Computing and Informatics, Vol. 44, 2025, 1009–1039, doi: 10.31577/cai_2025_5_1009

EXPLORING PERFORMANCE AND ENERGY OPTIMIZATION IN SERVERLESS COMPUTING: A REVIEW

Jasmine Kaur, Inderveer Chana, Anju Bala

Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
Patiala, 147004, Punjab, India
e-mail: {jkaur_phd20, inderveer, anjubala}@thapar.edu

Abstract. Serverless computing brings another revolution to cloud computing as function-as-a-service (FaaS), where the applications are abstracted as a group of functions. Serverless applications are cost-effective and manage resources efficiently, but the lack of performance modeling and energy optimization affects the potential users' broad adoption of serverless computing. Performance enhancement and energy optimization are necessary to guarantee serverless applications' service level agreement (SLA). This review paper presents various performance metrics in serverless computing, including cost, scalability, latency, energy consumption, resource utilization, fault tolerance, and response time. Based on these metrics, various performance modeling and energy optimization techniques have been explored to reduce energy consumption and improve system efficiency. Furthermore, the review investigates software platforms for implementing serverless computing, including AWS Lambda, Apache OpenWhisk, Azure Functions, and Google Cloud Functions, highlighting key findings and limitations. This comprehensive review serves as a guide for researchers, directing them toward new and promising research directions in the field.

Keywords: Serverless computing, performance metrics, performance modeling, energy optimization, serverless platforms

1 INTRODUCTION

Serverless computing represents an emerging paradigm in cloud computing used to deliver applications and services. This innovative approach involves executing small code snippets in the cloud without managing the underlying resources on which the code operates. Despite not eliminating the existence of servers, serverless computing shifts operational tasks, such as scalability, fault tolerance, maintenance, monitoring, and resource provisioning, to the cloud providers [1]. For the underlying infrastructure of cloud service providers, serverless computing also shifts the whole workload toward cloud vendors [2] and rapidly gains the attention of academics and IT practitioners. Serverless computing is an emerging cloud computing model that provides a platform to efficiently develop applications and bring them to market without managing the underlying infrastructure [3].

Serverless computing differs from traditional cloud computing because the infrastructure and platform on which the program runs are hidden from the users. In this way, users only have to do what their applications need, and the rest is left to the service provider [4]. There are some benefits of using serverless computing compared to cloud computing, such as cost savings, scalability, energy efficiency, ease of application development, and better resource utilization, but the rise of serverless computing has introduced some performance-related issues [5]. Unlike virtual machines and containers, serverless scenarios have a faster startup time but may still suffer from unpredictable and low performance [6].

Performance models addressed various performance-related issues in serverless computing. The performance modeling in serverless computing applications ensures that the cost and performance metrics of the workload remain within an acceptable range, thereby improving the quality of service [7].

1.1 Distinguishing Cloud Computing and Serverless Paradigms

Cloud computing is the traditional go-to solution for providing high performance and managing demanding tasks. Cloud computing is known to be reliable and has various options for delivering better user experiences. On the other hand, serverless computing is the cloud technology that uses a network of remote servers to host and manage data rather than a local server. Serverless computing refers to the application of providing backend services on a use-per basis [8]. Table 1 depicts the comparison between cloud computing and serverless computing. The companies using serverless backend services are charged based on usage rather than the number of servers or a fixed bandwidth. The term serverless refers to a cloud service that hides (or abstracts) the features of the cloud-based processor from the user. Serverless does not imply that servers are not required; it simply means that they are not defined or controlled by the user. In response to a request from the application, serverless delivers exact units of resources. In traditional cloud computing, resources must be allotted in advance to be available when needed [9]. Figure 1 compares cloud computing and serverless computing.

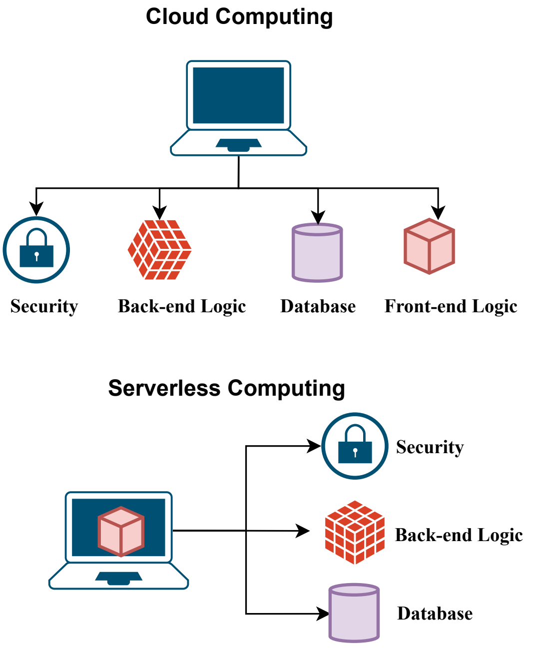

Figure 1. Comparative analysis of cloud computing and serverless computing architectures

1.2 Motivation and Our Contribution

The research motivation for this paper is outlined as follows:

- Recent studies have revealed that no surveys have been conducted to explore all

Factors	**Cloud Computing**	**Serverless Computing**
Autoscaling	Unavailable	Available
Server management	Required	Unavailable
Security	Less secure	More secure
Load balancing	Manual	Cloud provider handles load balancing
Cost	Expensive	Reduced cost
Availability	Low	High
Implementation stage	Difficult	Easy
Complexity	High	Less
Debugging	Easy	Difficult
Appropriate user	Administrator and developer	Developer

Table 1. Comparison between cloud computing and serverless computing

performance parameters [10, 11]. This indicates a critical need to investigate all performance parameters on a single platform.

- Some of the authors have published reviews on performance modeling in serverless computing [6, 7, 12]. However, various performance models have yet to be investigated by addressing the performance parameters.
- None of the authors have discussed the need for energy optimization in serverless computing in their survey, and have not addressed the optimization techniques. Hence, exploring optimization techniques for energy efficiency is required [13, 14].
- Software platforms used in serverless computing are essential to explore [15, 16].

The novel contributions of the review paper are also elaborated below:

- A comprehensive survey has been conducted to examine the existing literature in serverless computing. A comparative analysis was performed, evaluating cloud computing and serverless computing in terms of common factors.
- The survey has been explored based on various performance metrics used in serverless computing, including cost, scalability, latency, energy consumption, resource utilization, fault tolerance, and response time.
- The present review study has discussed in detail various modeling techniques for enhancing performance in serverless computing based on the above metrics.
- The current review article has explored several energy optimization strategies to reduce energy consumption and improve system efficiency.
- The investigation has been done on the software platforms used for implementing serverless computing, including AWS Lambda, Apache OpenWhisk, Azure Functions, and Google Cloud Functions, along with the key findings and limitations.
- The complete review helps guide researchers toward new and promising research directions.

1.3 Related Surveys and Our Work

The most suitable studies published on serverless computing are briefly presented here. The authors in [17] and [18] discussed the evolution of serverless computing. The implementation of serverless computing is not limited to the enhancement of infrastructure but is also employed for big data [19], video processing [20], and neural network training [21]. The authors in [22] covered white and grey literature. The paper [23] presented four use cases of FaaS and compared three serverless computing platforms: AWS Lambda, Azure Functions, and Google Cloud Functions. The authors in [24] evaluated FaaS platforms and performance features for micro-benchmarks, benchmark types, and other standard features. They presented function triggers, language runtimes, and external services. The authors in [25] modified or developed serverless tools and platforms and identified the challenges. The authors in [26] covered the emergence of serverless along with limitations such as inadequate performance, lack of coordination in functions, limited storage, and functional performance. Also, they identified the difference between AWS serverless and AWS server and five application areas that are suitable for serverless computing.

