REVIEW ARTICLE

# Performance Comparison of AWS EBS, EFS and S3 for Enterprise Cloud-Storage Workloads

**Mohammed Nazir**[1]

[1] Honda North America, Raymond, OH, United States

| Received | Revised | Accepted | Published |
|---|---|---|---|
| 05 February 2022 | 22 February 2022 | 17 April 2022 | 15 September 2022 |

## Abstract

**Background:** Enterprise workloads increasingly depend on cloud storage, yet Amazon Elastic Block Store (EBS), Amazon Elastic File System (EFS), and Amazon Simple Storage Service (S3) expose fundamentally different block, file, and object interfaces. Direct ranking by a single performance metric is therefore misleading.

**Objective:** To compare EBS, EFS, and S3 using workload-relevant dimensions including latency, I/O operations, throughput, concurrency, access semantics, and scalability, and to derive an evidence-based selection framework for enterprise deployments.

**Methods:** A structured review and technical evidence synthesis was performed using peer-reviewed cloud-storage benchmarking studies and authoritative AWS technical publications. Evidence was grouped by storage abstraction and by workload pattern, with emphasis on random transactional I/O, shared-file access, large-object transfer, and highly parallel analytics.

**Results:** EBS provides the strongest fit for latency-sensitive block workloads because performance is exposed through directly tunable IOPS and throughput; gp3 established a 3,000-IOPS and 125-MiB/s baseline with independent scaling to 16,000 IOPS and 1,000 MiB/s. EFS provides shared NFS semantics and scales aggregate access across many clients, but performance depends on metadata intensity, client concurrency, performance mode, and throughput configuration; service enhancements increased General Purpose read operations and per-client throughput, followed by a threefold increase in read throughput for read-heavy workloads. S3 provides the broadest horizontal scale for object workloads; request performance scales by prefix and parallelism, while object size, network path, and connection overhead strongly influence observed throughput. Strong read-after-write consistency removes an earlier application-level constraint without changing S3's object semantics.

**Conclusion:** No service is universally fastest. EBS is preferable for transaction-oriented and latency-critical block I/O, EFS for shared POSIX-like file access across concurrent compute nodes, and S3 for data lakes, backup, content distribution, and highly parallel large-object workloads. Enterprise architectures should benchmark the target access pattern and frequently combine all three services rather than selecting a single storage tier.

**Keywords:** Amazon Web Services; Amazon EBS; Amazon EFS; Amazon S3; cloud storage; performance; enterprise storage; benchmarking

**How to cite:** Nazir M. Performance Comparison of AWS EBS, EFS and S3 for Enterprise Cloud-Storage Workloads. International Journal of Business & Computational Science. 2022;2(1):26-33.

## 1. Introduction

Cloud-storage selection is often treated as a capacity and price decision, but enterprise application behavior is usually governed by latency, I/O concurrency, throughput, metadata activity, and access semantics. These factors become especially important when traditional storage-area-network, network-attached-storage, and object-storage functions are migrated to public cloud infrastructure. Early evaluations of cloud storage demonstrated that provider choice and access pattern can cause large differences in application performance, and that persistent storage must be benchmarked in the context of the application rather than judged from nominal capacity alone [1-4].

Within Amazon Web Services (AWS), EBS, EFS, and S3 occupy distinct layers of the storage hierarchy. EBS presents persistent block devices to EC2 instances and is therefore analogous to virtualized direct-attached or SAN storage. EFS provides a managed NFS interface that can be mounted concurrently by multiple compute instances. S3 exposes an HTTP-based object interface designed for very large namespaces, durable object storage, and massive request parallelism. Because their interfaces differ, common metrics are not completely interchangeable: an EBS IOPS figure, an EFS file-operation rate, and an S3 request rate describe different parts of the I/O stack.

This distinction has practical consequences. Small random database writes emphasize latency and queue behavior; shared application trees and user directories emphasize metadata operations and multi-client consistency; data lakes and backup repositories emphasize object-level throughput, concurrency, and network utilization. Prior studies of S3 and cloud object storage show that object size, geographic path, and parallelism materially alter achieved bandwidth [2,4-8]. Comparative studies of file, block, and object systems similarly show that storage performance changes with network delay, packet loss, access behavior, and client configuration [9].

The objective of this review is to compare AWS EBS, EFS, and S3 for enterprise cloud-storage workloads using a common workload-oriented framework. Rather than produce an artificial single score, the analysis evaluates where each service provides the most favorable performance envelope, which bottlenecks dominate, and how enterprises can combine services in multi-tier architectures.

