International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Google Cloud Functions in Serverless Computing: A Comparative Literature Review and Case Study

Aditya Srikar

Student Researcher, Delhi Public School, Navi Mumbai, India
Corresponding Author Email: *srikaraditya399[at]gmail.com*

Abstract: ***Serverless computing allows developers to run event-driven code without managing infrastructure and has become a key choice for APIs, data pipelines, and automation. This review evaluates Google Cloud Functions (GCF) in comparison with AWS Lambda and Azure Functions, focusing on architecture, performance, cost, and practical usage. Using a narrative review of peer-reviewed studies, official benchmarks, and vendor documentation from 2016–2025, the study synthesizes evidence most relevant to platform selection. Findings indicate that GCF Gen 2 reduces cold start times relative to Gen 1, supports multiple concurrent requests per instance, and integrates tightly with GCP services like Pub/, Firestore, and Cloud Storage. AWS Lambda offers highly predictable cold start latency when Provisioned Concurrency is used, while Azure Functions provides native stateful workflows through Durable Functions and pre-warmed instances. Cost models are similar across platforms, but GCF's generous free tier favors sporadic, short-duration workloads. Overall, platform choice should consider latency requirements, workflow statefulness, ecosystem fit, and workload patterns.***

Keywords: Google Cloud Functions, serverless computing, Functions-as-a-Service, AWS Lambda, Azure Functions

1. Introduction

Cloud computing has fundamentally transformed how organizations deploy and manage applications. One of the most impactful innovations within this domain is serverless computing, particularly Functions-as-a-Service (FaaS), which allows code execution in response to events without manual server provisioning. By abstracting server infrastructure, it enables developers to focus purely on writing code that reacts to triggers or events.

Google Cloud Functions, launched in 2017, is a prominent player in this space, offering developers a simple yet powerful way to build scalable, event-driven systems. Its capabilities have grown significantly with deep integrations into the broader Google Cloud Platform (GCP) ecosystem, allowing streamlined connections with services like Pub/Sub, Firestore, Cloud Storage, and more.

This paper reviews the architecture, use cases, and performance characteristics of Google Cloud Functions, comparing it with other leading platforms such as AWS Lambda and Azure Functions. The literature review will examine each platform's strengths, weaknesses, and trade-offs to provide insights into their suitability for different cloud workloads. Given the growing adoption of serverless computing, understanding how Google Cloud Functions performs in practice is critical for developers, enterprises, and decision-makers seeking cost-efficient, scalable, and low-maintenance solutions for modern cloud-native applications. As organizations increasingly migrate workloads to cloud-native environments, understanding the nuances of FaaS platforms has become vital for optimizing performance and cost.

2. Problem Definition

With several FaaS platforms on offer, organizations are struggling to choose the most suitable one. Decision-makers need to balance trade-offs between latency, cost, orchestration support, ecosystem compatibility, and scalability. Making the wrong choice leads to increased operating expenses, poorer response times, and decreased developer productivity. It is therefore critical to understand the relative strengths and weaknesses of GCF, Lambda, and Azure Functions for deploying effective, cost-efficient, and reliable serverless solutions.

2.1 Paper Organization

The rest of this paper is structured as below. Section 2 offers a systematic literature review of serverless computing and platform- specific features. Section 3 explains the research methodology applied for the narrative review, data sources, and inclusion criteria. Section 4 reports on the findings, performance metrics, cost models, real-world use cases, and comparative analysis, with accompanying comparative analysis. Section 5 concludes with main findings, implications for practice, and directions for future research and platform enhancement.

