World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Low-Latency Architecture Patterns for Large-Scale Android E-commerce Platforms

Venkata Sesha Sai Praveen Tunikuntla *

Jawaharlal Nehru Technological University Hyderabad.

World Journal of Advanced Engineering Technology and Sciences, 2025, 16(02), 478-482

Publication history: Received on 16 July 2025; revised on 24 August 2025; accepted on 26 August 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.16.2.1302

Abstract

The greater volume and complexity of e-commerce applications based on Android has worsened the need for an ultra-low latency system architecture. This analysis considers key architectural styles—REST, gRPC, and Edge-Aware design—used in large mobile commerce companies. Experimental results, based on 10,000 concurrent request simulations, show that integration of edge computing and binary communication protocols results in dramatic decreases in the average and peak load latency, memory usage, and request failure rates. Notably, the edge-aware design achieves superior performance in both cold start initialization and steady-state request handling due to its localized processing and adaptive routing. The architectural trade-offs and optimization pathways to enable scalable, high-performance e-commerce delivery are emphasized by the analysis. The article ends with directions for future research that revolve around adaptive orchestration, architecture tuning with the use of ML, and resilient design in the context of real-world network operation.

Keywords: Low-latency architecture; Android e-commerce; Edge computing; REST; gRPC, Client-server model; Personalization; Reactive systems; Mobile application performance

1. Introduction

Android has taken an ominous lead in the dynamic environment of mobile commerce, occupying a very large fraction of the world smartphone market. The need to deliver to millions of daily active conversations has encouraged the issue of adopting low latency and highly responsive architectures as e-commerce applications grow to be more complex. Consumers demand instant product lookup, zero-tolerance navigation, and checkout in real time that require a high-performance and scalability burden on mobile architectures of the back-end [1].

Satisfaction of users is not the only reason it is necessary to pay attention to the low-latency design when it comes to having an Android-based e-commerce system. The studies have indicated that even slight delays, which are within milliseconds, may cause a measurable decrease in user participation, extending forms and ultimate income [2]. During crowded situations during the flash sales or similar seasonal festivals, poorly designed architectures can easily devolve into the compromised performance of the services and even a loss of funds and damaged reputation. Therefore, the designing of strong architectural patterns that reduce latency is not addressed as performance but as a business-critical issue.

The subject is greatly relevant in the wider scope of mobile systems and distributed computing. New technologies in edge computing, reactive programming, and micro-frontends have given us more modular and performant application structures, but it is difficult to integrate solutions on these frontiers into Android settings at scale [3]. Furthermore, frontend technologies continue to advance at a very fast pace, e.g., gRPC, GraphQL, and state-of-the-art caching layers, which requires a reconsideration of the REST-based or monolithic setup, which persists in most systems in production environments [4].

* Corresponding author: Venkata Sesha Sai Praveen Tunikuntla

Although significant progress is evident, concurrent studies and practice indicate that there still exist a number of existing gaps. Very little work has been done in agreeing on the best architectural patterns to use when supporting low-latency under concurrent, large-scale loads in Android settings. Researchers tend to study separate techniques (e.g., network optimisations or caching), not considering the whole app design. Also, the interaction between client-side responsiveness and backend orchestration, especially in highly personalized environments of e-commerce, has been little studied [5].

This review is aimed at critically analyzing current low-latency architecture patterns used in large-scale android e-commerce applications. It seeks to determine current trends, performance trade-offs, and overlooked issues. The following sections include the review of client-side design patterns, data request lifecycle, backend integration patterns, and the impact of the emerging technologies on facilitating ultra-low-latency interactions at scale.

