International Journal of Advanced Research and Interdisciplinary Scientific Endeavours, Vol. 3(4), 2025
DOI: 10.61359/11.2206-2553

Workload-Driven Perspectives on Networked Filesystems: Benchmarking NFS/SMB and Version-Level Trade-offs

Pankaj Singh*

Email Correspondence*: pank_kiit@yahoo.com

[1] Independent Researcher, India.

Abstract:

This paper surveys network file system protocols with a systems focus on NFS and SMB/CIFS, explaining how transparency to applications interacts with protocol design, client caching, and failure handling in distributed storage stacks. It consolidates measurement methodology via widely used tools (Postmark for small-file, mail-server-like workloads and LADDIS/SPEC SFS for NFS server micro-operations), detailing metrics, load generation, and interpretability limits across client/server configurations. A versioned protocol comparison highlights, when NFSv4.1's stateful features and read delegations reduce request volume and improve throughput over NFSv3 for small-file access patterns, while also identifying workloads where added chattiness can degrade performance. Complementary production CIFS trace analysis quantifies real-world I/O volumes, read-write ratios, reopen distributions, and low sharing rates, yielding design implications for client-side caching, tiering, and write-optimized filesystems. Together, these benchmark-anchored and trace-driven perspectives provide practical guidance for performance evaluation and protocol selection in computer systems and storage research.

Keywords: Network File Systems, NFS, SMB/CIFS, Benchmarking, Performance Evaluation, Storage Protocols.

1. Introduction

Network file systems represent a fundamental abstraction in distributed computing, enabling transparent access to re- mote storage resources as if they were locally attached. The evolution of these systems spans several decades, with persistent files serving as the primary mechanism for data storage across diverse computing environments [1]. From early implementations in academic and corporate settings to modern cloud-native storage solutions, network file systems have consistently addressed the need for shared, centralized data management while maintaining application compatibility through transparent operation. The significance of network file systems extends beyond mere convenience; they enable critical infrastructure capabilities including simplified backup procedures, centralized administration, and collaborative workflows across shared workstations. In university environments and large enterprises, these systems form the backbone of data-intensive applications, supporting research computing, software development, and business operations. The transparency requirement that applications should operate identically on networked and local filesystems without modification has driven protocol design decisions with profound implications for performance, consistency, and reliability. This paper examines two dominant network file system protocols: Network File System (NFS), originating from the Unix ecosystem, and Server Message Block/Common Internet File System (SMB/CIFS), with roots in the DOS/Windows environment. Despite their divergent histories and architectural approaches, both protocols converge on similar

*Independent Researcher, India.

Article Timeline: Received: Nov 13, 2025; **Revised:** Nov 27, 2025; **Accepted:** Nov 27, 2025; **Published:** Nov 30, 2025

functionality while exhibiting distinct performance characteristics under various workloads. Understanding these characteristics requires systematic measurement methodologies that capture both controlled benchmark performance and real-world usage patterns. Our contribution lies in synthesizing multiple measurement perspectives: controlled benchmarking using established tools like Postmark and LADDIS, version-level protocol comparison focusing on NFSv3 versus NFSv4.1 improvements, and production trace analysis from CIFS deployments. This multi-faceted approach reveals workload-dependent trade-offs in protocol design, client caching effectiveness, and opportunities for storage stack optimization. The insights generated provide practical guidance for researchers and practitioners evaluating, deploying, or improving network file systems in diverse computing environments

2. Network File System Protocols: NFS and SMB/CIFS

Network file system protocols abstract the complexities of remote data access, presenting applications with familiar file system interfaces while handling network communication, caching, consistency, and failure recovery transparently. The two most prevalent protocols NFS and SMB/CIFS emerged from different computing traditions but share the common goal of location-transparent file access. NFS originated at Sun Microsystems in 1984 as part of the Unix 4.2 operating system development [1]. The initial design prioritized transparent operation with existing Unix applications, crash recovery capabilities, and reasonable performance. NFS version 2 (NFSv2), the first published specification, established a completely stateless protocol using UDP as its transport layer. This statelessness simplified server implementation and client recovery from server failures, clients could simply retransmit requests until the server responded. However, this design omitted file locking and provided only partial support for certain file operations like append-mode writes and removal of open files.

