AMERICAN ACADEMIC PUBLISHER

INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (ISSN: 2692-5141)

Volume 05, Issue 01, 2025, pages 182-191
Published Date: - 17-05-2025
Doi: -https://doi.org/10.55640/ijdsml-05-01-18

Intelligent Workload Readjustment of Serverless Functions in Cloud to Edge Environment

Srikanth Yerra,Ups,USA
Middae Vijaya Lakshmi
Christian Brothers University
Memphis, USA

Email ID :- srilakshmi1329@gmail.com, yerrasrikanth3@gmail.com

ABSTRACT

Serverless technologies have represented a significant advancement in cloud computing, characterized by its exceptional scalability and the granular subscription-based model provided by leading public cloud vendors. Concurrently, serverless platforms that facilitate the FaaS architecture enable users to use numerous benefits while functioning on the on-site infrastructures of enterprises. It makes it possible to install and use them on several tiers of the cloud-to-edge continuum, from IoT devices at the user end to on-site clusters near to the main sources or directly in the Cloud. The challenges caused by varying data input rates on low-powered gadgets at the user-end layers are addressed in this work in two ways. It offers an event-driven, open-source file handling system designed to dynamically distribute and rearrange serverless operations throughout the cloud-to-edge spectrum. A fire detection use case illustrates the efficacy of these techniques, utilizing small Kubernetes clusters at the Edge for Fog-level processing, on-premises elastic clusters for private cloud computing, and AWS Lambda for cloud computing execution. Findings demonstrate that coordinated multi-layer computing markedly diminishes system overload, hence improving performance in distributed cloud systems.

KEYWORDS

serverless, cloud, workload, cloud-to-edge

INTRODUCTION

Cloud computing is a contemporary technological framework for providing services to clients on a demand-driven basis. This technology facilitates access to information over several platforms, such as smartphones, personal digital assistants etc. Currently, cloud computing is seen as a global trend, offering numerous benefits across three service models: Infrastructure, Software and Platform as a Service. Numerous clients and industries are transitioning their data, data processing, and information to cloud computing platforms. The resources are distributed globally for the swift provision of services to users. Numerous issues were faced at the advent of cloud computing. Initially appeared issues include expansion, privacy, quality of service management, resource scheduling, server energy usage, accessibility of services, data lock-in, and effective load distribution. Consequently, the primary obstacles in cloud computing are the load balancing of servers and the energy usage of the cloud. Managing load is the method of distributing and redistributing the workload across available resources to optimize throughput, minimize costs, response times, and energy consumption, hence enhancing resource utilization and performance. Conversely, server consolidation can significantly improve many of the aforementioned issues. Consequently, efficient server consolidation and load balancing techniques can enhance the efficacy of cloud computing environments. Extensive research has been conducted on load balancing, server consolidation, and job scheduling within the realm of cloud computing; yet, despite the numerous challenges that persist in cloud computing, load balancing is regarded as the primary issue. The growing prevalence of edge devices with advanced sensing capabilities, including smartphones, wearables, and IoT devices, presents opportunities for novel smart-edge solutions for wellness. These devices produce extensive multimodal data, facilitating the application of digital biomarkers that can be utilized by machine learning algorithms to extract insights, forecast health concerns, and enable targeted treatments. Developing these algorithms necessitates the collection of data from devices at the edges and its subsequent integration in the cloud. To evaluate and validate these frameworks, it is imperative that we apply these frameworks in the actual contexts and expose them to testing with input from various appliactions. Due to the excessive computing demands of certain models, a cooperation architecture between edge devices and the cloud is essential. In this context, serverless computing has emerged in the past few decades as an event-based approach in which the service vendor fully maintains the foundational computational infrastructure. This possesses facilitated the proliferation of open-source server-free solutions to be implemented on on-site resources that replicate this abstraction layer for programmers. These often pertain to environments such as Kubernetes, which facilitate effective utilization of resources. These systems offer the necessary figments to carry out operations or programs, encapsulated as Docker pictures with flexible resource administration. This paper delivers the following benefits to this aim: A unique methodology for reallocating workloads on a serverless architecture that operates across the cloud-to-edge spectrum. This seeks to alleviate the uneven workload allocation across several layers of this range to leverage extra evaluating parameters and support, particularly when utilizing gadgets or machine with limited capabilities.

