

Serverless vs Containerized Microservices in Scalable Cloud Architectures: A Comparative Review

Name: Odai Abdelbaky

Date: July 17, 2025

Academic Code: 252000312

Table of Contents

Problem Statement.....	1
Research Objectives.....	2
Literature Review	2
Conclusion	5
References.....	5

Problem Statement:

There is a lack of clear, evidence-based guidance to help architects choose between serverless and containerized microservices for building scalable cloud-native systems. As organizations continue to adopt cloud-native architectures, architects and developers alike face the challenge of selecting a cloud-native deployment model. The two most common cloud-native deployment models are serverless computing and containerized microservices. The selection process between them is often unclear due to overlapping capabilities, performance trade-offs, and lack of unified decision frameworks.

The core of the problem is the lack of a clear evidence-based guide for decision-makers to determine which model best fits specific workloads, scalability needs, cost limits, and latency expectations. More so, serverless computing platforms, such as AWS Lambda, reduced operational overhead, and enabled systems to scale automatically. However, they typically do not customize workloads to reduce cold-starts, have execution time limits, and introduce vendor lock-in (Al-Khatib et al., 2024; Kambala, 2023). On the other hand, while containerized microservices provide more predictable workload performance and deployment control, they introduce more complexity in orchestration and resource constraints (Mera Menéndez et al., 2023). Many studies offer comparisons or hybrid model approaches, but their findings are limited to platforms. Moreover, none of the findings offer a consolidated framework for practitioners across different environments, such as edge, multi-cloud, or enterprise-scale applications (Manzoor et al., 2024; Navulipuri & Narayan, 2025).

This research aims to analyze recent literature, assess the pros and cons of each deployment model and provide a framework to identify the right architectural style based on workload types, scalability needs, and operational limitations.

Research Objectives

The primary goal of this research is **to evaluate and compare serverless and containerized microservices architectures to support the design of scalable and efficient cloud-native systems.**

To achieve this goal, the research will pursue the following objectives:

- **To Review** existing literature on serverless and containerized microservices in terms of performance, scalability, cost and operational complexity.
- **To Analyze and Compare** the pros and cons of each model depending on workload and deployment context (for example cloud, edge).
- **To Identify** key determinants that help to make architecture decisions, such as sensitivity to latency, amount of concurrency and cost.
- **To Propose** a decision-making framework or guidelines to assist system virtual architecture in deciding the correct architecture between serverless, container, and hybrid models depending on the application requirements.

Literature Review

As cloud-native development advances in popularity and usability, two distinct deployment models dominate which are serverless computing and containerized microservices. Serverless platforms (e.g., AWS Lambda) come with out-of-the-box simplicity, auto-scaling, and reduced cost, whereas containers offer developers greater control, flexibility, and consistent performance. Developers' decisions to utilize serverless or containerized microservices is an essential architectural decision to consider in the modern cloud systems.

This literature review investigates the following question: *Which model serverless or containerized microservice is more effective for building scalable cloud architectures?* To answer the research question, the literature review assesses mined literature of six studies from the academic and peer-reviewed research on the topics of performance, scalability, latency, cost, operational complexity, and use cases in the context of serverless and containerized microservices. Ultimately, this review highlights key insights from current literature and affords developers, engineers, and cloud architects guidance for future architectural decisions.

Paper 1 Serverless Computing vs. Containerization: Comparative Analysis for Cloud-Native Application Deployment

- **Summary:** The objective of this research is to study serverless computing by using AWS Lambda in comparison to containerized microservices deployed via Kubernetes. This research includes testing cloud-native application deployments utilizing both AWS Lambda and Kubernetes. The authors created a sample application in both environments and tested performance along various measures such as cold start latency, scalability, operational complexity, and cost. The authors concluded that serverless computing allows for faster development, lower costs for low-traffic workloads, and almost zero operational overhead. On the other hand, containerized deployments perform better under sustained load and provide researchers with better control over the runtime environment. The authors, therefore, recommend using a hybrid model involving both approaches depending on the workload.
- **Critical Analysis:** This paper provides a straightforward comparison of serverless architectures and containerized architectures, using one application deployed on both AWS

Lambda and in Kubernetes as a case study. One of the strengths of this paper is the applied aspect: the testing of both compute models using the exact same code base provides a fair performance & cost comparison but it also was clear about the operational trade-offs. However, this study is limited in scope. It used only one cloud platform (AWS) and it did not explore more complex workloads, nor the long-term performance trends. The lack of any more in-depth data on real-world deployments of cold-start mitigations also contributes to limiting its usefulness in terms of the bigger picture.

