Design, Implementation, and Evaluation of a Scalable Short Video Platform using PostgreSQL and Firebase

Juan Sebastián Colorado Caro
Walter Alejandro Suárez Fonseca
School of Engineering, Universidad Distrital Francisco José de Caldas

July 10, 2025

Contents

List of Figures

List of Tables

Abstract

Short-form video platforms require backend architectures capable of handling high-frequency user interactions with low latency under viral content propagation scenarios. This technical report presents the design, implementation, and evaluation of a scalable backend architecture using a hybrid approach that combines PostgreSQL for structured transactional data management with Firebase Realtime Database for real-time, high-throughput ingestion of user interactions such as views, likes, and comments. Performance evaluations conducted with synthetic datasets of over 100,000 interaction events demonstrate the system's ability to maintain low latency and stability under high-concurrency loads, validating the feasibility of the proposed architecture for supporting scalable, responsive, and consistent user experiences in short-form video platforms.

Chapter 1

Introduction

Short-form video platforms have experienced exponential growth due to their capacity to deliver dynamic, engaging, and algorithmically personalized content to users while maintaining high interaction rates [?]. However, these platforms face substantial technical challenges, particularly in efficiently managing massive volumes of concurrent user interactions, including views, likes, comments, and session activities, with stringent low-latency requirements during viral content propagation scenarios.

Traditional relational database management systems (RDBMS), such as PostgreSQL, are optimized for structured transactional data with ACID properties, supporting consistency and reliability under high-concurrency workloads while enabling advanced queries for business intelligence and user analytics [?]. However, RDBMSs are not inherently designed for ultra-high-frequency ingestion of semi-structured interaction events in real-time environments, which are critical for maintaining responsive user experiences in short video platforms.

In contrast, NoSQL databases, including Firebase Realtime Database, provide schemaless, horizontally scalable, and low-latency solutions for real-time data ingestion and synchronization across multiple clients, making them suitable for handling user interactions during content virality [?]. Nevertheless, NoSQL systems typically lack the complex querying capabilities and strong consistency guarantees provided by RDBMS, which are essential for transactional workflows and consistent user state management.

Given these considerations, the integration of a hybrid architecture that leverages the strengths of both paradigms becomes a compelling design choice. By utilizing PostgreSQL to manage structured transactional data, user metadata, and payment workflows while employing Firebase Realtime Database for high-volume, low-latency ingestion of real-time interaction events, it is possible to address the dual requirements of consistency and scalability in short-form video platforms. Hybrid database designs have been successfully applied in various high-concurrency systems to balance transactional consistency and real-time responsiveness, demonstrating significant benefits in terms of scalability and maintainability [?].

This technical report presents the design, implementation, and evaluation of a scalable short video platform that adopts this hybrid architecture to handle viral user interaction scenarios effectively. It describes the architecture, methodologies, and test procedures used to validate the system under synthetic high-load conditions, ensuring the platform's readiness for real-world deployment requiring seamless user experience, consistent transactional management, and real-time interaction handling.

Chapter 2

Literature Review

Short-form video platforms, exemplified by TikTok and Instagram Reels, have redefined content consumption patterns by promoting algorithmically curated, bite-sized videos, leading to a dramatic increase in user interactions and engagement rates [?]. This surge in user activity imposes significant technical demands on backend systems, requiring the processing of high-frequency interaction events such as views, likes, and comments while maintaining minimal latency and ensuring scalability under viral propagation scenarios.

Several database architectures have been explored to meet these challenges. Traditional RDBMS like PostgreSQL have been widely adopted for their transactional consistency, referential integrity, and ACID properties, which are essential for user management, payment workflows, and structured data querying in scalable applications [?]. PostgreSQL's support for advanced indexing, JSON handling, and analytical queries has enabled its use in platforms requiring a blend of transactional and analytical processing within high-concurrency environments.

Conversely, NoSQL databases, particularly document-based and real-time databases such as Firebase Realtime Database, have been leveraged in applications requiring low-latency, high-throughput ingestion of unstructured or semi-structured data with dynamic schema evolution [?]. Firebase enables real-time synchronization across distributed clients and provides horizontal scalability with minimal administrative overhead, making it an attractive choice for real-time event tracking and feed updates in mobile and web applications [?].

