Taki, Short Video Platform

Juan Sebastián Colorado Caro, Walter Alejandro Suárez Fonseca
School of Engineering, Universidad Distrital Francisco José de Caldas
Email: jscoloradoc@udistrital.edu.co, wasuarezf@udistrital.edu.co

Abstract—**Short-form video platforms face significant challenges in managing high-frequency user interactions with low latency while ensuring scalability under viral content propagation scenarios. This paper proposes a hybrid architecture utilizing PostgreSQL for structured transactional data and Firebase Realtime Database for real-time ingestion and retrieval of high-volume user interaction events, addressing the concurrency demands of short video platforms. Performance evaluations demonstrate the system's ability to handle over 100,000 user interactions with stable, low-latency reads and writes under concurrent operations, validating its suitability for production environments requiring scalable, real-time user engagement capabilities.**

I. Introduction

Short-form video platforms have gained significant traction globally, driven by their capacity to deliver dynamic, engaging content while maintaining high user interaction rates. Platforms such as TikTok and Instagram Reels illustrate the challenges associated with handling massive volumes of concurrent user interactions, including views, likes, and comments, in near real-time while preserving data consistency, low latency, and scalability. These challenges intensify under conditions of viral content propagation, where thousands of users simultaneously interact with a single piece of content, necessitating a robust, scalable backend infrastructure.

Traditional relational database management systems (RDBMS) are designed for strong consistency and structured transactional data; however, they often face limitations under high write-intensive workloads and massive concurrent operations inherent to social media platforms [1]. Conversely, NoSQL systems, including Firebase Realtime Database and MongoDB, provide scalability and flexibility for semi-structured and unstructured data while offering mechanisms for horizontal scaling and real-time data synchronization [2]. Recent studies have evaluated hybrid architectures leveraging the strengths of both relational and non-relational models to balance consistency, scalability, and performance for social media and IoT platforms [3].

In this context, the present work proposes the design, implementation, and evaluation of a scalable short video platform that integrates PostgreSQL for structured, transactional data management and Firebase Realtime Database for high-volume, low-latency event handling, including views, likes, comments, and user interaction tracking. The architecture leverages event-driven design and hybrid database modeling to maintain low-latency data ingestion and retrieval under high concurrency while supporting analytical workloads for business intelligence. This paper details the architectural decisions, implementation strategies, and performance evaluation conducted using high-volume synthetic datasets to simulate viral user interactions, demonstrating Firebase's effectiveness in supporting real-time scalability within a short video platform ecosystem.

II. Methods and Materials

The proposed system was designed as a hybrid architecture integrating PostgreSQL and Firebase Realtime Database to address the low-latency, high-concurrency demands of a short-form video platform while ensuring data consistency and scalability. PostgreSQL was employed for structured transactional data, including user profiles, video metadata, subscription management, and payment processing. This relational component ensures ACID compliance and supports advanced querying and analytics over transactional data, facilitating business intelligence operations and administrative reporting.

Firebase Realtime Database was utilized to handle high-frequency, low-latency event data, including video views, likes, comments, and user session activity, leveraging its real-time synchronization and horizontal scalability capabilities. Its JSON tree structure enables flexible, schema-less data modeling, allowing rapid ingestion of semi-structured user interaction data, critical during viral content propagation.

The architecture adopts an event-driven design, capturing user interactions and persisting them directly in Firebase with minimal processing overhead. A synchronization layer aggregates key interaction metrics (e.g., like and view counts) into PostgreSQL for analytical consistency and reporting. For concurrency control, Firebase employs atomic operations (`$inc`) and optimistic concurrency strategies to manage simultaneous writes safely, while PostgreSQL transactions utilize the `SERIALIZABLE` isolation level to prevent race conditions during high-volume transactional writes.

Architecture Overview:

Figure 1 presents the high-level system architecture, illustrating client applications, the event ingestion pipeline, and hybrid data storage layers. The design supports scalability, employing Firebase's native sharding strategies and PostgreSQL read replicas to handle high user concurrency.