Paper 2 Engineering Scalable Microservices: A Comparative Study of Serverless Vs. Kubernetes-Based Architectures

- **Summary:** In this paper, authors compare and contrast two microservice deployment options and containers, leveraging AWS Lambda (serverless), and Kubernetes (containerized). The authors built the same microservices in the AWS Lambdas and Kubernetes, using Apache JMeter to run the microservices under loads, the important measurements were: cost, latency, and scalability. Serverless is advantageous if your environment is dynamic or has low traffic and can scale automatically with the ability to see costs and scale metrics when deployed. Kubernetes is best suited for higher throughput scenarios where resources need to be managed for cost, allowing persistent control. The authors recommend there must be a hybrid combination of serverless for event-based processes and containers for core service or infrastructure.
- **Critical Analysis:** One of the main strengths of the paper is its overarching benchmarking approach to benchmarking using multiple workloads modeled through Apache JMeter to compare Lambda and Kubernetes. They provide valuable observations and implications around how workloads are classified and offered hybrid recommendations based on workloads. Their quantitative results are based on metrics, response latency, and monthly cost, and are valid and justifiable. One of the salient drawbacks is that the study solely investigated AWS taking away broader implications in other areas like Azure or GCP. They also didn't consider any implications related to CI/CD or DevOps, failure or real-time observability of errors between the two contrasting environments.

Paper 3 Comparing Cost and Performance of Microservices and Serverless in AWS: EC2 vs Lambda

- **Summary:** In this paper, the authors analyzed the performance and cost of deploying a microservice-based API with AWS Lambda and Docker containers running on EC2. To simulate user traffic they used Locust and concluded that serverless (Lambda) deployment was more cost effective and more scalable for small-to-medium workloads but restricted under high-load workloads (especially with cold starts), whereas EC2 containers provided a better throughput and more consistent performance in heavy use cases but required more management of the infrastructure. The paper concludes that your choice of deployment should depend on the volume of use and how sensitive you are to performance issues.
- **Critical Analysis:** This research article provides a beneficial side-by-side comparison of containerized microservices with EC2 (Elastic Compute Cloud) versus AWS Lambda. The analysis is founded upon actual AWS billings in two data centers and the use of real workloads. Its main strength is its real-world cost analysis and clear reporting on throughput performance under increasing load conditions. The study lacks any discussion of orchestration platforms, such as Kubernetes, that could influence the performance of containers. In addition, the study does not address other relevant enterprise adoption concerns such as developer productivity, integration complexity, or ongoing operational maintenance.

Paper 4 Serverless Architectures: A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications

- **Summary:** In this paper, they contrast serverless and containerized architectures on AWS and Azure, using two different cloud-native applications: video encoding and data ingestion pipelines.

They compared both platforms along the following three dimensions: latency; resource utilization; and developer effort.

In terms of their applications, they found that serverless was the better choice for lightweight or event-driven workloads, which are less latency-sensitive, as it was more agile and cost-effective than containerized applications.

Containerized workloads could provide more predictable latency and a better grasp of the amount of resources consumed, and so they would recommend containerized applications for high-performance or stateful applications.

They also proposed a decision rule based upon the high-level property of latency tolerance, which they believe provides a structured way of evaluating container or serverless workloads by applying a good enough heuristic, whereby, for latency-sensitive services, customers should prefer a container, and for flexibility, customers should consider serverless.

- **Critical Analysis:** This study's strength is its multi-cloud evaluation of AWS and Azure, providing further context on how serverless behaves across providers. It is also novel in terms of considering developer effort and utility using a latency-based decision-making framework. The consideration of complex workloads (e.g. video processing, data pipelines, etc) also makes this paper more relevant to the real-world. Nevertheless, this study did not include any container alternatives such as AWS Fargate or edge-based deployments. It also did not include infrastructure-as-code, monitoring practices, or anything in the deployment lifecycle, which are all concepts that are important in production-grade systems.

Paper 5 Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time IoT analytics

- **Summary:** In this paper, the focus is on the edge of computing and the exploration of serverless functions and Docker-based microservices for real-time IoT sensor data analytics. The authors assessed applications deployed on edge nodes for metrics like energy consumption, CPU usage, and cold starts.

The findings indicated that containerized microservices had better latency and resource efficiency on constrained interfacing edges. While the serverless model could be initiated easily. The authors propose a layered architecture in which containers interact and process data at the edge, while serverless functions manage aggregation of data on the cloud side and alerts.

- **Critical Analysis:** The paper considers both a serverless and a containerized microservices architecture of edge computing, which are frequently missed in a lot of cloud architecture studies. The paper also articulates performance measurements CPU utilization, memory, and energy consumption. Nevertheless, the focus on edge devices provides a different angle than the other papers. The workloads were also correspondingly limited to sensor reading workloads that also wouldn't generalize to more demanding edge use-cases like AI inference or video analytics. Likewise, the authors also do not reference hybrid orchestration frameworks like KubeEdge or AWS Greengrass that could be appropriate hybrid solutions to combine serverless and container-based.

