

Evaluating Serverless and Containerized Microservices: A Literature Review

Odai Atef Attiya Abdelbaky
عدي عاطف عطية عبدالباقي

Table of Contents

<i>Abstract</i>	<i>2</i>
<i>Introduction</i>	<i>3</i>
<i>Background</i>	<i>4</i>
<i>Research Objectives.....</i>	<i>5</i>
<i>Research Scope.....</i>	<i>6</i>
<i>Literature Review.....</i>	<i>6</i>
<i>Critical Analysis</i>	<i>9</i>
<i>Connections</i>	<i>10</i>
<i>Conflicts Between Research Findings.....</i>	<i>11</i>
<i>Synthesis Matrix for Literature Review.....</i>	<i>12</i>
<i>Research Methodology</i>	<i>19</i>
<i>Phase 1: Systematic Literature Search.....</i>	<i>19</i>
<i>Phase 2: Literature Review and Quality Assessment.....</i>	<i>19</i>
<i>Phase 3: Comparative Analysis and Data Synthesis</i>	<i>20</i>
<i>Phase 4: Framework Development & Validation</i>	<i>20</i>
<i>Limitations</i>	<i>20</i>
<i>Expected Outcome</i>	<i>20</i>
<i>Gantt Chart</i>	<i>21</i>
<i>Conclusion.....</i>	<i>21</i>
<i>References.....</i>	<i>22</i>

Abstract

As cloud-native computing continues to develop rapidly, the discussion around the best architectural styles has intensified serverless computing or containerized microservices for building scalable, economical, and high-performance applications. This research investigates the relative merits of these architectures by analyzing the fundamental differences of serverless computing and containerized microservices, reviewing scalability aspects, performance, costs, operational flexibility, and management overhead. The paper will enhance the knowledge of trade-offs between serverless and containerized microservice architectures across different deployment situations through a systematic review of the relevant peer-reviewed and empirical literature to enhance understanding. The research is based on the systematic review of 10 recent comparative studies, with a focus on the more demanding categories of enterprise applications, real-time data streams and scalable web services. The key research contribution from this research is the provision of an architectural selection framework that incorporates context and supports considered choice for cloud-native deployments. The context-aware architectural selection framework is validated through cross-platform studies and workload-specific studies and contributes evidence-based guidance to better sustain the challenges of architectural selection in cloud environments. The research shows that containerized microservices have better performance characteristics for long-running stateful applications that require predictable allocation of resources and fine-grained control, whereas serverless architectures are best for event-driven or arbitrary workloads that can benefit from auto scaling and the potential for lower operational overheads. This research also provides a fuller analysis of hybrid deployment options and recognizes some potential of these options while discussing some debates around hybrid options being the most cost effective when sustained periods of high demand are not occurring. Across the above summary, the critical overarching statement is that there is no single decision-making framework for organizations to follow therefore, decision-making frameworks based on the context of organizational objectives for scalable cloud architecture design are necessary.

Search Keywords: AWS Lambda, Azure Functions, Kubernetes, Google Functions, Function as a Service, Performance Analysis, Cloud Computing, Serverless Architecture, Containerized Microservices.

Introduction

Cloud-native development is booming throughout all the avenues of digital transformation. From enterprise applications to real-time data streaming to large-scale web services, we should not forget the possibility of cloud architectures impacting scalability, efficiency, and performance. Unfortunately, the rush to implement cloud-native technologies also creates a new challenge that threatens all of these advances we simply do not have a clear mechanism for deciding when to use a serverless microservice architecture versus a container-based microservice architecture. As with any architectural decision, cloud-native architectures are multi-dimensional, complex, and impactful on cloud-native development, and can be further complicated by use cost, unpredictable performance, and operational issues, as well. With cloud systems becoming more sophisticated, and their deployment characteristics further complicating the problem, their architectural trade-offs can become indecipherable even for the most experienced application developers.

The importance of architectural selection is highlighted across several domains, particularly in areas of development where performance and costs are paramount; including data-intensive processing applications, many high traffic applications, and mission critical enterprise systems; areas of development where there is a need to predict and optimize all aspects of deployment behavior to develop quickly, and fully understand operational and cost impacts of deployments. The existing literature highlights this problem, and shows that the complexities of modern cloud architectures pose challenges for development, along with the need to speak of architectural design factors with evidence-based selection criteria.

This research arises out of the serious discrepancy between the potential of serverless platforms such as AWS Lambda and containerized microservices and the necessity for clear, evidence-based architectural selection frameworks in cloud applications. The objective is to provide assistance to the developing situation by putting forward a systematic comparison that improves the decision-making process of cloud architecture on increasingly scattered parameters, including but not limited to cost, performance, scalability, and operational complexity. In doing so, it is anticipated that the resulting comparison addresses the needs of the cloud architecture stakeholder as a whole (developers, architects, decision makers), allowing better decision making in cloud-native deployments. In this research, ten relatively new research papers comparing serverless to containerized microservices and highlight the pros and cons of each architecture while suggesting guidance for the most suitable use cases when constructing scalable cloud-native architectures.

Background

Cloud computing has transformed the construction, deployment, and scaling of application techniques and saved more time for the engineers. It enables the acquisition of computing resources on demand with minimal or no initial expenditure. In this research, serverless computing and containerized microservices are two architectural styles that have become the most popular for engineers to deploy services that help to scale AI models and applications on cloud platforms.

