Ano VII, v.1 2026 | submissão: 02/06/2026 | aceito: 05/06/2026 | publicação: 08/06/2026

Latency reduction techniques in real-time processing pipelines with edge computing and serverless

Técnicas de redução de latência em pipelines de processamento em tempo real com edge computing e serverless

Técnicas de reducción de latencia en pipelines de procesamiento en tiempo real con computación en borde y serverless

Lucas Mohallem Ferraz

Abstract: This article examines latency reduction techniques in real-time processing pipelines, with an emphasis on the convergence of Edge Computing and Serverless architectures. The discussion adopts a historical-evolutionary approach, tracing the path from the consolidation of cloud computing—defined by NIST as on-demand access to shared, elastic, and measurable computational resources—to the decentralization promoted by edge architectures (NIST Technical Publications). It is argued that latency reduction depends not only on greater computational capacity but also on architectural decisions about processing placement, functional granularity, asynchronous communication, data partitioning, observability, and cold-start mitigation in serverless platforms.

Keywords: Edge Computing. Serverless. Latency. Real-time processing. Distributed systems.

1 Introduction

The evolution of distributed systems has progressively shifted the computational axis from centralized data centers toward hybrid, elastic, and geographically distributed architectures. Initially, computational models relied heavily on large, centralized data centers that stored, processed, and distributed information to clients and applications. However, the exponential growth in data volume and the number of connected devices made the need for more flexible and scalable architectures increasingly apparent. In this context, cloud computing consolidated concepts such as elasticity, self-service, and pay-as-you-go pricing, enabling greater operational efficiency and reduced infrastructure costs.

Despite the advantages offered by cloud computing, modern time-sensitive applications have begun to face challenges related to latency and network variability. In environments where system responses must occur within milliseconds, the physical distance between the user, the data source, and central data centers can significantly undermine perceived performance. This problem became even more pronounced in critical applications such as healthcare monitoring systems, autonomous vehicles, augmented reality, and connected industrial environments, where even minimal delays can cause significant operational impact or safety risks.

Against this backdrop, computational paradigms focused on decentralizing processing have emerged, most notably Edge Computing and Serverless Computing. Edge Computing brings processing and storage closer to the sources generating events, reducing the time required to transmit data to the central cloud. The Serverless model, in turn, introduces an operational abstraction based on event-driven execution and automatic scalability, reducing the need for direct infrastructure management. The combination of these approaches represents an important architectural evolution for modern distributed systems, especially in real-time processing scenarios.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.1 2026 | submissão: 02/06/2026 | aceito: 05/06/2026 | publicação: 08/06/2026

2 Historical Evolution: From Centralized Cloud to Programmable Edge

Historically, distributed computing has undergone various structural transformations driven by network growth and the demand for greater computational efficiency. The first major shift came with infrastructure virtualization, a technology that enabled the consolidation of multiple virtual machines on a single physical host. This innovation improved resource utilization and reduced operational costs, laying the groundwork for the rise of cloud computing. Virtualization also supported the expansion of scalable resource provisioning models, allowing organizations to consume infrastructure in a flexible, on-demand fashion.

The second significant transformation was the consolidation of cloud computing through the IaaS, PaaS, and SaaS models. These models redefined how applications were developed, deployed, and consumed, enabling elasticity, high availability, and infrastructure abstraction. However, as applications became increasingly dependent on real-time responses, the limitations of centralized architectures grew more evident. Rising data traffic, the need for continuous communication among smart devices, and the expansion of the Internet of Things intensified issues related to latency, bandwidth, and reliance on remote data centers.

In response to these limitations, the Edge Computing paradigm emerged, characterized by decentralized computational processing. Rather than routing all data to central servers, a portion of the processing occurs near the origin of events, reducing delays and improving operational efficiency. Simultaneously, the Serverless paradigm emerged as an evolution of cloud computing, simplifying application execution through server abstraction and event-driven, automatic scalability. As a result, Edge Computing and Serverless have come to represent complementary pillars in the design of modern architectures for latency-sensitive distributed applications.

3 Latency in Real-Time Pipelines

Latency in real-time processing pipelines refers to the time interval required for an event to be captured, processed, and delivered to the end consumer. In modern distributed architectures, this process involves multiple stages, including event capture, data serialization, network transmission, queuing, processing, persistence, and consumption. Each of these phases introduces small delays that, when accumulated, can compromise the system's overall performance. In critical applications such as financial systems or industrial monitoring, even differences of a few milliseconds can directly affect the user experience and operational reliability.

In distributed systems, latency analysis cannot be limited to the arithmetic mean of response times. Statistical metrics such as p95, p99, and p99.9 have become essential for accurately evaluating service quality, as they represent the worst-case scenarios users experience. In many mission-critical environments, a small fraction of extremely slow requests can be enough to breach service-level agreements and cause operational failures. Therefore, modern architectures must be designed not only with average performance in mind but also with stability, predictability, and resilience in the face of load fluctuations.

The primary sources of delay in distributed pipelines include geographic distance between components, network congestion, inefficient serialization, chained synchronous calls, queue contention, and delayed initialization of serverless functions. Additionally, issues related to remote storage, strong data replication, and the absence of backpressure mechanisms can significantly increase response times. Addressing latency, therefore, requires a systemic approach that accounts for both the physical aspects of infrastructure and the architectural decisions governing data flow and the organization of distributed services.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.1 2026 | submissão: 02/06/2026 | aceito: 05/06/2026 | publicação: 08/06/2026

4 Architectural Latency Reduction Techniques

One of the primary latency reduction techniques involves processing at the network edge, known as Edge Computing. In this model, operations such as filtering, aggregation, validation, and lightweight inference are performed close to the data source, reducing the volume of information transmitted to the central cloud. This approach significantly decreases round-trip time for requests while also lowering communication costs and bandwidth consumption. In large-scale distributed applications, edge computing also improves system availability by allowing certain functionalities to continue operating even when connectivity to central regions is disrupted.