2.2 Evolution of FaaS

Serverless computing has become central in simplifying application deployment while enhancing scalability and cost-efficiency. AWS Lambda, launched in 2014, was the first major FaaS offering, setting the foundation for serverless architectures. It is widely recognized for enabling event-driven workloads with extensive support for multiple languages such as Python, Node.js, Java, and Go (Spillner et al., 2017; Castro et al., 2019). Azure Functions, introduced in 2016, followed closely, providing enterprise-focused capabilities and integration within Microsoft's ecosystem, particularly with features like Durable Functions for long-running, stateful workflows (Keloth, 2023). Google Cloud Functions (GCF) entered the market in 2017, emphasizing simplicity, developer productivity, and seamless integration with the Google Cloud ecosystem (Hendrickson et al., 2016). Early studies (Adzic & Chatley,

2017; Spillner et al., 2017) highlight the evolutionary trend of serverless platforms in improving scalability, event-driven efficiency, and operational cost reduction.

2.3 Platform Features and Integrations

Each FaaS platform offers unique strengths and integrations. AWS Lambda is known for its breadth of integrations across the AWS ecosystem and multi-language support, making it suitable for complex, high-volume workloads. Azure Functions provides enterprise-focused features, including Durable Functions, which enable orchestrated, stateful workflows and pre-warmed instances to reduce cold start latency. Google Cloud Functions is distinguished by its lightweight, event-driven architecture and tight integration with GCP services such as Pub/Sub, Firestore, and Cloud Storage. This integration facilitates rapid prototyping, short-lived workloads, and streamlined data pipeline implementations, enhancing developer productivity while maintaining scalability (Kodakandla, 2024).

2.4 Recent Advances

Recent studies highlight improvements in GCF Gen 2, including reduced cold start times, support for HTTP/2, higher concurrency limits, longer execution times, and automatic scaling based on CPU utilization (Kodakandla, 2024). These advancements reduce performance gaps with AWS Lambda and Azure Functions, although GCF may still experience slightly higher cold start latencies—typically 0.5 to 2 seconds—compared to ~100–500 milliseconds on AWS Lambda with Provisioned Concurrency (Shilkov, 2021). Researchers have also examined runtime performance variability and language-specific behaviors across platforms (Adzic & Chatley, 2017).

Cost models differ across vendors. GCF offers 2 million free invocations per month (Google Cloud Pricing, 2024), making it cost-effective for short, sporadic workloads. AWS Lambda typically scales efficiently for high-volume or latency-sensitive workloads, while Azure Functions provides flexible pricing tiers, including a premium plan to mitigate cold start delays. These differences inform platform selection based on workload patterns, latency requirements, and ecosystem alignment.

3. Methodology

This study adopts a narrative review methodology to synthesize findings on Google Cloud Functions (GCF) in comparison with AWS Lambda and Azure Functions. The review spans publications and documentation from January 2016 to June 2025, covering the period since the introduction of Functions-as-a-Service platforms.

Sources and Search Strategy. Relevant materials were collected from Google Scholar, IEEE Xplore, ACM Digital Library, and vendor documentation, supplemented with industry benchmarks and case studies. The following search terms were used in various combinations: “serverless computing,” “Functions-as-a-Service,” “Google Cloud Functions Gen 2,” “AWS Lambda,” “Azure Functions,” “cold start,” “pricing,” and “Cloud Run.”

Inclusion and Exclusion Criteria. Studies were included if they addressed architectural features, performance benchmarks, pricing, or real-world applications of FaaS platforms. Both peer-reviewed research and authoritative vendor documentation were considered. Materials were excluded if they were purely opinion-based, lacked empirical grounding, or duplicated findings already captured in higher-quality sources.

Data Extraction. From the included sources, key information was compiled on supported runtimes, trigger mechanisms, scalability, cold and warm start latency, cost models, orchestration features, and application contexts. This ensured consistency when comparing the three major platforms.

Synthesis Approach. Findings were grouped thematically into categories: (i) architecture and platform evolution, (ii) performance and scalability, (iii) cost efficiency, and (iv) developer experience. Additionally, short use-case vignettes were extracted to illustrate practical applications of GCF across domains.

Limitations. Vendor documentation is continually evolving, and reported benchmarks often vary due to differences in experimental design. This heterogeneity, combined with the narrative scope of the review, means that results should be interpreted as representative trends rather than definitive performance rankings.

