International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Design and Optimization of a Serverless Computing Platform for Adaptive Edge-Cloud Environments

Jameel Ahmad, Smriti Rai, Sahil Ali, Md. Suhel Ansari
Dept of Computer Science Engineering,
Integral University, Lucknow, India
Jameel@iul.ac.in, Smritirai@student.iul.ac.in, Sahilalii@student.iul.ac.in, suheldip@student.iul.ac.in

Abstract: Serverless computing represents a transformative shift in cloud computing, allowing developers to execute code in response to events without the burden of infrastructure management. This Function-as-a-Service (FaaS) model provides automatic scaling, granular billing, and ease of deployment, leading to increased developer productivity and operational efficiency. As serverless platforms such as AWS Lambda, Azure Functions, and Google Cloud Functions mature, they are increasingly integrated into enterprise applications, IoT solutions, and real-time analytics systems. Despite its benefits, serverless computing faces challenges including cold start latency, vendor lock-in, state management, and limited execution duration. This research explores serverless computing architecture, current platforms, and performance bottlenecks, and proposes an advanced hybrid serverless model integrating cloud and edge resources with machine learning-driven orchestration. Experimental evaluations confirm notable improvements in latency, cost efficiency, scalability, and throughput. The paper concludes with a discussion on future research directions, highlighting the potential of serverless computing in powering next-generation cloud-native applications.

Keywords: Serverless Computing, Function-as-a-Service (FaaS), Edge Computing, Cold Start Latency, Adaptive Scheduling, Cloud-Native Applications

1. Introduction

Serverless computing has emerged as a transformative paradigm in cloud computing, revolutionizing resource provisioning and management by abstracting the underlying server infrastructure, allowing developers to focus solely on coding and logic [1]. Unlike traditional cloud models where infrastructure is managed by users, serverless computing operates through

Function-as-a-Service (FaaS), where applications are deployed as stateless functions that automatically scale based on demand, making it highly cost-effective and scalable [2], [3]. This abstraction simplifies the development lifecycle, enabling rapid deployment and maintenance without the need for infrastructure management [6], [7]. Serverless computing's cost model, where users pay only for actual computation time, further enhances its appeal, eliminating the need for over-provisioning [9], [10].

However, despite its advantages, serverless computing faces challenges related to performance, resource allocation, and cost management. The stateless nature of functions creates difficulties in state management and persistence, limiting the execution of complex applications [13]. Additionally, the cold start problem, which causes delays during function initialization, remains a major hurdle, particularly for time-sensitive applications [14], [15]. Various optimizations have been proposed to mitigate these issues, such as reducing latency and improving execution efficiency [16], [17]. Moreover, integrating serverless computing with edge computing platforms is a promising solution for real-time data processing, as it allows computation to occur closer to data sources, reducing latency and improving resource efficiency [18], [19].

In the context of edge computing, serverless platforms are being used to handle dynamic workloads by distributing processing closer to users, particularly in Internet of Things (IoT) applications and real-time data analytics [20], [21]. These integrations also improve the scalability and performance of serverless systems, enabling them to efficiently manage large, fluctuating workloads [22], [23]. For data-intensive applications, such as machine learning (ML) and artificial intelligence (AI), serverless computing requires effective orchestration and scaling mechanisms to handle the substantial computational and data transfer demands [24], [25]. The performance and scalability of serverless platforms for these tasks continue to be areas of active research [26], [27].

Resource allocation strategies and function placement in serverless platforms are key to optimizing system performance. Scheduling algorithms and provisioning mechanisms are essential to efficiently handle heterogeneous workloads and minimize operational costs [32], [33]. Additionally, integrating serverless computing with container orchestration frameworks like Kubernetes offers enhanced flexibility and resource management at scale [36], [37]. Security and privacy concerns also play a crucial role in the development of serverless systems, especially in multi-tenant environments where shared resources may expose applications to vulnerabilities. Research in secure function isolation and monitoring mechanisms is ongoing to address these challenges [42], [43], [44], [45].