Function-as-a-Service (FaaS), also known as serverless computing, handles server management for developers so they can focus on writing code and execute and run on FaaS. AWS Lambda, Azure Functions, and Google Cloud Functions are all examples of platforms that handle setting up infrastructure, scaling automatically, and handling errors. This method works best for workloads that are event-driven, short-lived functions, or services. It also saves money because you only pay for the usages and number of executions or access. But it has problems like cold start latency, a short execution time at the start of execution, and being locked into a vendor.

In **containerized microservices**, the applications/services/models are encapsulated in images and containers that operate uniformly across diverse environments and different operating systems. Orchestration platforms such as Kubernetes handle the containers, scalability, and reliability, providing developers with enhanced control, flexibility, and compatibility for complex, long-running applications/services/models. They also permit the utilization of diverse technology stacks within the same deployment. But it requires additional operational expertise, oversight, and infrastructure management to handle the scalability and reliability and monitor the whole solution.

The increase in the use of cloud platforms by organizations has made the selection between serverless and containerized microservices a critical architectural decision. No comprehensive framework is available to aid practitioners in decision-making suited to their unique situations, although existing research offers significant insights into the trade-offs between performance, scalability, and cost, sometimes the execution time. The lack of guidance requires a systematic comparative analysis to consolidate current knowledge and determine the best practices for implementing scalable cloud solution architectures that serve apps, services, and AI models.

Problem Statement

There is no clear guideline to help architects choose between serverless and containerized microservices for building scalable services on cloud platforms. As the organizations continue to adopt cloud-native solution architectures, architects and technical engineers face the challenge of selecting a cloud-native deployment architecture approach or the best practice solution architecture. Serverless and containerized microservices are two of the most popular deployment paradigms on cloud platforms. The choice between them is often confusing because they share many of the same capabilities, involve performance trade-offs, and have no consensus recommendations on decision-making frameworks.

The main problem is that decision-makers are not provided with yet another decision-making guide to establish sound rationale to help choose the best solution for their service based on workload criteria, scalability requirements, budget, acceptance latency, and engineering time. In addition to that, serverless computing platforms like AWS Lambda allowed systems to scale automatically and decreased operational overhead. However, they have some challenges regarding workloads to reduce cold starts, have execution time limits, and introduce vendor lock (Al-Khatib, 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, and there is a need for engineering skills (Mera Menéndez et al., 2023). Many studies offer comparisons or hybrid architecture approaches, but their findings are limited to platforms. Also, none of the findings offer a consolidated framework for practitioners across different environments, such as edge, multi-cloud, or enterprise-scale applications. This research review aims to analyze recent literature, assess the pros and cons of each deployment solution, and provide a guide to identify the right architectural style based on workload types, scalability needs, and operational limitations and engineering manpower.

Research Objectives

To evaluate and compare serverless and containerized microservices architectures in order to provide actionable, evidence-based guidance for designing cloud-native systems that are scalable and efficient. The research objectives to accomplish this aim are to:

- To conduct a systematic literature review of the selected papers of serverless and containerized microservices with respect to performance, scaling, cost, operational complexity, and workload-specific trade-offs.
- To extract and evaluate the advantages and disadvantages of serverless and containerized deployments across the contexts of their use.

- To recognize and evaluate the main attributes that contribute to architectural choices, such as latency sensitivity, transient bursts of concurrency, cost, and operational complexity.
- To provide a guideline that helps architects and technical engineers with deciding on the best deployment architecture that fits the app requirements based on types of workloads, workload scaling needs, cost, and operational complexity.

Research Scope

The research looks at how serverless and containerized microservices compare to each other in the context of scalable cloud architectures. The research focuses on the following areas:

- There are two types of solution architecture approaches that are focusing on the evaluation of architecture. The first approach is (FaaS) platforms that don't use servers, like AWS Lambda, Azure Functions, and Google Cloud Functions. The second approach is containerized microservices that are running and managed by tools like Kubernetes.
- Scalability, performance, cost, ease of deployment, operational and support complexity, and suitability for different types of workloads are some of the things that are looked at when judging.
- Application Context is all about apps deployed on cloud platforms, both stateless and stateful workloads for microservice architectures, without mentioning or including anything related to monolithic or traditional client-server architectures; it only focuses on microservice services.
- The research source scope includes research articles, conference papers, and reliable technical reports that were published between 2020 and 2025. So the results are based on current cloud computing practices.
- The research isn't limited to a certain area/region or cloud provider.

By keeping this scope, the research makes sure that the analysis is focused, relevant, and up-to-date, which directly helps architects and decision-makers choose the best way to design scalable cloud microservices applications, especially the serverless and containerized microservices that are available in all regions on all cloud platforms.

Literature Review

Giovanni and Charlotte's (2025) 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 is better for quicker development, lowered costs for low-volume workloads, and virtually zero operational overhead. However, containerized deployments are better for sustained load and provide researchers with better control over the runtime environment. The authors, therefore, recommend using a hybrid architecture involving both approaches depending on the workload.