4. Results and Discussion

4.1 Architecture

Google Cloud Functions (GCF) has undergone significant architectural evolution with its Gen 2 container-based model of execution, where one instance can service multiple concurrent requests. This is different from AWS Lambda and Azure Functions, in which each incoming request usually results in a new container instance being started. The idea is that GCF can be more resource-efficient and have lower infrastructure overhead, especially for bursty workloads. But the trade is less predictability in performance because simultaneous requests will be fighting over CPU or memory in the same container. AWS and Azure's "one request per instance" approach does have more isolation, so it has more consistent per-invocation performance but with higher cold-start frequency when scaling up.

4.2 Performance (Latency and Scaling)

Performance metrics reveal cold starts as the main concern for all FaaS platforms. GCF Gen 2 exhibits much improvement over Gen 1, with lowered cold start time via optimized container reuse and HTTP/2 support. However, it still tends to fall behind

AWS Lambda in general, especially when Lambda utilizes Provisioned Concurrency, which lowers startup latency to ~100–500 milliseconds. GCF's cold starts take anywhere

from 0.5–2 seconds, whereas Azure Functions lie somewhere in between, with slightly lower latency when pre-warmed.

Scaling behavior also differs. AWS Lambda and Azure Functions scale mostly horizontally by provisioning new containers per request. GCF Gen 2 scales horizontally and vertically, serving multiple requests per instance. This enhances throughput in
high-load conditions but can add latency variability under extreme concurrency.

4.3 Cost Models

Each of the three platforms uses the pay-per-invocation plus GB-second cost model, but their free tier and scaling model make practical distinctions.

Google Cloud Functions provides 2 million free invocations per month, which is the most liberal of the three.

AWS Lambda gives 1 million free invocations per month, most popular because of AWS's depth of ecosystem.

Azure Functions provides a comparable free edition (~1 million), but its paid tier offers pre-warmed instances at additional cost to minimize cold starts.

These models suggest that GCF is typically most economical for one-off, short-duration workloads, and AWS Lambda could be more efficient at very high scale. Azure sits in the middle, with enterprise-pro level features that are worth paying more for on some workloads.

Feature / Metric	Google Cloud Functions	AWS Lambda	Azure Functions
Launch Year	2017	2014	2016
Free Tier (per month)	2M invocations	1M invocations	~1M invocations
Concurrency Model	Multi-request per container	One request per container	One request per container
Cold Start Latency	~0.5–2s (Gen 2)	~100–500ms (with Provisioned Concurrency)	~0.5–1.5s (pre-warmed)
Strengths	GCP integration, cost-effective	Lowest latency, AWS ecosystem	Durable Functions, enterprise workflows
Best For	Bursty, GCP-linked workloads	Low-latency, high-scale apps	Microsoft environments

4.4 Application Evidence

Real-world usage validates these results. In healthcare, the Broad Institute uses Google Cloud Functions in the Terra.bio platform to manage genome analysis workflows. The architecture handles petabyte-scale biomedical data by firing compute tasks on data upload and job completion. GCF's close tie with GCP services provides event-driven execution at scale.

Within retail, Zalando has utilized AWS Lambda to deal with high-traffic event-driven workloads in e-commerce, taking advantage of AWS's mature stack and sub-second latency under load. Azure Functions has also gained popularity in enterprise workflows, with Durable Functions offering stateful orchestration that is essential to long-running business processes.

4.5 Platform Choice Guidance

Each platform has evident trade-offs:

Google Cloud Functions is optimal for companies already committed to GCP, especially those with bursty or sporadic workloads that appreciate its large free tier and strong integrations.

AWS Lambda continues as the go-to for applications requiring extremely low latency and taking advantage of AWS's wide ecosystem and established developer tooling.

Azure Functions is most compelling in Microsoft-oriented environments, where Durable Functions enable long-running, stateful workflow support that rivals lack.