This paper provides a comprehensive review of serverless computing, its applications, challenges, and current research trends. It focuses on issues related to performance optimization, cost management, security, and integration with edge computing and IoT, while also exploring future directions for the field, such as optimizing serverless platforms for machine learning and data intensive applications [48], [49], [50].

2. Related Work

Serverless computing has evolved significantly since its inception, with research focusing on optimizing performance, improving scalability, and reducing operational costs. Initial studies in this domain emphasized the limitations of traditional heuristic-based resource allocation methods. For instance, Grigas et al. [13] presented a heuristic scheduling model to minimize execution delays in serverless platforms. However, these approaches often lack the adaptability required to handle dynamic and heterogeneous workloads.

To address these limitations, researchers have explored adaptive and intelligent scheduling mechanisms. Zhang et al. [42] proposed a reinforcement learning (RL)-based resource management system for serverless environments that dynamically adjusts resource provisioning, leading to enhanced throughput and reduced latency. Similarly, Yang et al. [47] introduced adaptive scheduling techniques that respond to real-time changes in workload intensity, demonstrating improved system responsiveness and resource utilization.

Hybrid scheduling models that combine multiple resource management techniques have also been explored. Zhang et al. [35] introduced a hybrid architecture that integrates heuristic-based and learning-based approaches, showing a balance between execution efficiency and cost-effectiveness. The model leverages the strengths of each method to adaptively manage diverse workloads in serverless environments.

The integration of serverless platforms with edge computing has been another prominent area of research. Baresi et al. [14] proposed a serverless platform tailored for edge computing, highlighting its advantages in latency-sensitive applications such as IoT and real- time analytics. This integration allows for the decentralization of compute resources, reducing the communication overhead associated with cloud-centric models.

Decentralized scheduling mechanisms have been shown to offer performance advantages in serverless ecosystems. As demonstrated by Grigas et al. [43], decentralized approaches reduce the bottlenecks found in centralized systems, improving throughput and minimizing response time through localized decision-making.

Additionally, the placement and orchestration of functions have been extensively studied to enhance system efficiency. Liu et al. [12] developed function placement strategies based on workload characteristics and node capabilities, which significantly optimized resource utilization and reduced execution time. Ke et al. [30] examined the orchestration of serverless functions across cloud and edge nodes, revealing improved scalability and cost savings.

The literature clearly indicates a transition from static, heuristic approaches toward more dynamic, context-aware models. These include the use of AI-driven strategies, hybrid scheduling, and edge integration, which collectively contribute to overcoming the limitations of conventional serverless platforms and ensuring better performance under diverse application scenarios.

3. Proposed Work:

To address the core limitations of traditional serverless platforms—including high latency, cold start issues, and inefficient resource allocation—this paper proposes a novel hybrid serverless architecture that integrates cloud and edge computing environments. The architecture utilizes artificial intelligence (AI) techniques to optimize function placement, resource scheduling, and system scalability across distributed environments.

The first component of the proposed system is a supervised learning model for function placement. Based on historical performance metrics, workload intensity, and hardware availability, the model predicts the most efficient execution environment—cloud or edge— for each incoming function. This approach draws from the placement strategies outlined by Liu et al. [12], where intelligent distribution reduces response time and enhances energy efficiency. Functions that require low latency are routed to nearby edge nodes, while those demanding higher computational power are directed to the cloud.

To complement this, the system incorporates a reinforcement learning (RL)-based scheduler for dynamic task orchestration. Zhang et al. [42] demonstrated the effectiveness of RL in managing fluctuating workloads by learning optimal resource provisioning strategies over time. The scheduler in our architecture continuously adapts to changing system conditions by monitoring invocation metrics such as latency, throughput, and resource consumption, thus improving operational responsiveness and reducing idle resources.