Tripath's (2025) paper 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. The study reported that AWS Lambda functions built with Node.js need between 250 to 500 milliseconds to start after being idle, while Java-based functions experience cold starts that extend beyond one second, and Kubernetes provides sub-millisecond response times for latency-sensitive applications through its pre-warmed deployments. The implementation of Q-learning models trained on invocation patterns has resulted in a 60% reduction of cold start latency according to AI-based systems. Kubernetes pod spin-up times vary between 5–20 seconds depending on image size and node availability, while serverless spawns containers within ~300–500ms for warm functions. Serverless is advantageous if the 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.

In this research, 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 study reported AWS Lambda handled as many as 1000 concurrent users with average latency under 1000 ms after initial cold-start period, while EC2 container maintained latency levels under 1000 ms initially even while handling up to 500 concurrent users before performance deteriorated significantly. The cost analysis indicated that the AWS Lambda functions reduced operational costs for workloads under 447,427 requests per month; however, for sustained workloads over 447,427 monthly requests the dedicated EC2 container was less expensive. The paper concludes that the choice of deployment should depend on the volume of use and how sensitive the performance issues are (Allen, Li, Abdelfattah, Cerny & Taibi, 2024).

In Kodakandla's (2021) paper, it contrasts serverless and containerized architectures on AWS and Azure, using two different cloud-native applications: video encoding and data ingestion pipelines. The authors compared both platforms along the following three dimensions: latency, resource utilization, and developer effort. In terms of applications, they determined that serverless was a better fit for lightweight or event-driven workloads, which have lower latency sensitivity, because serverless is more flexible and cost-effective than these workloads running in containers. 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. The quantitative results indicated that serverless functions had cold start latency ranging from 200-700ms across providers, where containerized workloads were not directly compared in this study. Serverless platforms can handle up to 5,000+ concurrent requests per second (Azure Functions), with AWS Lambda supporting 3,000+ and Google Functions 2,500+, while containers were not benchmarked in this comparative analysis. Serverless deployments cost \$0.17-\$0.25 per million requests depending on the provider, with Google Cloud Functions being the most cost-effective at \$0.17 for low-

volume workloads. The authors 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 consider a container, and for flexibility, customers should consider serverless.

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 architecture could be initiated easily. Quantitative results indicated that microservices were able to process IoT events at an average latency of 100 ms and CPU utilization around 20% in low-demand scenarios, while serverless functions (Flask) showed similar latency of 100 ms but with additional overhead from the OpenFaaS API Gateway. In high-demand scenarios with auto-scaling, Flask functions achieved a minimum latency of 150 ms while static microservices were nearly 10 times slower. Classic serverless functions showed much higher latency (1200 ms average) and CPU utilization up to 95%. Resource utilization per processed event was generally lower for microservices compared to serverless functions. 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 (*Tusa et al., 2024*).

Menéndez, Gayo, Canal & Fernández (2023) implemented nine microservices on 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 authors 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. Quantitative results indicated that AWS EKS with DocumentDB achieved the best performance with 61 ms average latency and 0% error rate at 10,000 users, with monthly costs of €157.14. AWS Lambda with DynamoDB was the most cost-effective at €19.23 per month but had higher latency of 219-5807 ms depending on load, while AWS EKS without persistence had latency around 161-367 ms but needed provisioned resources with monthly costs of €92. This paper is used to establish the importance of the application and the patterns of workload to help decide which deployment architectures to use and how much operational overhead is acceptable, advocating for hybrid or mixed strategies.

Falope & Oladoja, in 2025 provide qualitative analysis indicating that serverless functions excel in event-driven workloads due to their low startup latency but may experience latency issues during cold starts, while containerized deployments typically offer more consistent performance for long-running tasks and high-throughput applications. Cost analysis suggests that serverless is cost-effective for applications with unpredictable or low traffic since users pay only for execution time, while containers can be more economical for consistent, long-running workloads through efficient resource pooling and utilization.

In *Wen, Chen, Jin & Liu (2022)* systematic literature review, the authors look at more than 160 papers from 17 different research areas, such as optimizing performance, moving applications, deploying in multiple clouds, and testing in serverless architectures. It looks at the most common trends, opportunities, and problems in serverless computing as it is used in business and academia. The reported numerical outcomes include that the survey analyzed 164 research papers across 17 research directions, with performance being the largest research area, accounting for 40.85% (67/164) of all papers, programming framework as the second largest at 19.51% (32/164), and resource management as the third largest, at 13.41% (22/164). Within performance studies, 89.55% (60/67) focused on performance optimization, with cold start performance representing 56.67% (34/60) of performance optimization efforts.

The *Nadeem, Farhan, Qadir & Minhas in (2024)* analyze a web application developed in a monolith, microservice, and serverless architecture (using Lambda and Fargate) under various traffic circumstances. They used JMeter to evaluate response times and costs under low, medium, and high traffic conditions. They found that in low-load conditions, Lambda performed the best for both cost and latency. Nevertheless, Fargate experienced substantially elevated expenses, particularly during batch processing and peak load, and its cost benefit waned as traffic escalated. Numerical outcomes reported suggest that under low traffic Lambda demonstrated superior performance with response times approximately 3 times better than other architectures and provided cost savings of 35.071% compared to Monolith, 32.092% compared to Microservice, and 365.18% compared to Fargate architectures. Under high traffic, Lambda maintained the best average response time performance, while Fargate costs escalated dramatically increasing up to 2520.58% compared to base cost when running batch jobs, with monthly costs reaching over \$22,000 compared to Lambda's range of \$200-1,200 depending on batch job intensity.