## 2. Methods

### 2.1 Review design and evidence selection

A structured narrative review was undertaken to synthesize performance evidence relevant to block, file, and object storage in AWS. Searches emphasized peer-reviewed studies that measured Amazon S3, EC2-attached storage, cloud file systems, or comparable network/object storage under controlled or realistic workloads. Studies were retained when they reported at least one performance-relevant outcome such as latency, throughput, request rate, scalability, object-size sensitivity, metadata behavior, or end-to-end application runtime. Vendor-independent studies were prioritized for comparative interpretation, while authoritative AWS publications were used to establish service capabilities and published performance limits.

The evidence was organized into four workload classes: (i) latency-sensitive random block I/O; (ii) shared-file workloads with metadata and concurrent clients; (iii) sequential and large-object transfer; and (iv) horizontally parallel analytics, data-lake, and backup workloads. This approach was selected because a single

synthetic benchmark cannot be applied equivalently to block devices, NFS file systems, and HTTP object stores without introducing interface-specific translation overhead.

Performance was assessed using five primary dimensions: per-operation latency, sustainable throughput, I/O or request concurrency, sensitivity to access granularity, and scaling behavior. Secondary dimensions included consistency semantics, client topology, operational elasticity, and the degree to which performance can be provisioned independently of capacity.

### 2.2 Comparative interpretation

Reported values were not normalized into a single cross-service benchmark because the units represent non-equivalent operations. Instead, the review used a service-envelope approach: EBS was interpreted through block IOPS, throughput, queue depth, and latency; EFS through file-operation latency, NFS concurrency, and file-system throughput; and S3 through request rate, object transfer throughput, and prefix-level parallelism. Where official service limits are cited, they represent the published capability of the relevant service generation and do not imply that every workload will sustain that level.

## 3. Results

### 3.1 Characteristics of the evidence base

The included literature spans early S3 measurements, public-cloud comparative benchmarking, EC2 workflow storage experiments, object-storage benchmark frameworks, and broader comparisons of block/file/object architectures. Across these studies, three findings recur. First, performance varies markedly with file or object size because protocol and request overhead constitute a larger fraction of small transfers [2,5,7]. Second, concurrency is essential to saturate modern cloud storage and network paths, but the optimal degree of parallelism differs by service and workload [2,6-8]. Third, geographic and network placement can change throughput by multiples, making client location and network path part of the storage benchmark rather than a separate concern [5,7,9].

### 3.2 Amazon EBS: latency and provisioned block performance

EBS is structurally favored when applications require a persistent block device with a conventional file system or database layered above it. Its main advantage is that the application can issue block I/O without object or NFS protocol translation. EBS performance is therefore naturally expressed through IOPS, throughput, I/O size, queue depth, and latency. CloudCmp and EC2 workflow studies showed that persistent storage performance materially affects application runtime and that the storage path must be selected alongside compute resources [3,4].

The gp3 volume generation further separated performance from capacity. Published specifications established a baseline of 3,000 IOPS and 125 MiB/s for any gp3 volume size, with independent scaling to 16,000 IOPS and 1,000 MiB/s [11]. This model is advantageous for enterprise databases and transaction-processing systems because capacity does not need to be overprovisioned merely to obtain more IOPS. The remaining performance ceiling is not purely a volume property: the EC2 instance's EBS bandwidth, I/O size, queue depth, operating-system scheduler, and application concurrency can constrain delivered performance.

For random small-block workloads, EBS is consequently the strongest default choice among the three services. However, it is not a shared file system by default, and its performance does not automatically scale across thousands of clients in the way EFS and S3 can. Horizontal scale normally requires application-level sharding, multiple volumes, cluster-aware software, or a higher-level distributed data service.

### 3.3 Amazon EFS: multi-client file semantics with workload-sensitive scaling

EFS addresses a different problem: multiple compute instances require simultaneous access to a managed NFS file system. This eliminates the need to build a file-serving layer on top of block storage and makes EFS attractive for content repositories, shared application data, home directories, software build trees, containerized workloads, and scale-out analysis that depends on file semantics.

The performance trade-off is that each file and metadata operation traverses a distributed file-service path. Studies of cloud and network file systems consistently show that metadata intensity, access granularity, and network conditions can dominate performance [4,6,9]. Small-file workloads therefore behave differently from streaming large files. AWS service improvements raised the General Purpose mode read-operation limit to 35,000 operations/s while retaining a lower write-operation ceiling, and later increased per-client throughput to 500 MB/s [14,15]. Read-heavy EFS workloads then received a threefold increase in metered read throughput, allowing bursting-mode file systems to reach 300 MB/s of read throughput or 300 MB/s per TiB of stored Standard-class data, whichever was higher [16].