In short, there is no one platform that wins on all fronts. Rather, platform selection should be determined by patterns of workload, latency tolerance, and ecosystem fit.

5. Conclusion

This review highlights that Functions-as-a-Service (FaaS) platforms present distinct trade-offs rather than a universal best choice. Google Cloud Functions (GCF) offers ease of integration within the Google Cloud ecosystem and a generous free tier, making it attractive for cost-sensitive, sporadic workloads. However, it continues to face challenges with cold start latencies despite Gen 2 improvements. AWS Lambda remains the industry benchmark for low-latency event-driven applications, benefitting from Provisioned Concurrency and the breadth of AWS integrations. Azure Functions, by contrast, excels in enterprise contexts through features such as Durable Functions, enabling stateful, long-running workflows that neither AWS nor GCF match directly.

The practical implication is that organizations should map workload characteristics to platform strengths: GCF for GCP-linked and bursty tasks, Lambda for highly latency-sensitive applications at scale, and Azure for Microsoft-centric, workflow-driven environments. This workload-driven approach ensures both performance optimization and cost efficiency.

Nevertheless, this analysis is constrained by limitations. Vendor documentation evolves rapidly, benchmarks are often heterogeneous in methodology, and the narrative review approach synthesizes trends rather than producing definitive rankings.

Looking forward, several developments shape the trajectory of FaaS platforms. Edge and hybrid deployments, enabled by services like Anthos and Eventarc, will expand serverless computing beyond centralized cloud regions.

GCF Gen 2's evolving features—such as pre-warming and enhanced concurrency—may further close performance gaps with AWS and Azure. Improvements in observability, particularly through OpenTelemetry, are expected to provide developers with more granular insights into performance bottlenecks. Emerging AI-driven autoscaling mechanisms promise smarter resource allocation, while sustainability commitments, such as Google's goal of operating on carbon-free energy by 2030, position FaaS as a potential driver of greener cloud strategies.

Future research should prioritize systematic benchmarking and reproducible mini-tests across platforms to provide more consistent, transparent comparisons that guide both academic and industry decision-making.

References

[1] G. Adzic and R. Chatley, "Serverless computing: Economic and architectural impact," in *Proc. 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 884–889.
[2] J. Spillner, C. Mateos, and D. A. Monge, "FaaSter, better, cheaper: The prospect of serverless scientific computing and HPC," *arXiv preprint arXiv:1705.01878*, 2017.
[3] M. Roberts, "Serverless architectures," *Martin Fowler Blog*, 2016. [Online]. Available: https://martinfowler.com/articles/serverless.html
[4] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Communications of the ACM*, vol. 62, no. 12, pp. 44–54, 2019.
[5] H. Keloth, "A comprehensive study of Azure Functions: Features, architectures, use case and functionality," *Int. J. Adv. Res. Sci. Commun. Technol.*, Oct. 2023. [Online]. Available: http://dx.doi.org/10.48175/IJARSCT-13132
[6] N. Kodakandla, "Serverless architectures: A comparative study of performance, scalability, and cost in cloud-native applications," 2024. [Online]. Available: https://www.researchgate.net/publication/386876894_Serverless_Architectures_A_Comparative_Study_of_Performance_Scalability_and_Cost_in_Cloud-native_Applications
[7] M. Shilkov, "Comparison of cold starts in serverless functions across AWS, Azure, and GCP," 2021. [Online]. Available: https://mikhail.io/serverless/coldstarts/big3/
[8] Google Cloud, "Event-driven architectures with Cloud Functions," *Google Cloud*, n.d. [Online]. Available: https://cloud.google.com/functions/docs/concepts/overview
[9] Google Cloud, "Reference architecture: Anthos hybrid environment (part 1)," *Google Cloud*, 2023. [Online]. Available: https://cloud.google.com/architecture/anthos-hybrid
[10] Google Sustainability, "2024 Environmental Report," 2024. [Online]. Available: https://sustainability.google/reports/google-2024-environmental-report/