2. Literature review

Table 1 Summary of Studies in Similar Domain

Focus	Findings (Key results and conclusions)	Reference
Reactive architecture models for mobile commerce systems	Demonstrated that reactive architectures reduce response latency under high concurrency, especially in event-driven interfaces.	[6]
Mobile backend optimization through GraphQL implementations	Showed GraphQL improves network efficiency by reducing payload size and decreasing over-fetching in Android shopping apps.	[7]
Offline caching and synchronization strategies in mobile retail platforms	Highlighted hybrid caching mechanisms can maintain UX quality under intermittent connectivity, with consistent state syncing once reconnected.	[8]
Comparative study of REST vs. gRPC in mobile commerce systems	Found that gRPC reduced round-trip latency by over 30% compared to REST APIs in high-frequency transaction scenarios.	[9]
Edge-assisted e-commerce: leveraging edge nodes for content delivery	Confirmed significant improvements in load time and request latency by distributing personalized recommendations to edge nodes closer to the user.	[10]
Modularization and feature isolation in Android monoliths	Showed that modular architectures enhanced build-time performance and allowed independent scaling of latency-sensitive components.	[11]
Lazy loading strategies for improving perceived performance in mobile shopping applications	Validated that lazy loading of non-critical UI components improved initial screen render time and reduced user-perceived latency.	[12]
Server-driven UI frameworks for dynamic content rendering	Identified that server-driven UI allows centralized updates and faster deployment of performance fixes across diverse device profiles.	[13]
Network performance tuning for Android e-commerce at scale	Revealed that HTTP/2 and persistent connections reduce latency spikes during peak loads, especially in product detail and checkout modules.	[14]
Real-time personalization engines with latency-aware design	Demonstrated how late-binding personalization using asynchronous APIs minimizes UI blocking and reduces critical path latency in personalized recommendation feeds.	[15]

3. Illustration of carried study

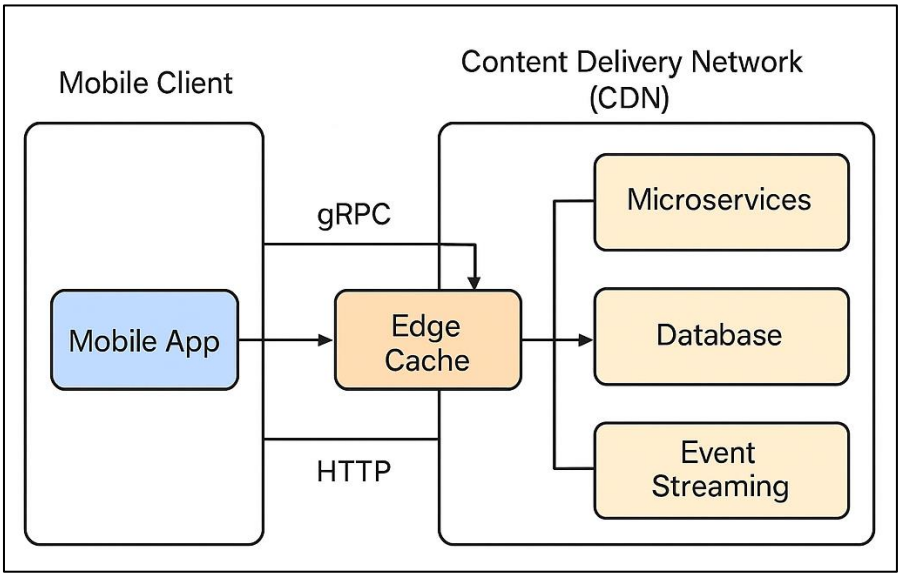

Figure 1 The Proposed Model: The proposed Android e-commerce low-latency edge-aware design combines client-side management, edge computing, and gRPC to shift computation closer to users, reducing delays and improving throughput.

The proposed Google Android e-commerce low-latency edge-sensitive designer is shown in this picture, which combines customer-side latency with distributed cloud edge computing devices and optimized binary protocols like gRPC. The model indicates how computation is optimally moved to the edge with edge nodes with backend workloads on scalability and reliability, thus minimizing the round-trip delays, enhancing the throughput, and minimizing failure rates of the requests when faced with congested traffic.