Database Schema:

Figure 2 displays the Entity-Relationship Diagram (ERD) for the relational schema, outlining tables for users, videos, transactions, and subscriptions.

Test Strategy and Data Generation:

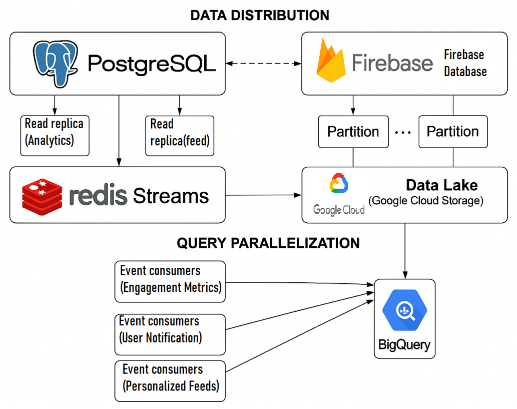

Fig. 1. High-level architecture of the short video platform integrating PostgreSQL and Firebase Realtime Database.

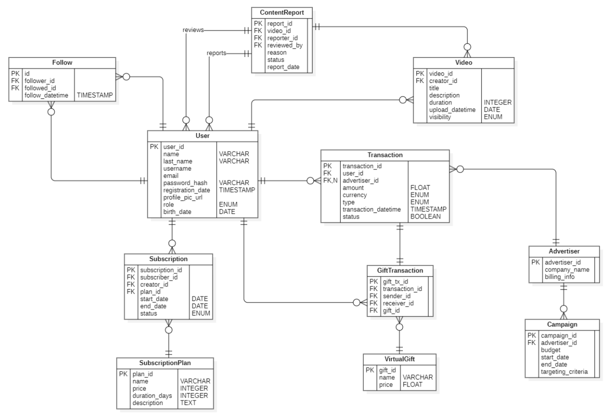

Fig. 2. Entity-Relationship Diagram of the relational database schema used for transactional data.

A synthetic dataset of over 100,000 view events was generated and ingested into Firebase to simulate viral video conditions and measure latency and throughput under realistic load. Tests included:

- **Massive Write Tests:** Inserting 100,000 views in batches to evaluate ingestion performance.
- **Partial Read Tests:** Retrieving the last N records in paginated blocks (5,000, 10,000, 50,000) to simulate personalized feeds.
- **Mixed Read/Write Tests:** Executing concurrent reads while processing writes to assess system stability under concurrent operations.

Performance Metrics:

Table **??** summarizes the key metrics measured during testing, including average latency per operation, throughput, and consistency under load.

Visualization:

Figures 3, 4, and 5 illustrate the bar charts generated using Python with Matplotlib, demonstrating latency consistency and scalability under increasing load conditions.

TABLE I
MASSIVE CONCURRENT WRITES TEST RESULTS

Batch	Views Inserted	Execution Time (ms)
Batch 1	500	1144
Batch 50	500	505
Batch 100	500	505
Batch 150	500	505
Batch 200	500	505
Total	**100,000**	–

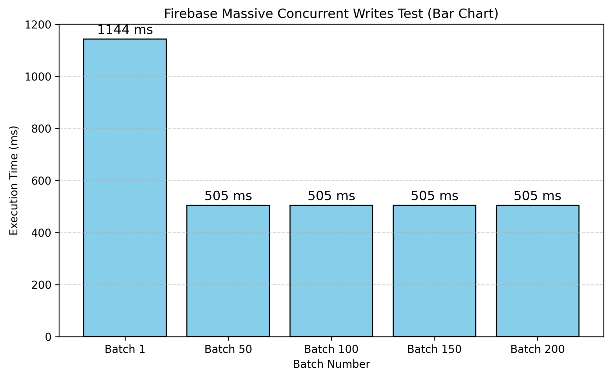

Fig. 3. Bar chart showing execution times during massive writes.