Chen, Zhao, Li, Wang & Zhou (2025) work on the problem of unpredictable billing in serverless edge environments, especially for IoT and wireless networks. They suggested two online reinforcement learning-based scheduling algorithms to help find the best placement for functions while staying within budget. Simulations reported achieving near-optimal solutions within 3% of an ILP solver and being 5 orders of magnitude faster in decision-making time and cost savings of up to 28% compared to standard serverless scheduling methods, also with constrained budgets.

Critical Analysis

The reviewed literature contains useful knowledge concerning serverless and containerized microservices in cloud-native architectures, with emphasis on scalability, performance, and cost trade-offs. Across the studies there is a recommendation for some hybrid architecture to harness the strengths of both containerized and serverless architectures, but with workloads that are bursty or measures of latencies. What is getting the most research interest, especially in the last year, is the emergence of new trends, for example, AI-driven orchestration and edge computing, which are very much relevant with respect to cloud-native solutions.

In some ways the literature shows its strengths, but it also has some limitations. Many of the studies rely on precise platforms (e.g., AWS Lambda, Google Cloud Run, etc.), which typically oversimplify simulation, which needs to be exercised in real conditions. The cold-start latency in serverless is an inherent challenge, and the cost-effectiveness findings in the various studies

depend largely on the workload type and scale of workload being considered. Empirical implementations of hybrid architectures are sparse, meaning limited practical insight for implementation.

With respect to methodology, the studies use various experimental evaluations, literature synthesis, and some simulations, which have led to conflicting results sometimes, including performance, scaling, and cost. Overall, these differences suggest that architecture has to be set in context and needs to rely less on a range of presumptions from the previous research regarding workload type, complexity of operations, and most importantly, cost sensitivities. In summary, the studies provide a good basis for understanding serverless and containerized microservices but make it clear that further research is required, especially evaluating hybrid architectures with real-world workloads across multiple platforms. The studies support the current research focus on a synthesis of the literature to provide the basis for guided decisions in designing a scalable cloud architecture.

Connections

1. Shared Recognition of Trade-offs

All ten papers agree that serverless and containers have different strengths. Papers 1, 2, and 6 state explicitly that there is no one size that can fit all; the deployment architecture must match the workload. This consensus is the basis across this body of literature: serverless is for simplistic and burst traffic; containers are for complex, stateful, or high-throughput systems. Paper 9 supports this point by indicating that the decision is very much workload-dependent, especially in multi-cloud situations where hybrid decisions are very often imposed through interoperability requirements.

2. Recommendation of Hybrid Architecture

Papers 1, 2, and 6 recommend hybrid configurations where serverless is used for event-driven or less critical workloads and containers are used for performance-critical services. This connection shows an industry trend of adopting a polyglot architecture and blending deployment architectures with the aim of optimizing flexibility and minimizing costs. Paper 7 provides some empirical support for hybrid strategies by showing increased reliability in critical infrastructures. Paper 10 further supports this idea, by advancing the case for AI-based orchestration mechanisms for hybrid serverless-container deployments.

3. Cold Start as a Universal Bottleneck

Papers 2, 3, 4, and 5 all highlight cold start latency as a common issue in serverless talks. Paper 4 advocated and indicated the extent of this with suggested latency thresholds (e.g., don't use serverless if the need is for a consistent sub-200 ms response time), which does lend some fidelity to the qualitative concerns raised in the others. Paper 8 extends this by reinforcing that latency remains the biggest hurdle for the bar of real-time analytics and high-frequency trading workloads, showing the inherent performance limitations of serverless in latency-sensitive use cases.

4. Scalability is a typical strength of serverless.

Papers 1, 2, and 3 consistently suggest that auto-scaling is a common strength of serverless, particularly for workloads that are less regular or planned occurrences. They are the same in use case and their capacity, from the most basic APIs (for example, Paper 3) to a cloud-native pipeline (example, Paper 4). Paper 7 provides quantitative data that corroborate that serverless scales more efficiently than containerized systems do for unpredictable traffic patterns, and Paper 9 also explains additional potential scalability advantages of serverless in federated multi-cloud deployments.

5. Cost Sensitivity Findings

Papers 1, 3, and 6 offer specific cost comparisons and reach conclusions that serverless tends to be cheaper at low-to-medium workloads but tends to be cheaper than containers at scale. One point made in paper 6 was the difficulty running Fargate at a cost between serverless and containers, and it reminded the audience that cost modelling is part of any architecture decision-making, the same with any modelling. Paper 8 contributes by pointing out that cost inefficiencies in serverless are readily apparent in longer-running or more resource-heavy tasks. Paper 10 contributes by providing predictive cost optimization techniques with the inclusion of AI. They discuss ongoing work to optimize hybrid deployments.

6. Developer Experience / Maintenance

Papers 1, 2 & 4 identified that it is easier to develop and maintain serverless-like infrastructures. There is a connection here too, since ultimately if there is less out of DevOps, there will be an improvement in managing DevOps, but it was also recognized in paper 6 that containers provide the same flexibility for complex systems. According to Paper 7, serverless increases simplicity in development but has operational blind spots around how to monitor an application, whereas Paper 9 points out that "containers improve the debugging and observability capabilities." Paper 10 presents a point of balance, recognizing that AI-driven orchestration includes trade-offs, including how they might lessen the effective DevOps gap of containers and serverless applications.