Paper 6 Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time IoT analytics

- Summary:** In this paper, they implemented nine microservices on all three platforms (AWS Lambda (serverless), AWS Fargate (managed containers), and Kubernetes (self-managed containers)) and compared AWS Lambda (serverless), AWS Fargate (managed containers), and Kubernetes (self-managed containers).
 The researchers compared the three platforms based on cost, latency, cold starts, and aggregating operational complexity.
 AWS Lambda was shown to be the most economical for spare, short-lived workloads.
 Fargate provided generally scalable performance without cold start issues.
 Kubernetes provided, generally, the greatest control; however, it had the highest level of management effort.
 This paper is used to establish the importance of the application and the patterns of workload to help decide which deployment models to use, and how much operational overhead is acceptable, advocating for hybrid or mixed strategies.
- Critical Analysis:** This is the most comprehensive study of the group that compares Lambda, Fargate, and Kubernetes across real-world microservices instances (nine instances in total). It has very good coverage, as it provides a balanced consideration of latency, cold starts, cost, and operational complexity. Including the AWS Fargate option, as a middle, in the deployment comparison is a valuable addition as it also provides some nuance in thinking about serverless versus containers. However, it does focus exclusively on AWS services and so does limit any insights across platforms. It also does not explore multi-cloud strategies or transitioning from one deployment model to another, which is increasingly becoming common in enterprises today.

Conclusion

This literature review has assessed the advantages, disadvantages, and compromises of serverless computing and containerized microservices when constructing scalable cloud architectures. The six reviewed papers shared a general consensus that these two models serve different use cases: **serverless computing works best for lightweight, event-driven, and dynamic tasks**, while **containers provide better performance, control, and reliability when delivering complex or latency-critical services**.

The general conclusion is simple: the model chosen for deployment should be a function of the inherent workload characteristics, predicted scalability needs, operational objectives, and available infrastructure. An increasing number of scholars and practitioners promote the idea of **hybrid cloud-native architectures** that utilize a blend of serverless and containerized components that deliver the benefits of performance, cost efficiency, and operational maintainability.

As cloud platforms continue to develop, providing new services such as AWS Fargate, Azure Container Apps and edge-native runtimes, more research is needed on the transition paths between containerized and serverless models, security issues, and automation of mixed mode environments. However, this represents a great opportunity for the development of flexible, resilient, and efficient cloud-native systems.

References

Giovanni, Alessandro & Charlotte, Luna. (2025). Serverless Computing vs. Containerization: Comparative Analysis for Cloud-Native Application Deployment.
https://www.researchgate.net/publication/390193315_Serverless_Computing_vs_Containerization_Comparative_Analysis_for_Cloud-Native_Application_Deployment

Tripathi, K. (2025). Engineering Scalable Microservices: A Comparative Study of Serverless vs. Kubernetes-Based Architectures - IJSRET. IJSRET. <https://ijsret.com/2025/04/14/engineering-scalable-microservices-a-comparative-study-of-serverless-vs-kubernetes-based-architectures>

Allen, Christopher & Li, Xiaozhou & Abdelfattah, Amr & Černý, Tom & Taibi, Davide. (2024). Comparing Cost and Performance of Microservices and Serverless in AWS: EC2 vs Lambda. 10.1007/978-3-031-61816-1_5.
[https://www.researchgate.net/publication/381761885 Comparing Cost and Performance of Microservices and Serverless in AWS EC2 vs Lambda](https://www.researchgate.net/publication/381761885_Comparing_Cost_and_Performance_of_Microservices_and_Serverless_in_AWS_EC2_vs_Lambda)

Kodakandla, Naveen. (2021). Serverless Architectures: A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications. 5. 136-150.
[https://www.researchgate.net/publication/386876894 Serverless Architectures A Comparative Study of Performance Scalability and Cost in Cloud-native Applications](https://www.researchgate.net/publication/386876894_Serverless_Architectures_A_Comparative_Study_of_Performance_Scalability_and_Cost_in_Cloud-native_Applications)

Tusa , F. et al. (2024) Microservices and serverless functions-lifecycle, performance, and resource utilisation of edge based real-time IOT analytics, Future Generation Computer Systems.
<https://www.sciencedirect.com/science/article/pii/S0167739X24000529>

Menéndez, J. M., Gayo, J. E. L., Canal, E. R., & Fernández, A. E. (2023) A comparison between traditional and Serverless technologies in a microservices setting. arXiv.org.
<https://arxiv.org/abs/2305.13933>