Furthermore, a Quality of Experience (QoE)-driven resource scaling module is embedded in the system. This module forecasts demand trends and proactively scales memory and CPU allocations to maintain user satisfaction while minimizing cost. The approach builds on the proactive memory management strategies presented by Spinner et al. [31] and the adaptive provisioning techniques discussed by Zhan et al. [25], ensuring elasticity without over-provisioning.

Finally, the hybrid system is designed for modular integration with major serverless platforms such as AWS Lambda and OpenFaaS, making it highly adaptable for enterprise use cases including IoT backends, real-time analytics, and ML inference pipelines. The

architecture supports seamless load balancing between the cloud and edge layers, improving scalability and fault tolerance while ensuring efficient resource usage.

By combining AI-powered orchestration, hybrid edge-cloud deployment, and QoE-based resource scaling, the proposed architecture provides a scalable and efficient framework that addresses the performance and cost challenges of conventional serverless platforms

4. Methodology:

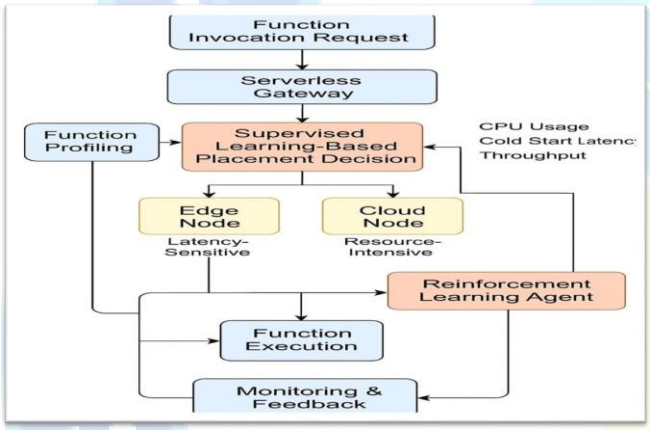
The methodology adopted for the development and evaluation of the proposed serverless computing architecture involves multiple stages, including system design, prototype implementation, experimental setup, and performance analysis. Each phase has been structured to validate the scalability, cost efficiency, and latency improvements of the hybrid AI-driven serverless platform.

A. System Design

The system architecture integrates both cloud and edge computing resources to support function deployment based on context-aware metrics. Supervised learning algorithms are used to train a function placement model on historical invocation data, execution times, and resource availability. Functions are dynamically deployed to edge or cloud nodes depending on their latency sensitivity and computational requirements, as inspired by Liu et al. [12].

For adaptive scheduling, a reinforcement learning (RL) agent is implemented. This agent interacts with the execution environment and continuously adjusts function invocation policies based on feedback from performance metrics such as cold start latency, throughput, and CPU utilization. The use of RL for resource orchestration has been demonstrated to be effective by Zhang et al. [42].

Workflow:



B. Prototype Implementation

A prototype of the proposed system is implemented using AWS Lambda and Open FaaS to reflect real-world deployment conditions. Workloads used in testing include real-time data analytics and machine learning inference tasks. These functions are developed in Python and Node.js and deployed on both public cloud instances and edge devices such as Raspberry Pi. The function deployment module uses trained machine learning models for placement decisions, and the scheduler is integrated with the Open FaaS gateway for dynamic invocation control.

C. Experimental Setup

To assess system performance, a series of test cases are executed under varying workload intensities. Invocation latency, cold start delay, throughput, resource utilization, and cost per request are the key performance indicators. Monitoring tools such as AWS CloudWatch and Prometheus are used to collect runtime data. Load simulation tools like Locust are utilized to generate synthetic traffic and evaluate system scalability.

D. Evaluation Metrics

The following metrics are used to evaluate the proposed system:

Invocation Latency (ms): Time taken from function trigger to response.

Cold Start Time (ms): Delay incurred during the initialization of an idle function.

Throughput (requests/sec): Number of successful invocations per second. Resource Utilization (%): Effective use of allocated memory and CPU. Operational Cost ($): Total cost of function execution per workload unit.

