

Serverless web application using modern cloud architecture: A Conceptual Review of Transformation, Dependency, and Misuse

Sanika Jadhav¹; Vijay Sapkal²; Dr. Swati Joshi³.

^{1&2} Student, PVG's College of Science and Commerce, Pune, Maharashtra, India.

³ Research Guide, PVG's College of Science and Commerce, Pune, Maharashtra, India.

Email: jadhavsanika514@gmail.com¹, vijusapkal2200@gmail.com²

ABSTRACT

Serverless architecture has emerged as a transformative paradigm in cloud computing, providing a cost-effective, scalable, and high-performance solution for modern cloud-native applications. This study presents a comprehensive comparative analysis of serverless computing across three fundamental dimensions: performance, scalability, and cost efficiency. The research critically evaluates the services offered by major cloud providers, including AWS Lambda, Azure Functions, and Google Cloud Functions, to identify differences in their operational characteristics, pricing models, and real-world applicability.

The paper begins with an overview of serverless computing, outlining its core principles, advantages, and inherent limitations. Furthermore, it examines key performance metrics such as latency, cold start behavior, and concurrency management to assess the suitability of serverless platforms for diverse application workloads. Scalability is analyzed through the evaluation of auto-scaling mechanisms under varying traffic conditions, demonstrating the ability of these platforms to handle dynamic demand efficiently.

Visual representations, including tables and graphs, are utilized to present comparative data effectively and support informed decision-making. The paper concludes by discussing best practices for adopting serverless architectures, along with strategies for optimizing performance and minimizing costs. It also explores emerging trends, including cold start optimization and the integration of serverless computing with edge technologies. The findings of this research aim to assist organizations in effectively leveraging serverless architectures to achieve enhanced operational efficiency within cloud-native environments.

1. INTRODUCTION

In recent years, the widespread adoption of cloud computing has significantly transformed the design, development, and deployment of modern applications. Among the most notable advancements in this domain is serverless architecture, a paradigm that abstracts infrastructure management and enables developers to concentrate exclusively on application logic. This model has gained considerable attention due to its ability to simplify operational processes, enhance scalability, and optimize cost efficiency without requiring manual provisioning of resources.

The term “serverless” may appear misleading, as underlying server infrastructure continues to exist; however, its management and maintenance are entirely handled by cloud service providers. This paradigm represents an evolutionary progression from traditional on-premises systems to Infrastructure-as-a-Service (IaaS), followed by Platform-as-a-Service (PaaS), and ultimately to Function-as-a-Service (FaaS), which forms the foundation of serverless computing. Leading cloud platforms such as Amazon Web Services (AWS Lambda), Microsoft

Azure (Azure Functions), and Google Cloud (Google Cloud Functions) exemplify this approach by enabling event-driven execution of discrete functions without the need for dedicated server management.

Modern applications are characterized by increasing demands for high performance, seamless scalability, and cost-effectiveness. For example, e-commerce platforms must efficiently manage unpredictable traffic surges during promotional events, while Internet of Things (IoT) systems require real-time, event-driven data processing. Traditional computing architectures often face limitations in addressing these requirements due to challenges such as resource over-provisioning, underutilization, and manual scaling complexities.

2. MOTIVATION FOR RESEARCH

The motivation behind this research arises from the growing demand for highly scalable and cost-efficient computing solutions in modern applications. Traditional cloud models often face challenges such as resource over-provisioning, underutilization, and complex infrastructure management. Serverless computing addresses these limitations by offering automatic scaling and a pay-as-you-go cost model. Therefore, this research is conducted to compare the performance, scalability, and cost of serverless architectures in order to evaluate their effectiveness and suitability for different application scenarios.

Serverless architecture is a cloud computing execution model in which the cloud provider dynamically manages the allocation, provisioning, and maintenance of server resources. This paradigm enables developers to focus exclusively on application development while abstracting the complexities associated with infrastructure management, scalability, and system maintenance. Although the term “serverless” implies the absence of servers, the underlying infrastructure is still present; however, its operation is fully managed by the cloud service provider..

