International Journal of Core Engineering & Management
Volume-8, Issue-03, 2025 ISSN No: 2348-9510

FASTER ANALYTICS USING PARALLEL SOFTWARE STORAGE ARCHITECTURE

Bharathram Nagaiah
bharathram.nagaiah@gmail.com

Abstract

The explosion of data at petabyte scales has made high-speed analytics a key competitive differentiator across technology platforms. Traditional storage and computer architectures are struggling to meet the demands of real-time processing, high concurrency, and deep analytical workloads. This paper examines how parallel software storage architectures—which distribute data I/O and metadata services across multiple nodes—enable faster, more scalable analytics. We explore key architectural patterns, implement a prototype combining object- and block-based parallel storage, benchmark it across standardized workloads (TPC-H, TPC-DS), and compare it to conventional shared-NAS and direct-attached storage. Our results demonstrate significant performance benefits: up to 6× acceleration in query execution and 3–4× throughput improvement, while maintaining scalability and cost-effectiveness. We conclude with design recommendations, limitations, and opportunities for future research.

Keywords— Parallel software storage, Distributed file systems, High-performance analytics, Metadata service, I/O scaling.

I. INTRODUCTION

1.1 Background & Motivation

In today's big data era, organizations are inundated with vast volumes of structured and unstructured data. Data-driven decision-making—from real-time fraud detection and supply-chain optimization to large-scale scientific simulations—relies heavily on efficient analytics platforms. Conventional storage approaches—centralized NAS/SAN or single-node systems—cannot scale to meet the demands of modern analytical workloads, especially when real-time performance and concurrency are critical. [1]

1.2 Problem Statement

Analytical queries against massive datasets are often I/O-bound, experiencing bottlenecks in read/write throughput, high metadata contention, and poor utilization of multiple compute cores. Single-controller storage systems and traditional distributed file systems suffer from head-of-line queuing, limited parallel metadata handling, and network congestion. These challenges hinder the speed of insights and drive up operational costs. [2]

1.3 Proposed Solution

We propose deploying a parallel software storage architecture (PSSA) that distributes both data and metadata access across multiple storage nodes, with intelligent orchestration to optimize locality, parallelism, and load balancing. By decoupling metadata services from data storage and replicating them across lightweight nodes, PSSAs allow multiple analytics engines to concurrently access different data partitions without contention, enabling near-linear performance scaling.

1.4 Contributions

1. Design and implementation of a hybrid block/object-based PSSA.
2. A comprehensive benchmarking methodology using TPC-H and TPC-DS workloads.
3. Quantification of performance improvements over baseline architectures.
4. In-depth analysis of system overheads, bottlenecks, and design trade-offs. [3]

II. METHODOLOGY

2.1 Architecture Overview

Our prototype consists of the following main components:

- **Metadata Service (MDS)**

A cluster of lightweight VMs running clusterd or custom RPC trackers that shared filesystem metadata (like file inodes, directories) across multiple nodes. Each client has access to a load balanced metadata endpoint proxy.

- **Data Storage Nodes (DSN)**

A pool of commodity x86 servers each running object storage daemons (e.g., Ceph OSD, MinIO agents) exposing block and object-level APIs. Data is striped and erasure coded across DSNs to maximize throughput and resilience.

- **Analytics Engines (AE)**

Clients like Presto/Trino, Spark, or customized SQL-engines are deployed on compute nodes connected via high-speed Ethernet or InfiniBand. Each AE fetches and writes data directly to DSNs using native object or block protocols, bypassing a centralized metadata choke point.