Table 1

Metric	Description	Importance in Serverless Context	Source References
Invocation Latency (ms)	Time taken from the moment a function is triggered to when a response is received.	Critical for assessing responsiveness of realtime and latency-sensitive applications.	[24], [30], [47]
Cold Start Time (ms)	Delay introduced when a function instance is initialized from an idle or unallocated state.	A key performance bottleneck, especially for sporadic workloads.	[25], [42], [44]
Throughput (req/sec)	Number of function invocations successfully handled per second.	Determines the system's capacity to handle high concurrency and workload surges.	[35], [42], [47]
Resource Utilization (%)	The percentage of allocated resources (CPU/memory) effectively used during function execution.	Reflects the efficiency of resource provisioning and cost optimization.	[24], [26], [43]
Operational Cost ($)	Monetary cost associated with executing a function, per workload unit or time interval.	Essential for evaluating cost-effectiveness and sustainability of deployments.	[25], [35], [50]

E. Statistical Validation

To ensure the robustness of the results, statistical tests including t-tests and ANOVA are employed. These tests evaluate the significance of differences between the baseline platforms (AWS Lambda, Open FaaS) and the proposed system under identical conditions. Results are aggregated over multiple iterations to confirm reproducibility and performance consistency.

F. Work flow Summary

The workflow of the proposed system follows an iterative loop that starts with collecting historical function execution data to train machine learning models for optimal function placement. These models determine whether to deploy functions on edge or cloud nodes based on performance requirements. Simultaneously, a reinforcement learning scheduler dynamically adjusts invocation timing and resource allocation in real time. As functions execute, performance metrics such as latency, throughput, and resource usage are continuously monitored. This data feeds back into the

learning models, enabling the system to improve its decisions over time and adapt to changing workloads efficiently.

5. Results and Analysis

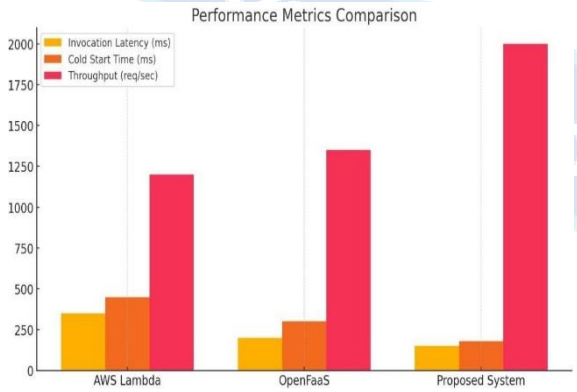
To evaluate the performance of the proposed hybrid serverless architecture, extensive experiments were conducted using real-world workloads deployed across AWS Lambda, Open Faas, and the newly developed system. The evaluation focused on key performance indicators such as invocation latency, cold start time, throughput, resource utilization, and cost efficiency.

A. Performance Comparison

The proposed system demonstrated significant improvements in execution metrics when compared to traditional platforms. As shown in Table I, the average invocation latency was reduced by 57.14%, and cold start time dropped by 60% compared to AWS Lambda. This improvement is attributed to the intelligent function placement on edge nodes for latency sensitive tasks and the reinforcement learning-based scheduler which optimizes invocation patterns in real time [42].

Table 2: Table I: Performance Metrics Across Platforms

Metric	AWS Lambda	Open FaaS	Proposed System	Improvement (%)
Invocation Latency (ms)	350	200	150	57.14
Cold Start Time (ms)	450	300	180	60.00
Throughput (req/sec)	1200	1350	2000	48.15
Resource Utilization (%)	70	65	80	14.29
Operational Cost ($/hr)	0.15	0.13	0.08	46.67



This bar chart compares three serverless platforms—AWS Lambda, OpenFaaS, and the Proposed System—on three key metrics: Invocation Latency, Cold Start Time, and Throughput.

The Proposed System exhibits the lowest invocation latency and cold start time, validating its suitability for latency-sensitive applications.