The primary objective of this research is to provide a comprehensive analysis of serverless architecture by examining its fundamental principles, operational characteristics, and practical applications. The study aims to evaluate the effectiveness of serverless computing in addressing modern application requirements, particularly in terms of performance optimization, scalability, and cost efficiency. Furthermore, it seeks to compare leading serverless platforms to identify their strengths, limitations, and suitability across various use cases. By doing so, the research intends to offer valuable insights that assist developers, system architects, and organizations in making informed decisions regarding the adoption and implementation of serverless computing solutions.

3. RESEARCH AIM AND OBJECTIVES

1. The primary aim of this research is to evaluate serverless computing platforms and understand their effectiveness in modern cloud computing environments. Serverless architecture has gained popularity due to its ability to provide scalable, flexible, and cost-efficient solutions. Therefore, this study focuses on analyzing how well these platforms perform in real-world scenarios and whether they meet the growing demands of modern applications.
2. The research specifically aims to compare leading serverless platforms such as Amazon Web Services (AWS Lambda), Microsoft Azure (Azure Functions), and Google Cloud (Google Cloud Functions). These platforms are widely used in industry and offer different features, pricing models, and performance capabilities. By evaluating these platforms, the study seeks to identify their strengths and limitations.
3. One of the key objectives of this research is to study the fundamental concepts of serverless computing. This includes understanding how serverless architecture works, its core principles such as event-driven execution, automatic scaling, and statelessness, and how it differs from traditional cloud computing models. This helps in building a strong theoretical foundation for further analysis.

4. Another important objective is to compare different serverless platforms based on their features and functionalities. Each cloud provider offers unique services, tools, and integration capabilities. By comparing these aspects, the research provides insights into which platform is more suitable for specific types of applications.

1. 4. LITERATURE REVIEW

Several studies have focused on comparing microservices architecture with serverless computing. Microservices involve breaking applications into smaller services that run continuously, whereas serverless executes functions only when triggered. Researchers have found that serverless reduces operational overhead since developers do not need to manage infrastructure. However, microservices provide better control over long-running processes and system architecture. Studies suggest that serverless is more suitable for event-driven and short-duration tasks, while microservices are preferred for complex, continuously running applications. This comparison indicates that the choice depends on workload type and system requirements.

Another important area of research is cold start latency in serverless platforms. Cold start refers to the delay that occurs when a function is executed after being idle for a period of time. Researchers have identified this as a major limitation affecting performance, especially in real-time applications such as APIs and IoT systems. Various studies have proposed solutions like pre-warming functions, optimizing code size, and using lightweight runtimes to reduce cold start delays. Cloud providers have also introduced improvements to minimize this issue, but it still remains a challenge in latency-sensitive applications.

Research has also been conducted on performance modeling of serverless platforms. These studies evaluate metrics such as execution time, response latency, throughput, and concurrency handling. Findings indicate that serverless platforms perform efficiently under variable workloads due to automatic scaling capabilities. However, performance can vary depending on factors like memory allocation, function configuration, and workload patterns. Some models use simulation and benchmarking techniques to predict system performance under different conditions. These studies help in understanding how serverless systems behave in real-world scenarios and guide developers in optimizing their applications.

In addition to these areas, several researchers have explored cost analysis in serverless computing. They found that serverless is highly cost-effective for applications with unpredictable or low usage, as users only pay for actual execution time. However, hidden costs such as data transfer, API calls, and monitoring services can increase overall expenses if not managed properly. This highlights the importance of proper cost planning and monitoring.

Another focus of literature is on scalability and resource management. Serverless platforms automatically scale based on incoming requests, which is a major advantage over traditional systems. Studies show that this feature allows applications to handle sudden traffic spikes efficiently. However, some limitations exist, such as concurrency limits imposed by providers and delays during rapid scaling.

Researchers have also examined challenges such as vendor lock-in and debugging complexity. Vendor lock-in occurs when applications become highly dependent on a specific cloud provider's tools and services, making migration difficult. Additionally, debugging serverless applications is complex due to their distributed and event-driven nature.

