drop in 8 MB records (2,476,795 KBps for a 16 MB file with an 8 MB record versus NFS 4,026,443 KBps). For medium files, SMB/CIFS tends to outperform NFS in the best cases, offering higher maximum throughput, but NFS shows less variability and better stability.

For large files, NFS shows solid performance, with values between 3.3 and 4.6 × 10^6 KBps, maintaining good regularity and not showing sharp drops. SMB/CIFS reaches similar or slightly higher levels in some points (4,740,000 KBps for a 512 MB file with a 2 MB record). In this case, both protocols offer similar performance, but SMB/CIFS has a slight advantage in maximum throughput, while NFS is more stable.

For very large files, both suffer significant degradation, i.e., an abrupt drop in performance. NFS is more affected as files grow, but SMB/CIFS tends to be slightly superior, maintaining writing rates consistently above those of NFS, especially from 2 GB. This drop may be associated with cache and buffer limitations, protocol overhead, or structural limitations of the file system and network.

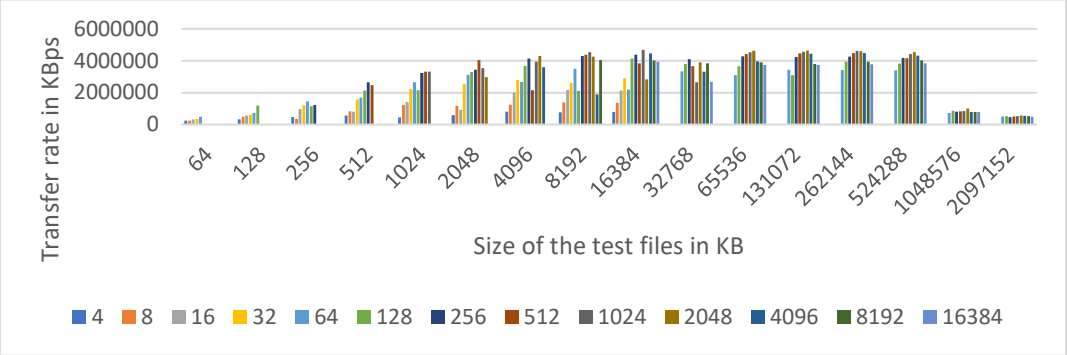


Figure 9 – Report of writing operation on NFS files.

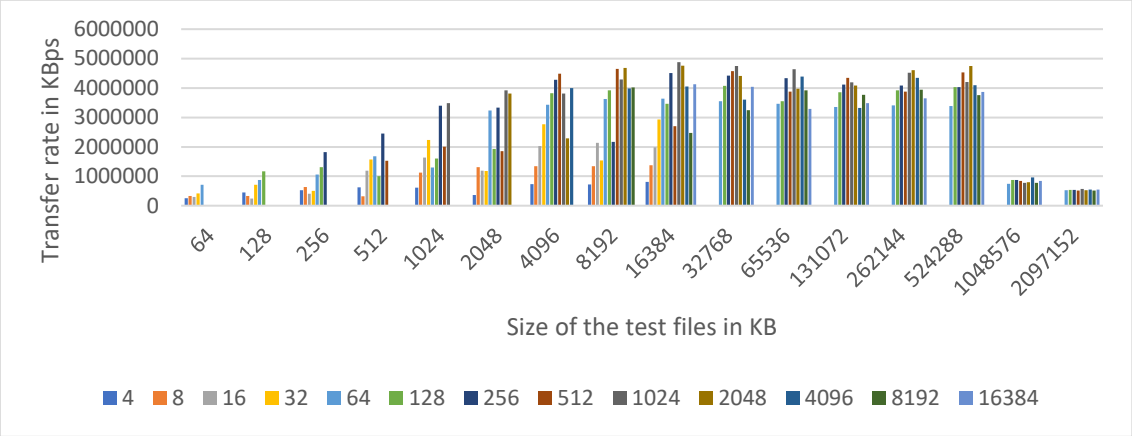


Figure 10 – Report of writing operation on SMB/CIFS files.