The evolution of NFS continued with NFSv3 in 1995, which introduced 64-bit file sizes, TCP transport support, and enhanced protocol requests for improved performance [2]. A more substantial architectural shift occurred with NFSv4 and NFSv4.1, which transitioned to a stateful protocol design [3], [4]. This statefulness enabled sophisticated features including built-in file locking, enhanced security mechanisms, and finer-grained control over client-side caching. The protocol's growing complexity reflected an evolving understanding of distributed storage requirements in increasingly networked computing environments.

SMB/CIFS presents a contrasting historical trajectory, beginning as an IBM-designed protocol for DOS operating systems in the 1980s. Originally operating over NetBIOS session layers and NetBIOS Frames (NBF) transport, SMB provided file, printer, and serial port sharing capabilities [5]. Microsoft's subsequent development of the protocol introduced numerous proprietary extensions, though some aspects were standardized through X/Open in 1992 [6]. The protocol's closed nature prompted reverse-engineering efforts, most notably the Samba implementation for Unix-like systems [7]. Microsoft eventually published protocol specifications, facilitating broader interoperability [8].

Despite their different origins, NFS and SMB/CIFS con- verge on core functionality: both provide transparent network file access, support authentication and authorization mechanisms, and implement caching strategies to improve performance. Their differences lie in default behaviors, security models, and optimization targets NFS traditionally emphasizing Unix semantics and SMB/CIFS prioritizing Windows integration. Understanding these protocols' evolution provides context for interpreting performance measurements and design trade-offs explored in subsequent sections.

3. Benchmarking Methodology and Tools

Performance evaluation of network file systems requires systematic benchmarking approaches that capture relevant workload characteristics while controlling for confounding variables. Benchmarks typically automate standardized tests that generate specific I/O patterns, measuring performance through metrics like throughput, latency, and operations per second. The choice of benchmark depends on evaluation goals whether assessing general filesystem performance, protocol-specific behavior, or workload-representative characteristics.

Postmark represents a widely-used benchmark designed to simulate workloads typical of mail and news servers [9]. These environments frequently handle numerous small files with create, delete, read, and append operations. The benchmark initializes a configurable number of files with randomly sized content, then executes transactions consisting of randomly selected operation pairs. Each transaction combines either file creation or deletion with either file reading or appending. The pseudo-random nature of operation selection with fixed seeds ensures reproducible results across test runs.

Postmark's primary metric is transactions processed per second, with additional measurements for individual operation types including throughput in bytes/second for read and write operations. The benchmark's popularity in storage research is evidenced by its use in 30 of 106 filesystem-related papers surveyed between 1999-2006 [9]. However, this same study identified limitations in contemporary benchmarking practices, including insufficient reporting of benchmark parameters and inadequate runtime duration on modern hardware. These methodological shortcomings complicate result comparison and reproduction across studies.

LADDIS adopts a different approach specifically designed for NFS server evaluation [10]. Developed collaboratively by multiple NFS server vendors and incorporated into the SPEC SFS benchmark suite, LADDIS generates synthetic workloads at the NFS protocol level rather than through filesystem APIs. This design isolates server performance from client implementation variations, enabling more direct comparison across different NFS server implementations. The benchmark can distribute load generation across multiple client systems, with manager and load generator components communicating via TCP.

The LADDIS workload generator selects NFS operations randomly from a predefined distribution based on operational studies of production NFS servers. It measures response times while progressively increasing request rates until server saturation. This approach captures performance degradation patterns under increasing load, providing insights into server scalability. However, the protocol-level implementation creates version dependencies, early LADDIS versions supported only NFSv2 over UDP, and subsequent updates lagged behind protocol evolution.

Comparative analysis of these benchmarking approaches reveals complementary strengths: Postmark evaluates end-to- end filesystem performance including client-side components, while LADDIS focuses specifically on server implementation efficiency. Researchers must select tools aligned with their evaluation objectives while acknowledging methodological limitations. Proper benchmarking requires careful parameter selection, adequate run durations, and comprehensive reporting to enable meaningful result interpretation and comparison.