These changes reinforce a central EFS characteristic: performance is often aggregate and concurrency-driven. A single-threaded metadata-heavy workload may not benefit from the same scaling behavior as a multi-client sequential read workload. EFS is therefore best evaluated with the intended number of clients, directory layout, average I/O size, NFS mount behavior, and sustained-versus-bursty throughput pattern.

### 3.4 Amazon S3: object-scale concurrency and large-transfer efficiency

S3 is optimized for object storage rather than block or POSIX file semantics. Early studies established that S3 throughput depends strongly on object size, client location, and the number of concurrent requests [2,5,7]. Baun et al. observed that protocol overhead is proportionally more important for small objects and that parallelism becomes more beneficial as transfer size grows [7]. Jones et al. similarly showed that parallel I/O can drive S3 to network-scale throughput in data-intensive environments [8].

S3's request model is designed for horizontal expansion. Published service guidance increased per-prefix request performance to at least 3,500 write-class requests per second and 5,500 read-class requests per second, with performance scaling across multiple prefixes [12]. For enterprise analytics, large backup streams, media repositories, and data lakes, this model can produce higher aggregate concurrency than a single block device or NFS client. The practical bottleneck often shifts to client CPU, TCP connections, EC2 network bandwidth, object size, and the ability of the application to issue asynchronous or parallel requests.

A major semantic change occurred when S3 introduced strong read-after-write consistency for GET, PUT, and LIST operations without a reported performance penalty [13]. This reduced the need for external consistency layers in analytics workflows, although it did not convert S3 into a file system: object updates, directory emulation, and small random in-place writes still differ fundamentally from EBS and EFS behavior.

### 3.5 Cross-service performance comparison

Across the evidence, the most consistent conclusion is that no service has a universal performance advantage. EBS dominates when the unit of work is a latency-sensitive block I/O and the application can exploit a directly attached volume. EFS dominates when the architectural requirement is shared file access from multiple compute nodes and the workload benefits from aggregate file-system concurrency. S3 dominates when the workload can be expressed as object operations and can parallelize across objects, clients, or prefixes. The apparent winner can reverse when the access pattern changes: large sequential objects favor S3's scalable request model; small-file metadata workloads favor avoiding object translation; random database pages favor EBS; and shared file trees favor EFS even if their individual operations carry higher latency than EBS.

## 4. Discussion

### 4.1 Implications for enterprise workload placement

Enterprise storage should be selected from the application access pattern outward. Relational databases, transactional middleware, virtual-machine boot volumes, and applications with frequent small random reads and writes generally align with EBS. The ability to provision IOPS and throughput independently of capacity makes gp3 particularly suitable when performance is predictable and capacity is moderate. For the highest I/O intensity, provisioned-IOPS EBS classes can extend this model further, but the instance-to-EBS path must be sized accordingly.

EFS is preferable where file sharing is an architectural requirement rather than an implementation convenience. Examples include horizontally scaled web tiers, content-management repositories, shared development environments, container clusters, and workflows in which many nodes need a consistent namespace. In these cases, replacing EFS with EBS would require an additional clustered file layer, while replacing it with S3 could require application changes to remove POSIX assumptions. The performance penalty of NFS semantics may therefore be justified by a substantial simplification of the application architecture.

S3 should be treated as a parallel object platform, not as a remote disk. Its best performance is obtained when applications use sufficiently large objects, concurrency, multipart transfer where appropriate, and direct object APIs. Mounting S3 through a file-system translation layer may be operationally convenient, but it can introduce metadata and semantic overhead that obscures native object-store strengths. This is consistent with broader studies in which object storage remained robust under Internet-like conditions while file/block protocols were more sensitive to network complexity [9].

### 4.2 Multi-tier architectures are usually superior to single-service designs

For large enterprise platforms, the most effective architecture is often composite. A database can maintain active pages on EBS, application servers can share configuration or content through EFS, and historical objects, backups, logs, and analytics datasets can reside in S3. This pattern aligns the access semantics of each workload with the storage abstraction that implements them natively. It also reduces the temptation to force one service into an unsuitable role merely to simplify procurement or operations.

