Scalable Cloud Storage System: A Drive Storage Platform with Enhanced Analytics Capabilities

Cristian David Casas Diaz
Engineering Faculty
Universidad Distrital Francisco Jose de Caldas
Email: cdcasasd@udistrital.edu.co

Dylan Alejandro Solarte Rincón
Engineering Faculty
Universidad Distrital Francisco Jose de Caldas
Email: dasolarter@udistrital.edu.co

***Abstract*—Cloud storage has become an essential service for individuals and organizations, enabling seamless access to data across multiple devices. However, existing platforms often lack robust analytics capabilities and face challenges in efficiently handling large datasets. This paper presents the design and implementation of a highly scalable cloud storage system inspired by Google Drive, with enhanced analytics capabilities. Our solution integrates distributed database architecture with advanced security features and a comprehensive data analysis layer. Initial testing demonstrates high availability (99.9%), efficient storage management, and advanced analytics capabilities, allowing users and administrators to gain insights from stored data without compromising performance or security.**

***Index Terms*—Cloud storage, distributed databases, analytics, data architecture, database system**

I. Introduction

Cloud storage services have revolutionized how individuals and organizations store, access, and share digital content. These platforms enable users to upload files, organize them hierarchically, and access them from multiple devices, ensuring data availability and persistence. According to IBM, well-designed data architecture is fundamental for organizations to effectively manage, secure, and derive value from their data assets [1].

Despite the maturity of cloud storage platforms, several challenges persist. First, many existing solutions struggle with efficiently handling large datasets and maintaining performance during peak usage. Second, there's limited integration between storage and analytics capabilities, requiring users to employ separate tools for data analysis. Third, security concerns persist regarding data encryption and access control mechanisms.

Previous solutions have addressed some of these challenges. For example, Dropbox implemented block-level file synchronization to optimize bandwidth usage [2], while Microsoft OneDrive focused on seamless integration with office productivity tools [3]. Google Drive pioneered real-time collaboration features but has limitations in its analytics capabilities [4]. However, a comprehensive solution addressing performance, analytics, and security in an integrated manner remains elusive.

In this paper, we present a scalable cloud storage system inspired by Google Drive but enhanced with robust analytics capabilities. Our solution employs a hexagonal architecture pattern, separating core business logic from external interfaces to ensure modularity and maintainability. We implement distributed database systems, with separate storage mechanisms for metadata and file content, and incorporate advanced analytics capabilities allowing users and administrators to gain insights from stored data.

II. Methodology

A. System Architecture

Our platform adopts a hexagonal (ports and adapters) architecture, ensuring strict separation between domain logic and infrastructure. This modularity enables each service to scale and evolve independently. Figure 1 presents the high-level architecture.

B. Domain-Centric Design

The system is organized around six independent domain services:

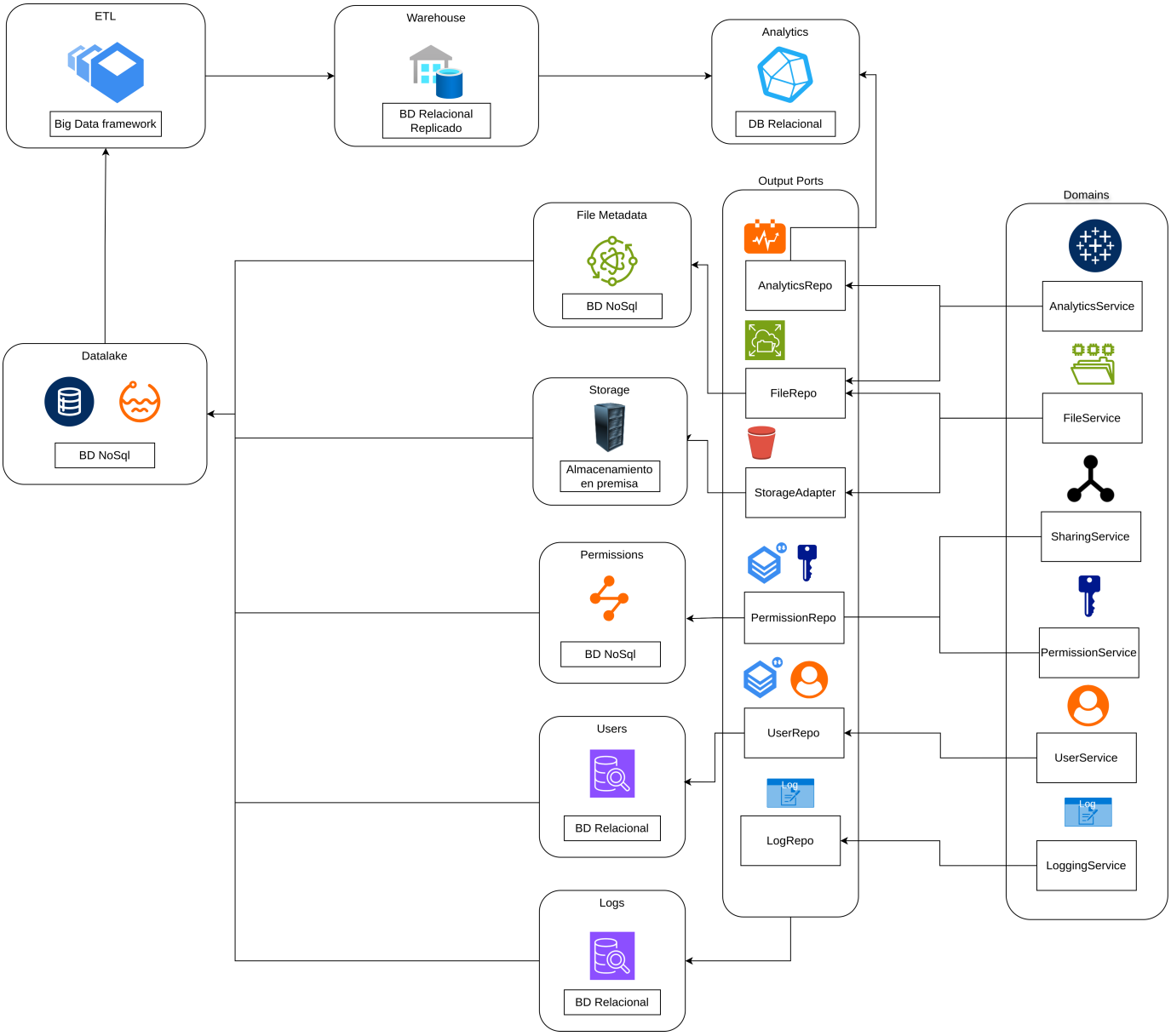
- **UserService:** Manages user accounts, sessions, and subscription plans.
- **FileService:** Handles file metadata, upload/download, versioning, and compression.
- **SharingService:** Manages shareable links and access configurations.
- **PermissionService:** Governs access rights through a graph-based permission model.
- **LoggingService:** Records user activities and authentication events.
- **AnalyticsService:** Aggregates and exposes usage metrics and visual dashboards.

C. Database Design

A hybrid and distributed storage strategy is adopted, selecting the most appropriate model per domain:

- **Relational:** Structured entities such as users, sessions, plans, activity logs, version history, and metrics are stored here to ensure integrity and transactional consistency.
- **NoSQL:** Manages flexible metadata, folder hierarchies, tags, and sharing maps. Suitable for nested or schemaless documents.
- **Graph-based:** Models complex permission relationships and shared resources using community-based partitioning. This facilitates efficient traversal of role-based access and inherited permissions.

Fig. 1: High-level architecture showing hexagonal separation of services, adapters, and storage technologies.



D. *Big Data Architecture Integration*

To support high-volume data workloads and analytical processing, the architecture integrates components commonly found in big data ecosystems:

- A **Data Lake** serves as the initial collection point for raw, semi-structured logs and event data.
- An **ETL module** processes and cleans this data, performing transformations and filtering operations.
- A **Data Warehouse** stores normalized, structured analytics-ready information across user, file, and session domains.
- A **specialized Data Mart** focuses exclusively on metrics consumed by the **AnalyticsService**, enabling low-latency dashboards and predictive models.

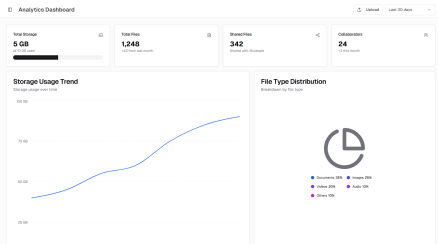
This layered approach supports both batch and real-time analytical queries, aligns with separation of concerns, and optimizes performance under scale.

E. *Concurrency and Scalability Mechanisms*

The architecture supports high concurrency through:

- **Optimistic concurrency control**, using version fields or timestamps to detect update collisions.
- **Atomic operations and short-lived transactions**, especially in fields like used storage, session counters, and version creation.
- **Lock ordering protocols**, applied to folder and file operations, to prevent deadlocks in hierarchical structures.