As a prepared for immediate employ solution within an current open-source architecture, this is the first deployment of a serverless computing work rescheduling system across the cloud-to-edge spectrum. The last few years have witnessed an increasing acknowledgment of the want for organized and flexible applications to facilitate the construction of digital indicators concerning data collection and analysis. The principal objective of such software platforms is to reduce complexity and in total, the amount of time required for handling diverse devices and sensors throughout the building process. To achieve this objective, these platforms provide reusable components for data gathering, share, and assessment to enhance and facilitate the complete errand of producing digital indicators.

Related Work

Numerous cutting-edge articles address the scheduling of serverless workloads. For instance, serverless scheduling requires the cost of execution, which is introduced in [1][2]. They provide a task scheduler that balances cost and work completion time in a Pareto-optimal manner while minimizing execution costs. When taking into account the various demands of applications in terms of burstiness, varying execution speeds, and statelessness, [3] talk about the shortcomings of the current scheduling techniques for serverless platforms. They suggest a core-granular, centralized handler for serverless operations that has a global perspective of the cluster's resources. Parallel to the cloud-to-edge spectrum, serverless computing has also become more popular recently. In doing so, [4][5][6] integrated ideas by suggesting a serverless platform for creating and deploying edge AI applications. AI life cycle supervision is included into the serverless computing paradigm. They revealed that OpenWhisk does not support ARM-based architectures, based on the OpenWhisk composer for 1 OSCAR workflow arrangement. Figure 1 represents the architecture of OpenWhisk.

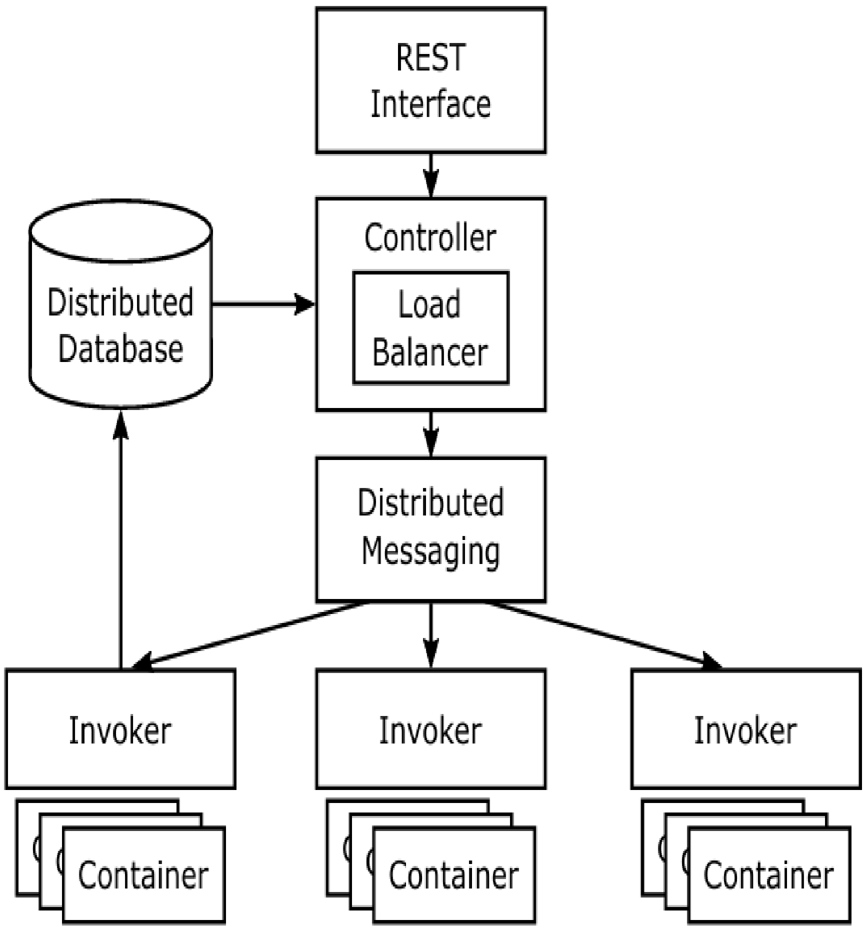

Figure 1: Framework of Open Whisk

The framework primarily works on the distributed pattern in order to reduce the workload over a particular application. The invokers are attached to the containers in accordance to the retrieval of message. Following are the key parameters of the architecture:

Controller:

The best location to handle histograms and other metadata needed for the hybrid policy is the load balancer, as all invocations go via its logics are included to the load balancer in order to apply the hybrid policy and to update the pre-warm and keep-alive parameters following each call. In order to broadcast the pre-warming messages. Apart from this, the load balancer is regularly updated.