Analysing the figures below (11 and 12), it can be noted that for small files, NFS showed initially stable and increasing performance, standing out occasionally in a 64 KB file with a 64 KB record (7,191,011 KBps), but with fluctuations and sharp drops as record sizes increase, particularly in 256 KB files, where the value drops to 4,547,069 KBps (in a 64 KB record). SMB/CIFS, on the other hand, recorded higher values in smaller records (4 KB and 8 KB) and maintained a more linear performance curve in the 128 KB to 512 KB range, surpassing NFS in various points, such as in a complete read of a 128 KB file with a 128 KB record, demonstrating greater consistency. Thus, for workloads with small files and sequential access, the data suggests that SMB/CIFS may be the most efficient choice. On the other hand, NFS shows an advantage in specific scenarios, such as reading in 64 KB blocks.

For medium files, NFS achieved high values in small records, such as in 1 MB files with a 64 KB record (8,172,386 KBps), but shows greater variability as file size increases, with noticeable drops, for example, in a 2 MB file with a 256 KB record (4,121,553 KBps). SMB/CIFS offers more stable and predictable performance, reaching 8,774,635 KBps in a 2 MB file with a 128 KB record, and sustaining consistent values up to 16 MB files, making it more suitable for continuous workload scenarios.

For large files, NFS shows instability in 32-64 MB files, with reduced performance in a 32 MB file with a 128 KB record (4,848,197 KBps), but gradually stabilised in

larger volumes, converging to values close to SMB/CIFS. On the other hand, SMB/CIFS had an advantage in 32 MB files with a 128 KB record (5,308,191 KBps) and maintained slight superiority in 64 MB files. However, between larger files (128 MB and 512 MB), both protocols show equivalent performance, for example, in a 128 MB file with a 128 KB record, NFS reached 5,308,600 KBps versus 5,466,549 KBps on SMB. SMB/CIFS offers better balance between performance and stability for large files, while NFS requires more careful configuration, but can achieve similar performance.

For very large files, the performance of both protocols converges and becomes practically equivalent, with small alternating advantages. A detailed analysis shows that the advantages are minimal and alternating, while SMB/CIFS records slightly higher values in some scenarios (1 GB file with a 1 MB record), and in other cases, such as in a 1 GB file with a 128 KB record, SMB/CIFS recorded 5,853,721 KBps, while NFS obtained 5,769,185 KBps. In a 2 GB file with a 128 KB record, NFS reached 5,865,147 KBps, versus 5,981,038 KBps for SMB. NFS maintains parity or even small advantages in other scenarios, such as in 1 GB files with a 4 MB record (NFS obtained 6,448,775 KBps while SMB/CIFS had 6,437,136 KBps). These are marginal, confirming equivalence in handling very large files.

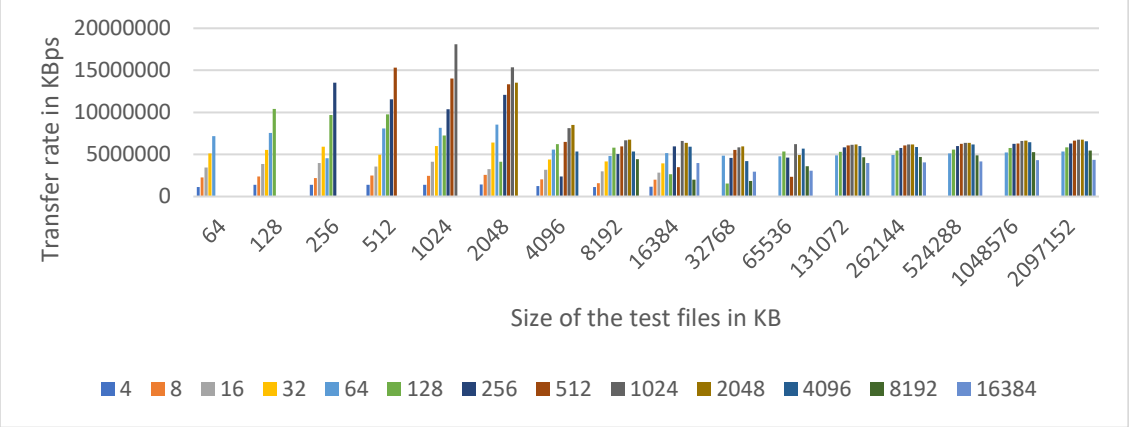


Figure 11 – Report of reading operation on NFS files.