After completing an analysis of the existing surveys, it has been noticed that there is a need to analyze performance enhancement and energy optimization in serverless computing, which is included in this survey. This survey summarizes the comparison of platforms based on common characteristics and combines the existing research on serverless computing, and is an enhancement of existing surveys. Table 2 summarizes the comparative study of the existing surveys with the proposed survey in serverless computing.

Ref.	**Year**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
[17]	2018	✓	✗	✗	✗	✗	✗	✗
[18]	2018	✓	✗	✗	✗	✗	✗	✗
[10]	2021	✓	✓	✓	✗	✗	✓	✓
[27]	2019	✓	✗	✗	✗	✗	✗	✓
[4]	2022	✓	✗	✓	✗	✓	✗	✓
[11]	2020	✓	✗	✗	✗	✗	✗	✗
[23]	2020	✗	✗	✗	✗	✗	✓	✗
[24]	2020	✓	✓	✓	✗	✗	✗	✗
[1]	2017	✓	✗	✗	✗	✗	✓	✓
[28]	2022	✓	✗	✓	✓	✗	✗	✓
[29]	2022	✓	✗	✓	✓	✗	✓	✓
[30]	2023	✓	✓	✓	✓	✗	✓	✓
Our Survey		✓	✓	✓	✓	✓	✓	✓

1 – Serverless Computing, 2 – Cloud Computing vs. Serverless Computing,
3 – Performance Metrics, 4 – Performance Enhancement, 5 – Energy Optimization,
6 – Platforms in Serverless Computing, 7 – Research Directions

Table 2. Comparison of existing surveys with our survey

1.4 Structure of the Survey Paper

The survey paper has been organized into the following sections as shown in Figure 2. In Section 2, several research questions and the review methods are discussed. Section 3 explores the performance metrics addressed in serverless computing. Section 4 conducts a systematic review of performance modeling in serverless computing. Section 5 focuses on measuring energy optimization in serverless computing. Section 6 presents and compares various existing platforms used for serverless computing. Section 7 summarizes the review with potential gaps and future research directions. Finally, Section 8 concludes the review and provides recommendations for future research.

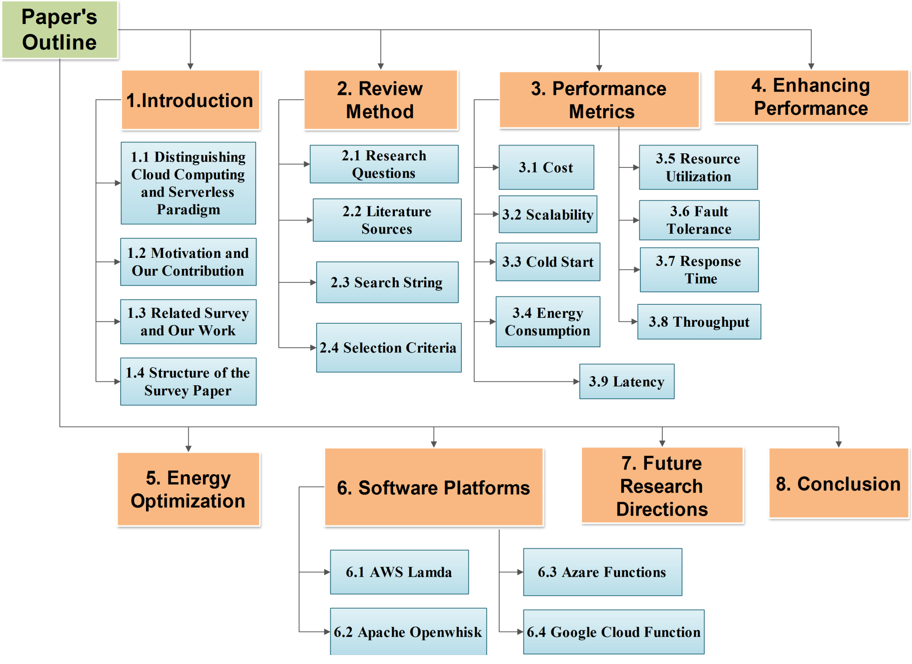

Figure 2. Outline of the paper

2 REVIEW METHOD

The systematic review was conducted with relevant articles on serverless computing. To provide a systematic, transparent, and understandable review of the paper, multiple journals, articles, and sites were visited for the various applications of serverless computing. The main objective of a systematic review is to write an article to understand, to find a good piece of information after reviewing, to identify the problem, to repeat, or to distinguish between research. Various magazines, digital libraries, and websites are accessed to find relevant articles.

2.1 Research Questions

To determine the scope of the systematic literature review, various research questions were formulated, as shown in Table 3.

2.2 Literature Sources

In this review, various search platforms are used as sources of literature presented in Table 4

ID	Review Questions	Section
RQ1	What are the various performance metrics used in serverless computing?	Section 3
RQ2	How can performance in serverless computing be enhanced based on these performance metrics?	Section 4
RQ3	What are the possible measures for optimum energy use in serverless computing?	Section 5
RQ4	What software tools and platforms are used to implement serverless computing?	Section 6
RQ5	What are the gaps and future research directions in serverless computing?	Section 7

Table 3. Review questions

Source	URL
IEEE	`https://ieeexplore.ieee.org`
Springer	`https://link.springer.com`
Elsevier ScienceDirect	`https://www.sciencedirect.com`
ACM	`https://dl.acm.org`

Table 4. Sources of knowledge

2.3 Search String

```
(\serverless" OR \function-as-a-service" OR \FaaS" ) AND (\computing"
OR \architecture" OR \model" OR \application" OR \tools"
OR \performance" OR \scalability" OR \energy" OR \platform"
OR \programming").
```

2.4 Selection Criteria

The study selection process followed in this study is shown in Figure 3 using a PRISMA-style flow diagram. Initially, 2 624 records were identified through database searching using relevant keywords. After removing 177 duplicate records, a total of 2 447 unique records were subjected to screening, which was conducted in three stages:

- First, titles were reviewed, and 557 irrelevant records were excluded.
- Next, 1 890 abstracts were assessed, leading to the exclusion of 242 additional records.
- Finally, the full text of the remaining 1 648 articles was evaluated for eligibility.

  During full-text screening, 226 articles were excluded for the following reasons:

- Not focused on serverless computing ($n = 120$).
- No energy optimization methodology ($n = 85$).

- Not peer-reviewed ($n = 50$).
- Other reasons, such as incomplete or duplicate content ($n = 53$).