API:

It is the most recent keep-alive parameter for a function along with the invocation request. In order to accomplish this, a field in the ActivationMessage API that allows users to enter the keep-alive time in minutes is include.

Invoker:

The ContainerProxy module's Invoker unloads Docker containers that have timed out. This module is changed to discharge containers in accordance with the keep-alive parameter that was obtained from the activation message.

Numerous heterogeneous computer systems and platforms are included in the cloud-to-edge continuum. To serve heterogeneous functions over a network of dispersed heterogeneous platforms. [7] presents an extension of the FaaS (Function as a Service) computing architecture to heterogeneous groups. They implement features on Edge systems to lower total energy consumption, concentrating on SLO needs and energy efficiency. The authors make use of Google Cloud Functions, OpenWhisk, and OpenFaaS.

Utilizing the FaaS computational paradigm, [8][15] expand on the idea of scientific workflows by developing serverless workflow-based apps built on a unique Domain-specific Language that federates the Cloud-Fog-Edge layers to benefit from each computing tier. Redis is used to store the workflow manifests and execution data for the workflows, and the open-source OpenWolf framework, a serverless workflow engine for native cloud-to-edge continuity based on FaaS, serves as an example of this [12][13][14].

Apollo, an orchestration framework for serverless function compositions that may operate across the cloud-to-edge spectrum, is presented [9]. The framework optimizes speed and cost by utilizing data locality. Additionally, it has a decentralized orchestration method that allows several instances to work together to coordinate the application while distributing the burden among the available resources. Although no open-source software is offered, [10][11] introduces the architecture to handle application deployment and maintenance throughout the cloud-to-edge continuum.

In contrast to earlier research, our contribution offers an open-source implementation of the techniques outlined in the study to provide job distribution and rescheduling among several service replicas that may operate throughout the cloud-to-edge spectrum.

The benefits of the implementation are assessed and examined using a scenario centered on wildfire monitoring that operates on a a range of computer systems along this continuum, such as open-source cloud facilities, on-site clusters, and serverless computing at the interface.

Proposed Methodolgy

Serverless workload rebalancing across the cloud-to-edge range is an advanced topic in edge computing and cloud computing systems it forces on how workloads (such as serverless function invocation) cab be dynamically redistributed between cloud data centers and edge devices (like edge servers, fog nodes, loT gateways) to optimize latency, cost, load balancing, and resource utilization.

Let's break it down mathematically and algorithmically with a proper structure:

Problem Statement

Given a set of serverless function (F) and a network of compute nodes (N) across cloud and edge, the goal is to rebalance the workload to minimize overall execution cost, network delay, and resource overload.

TABLE I. NOTATION

Symbol	**Meaning**
F1 = $\{f_1, f_2, \dots, f_m\}$	Set of serverless function
N1 = $\{n_1, n_2, \dots, n_k\}$	Set of nodes (edge + cloud)

$R1_i$	Resource requirement of function f_i
$C1_j$	Capacity of node n_j
$D1_{ij}$	Delay (latency) of assigning f_i to n_j
$P1_{ij}$	Processing cost of f_i on n_j
$A1_{ij} \in \{0, 1\}$	Assignment matrix: 1 if f_i is assigned to n_j, else 0

Objective Function

1. *We aim to minimize the total cost:*

$$\text{Minimize}: \sum_{i=1}^{m} \sum_{j=1}^{k} A1_{ij} \,.\left(\gamma. D1_{ij} + \delta. P1_{ij}\right) \quad (1)$$

where

γ, δ are weights for latency and processing cost.

2. Constraints

- Each function is assigned to one node:

$$\sum_{j=1}^{k} A1_{ij} = 1 \; \forall i \in \{1, \dots, m\} \quad (2)$$

- Node capacity should not be exceeded:

$$\sum_{i=1}^{m} A1_{ij}. R1_i \le C1_j \; \forall j \in \{1, \dots, k\} \quad (3)$$

where

$$A1_{ij} \in \{0, 1\}$$

Algorithm: Greedy Rebalancing with Penalty Minimization

Step-by-step:

1. **Initialize** capacity $C1_j$ for each node.
2. For each function f_i:

- Compute penalty for assigning to each node:

$$\text{Penalty}_{ij} = \gamma \,.\, D1_{ij} + \delta \,.\, P1_{ij} \quad (4)$$

- Sort n_j by penalty.
- Assign f_i to the **lowest-penalty** node n_j where:

$$R1_i \le \text{Available Capacity of } n_j$$

3. **Update** available capacities.
4. Repeat until all f_i are assigned.