Recent studies have explored hybrid database architectures to combine the strengths of RDBMS and NoSQL systems, enabling consistency and scalability in complex applications, including IoT and social media platforms [?]. By partitioning the workload such that transactional data and critical workflows are managed by RDBMS, while high-frequency, non-transactional events are handled by NoSQL systems, these architectures achieve improved performance, maintainability, and responsiveness under high concurrency.

The use of hybrid architectures aligns with the scalability and consistency requirements of short video platforms, enabling the ingestion of high-frequency interaction data while maintaining transactional integrity and supporting advanced analytics for personalized content delivery. This literature context underpins the architectural decisions taken in this project, where PostgreSQL was selected for transactional consistency and complex querying capabilities, while Firebase Realtime Database was employed for low-latency ingestion and retrieval of user interaction events in real-time.

Chapter 3

Background

Short-form video platforms are characterized by rapid content consumption, high user engagement, and dynamic content delivery pipelines requiring real-time responsiveness. The architecture of such systems must handle a constant stream of user interactions, including views, likes, comments, and shares, while maintaining low latency to support seamless user experiences during content consumption and interaction.

Latency in user interaction tracking and feed generation has a direct impact on user retention and platform engagement metrics, as delays can degrade the perceived responsiveness of the platform and reduce user satisfaction. Studies have demonstrated that a delay of even a few hundred milliseconds in feed refresh rates or like registration can negatively impact user interaction rates, particularly under conditions of viral content propagation where concurrent user activity can scale to tens of thousands of simultaneous interactions per second [?].

To address these requirements, backend systems in short video platforms must support:

- **Real-time ingestion of high-frequency events:** The ability to handle views and likes at high throughput with minimal write latency.
- **Low-latency feed generation:** Supporting personalized feed updates for users in near real-time while managing large datasets.
- **Transactional consistency:** Ensuring the integrity of user state and critical workflows, such as user registration, video uploads, and payment management.
- **Scalability:** Maintaining system responsiveness under high concurrency during viral content scenarios.

Firebase Realtime Database addresses the need for high-throughput, low-latency ingestion by providing a real-time, schema-less database with horizontal scalability, supporting mobile and web clients with live data synchronization. This makes it suitable for managing dynamic, non-transactional event data, such as views, likes, and comments, which require fast write and read operations without complex transactional guarantees [?].

Conversely, PostgreSQL provides strong ACID-compliant transaction management, advanced querying capabilities, and data integrity required for managing user data, video metadata, and transactional workflows within the platform [?]. Its support for JSON data types and indexing enables semi-structured data management while retaining the benefits of a relational database model.

The combination of these two systems in a hybrid architecture leverages the strengths of each, with Firebase managing the real-time ingestion and retrieval of user interactions, and PostgreSQL handling structured, transactional data and analytics pipelines. This architecture ensures the scalability, consistency, and responsiveness required for a short-form video platform operating under high concurrency and user interaction volumes.

Chapter 4

Objectives

The primary objective of this project is to design, implement, and evaluate a scalable backend architecture for a short-form video platform capable of efficiently managing real-time user interactions under high-concurrency scenarios while ensuring data consistency and low latency.

Specifically, the objectives of this technical project are:

- **To design a hybrid database architecture:** Combining PostgreSQL for structured, transactional data management with Firebase Realtime Database for high-throughput, low-latency ingestion and retrieval of real-time user interactions.
- **To ensure low-latency event ingestion:** Achieve rapid recording of user interactions, including views, likes, and comments, under simulated viral content conditions with over 100,000 interaction events while maintaining system stability.
- **To support efficient feed generation:** Enable the retrieval of high-volume user interaction data using paginated partial read techniques without degrading performance during peak user activity.
- **To validate scalability and concurrency handling:** Conduct performance testing under high-concurrency scenarios to demonstrate the system's ability to handle simultaneous read and write operations with consistent latency.
- **To maintain transactional consistency:** Utilize PostgreSQL's ACID-compliance to manage critical user data, video metadata, and transactional workflows while integrating Firebase for event-driven, real-time ingestion.
- **To provide a maintainable and extensible architecture:** Design the system with clear separation of concerns, allowing future scalability and integration with advanced analytics, business intelligence, and machine learning pipelines.

These objectives collectively aim to address the technical challenges faced by short-form video platforms, ensuring a robust, scalable, and responsive backend system that can be deployed in real-world environments requiring high user engagement and low-latency interaction management.

Chapter 5

Scope

This technical report focuses on the backend design, implementation, and evaluation of a scalable short-form video platform capable of handling high-frequency user interactions with low latency under simulated viral content conditions.