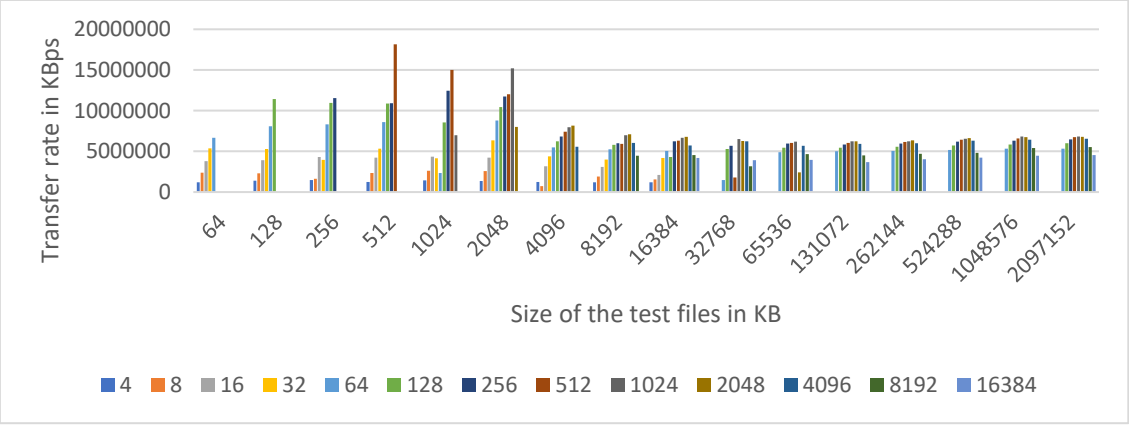


Figure 12 – Report of reading operation on SMB/CIFS files.

Thread Tests

```
Output is in kBytes/sec
Time Resolution = -0.000000 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.
Throughput test with 4 processes
Each process writes a 512 kByte file in 4 kByte records
```

Figure 13 – Parameters for performance tests in IOZone.

Figure 16 consolidates the data presented in Figures 14 and 15, allowing for clearer analysis, where it is observed that SMB/CIFS tends to be more robust in write operations and random accesses, making it advantageous in transactional application environments (databases, shared file systems with multiple clients writing and reading random blocks), while NFS demonstrates greater efficiency in repeated reads and reverse sequential accesses, which may be advantageous in read-intensive scenarios, such as HPC (High Performance Computing), where large volumes of data are read in predictable or repetitive patterns.