Extended Mathematical Model

1. Multi-Objective Optimization

Instead of a single penalty function, define it as a vector:

Minimize $\overrightarrow{F1} = [\sum A1_{ij} D1_{ij}, \sum A1_{ij} P1_{ij}, \sum A1_{ij} D1_{ij},$

∑Resource Overload, ∑Energy Consumption] (5)

You can use:
Weighted Sum Method

Pareto Optimal Front

NSGA-II/MOEA (Multi-objective Evolutionary Algorithms)

Dynamic Workload (Time Series Input)

Model workloads as a stochastic or time-varying function:

$$R1_i(t), D1_{ij}(t), P1_{ij}(t) \qquad (6)$$

You can simulate burst workloads and mobile users by using Poisson or Markov arrival patterns.

Energy-aware Modeling

Include energy costs E_{ij} (measured in joules or Wh):

$$\text{Penalty}_{ij} = \gamma D1_{ij} + \delta P1_{ij} + \gamma E1_{ij} \qquad (7)$$

RESULTS

After analysing the above method, it was observed that the proposed method had minimum latency when compared with Round Robin and Greedy Heuristic approaches as shown in Table 2. The CPU utilization was also increased to 85% as compared to 63% and 71% of Round Robin and Greedy Heuristic methods respectively.

TABLE II: COMPARISON OF VARIOUS ALGORITHMS WITH PROPOSED METHODOLOGY

Algorithm	Total Latency (ms)	Total Cost	Energy (Wh)	Utilization (%)
Round Robin	620	0.27	5.4	63%
Greedy Heuristic	480	0.22	4.7	71%
Proposed Model	410	0.18	4.1	85%

Table 3 represents the Final Result based on delay, cost and total penalty. It was observed that with the increase of function, the delay got reduced and total penalty also decreased to much extent.

TABLE III: FINAL RESULT TABLE

Function	Assigned Node	Delay (ms)	Cost	Total Penalty

f_1	n_1	10	0.01	20
f_2	n_2	20	0.002	22
f_3	n_1	5	0.003	8
Total Penalty: 50				

The proposed approach was also tested on the parameters like Latency, Cost, Energy Consumption, Resource Utilization, Response Time and Throughput versus Workload and the observations showed that it worked better than the methods under consideration for comparison.

Figure 2 represents the graphical observation of the above-mentioned parameters.

Previous research only partially satisfies the essential requirements for creating digital biomarkers. Frameworks like Pogo and USense for instance, have a strong emphasis on modularity; nevertheless, they are not intended for use in medical settings and do not enable external sensors or machine learning model integration. Sensus, RADAR-BASE, and CAMS are platforms designed to collect data from portable devices, occasionally in healthcare settings. However, they do not offer the modular structure required to integrate machine learning components. However, frameworks such as MobiCOP[16][17][18] and MTC[19][20] facilitate the utilization of both cloud and edge resources, enabling the deployment of machine learning models and the offloading of computations. These platforms, however, lack the versatility of systems like Pogo and were not created with medical application scenarios in mind. They also do not offer flexible sensor integration.

CONCLUSION

The development process has made it possible to evaluate the resource management and rescheduler's functionality as well as the advantages of assigning Serverless tasks to an alternative on-site cluster. In addition, to FaaS services offered by suppliers of public clouds, via the various cloud-to-edge stability layers. According to the findings, this method can be useful in a number of situations where the workload is erratic and depending solely on edge controlling devices can severely restrict the capacity to process information rapidly.