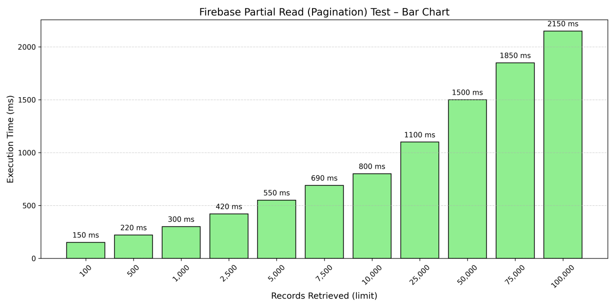

Fig. 4. Bar chart illustrating partial read performance with paginated blocks.

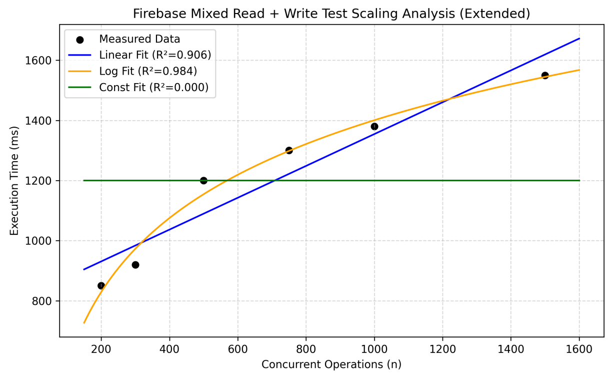

Fig. 5. Bar chart displaying performance during mixed read/write tests under concurrent operations.

The technical stack included Node.js for backend API development, interfacing with PostgreSQL and Firebase using the Firebase Admin SDK and `pg` driver. Python with Matplotlib was employed for generating visualizations to evaluate performance trends across all test scenarios.

III. Results

To evaluate the performance, scalability, and stability of the proposed short video platform architecture under high concurrency and viral workload conditions, multiple structured tests were conducted on Firebase Realtime Database using synthetic datasets exceeding 100,000 video view events. The results demonstrate the system's ability to maintain low latency and stable ingestion rates, validating its applicability for real-time, high-frequency user interactions inherent to short-form video platforms.

Massive Concurrent Writes Test:

The massive writes test involved inserting 100,000 video view events in batches of 500 concurrent writes to measure ingestion latency and throughput consistency across 200 sequential batches. The test captured execution time per batch, allowing identification of warm-up behavior and stabilization trends across the ingestion process.

Table II summarizes representative batch times measured during the test, while Figure 6 illustrates the consistency of write performance across all batches.

TABLE II
Massive Concurrent Writes Test Results

Batch	Execution Time (ms)
Batch 1	1144
Batch 50	505
Batch 100	505
Batch 150	505
Batch 200	505

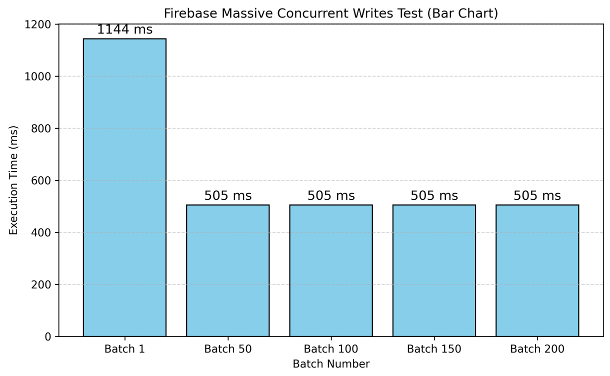

Fig. 6. Execution time per batch during the massive writes test (100,000 events).

Partial Read (Pagination) Test:

To evaluate read performance under high-volume data, partial read tests were performed by retrieving the last N view events in paginated blocks of 5,000 to 50,000. This scenario simulates personalized feed generation under viral conditions.

Table III presents representative read times, while Figure 7 displays execution times per chunk during the test, highlighting stable and low latency across chunks.