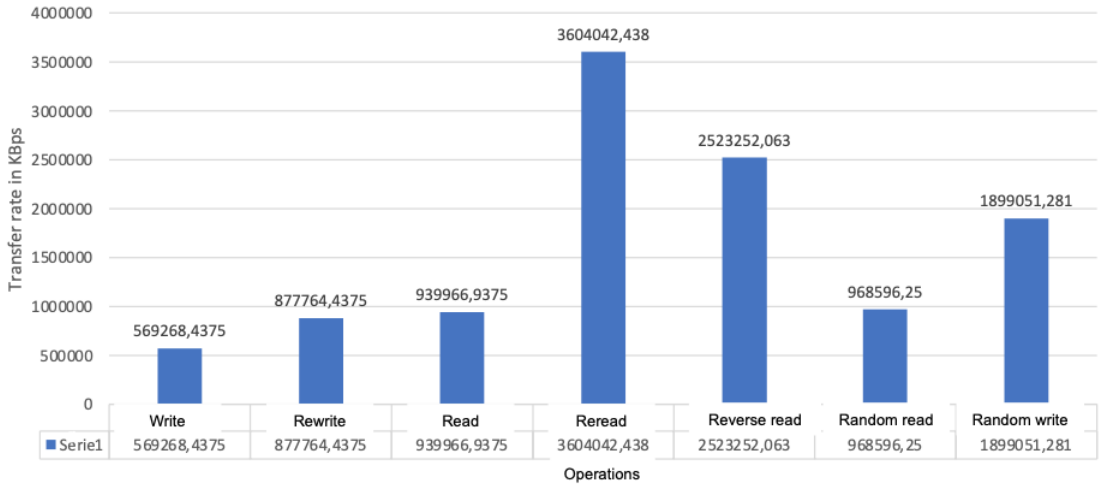


Figure 14 - Performance test report on NFS files using 4 concurrent processes.

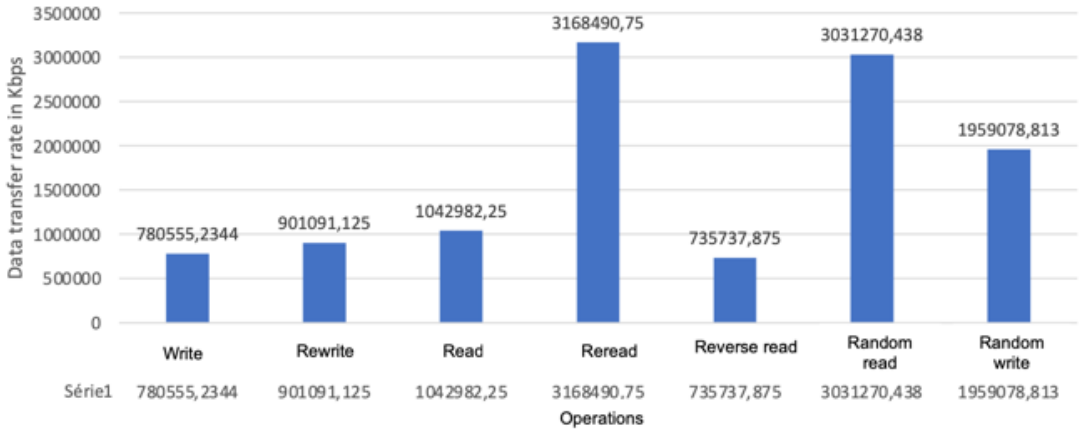


Figure 15 – Performance test report on SMB/CIFS files using 4 concurrent processes.

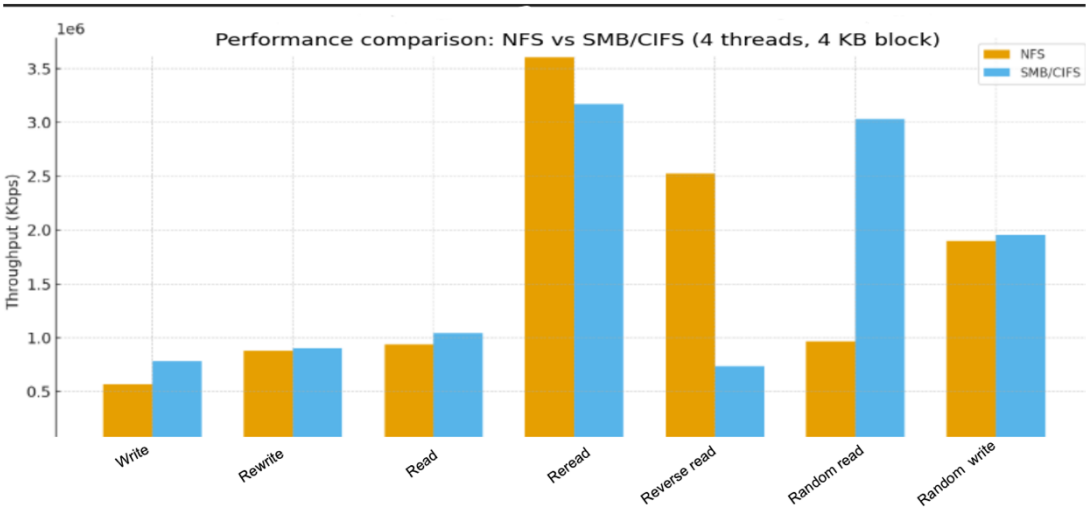


Figure 16 – Comparison between NFS and SMB/CIFS.