Performance testing should therefore reproduce the complete path: the EC2 instance type, EBS optimization limits, EFS mount and throughput configuration, S3 request concurrency, object or I/O size distribution, read/write ratio, encryption, network placement, and realistic queue depth. Short synthetic benchmarks that omit these variables can overstate achievable throughput or understate tail latency. Bocchi et al. reported that data-center selection alone could change cloud-storage throughput by factors of two to ten, illustrating why a benchmark detached from deployment topology can be misleading [5].

### 4.3 Benchmarking considerations

A fair enterprise benchmark should use workload-specific tools rather than a single universal test. EBS is appropriately tested with block-I/O generators such as fio using representative block sizes, random/sequential mixes, queue depth, and read/write ratios. EFS should be tested with file-system workloads that include metadata operations, directory traversal, concurrent clients, and sustained throughput periods. S3 should be tested with object-level GET/PUT operations across realistic object-size distributions, connection pools, and parallel request counts. End-to-end application benchmarks should then validate whether microbenchmark improvements translate into user-visible performance.

Tail behavior is also important. Median throughput may hide transient throttling, burst-credit exhaustion, connection setup costs, or instance network limits. For production systems, p95/p99 latency, error/retry behavior, sustained throughput after warm-up, and scaling efficiency with additional clients should be evaluated alongside average performance.

### 4.4 Limitations

The literature contains substantially more independent performance research on S3 and generic object storage than on managed EFS, and EBS evolves rapidly across volume generations. Consequently, the comparison combines peer-reviewed measurements with authoritative service specifications. Direct numeric comparisons across EBS IOPS, EFS file operations, and S3 requests are intentionally avoided because the operations are not semantically equivalent. Published limits also represent service capabilities rather than guaranteed application performance. The results should therefore be interpreted as an evidence-based selection framework and not as a substitute for workload-specific testing.

## 5. Conclusion

AWS EBS, EFS, and S3 are complementary rather than interchangeable storage services. EBS provides the strongest performance fit for latency-sensitive, transaction-oriented block workloads because applications can directly tune IOPS and throughput. EFS provides scalable shared-file access and is most effective when multiple clients require a common NFS namespace, although metadata and small-operation latency can constrain single-client performance. S3 provides the broadest horizontal scaling for object workloads and can achieve very high aggregate throughput when requests are parallelized and objects are sufficiently large. The principal enterprise design rule is therefore to match storage semantics to workload behavior. Performance comparisons that ignore access granularity, concurrency, network path, and service abstraction can be misleading. Multi-tier architectures using EBS for active transactional data, EFS for shared file access, and S3 for object-scale data are often better aligned with enterprise performance requirements than attempts to standardize on a single storage service.

## Declarations

Ethics approval and consent to participate: Not applicable.

Consent for publication: Not applicable.

Data availability: No primary dataset was generated; the article synthesizes published literature and publicly available technical documentation.

Competing interests: None declared.

## References

[1] Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel SL. Amazon S3 for science grids: a viable solution? In: Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing. New York: ACM; 2008. p. 55-64. doi:10.1145/1383519.1383526.

[2] Li A, Yang X, Kandula S, Zhang M. CloudCmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. New York: ACM; 2010. p. 1-14. doi:10.1145/1879141.1879143.

[3] Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman BP, et al. Data sharing options for scientific workflows on Amazon EC2. In: 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE; 2010. p. 1-9. doi:10.1109/SC.2010.17.

[4] Bocchi E, Mellia M, Sarni S. Cloud storage service benchmarking: methodologies and experimentations. In: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet). IEEE; 2014. p. 395-400. doi:10.1109/CloudNet.2014.6969027.

[5] Zhao D, Yang X, Sadooghi I, Garzoglio G, Timm S, Raicu I. High-performance storage support for scientific applications on the cloud. In: Proceedings of the 6th Workshop on Scientific Cloud Computing. New York: ACM; 2015. p. 33-36. doi:10.1145/2755644.2755648.

[6] Goncalves GD, Drago I, Vieira AB, Couto da Silva AP, Almeida JM, Mellia M. Workload models and performance evaluation of cloud storage services. Comput Netw. 2016;109:183-199. doi:10.1016/j.comnet.2016.03.024.

[7] Baun C, Cocos HN, Spanou RM. OSSperf: a lightweight solution for the performance evaluation of object-based cloud storage services. J Cloud Comput. 2017;6:24. doi:10.1186/s13677-017-0096-x.