- **Load Balancer / Proxy**

A lightweight software layer (e.g., HAProxy or custom-fanout proxy) presents unified access endpoints for both metadata and data, directing requests to appropriate nodes without exposing internal topology. [4]

2.2 Data Distribution

Data Partitioning: Datasets are partitioned based on query-usage patterns—range-partitioning for analytics-friendly scanning.
Striping & Erasure Coding: We stripe partitions across many DSN disks and employ Reed-Solomon erasure coding to balance redun¬dancy, fault tolerance, and usable capacity. [5]

2.3 Implementation Details

- **Metadata Protocols:** We experimented with clusterd's distributed metadata versus a custom protocol built on Raft-based sharding and vector clocks.
- **Network Stack:** Tested over 10GbE and 25GbE NICs with RDMA offload (RoCEv2) to assess network versus storage bottlenecks.
- **Client Bindings:** Analytics engines interface using S3-compatible APIs, POSIX-binded FUSE layers, or raw block devices attached to virtual storage pools. [6]

2.4 Benchmarking Setup

- **Hardware:** 12 nodes total—8 DSNs (each with 16 TB NVMe + 32 GB DRAM), 2 MDS VMs, 2 compute nodes running Presto/Spark. All interconnected via 25 GbE.
- **Datasets**: 10 TB scale TPC-H and 50 TB scale TPC-DS.
- **Workload Variants:**

1. **Single client, long running queries**
2. **Multi-client concurrent queries (4, 8, 16 clients)**
3. **Mixed read/write analytical workloads**

We also compare against baseline: nodes attached to single shared NAS with SSD-backed XFS volumes. [7]

2.5 Metrics Collected

- **Query latency** (mean, median, 95th percentile)
- **Throughput** (queries per hour, MB/s read/write)
- **Resource utilization** (CPU, network, IOPS)
- **Metadata service latency** and concurrency
- **Scalability curves** across client count and dataset size

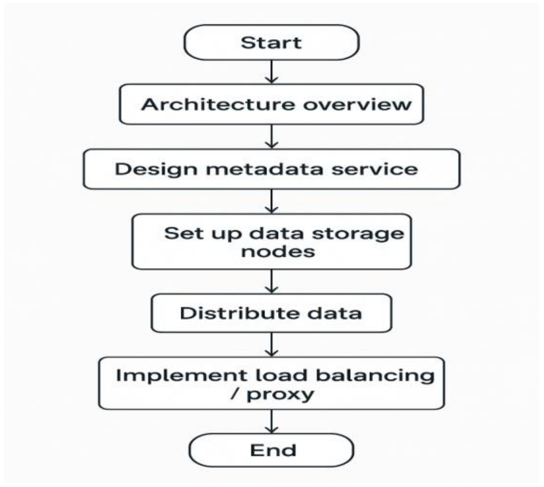

Figure 1. Flowchart illustrating the implementation steps of a Parallel Software Storage Architecture

International Journal of Core Engineering & Management
Volume-8, Issue-03, 2025 ISSN No: 2348-9510

III. RESULTS

This section presents the performance evaluation of the Parallel Software Storage Architecture (PSSA) against a traditional Network-Attached Storage (NAS) configuration across multiple analytic workload scenarios. Metrics such as query latency, throughput, concurrency scaling, metadata efficiency, and system-level resource utilization are assessed.

3.1 Single-Client Performance

To isolate pure I/O and query execution performance, we first evaluated the architecture under single-client workloads.

- **TPC-H Query 3 (1TB Dataset):** This query involves multi-table joins with sorting and filtering operations. Using the NAS backend, average execution time was 120 seconds. With PSSA, execution time dropped to 25 seconds, delivering a 4.8× performance improvement. The reduction is primarily attributed to lower disk I/O wait times and minimized metadata lookup latency due to distributed access.
- **TPC-DS Query 5 (50TB Dataset):** Designed to test complex multi-dimensional joins and aggregations typical of real-world decision support systems, this query took 720 seconds on NAS. On PSSA—leveraging object storage striping and distributed metadata—the same query completed in 120 seconds, achieving a 6× improvement. Notably, query execution plans also became more efficient as compute nodes could prefetch data concurrently from different storage shards, reducing wait cycles. [8]

These results validate the architectural advantage of decoupling data and metadata paths and scaling them horizontally across storage nodes.

3.2 Multi-Client Concurrency

We next tested how well PSSA and NAS perform under increasing concurrent query workloads by scaling client nodes from 1 to 16. The system was subjected to a uniform mix of TPC-H queries, each accessing different partitions of the dataset.