CPU Utilisation

In write operations, the SMB/CIFS protocol shows a slightly higher average global performance, although the overlap in standard deviations suggests comparable efficiency to NFS. NFS stands out in processing larger files, with less relative overhead, while SMB/CIFS shows better performance with medium-sized files.

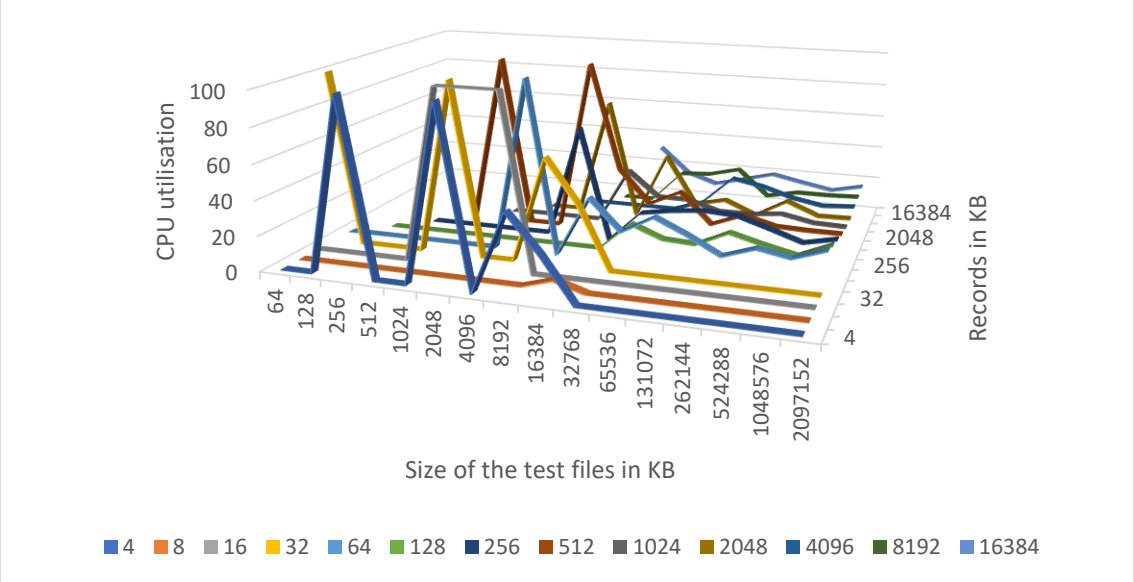


Figure 17 - CPU utilisation report, writing operation on NFS files.