[8] Jones MS, Kepner J, Arcand WR, Bestor DA, Bergeron WJ, Gadepally VN, et al. Performance measurements of supercomputing and cloud storage solutions. In: 2017 IEEE High Performance Extreme Computing Conference (HPEC). IEEE; 2017. doi:10.1109/HPEC.2017.8091073.

[9] Ou Z, Song M, Hwang ZH, Yla-Jaaski A, Wang R, Cui Y, et al. Is cloud storage ready? Performance comparison of representative IP-based storage systems. J Syst Softw. 2018;138:206-221. doi:10.1016/j.jss.2018.01.015.

[10] Amazon Web Services. How do you select your storage solution? AWS Well-Architected Framework: Performance Efficiency Pillar. 2020.

[11] Amazon Web Services. New Amazon EBS gp3 volume lets you provision performance apart from capacity. AWS News Blog. 2020 Dec 1.

[12] Amazon Web Services. Amazon S3 announces increased request rate performance. AWS What's New. 2018 Jul 17.

[13] Amazon Web Services. Amazon S3 now delivers strong read-after-write consistency automatically for all applications. AWS What's New. 2020 Dec 1.

[14] Amazon Web Services. Amazon Elastic File System announces 400% increase in read operations for General Purpose mode file systems. AWS What's New. 2020 Apr 1.

[15] Amazon Web Services. Amazon Elastic File System increases per-client throughput by 100%. AWS What's New. 2020 Jul 23.

[16] Amazon Web Services. Amazon Elastic File System triples read throughput. AWS What's New. 2021 Jan 29.

## Tables

**Table 1.** Included studies and evidence relevant to cloud-storage performance.

| Study | Year | Storage focus | Design / workload | Key performance finding |
|---|---|---|---|---|
| Palankar et al. [1] | 2008 | Amazon S3 | Single- and multi-node GET performance; remote locations; science-grid use | S3 performance varies with object size, concurrency, and access location; parallel access improves aggregate throughput. |
| Li et al. [2] | 2010 | Public-cloud persistent storage including Amazon services | CloudCmp microbenchmarks and application case studies | Persistent-storage performance and cost differ substantially across cloud providers and materially affect application choice. |
| Juve et al. [3] | 2010 | Amazon EC2 storage options | Three scientific workflows using multiple data-sharing strategies | Storage architecture and data placement significantly affect end-to-end workflow performance and cost. |
| Bocchi et al. [4] | 2014 | Amazon S3 and competing cloud object stores | Upload/download benchmarking across data centers and file sizes | No universal winner; file size and data-center placement strongly affect throughput, with multi-fold variation by location. |
| Zhao et al. [5] | 2015 | S3FS, HDFS, Ceph, FusionFS on EC2/other platforms | Metadata-, write-, and read-intensive microbenchmarks | Cloud storage performance depends strongly on metadata handling, locality, and the workload's read/write structure. |
| Goncalves et al. [6] | 2016 | Cloud-storage workload models | Measured and synthetic workload characterization | Realistic workload structure is essential because synchronization and sharing behavior drive network and storage load. |
| Baun et al. [7] | 2017 | Amazon S3 and other object stores | Sequential/parallel CRUD benchmark with multiple object sizes | Protocol overhead penalizes small objects; larger downloads benefit more from parallelism; network path affects observed S3 bandwidth. |
| Jones et al. [8] | 2017 | Amazon S3 versus high-performance file storage | Parallel I/O from single and multiple clients | S3 can approach available network bandwidth with sufficient parallelism; storage architecture should match data-sharing pattern. |
| Ou et al. [9] | 2018 | Object, NFS, and iSCSI storage | Microbenchmarks plus PostMark/FileBench under varied network conditions | Block storage excels in favorable networks; object storage is more resilient to Internet-like conditions; client configuration matters. |

**Table 2.** Published service characteristics relevant to enterprise performance.

| Service | Abstraction | Access model | Representative published performance characteristics | Workloads best aligned | Principal performance controls |
|---|---|---|---|---|---|
| EBS gp3 | Block | EC2-attached persistent block device | 3,000 IOPS and 125 MiB/s at baseline; provisionable to 16,000 IOPS and 1,000 MiB/s | Random I/O, databases, boot volumes, transactional applications | Volume and instance EBS limits, I/O size, queue depth, application concurrency |
| EFS General Purpose / Bursting or Provisioned | File (NFS) | Shared file system mounted by multiple clients | Up to 35,000 read operations/s in General Purpose mode; per-client throughput increased to 500 MB/s; read-heavy throughput increased threefold in 2021 | Shared repositories, web/content tiers, home directories, containers, scale-out file workloads | Metadata rate, client count, file size, burst balance/throughput mode, NFS behavior |
| S3 Standard | Object (HTTP) | Bucket/object API with massive parallel request model | At least 3,500 write-class and 5,500 read-class requests/s per prefix; scalable across prefixes | Data lakes, backup, static content, analytics, media, large-object transfer | Object size, request parallelism, network path, connection reuse, client/instance bandwidth |