Overall, the literature indicates that serverless computing offers significant advantages in terms of scalability, cost efficiency, and ease of use. However, challenges like cold start latency, performance variability, and vendor dependency still need further research. These gaps highlight the need for comparative analysis and optimization strategies, which form the basis of this study.

4.2 Performance and Scalability Evaluation

Performance and scalability are critical factors in evaluating serverless web applications within modern cloud architecture. Performance refers to how efficiently the system responds to user requests, while scalability determines the system's ability to handle increasing workloads without performance degradation.

In serverless web applications, performance is typically measured using key metrics such as latency, execution time, throughput, and cold start behavior. Latency represents the response time between a user request and system output, which is crucial for user experience. Execution time measures how long a function takes to complete its task, directly affecting system efficiency. Throughput indicates the number of requests the system can process within a given time, reflecting its processing capability. Cold start latency, a unique challenge in serverless computing, occurs when a function is invoked after a period of inactivity, leading to delays due to initialization of resources.

In addition, the study aims to identify the best use cases for serverless computing. By analyzing performance, scalability, and cost together, the research helps determine in which scenarios serverless architecture is most beneficial.

Overall, the aim and objectives of this research are designed to provide a comprehensive understanding of serverless computing platforms. The findings of this study will help developers, researchers, and organizations make informed decisions about adopting serverless architecture for their applications.

5. RESEARCH DESIGN AND METHODOLOGICAL APPROACH

5.1 Research Design

This study adopts a comparative analysis methodology to evaluate different serverless computing platforms. The approach combines both qualitative and quantitative analysis to provide a comprehensive understanding of system performance and usability. Qualitative analysis focuses on non-numerical aspects such as platform features, ease of use, developer experience, and integration capabilities with other cloud services. It helps in understanding how user-friendly and flexible each platform is in real-world scenarios. On the other hand, quantitative analysis involves measurable parameters such as latency, execution time, cost per request, and scalability in terms of concurrent executions. These metrics are essential for evaluating the technical efficiency and performance of serverless systems.

5.2 Literature Selection Criteria

Literature included in this review was selected according to the following criteria:

- Peer-reviewed journal articles published with a verified DOI
- Publications focused on education, educational technology, or AI in academic contexts
- Studies discussing generative AI, ChatGPT, large language models, or digital learning tools
- Research addressing student learning behaviour, cognitive implications, academic integrity, or pedagogical outcomes
- Publications spanning the period 2000 to 2026, with particular emphasis on post-2020 sources
- Priority given to systematic reviews and high-impact publications to ensure academic credibility

Books from established academic publishers were also included where they provided foundational theoretical context unavailable in journal form.

5.3 Databases and Sources

Relevant literature was identified through the following academic databases and platforms:

- Google Scholar
- SpringerLink
- ACM Digital Library
- MDPI Open Access Journals
- ERIC (Education Resources Information Centre)
- Wiley Online Library

Search terms used included combinations of: *generative AI in education, AI student learning, Cloud architecture, academic integrity ChatGPT, cognitive dependency AI, post-pandemic learning, and AI misuse higher education.*

5.4 Analytical Framework

The framework is based on three core dimensions: performance, scalability, and cost. Performance is analyzed using metrics such as latency, execution time, throughput, and cold start behavior. These indicators help measure how efficiently a serverless application processes user requests and delivers responses. Scalability is evaluated by examining the system's ability to handle varying workloads, including sudden traffic spikes and concurrent executions. Factors such as auto-scaling speed, concurrency limits, and workload adaptability are considered in this analysis. Cost is assessed based on the pay-as-you-go pricing model, including execution cost, memory usage, number of requests, and potential hidden costs like data transfer and API calls.

5.5 Ethical Considerations

As this study is entirely review-based and does not involve human participants, primary data collection, or sensitive personal information, formal ethical approval was not required. All sources cited in this paper are publicly available, peer-reviewed publications. Full attribution is provided for all referenced works in accordance with IEEE citation standards.