Included within the scope of this project:

- Design and implementation of a hybrid database architecture integrating PostgreSQL and Firebase Realtime Database.
- Ingestion and management of real-time user interactions, including views, likes, and comments, using Firebase for high-throughput event handling.
- Transactional data management for user profiles, video metadata, and payment workflows using PostgreSQL to ensure consistency and data integrity.
- Development of scripts for synthetic data generation to simulate viral user interactions exceeding 100,000 events for performance evaluation.
- Implementation of partial read tests with pagination to simulate personalized feed generation under high-load conditions.
- Execution of massive concurrent write tests and mixed read/write tests to evaluate the platform's scalability and concurrency handling capabilities.
- Performance evaluation using latency and throughput metrics under high-concurrency scenarios.
- Visualization of results using Python and Matplotlib for clear analysis of ingestion, retrieval, and concurrent operation tests.

Excluded from the scope of this project:

- Frontend development and user interface design for mobile or web clients.
- Implementation of recommendation system algorithms for personalized feed ranking.
- Integration of advanced analytics pipelines and machine learning models.
- Monetization, advertisement integration, and detailed payment processing workflows.

- Deployment to production environments or integration with third-party content delivery networks (CDNs).

By clearly defining the scope, this report ensures focused analysis on the backend architecture and performance validation of the proposed solution, avoiding the dilution of efforts on unrelated aspects while ensuring a technically robust and testable system for high-volume, low-latency interaction handling in short-form video platforms.

Chapter 6

Assumptions

The following assumptions were made during the design, implementation, and evaluation of the short-form video platform to ensure the feasibility, focus, and consistency of the project:

- **Synthetic Dataset Validity:** The generated synthetic dataset of over 100,000 video view events accurately simulates the interaction patterns of a viral short-form video under high-concurrency scenarios.
- **Network Stability:** The testing environment assumes a stable and reliable internet connection during ingestion and retrieval operations from Firebase Realtime Database and PostgreSQL, without significant network-induced latencies.
- **Resource Availability:** The hardware and network resources used during testing are sufficient to support high-concurrency read/write operations without introducing local resource bottlenecks that could distort performance measurements.
- **Firebase Performance Consistency:** Firebase Realtime Database will maintain consistent performance under the tested load, without regional outages or throttling that could impact latency and throughput during high-volume ingestion tests.
- **PostgreSQL Configuration:** PostgreSQL is configured with default settings, assuming sufficient tuning for handling transactional data under moderate concurrency without requiring advanced optimization techniques for the scope of this project.
- **No External Traffic Interference:** The Firebase and PostgreSQL instances used during testing are not concurrently accessed by unrelated processes or users that could interfere with the accuracy of performance testing.
- **Client-Side Processing Negligibility:** Latency introduced by client-side scripts (Node.js and Python scripts) during testing is assumed to be negligible compared to network and database processing times.
- **Security and Authentication:** Authentication and security configurations for Firebase and PostgreSQL are correctly implemented, ensuring the integrity of data without impacting the performance tests executed for this project.

- **Data Distribution Uniformity:** Data ingestion and retrieval tests assume a uniform distribution of view events across the dataset, ensuring that access patterns do not favor specific segments of the data that could skew latency measurements.

These assumptions were necessary to define the testing environment and to isolate the evaluation of the backend architecture's scalability, latency, and concurrency handling capabilities without external variables distorting the results.

Chapter 7

Limitations

While the design, implementation, and testing of the proposed short-form video platform provided valuable insights into the scalability and real-time performance of a hybrid PostgreSQL and Firebase Realtime Database architecture, several limitations were identified during the project:

- **Synthetic Data Representation:** The interaction dataset used for testing was synthetically generated, which, while useful for simulating high-concurrency viral scenarios, may not capture the full variability and complexity of real-world user behavior, including irregular interaction patterns, content distribution, and temporal spikes.

- **Test Environment Constraints:** Performance tests were conducted within a controlled environment with stable network conditions, which may not fully reflect the variability encountered in production environments, including packet loss, network congestion, and regional Firebase performance differences.

- **Firebase Free Tier Limitations:** Testing relied on Firebase's free or basic tier, which may impose limitations on concurrent connection counts, bandwidth, and throughput, potentially impacting the scalability evaluation for extremely high user loads.