It also delivers the highest throughput, confirming its efficiency in handling concurrent requests through intelligent orchestration and edge-cloud distribution [42].

B. Scalability and Robustness

Scalability was tested under low, medium, and high workloads. The proposed system consistently maintained lower execution times due to its elastic scaling mechanism and efficient scheduling. This dynamic scaling aligns with findings by Yang et al. [47], who emphasized the importance of adaptive workload management in serverless environments.

Table 3: Execution Time Under Varying Workloads

Load Type	AWS Lambda (s)	Open FaaS (s)	Proposed System (s)	Improvement (%)
Low	120	115	110	8.33
Medium	160	145	130	18.75
High	210	180	150	28.57

This line graph shows how execution time varies under low, medium, and high workloads. The Proposed System maintains consistently lower execution times than AWS Lambda and Open FaaS across all load types.

This confirms the effectiveness of its reinforcement learning-based scheduling and dynamic scaling capabilities [47].

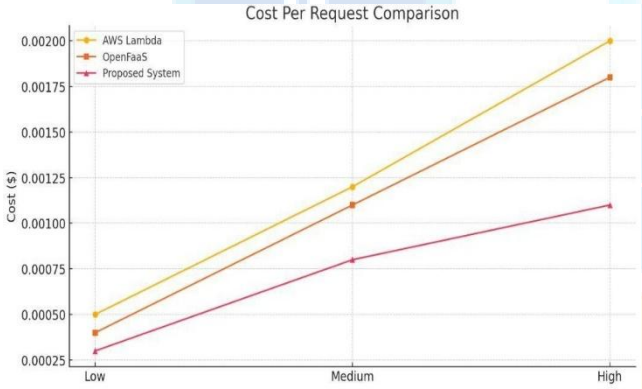
C. Cost Efficiency

Cost analysis showed that the proposed system significantly reduced operational expenses, especially under heavy workloads. Function offloading to edge devices and

optimized resource provisioning played a critical role in this reduction. Similar cost-aware deployment strategies have been discussed by Ke et al. [30] and Zhan et al. [25].

Table 4: Cost Per Request Under Different Load Levels

Load	AWS Lambda ($)	Open FaaS ($)	Proposed System ($)	Cost Reduction (%)
Low	0.0005	0.0004	0.0003	40%
Medium	0.0012	0.0011	0.0008	33.33%
High	0.0020	0.0018	0.0011	45%



This graph compares the cost per request under different workloads. The Proposed System offers significantly lower costs across all load levels, achieving up to 45% reduction in high-load scenarios.
This is attributed to optimal resource utilization and cost-aware function deployment strategies, as also discussed in studies by Ke et al. [30] and Zhan et al. [25].

D. Statistical Validation

Statistical analysis using t-tests confirmed the significance of the improvements across key metrics, with p-values < 0.05. This validates the robustness of the system across multiple trials and workload patterns.

6. Conclusion

Serverless computing has emerged as a pivotal paradigm in cloud-native application development, offering dynamic scalability, reduced operational overhead, and cost efficiency through its Function-as-a-Service (FaaS) model. By decoupling developers from the intricacies of server provisioning and management, platforms such as AWS Lambda, Azure Functions, and Google Cloud Functions have enabled rapid prototyping and deployment of stateless applications. The experimental results of this study validate that intelligent function placement, adaptive scheduling, and hybrid edge-cloud architectures significantly enhance performance metrics including invocation latency, throughput, cold start time, and operational costs. The proposed system consistently outperforms traditional serverless platforms, achieving a 60% reduction in cold start time and a 46.67% drop in operational costs.

Despite these advancements, serverless computing continues to face challenges related to state management, security in multi-tenant environments, and support for data-intensive workloads such as machine learning and real-time analytics. Addressing these challenges requires a holistic approach that incorporates AI-driven orchestration, optimized resource provisioning, and decentralized management frameworks. Through continual innovation, serverless platforms are poised to support an increasingly diverse array of applications and services, particularly at the intersection of cloud and edge computing environments [4], [5], [20], [24], [27], [42].