Conflicts Between Research Findings

1. Serverless Performance Reliability

Papers 1 and 2 find that serverless is performing well across a range of loads with acceptable latency and auto-scaling. Papers 3 and 4 argue that cold-start latency and concurrency limits in serverless architectures can add considerable latencies for API-driven or real-time systems. Paper 7 notes that while latency can be minimized in serverless with provisioned concurrency, this compounds the cost overhead. Paper 8, however, discusses how new platforms like Cloudflare Workers greatly reduce the severity of cold-start challenges, among other considerations, therefore making serverless more feasible in latency-sensitive use cases.

Conflict: serverless appropriate for latency-sensitive applications? Some papers say yes (if managed), while others say no.

2. Best Deployment for Cost Efficiency

Papers 1 and 3 are very clear that serverless is the most cost-effective solution for intermittent workloads. On the other hand, Paper 6 argues that AWS Fargate (a managed container service) has an equal or better cost-performance balance in many situations, especially while considering cold starts and idle latency. Paper 9 also takes the opposite point of view. They find that long-running workloads in serverless are quickly cost disadvantageous relative to containers. Paper 10 goes on to find that multi-cloud cost arbitrage possibilities with containers can sometimes out-do serverless pricing models.

Conflict: Some papers found serverless to clearly be cheaper, while other studies show that managed containers may present the best total cost.

3. Scale Advantage

In both Paper 2 and Paper 3, we see that serverless has the advantage of scaling on demand, without the need to intervene manually. On the other hand, Paper 4 shows that Kubernetes containers also provide strong horizontal scaling and more predictability once workloads are sustained. Paper 7 supports Paper 4, indicating that Kubernetes scaling is more predictable for steady-state applications, while Paper 8 demonstrates that serverless has greater elasticity when reacting to unpredictable spikes in traffic.

Conflict: There are competing views as to whether one can scale reliably at high volume or not.

4. Preferred Architecting Model

Paper 4, however, prefers strict use-case separation: containers for anything latency-sensitive and persistent, and serverless for light, burstable work. In contrast, Papers 1 and 6 favor hybrid forms, stating that many applications can and should mix both forms. Paper 9 reiterates the hybrid approach and suggests that enterprise workloads can benefit from a base load in a container and burst using serverless. Paper 10 takes another spin by proposing a "container-first" approach that uses serverless as an adjunct rather than a core architecture.

Conflict: Some papers prefer definitional separation of use cases, and some are advocating for a blended approach.

Synthesis Matrix for Literature Review

Study	Authors	Year	Key Findings/Results	Methodology	Limitations	Strengths
Serverless Computing vs. Containerization: Comparative Analysis for Cloud-Native Application	-Alessandro Matteo Giovanni -Luna James Charlotte	2025	Serverless was faster to make, cost less when the use was low or changing, and was easier to run. Containers worked better when there	Made a simple microservice app two ways: One as an AWS Lambda function (serverless). The other, with containers run by	It was just one app on one cloud (AWS). No, look at the long run (weeks or months) to see how autoscaling	A good way of doing the benchmark of the same app on two types of systems.

Deployment			was a lot of use, and they were good where the goal is to keep things going for a long time or keep data.	Kubernetes. Measured how long it took to start up after a cold start, how well they scaled, how hard they were to run, and how much they cost at different levels of use.	works over time. No look at how to fix the cold start issue found in serverless (provisioned concurrency in Lambda).	Good description of the metrics used to measure latency, cost per call, and how much work has to be done to run the app. Good talk about when to use either (containers or serverless) depending on what kind of app it is.
Engineering Scalable Microservices: A Comparative Study of Serverless Vs. Kubernetes-Based Architectures	- Sreenivasulu Navulipuri	2025	Cold starts in serverless functions were notable under sporadic access but less of an issue under sustained traffic. Kubernetes had better predictable performance and was preferred for heavy/consistent workloads. Serverless was significantly cheaper for low-	Built identical microservice applications (e.g., file conversion, user management) in: AWS Lambda (FaaS) Docker containers deployed on Kubernetes (EKS) Simulated varied workloads using Apache JMeter.	Only on AWS services (Lambda and EKS); might not work for GCP or Azure. No look at the impact on CI/CD pipelines or the overhead of DevOps on both setups. No user experience metrics or perf metrics for the integrations	Lays out a multi-metric benchmarking setup with repeatable workloads. Gives a cost vs performance picture based on real cloud pricing.

			invocation workloads. Proposed a hybrid architecture where serverless handles burst traffic and containers manage core services.	Benchmarked metrics: startup time, CPU/memory usage, response latency, scaling behavior, and monthly costs.	(e.g., API Gateway latency).	Brings up a workload classification idea to decide between deployment architectures.
Comparing Cost and Performance of Microservices and Serverless in AWS: EC2 vs Lambda	-Christopher Allen -Xiaozhou Li -Tom Černý -Amr S. Abdelfattah	2024	Lambda was 30-60% cheaper at low/med traffic was easier to run and grow EC2 based containers had better throughput at high weight allowing more control but cost more in care. Lambda had cold start issues and burst loads. Recommended using Lambda for event microservices and use EC2 for core work	Deployed a CRUD based REST API as: -Containerized service on EC2 (Docker) and -Stateless functions on AWS Lambda. Run different workloads on Locust: low, med, high. Monitor CPU and RAM use, response times and the monthly bill from AWS billing.	No coverage of Kubernetes or orchestration layers, only EC2 + Docker. Didn't evaluate deployment pipelines, autoscaling logic, or DevOps complexity. Cold start mitigation options in Lambda not explored	Use of real billing data from AWS Variety of load scenarios; show where problems occur. Can be utilized to analyze how the cloud system will respond to real time work load. Can be used to obtain the impact of the difference in the work load on the cloud.