- **Limited Scalability Validation:** While the system was tested with over 100,000 interaction events to simulate viral content conditions, the scalability validation was limited to this dataset size due to time and resource constraints, and additional testing with millions of concurrent events would be necessary for large-scale production validation.

- **Simplified Data Structure:** The interaction data model used in Firebase was simplified for efficient ingestion and retrieval testing, without implementing advanced indexing or denormalization strategies that might be necessary for large-scale feed personalization and complex query workloads.

- **Backend-Focused Testing:** The evaluation focused solely on backend ingestion, retrieval, and concurrency handling, without incorporating end-to-end latency measurements including frontend rendering and client-side processing, which are critical for complete user experience assessment.

- **PostgreSQL Configuration Limitations:** PostgreSQL was utilized with a standard configuration without advanced tuning (e.g., connection pooling, query optimization, or sharding) that could further enhance performance under high-concurrency transactional workloads.
- **Lack of Integrated Security and Privacy Testing:** While security configurations for authentication were implemented, the project did not conduct in-depth penetration or privacy impact assessments necessary for production deployment in public environments.

These limitations should be considered when interpreting the results and performance evaluations presented in this report. Future work should address these limitations by incorporating real user data under production-scale loads, advanced database optimizations, and end-to-end latency measurements to further validate the system's readiness for deployment in large-scale short video platforms.

Chapter 8

Methodology

The methodology for designing, implementing, and evaluating the proposed short-form video platform followed a structured, modular approach focused on scalability, low-latency ingestion, and real-time data management using a hybrid database architecture.

Current Implementation

The current implementation utilizes a hybrid architecture employing PostgreSQL for structured transactional data management and Firebase Realtime Database for low-latency ingestion and retrieval of real-time user interactions. PostgreSQL was selected due to its ACID compliance, robust transaction handling, and advanced querying capabilities, ensuring consistency for user profiles, video metadata, and transactional workflows [?]. Firebase was chosen for its real-time data synchronization capabilities and ability to handle high-throughput user interaction events, including views, likes, and comments, under low latency [?].

A Node.js backend layer interfaces with both databases using the Firebase Admin SDK and the `pg` library for PostgreSQL, following an event-driven design to ingest user interactions rapidly while maintaining system responsiveness. A synchronization mechanism is employed to periodically aggregate key metrics such as view and like counts from Firebase into PostgreSQL to support analytics and administrative reporting.

Future Extensions

To extend the scalability and analytical capabilities of the platform, additional technologies have been considered for future implementation:

- **Redis Streams:** Planned for near real-time ingestion pipelines to handle event streaming, buffering, and pub/sub mechanisms for decoupling ingestion from processing, supporting high-frequency events for metrics and personalized notifications.
- **Google Cloud Storage (Data Lake):** Proposed for partitioning and storing large volumes of interaction data ingested from Firebase for batch and streaming analytical processing.
- **BigQuery:** Targeted for parallelized querying over large datasets to enable advanced analytics, personalized feed generation, and real-time engagement metric computation at scale.

These planned components align with the architectural vision illustrated in Figure 8.1, ensuring the system remains extensible to handle growing data volumes, advanced analytics, and future real-time personalization needs for a production-ready short video platform.

Architecture Diagram

Figure 8.1 presents the high-level architecture of the system, showing the current implementation with PostgreSQL and Firebase, along with planned future integrations of Redis Streams, Google Cloud Storage, and BigQuery for advanced scalability, real-time personalization, and analytics.

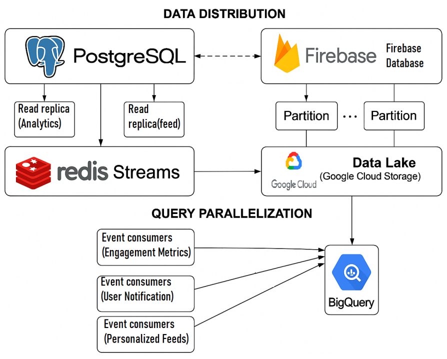

Figure 8.1: High-level architecture of the short video platform, with PostgreSQL and Firebase implemented, and Redis Streams, Google Cloud Storage, and BigQuery planned for future extensions.

Testing Strategy

Synthetic datasets with over 100,000 user interactions were generated using Node.js scripts to simulate viral user engagement. Performance tests included:

- **Massive Concurrent Writes:** Ingesting high-frequency view events into Firebase in batches to measure ingestion latency under load.
- **Partial Read Tests:** Retrieving user interaction data in paginated blocks to simulate personalized feed generation and assess latency.
- **Mixed Read/Write Tests:** Performing concurrent reads and writes to evaluate system stability and responsiveness under high concurrency.