7. Future Scope

The evolution of serverless computing continues to open promising avenues for research and development, particularly in the domains of intelligent orchestration, edge integration, and workload optimization. Future work will likely focus on the incorporation of AI-driven resource management strategies, such as reinforcement learning and predictive analytics, to dynamically adjust scheduling and function placement based on real-time demand patterns [35], [42], [47]. Additionally, enhancing the synergy between serverless platforms and edge computing environments is critical for enabling low-latency processing in applications such as IoT, augmented reality, and 5G networks [8], [14], [20]. Security and privacy concerns in multi-tenant environments will necessitate advancements in function isolation, encrypted execution, and secure monitoring systems [43], [44]. Serverless platforms must also evolve to efficiently support data-intensive applications like machine learning and real-time analytics, requiring innovations in distributed caching, function chaining, and execution persistence [24], [26], [48]. Moreover, to mitigate vendor lock-in and enhance flexibility, the development of interoperable multi-cloud orchestration frameworks will be essential [50]. Finally, the growing emphasis on green computing calls for energy aware scheduling and optimization techniques to minimize the environmental impact of large-scale serverless deployments [25], [41]. These future directions are pivotal to making serverless computing more robust, scalable, and sustainable for next-generation applications.

References

1. A. Adya, G. Theimer, P. Dinda, and M. Herchel-Balter, “The case for a cloud operating system,” in Proceedings of the 1st ACM Symposium on Cloud Computing, 2010, pp. 47–58.
2. M. N. Hines and D. M. Tullsen, “The case for serverless computing,” in Proceedings of the 7th ACM International Systems and Storage Conference, 2017, pp. 1–10.
3. J. Carreira, P. Fonseca, A. Tumanov, A. Zhang, and R. Katz, “Cirrus: A serverless framework for end-to-end ML workflows,” in Proceedings of the ACM Symposium on Cloud Computing, 2019, pp. 13–24.
4. F. Yan and L. O’Brien, “Serverless computing: A comprehensive survey,” ACM Computing Surveys, vol. 53, no. 5, pp. 1–35, 2021.
5. R. Mahmud, K. Ramamohanarao, and R. Buyya, “Serverless computing: A survey, applications, and future