Serverless Architectures : A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications	-Naveen Kodakandla	2021	Serverless offered versus what Containers offered: Serverless: -Less developer effort -Cost-efficiency with intermittent traffic -Auto-scaling with very little config Containers: -Better performance for stateful/compute-heavy workloads -More control over resources -More predictable latency He also established a latency threshold rule: serverless is an option for workloads with a latency tolerance of less than ~200ms. If the workload is at greater than 200ms latency, the selection will be containers.	Compared implementations of video encoding service and event-driven data pipeline in: AWS Lambda & Azure Functions (serverless architecture) Kubernetes deployed containers Benchmarks: Cold starts, latency, resource usage, developer complexity, and scaling limits	Consider both AWS and Azure provides multi-cloud view. Realistic test cases encoding and data processing are common workloads in the cloud.	Not a direct comparison with containers on EC2 or managed container services. Did not consider hybrid use cases that other papers considered.
Microservices and serverless functions—lifecycle, performance, and resource utilisation of	-Francesco Tusa -Stuart Clayman -Alina Buzachis		Containers experienced: - Lower latency (approx. a 25% improvement) and	Developed a sensor analytics pipeline of air quality and motion sensors using: -Containerized	Restricted to sensor analytics generalization to other workloads (e.g., video, AI inference) not	Relevant to real-world IoT workflows , applicable to Smart Waste or

edge based real-time IoT analytics	-Maria Fazio		better resource utilization. - Increased offline resilience when edge connectivity was minimal. Serverless experienced: - Ease of deployment/scalability in an IoT, cloud-based context. - Less suitable for real-time edge analytics due to longer start times. Containers are ideal for edge analytics, serverless is ideal for cloud-based aggregation and alerting.	microservices on edge nodes. -Serverless functions pushed by cloud messaging services Monitored cold starts, delays in data transmission, compute cycles, energy usage, and resource constraints.	promised. Didn't investigate hybrid orchestration architectures (e.g., KubeEdge, AWS). Some data was simulated and not from long-term production environments.	Smart City rollouts. Metrics measured were over performance such as energy and memory usage. Provides guidance on designing hybrid cloud/edge workloads.
A comparison between traditional and Serverless technologies in a microservices setting	-Juan Mera Menéndez -Jose Emilio Labra Gayo -Enrique Riesgo Canal -Aitor Echevarría Fernández	2023	They developed nine microservices prototypes deployed in all three environments. They benchmarked: - Cold start time - Memory and CPU usage report - Execution latency - Cost across 30-day cycle They also considered infrastructure management complexity.	- Lambda: Great for infrequent and burst workloads, minimal management overhead, but had high cold starts for API service. - Fargate: Cost-effective, less management than traditional Kubernetes(container runtime),	Does not consider hybrid architectures or transitions (e.g., Lambda to Fargate). Only AWS services, without GCP/Azure equivalent comparisons. Benchmarks did not include metrics on failover, high availability, or observability.	Contains AWS Fargate, usually excluded in serverless/container blendings. All real-world prototype apps provide realistic benchmarks database application

				consistent performance, no cold start. - Kubernetes: Best fit for consistent workloads and high throughput, however the highest amount of management overhead. Recommendations are: - Use Lambda for event driven jobs - Use Fargate when there is a need to balance auto-scaling and cost - Use Kubernetes when complex systems exist and there is a management control		ns. Contains metrics related to developer experience and operational complexity.
A Comparative Analysis of Serverless and Containerized Architectures in Cloud-Native	Falope Samson & Oladoja Timilehin	2025	Event-driven workloads are best served by serverless; stateful, lengthy, and intricate microservice chains are better served by containers. This	Review of the literature plus experimental comparisons with FaaS and Docker/Kubernetes services	Limited experimental depth and a heavy reliance on secondary data	Covers a range of workload scenarios and offers insightful qualitative information.