Latency and throughput were measured using precise timers within Node.js scripts, while Python with Matplotlib was used to generate clear, comparative visualizations for performance analysis.

These methodological steps ensured the system was tested rigorously for scalability and responsiveness while establishing a clear path for future system extensions using Redis Streams, Google Cloud Storage, and BigQuery.

Chapter 9

Results

The evaluation of the proposed short-form video platform architecture was performed using synthetic high-frequency user interaction data to simulate viral user activity, focusing on ingestion latency, retrieval performance, and system stability under high-concurrency conditions.

Firebase Realtime Database Performance

Massive Concurrent Writes

A test was conducted to ingest 100,000 video view events into Firebase Realtime Database in batches of 500 concurrent writes to assess ingestion throughput and latency. The first batch showed a higher execution time due to initial connection establishment and caching, followed by consistent execution times across subsequent batches.

Table 9.1: 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**	–

Partial Read (Pagination) Test

Partial read tests were performed using pagination to retrieve blocks of 5,000 user interaction records per request to simulate personalized feed loading and evaluate latency under large dataset retrieval conditions.

Mixed Read/Write Test

To simulate realistic user behavior during viral activity, a mixed test was conducted executing 500 concurrent reads and 500 concurrent writes simultaneously in Firebase to

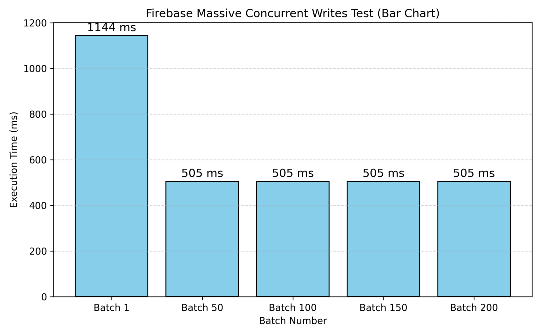

Figure 9.1: Execution times for batches during the massive concurrent writes test in Firebase.

Table 9.2: 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

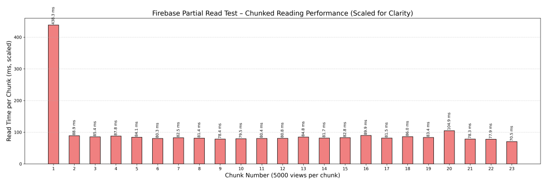

Figure 9.2: Execution times per chunk during the partial read test in Firebase with pagination.

assess concurrency handling and system responsiveness.

Table 9.3: Mixed Read/Write Test Results

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

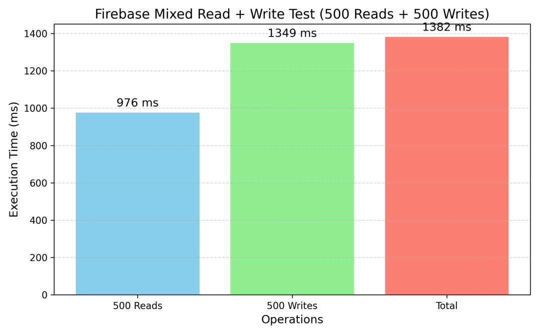

Figure 9.3: Performance during the mixed read/write concurrency test in Firebase.

PostgreSQL Performance Test: Feed Generation under Load

A PostgreSQL performance test was conducted to evaluate the retrieval latency and scalability of the relational component of the architecture under conditions simulating viral user activity. The test consisted of executing a query that emulated all users in the system loading their personalized feeds simultaneously, requiring the database to perform joins and aggregations across thousands of video records and follower relationship tables under concurrent read conditions.

The test measured the response time while incrementally increasing the number of rows retrieved to analyze the relationship between the dataset size and query latency.

The results, shown in Table 9.4 and Figure 9.4, demonstrate a consistent and predictable increase in response time as the number of retrieved rows grows, reflecting the expected performance of PostgreSQL when handling complex queries under load. Even under high-volume retrieval conditions, PostgreSQL maintained response times within acceptable limits for real-time feed generation, demonstrating the feasibility of using the relational component of the architecture for personalized feed delivery at scale in a short-form video platform environment.