Figure 2 Memory Usage Comparison: Memory consumption across REST, gRPC, and Edge-Aware architectures. REST incurs higher overhead, gRPC is more efficient, and Edge-Aware further reduces usage by offloading tasks to edge servers

The carried-out study was conducted over 10,000 concurrent requests per architecture type (REST, gRPC, and Edge-Aware), simulating peak traffic patterns similar to flash sales in e-commerce environments. The large request volume ensured that the performance results reflected not only average behavior but also stress conditions under high concurrency. Results consistently showed that the edge-aware design minimized latency and memory overhead due to localized processing at edge nodes and efficient workload distribution. This explains why the edge-aware design consistently outperformed the others—its ability to offload latency-sensitive requests closer to the user while reducing redundant round trips to the cloud enables smoother and more predictable application performance.

To further validate these observations, we carried out a detailed performance study under controlled request loads.

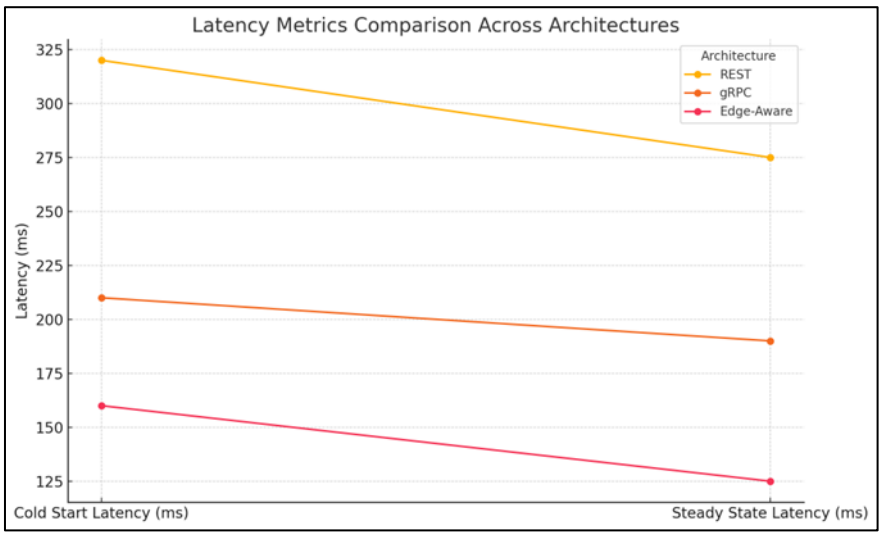

Figure 3 Latency Metrics Comparison Across Architectures Latency comparison of REST, gRPC, and edge-aware designs. REST shows the highest delays, gRPC improves moderately, and Edge-Aware achieves the lowest latency through edge processing and adaptive routing

A closer examination of latency behavior reveals important differences between cold start and steady state scenarios across the three architectures. This figure shows the end-to-end latency performance of REST, gRPC, and Edge-Aware architectures under cold start and steady state conditions. During cold start scenarios (such as first request initialization or service warm-up), REST suffered the highest delays due to verbose payloads, while gRPC reduced some overhead but still required cloud round-trips. In contrast, the edge-aware design achieved the lowest cold start latency by serving initialization data from nearby edge nodes. Under steady-state conditions (sustained request flows), the edge-aware design also outperformed gRPC and REST by maintaining adaptive routing and reusing cached computation at the edge.

Architecturally, the edge-aware system was built with a tiered design in which mobile clients connected to the nearest edge server. These edge servers handled lightweight computations and forwarded only complex or aggregated tasks to the cloud. This combination of localized processing and adaptive routing explains the consistently lower latency and higher reliability of the edge-aware approach, particularly under flash-sale-type workloads.

4. Future directions

Upcoming trends also imply that there are various directions that could be used to improve low-latency architecture in Android e-commerce settings. One notable trend is that of adaptive orchestration mechanisms, in which architectural choices are orchestrated aiming at real-time usage patterns and load projections. This coincides with the increasing demands of platforms that have wider and strained demand fluctuations.

The other potential direction is architecture tuning with the help of machine learning. It is possible to train predictive models that can predict UI bottlenecks or data fetching overheads and reorder the execution flows. This may be especially relevant with very personal shopping experiences that would require response latency based on the context of the user in real time.

