International Journal of Multidisciplinary Research and Growth Evaluation.

Serverless Computing: Impacts on Cloud Development and Data Storage

Nitin Dhyani [1], **Lakshay Babbar** [2]
[1, 2] SGTBIMIT, GT Karnal Road, Near Dhawan Jewelers, Derawal Nagar, Gujranwala Town, Delhi, India

* Corresponding Author: **Nitin Dhyani**

Article Info

ISSN (online): 2582-7138
Volume: 06
Issue: 03
May-June 2025
Received: 02-04-2025
Accepted: 01-05-2025
Page No: 502-505

Abstract
Serverless computing is transforming the way cloud applications are developed and managed. It allows developers to concentrate on writing code without the burden of managing servers, scaling, or infrastructure. This review paper examines how serverless has evolved, its fundamental concepts, and its current status in both research and real-world applications. We assess its advantages, such as automatic scaling and cost-effectiveness, while also addressing its challenges—like cold start delays, limited support for stateful applications, and debugging complexities. By analyzing various research papers and practical case studies, we underscore what functions effectively and what requires further enhancement. The paper additionally explores future directions that could enhance serverless's reliability, efficiency, and applicability to a broader array of applications. In summary, this review provides a comprehensive overview of the future of serverless computing and the necessary steps for it to dominate the cloud.

Keywords: Serverless Computing, Function-As-A-Service, Automatic Scaling, Cost-Efficiency, Cold Start Latency

1. Introduction

Cloud computing has evolved significantly over the years, and one of the most exciting developments is serverless computing. In this model, developers do not have to worry about managing servers or handling backend operations. They simply write small pieces of code called functions, and the cloud provider takes care of everything else. This concept is known as Function-as-a-Service (FaaS), and it's the core of serverless architecture.

To understand how different this is from the traditional cloud model, Table 1 gives a clear comparison. It shows how serverless shifts the focus from infrastructure to just writing and running code, making things easier and more efficient. Now, serverless has its own set of benefits. It's scalable, cost-effective, and great for deploying apps quickly. But of course, it's not all perfect. Problems like cold start delays, time limits on execution, and difficulty in debugging still exist. Interestingly, if we look at Figure 1, we can see how the popularity of the term "serverless" has grown over time, even compared to older technologies like Map Reduce.

In this paper, we're diving deep into what serverless computing brings to the table. From its architecture to real-world use, from benefits to its biggest challenges—everything's covered. And to make it even clearer, Table 2 later in the paper gives a summary of the main challenges in serverless computing, along with possible solutions being explored. The goal is to figure out not just what serverless is today, but where it's headed next.

2. Summary of Papers

The foundational view of serverless computing defines it as a natural next step in cloud abstraction. It highlights how serverless models abstract away infrastructure management, enabling developers to focus solely on code. This work introduces Function-as-a-Service (FaaS) as a core component of serverless, and discusses scalability, fine-grained billing, and event-driven execution as key benefits. However, it also notes that serverless systems still face several research challenges, including state management, performance predictability, and debugging limitations [4].

- The more system-oriented perspective proposes an always-on recording framework to improve debugging and observability in serverless environments. It identifies how traditional observability tools fall short due to the short-lived and stateless nature of serverless functions. The proposed solution involves lightweight trace capturing that enables post-mortem analysis without incurring high-performance overhead, helping developers trace failures and optimise execution paths [5].
- An empirical evaluation of serverless infrastructure is offered in another study, comparing major cloud providers like AWS Lambda, Google Cloud Functions, and Azure Functions. This research measures performance metrics such as cold start latency, scalability, and cost-efficiency. It concludes that while serverless offers strong scalability, cold start times vary significantly across platforms and languages. The study also emphasises that workload type and runtime environment can drastically affect function execution times [2].
- The rise of serverless computing, mapping out trends in both industry adoption and academic research. It identifies how enterprises are rapidly embracing serverless for event-driven architectures, especially in areas like IoT, real-time analytics, and microservices. The study also outlines future research directions, particularly in optimizing function orchestration, improving portability, and designing better tools for developers [3].
- The evolution of serverless computing. It categorizes existing research based on performance, architecture, application development, and security. The paper points out that despite serverless's growing popularity, there is still limited consensus on standardization, and that vendor lock-in and lack of visibility remain significant concerns. It also calls for more studies on real-world production workloads [6].
- The architectural design of serverless systems, describing how FaaS works in conjunction with Backend-as-a-Service (BaaS) to deliver full applications. The paper illustrates use cases across domains like machine learning, data processing, and web applications, and discusses patterns for designing scalable, loosely coupled systems. The research also highlights challenges in integrating serverless with legacy systems and stateful services [1].
- A comprehensive analysis that synthesizes serverless computing's core principles with real-world limitations. It explores the economic model of serverless, pay-per-use billing, and how it influences design decisions. This paper also dives into technical challenges like security, lack of state, cold starts, and limited resource configurations. It concludes by suggesting hybrid models and new abstractions as potential solutions [7].
- Examine serverless computing by contrasting the hype with real limitations. It argues that while serverless solves several pain points, it also introduces complexity in areas like latency, testing, and vendor lock-in. The paper advocates for a more balanced view, recognizing that serverless is not a one-size-fits-all solution. It calls for better tooling, transparent pricing, and standard APIs to make serverless more developer-friendly [8].