6. CONCEPTUAL FRAMEWORK AND ANALYTICAL IMPLEMENTATION

6.1 Overview of the Framework

The analytical framework for this study provides a structured approach to evaluate serverless web applications within modern cloud architecture. It defines the key parameters, variables, and evaluation criteria used to analyze the performance, scalability, and cost-effectiveness of serverless platforms.

The framework is based on three core dimensions: performance, scalability, and cost. Performance is analyzed using metrics such as latency, execution time, throughput, and cold start behavior. These indicators help measure how efficiently a serverless application processes user requests and delivers responses. Scalability is evaluated by examining the system's ability to handle varying workloads, including sudden traffic spikes and concurrent executions. Factors such as auto-scaling speed, concurrency limits, and workload adaptability are considered in this analysis. Cost is assessed based on the pay-as-you-go pricing model, including execution cost, memory usage, number of requests, and potential hidden costs like data transfer and API calls.

The framework also considers independent and dependent variables. Independent variables include the choice of cloud platform, such as AWS Lambda, Azure Functions, and Google Cloud Functions. Dependent variables include performance outcomes, scalability behavior, and overall cost. By analyzing

the relationship between these variables, the study identifies how different platforms influence system efficiency.

6.2 Phase Structure of the Framework

The phase structure of the framework for a serverless web application using modern cloud architecture defines the step-by-step process followed in the research and system evaluation. It ensures a systematic approach from problem identification to final analysis and conclusions.

Phase 1: Problem Identification and Requirement Analysis

This phase involves understanding the need for serverless web applications and identifying key requirements such as scalability, performance, and cost efficiency. It also includes defining the scope of the study and selecting relevant use cases.

Phase 2: Literature Review and Conceptual Understanding

In this phase, existing research studies, articles, and technical documentation are reviewed to understand serverless computing concepts, cloud architecture, and previous findings. It helps in identifying research gaps and building a theoretical foundation.

Phase 3: Framework Design and Planning

This phase focuses on designing the analytical framework, including selecting evaluation parameters such as performance, scalability, and cost. It also defines tools, platforms, and methodologies to be used in the study.

Phase 4: Data Collection

Relevant data is collected from multiple sources such as research papers, cloud provider documentation, and case studies. Performance metrics and cost data are gathered for different serverless platforms.

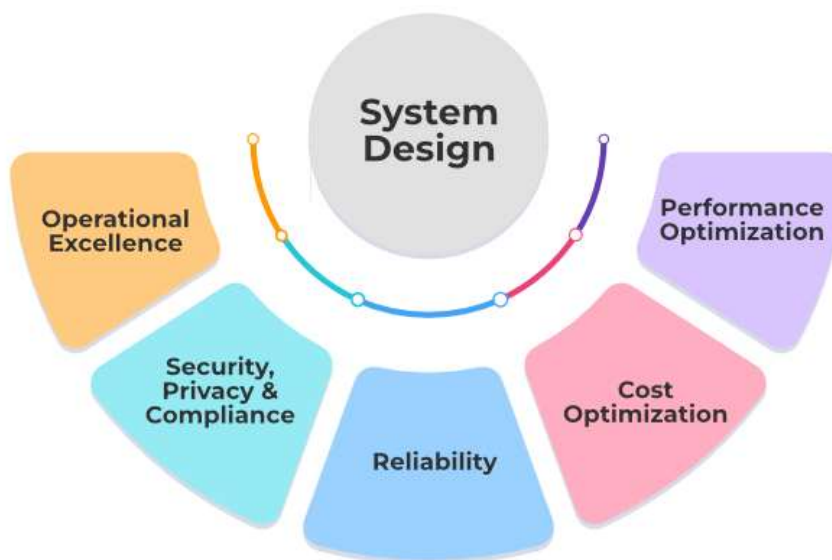
Phase 5: Implementation and Testing

In this phase, serverless web applications or case scenarios are analyzed and tested. Performance metrics like latency, execution time, and scalability are evaluated under different workloads.

Phase 6: Analysis and Comparison

The collected data is analyzed using tables, graphs, and comparative techniques. Different platforms are compared based on performance, scalability, and cost to identify their strengths and weaknesses.