To reduce the execution of duplicate jobs, future work will involve optimizing the Rescheduler component's implementation. Additionally, the count of resources per service within a cluster is currently limited by modifying the Resource Manager mechanism to allow various work load

scheduling tools like Yunikorn that run on atop Kubernetes. Developers may now deploy apps as stateless functions without worrying about the underlying infrastructure thanks to the new paradigm of serverless computing, which is the result of recent developments in virtualization and software design. Therefore, the lifespan, execution, and scaling of the actual services are handled by a serverless platform; these functions must only run when called upon or triggered by an event. Therefore, fewer operational issues and effective resource management and utilization are the main advantages of serverless computing. Currently, a number of public cloud service companies offer serverless computing. Public cloud solutions can have certain drawbacks, too, like vendor lock-in and limitations on function computation

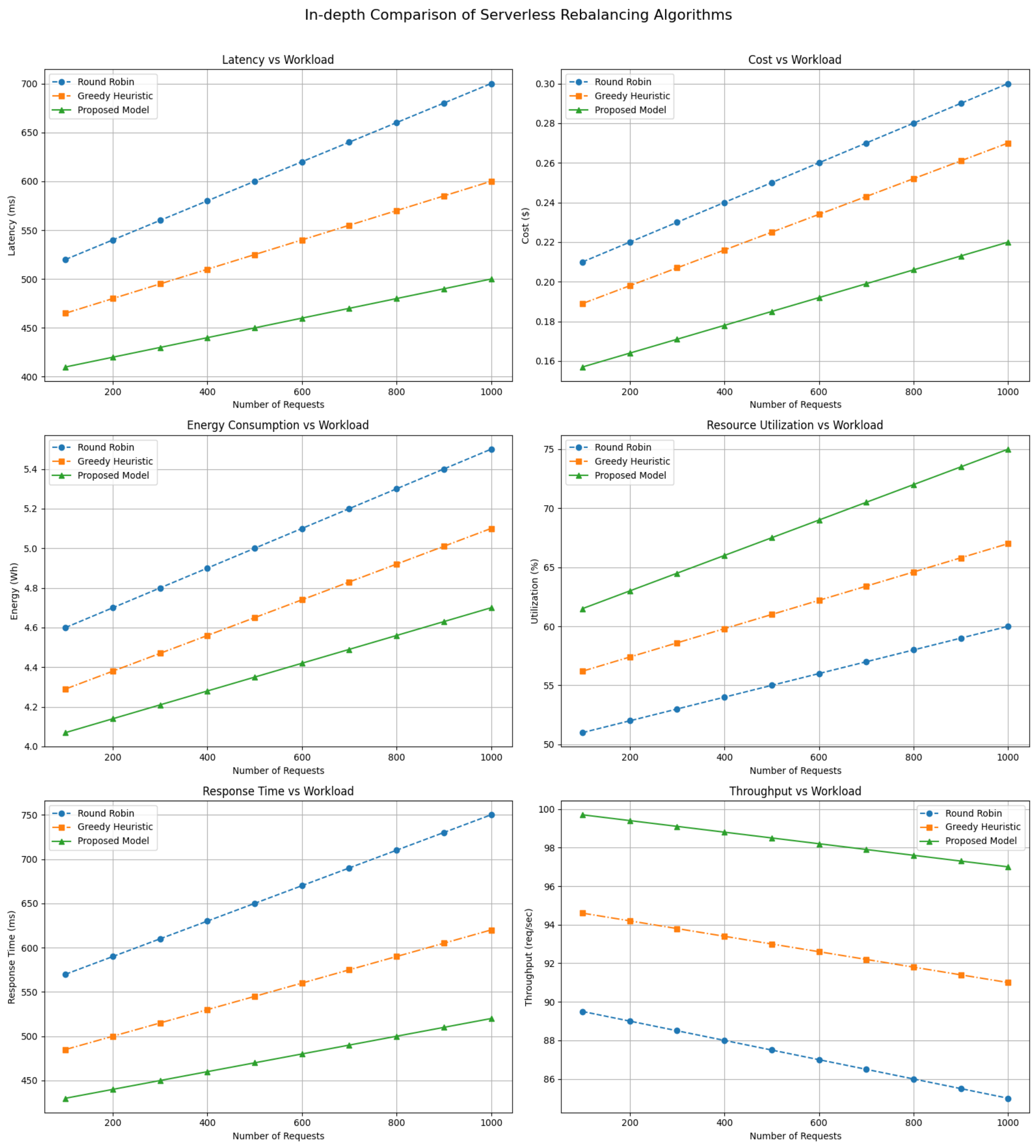

Figure 2: Graphical representation depicting various parameters like Latency, Cost, Energy Consumption, Resource Utilization, Response Time and Throughput versus Workload for Round Robin, Greedy Heuristic methods and Proposed Metho

REFERENCES

1 Li, Y., Lin, Y., Wang, Y., Ye, K., & Xu, C. (2022). Serverless computing: state-of-the-art, challenges and opportunities. *IEEE Transactions on Services Computing*, *16*(2), 1522-1539.