Fig. 2: Administrator analytics dashboard mockups illustrating storage usage, activity monitoring, and predictive insights.



(a) Storage distribution overview

(b) User activity and access trends

(c) Predictive storage consumption

- **Snapshot reads** for analytics queries, ensuring consistency across time-series reports even during concurrent writes.

These techniques are tailored to resolve common concurrency issues identified during design analysis. These include simultaneous file uploads by different users, concurrent updates to sharing permissions, downloads overlapping with file deletion or movement, and race conditions in storage quota updates.

F. Data Ingestion and Testing

Synthetic data generation was performed using custom Python scripts and the Faker library. The following volumes were inserted across the active domains:

- **User Domain:** 5,000 users, 12,000 sessions, 3 plan types.
- **File Domain:** 20,000 files, 5,000 folders, 100 tags.
- **Logging Domain:** 30,000+ activity/authentication logs, 8,000 file versions.
- **Analytics Domain:** Over 80,000 entries across usage, sharing, performance, and tag metrics.

Due to the absence of an active Neo4j instance, no test data was generated for permission graphs. Likewise, the ETL pipeline and data lake components were architecturally defined but not physically deployed.

G. Storage Optimization Strategies

File content is decoupled from metadata and stored in binary format using a tiered strategy. Although no storage backend was implemented in this phase, the system design assumes an **on-premise storage server** that emulates the behavior of an object store such as AWS S3.

This server would expose a simplified API for upload, retrieval, and deletion of blobs. It is expected to support bucket-based organization, access control, and block-level deduplication. Integration with the FileService is designed via an abstract storage adapter, allowing future replacement or migration to a cloud-native solution.

The intended optimizations include:

- **Chunked file transfers** to ensure reliable upload of large files.
- **Data compression and deduplication** before storage to minimize redundancy and cost.
- **Hot and cold storage tiers**, where frequently accessed files remain readily available while infrequent ones are moved to lower-cost archival tiers.
- **Caching** for frequently accessed metadata and folder structures to reduce latency.

This design balances performance and efficiency while maintaining flexibility to adapt to institutional infrastructure or future cloud deployments.

H. Analytics Capabilities

Although the current system does not implement a functioning analytics engine, a set of visual mockups were developed to illustrate the intended behavior and interface of the future analytics dashboard. These mockups propose features such as:

- **User-level storage and activity insights**
- **Trend analysis** for access behavior and system performance
- **Predictive visualizations** based on historical usage patterns

These components are expected to be powered by the specialized data mart in the analytical stack, aggregating metrics from the data warehouse. The mockups serve as design references for future implementation.

III. Results

To validate the proposed architecture and database model, the system was deployed on Railway using its free-tier PostgreSQL and MongoDB services. Four Python-based scripts were developed to simulate realistic user behavior and generate representative workloads across the main domains.

A. System Deployment

The deployment was separated by domain:

- **Relational Databases (PostgreSQL):** Three separate instances were created for User, Logging, and Analytics domains.
- **Document Database (MongoDB):** One shared instance was used to store folders, files, and tags in a flexible document-oriented format.

Neo4j, which is intended for future use in the permission and sharing services, was not deployed at this stage. As a result, no data was ingested or tested against this database.

TABLE I: Query Performance Summary Across Domains

Query ID	Description	Time (ms)
Relational Queries (PostgreSQL)		
User Q1	Plan distribution & revenue	2.064
User Q2	Premium users >80% quota	2.175
User Q3	Top 10 active sessions	0.657
User Q4	Inactive paid users >90d	3.858
User Q5	Free users >80% quota	2.873
Log Q1	Activity by type/resource	4.781
Log Q2	Failed logins/day/user	2.925
Log Q3	Last successful login	9.218
Log Q4	Files with most versions	13.155
Log Q5	Daily activity summary	4.701
Analytics Q1	Top accessed files	7.662
Analytics Q2	Most active users	4.651
Analytics Q3	Weekly performance trend	0.278
Analytics Q4	Revoked vs created links	5.730
Analytics Q5	Top tags (search & assign)	2.099
Non-Relational Queries (MongoDB)		
NoSQL Q1	Top 10 most downloaded files (last 30 days)	20
NoSQL Q2	Storage usage per user (active files only)	39
NoSQL Q3	Folders with highest file count (top 15)	30
NoSQL Q4	Tag popularity across files and folders	39
NoSQL Q5	Public/confidential file risk detection	16

B. Data Ingestion and Load Testing

Synthetic data was generated using Python scripts and the Faker library, simulating realistic user and system behavior. The following data volumes were inserted:

- **User Domain:** 5,000 users, 5,000 subscription plans, and approximately 12,000 sessions.
- **File Domain:** 5,000 folders, 20,000 files, and 100 tags.
- **Logging Domain:** 10,000 authentication log entries, 20,000 activity logs, and 8,000 file version records.
- **Analytics Domain:** 30,000 user usage metrics, 25,000 file access metrics, 15,000 sharing activity entries, 6,000 system performance snapshots, and 5,000 tag usage metrics.