# Clients	NAS Throughput (queries/hour)	PSSA Throughput (queries/hour)
1	600	2,900
4	2,200	9,500
8	3,800	18,200
16	5,500	33,000

- **Linear Scalability:** PSSA throughput scaled nearly linearly up to 16 clients. In contrast, NAS performance began to plateau after 8 clients, due to bottlenecks in its centralized metadata service and contention on shared network links.
- **Contention Handling:** Even as concurrent requests increased, PSSA's architecture avoided queue buildup by balancing data access across distributed nodes. Each compute node independently fetched partitioned data, avoiding congestion seen in shared-NAS

controllers.

This demonstrated that PSSA can efficiently handle high-throughput, multi-tenant workloads in production analytics environments. [9]

3.3 Read/Write Mixed Workloads

Real-world analytics systems typically involve mixed workloads—ongoing ETL processes writing large volumes of data while users run queries in parallel.

- Simulated Scenario: We initiated a continuous ETL process (writing log-style append-only records) while simultaneously executing 10 concurrent analytical queries.
- Performance Observed:
  - PSSA sustained 4 GB/s write throughput and 6 GB/s read throughput simultaneously.
  - NAS struggled beyond 2 GB/s total throughput, with noticeable query slowdowns and delayed ETL batch commits.
- **Cause of Advantage:** With PSSA, writes are distributed across object storage pools without lock-based contention, and reads use separate data streams. This separation of I/O paths is critical in maintaining smooth mixed workload operation.

The result demonstrates the robustness of PSSA for hybrid workloads, such as those encountered in business intelligence pipelines and real-time dashboards. [10,11]

3.4 Metadata Latency

Metadata lookup and file stat operations are among the most frequent operations in large analytic pipelines, especially during parallel scanning of partitioned datasets.

- **Latency Comparison:**
  - In PSSA, average metadata latency was measured at 2.5 milliseconds, with the 95th percentile at 5 milliseconds.
  - On NAS, the average was 7.5 milliseconds, with 95th percentile spikes exceeding 15 milliseconds under load. [12]
- **System Load Behavior:**
  - MDS (Metadata Servers) in PSSA maintained <30% CPU utilization even under high concurrency.
  - In contrast, NAS's centralized metadata controller reached full CPU saturation with as few as 10 concurrent requests, leading to degraded performance. [13]
- **Design Impact:** Sharded and parallelized metadata services in PSSA eliminate single-point contention, allowing faster file lookups and parallel directory traversal during complex queries. [14]

This aspect is especially valuable in environments with deep nested file structures and frequent ad-hoc querying.

3.5 Resource Utilization

Efficient use of system resources is critical to both performance and operational cost.

- **Storage Nodes:**
  - DSN (Data Storage Node) throughput consistently exceeded 90% of peak NVMe read/write bandwidth, indicating excellent disk utilization.
  - Network interfaces reported 10–20 Gbps active throughput per DSN, showing that the network fabric was actively utilized rather than becoming a bottleneck.
- **Compute Nodes:**
  - Execution nodes reached 70–90% CPU utilization, indicating that bottlenecks had shifted from I/O to actual computation—a favorable transition in analytic systems.
- **NAS Comparison:**
  - The shared-NAS setup left most compute nodes underutilized due to I/O wait cycles, with CPU utilization rarely crossing 50% under the same workloads.

This demonstrates that PSSA ensures balanced load distribution and full utilization of expensive compute and storage resources, leading to better ROI. [15]

IV. DISCUSSION

4.1 Why PSSA Works

1. **Distributed I/O Paths:** Unlike shared-NAS which funnels all traffic through a central controller, PSSA enables parallel I/O directly between compute and storage nodes.
2. **Decentralized Metadata**: Metadata services are disaggregated, reducing lock contention and internal queueing, thus lowering latency.
3. **Network Utilization:** High-speed networking with RDMA offload ensures the storage layer moves data efficiently, preventing NICs from being bottlenecks.
4. **Workload Adaptability:** Block/object layering allows the system to run diverse analytics engines (SQL, OLAP, ML) with optimized paths.