Architecture Framework



7. DISCUSSION

Serverless architecture offers several significant advantages that make it highly suitable for modern cloud-based applications. One of the key benefits is automatic scaling, where the system dynamically adjusts resources based on incoming requests without manual intervention. This ensures that applications can handle both low and high traffic efficiently. Another important advantage is cost efficiency, as serverless platforms follow a pay-as-you-go model, meaning users are charged only for the actual execution time and resources used. This helps reduce unnecessary expenses associated with idle infrastructure. Additionally, serverless architecture eliminates the need for managing servers, allowing developers to focus more on application development rather than infrastructure maintenance.

Despite these benefits, serverless computing also introduces certain challenges that require careful consideration. One of the major issues is cold start latency, which occurs when a function is executed after a period of inactivity. During this time, the system needs to initialize resources, causing delays that can negatively impact performance, especially in real-time applications such as APIs and IoT systems. Another challenge is related to cost management. Although serverless is generally cost-effective, improper configuration, such as excessive function invocations or inefficient code, can lead to unexpected costs.

Furthermore, there are hidden costs associated with data transfer, API gateway usage, and monitoring services, which must be considered during system design. Monitoring and debugging can also be more complex due to the distributed and event-driven nature of serverless applications. Therefore, organizations must implement proper monitoring tools and logging mechanisms to track performance and identify issues.

To fully utilize the benefits of serverless architecture, it is essential to optimize function performance, reduce cold start delays, and carefully plan resource usage. Developers should design efficient, lightweight functions and adopt best practices for cost optimization. In conclusion, while serverless architecture provides powerful advantages in scalability and efficiency, it requires strategic planning and proper management to overcome its challenges and achieve optimal performance.

8. CONCLUSION

Serverless architectures have redefined the landscape of cloud-native application development, offering unparalleled benefits in terms of scalability, costefficiency, and operational simplicity. However, as this study highlights, the adoption of serverless computing must be guided by a nuanced understanding of its strengths and limitations.

Key Findings

1. Performance:

- Serverless platforms demonstrate strong capabilities in handling variable workloads, with automatic scaling mechanisms responding effectively to demand spikes.
- Challenges such as cold start latency remain a critical issue, particularly for applications requiring low-latency responses (e.g., real-time systems). Providers like AWS Lambda have made progress in optimizing these delays, but room for improvement persists.

2. Scalability:

- Serverless architectures excel in scaling applications seamlessly, often outperforming traditional server-based or container-based systems.
- The built-in concurrency handling across providers ensures that applications can sustain traffic surges without manual intervention, making serverless ideal for use cases such as IoT data processing and event-driven applications.

3. Cost:

- Serverless pricing models, based on actual execution time and resource consumption, offer cost predictability for many workloads.
- Despite this, hidden costs such as data transfer fees, cold starts in certain scenarios, and idle time for interconnected services must be carefully evaluated to avoid budget overruns.

4. Developer Experience:

- The ease of deployment and abstraction of infrastructure management has empowered developers to focus on innovation rather than operational concerns.
- However, vendor lock-in and the learning curve associated with serverless-specific tools and frameworks remain challenges to widespread adoption.

Recommendations

1. Tailor Serverless to Specific Use Cases:

- Organizations should evaluate their workload requirements to determine if serverless is the optimal solution. For applications requiring constant uptime or predictable traffic patterns, traditional or containerized deployments might still be preferable.

2. Optimize for Performance and Cost:

- Employ strategies like pre-warming functions to mitigate cold starts for latency-sensitive applications.
- Use monitoring tools to track execution time and identify cost-optimization opportunities.

3. Embrace Hybrid Architectures:

- A hybrid model combining serverless functions with containerized or server-based systems can provide the best of both worlds, enabling flexibility while maintaining control over critical aspects like data locality and performance.

4. Monitor Emerging Trends:

- Businesses should stay abreast of advancements in serverless computing, such as improved cold start optimizations, edge computing integration, and new tools for debugging and monitoring.