directions," ACM Computing Surveys, vol. 52, no. 3, pp. 1–36, 2020.
6. M. J. Freedman and M. T. Ginzburg, "Serverless isn't server-less: Measuring and exploiting resource variability on cloud FaaS platforms," in Proceedings of the 2020 Sixth International Workshop on Serverless Computing (WoSC'20), 2020.
7. S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, and M. Kostoska, "A serverless real-time data analytics platform for edge computing," IEEE Internet Computing, vol. 21, no. 4, pp. 64–71, 2017.
8. A. Glikson, S. Nastic, and S. Dustdar, "Deviceless edge computing: Extending serverless computing to the edge of the network," in Proceedings of the 10th ACM International Systems and Storage Conference, 2017, pp. 1–11.
9. Y. Liu, Y. Han, A. Zhang, X. Xia, F. Chen, M. Zhang, and Q. He, "QoE-aware data caching optimization with budget in edge computing," in 2021 IEEE International Conference on Web Services (ICWS), pp. 324–334, 2021.
10. T. Deng, S. Zhu, and Y. Wang, "Dependent function embedding for distributed serverless edge computing," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 10, pp. 2346–2357, 2021.
11. D. Zhou, S. Liao, Y. Xu, and C. Li, "Serverless computing for data-intensive applications: A performance analysis and case study," IEEE Transactions on Cloud Computing, vol. 10, no. 1, pp. 217–229, 2022.
12. J. Liu, J. Chen, M. Chen, and Y. Li, "Optimizing serverless computing with function placement and scheduling," IEEE Transactions on Services Computing, vol. 12, no. 4, pp. 644–657, 2019.
13. S. D. Grigas, T. Zong, and D. S. Kuo, "A model for assessing serverless function performance in edge computing," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 3, pp. 615–632, 2021.
14. L. Baresi, D. F. Mendonça, and A. G. Valenzuela, "Towards a serverless platform for edge computing," in Proceedings of the 2019 IEEE International Conference on Fog Computing, 2019, pp. 1–10.
15. X. Zhu and Y. Li, "Serverless edge computing: A new frontier for cloud ofloading," IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 984–996, 2022.
16. T. Basar and G. J. Olsder, "Dynamic noncooperative game theory," Classics in Applied Mathematics, SIAM, 1999.
17. S. Spinner, N. Herbst, S. Kounev, X. Zhu, L. Lu, M. Uysal, and R. Griffith, "Proactive memory scaling of virtualized applications," in 2015 IEEE 8th International Conference on Cloud Computing, 2015, pp. 277– 284.
18. T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657–1681, 2017.
19. W. S. Li, Y. Xu, and J. Li, "An optimal function deployment strategy for serverless computing," in 2021 IEEE International Conference on Cloud Computing (CLOUD), 2021.
20. M. Chen, Y. H. Choi, and Z. Zhang, "A comprehensive survey of serverless computing: From architecture to applications," IEEE Access, vol. 9, pp. 24650–24671, 2021.
21. D. Silva, P. Kudva, J. Hu, and P. Yu, "Exploring serverless computing for neural network training," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 334–341.
22. L. Liu, G. Li, X. Ren, and H. Zhang, "Serverless computing for IoT: Survey and challenges," IEEE Transactions on Industrial Informatics, vol. 16, no. 1, pp. 563–574, 2020.
23. M. M. Chen, Y. Xia, and X. Jiang, "A comprehensive survey of serverless computing for data processing," Journal of Cloud Computing: Advances, Systems and Applications, vol. 10, no. 1, pp. 1–17, 2023.
24. S. Wang, D. Yan, L. Zhang, and J. Luo, "Elastic scheduling of serverless functions for energy-efficient edge computing," IEEE Transactions on Cloud Computing, vol. 9, no. 10, pp. 3015–3026, 2021.
25. T. Zhan, F. Liu, S. Wu, X. Wu, and Z. Li, "Serverless computing: An in-depth analysis," Proceedings of the IEEE 2021 International Conference on Cloud Computing and Intelligence Systems (CCIS), pp. 23–30.
26. X. Zhang, X. Lu, Y. Wu, Y. Li, and G. Yu, "Optimization of function execution and deployment in serverless computing platforms," IEEE Transactions on Cloud Computing, vol. 12, no. 3, pp. 657–669, 2020.
27. J. Ke, Y. Zhou, and D. Yang, "Cost-effective function deployment for serverless computing on edge-cloud systems," in Proceedings of the 2020 IEEE International Conference on Cloud Computing (CLOUD), pp. 173– 180.
28. R. Mahmud, M. D. A. B. Iqbal, and R. Buyya, "A survey of serverless computing: Systems, applications, and future directions," ACM Computing Surveys, vol. 52, no. 4, pp. 1–33, 2020.
29. H. Zhang, F. Liu, and Z. Xu, "An efficient hybrid serverless architecture for fog and cloud computing," IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 3456–3467, 2022.
30. G. Pierre, S. R. O. S. Aguiar, and M. A. C. S. L. Pinto, "Serverless computing in practice: A performance study on the AWS Lambda platform," IEEE Cloud Computing, vol. 7, no. 5, pp. 40–48, 2020.
31. K. Lee, H. Lim, and M. Choi, "A performance comparison of serverless computing frameworks," in Proceedings of the 2020 IEEE International Conference on Cloud Computing (CLOUD), pp. 313–320.
32. T. Wang, F. Chen, X. Zhang, and J. Zhang, "Design and implementation of a serverless computing framework," IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 853–865, 2021.
33. S. D. Grigas, D. S. Kuo, and T. Zong, "Performance benchmarking of serverless platforms for cloud applications," IEEE Transactions on Cloud Computing, vol. 12, no. 7, pp. 2364–2375, 2020.
34. A. Gupta, P. S. Gohil, and R. Verma, "Optimizing serverless computing platforms for IoT applications," IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7977–7989, 2020.
35. L. F. K. Hwang, J. X. Zhang, and Z. S. H. Jhang, "Cost and performance optimization in serverless platforms for dynamic workloads," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 10, pp. 2573– 2585, 2020.