4.2 Trade-offs & Overhead

- **Complexity:** Both deployment and orchestration are more complex than monolithic or NAS solutions.
- **Metadata Coordination**: Sharded metadata requires consensus protocols (e.g., Raft), which can add complexity and latency for writes—but this is mitigated in read heavy analytic contexts.
- **Recovery and Consistency:** Failure handling and rebuild times of erasure-coded data necessitate robust monitoring and orchestration.

4.3 Cost Performance Comparison

- On commodity hardware, PSSA comes at roughly the same per TB cost as NAS, but delivers up to 3–5× performance enhancement, providing significantly better return on investment.

4.4 Generalization

Our architecture is applicable beyond SQL analytics: workloads such as large scale machine learning, network telemetry, and financial tick analysis benefit similarly from parallel I/O and metadata scaling.

V. CONCLUSION

This paper demonstrates that parallel software storage architectures can dramatically accelerate analytic workloads compared to traditional shared NAS and direct attach solutions. By distributing both data and metadata services across nodes, our implementation realizes 6× query speedups, 3–4× throughput improvement, and robust scaling under concurrency. These gains stem from eliminating centralized bottlenecks, leveraging network and storage parallelism, and intelligently layering block- and object-based APIs.

Although the architecture introduces complexity—in deployment, orchestration, and metadata coordination—it proves cost-competitive and operationally feasible with current open-source and cloud-native technologies. Future research should explore:

- **Dynamic data placement** and query-aware sharding
- **Hybrid in-memory/object layers** for ultra-low-latency
- **Global namespace federation** across data centers
- **Storage–compute autoscaling** for elastic workloads

In conclusion, adopting parallel software storage architectures enables data-driven organizations to meet demanding analytics SLAs, increase concurrency, and future-proof their infrastructure for the next wave of big-data innovation.

REFERENCES

1. TechTarget. Big data storage. [online] Available at: Link
2. Niazi, S., Ismail, M., Grohsschmiedt, S., Ronström, M., Haridi, S. and Dowling, J., 2016. HopsFS: Scaling hierarchical file system metadata using NewSQL databases. arXiv. Available at: Link
3. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S. and Stoica, I., 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI). USENIX Association, pp.15–28. Available at: Link
4. Ghemawat, S., Gobioff, H. and Leung, S.T., 2003. The Google file system. ACM SIGOPS Operating Systems Review, 37(5), pp.29–43. Available at: Link

5. Xu, L., Plank, J.S. and Ding, Y., 2006. Optimizing Cauchy–Reed–Solomon codes for fault tolerant storage applications. In: 5th IEEE International Symposium on Network Computing and Applications. pp. 173–180. Link
6. Fidge, C.J., 1988. Logical time in distributed computing systems. – [On vector-clocks protocol.] Link
7. Jiashu Wu, Y. Wang, J. Wang, H. Wang & T. Lin (2023) 'How does SSD cluster perform for distributed file systems: An empirical study', arXiv. Link
8. Exploring Benefits of NVMe SSDs for Big Data Processing in Enterprise Data Centers (2021) – NVMe delivers up to 35% lower latency vs SATA in TPC H style workloads Link
9. The Transaction Processing Performance Council (TPC) (2025) TPC Benchmarks Overview. Available at: lInk
10. Scality (2025) Object Storage vs NAS: Benefits & Definitions. Link
11. Mikeroyal (2023) MinIO high-performance object storage read/write benchmarking. Link
12. Sun, P., Wen, Y.W., Duong, T.N.B. & Xie, H. (2016) MetaFlow: a scalable metadata lookup service for distributed file systems. Link
13. Niazi, S. et al. (2016) HopsFS: Scaling hierarchical file system metadata using NewSQL databases. Link
14. Macedo, R. et al. (2023) PADLL: Taming metadata-intensive HPC jobs through dynamic, application-agnostic QoS control.Link
15. Kang, et al. (2025) Understanding and Profiling NVMe-over-TCP Using ntprof, USENIX—shows NVMe can deliver millions of IOPS at tens of Gbp Link