A total of 93 studies were included in the final analysis. To determine whether the publication is suitable for the topic of this research, the inclusion and exclusion criteria were developed and applied as follows:

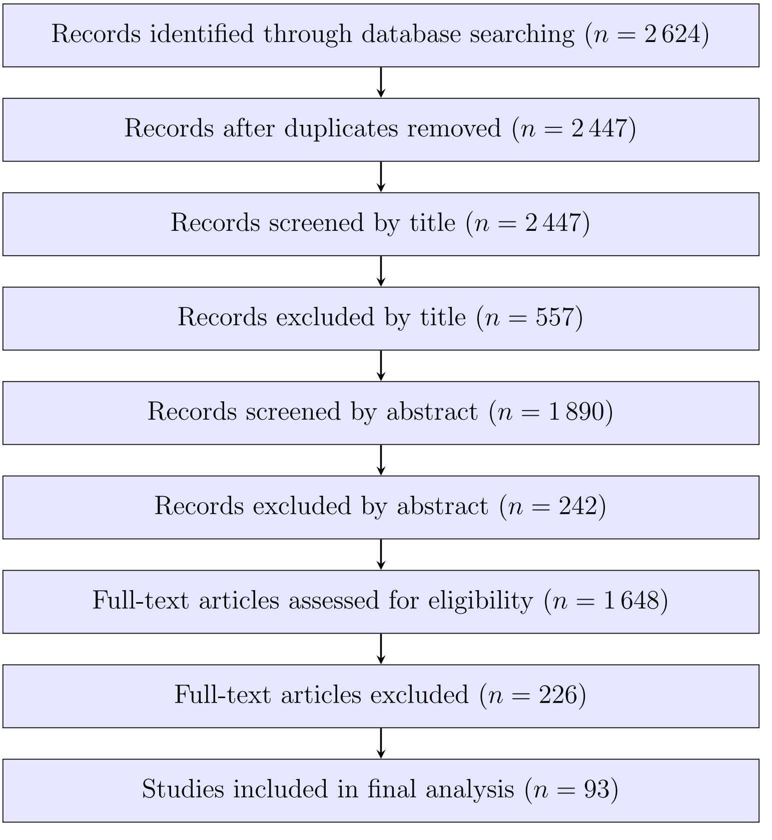

Figure 3. PRISMA flow diagram representing the selection process of studies for inclusion

Inclusion criteria:

- Articles published in peer-reviewed journals, conference proceedings, and articles published in reputed journals.
- Publications published online from 2016 to 2024.
- Articles that are written in notable journals in English.

Exclusion criteria:

- Publications not published in English.
- Publications that are duplicates of other previous publications.
- Publications without accessible full text.

3 PERFORMANCE METRICS: SERVERLESS COMPUTING (RQ1)

As serverless computing is gaining popularity in the modern world, researchers and practitioners have come across various performance metrics related to serverless computing. In the following, some of the most critical factors that can be used to demonstrate the performance of serverless computing will be discussed.

3.1 Cost

Cost is a fundamental parameter to consider. It helps reduce resource usage when a serverless function is idle and while it is executing. Another important factor is the pricing model, which includes comparisons to other cloud computing procedures. For example, serverless functions are currently less expensive for CPU-bound computations, whereas I/O-bound functions may be cheaper on dedicated containers and VMs [31].

3.2 Scalability

Serverless computing must provide operational scalability. For example, when there are many requests for a serverless application, these incoming requests need to be processed. The serverless provider must provide the necessary resources to execute all these requests by scaling up the resources [32].

3.3 Cold Start

Serverless computing has many performance issues; they need to be activated when called upon. This activation process takes some time and leads to a delay in executing applications, which is known as a cold start. So to improve performance, it is important to reduce the cold start by keeping the functions warm [33, 34].

3.4 Energy Consumption

Energy-aware scheduling is done to reduce energy consumption [35]. The main purpose of this type of scheduling is to put the execution environment or inactive containers in a cold state. The transformation from a cold state to active mode experiences delays in the execution of invoked functions, which may go beyond the time limit defined by the customer [4].

3.5 Resource Utilization

Serverless computing automatically scales resources and clarifies the evolution of online services with stateless functions. However, it is still significant for users to allocate relevant resources due to the numerous function types and input sizes. Lack of resource allocation management leaves functions either over-provisioned or under-provisioned and causes low resource utilization [36]. There is a need to efficiently increase the resource utilization for the provider while managing resources dynamically to improve function response times [37].

3.6 Fault Tolerance

In recent years, serverless computing has gained popularity with increasing applications built on Functions as a Service (FaaS) platforms. FaaS platforms encourage retry-based fault tolerance, which is insufficient for programs that change shared states [38].

3.7 Response Time

Response time is a crucial performance metric in serverless computing, measuring the time from when a client requests a serverless function to when the response is received. Optimizing response time ensures that users experience minimal delays when interacting with serverless applications, enhancing overall user satisfaction and experience. Factors such as function execution time, cold start latency, network latency, and workload fluctuations influenced response time. For maintaining the efficiency and reliability of serverless applications, there is a need to improve response time [39].

3.8 Throughput

Throughput refers to the rate at which serverless functions can process a specific volume of requests within a specified time. High throughput indicates that the serverless architecture can handle many concurrent requests efficiently. Optimizing function execution time, concurrency settings, resource allocation, and network performance can achieve optimal throughput. Monitoring throughput ensures that serverless applications can scale effectively to meet varying workloads and maintain consistent performance under heavy loads [40].

3.9 Latency

Serverless applications operate independently of a fixed server location; their code can run on any server. Therefore, cloud vendors can run the application on servers close to the end user's location. The end user requests do not have to travel across the Internet to access the original server, thereby decreasing latency [10].

Ref.	**Year**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
[40]	2021	✓	✗	✓	✗	✓	✗	✗	✓	✗
[41]	2019	✗	✓	✗	✗	✗	✗	✗	✗	✗
[39]	2020	✓	✗	✗	✗	✗	✗	✓	✗	✓
[42]	2020	✗	✓	✗	✗	✓	✗	✗	✗	✗
[43]	2018	✓	✗	✗	✗	✗	✗	✗	✓	✗
[4]	2022	✗	✗	✗	✓	✗	✗	✗	✗	✗
[44]	2021	✗	✓	✗	✗	✓	✗	✗	✗	✗
[45]	2022	✓	✓	✗	✓	✓	✗	✓	✗	✓
[46]	2021	✓	✓	✗	✗	✓	✗	✗	✗	✗
[28]	2022	✓	✓	✗	✗	✗	✗	✗	✗	✓
[47]	2020	✓	✓	✗	✗	✗	✗	✗	✗	✗
[48]	2021	✗	✗	✗	✗	✗	✗	✗	✗	✗
[49]	2022	✓	✓	✗	✗	✗	✗	✓	✗	✓
[50]	2022	✓	✗	✓	✗	✗	✗	✗	✗	✗
[25]	2019	✓	✗	✗	✗	✗	✓	✗	✗	✗
[17]	2018	✓	✗	✗	✗	✗	✗	✗	✗	✗
[31]	2020	✓	✗	✗	✗	✗	✗	✗	✗	✗
[51]	2018	✗	✗	✓	✗	✗	✗	✓	✗	✓
[43]	2018	✓	✗	✓	✗	✗	✗	✗	✗	✗
[52]	2018	✓	✓	✓	✗	✓	✓	✗	✗	✗
[53]	2020	✓	✗	✗	✗	✗	✗	✗	✗	✗
[54]	2022	✗	✗	✓	✗	✗	✗	✗	✗	✗
[55]	2023	✗	✗	✓	✗	✗	✗	✗	✗	✓
[56]	2021	✗	✗	✓	✗	✗	✗	✗	✗	✗
[57]	2023	✓	✗	✓	✗	✓	✗	✓	✓	✓
[58]	2023	✓	✓	✗	✗	✓	✗	✗	✗	✗
[59]	2024	✗	✗	✗	✓	✗	✗	✗	✗	✗
[60]	2019	✗	✗	✗	✗	✓	✗	✗	✗	✗
Proposed Survey		✓	✓	✓	✓	✓	✓	✓	✓	✓

1 – Cost, 2 – Scalability, 3 – Cold Start, 4 – Energy Consumption,
5 – Resource Utilization, 6 – Fault Tolerance, 7 – Response Time, 8 – Throughput,
9 – Latency

Table 5. Summary of the related works based on the performance parameters in serverless computing

Table 5 concluded that these parameters could significantly impact system performance. As per our literature review, some authors have considered specific metrics in their studies. Wen et al. [2] evaluated cost, cold start, and resource utilization. Perez et al. [61] considered scalability and resource utilization. Kim and Lee [39] examined cost, response time, and latency. These parameters have been researched, but some issues remain for further investigation. Section 4 identifies the existing performance metrics, then overviews the studies on them, and finally analyzes each performance metric for subsequent research.