Table 1: Comparison of Serverless Computing with Traditional Cloud Models

Aspect	Serverless Computing	Traditional Cloud (IaaS/PaaS)
Infrastructure Mgmt.	Handled entirely by cloud provider	Managed by user (IaaS) or partially (PaaS)
Scalability	Automatic and dynamic	Manual or semi-automatic
Cost Model	Pay-per-execution (per request)	Pay-per-instance (uptime-based)
Deployment Unit	Functions (small code units)	Full applications or containers
Cold Start Delay	Common concern	Not applicable or minimal
State Management	Stateless by default	Can maintain persistent state
Execution Time Limits	Usually restricted (e.g., 15 mins in AWS Lambda)	No fixed time limit
Use Case Fit	Event-driven, microservices, lightweight tasks	Long-running processes, heavy computation
Developer Control	Limited access to underlying infrastructure	Full or partial control (depending on model)

Table 2: Challenges and Solutions in Serverless Computing

Challenge	Description	Suggested Solution
Cold Start Latency	Delay in initialising functions after idle time	Pre-warming, provisioned concurrency
Vendor Lock-in	Dependence on cloud providers' platforms	Multi-cloud deployment
Debugging Complexity	Difficulty tracing errors in distributed systems	Advanced observability tools

Table 2: Summary of Key Serverless Computing Research Contributions

Focus Area	Research Contribution	Key Findings	Limitations/Challenges
Fundamental Concepts & Definition	Provides a foundational overview of serverless computing and FaaS.	Emphasizes abstraction from infrastructure, scalability, and fine-grained billing.	Challenges in state management, performance predictability, and debugging.
Architecture & Use Cases	Explores integration of FaaS with BaaS and illustrates application patterns.	Demonstrates use in ML, web apps, and data pipelines; emphasizes loosely coupled design.	Integration with legacy and stateful systems is difficult.
Empirical Evaluation	Compares AWS Lambda, GCP	Cold start times vary significantly; runtime	Platform-specific behavior; lack of uniformity.
	Functions, Azure Functions.	and workload affect performance.	
Industry Trends & Future Scope	Analyzes adoption trends and outlines future research areas.	Identifies rapid growth in industry usage (IoT, analytics); calls for orchestration tools.	Limited real-world production workload studies.

Literature Review	Categorizes existing serverless research.	Highlights gaps in performance, portability, and standardization.	Vendor lock-in and lack of standard APIs remain major issues.
Observability & Debugging	Proposes always-on tracing framework for serverless.	Enables post-mortem analysis with low overhead; improves failure tracking.	Added overhead, especially in high-throughput systems.
Economic & Technical Analysis	Evaluates serverless pricing, configuration, and resource limits.	Pay-per-use billing influences design; suggests hybrid models.	Security, latency, and state handling remain complex.
Critical Perspective	Provides a balanced critique of serverless "hype".	Notes improvements but questions universality of benefits.	Cold starts, vendor lock- in, testing difficulties persist.