**Table 3.** Workload-to-service fit for common enterprise storage patterns.

| Workload | Latency sensitivity | Shared-file requirement | Horizontal concurrency | Preferred service | Rationale |
|---|---|---|---|---|---|
| OLTP database / small random 4-16 KiB I/O | Very high | Low | Low to moderate | EBS | Native block I/O and tunable IOPS/throughput minimize protocol translation. |
| Shared web content / CMS / application files | Moderate | Very high | Moderate | EFS | Concurrent NFS access avoids building a separate clustered file layer. |
| Container shared persistent file data | Moderate | Very high | Moderate to high | EFS | Shared namespace and elastic client population are primary requirements. |
| Backup repository / immutable objects | Low | Low | Very high | S3 | Object semantics, durability, and parallel large transfers fit the access model. |
| Data lake / analytics input | Low to moderate | Low | Very high | S3 | Horizontal request concurrency and namespace scale outweigh file-system semantics. |
| Single-host enterprise application requiring POSIX file system | Very high | Low | Low | EBS | Local file system on block device offers the shortest I/O path. |
| Large shared sequential reads from many nodes | Moderate | Very high | High | EFS or S3 | EFS when POSIX/NFS is required; S3 when object-native applications can parallelize. |
| Archival/static content distribution | Low | Low | Very high | S3 | Object access and web-scale fan-out are natural service characteristics. |

**Table 4.** Common performance bottlenecks and optimization actions.

| Service | Bottleneck | Observed effect | Optimization action |
|---|---|---|---|
| EBS | Suboptimal queue depth | Delivered IOPS below provisioned potential | Tune queue depth and number of jobs; validate I/O size and read/write mix |
| EBS | EC2 instance EBS-bandwidth limit | Volume benchmarks plateau below nominal volume limit | Use an instance class with sufficient EBS bandwidth; stripe where architecture permits |
| EFS | Metadata-intensive small-file access | High operation latency and poor throughput despite low byte volume | Increase parallelism carefully, distribute directory activity, validate mount options, benchmark metadata separately |
| EFS | Burst/throughput constraint | Throughput falls during sustained activity | Monitor throughput mode and burst behavior; use provisioned throughput where sustained demand is known |
| EFS | Single-client saturation | Aggregate file-system capability not reached | Use multiple clients or redesign job parallelism when workload permits |
| S3 | Small object/request overhead | Low effective bandwidth with many tiny objects | Batch/aggregate data where possible; use parallel requests judiciously; reuse connections |
| S3 | Insufficient request concurrency | Network underutilized | Increase asynchronous requests, multipart operations, and prefix/object parallelism |

| Service | Bottleneck | Observed effect | Optimization action |
|---|---|---|---|
| S3 | Network path / placement | Large variation between regions or client locations | Co-locate compute and storage; test intended region and instance network characteristics |
| All | Unrealistic microbenchmark | Good synthetic result but poor application performance | Reproduce production I/O size, concurrency, access pattern, encryption and cache behavior |

**Table 5.** Comparative enterprise storage-selection framework.

| Dimension | EBS | EFS | S3 |
|---|---|---|---|
| Access unit | Blocks | Files/directories | Objects |
| Primary performance emphasis | IOPS + latency + throughput | File-operation latency + aggregate throughput + client concurrency | Request rate + object throughput + massive concurrency |
| Strongest enterprise role | Transactional primary storage | Shared application/file tier | Data lake, backup, content/object tier |
| Scaling model | Scale volume/instance performance or shard across volumes/nodes | Scale aggregate access across clients and throughput configuration | Scale requests across objects, clients and prefixes |
| Small-operation behavior | Strong when properly provisioned | Can be metadata-limited | Request/protocol overhead can dominate |
| Large sequential transfer | Strong within volume/instance bandwidth | Strong with sufficient client/file-system throughput | Strong when transfers are parallel and network bandwidth is available |
| Shared multi-client semantics | Not native for general gp3 use | Native NFS shared namespace | Native multi-client object access but not POSIX file semantics |
| Recommended benchmark style | fio/DiskSpd with realistic block mix | FileBench/IOzone/custom NFS workload with metadata and multiple clients | GET/PUT workload with realistic object-size distribution and concurrency |