4 ENHANCING PERFORMANCE: SERVERLESS COMPUTING (RQ2)

Serverless applications and Function-as-a-Service(FaaS) have gained popularity because of resource management, scalability, and a pay-as-you-go pricing model. In this paper, the prediction and optimization of cost and performance of serverless applications have been analyzed [62].

The authors in [7] proposed a performance model to improve serverless systems' resource usage and quality of service by lowering operational costs. The study confirmed the proposed model's applicability and correctness through extensive testing on AWS Lambda. It demonstrated that the proposed model can compute critical performance measures such as the steady state's average response time and number of function instances.

HotC is a container-based runtime management framework that develops lightweight containers to improve network performance and reduce cold start. The result indicated that HotC has a lesser overhead and improved performance [63].

A performance model is proposed by performing experimentation on AWS Lambda that can measure various performance parameters based on cold and warm query response time [6]. Several implementation issues, including reusability, lifecycle management, container discovery, and function scalability, are covered in depth. The result indicated that the proposed prototype achieves greater throughput than other platforms [64].

According to [65], latency can be within an acceptable range by extending delays caused by cold starts by breaking more strict SLAs. This paper analyzed the performance of serving deep learning models. In this finding, warm serverless function executions are acceptable regarding latency, but cold starts produce substantial overhead. In [66], FaaS platforms enable users to run random functions without being concerned about operational issues. However, there are several performance issues. By considering these issues, the author identified six performance challenges and presented a roadmap to solve them in the future.

The authors in [43] stated that applications have multiple independent functions that can be implemented in various programming languages. This paper explained the influence of the choice of language runtime on the performance and cost of serverless function execution. The authors analyzed cost and performance metrics for Azure Functions and AWS Lambda. For optimum cost management and performance of serverless applications, Python is a clear choice on AWS Lambda.

Serverless computing is gaining popularity among cloud providers. As a result, the Function-as-a-Service programming model boosts the popularity of stateless function calls to create a service. The existing technologies are suitable for data centers, but they cannot deliver the same level of performance in edge computing systems. The authors in [67] addressed the issue by offering a system for efficiently dispatching stateless tasks to network executors while maintaining short and long-term fairness. In [68], it is stated that disaggregating compute and storage services allows for an attractive separation of issues around autoscaling resources in a serverless environment. However, it introduced consistency and performance challenges for applications written on FaaS platforms. In this paper, HydroCache is presented, which is a distributed cache co-located with a FaaS compute layer that overcomes these limitations.

The authors in [69] stated that FaaS is a novel, but promising service model in cloud computing. The importance of FaaS can be seen in public service providers with their own FaaS infrastructures. Also, the open-source community makes the best efforts to implement FaaS initiatives. This paper showed the performance differences between Python 3, Fission Kubeless, Node.js of OpenFaaS, and Knative platforms. It also showed how the supported auto-scaling algorithms of the examined FaaS systems affect the performance of the function runtimes. Finally, it proposed solutions to increase the performance of the Python 3 runtime of Kubeless and OpenFaaS.

Table 6 evaluated the performance parameters along with the contribution of existing research, its results, and the scope of improvement in terms of performance. So to improve performance, there is a need to optimize energy consumption in serverless computing as discussed in Section 5

5 ENERGY OPTIMIZATION: SERVERLESS COMPUTING (RQ3)

Autoscaling always needs to make a deal between optimizing for cost-allocated resources and optimizing for application performance [71]. Serverless platforms are designed to respond to requests by offloading processing to edge nodes quickly [67]. Over the last 10 years, data center energy consumption has only grown by 6 % despite an increase in usage [72]. The power draw is loosely correlated to the CPU load, although this has been improving in recent years. Even so, the utilization of servers is poor – only 50 % in the best hyper-scale facilities [73].

The authors in [74] explained the efficiency of the serverless computing paradigm. The survey aimed to extend the internal mechanics of serverless computing and explore the scope for efficiency within the paradigm by studying approximation approaches and function reuse. From the analysis, it was visualized that the future generation of highly scalable applications will mostly rely on the serverless computing paradigm, identifying the extent of efficiency that could bring significant benefits to the providers, developers, and users. The authors in [75] described the energy-aware resource scheduling for serverless edge computing. The authors evaluated the well-known benchmarks using real-world implementations on a Raspberry Pi. Experimental results achieved outstanding improvements of up to 33 % in helping the bottleneck node's operational availability while preserving the quality of service. Serverless can unconditionally offer its portability and resource efficiency at the edge with energy awareness. Decentralization of the scheduler was essential to cover the mobile edge computing area. The authors in [76] explained the energy-efficient serverless on bare-metal single-board computers. Systematically designed implementation of MicroFaaS was presented, and a thorough

Author [Ref.]	**Parameters**	**Technique**	**Contribution**	**Results**	**Future Scope**
Lin and Khazaei [62]	Cost Scalability	Analytical Model	Analysis of serverless system's performance, usage, and cost	Accurate analytical model and scalability demonstration	Addressing cold start latency and communication overhead
Mahmoudi and Khazaei [7]	Cost Cold start Response Time Resource Utilization	Analytical Model	Improved quality of service and cost-effectiveness of serverless platforms	Predicted application's cost/performance and achieved savings in cost and energy	Enhancement in performance, cost, and energy efficiency
Suo et al. [63]	Cold start	Exponential Smoothing Model Markov Chain Model	Mitigation of cold start and improved network performance	Reduced overhead and improved performance.	Execution time reduction.
Mahmoudi and Khazaei [6]	Cold start Response time	Analytical Model	Prediction of performance metrics for improved quality of service	Preemptive workload handling, diverse function instances, and improved scaling strategies	Enhanced services and scalability
McGrath and Brenner [64]	Scalability Throughput	Performance Evaluation	Evaluation of serverless platform's execution performance	Greater throughput and scaling trends were observed	Improving serverless platform quality and maximizing potential
Ishakian et al. [65]	Cold start Latency	Deep Learning	Assessment of serverless computing for large neural network models	Impact of cold starts on latency distribution and SLA risks	Addressing cost and memory allocation issues
Van Eyk et al. [66]	Cost	SPEC RG Cloud Group	Identification of performance-related challenges	Plotting a roadmap for overcoming performance issues	Addressing new performance-related challenges
Jackson and Clynch [43]	Cost	Performance Testing Framework	Analysis of cost and performance metrics for AWS Lambda and Azure Functions	Identification of Python as the optimum choice for AWS Lambda	Developing cost-effective solutions
Cicconetti et al. [67]	Response time	Efficient Dispatching	Efficient dispatching of tasks to minimize response time	Mobility and service request pattern variations observed	Long-term allocation improvements
Wu et al. [68]	Consistency	Distributed Cache	Mitigation of performance and consistency challenges	Significant performance improvements and consistency protection	Dynamic scheduling and metadata management improvements
Balla et al. [69]	Scalability Consistency	Auto-scaling Algorithms	Influence of auto-scaling algorithms on function runtimes	Performance enhancements for specific runtimes.	Improving runtime performance further
Khatri et al. [70]	Cost	Machine Learning	Identification of bottlenecks and performance measurement scope	Areas of improvement identified with performance measurement	Leveraging AI/ML for improved performance

Table 6. Summary of the related works on enhancing performance in serverless computing

evaluation and cost analysis were conducted. Results showed a 5.6 $\times$ increase in energy efficiency and a 34.2 % decrease in the total cost of ownership compared to the baseline. The MicroFaaS cluster was 32.5–34.2 % cheaper than a conventional cluster with equivalent throughput. The node was put into a low-energy sleep state if the computational capacity offered by a node was not required at any given time.