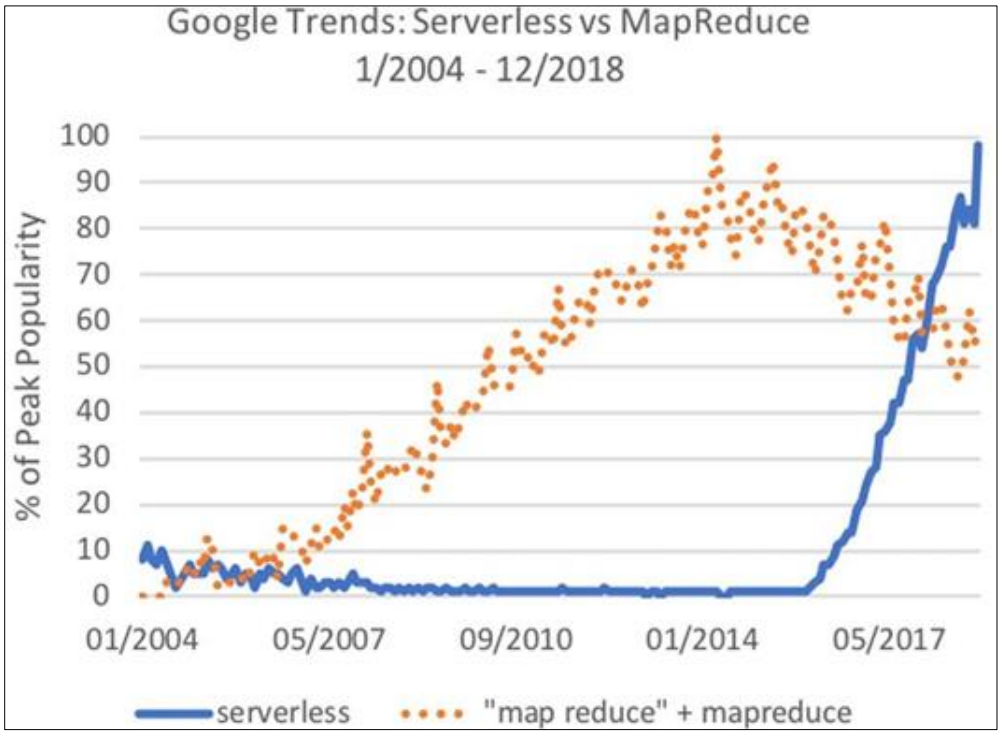

Fig 1: Google Trends for "Serverless" and "Map Reduce" from 2004 to time of publication

3. Conclusion

Serverless computing has rapidly grown as a modern cloud model that simplifies deployment and automatically scales applications without the need to manage servers. Through this review, we have examined how different studies articulate their core ideas, performance, architecture, use cases, and future possibilities. The serverless model is particularly advantageous for event-driven applications, microservices, and short-lived tasks, providing benefits such as reduced operational costs, scalability, and developer flexibility.

At the same time, it's clear that serverless is not a one-size-fits-all solution. Cold start delays, debugging issues, platform dependency, and lack of standardisation are some challenges that still need attention. Some papers suggest using hybrid models or always-on monitoring to overcome these problems.

In short, serverless computing holds great potential, but these limitations must be addressed. More research is needed on portability, orchestration tools, and performance tuning to make serverless more reliable and efficient for a wider range of applications.

4. References

1. Ghorbian M, Ghobaei-Arani M. Serverless Computing: Architecture, Concepts, and Applications. arXiv preprint arXiv:2501.09831 [Internet]. 2025. Available from: https://arxiv.org/abs/2501.09831
2. Bodner T, Radig T, Justen D, Ritter D, Rabl T. An Empirical Evaluation of Serverless Cloud Infrastructure for Large-Scale Data Processing. arXiv preprint arXiv:2501.07771 [Internet]. 2025. Available from: https://arxiv.org/abs/2501.07771
3. Castro P, Ishakian V, Muthusamy V, Slominski A. The server is dead, long live the server: Rise of Serverless Computing, Overview of Current State and Future Trends in Research and Industry. arXiv preprint arXiv:1906.02888 [Internet]. 2019. Available from: https://arxiv.org/abs/1906.02888
4. Jonas E, Schleier-Smith J, Sreekanti V, Tsai CC, Khandelwal A, Pu Q, *et al*. Cloud programming simplified: A berkeley view on serverless computing. arXiv preprint arXiv:1902.03383 [Internet]. 2019. Available from: https://arxiv.org/abs/1902.03383
5. Kharbanda S, Fonseca P. Always-On Recording Framework for Serverless Computations: Opportunities and Challenges. In: Proceedings of the 1st Workshop on SErverless Systems, Applications and MEthodologies; 2023 May; p. 41-49.
6. Wen J, Chen Z, Jin X, Liu X. Rise of the planet of serverless computing: A systematic review. ACM Transactions on Software Engineering and Methodology. 2023;32(5):1-61.
7. Kundavaram VNK. Serverless Computing: A Comprehensive Analysis of Infrastructure Abstraction in Modern Cloud Computing. International Journal For

Multidisciplinary Research. 2024;6(6). https://doi.org/10.36948/ijfmr.2024.v06i06.30737

8. Hellerstein JM, Faleiro J, Gonzalez JE, Schleier-Smith J, Sreekanti V, Tumanov A, *et al*. Serverless computing: One step forward, two steps back. arXiv preprint arXiv:1812.03651 [Internet]. 2018. Available from: https://arxiv.org/abs/1812.03651