## Figures

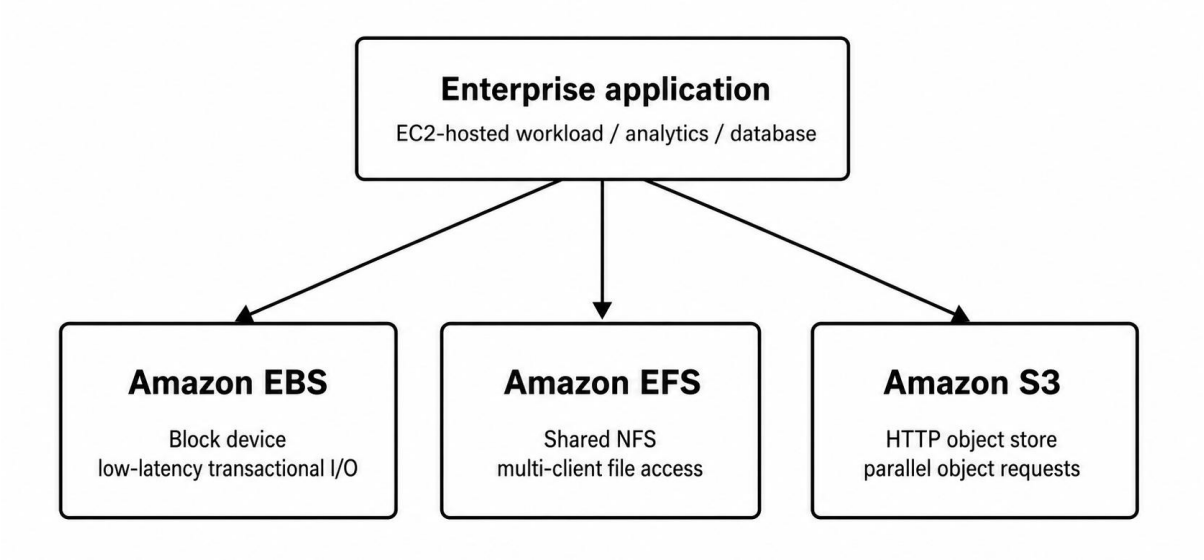


**Figure 1.** Architectural comparison of EBS, EFS, and S3 access paths for enterprise cloud workloads.

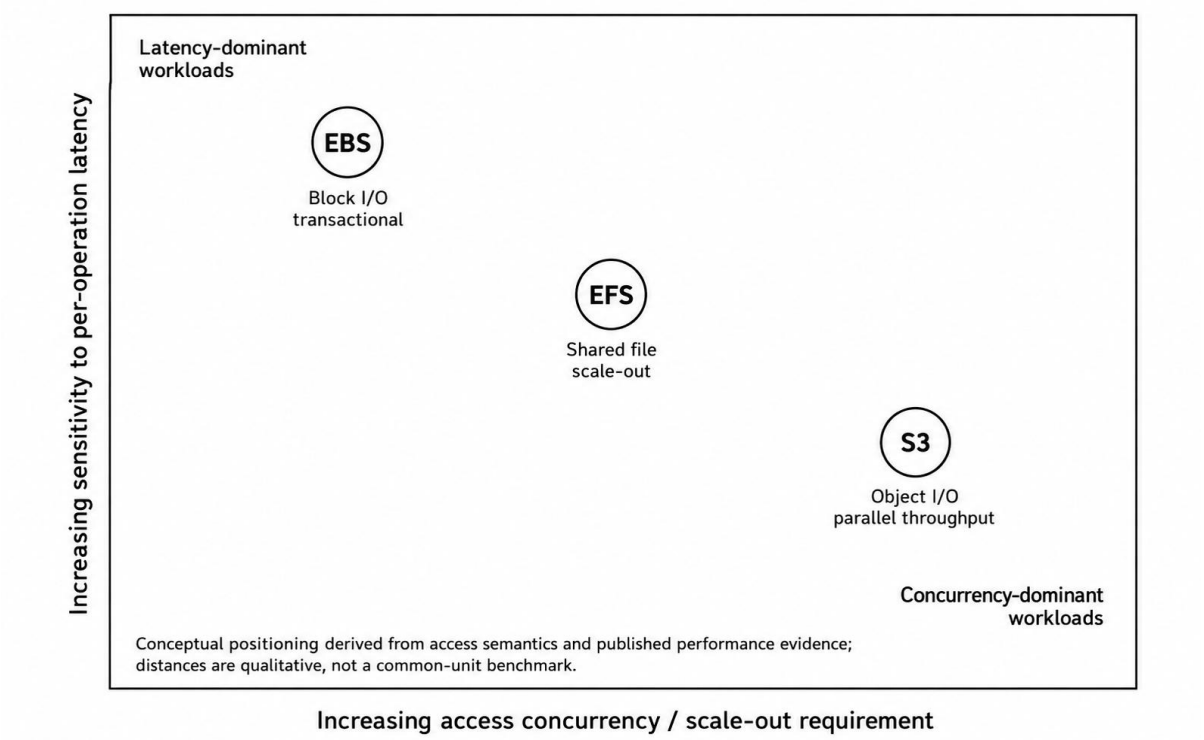


**Figure 2.** Qualitative workload positioning of EBS, EFS, and S3 according to latency