The authors in [77] presented energy consumption as a significant challenge in the green cloud environment, because of which the Dynamic Voltage Frequency Scaling (DVFS) scheduling strategy is the most promising. DVFS saved energy by lowering the processor frequency for virtual machines (VMs), which increases errors during workflow execution, thus decreasing the system's reliability. As a result, this article addressed the DVFS issue by providing a novel Smart Energy and Reliability Aware Scheduling algorithm (SERAS) for cloud-based workflow execution. The SERAS technique divided the workflow's target deadline into tasks. The suggested algorithm used the DVFS technique to reduce the frequency of processors for VMs without violating task deadlines. As a result, the SERAS algorithm assigned jobs to the most relevant VMs with the necessary frequency levels while ensuring the green cloud system's reliability and completion time requirements. The SERAS algorithm outperforms its competitors while meeting the required dependability and completion time levels.

The authors in [13] stated that energy consumption is one of the fundamental design requirements for heterogeneous distributed systems. Numerous algorithms are used to study the problem of minimizing the energy consumption of a real-time parallel application. This study used combined global DVFS-enabled and non-DVFS energy-efficient scheduling algorithms. In [4], the authors presented energy-aware scheduling, and the main idea in this type of scheduling is to put the inactive containers or execution environment in a cold-state mode to reduce energy consumption. In [78], the authors introduced FaaS to heterogeneous computing and supports heterogeneous platforms, i.e., FDN (Function Delivery). FDN offered energy efficiency and Service Level Objective (SLO) requirements. The authors in [79] optimized energy consumption by dynamic consolidation of Virtual Machines (VMs) using live migration of the VMs and switching idle servers to sleep mode or shutdown. Table 7 depicts the energy optimization analysis in serverless computing.

To conduct performance enhancement and energy optimization analysis, researchers have access to open-source platforms in serverless computing, as mentioned in Section 6.

6 SOFTWARE PLATFORMS: IMPLEMENTATION OF SERVERLESS COMPUTING (RQ4)

Serverless computing can simplify application deployment and thus alleviate developers' efforts from tedious and error-prone server management. Various commodity

Author [Ref.]	**Parameters**	**Technique**	**Contribution**	**Results**	**Future Scope**
Aslanpour et al. [75]	Energy Throughput	Energy Aware Scheduling	Reduced energy consumption without overhead Evaluated on Raspberry Pis	Up to 33 % improvement in node availability	Need to address scalability
Kallam et al. [80]	Energy Time	Linear Optimization	Reduced energy consumption Lower execution time	Reduced energy by 16 % and execution time by 20 %	Integrate energy efficiency Consider data distribution
Gunasekaran et al. [14]	Energy Latency	Optimization	Reduced response latency Improved container utilization	31 % reduction in energy consumption	Need for comparison
Aslanpour et al. [81]	Energy Cost	Energy Modeling	Enhanced consumption efficiency Validated in Smart Agriculture	95 % accuracy in energy model	Address edge computing needs Consider renewable sources
Hassan et al. [77]	Energy Time	Scheduling	Smart Energy and Reliability Aware Scheduling algorithm for workflow execution in the cloud environment.	Improved reliability	Need to enhance energy consumption Consider additional metrics
Xie et al. [13]	Energy Scalability	Optimization	Minimized energy usage	Saved 36.25–55.65 % of energy	Minimize energy consumption.
Shafiei et al. [4]	Energy Resource Utilization	–	Comprehensive review Classified applications	Overview of advancements	Focus on security, privacy, and cost prediction
Jindal et al. [78]	Energy Response Time	FDN	Introduced Function Delivery Network	Improved performance	Expand to other heterogeneous computing devices
Denninnart and Salehi [74]	Energy Cost	Efficiency	Improved efficiency by function reuse and approximation approaches	identified scope for improvements	Maintaining all the functions in memory for the large-scale serverless cloud
Byrne et al. [76]	Energy Cost	MicroFaaS	Energy-efficient serverless on single-board computers	Increased energy efficiency and reduced cost	Consider low-energy sleep state

Table 7. Summary of related works on energy optimization in serverless computing

serverless platforms, including AWS Lambda, Google Cloud Functions, Azure Functions, and Apache OpenWhisk Compute, have emerged [82]. These commodity serverless computing platforms frequently act in a black-box fashion, and developers do not need to pay attention to the underlying implementation details [2]. Different companies have already started combining the power of edge with the operational simplicity of serverless, providing edge platforms for deploying serverless functions [83]. Different scenarios make it challenging for a service developer to differentiate and select the proper serverless platform [84]. Figure 4 shows different platforms that are used in serverless computing.

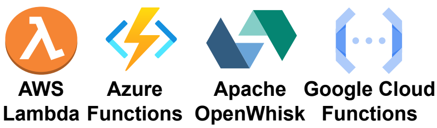

Figure 4. Serverless platforms

6.1 AWS Lambda

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. Lambda is named after functions from the lambda calculus and programming. Those functions act as a good analogy for the service.

The author in [15] explained the analysis of serverless computing techniques in the cloud software framework, in which AWS Lambda and Azure platforms were used. The user gets access to the serverless model through a mobile phone, the HTTP request is passed through the domain name server routing, and the request outcome is provided through the content delivery network, which communicates to the object store. Serverless cloud computing includes specific challenges, such as a process that takes a long time to run. The authors in [16] explained the framework and a performance assessment for serverless map-reduce on AWS Lambda in which HyperFlow and AWS Lambda platforms were used. The results indicated that AWS Lambda provided a convenient computing platform for general-purpose applications that fit within the constraints of the service (3 008 MB of RAM, 512 MB of disk space, and 15 minutes of maximum execution time). Architecture did not fit in the Lambda memory (maximum of 1 536 MB at that time), and they did not proceed to compute the final output.

6.2 Apache OpenWhisk

Apache OpenWhisk is an open-source and serverless cloud platform that performs functions responding to events. The platform used a function-as-a-service (FaaS) model to manage infrastructure and servers for cloud-based applications.

The authors in [85] explained the distributed analysis and benchmarking framework for the Apache OpenWhisk serverless platform. OpenWhisk functions are written in JavaScript and Java, compared to the Spring web-based application, which executes the same function. The analysis indicated that the latency of the OpenWhisk functions had increased the number of requests compared with the spring-based application. The automatic scaling recommended by OpenWhisk was not predictable by the user, which can cause latency bottlenecks [86]. The results of each experiment showed that OpenWhisk could outperform a solution that employed the same functionality through container-based virtualization. It also demonstrated how close Open Whisk was performance-wise to being a more outstanding solution that did not suffer from the overheads of virtualization. The cold start problem arose and highlighted the impact of the choice of language runtime [87].