4. Performance Analysis Across Protocol Versions

Protocol evolution introduces features aimed at improving performance, security, and functionality, but these enhancements often involve trade-offs that manifest differently across workloads. The transition from NFSv3 to NFSv4.1 exemplifies these trade-offs, with stateful operations and delegation mechanisms significantly altering protocol behavior and performance characteristics.

Empirical evaluation of NFS protocol versions reveals workload-dependent performance patterns. Chen et al. [11] conducted comprehensive comparisons using enterprise-class hardware on 10GbE networks with artificial latency injection capabilities. For data-intensive workloads like sequentially reading large (20GB) files with multiple threads, minimal differences emerged between NFSv3 and NFSv4.1 throughput variations remained within 2%, with network or storage sub- systems typically becoming the bottleneck before protocol differences manifested.

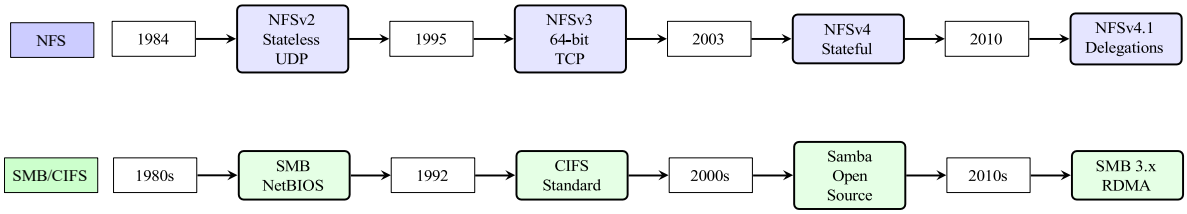

Fig. 1: Evolution timeline of NFS and SMB/CIFS protocols showing major versions and key features.

The most significant NFSv4.1 improvement comes from read delegations, which enhance client caching effectiveness. In stateless NFSv3, clients must periodically query servers for file modification times to maintain cache coherence, generating substantial GETATTR request volume. NFSv4.1's stateful de- sign allows servers to grant read delegations to single clients, enabling aggressive caching without continuous validation. When other clients attempt to write, servers recall delegations to maintain consistency. This mechanism dramatically reduces protocol chatter for read-dominated workloads accessing infrequently modified files.

Microbenchmarks evaluating small-file access demonstrate delegation benefits clearly. When multiple clients repeatedly read from a pool of 10,000 small (4KB) files, NFSv4.1 achieves approximately double the throughput of NFSv3 after stabilization [11]. Protocol-level analysis reveals NFSv3 generates 8.3× more requests, with 95% being GETATTR operations for cache validation. This reduction in request volume translates directly to improved throughput and reduced server load, particularly beneficial in high-latency network environments where round-trip times magnify the cost of frequent metadata operations.

However, the stateful nature of NFSv4.1 introduces over- heads absent in NFSv3. OPEN and CLOSE operations maintain server-side file state, increasing protocol requests for workloads with numerous file operations. Filebench fileserver workload measurements show 56% more protocol requests in NFSv4.1 compared to NFSv3, resulting in 8-18% performance degradation [11]. This demonstrates the fundamental trade-off: reduced validation traffic versus increased state maintenance overhead.

The performance impact of NFSv4.1 features thus depends critically on workload characteristics. Workloads with high read-to-write ratios, infrequently modified shared files, and sequential access patterns benefit substantially from delegations and other NFSv4.1 features. Conversely, workloads with frequent file creation/deletion, numerous concurrent writers, or predominantly private files may experience performance

degradation due to state maintenance overhead. These nuanced relationships underscore the importance of workload-aware protocol selection and configuration.

Table I: Performance comparison between NFSv3 and NFSv4.1 across different workloads

Workload Characteristic	NFSv3 Performance	NFSv4.1 Performance	Relative Improvement
Large file sequential read	High	Equivalent	0-2%
Small file random read	Baseline	2× higher	+100%
Metadata-intensive	High	Lower	-8-18%
Mixed read-write	Moderate	Higher	+15-40%
High-latency network	Low	Higher	+25-60%

5. Production Workload Characterization

Beyond controlled benchmarking, understanding real-world usage patterns provides crucial insights for storage system design and optimization. Leung et al. [12] conducted extensive measurement of production CIFS servers at NetApp, capturing network traffic to analyze access patterns, I/O characteristics, and sharing behaviors in corporate and engineering environments.