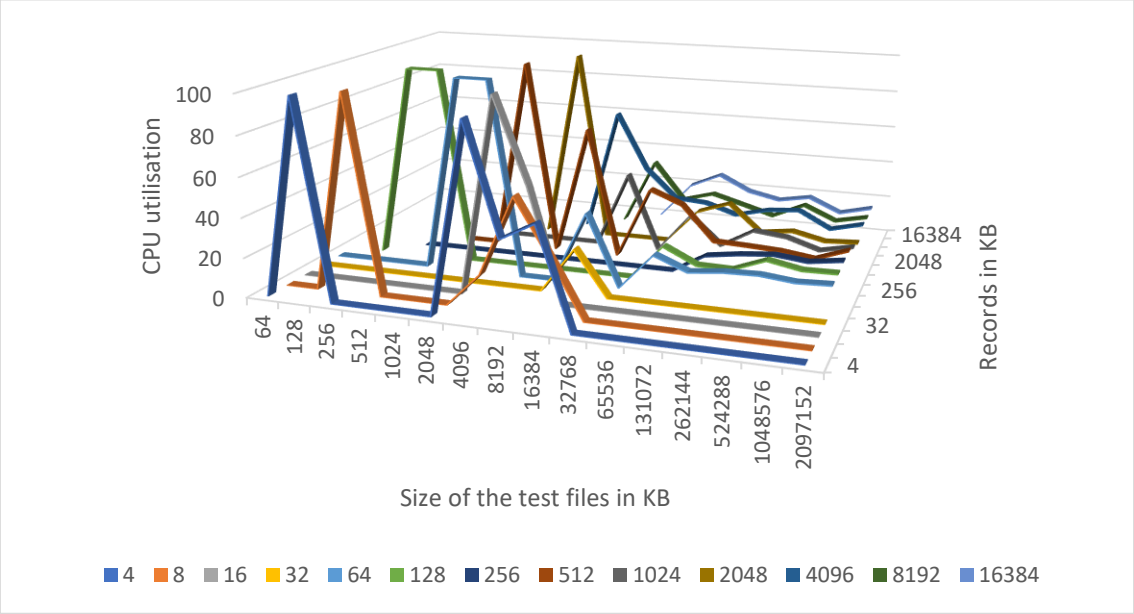


Figure 18 - CPU utilisation report, writing operation on SMB/CIFS files.

For read operations, SMB/CIFS generally shows a slightly higher average load than NFS (91.96% vs. 89.18% overall), particularly in medium (99.77% vs. 92.27%) and large files (84.67% vs. 81.96%), which reflects greater irregularity and occasional high load, probably due to protocol overhead. The differences are small (<5% in most cases), and NFS has comparable or superior performance in very large files and specific block sizes (e.g., 64 KB and 512 KB). Both protocols approach CPU saturation (95-100%) for files >32 MB, indicating a shared bottleneck in data transfer rate or *buffer*.

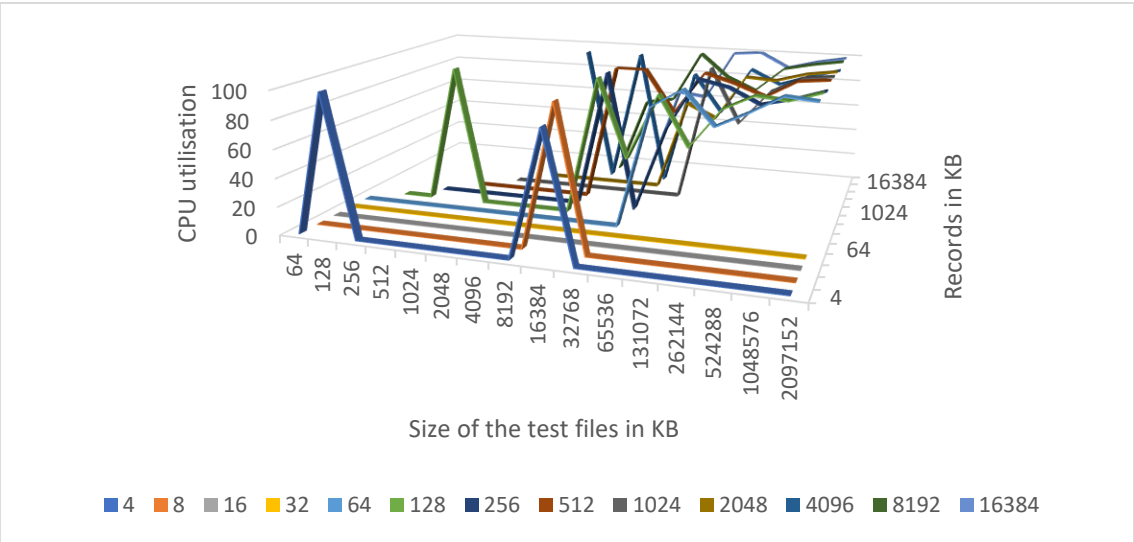


Figure 19 – CPU utilisation report, reading operation on NFS files.