6.3 Azure Functions

Microsoft Azure, formerly known as Windows Azure, is Microsoft's public cloud computing platform. It provides a range of cloud services including computing, analytics, storage, and networking. The authors in [88] explored Azure Functions and showed how to set up the development environment and then develop a simple program with Azure Functions.

6.4 Google Cloud Functions

Google Cloud Functions is a serverless execution environment for connecting and building cloud services. Simple single-purpose functions attached to events emitted from cloud infrastructure and services can be written with cloud functions. The function is triggered when an event being watched is fired. The authors in [89] explained the efficient processing of latency-sensitive serverless DAGs at the edge of the Google Cloud functions. From the results, the earliest deadline first (EDF) achieved better deadline miss rates than SRSF for DAGs with smaller inputs of 5 KB and 40 KB, and performance gets very close for DAGs with inputs of 105 KB. For DAG functions, each sub-function shares the same deadline. The EDF order was based on the deadlines of the tasks, i.e., it did not consider the function's execution time.

The serverless execution of scientific workflows with experiments using Google Cloud functions is described. Prototype workflow executor functions using Google Cloud Functions are developed and coupled with the HyperFlow workflow engine. Findings indicated that the simple mode of operation makes this approach easy to use, although there were costs involved in preparing portable application binaries for execution in a remote environment. There was a need to develop custom binaries or execution time limits [90]. The authors in [91] described the fast provisioning and scalable custom serverless container runtimes at Alibaba Cloud Function Compute. Evaluation results showed that FAASNET finished provisioning 2 500 function containers on 1 000 virtual machines in 8.3 seconds, scales 13.4× and 16.3 × faster than Alibaba Cloud's current FaaS platform. Solutions cannot fundamentally solve the high costs incurred during function environment provisioning. The comparative analysis of existing platforms for serverless computing is given below in Table 8

7 FUTURE RESEARCH DIRECTIONS: SERVERLESS COMPUTING (RQ5)

Serverless computing is an innovative concept that simplifies the development of applications globally. However, a literature review revealed certain gaps that have not been adequately recognized by researchers. Recent studies have identified various challenges that serverless computing faces, as illustrated in Figure 5.

Figure 5. Serverless computing future research directions

7.1 Addressing the Cold Start Problem in Serverless Computing

The cold start problem remains a significant challenge in serverless computing, causing delayed response times for users due to the initialization of functions. However, resolving this issue without compromising the primary features of serverless architecture is essential. There is a need to explore innovative solutions that mitigate cold start delays while preserving the primary features of serverless, such as scalability and cost-effectiveness. By investigating techniques such as container reuse or pre-warming, researchers can enhance user experience without sacrificing the inherent advantages of serverless computing [6, 7, 62, 63, 65].

Author [Ref.]	**Parameters**	**Platforms**	**Findings**	**Limitations**
Andi [15]	Time Cost	AWS Lambda Azure Functions Google Cloud	No limited functionality in Azure and AWS Lambda Google Cloud had a limit of 1 000 per project.	Processes are taking a long time to run.
Kuntsevich et al. [85]	Scalability Cold start	Apache OpenWhisk	Latency increased compared to spring-based applications due to OpenWhisk functions.	Automatic scaling by OpenWhisk is causing unexpected latency bottlenecks.
Perez et al. [92]	Cost Throughput	AWS Lambda Azure Functions OpenWhisk	AWS provided 1 million invocations or 400,000 GB-seconds free per month.	Lambda is not a significant drain on infrastructure yet.
Malawski et al. [90]	Scalability Cost	AWS Lambda HyperFlow Google Cloud Functions	Simple mode of operation Costs involved in preparing portable application binaries	Need for creation of execution time limits or custom binaries.
Gimenez et al. [16]	Throughput Time	AWS Lambda	Specifications: 3 008 MB RAM, 512 MB disk space, 15 minutes maximum execution time.	Architecture is not fitting Lambda memory Failure to compute the final result.
Wang et al. [91]	Scalability Cost	Alibaba Cloud Function	FAASNET provisioned 2 500 function containers on 1 000 virtual machines in 8.3 seconds	High costs incurred during function environment provisioning
Lyu et al. [89]	Latency	Google Cloud Functions	EDF achieved higher deadline miss rates than SRSF for smaller DAG sizes Performance closer for larger DAG sizes	Function execution time

Table 8. Summary of related works on targeted software platforms for serverless computing

7.2 Energy-Efficient System Design in Serverless Computing

An efficient technique, such as dynamic resource allocation, workload consolidation, and power-aware scheduling, can be developed that reduces energy consumption while keeping the reliability requirement of the system [7, 77]. The methods need to be developed for energy-aware scheduling to help delay non-latency-sensitive tasks to reduce overall energy consumption [4].

7.3 Enhanced Quality of Service Management for Serverless Applications

Efficient resource allocation and quality of service (QoS) management are essential in ensuring optimal performance in serverless environments. Auto-scaling mechanisms must be developed to effectively manage function resources without affecting costs or fault tolerance. By implementing workload prediction and resource provisioning algorithms, researchers can maintain high QoS standards while mitigating operational costs and enhancing fault tolerance [10]. Different degrees of QoS will be evaluated for stateful serverless applications, as current serverless platforms are mostly stateless [1].

7.4 Legacy System Migration to Function-as-a-Service (FaaS)

Researchers are working on the open question of how to decompose legacy systems into FaaS without degrading performance. Finding optimal automatic migration solutions for legacy systems is an interesting research direction [1]. Moreover, research on tools for checking whether a legacy system will fit the serverless paradigm is crucial. Also, developing and enhancing automatic and semi-automatic analysis strategies based on artificial intelligence could be another future research field [10].

7.5 Development of Performance Models for Workload Optimization

Developing autonomous middleware for workload optimization is one of the research challenges in serverless computing. This middleware will incorporate preemptive workload handling, support heterogeneous function instances, and integrate both FaaS and IaaS paradigms. Additionally, expanding performance models by using various auto-scaling patterns will enhance overall performance management in serverless computing [6].

7.6 Performance Enhancement

In future research, enhancing performance across various dimensions such as scalability, cost-effectiveness, energy efficiency, mitigation of cold start issues, fault tolerance mechanisms, and optimizing resource utilization are promising directions. Exploring various approaches and technologies to comprehensively address these aspects can greatly enhance the advancement of the field [7, 43, 66, 70, 93].

7.7 Emerging Technologies and Approaches in Serverless Computing

Recent advancements are defining the future of serverless computing through new technologies and methodologies focused on performance and energy optimization. One notable direction is the integration of edge computing with serverless architectures [89], enabling reduced latency and energy consumption by executing functions closer to the data source. Platforms such as AWS Greengrass and Cloudflare Workers support these capabilities.

Another innovation involves lightweight virtualization technologies like Firecracker microVMs, which enable faster startup times and better resource isolation. Similarly, WebAssembly (WASM)-based serverless runtimes are emerging as efficient and secure alternatives for function execution. Moreover, AI-driven autoscaling and scheduling, particularly using deep reinforcement learning [54], is being explored to optimize function invocation, reduce cold start, and balance workloads dynamically. Finally, tools like Knative and OpenFaaS provide enhanced orchestration and hybrid deployment options, marking a shift toward more flexible and intelligent serverless ecosystems.These emerging technologies represent promising paths toward addressing key challenges in serverless environments and should be explored further in both academia and industry.