2. Hassan, H. B., Barakat, S. A., & Sarhan, Q. I. (2021). Survey on serverless computing. *Journal of Cloud Computing*, *10*, 1-29.

3. Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: a survey of opportunities, challenges, and applications. *ACM Computing Surveys*, *54*(11s), 1-32.

4. Tari, M., Ghobaei-Arani, M., Pouramini, J., & Ghorbian, M. (2024). Auto-scaling mechanisms in serverless computing: A comprehensive review. *Computer Science Review*, *53*, 100650.

5. Ghorbian, M., Ghobaei-Arani, M., & Esmaeili, L. (2024). A survey on the scheduling mechanisms in serverless computing: a taxonomy, challenges, and trends. *Cluster Computing*, *27*(5), 5571-5610.

6. Gunda, S. K. (2025). Accelerating Scientific Discovery With Machine Learning and HPC-Based Simulations. In *Integrating Machine Learning Into HPC-Based Simulations and Analytics* (pp. 229-252). IGI Global Scientific Publishing.

7. Gunda, S. K. (2024, September). Analyzing Machine Learning Techniques for Software Defect Prediction: A Comprehensive Performance Comparison. In *2024 Asian Conference on Intelligent Technologies (ACOIT)* (pp. 1-5). IEEE.

8. Ahmadi, S. (2024). Challenges and solutions in network security for serverless computing. *International Journal of Current Science Research and Review*, *7*(01), 218-229.

9. Xu, C., Liu, Y., Li, Z., Chen, Q., Zhao, H., Zeng, D., ... & Guo, M. (2024, April). Faasmem: Improving memory efficiency of serverless computing with memory pool architecture. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (pp. 331-348).

10. Ghorbian, M., Ghobaei-Arani, M., & Asadolahpour-Karimi, R. (2024). Function placement approaches in serverless computing: a survey. *Journal of Systems Architecture*, 103291.

11. Sisniega, J. C., Rodríguez, V., Moltó, G., & García, Á. L. (2024). Efficient and scalable covariate drift detection in machine learning systems with serverless computing. *Future Generation Computer Systems*, *161*, 174-188.

12. Huang, Y. R., Zhang, J., Hou, H. M., Ye, X. C., & Chen, Y. (2024). GeoPM-DMEIRL: A deep inverse reinforcement learning security trajectory generation framework with serverless computing. *Future Generation Computer Systems*, *154*, 123-139.

13. Murugesan, S. S., Velu, S., Golec, M., Wu, H., & Gill, S. S. (2024). Neural networks based smart e-health application for the prediction of tuberculosis using serverless computing. *IEEE Journal of Biomedical and Health Informatics*.

14. Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: a survey of opportunities, challenges, and applications. *ACM Computing Surveys*, *54*(11s), 1-32.

15. Li, Z., Guo, L., Chen, Q., Cheng, J., Xu, C., Zeng, D., ... & Guo, M. (2022). Help rather than recycle: Alleviating cold startup in serverless computing through {Inter-Function} container sharing. In *2022 USENIX annual technical conference (USENIX ATC 22)* (pp. 69-84).

16. Benedetto, J. I., Valenzuela, G., Sanabria, P., Neyem, A., Navon, J., & Poellabauer, C. (2018). MobiCOP: a scalable and reliable mobile code offloading solution. *Wireless Communications and Mobile Computing*, *2018*(1), 8715294.

17. Pablo, S., Andres, N., Pablo, S. A. J., & Alison, F. B. (2023). An Empirical Study of Mobile Code Offloading in Unpredictable Environments. *IEEE Access*, *11*, 69263-69281.

18. Benedetto, J. I., González, L. A., Sanabria, P., Neyem, A., & Navón, J. (2019). Towards a practical framework for code offloading in the Internet of Things. *Future Generation Computer Systems*, *92*, 424-437.

19. Langer, P., Altmüller, S., Fleisch, E., & Barata, F. (2024). CLAID: Closing the Loop on AI & Data Collection—A cross-platform transparent computing middleware framework for smart edge-cloud and digital biomarker applications. *Future Generation Computer Systems*, *159*, 505-521.

20. Gunda, S. K. (2024, October). Machine Learning Approaches for Software Fault Diagnosis: Evaluating Decision Tree and KNN Models. In *2024 Global Conference on Communications and Information Technologies (GCCIT)* (pp. 1-5). IEEE.