Environment s			emphasizes cost-performance trade-offs.			
Rise of the Planet of Serverless Computing: A Systematic Review	Jinfeng Wen, Zhenpeng Chen, Xin Jin & Xuanzhe Liu	2022	Identifies 17 areas for serverless computing research; trends include migration, multi-cloud adoption, optimization, and tooling and interoperability gaps	Systematic review of more than 160 studies in the literature	High-level analysis without a direct comparison to containerization	Thorough coverage that identifies new research gaps
Cost and Performance Analysis of Monolithic, Microservice, and Serverless Architectures	Nadeem et al.	2024	In terms of cost and latency, Lambda performs better than the others under low load; container-based Fargate is more costly under high load.	Web application deployment in serverless, microservice, and monolithic forms is empirically tested; JMeter is used to simulate traffic.	One workload type and just two serverless/container services were tested.	Several architectural styles are covered by comparative design, and there are definite cost-performance trends.
Cost Optimization for Serverless Edge Computing with Budget Constraints using Deep Reinforcement Learning	Chen et al.	2025	Reinforcement learning-based schedulers are five orders of magnitude faster than ILP and produce deployment decisions that are nearly optimal within budgetary constraints.	Assessment of AI-based serverless edge scheduling algorithms through simulation	Edge computing-specific; no container comparison	Creative AI methodology; extremely effective budget-conscious decision-making

Research Methodology

This research is being conducted using a systematic literature review approach. The work involved searching for, gathering, and critically evaluating academic papers, literature, and case studies of serverless computing, containerization, and cloud-native application deployment's use of these technologies. The selection process was based on timeliness, relevance to scalability, performance, and cost while considering the credibility of the literature.

The literature was organized and read for the purpose of tracing patterns, findings, strengths, and limitations across studies. Thematic analysis sorts to understand common trends and identify gaps in current literature. In this way, the literature review provides a synthesis of the state of the topic in question that does not require experimentation or empirical analysis.

Phase 1: Systematic Literature Search

The first phase is an extensive literature search to discover and collect existing research on comparisons between serverless computing and containerized microservices. This search includes finding scholarly articles, conference proceedings, industry reports, and other technical documentation. The objective of this search is to produce a map of the current landscape of cloud architecture comparisons and to identify gaps in knowledge and practice that the proposed framework is attempting to bridge.

Data Sources: Scholarly repositories and networks including ResearchGate, arXiv, ScienceDirect, Future Generation Computer Systems, International Journal of Science Research in Engineering and Technology and Journal of Information Systems Research and Computing.

Selection Criteria: publications in English language, publications between 2020 and 2025, publications comparing serverless and containerized architectures; and publish on scalability, performance, cost analysis and operational complexity.

Phase 2: Literature Review and Quality Assessment

The second phase consists of undertaking a systematic review of included literature with goal of assessing both the architectures and relevance of studies. The goal is to apply critical analysis to determine the trustworthiness and transferability of results across studies.

Inclusion criteria: peer-reviewed journal articles, conference papers, and technical reports which involve empirical comparisons of serverless and containerized architectures and their microservices implementations.

Exclusion criteria: articles which focus only on monolithic architecture, theoretical papers that do not offer empirical comparisons, and articles published prior to 2020.

Quality evaluation: studies will be evaluated based on the grading of study design method, clarity of method, adequacy of participant sample size and reproducibility of results. It was determined a total of 10 studies were included based on being directly relevant to comparisons of serverless vs containerized microservices architectures: Giovanni & Charlotte (2025), Tripathi (2025), Allen et al. (2024), Kodakandla (2021), Tusa et al. (2024), Menéndez et al. (2023), Falope & Oladoja (2025), Wen et al. (2022), Nadeem et al. (2024), Chen et al. (2025).

Analysis: a qualitative content analysis will be used to identify themes associated with performance characteristics, cost models, scalability patterns, operational requirements,

and the specific findings of comparisons of AWS Lambda vs EC2, edge computing study, and real time IoT analytics implementations.

Phase 3: Comparative Analysis and Data Synthesis

This stage entails the methodical comparison and synthesis of findings across the selected studies, establishing areas of agreement and disagreement related to architectural performance and fitness.

Data Extraction: The main data extracted was performance indicators, cost data, scalability indicators, operational complexity metrics, and recommendations for use cases from each paper. In particular, we extracted data from Allen et al. (2024) related to AWS Lambda vs EC2 performance comparisons, case studies on edge computing (Tusa et al., 2024), cost optimization strategies (Chen et al., 2025), and systematic reviews related to the evolution of serverless computing (Wen et al., 2022).

Analysis Method: Cross-study comparison matrices were created to understand similarities and differences in findings across different platforms, workloads, and application contexts, including IoT analytics, microservices deployments, and cloud-native environments.

Statistical Synthesis: Where feasible, we used to quantify findings, particularly from cost and performance studies, to aggregate and identify the distributions of performance differences and the cost implications between serverless (AWS Lambda) and containerized (Kubernetes/EC2) architectures.

Phase 4: Framework Development & Validation

Based on the insights derived from the previous phases, a framework for making decisions about when to select serverless and containerized microservices will be developed. This framework will help inform decision makers and architects with contextually based selection criteria.

Development: The identified performance characteristics, cost models, scalability patterns, and operational requirements will be integrated into a unified selection framework.

Framework Components: The framework will incorporate workload type, organizational constraints, performance requirements, and cost optimization considerations.

Validation Method: The framework will be assessed for clarity, applicability, comprehensiveness, and usability for real life architectural decision making options.