8 CONCLUSION

In this paper, a comprehensive review has been conducted to study specific performance metrics related to serverless computing. Based on these parameters, an analysis has been carried out to enhance performance and optimize energy consumption in serverless computing. Researchers have evaluated open-source platforms to analyze performance enhancement and energy optimization comprehensively. The paper concludes with suggestions for future directions.

REFERENCES

[1] Baldini, I.—Castro, P.—Chang, K.—Cheng, P.—Fink, S.—Ishakian, V.—Mitchell, N.—Muthusamy, V.—Rabbah, R.—Slominski, A.—Suter, P.: Serverless Computing: Current Trends and Open Problems. In: Chaudhary, S., Somani, G., Buyya, R. (Eds.): Research Advances in Cloud Computing. Springer Singapore, 2017, pp. 1–20, doi: 10.1007/978-981-10-5026-8_1.

[2] Wen, J.—Liu, Y.—Chen, Z.—Ma, Y.—Wang, H.—Liu, X.: Understanding Characteristics of Commodity Serverless Computing Platforms. CoRR, 2020, doi: 10.48550/arXiv.2012.00992.

[3] Lloyd, W.—Ramesh, S.—Chinthalapati, S.—Ly, L.—Pallickara, S.: Serverless Computing: An Investigation of Factors Influencing Microservice Performance. 2018 IEEE International Conference on Cloud Engineering (IC2E), 2018, pp. 159–169, doi: 10.1109/IC2E.2018.00039.

[4] Shafiei, H.—Khonsari, A.—Mousavi, P.: Serverless Computing: A Survey of Opportunities, Challenges, and Applications. ACM Computing Surveys, Vol. 54, 2022, No. 11s, Art. No. 239, doi: 10.1145/3510611.

[5] Wu, M.—Mi, Z.—Xia, Y.: A Survey on Serverless Computing and Its Implications for JointCloud Computing. 2020 IEEE International Conference on Joint Cloud Computing, 2020, pp. 94–101, doi: 10.1109/JCC49151.2020.00023.

[6] Mahmoudi, N.—Khazaei, H.: Temporal Performance Modelling of Serverless Computing Platforms. Proceedings of the 2020 Sixth International Workshop on Serverless Computing (WoSC '20), 2020, pp. 1–6, doi: 10.1145/3429880.3430092.

[7] Mahmoudi, N.—Khazaei, H.: Performance Modeling of Serverless Computing Platforms. IEEE Transactions on Cloud Computing, Vol. 10, 2022, No. 4, pp. 2834–2847, doi: 10.1109/TCC.2020.3033373.

[8] Rahman, M. M.—Hasan Hasibul, M.: Serverless Architecture for Big Data Analytics. 2019 Global Conference for Advancement in Technology (GCAT), IEEE, 2019, pp. 1–5, doi: 10.1109/GCAT47503.2019.8978443.

[9] Ghosh, B. C.—Addya, S. K.—Somy, N. B.—Nath, S. B.—Chakraborty, S.—Ghosh, S. K.: Caching Techniques to Improve Latency in Serverless Architectures. 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), IEEE, 2020, pp. 666–669, doi: 10.1109/COMSNETS48256.2020.9027427.

[10] Hassan, H. B.—Barakat, S. A.—Sarhan, Q. I.: Survey on Serverless Computing. Journal of Cloud Computing, Vol. 10, 2021, No. 1, Art. No. 39, doi: 10.1186/s13677-021-00253-7.

[11] Taibi, D.—Spillner, J.—Wawruch, K.: Serverless Computing – Where Are We Now, and Where Are We Heading? IEEE Software, Vol. 38, 2021, No. 1, pp. 25–31, doi: 10.1109/MS.2020.3028708.

[12] Mahmoudi, N.—Khazaei, H.: SimFaaS: A Performance Simulator for Serverless Computing Platforms. CoRR, 2021, doi: 10.48550/arXiv.2102.08904.

[13] Xie, G.—Zeng, G.—Xiao, X.—Li, R.—Li, K.: Energy-Efficient Scheduling Algorithms for Real-Time Parallel Applications on Heterogeneous Distributed Embedded Systems. IEEE Transactions on Parallel and Distributed Systems, Vol. 28, 2017, No. 12, pp. 3426–3442, doi: 10.1109/TPDS.2017.2730876.

[14] Gunasekaran, J. R.—Thinakaran, P.—Chidambaram, N.—Kandemir, M. T.—Das, C. R.: Fifer: Tackling Underutilization in the Serverless Era. CoRR, 2020, doi: 10.48550/arXiv.2008.12819.

[15] Andi, H. K.: Analysis of Serverless Computing Techniques in Cloud Software Framework. Journal of IoT in Social, Mobile, Analytics, and Cloud, Vol. 3, 2021, No. 3, pp. 221–234.

[16] Giménez-Alventosa, V.—Moltó, G.—Caballer, M.: A Framework and a Performance Assessment for Serverless MapReduce on AWS Lambda. Future Generation Computer Systems, Vol. 97, 2019, pp. 259–274, doi: 10.1016/j.future.2019.02.057.

[17] Van Eyk, E.—Toader, L.—Talluri, S.—Versluis, L.—Uță, A.—Iosup, A.: Serverless Is More: From PaaS to Present Cloud Computing. IEEE Internet Computing, Vol. 22, 2018, No. 5, pp. 8–17, doi: 10.1109/MIC.2018.053681358.

[18] Sewak, M.—Singh, S.: Winning in the Era of Serverless Computing and Function as a Service. 2018 3rd International Conference for Convergence in Technology (I2CT), 2018, pp. 1–5, doi: 10.1109/I2CT.2018.8529465.

[19] Werner, S.—Kuhlenkamp, J.—Klems, M.—Müller, J.—Tai, S.: Serverless Big Data Processing Using Matrix Multiplication as Example. 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 358–365, doi: 10.1109/BigData.2018.8622362.

[20] Ao, L.—Izhikevich, L.—Voelker, G. M.—Porter, G.: Sprocket: A Serverless Video Processing Framework. Proceedings of the ACM Symposium on Cloud Computing (SoCC '18), 2018, pp. 263–274, doi: 10.1145/3267809.3267815.

[21] Feng, L.—Kudva, P.—Da Silva, D.—Hu, J.: Exploring Serverless Computing for Neural Network Training. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 334–341, doi: 10.1109/CLOUD.2018.00049.

[22] Taibi, D.—El Ioini, N.—Pahl, C.—Niederkofler, J. R. S.: Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review. In: Ferguson, D., Helfert, M., Pahl, C. (Eds.): Proceedings of the 10th International Conference on Cloud Computing and Services Science CLOSER – Volume 1. SciTePress, 2020, pp. 181–192, doi: 10.5220/0009578501810192.

[23] Rajan, A. P.: A Review on Serverless Architectures – Function as a Service (FaaS) in Cloud Computing. TELKOMNIKA (Telecommunication, Computing, Electronics and Control), Vol. 18, 2020, No. 1, pp. 530–537, doi: 10.12928/telkomnika.v18i1.12169.