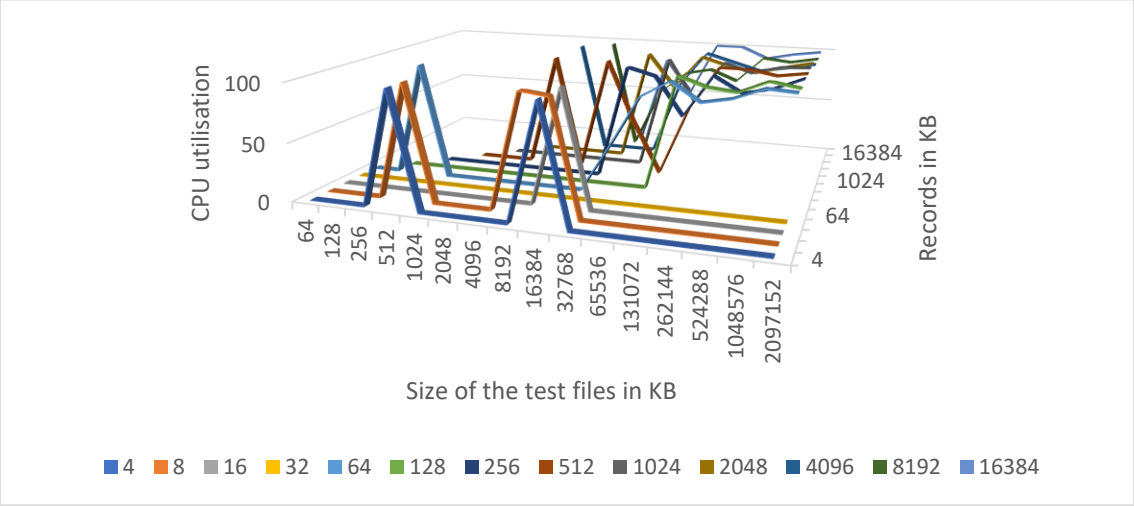


Figure 20 – CPU utilisation report, reading operation on SMB/CIFS files.

Conclusion

Therefore, after an exhaustive analysis of the test data, it can be observed that for the write process, NFS shows consistency and efficiency in small to medium files, suggesting better suitability in environments for incremental backups, log systems, and high-fragmentation applications. SMB/CIFS demonstrates better performance in medium and large workloads, achieving higher throughput peaks, a characteristic that makes it more appropriate in environments focused on transferring large volumes of data (multimedia, virtual machines).

Regarding reading, the results show that for practical applications, the choice should consider the access pattern, the predominant file size, and the network infrastructure. SMB/CIFS tends to offer a more balanced overall performance, particularly more efficient for large and very large files and for larger record sizes, being the preferred choice for scenarios involving files or records of larger size, ideal for file sharing in companies, media and entertainment industry, and big data. While NFS can be optimised to obtain competitive performance in specific scenarios, especially for smaller record sizes and certain medium file sizes, it is less consistent overall, ideal for web servers, application hosting, and virtualisation.

In performance tests with 4 processes, SMB/CIFS was shown to be more suitable for scenarios with intensive writes and random accesses, while NFS for scenarios with repeated reads and non-linear predictable patterns, although the choice between NFS and SMB should, therefore, consider the workload profile, such as in collaboration environments, application servers, virtualisation, databases, and offices (SMB/CIFS) and scientific environments, HPC, and *big data* oriented to reading (NFS).

Regarding CPU usage, NFS and SMB/CIFS exhibit similar performance in CPU consumption for write operations, with no clear protocol superiority. For small and medium files, both show high CPU utilisation, making it advisable to optimise performance with larger data record sizes or local storage. On the contrary, for large or very large files, both show efficiency, with CPU consumption varying between 0% and 20%. NFS stands out for efficiency in high-volume environments

with large and very large files, reducing CPU load, while SMB/CIFS may be more suitable in scenarios prioritising compatibility over minimal CPU utilisation. For better optimisation, it is recommended to adjust record sizes to ≥ 64 KB, to stabilise utilisation and reduce variability.

Despite the relevance of the findings for network administrators in Windows environments, for large-scale deployments, heterogeneous environments (Linux and Windows), or critical and real workloads (VDI, backup, databases, virtualisation, HPC), it is recommended to conduct new tests with native clients, optimised configurations (NFSv4.2, SMB Multichannel, RDMA), high-performance storage (SSD/NVMe, RAID), and complementary metrics (latency, IOPS, consistency).