Another relevant technique is functional pipeline partitioning, in which different processing stages are distributed according to their time sensitivity. Operations requiring immediate responses remain close to the data source, while analytical, historical, or lower-priority tasks are offloaded to cloud-based central environments. Complementing this, event-driven architectures use queues, streaming brokers, and distributed logs to decouple producers from consumers, reducing synchronous dependencies between services. This decoupling increases system scalability and eliminates bottlenecks associated with concurrent processing.

In addition, techniques specific to the serverless paradigm have become essential for latency reduction. These include cold start mitigation, the use of provisioned concurrency, lazy initialization, and the adoption of lighter-weight runtimes. Hierarchical caches, selective data replication, and incremental processing based on windows and partial aggregations also play a key role. Rather than recalculating complete states, systems maintain continuously updated summary information, thereby reducing the computational cost of real-time operations. The combination of Edge Computing, Serverless, and advanced architectural optimizations enables the construction of highly responsive and scalable pipelines.

5 Proposed Reference Model

A modern low-latency pipeline based on Edge Computing and Serverless can be structured across multiple specialized layers. The first layer corresponds to data-source devices, including sensors, embedded systems, mobile devices, and browsers. These elements act as continuous event producers and represent the primary source of data generation in contemporary distributed applications. The growth of the Internet of Things has significantly increased the number of connected devices, necessitating architectures capable of processing large volumes of data in minimal timeframes.

The second layer consists of edge nodes responsible for initial processing operations. At this stage, filtering, normalization, authentication, lightweight inference, and caching mechanisms take place. Data can then be forwarded to a regional serverless layer, composed of event-driven functions responsible for enrichment, routing, and persistence. The use of serverless functions enables automatic on-demand scaling, reducing waste of computational resources and simplifying infrastructure management.

Finally, the pipeline incorporates a streaming layer responsible for queues, distributed logs, and backpressure mechanisms, as well as an analytics layer for historical storage, auditing, and training machine learning models. The central architectural decision in this model is determining where each operation will be executed, taking into account time sensitivity, data movement costs, consistency requirements, and regulatory constraints. The proposed reference model thus seeks to balance performance, scalability, and operational efficiency in real-time distributed environments.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.1 2026 | submissão: 02/06/2026 | aceito: 05/06/2026 | publicação: 08/06/2026

6 Critical Discussion

The convergence of Edge Computing and Serverless represents a significant advancement in the design of modern distributed systems, yet it does not eliminate the inherent complexity of these environments. In practice, decentralizing processing reduces latency but introduces new challenges related to distributed deployment, data synchronization, security, and observability. The more geographically dispersed the edge nodes, the more difficult it becomes to manage operations and maintain consistency across system components.

The Serverless paradigm, in turn, significantly reduces the burden of infrastructure management, allowing teams to focus their efforts on developing application functionality. However, serverless platforms can impose limitations on maximum execution time, performance variability, and vendor lock-in. Issues such as cold starts and connectivity constraints can also affect highly time-sensitive applications, particularly in scenarios with unpredictable load patterns or sudden bursts of requests.

Therefore, the architectural design of real-time pipelines must account for multiple factors beyond simple latency reduction. Aspects such as operational cost, portability, governance, resilience, and maintainability must be evaluated holistically. For critical applications, the use of advanced observability metrics, distributed tracing, regional failure simulation, and continuous monitoring of latency percentiles is strongly recommended. Only an integrated approach that combines software engineering and distributed architecture can ensure a proper balance among performance, reliability, and long-term operational sustainability.

Conclusion

Latency reduction in real-time processing pipelines has become one of the foremost challenges of contemporary distributed computing. The growth of the Internet of Things, mobile applications, and intelligent systems has amplified the demand for fast, stable, and scalable responses. In this context, Edge Computing stands out as a fundamental strategy for bringing processing and storage closer to the sources generating events, reducing delays caused by the distance between devices and central data centers.

In parallel, the Serverless paradigm introduced a more flexible and automated operational model based on on-demand execution and automatic scalability. This approach has significantly reduced the effort required for infrastructure administration, enabling greater agility in developing distributed applications. Nevertheless, challenges such as cold starts, performance variability, and vendor dependency demonstrate that adopting these technologies requires careful architectural planning and adequate optimization mechanisms.

It is concluded that the combination of Edge Computing and Serverless represents a mature stage in the evolution of modern distributed systems. However, building low-latency pipelines requires a systemic approach encompassing functional partitioning, asynchronous communication, advanced observability, incremental processing, and the intelligent use of hierarchical caches. Accordingly, organizations seeking to operate critical real-time applications must adopt architectures that balance performance, scalability, reliability, and operational efficiency across highly distributed computational environments.

References

AMAZON WEB SERVICES. What is AWS Lambda? Available at: AWS Lambda official documentation. Accessed: May 7, 2026.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.1 2026 | submissão: 02/06/2026 | aceito: 05/06/2026 | publicação: 08/06/2026

AMAZON WEB SERVICES. Improving startup performance with Lambda SnapStart. Available at: AWS Lambda official documentation. Accessed: May 7, 2026.
JONAS, E. et al. Cloud Programming Simplified: A Berkeley View on Serverless Computing. Berkeley: University of California, 2019.
MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. NIST Special Publication 800-145. Gaithersburg: National Institute of Standards and Technology, 2011.
SHI, W. et al. Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 2016.
SHI, W.; DUSTDAR, S. The Promise of Edge Computing. Computer, IEEE, 2016.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.