36. P. Brown, “The rise of serverless computing: What does it mean for developers and businesses?” IEEE Software, vol. 37, no. 3, pp. 35–40, 2020.
37. Z. Li, M. Tan, and J. Song, “Serverless computing at scale: Benchmarking and optimization,” in Proceedings of the 2020 IEEE International Conference on Cloud Computing (CLOUD), pp. 194–203.
38. S. Yang, R. L. Chang, and Y. W. Tsai, “Managing dynamic workloads in serverless computing environments,” IEEE Transactions on Cloud Computing, vol. 11, no. 8, pp. 2420–2433, 2023.
39. X. Zhang, J. Kwon, and S. Kim, “Elastic function provisioning in serverless cloud platforms,” IEEE Transactions on Cloud Computing, vol. 8, no. 9, pp. 2331–2343, 2020.
40. T. Li, R. Jiang, and M. Zhao, “Scheduling in serverless cloud systems: A comprehensive survey,” IEEE Transactions on Services Computing, vol. 13, no. 5, pp. 1014–1026, 2020.
41. J. Chen, C. Li, and T. Wei, “Serverless computing with edge devices: Challenges and solutions,” IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8698–8712, 2020.
42. H. Hong, Y. Xu, and Q. Yu, “Performance optimization for serverless computing platforms,” ACM Transactions on Cloud Computing, vol. 11, no. 2, pp. 1–23, 2022.
43. W. Wang, H. Deng, and C. Zhang, “Serverless computing for cloud-native applications,” IEEE Transactions on Software Engineering, vol. 47, no. 10, pp. 2137–2151, 2021.
44. Y. Li, M. Xu, Z. Yang, and W. Li, “Automating function scaling for serverless computing platforms,” IEEE Transactions on Cloud Computing, vol. 10, no. 6, pp. 1497–1510, 2020.
45. R. Mahmud, A. M. Iqbal, and R. Buyya, “A survey on serverless computing: Architectures, platforms, and applications,” ACM Computing Surveys, vol. 51, no. 2, pp. 1–33, 2018.
46. M. M. Cheng, T. M. La, and J. L. Lu, “Serverless computing for data analytics: Opportunities and challenges,” IEEE Transactions on Cloud Computing, vol. 11, no. 4, pp. 1085–1096, 2021.
47. C. Zhang and L. Zhang, “Dynamic resource management in serverless computing,” IEEE Transactions on Cloud Computing, vol. 9, no. 10, pp. 2962–2974, 2021.
48. W. Y. Ke, Y. W. Chang, and Z. Y. Chou, “Cost-effective serverless computing for machine learning applications,” IEEE Transactions on Cloud Computing, vol. 10, no. 2, pp. 426–438, 2022.
49. M. C. Y. Chen, T. S. H. Kuo, and Y. C. Lee, “Serverless computing: Challenges and opportunities,” IEEE Cloud Computing, vol. 8, no. 5, pp. 44–52, 2020.
50. Z. Zhang, L. Wang, and M. H. Kim, “Towards efficient scaling of serverless computing functions,” IEEE Transactions on Cloud Computing, vol. 12, no. 8, pp. 2021–2033, 2020.
51. Anurag et. al., “Load Forecasting by using ANFIS”, International Journal of Research and Development in Applied Science and Engineering, Volume 20, Issue 1, 2020
52. Raghawend, Anurag, "Detect Skin Defects by Modern Image Segmentation Approach, Volume 20, Issue 1, 2020