sensitivity and required concurrency. The positioning is conceptual and should not be interpreted as a common-unit numerical benchmark.

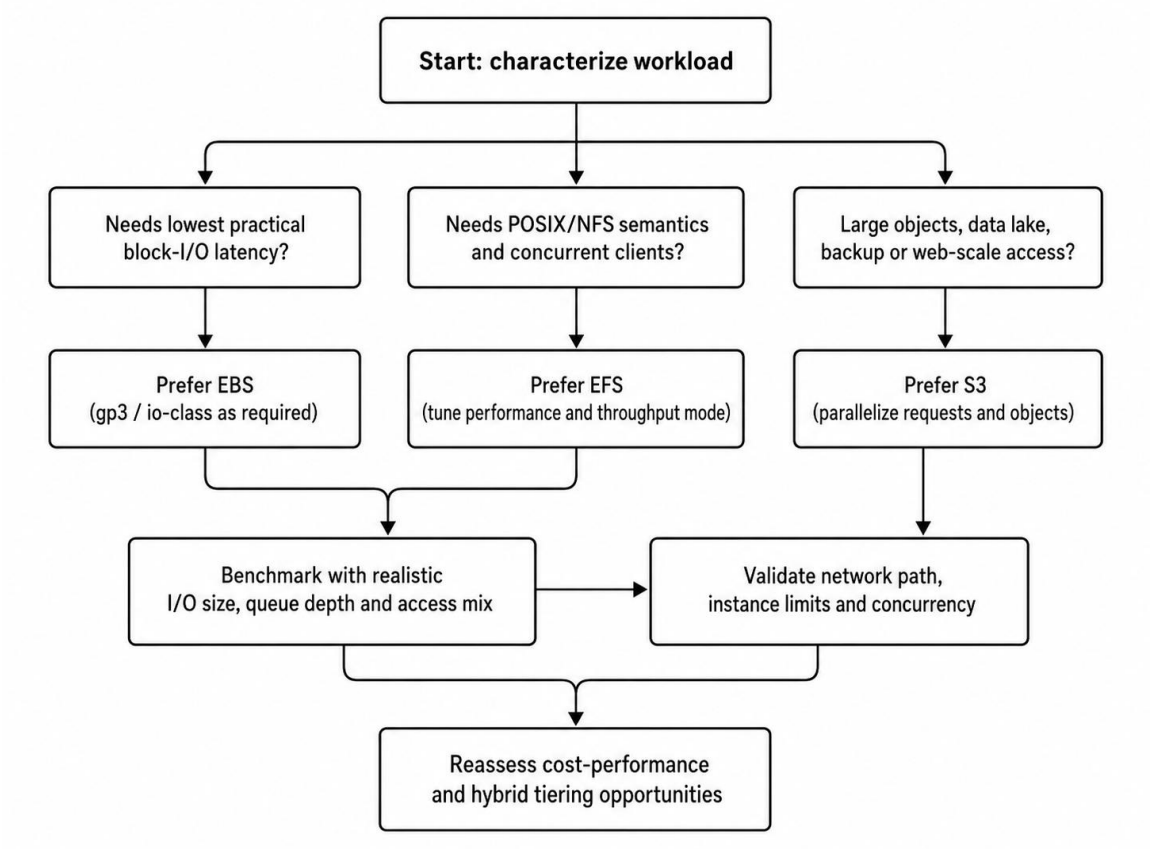


**Figure 3.** Evidence-based workflow for selecting and validating an AWS storage service for enterprise workloads.