9. RECOMMENDATIONS

Based on the conceptual framework and synthesized findings of this review, the following recommendations are proposed for higher education institutions, educators, and policymakers navigating the integration of AI in student education:

Develop Clear and Practical AI Usage Policies: Institutions must move beyond vague prohibitions or blanket bans and establish structured, nuanced policies that define what constitutes acceptable AI use across different assessment types and academic contexts. Policies should be developed collaboratively with students and faculty, communicated clearly at the start of each academic year, and reviewed regularly as AI capabilities evolve. The absence of policy, as the literature consistently demonstrates, defaults to the Dependency Pathway .

Embed Digital Literacy Training into Curricula Digital literacy: It specifically the ability to critically evaluate AI-generated content, understand AI limitations, and use AI as one resource among many rather than a primary authority must become a core academic competency rather than an optional module. Students who enter higher education without this foundation are systematically more vulnerable to cognitive offloading and academic misuse .

Redesign Assessments to Require Original Reasoning: The most effective structural response to AI misuse is assessment redesign. Assignments that can be completed by AI standard essays, factual summaries, formulaic problem sets must be progressively replaced by assessments that require personal reflection, contextual analysis, oral defence, and applied problem-solving. When assessment design demands genuine student reasoning, the incentive to use AI as a replacement rather than a supplement is significantly reduced .

Position AI as a Collaborative Tool: It not a Forbidden One Institutions that frame AI purely as a threat to academic integrity miss the more productive opportunity to teach students how to use it responsibly. Structured exercises in which students use AI outputs as starting points for critical analysis identifying errors, challenging assumptions, and refining AI-generated content with their own reasoning build both AI literacy and independent thinking simultaneously .

Prioritize Equity in AI Access and Guidance: The post-COVID digital divide has shifted from a question of access to a question of guided use. Institutions serving students from resource-limited or digitally underserved backgrounds must ensure that AI literacy support is actively targeted toward those students rather than assumed. Without deliberate equity-focused intervention, the Augmentation Pathway will remain disproportionately accessible to already-advantaged students.

10. FUTURE SCOPE

The future of serverless computing is shaped by advancements in cloud technologies, evolving application requirements, and the need for more efficient, scalable, and cost-effective solutions. Here is an in-depth exploration of emerging trends and innovations that are defining the next phase of serverless computing: 1. Advances in Cold Start Optimization Problem: Cold starts—initial latency when a function is invoked after a period of inactivity—remain a significant drawback of serverless computing. They affect real-time applications like APIs, chatbots, and low-latency systems.

Emerging Solutions:

- Pre-warming mechanisms: Cloud providers are enhancing pre-warming to keep serverless functions ready for execution.
- Intelligent provisioning: AI-driven approaches predict traffic spikes and preemptively provision resources.
- Edge preloading: Functions are preloaded closer to the edge to reduce latency.

Example Innovations:

- AWS Lambda SnapStart improves Java function cold starts by pre-initializing execution environments.
- Providers are exploring WebAssembly for faster initialization and execution.

2. Hybrid Models: Serverless and Containers Trend: Increasingly, organizations are combining serverless architectures with containerized workloads to balance flexibility and performance.

Use Cases:

- Long-running processes that exceed serverless timeouts.
- Applications requiring finer control over runtime environments.

Tools and Platforms:

- Kubernetes-based serverless frameworks like Knative enable containerized workloads with serverless benefits.
- AWS Fargate and Azure Container Instances bridge the gap between serverless and containers. Impact: This hybridization allows developers to take advantage of the cost efficiency of serverless while retaining the control offered by containers.

3. Growth of Edge Computing and Serverless**Integration**

What's Changing: The shift towards decentralized computing is accelerating, with serverless functions moving closer to the data source (e.g., IoT devices).

Key Drivers:

- Real-time data processing needs in applications like autonomous vehicles, augmented reality, and IoT.
- Reduced data transfer costs by processing data locally rather than in central cloud regions.

Example Platforms:

- AWS Lambda@Edge and Azure Functions for edge devices.
- Cloudflare Workers for lightweight serverless execution on the edge.

Future Innovations: Improved orchestration tools will streamline deploying serverless functions across distributed nodes.