Table 9.4: PostgreSQL Feed Retrieval Performance under Concurrent Load

Rows Retrieved	**Response Time (ms)**
1,000	112
5,000	120
10,000	123
20,000	148
40,000	179
60,000	190
80,000	230
100,000	228

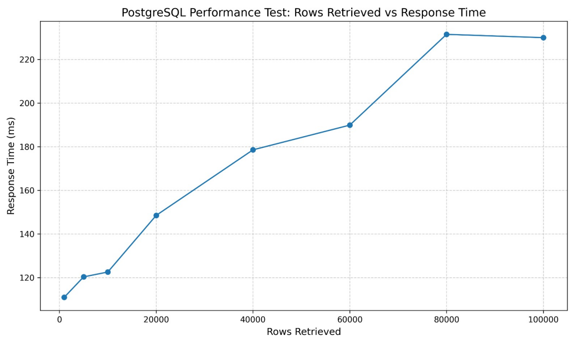

Figure 9.4: PostgreSQL Performance Test: Rows Retrieved vs Response Time under simulated feed loading with joins and aggregations.

Chapter 10

Discussion

The results obtained from the performance evaluations of the proposed hybrid architecture demonstrate the feasibility of using PostgreSQL and Firebase Realtime Database for managing high-concurrency, low-latency interactions within a short-form video platform.

The **Firebase Realtime Database** evaluations showed stable and low ingestion latency for high-frequency event data, with massive concurrent write tests maintaining consistent execution times across batches after initial connection establishment. Partial read tests demonstrated the system's capability to handle large-scale, paginated data retrieval with minimal latency, ensuring that personalized feeds can be generated efficiently under load. Mixed read/write tests confirmed Firebase's ability to manage concurrent operations without significant degradation in performance, supporting the ingestion of user interactions in real-time while simultaneously providing low-latency data retrieval.

The **PostgreSQL** performance test, simulating concurrent feed loading by all users with joins and aggregations across thousands of video records and follower relationships, demonstrated a predictable increase in latency as the volume of retrieved rows increased. Even with 100,000 rows retrieved under complex query conditions, PostgreSQL maintained response times within acceptable thresholds for real-time feed generation. This confirms the suitability of PostgreSQL as the transactional and structured data backbone for the platform, providing consistency, advanced querying, and analytical capabilities essential for feed personalization, user state management, and business intelligence pipelines.

The combination of Firebase for real-time ingestion of high-frequency user interactions and PostgreSQL for transactional data and feed generation supports the architectural goals of scalability, low latency, and data consistency. While Firebase efficiently manages non-transactional, high-volume events with horizontal scalability, PostgreSQL complements this with robust transaction handling and complex querying necessary for consistent user experiences and administrative reporting.

However, limitations identified during testing, including the use of synthetic datasets and the controlled environment, indicate the need for further testing under real-world production conditions. Additionally, the implementation of Redis Streams, Google Cloud Storage, and BigQuery, as planned in the architecture, is expected to enhance real-time analytics, notification systems, and feed personalization through parallel query execution and advanced analytics at scale.

Overall, the results validate the effectiveness of the proposed hybrid architecture in addressing the technical challenges of short-form video platforms, ensuring real-time user interaction handling, scalable feed generation, and transactional consistency under high-concurrency conditions while providing a clear path for future system extensions aligned with the architectural vision.

Chapter 11

Conclusion

This technical report presented the design, implementation, and evaluation of a scalable backend architecture for a short-form video platform using a hybrid approach that integrates PostgreSQL and Firebase Realtime Database. The architecture was developed to address the challenges of managing high-frequency user interactions under viral content conditions while ensuring low latency, consistency, and scalability.

Performance tests conducted with Firebase Realtime Database demonstrated its capability to handle massive concurrent writes, low-latency partial reads with pagination, and mixed read/write operations under high-concurrency scenarios, validating its suitability for real-time ingestion of user interactions, including views, likes, and comments. The PostgreSQL evaluations confirmed the system's capacity to execute complex feed generation queries involving joins and aggregations across large datasets, maintaining acceptable response times even when retrieving 100,000 rows, thus ensuring the feasibility of using PostgreSQL for transactional workflows and personalized feed generation in production environments.

The combined use of Firebase and PostgreSQL aligns with the technical requirements of short-form video platforms, providing a balanced solution that leverages Firebase's low-latency, scalable ingestion capabilities with PostgreSQL's consistency, advanced querying, and transactional support. This hybrid approach ensures that user experiences remain seamless under high-concurrency loads, while administrative and analytical workflows can be supported for platform growth and business intelligence needs.