Limitations

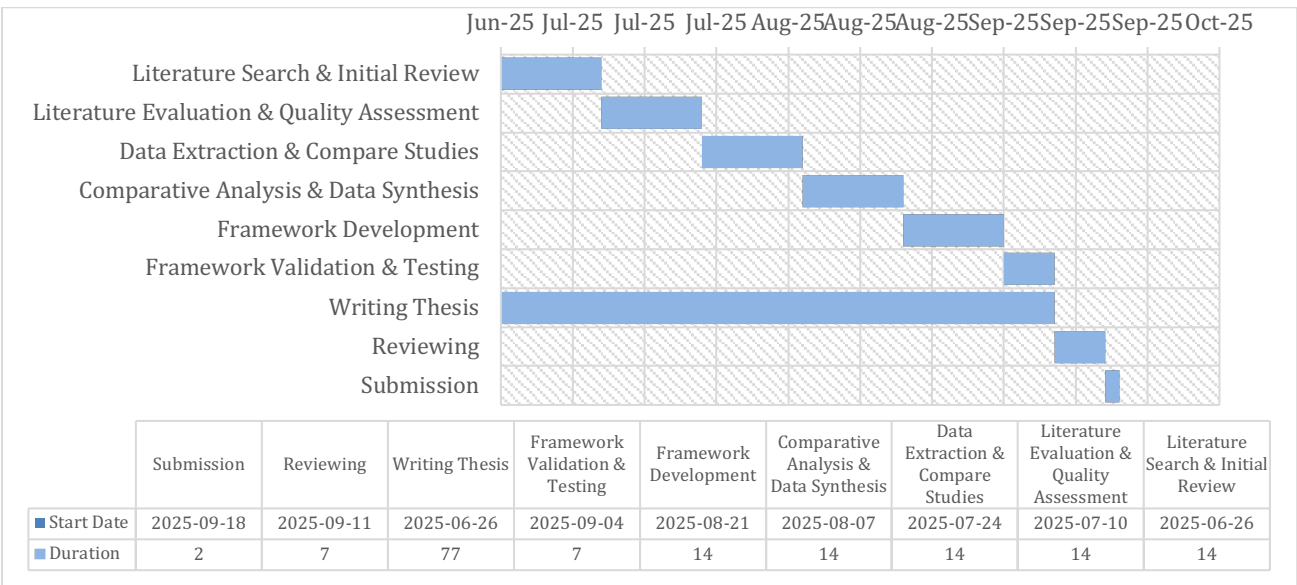
While this research methodology acknowledges possible constraints, there may be also bias toward published results, with the possibility of variability between experimental conditions, time (the speed technology is changing could make some of the results somewhat outdated), and the representativeness of studies with regard to cloud platforms and use cases, all of which are to be expected. Study selection and quality assessment, as well as open reporting differences in results, helped manage these limitations.

Expected Outcome

This study will enable a comprehensive comparison of serverless computing and the use of containers, specifically in regard to deploying cloud-native applications. The study will provide

clear insights into the operational complexity, cost, scalability, performance, and deployment efficiency of each approach through systematic analysis of existing research findings and comparative studies. The purpose of the research is to produce advocacy guidelines on how to choose the best architecture based on workload characteristics and business requirements, synthesized from empirical findings across multiple studies and documented case studies from the selected literature. The research will consolidate performance metrics, cost analyses, and operational experiences reported across the 10 selected studies to establish evidence-based selection criteria. The findings should support academic research as well as real-world cloud-native system design choices by establishing a framework for businesses that want to harmonize cost, performance, and operational agility based on validated patterns and trends identified in current literature and documented implementation experiences.

Gantt Chart



Conclusion

This literature review has assessed the advantages, disadvantages, and compromises of serverless computing and containerized microservices when constructing scalable cloud architectures. The ten reviewed papers including Giovanni & Charlotte (2025), Allen et al. (2024), and Tripathi (2025), shared a general consensus that these two architectures 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. Based on the systematic analysis of these studies, this research has developed a context-aware architectural selection framework that guides decision-making based on workload characteristics, predicted scalability needs, operational objectives, and available infrastructure. The general conclusion is simple: the architecture 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. The developed framework incorporates these hybrid deployment strategies while addressing the specific findings from cost analyses (Allen et al., 2024), edge computing implementations (Tusa et al., 2024), and systematic performance evaluations across the reviewed literature. As cloud platforms continue to develop, they should continue providing new services such as AWS Fargate, Azure Container Apps, and edge-native runtimes. Also, more research is needed on the transition paths between containerized and serverless architectures, 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
<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
- 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
- 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.
<https://arxiv.org/abs/2305.13933>
- Falope, Samson & Oladoja, Timilehin. (2025). A Comparative Analysis of Serverless and Containerized Architectures in Cloud-Native Environments.

https://www.researchgate.net/publication/388421522_A_Comparative_Analysis_of_Serverless_and_Containerized_Architectures_in_Cloud-Native_Environments

Wen, Jinfeng & Chen, Zhenpeng & Jin, Xin & Liu, Xuanzhe. (2022). Rise of the Planet of Serverless Computing: A Systematic Review.

<https://arxiv.org/abs/2206.12275>

Nadeem, Muhammad & Farhan, M. & Qadir, Muhammad Azeem & Minhas, Nasir M. (2024). Cost and Performance Analysis of Monolithic, Microservice, and Serverless Architectures.

<https://jisrc.szabist.edu.pk/ojs/index.php/jisrc/article/view/218>

Chen, X. & Zhao, Q. & Li, Y. & Wang, J. & Zhou, M. (2025). Cost Optimization for Serverless Edge Computing with Budget Constraints using Deep Reinforcement Learning.

<https://arxiv.org/abs/2501.12783>