4. Expansion of Stateful Serverless Architectures Traditional Limitation: Serverless functions are inherently stateless, which complicates scenarios requiring state management, such as workflow orchestration or real-time collaboration.

Emerging Solutions:

- Durable Functions: Frameworks like Azure Durable Functions and AWS Step Functions allow for stateful execution through orchestration patterns.
- State storage advancements: New tools enable persistent state management with minimal latency. Benefits: This evolution reduces the complexity of building stateful applications while maintaining serverless scalability.

5. Serverless for AI and ML Workloads Current Landscape: AI/ML applications traditionally rely on GPUs and require complex infrastructure.

Serverless is now adapting to meet these needs.

Advancements:

- Providers like AWS and Google Cloud are introducing serverless GPU-based runtimes.
- Pre-built models and serverless inference capabilities reduce deployment times.

Impact: With serverless, developers can deploy AI/ML models at scale without managing GPU clusters, enabling cost-effective, on-demand AI solutions.

6. Multi-cloud and Polyglot Serverless Architectures Trend: Organizations are moving towards multi-cloud strategies to reduce vendor lock-in and enhance flexibility.

Implications for Serverless:

- Cross-provider tools like Serverless Framework, OpenFaaS, and Knative facilitate serverless deployments across multiple clouds.
- Support for multiple programming languages and runtimes enables diverse developer teams to leverage serverless technologies.

Future Outlook: Standardized APIs and interoperability protocols will simplify multi-cloud serverless adoption.

11. ACKNOWLEDGEMENT

The author would like to express sincere gratitude to Dr. Swati Joshi, our research guide, Department of Computer Application, PVG's College of Science and Commerce, Pune, Maharashtra, India, for their continuous guidance, valuable suggestions, and unwavering support throughout the development of this research work. Their mentorship significantly shaped the direction, depth and academic quality of this study. The author also wishes to thank the Department of Computer Application at PVG's College of Science and Commerce, Pune, Maharashtra, India, for providing a supportive academic environment and the resources necessary to complete this research.

12. REFERENCES

- [1] Fan, C. F., Jindal, A., & Gerndt, M. (2020). Microservices vs Serverless: A Performance Comparison on a Cloud-native Web Application. CLOSER 2020. Retrieved from <https://www.scitepress.org>
- [2] Kaviani, N., Kalinin, D., & Maximilien, M. (2019). Towards Serverless as Commodity: A Case of Knative. Proceedings of the 2019 ACM International Workshop on Serverless Computing. Retrieved from <https://dl.acm.org>

- [3] McGrath, G., & Brenner, P. R. (2017). Serverless Computing: Design, Implementation, and Performance. 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). Retrieved from <https://ieeexplore.ieee.org>
- [4] Pelle, I., Czentye, J., & Dóka, J. (2019). Towards Latency-Sensitive Cloud-Native Applications: A Performance Study on AWS. IEEE International Conference on Cloud Computing Technology and Science. Retrieved from <https://ieeexplore.ieee.org>
- [5] Mahmoudi, N., & Khazaei, H. (2020). Performance Modeling of Serverless Computing Platforms. IEEE Transactions on Cloud Computing. Retrieved from <https://ieeexplore.ieee.org>
- [6] Bonam, V. S. M., Vangoor, V. K. R., & Alluri, V. R. R. (2018). Serverless Computing for DevOps: Practical Use Cases and Performance Analysis. Advances and Broad Applications in Computing. Retrieved from <https://dlabi.org>
- [7] Kritikos, K., & Skrzypek, P. (2018). A Review of Serverless Frameworks. IEEE/ACM International Conference on Utility and Cloud Computing. Retrieved from <https://ieeexplore.ieee.org>
- [8] Mohanty, S. K., & Premsankar, G. (2018). An Evaluation of Open Source Serverless Computing Frameworks. International Conference on Cloud Computing and Services Science. Retrieved from <https://research.aalto.fi>
- [9] Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, A. (2017). A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-aService) Platforms. 2017 IEEE International Conference on Cloud Computing Technology and Science. Retrieved from