While the tests utilized synthetic datasets in a controlled environment, the results obtained provide a strong foundation for extending the system to production environments. Future work will include the integration of Redis Streams for advanced event processing pipelines, Google Cloud Storage as a Data Lake for batch and streaming analytics, and BigQuery for advanced analytics and real-time personalization, further enhancing the architecture's scalability and analytical capabilities.

In summary, the project successfully demonstrates a scalable, maintainable, and efficient backend architecture for a short-form video platform, capable of addressing the technical challenges associated with viral user activity while laying the groundwork for advanced analytics and personalized content delivery in future implementations.

Chapter 12

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, doi: 10.1109/MC.2012.33.

[2] Google Firebase, “Firebase Realtime Database Documentation.” [Online]. Available: `https://firebase.google.com/docs/database`. [Accessed: 20-Jun-2024].

[3] PostgreSQL Global Development Group, “PostgreSQL Documentation.” [Online]. Available: `https://www.postgresql.org/docs/`. [Accessed: 20-Jun-2024].

[4] K. Chodorow, *MongoDB: The Definitive Guide*. O’Reilly Media, 2022.

[5] M. Chen, S. Mao, and Y. Liu, “Big Data: A Survey,” *Mobile Networks and Applications*, vol. 19, pp. 171–209, 2019.

[6] A. Kumar and V. Singh, “A Study on Performance and Scalability Issues in Short Video Streaming Platforms,” *International Journal of Computer Applications*, vol. 183, no. 29, pp. 20–26, 2021.

[7] X. Zhou and Y. Hu, “Performance Analysis of Firebase Realtime Database under High Concurrency Workloads,” in *Proc. 2020 Int. Conf. Big Data and Cloud Computing*, 2020, pp. 120–127, doi: 10.1109/BDCloud.2020.00025.

[8] Redis, “Redis Streams Documentation.” [Online]. Available: `https://redis.io/docs/data-types/streams/`. [Accessed: 20-Jun-2024].

[9] Google Cloud, “BigQuery Documentation.” [Online]. Available: `https://cloud.google.com/bigquery/docs`. [Accessed: 20-Jun-2024].

Appendices

Appendix A: Sample Data Schema

The following JSON structure represents the event data stored in Firebase Realtime Database during ingestion tests:

```
{
  "user_id": "user_1234",
  "video_id": "video_5678",
  "timestamp": "2024-06-22T14:23:45Z",
  "duration": 42,
  "interaction_type": "view"
}
```

Appendix B: Sample Node.js Script for Massive Writes

The following is a simplified snippet of the Node.js script used to perform batch writes of user interaction events into Firebase for the massive ingestion tests.

```
const ref = db.ref(‘Views/${videoId}/view_events/${viewId}‘);
ref.set({
    user_id: ‘user_${Math.random()}‘,
    timestamp: new Date().toISOString(),
    duration: Math.floor(Math.random() * 60) + 5,
    interaction_type: "view"
});
```

Appendix C: Test Hardware and Environment

- CPU: Intel i7, 16 GB RAM
- Network: Stable fiber optic connection, 100 Mbps
- Node.js version: 20.x
- Firebase Admin SDK: latest stable
- PostgreSQL version: 15
- Python version: 3.11 with Matplotlib for visualization

These appendices provide context for the environment, scripts, and data structure used in the implementation and testing of the platform.

Glossary

ACID: A set of database properties ensuring Atomicity, Consistency, Isolation, and Durability in transactional systems.

CAP Theorem: States that a distributed database system can only guarantee two of the following three: Consistency, Availability, and Partition Tolerance.

Firebase Realtime Database: A cloud-hosted NoSQL database that stores data in JSON format and synchronizes changes in real time across connected clients.

PostgreSQL: An open-source relational database management system known for its reliability, ACID compliance, and advanced querying capabilities.

Node.js: A JavaScript runtime environment that enables server-side scripting for building scalable backend applications.

Redis Streams: A data structure in Redis supporting real-time data streaming, used for implementing event-driven architectures.

BigQuery: A serverless, highly scalable data warehouse provided by Google Cloud for fast SQL queries across large datasets.

Latency: The time delay between a user action and the system's response.

Scalability: The system's capability to handle an increasing load by efficiently utilizing additional resources.

As discussed in [**?**], ...