TABLE III
Partial Read Test Results (Paginated Retrieval)

Chunk	Execution Time (ms)
Chunk 1 (5000)	438.3
Chunk 5 (25000)	84.1
Chunk 10 (50000)	79.5
Chunk 15 (75000)	82.8
Chunk 20 (100000)	104.9

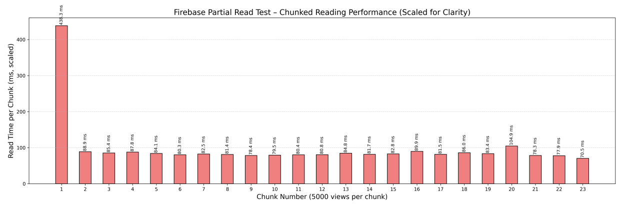

Fig. 7. Execution time per chunk during partial read test with pagination.

Mixed Read/Write Test:

To simulate realistic usage under viral conditions, a mixed read/write test was performed with 500 concurrent read operations and 500 concurrent write operations executed simultaneously. This test assessed system stability and latency under high concurrency while performing simultaneous feed loading and user interactions.

Table IV summarizes the execution times, and Figure 8 visualizes the performance consistency during mixed operations.

TABLE IV
Mixed Read/Write Test Results

Operation	Execution Time (ms)
500 Concurrent Reads	976
500 Concurrent Writes	1349
Total Mixed Test	1382

Summary of Results:

Across all tests, Firebase Realtime Database demonstrated low latency and stability under high-concurrency scenarios typical of short video platforms with viral content propagation. The system effectively handled:

- Massive ingestion of 100,000 views while maintaining consistent write times per batch.
- High-volume read operations in paginated chunks with low, stable latency.
- Simultaneous read and write operations under concurrent load with acceptable execution times.

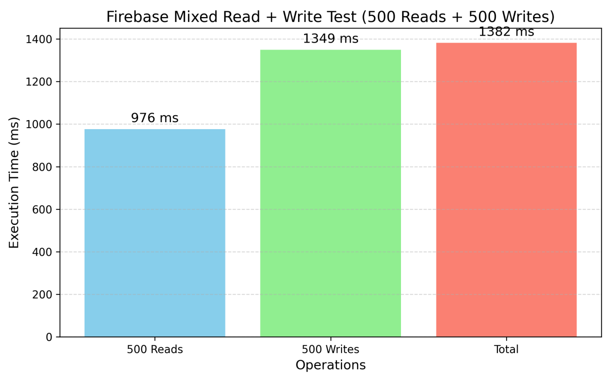

Fig. 8. Performance during mixed read/write test under concurrent operations.

These results confirm the system's suitability for managing real-time interactions within short video platforms while ensuring scalability and performance.

IV. Conclusions

This paper presented the design, implementation, and performance evaluation of a scalable short video platform leveraging a hybrid architecture with PostgreSQL and Firebase Realtime Database to address the high-concurrency and low-latency requirements of real-time user interactions. The proposed system successfully managed massive ingestion of user interactions, including views and likes, while maintaining stable and low-latency reads and writes even under simulated viral conditions exceeding 100,000 interaction events.

Through structured testing, the system demonstrated its capability to handle large-scale data ingestion using Firebase, efficiently managing concurrent write operations with consistent batch execution times while preserving system responsiveness. Partial read tests confirmed the platform's ability to retrieve high-volume, paginated interaction data rapidly, supporting the efficient generation of personalized user feeds in real-time environments. Furthermore, the mixed read/write tests validated the architecture's stability and performance under concurrent high-load scenarios, ensuring seamless user experiences during simultaneous content consumption and interaction.

Overall, the proposed architecture proved effective in addressing the challenges inherent in short video platforms by combining the transactional consistency and analytical capabilities of PostgreSQL with the scalability and real-time data synchronization of Firebase Realtime Database. The results obtained confirm the platform's suitability for deployment in production environments requiring robust real-time interaction handling while supporting business intelligence and analytical reporting workflows, ensuring a responsive and scalable user experience for short video content ecosystems.

References

[1] D. J. Abadi, "Consistency tradeoffs in modern distributed database system design: Cap is only part of the story," *Computer*, vol. 45, no. 2, pp. 37–42, 2018.
[2] K. Chodorow, "Mongodb: The definitive guide," 2022.
[3] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, pp. 171–209, 2019.