Batch insertions and proper indexing were used to minimize latency. The ingestion time per script ranged from 10 to 40 seconds depending on volume and complexity. Overall, the system handled over 150,000 records across relational and NoSQL databases without performance degradation.

C. Query Performance Evaluation

To assess performance across domains, representative analytical queries were executed over both relational (PostgreSQL) and non-relational (MongoDB) databases. Table I summarize the execution times in milliseconds.

Relational queries involved joins, filters, and aggregations across structured tables such as 'user', 'activity_log', 'authentication_log', and 'file_access_metrics'. Most queries completed in under 5 ms, with only two exceeding that due to heavy aggregations over version histories and login tracking.

Non-relational queries were performed using MongoDB's aggregation framework on the 'files', 'folders' and 'tags' collections. These operations included groupings, tag-based filtering, and lookup stages. All queries remained under 40 ms, even with pipelines involving multiple stages and nested aggregations.

Comparative Analysis. Relational queries exhibited extremely low response times due to effective indexing, normalized schemas, and the limited dataset size. Most operations executed in less than 5 milliseconds, with peak times observed in queries involving heavy aggregation or complex filtering (e.g., version counts and login tracking).

In contrast, non-relational (MongoDB) queries showed slightly higher execution times, ranging between 16 and 39 milliseconds. This is expected due to the nature of aggregation pipelines, which include `unwind`, `lookup`, and `group` stages over large collections. However, the observed performance remains acceptable and consistent with real-time analytical expectations for document-based models.

Overall, both database models demonstrated strong performance under moderate load, validating the architectural choice of hybrid persistence across structured and semi-structured domains.

D. Analytics Mockups and Intended Behavior

Although an analytics module has not been implemented, a series of dashboard mockups were created to illustrate the envisioned user interface and data visualization capabilities. These visual prototypes showcase:

- Storage usage by file type and user segment
- Access trends and user activity heatmaps
- Predictive insights for proactive storage planning

Figures 2a–2c represent these mockups. They do not depict real-time system output but serve as conceptual guidelines for the future integration of the AnalyticsService with the data mart.

IV. Conclusions

This paper presented the design and partial implementation of a scalable, secure, and modular cloud storage platform inspired by Google Drive, with a particular emphasis on database architecture, analytics integration, and concurrency-aware modeling.

The system adopts a hexagonal architecture pattern that enables separation of concerns between domain logic and infrastructure, supporting maintainability and independent scaling of services. A hybrid persistence model—relational for structured data, NoSQL for flexible metadata, and graph-based storage for future permission modeling—was adopted to address the diversity of storage and access patterns.

Performance evaluations demonstrated that both relational and non-relational queries can be executed efficiently under synthetic workloads, with most queries completing in less than 10 milliseconds. This validates the indexing strategies, aggregation models, and schema design.

Key components such as the AnalyticsService, ETL module, data lake, and data mart were outlined but not fully implemented. Similarly, the on-premise storage server and Neo4j-based database module remain conceptual and were not deployed nor tested in this phase. However, the mockups and architectural hooks lay the groundwork for future integration.

Overall, this work provides a solid foundation for a modular and extensible cloud storage platform with academic and practical relevance in database engineering contexts.

References

[1] IBM, "What is data architecture?" 2025, [Online]. Available: https://www.ibm.com/es-es/topics/data-architecture. [Accessed May 13, 2025].

[2] P. J. Leach, M. Mealling, and R. Salz, "A universally unique identifier (uuid) urn namespace," RFC Editor, Tech. Rep. RFC 4122, Jul 2005.

[3] A. W. Services, "What is data architecture?" 2025, [Online]. Available: https://aws.amazon.com/es/what-is/data-architecture/. [Accessed May 13, 2025].

[4] G. Workspace, "Google drive: Share files online with secure cloud storage," 2025, [Online]. Available: https://workspace.google.com/intl/es-419/products/drive/. [Accessed March 17, 2025].