Their analysis revealed striking data access concentration: only 18% and 6% of allocated storage capacity was accessed during three-month observation periods in corporate and engineering datasets respectively. This strongly suggests most stored data serves archival purposes written once and rarely accessed thereafter. File reopen analysis confirmed this pattern, with 65% of files opened only once and 94% opened fewer than five times during measurement periods. Those files that were reopened typically saw subsequent accesses within one minute of closure, indicating temporal locality in access patterns.

I/O amount analysis showed read-dominated workloads but with lower read-write ratios than historically observed. Byte read-write ratios measured approximately 2:1 in both datasets, substantially lower than the 4:1 or higher ratios reported in earlier studies [12]. This shift likely reflects improved client-side caching, reducing read traffic and changing workload composition as network file systems host more user data and fewer system files. Additionally, mixed read-write I/O accounted for over 20% of transferred bytes, significantly higher than the less than 7% reported in previous studies.

File sharing analysis revealed limited concurrent access in production environments. Only 23.9% and 2.9% of files were ever opened by multiple users in corporate and engineering datasets respectively, with concurrent sharing dropping to 7.3% and 0.3%. These low sharing rates validate design optimizations targeting non-shared files, such as NFSv4.1's read delegations. The results suggest that optimizing for exclusive access patterns benefits most real-world usage while specialized mechanisms can handle rare sharing cases.

These workload observations suggest several storage system optimizations. The predominance of rarely-accessed files supports automated tiering systems that identify and mi- grate cold data to denser, more power-efficient storage. The increased write component in modern workloads justifies write-optimized filesystem designs like WAFL [13] and log- structured approaches [14]. Limited file sharing indicates client-side caching strategies can aggressively cache file data with relatively infrequent validation requirements.

6. Implications For System Design and Evaluation

The combined insights from controlled benchmarking and production workload analysis yield concrete guidance for network file system design, deployment, and evaluation. These implications span multiple system layers from protocol selection to storage hardware configuration. Protocol selection should consider workload characteristics rather than defaulting to the latest version. NFSv4.1's delegation features provide significant benefits for read-heavy workloads accessing mostly private files, particularly in high- latency environments. However, metadata-intensive workloads with frequent file creation and deletion may perform better with NFSv3's stateless approach. Similarly, SMB/CIFS implementations continue to evolve with features like SMB Direct (RDMA) in SMB 3.0, offering performance advantages in specific environments. Decision frameworks should incorporate workload analysis rather than relying on version numbers alone. Client caching strategies must balance performance benefits against consistency requirements. Production traces showing limited file sharing suggest aggressive caching is generally safe, with validation mechanisms needed primarily for shared files. Protocol designers can optimize for this common case while providing strong consistency guarantees when explicitly requested. Application developers should understand caching implications and use appropriate synchronization primitives when strong consistency is required.

Storage tiering and data placement algorithms can leverage observed access patterns. The strong skew toward rarely- accessed files supports automated data classification and migration systems. Cold data can move to higher-density, lower- performance storage tiers with minimal impact on user experience. Hot data identification should consider both access frequency and recency, with the understanding that most files become cold shortly after creation. Benchmarking methodologies must evolve to address identified limitations. Researchers should ensure adequate bench- mark duration, comprehensive parameter reporting, and work- load selection representative of modern use cases. The integration of microbenchmarks like LADDIS with application- level benchmarks like Postmark provides complementary perspectives. Future benchmark development should incorporate production workload characteristics including mixed read- write I/O, realistic file size distributions, and concurrent access patterns.

System evaluation frameworks should incorporate multiple measurement approaches. Controlled benchmarking provides reproducible performance comparisons, while production tracing captures real-world behavior. Together, these approaches validate each other and provide comprehensive system under-standing. Researchers should explicitly document evaluation methodologies to enable proper interpretation and comparison of results across studies.