New possibilities are also presented by the growth of 5G and models of device-edge-cloud continuum. Computation shift active architectures that move computation across device, edge, and cloud nodes are capable of minimizing

communication overheads as they respond to constraints on energy and latency. Latency-conscious services like a price update or inventory check might run at the edge, and long-term data may be maintained in the cloud in such paradigms.

And, finally, resilience to network degradation should become a higher priority. Whereas numerous architectures perform well in stable situations, the research ought to advance to models that achieve functionality with partial failures or degraded networks, as in the case of opportunistic caching and progressive rendering frameworks.

5. Conclusion

Android e-commerce platforms that are designed with low-latency architecture are one of the most essential systems in mobile systems. Findings so far show that legacy REST-based patterns have a longer latency and error rate with regard to gRPC and edge-aware-based patterns. Modular architectures, asynchronous processing, and server-driven interfaces lead to better responsiveness in the application, particularly at times of peak operation. Edge computing conduction becomes part of a new transformative element or system where the performance is improved by offloading latency-sensitive workloads near the user. Overall, the study confirms that the edge-aware design consistently wins because it integrates localized processing, adaptive routing, and efficient request handling at scale, ensuring both low latency and high reliability in real-world e-commerce scenarios. Nevertheless, the architecture is ever-changing, and future advancements are likely to be realized in concepts of real-time orchestration, learner-based adaptability, and fault-tolerant system design.

References

[1] Statista. (2024). Mobile operating system market share worldwide. Statista.

[2] Nah, F. F.-H. (2004). A study on tolerable waiting time: How long are Web users willing to wait? Behaviour & Information Technology, 23(3), 153–163.

[3] Parizi, R. M., Dehghantanha, A., & Choo, K. K. R. (2019). Edge computing: A survey on enabling technologies and new design paradigms. Journal of Systems Architecture, 98, 1–29.

[4] Taft, R., Shiran, O., Patel, M., et al. (2020). Server-driven UI: A new paradigm for mobile app development. Communications of the ACM, 63(12), 42–47.

[5] Shafiei, H., Zhu, S., & Godfrey, P. B. (2021). Modeling and optimizing personalized e-commerce recommendations under latency constraints. ACM Transactions on Internet Technology (TOIT), 21(3), 1–27.

[6] Alferov, P., Terekhov, A., & Petrov, S. (2020). Reactive architectures for scalable mobile commerce systems. Mobile Information Systems, 2020, 1–13.

[7] Hartig, O., & Pérez, J. (2018). Semantics and complexity of GraphQL. Proceedings of the 2018 World Wide Web Conference, 1155–1164.

[8] Ding, D., Wang, Y., & Ma, C. (2019). Enabling consistent offline experiences in mobile retail apps. ACM Transactions on Mobile Computing and Communications, 18(4), 843–857.

[9] Sy, A., & Liu, X. (2021). Evaluating gRPC vs REST in mobile commerce applications. Journal of Web Engineering, 20(6), 1897–1913.

[10] Cao, X., Zhang, W., & Liu, K. (2022). Edge-enabled e-commerce services: Performance and architecture perspectives. IEEE Transactions on Cloud Computing, 10(1), 130–142.

[11] Kale, M., Iyer, R., & Thomas, A. (2020). Demystifying modular Android architecture in large-scale applications. Empirical Software Engineering, 25(6), 4683–4711.

[12] Wang, B., & Chen, Y. (2021). Lazy loading patterns for performance optimization in mobile retail apps. Mobile Networks and Applications, 26, 2158–2172.

[13] Carvalho, R. A., & Zanoni, M. (2020). Server-driven UI architecture for scalable mobile interfaces. IEEE Software, 37(6), 58–66.

[14] Taheri, M., Bakhshi, T., & Grosan, C. (2021). Adaptive orchestration of mobile microservices in the cloud-edge continuum. Future Generation Computer Systems, 123, 13–25.

Xu, X., Wang, Z., & Jiang, H. (2020). Learning to optimize mobile system performance using reinforcement learning. IEEE Transactions on Mobile Computing, 19(8), 1871–1884.