This paper provides a solid basis for decision-making in Windows Server DFS environments, but reinforces the need for contextualised evaluations, considering the specific usage profile, available infrastructure, and institutional performance requirements.

REFERENCES

Alangeh, L. N. (2021). *Distributed file systems: A literature review* [Unpublished manuscript]. Department of Computer Engineering, Institute of Natural and Applied Sciences, Gazi University. https://www.researchgate.net/publication/355175965

Anderson, T. E., Peter, S., Canini, M., Kim, J., Kostić, D., Kwon, Y., Reda, W., Schuh, H. N., & Witchel, E. (2020). Performance and Availability via Client-local NVM in a Distributed File System. *14th USENIX Symposium on Operating Systems Design and Implementation*, 1011–1027. https://www.usenix.org/conference/osdi20/presentation/anderson

Bžoch, P. (2012). Distributed file systems: The state of the art and concept of Ph.D. thesis (Technical Report No. DCSE/TR-2012-02). University of West Bohemia, Department of Computer Science and Engineering. http://www.kiv.zcu.cz/publications/

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2013). *Sistemas Distribuídos: Conceitos e projeto*. Bookman.

Dave, S., & Raghuvanshi, A. (2012). Fault Tolerance Techniques in Distributed System. *International Journal of Engineering Innovation & Research*, *1*(2), 2277–5668.

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

Farncomb, J. (2015, 22 de Julho). *Windows NFS vs Linux NFS Performance Comparison*. RootUsers. https://www.rootusers.com/windows-nfs-vs-linux-nfs-performance-comparison/

Huang, H. H., & Grimshaw, A. S. (2011). Design, implementation and evaluation of a virtual storage system. *Concurrency and Computation: Practice and Experience*, *23*(4), 311–331. https://doi.org/10.1002/cpe.1644

Honwadkar, K. N., & Sontakke, T. R. (2011). Improvement in capacity and efficiency of network storage by configuring hard disk drives on nodes of a LAN. *International Journal of Computer Applications, 19*(3), 15–21. https://doi.org/10.5120/2338-3048

Kumar, S. (2019). *Performance modeling of a distributed file-system*. https://doi.org/10.48550/arXiv.1908.10036

Lee, J. Y., Kim, M. H., Shah, S. A. R., Ahn, S. U., Yoon, H., & Noh, S. Y. (2021). Performance evaluations of distributed file systems for scientific big data in fuse environment. *Electronics (Switzerland)*, *10*(12). 1471. https://doi.org/10.3390/electronics10121471

Microsoft Learn. (2025a, 16 de Agosto). *Visão geral de Namespaces DFS*. Learn. https://learn.microsoft.com/pt-br/windows-server/storage/dfs-namespaces/dfs-overview?tabs=server-manager

Microsoft Learn. (2025b, 16 de Agosto). *Visão geral sobre a Replicação do DFS*. Learn. https://learn.microsoft.com/pt-br/windows-server/storage/dfs-replication/dfsr-overview?source=recommendations

Noor, A. S. M., Zian, N. F. M., & Bahri, F. N. M. S. (2019). Survey on replication techniques for distributed system. *International Journal of Electrical and Computer Engineering (IJECE), 9*(2), 1298. https://doi.org/10.11591/ijece.v9i2.pp1298-1303

Sie, S., Kyaw, T., Moh, D., & Khaing, M. (2019). Analysis and optimization of distributed file system performance. *IJCIRAS*, *2*(3), 39–44. http://www.ijciras.com/PublishedPaper/IJCIRAS1317.pdf

Seth, V. (2023). Analyzing and comparing the performance of SMB and NFS protocols for efficient file sharing in Linux environments. *International Journal of Scientific Research & Engineering Trends, 9*(1), 1–4. https://ijsret.com/wp-content/uploads/IJSRET_V9_issue1_138.pdf

Zhao, T., & Yuan, H. (2014). Parameter Analysis Model of Distributed File Based on Ipmroved Analytic Hierarchy Process. *Asian Network for Cientific Information*, *13*(11), 1908–1911. https://doi.org/10.3923/itj.2014.1908.1911