7. Conclusion

Network file systems remain critical infrastructure com- ponents in diverse computing environments, with NFS and SMB/CIFS serving as the dominant protocols despite their different origins and evolutionary paths. Our examination of these systems through multiple measurement perspectives reveals nuanced performance characteristics and optimization opportunities. Benchmarking tools like Postmark and LADDIS provide complementary evaluation approaches, with Postmark simulating application-level workloads and LADDIS focusing on protocol-level server performance. Both require careful parameter selection and comprehensive reporting to yield meaning- ful, reproducible results. Version-level protocol comparisons demonstrate that newer is not universally better, NFSv4.1's delegation mechanism dramatically improves small-file read performance but increases overhead for metadata-intensive operations. Production workload analysis reveals substantial opportunities for storage optimization, with most files rarely accessed after creation and limited sharing between users. These patterns support aggressive client caching, automated

storage tiering, and write-optimized filesystem designs. The evolution of read- write ratios toward more balanced patterns suggests continuing workload changes that storage architects must accommodate. Future work should develop more sophisticated benchmarking methodologies that better capture modern workload characteristics, including cloud-native applications and containerized environments. Protocol evolution will likely continue addressing the tension between performance optimization and strong consistency guarantees. Storage systems research should further explore automated data management based on observed access patterns, potentially using machine learning techniques for prediction and optimization. Ultimately, effective network file system design, evaluation, and deployment require understanding the complex interaction between protocol features, workload characteristics, and system configuration. The multi-faceted measurement approach demonstrated in this paper provides a framework for developing this understanding and making informed decisions in both research and practice.

8. References

[1] Sandberg, R., et al. (1986). The sun network file system: Design, implementation and experience. In Proceedings of the Summer 1986 USENIX Technical Conference and Exhibition.

[2] Callaghan, B., Pawlowski, B., & Staubach, P. (1995). NFS version 3 protocol specification (RFC 1813).

[3] Beame, C., Thurlow, R., Callaghan, B., Robinson, D., Noveck, D., Eisler, M., & Shepler, S. (2003). Network file system (NFS) version 4 protocol (RFC 3530).

[4] Shepler, S., Eisler, M., & Noveck, D. (2010). Network file system (NFS) version 4.1 protocol (RFC 5661).

[5] Hertel, C. (2003). Implementing CIFS: The common internet file system. Prentice Hall Professional Technical Reference.

[6] X/Open Company Ltd. (1992). Protocols for X/Open PC internetworking: SMB, version 2 (Tech. Rep.). X/Open Company Ltd.

[7] Tridgell, A. (2003). How Samba was written. Accessed from https://www.samba.org/ftp/tridge/misc/french_cafe.txt.

[8] Microsoft. (2016). Common internet file system (CIFS) protocol. Accessed from https://msdn.microsoft.com/en-us/library/ee442092.aspx.

[9] Traeger, A., Zadok, E., Joukov, N., & Wright, C. (2008). A nine year study of file system and storage benchmarking. ACM Transactions on Storage, 4(2), 5:1–5:56.

[10] Wijtle, M., & Keith, B. (1993). LADDIS: The next generation in NFS file server benchmarking. In Proceedings of the USENIX Summer 1993 Technical Conference (pp. 9:1–9:31).

[11] Chen, M., Hildebrand, D., Kuenning, G., Shankaranarayana, S., Singh, B., & Zadok, E. (2015). Newer is sometimes better: An evaluation of NFSv4.1. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (pp. 165–176).

[12] Leung, W., Pasupathy, S., Goodson, G., & Miller, E. (2008). Measurement and analysis of large-scale network file system workloads. In USENIX 2008 Annual Technical Conference (pp. 213–226).

[13] Hitz, D., Lau, J., & Malcolm, M. (1994). File system design for an NFS file server appliance. In Proceedings of the USENIX Winter 1994 Technical Conference (p. 19).

[14] Rosenblum, M., & Ousterhout, J. (1992). The design and implementation of a log-structured file system. ACM Transactions on Computer Systems, 10(1), 26–52.

9.Conflict of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

10.Funding

No external funding was received to support or conduct this study.