[24] Scheuner, J.—Leitner, P.: Function-as-a-Service Performance Evaluation: A Multivocal Literature Review. Journal of Systems and Software, Vol. 170, 2020, Art. No. 110708, doi: 10.1016/j.jss.2020.110708.

[25] Yussupov, V.—Breitenbücher, U.—Leymann, F.—Wurster, M.: A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools. Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19), 2019, pp. 229–240, doi: 10.1145/3344341.3368803.

[26] Jonas, E.—Schleier-Smith, J.—Sreekanti, V.—Tsai, C. C.—Khandelwal, A.—Pu, Q.—Shankar, V. et al.: Cloud Programming Simplified: A Berkeley View on Serverless Computing. CoRR, 2019, doi: 10.48550/arXiv.1902.03383.

[27] Castro, P.—Ishakian, V.—Muthusamy, V.—Slominski, A.: The Rise of Serverless Computing. Communications of the ACM, Vol. 62, 2019, No. 12, pp. 44–54, doi: 10.1145/3368454.

[28] Li, Z.—Guo, L.—Cheng, J.—Chen, Q.—He, B.—Guo, M.: The Serverless Computing Survey: A Technical Primer for Design Architecture. ACM Computing Surveys (CSUR), Vol. 54, 2022, No. 10s, Art. No. 220, doi: 10.1145/3508360.

[29] Cassel, G. A. S.—Rodrigues, V. F.—da Rosa Righi, R.—Bez, M. R.—Nepomuceno, A. C.—da Costa, C. A.: Serverless Computing for Internet of Things: A Systematic Literature Review. Future Generation Computer Systems, Vol. 128, 2022, pp. 299–316, doi: 10.1016/j.future.2021.10.020.

[30] Wen, J.—Chen, Z.—Jin, X.—Liu, X.: Rise of the Planet of Serverless Computing:

A Systematic Review. ACM Transactions on Software Engineering and Methodology, Vol. 32, 2023, No. 5, Art. No. 131, doi: 10.1145/3579643.

[31] Cordingly, R.—Shu, W.—Lloyd, W. J.: Predicting Performance and Cost of Serverless Computing Functions with SAAF. 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020, pp. 640–649, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00111.

[32] Jia, Z.—Witchel, E.: Nightcore: Efficient and Scalable Serverless Computing for Latency-Sensitive, Interactive Microservices. Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21), 2021, pp. 152–166, doi: 10.1145/3445814.3446701.

[33] Solaiman, K.—Adnan, M. A.: WLEC: A Not So Cold Architecture to Mitigate Cold Start Problem in Serverless Computing. 2020 IEEE International Conference on Cloud Engineering (IC2E), 2020, pp. 144–153, doi: 10.1109/IC2E48712.2020.00022.

[34] Vahidinia, P.—Farahani, B.—Aliee, F. S.: Cold Start in Serverless Computing: Current Trends and Mitigation Strategies. 2020 International Conference on Omni-Layer Intelligent Systems (COINS), IEEE, 2020, pp. 1–7, doi: 10.1109/COINS49042.2020.9191377.

[35] Arora, S.—Bala, A.: A Survey: ICT Enabled Energy Efficiency Techniques for Big Data Applications. Cluster Computing, Vol. 23, 2020, No. 2, pp. 775–796, doi: 10.1007/s10586-019-02958-6.

[36] Yu, H.—Wang, H.—Li, J.—Yuan, X.—Park, S. J.: Accelerating Serverless Computing by Harvesting Idle Resources. Proceedings of the ACM Web Conference 2022 (WWW '22), 2022, pp. 1741–1751, doi: 10.1145/3485447.3511979.

[37] Mampage, A.—Karunasekera, S.—Buyya, R.: Deadline-Aware Dynamic Resource Management in Serverless Computing Environments. 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 483–492, doi: 10.1109/CCGrid51090.2021.00058.

[38] Sreekanti, V.—Wu, C.—Chhatrapati, S.—Gonzalez, J. E.—Hellerstein, J. M.—Faleiro, J. M.: A Fault-Tolerance Shim for Serverless Computing. Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20), 2020, doi: 10.1145/3342195.3387535.

[39] Kim, J.—Lee, K.: I/O Resource Isolation of Public Cloud Serverless Function Runtimes for Data-Intensive Applications. Cluster Computing, Vol. 23, 2020, No. 3, pp. 2249–2259, doi: 10.1007/s10586-020-03103-4.

[40] Wen, J.—Chen, Z.—Liu, Y.—Lou, Y.—Ma, Y.—Huang, G.—Jin, X.—Liu, X.: An Empirical Study on Challenges of Application Development in Serverless Computing. Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021), 2021, pp. 416–428, doi: 10.1145/3468264.3468558.

[41] Pérez, A.—Risco, S.—Naranjo, D. M.—Caballer, M.—Moltó, G.: On-Premises Serverless Computing for Event-Driven Data Processing Applications. 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 2019,

pp. 414–421, doi: 10.1109/CLOUD.2019.00073.

[42] Enes, J.—Expósito, R. R.—Touriño, J.: Real-Time Resource Scaling Platform for Big Data Workloads on Serverless Environments. Future Generation Computer Systems, Vol. 105, 2020, pp. 361–379, doi: 10.1016/j.future.2019.11.037.

[43] Jackson, D.—Clynch, G.: An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions. 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 154–160, doi: 10.1109/UCC-Companion.2018.00050.

[44] Golec, M.—Ozturac, R.—Pooranian, Z.—Gill, S. S.—Buyya, R.: iFaaS-Bus: A Security- and Privacy-Based Lightweight Framework for Serverless Computing Using IoT and Machine Learning. IEEE Transactions on Industrial Informatics, Vol. 18, 2022, No. 5, pp. 3522–3529, doi: 10.1109/TII.2021.3095466.

[45] Singh, P.—Kaur, A.—Gill, S. S.: Machine Learning for Cloud, Fog, Edge and Serverless Computing Environments: Comparisons, Performance Evaluation Benchmark and Future Directions. International Journal of Grid and Utility Computing, Vol. 13, 2022, No. 4, pp. 447–457, doi: 10.1504/IJGUC.2022.125151.

[46] Grafberger, A.—Chadha, M.—Jindal, A.—Gu, J.—Gerndt, M.: FedLess: Secure and Scalable Federated Learning Using Serverless Computing. 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 164–173, doi: 10.1109/BigData52589.2021.9672067.

[47] Bebortta, S.—Das, S. K.—Kandpal, M.—Barik, R. K.—Dubey, H.: Geospatial Serverless Computing: Architectures, Tools and Future Directions. ISPRS International Journal of Geo-Information, Vol. 9, 2020, No. 5, Art. No. 311, doi: 10.3390/ijgi9050311.

[48] Gill, S. S.: Quantum and Blockchain Based Serverless Edge Computing: A Vision, Model, New Trends and Future Directions. Internet Technology Letters, Vol. 7, 2024, No. 1, Art. No. e275, doi: 10.1002/itl2.275.

[49] Mateus-Coelho, N.—Cruz-Cunha, M.: Serverless Service Architectures and Security Minimals. 2022 10$^{\text{th}}$ International Symposium on Digital Forensics and Security (ISDFS), IEEE, 2022, pp. 1–6, doi: 10.1109/ISDFS55398.2022.9800779.

[50] Marin, E.—Perino, D.—Di Pietro, R.: Serverless Computing: A Security Perspective. Journal of Cloud Computing, Vol. 11, 2022, No. 1, Art. No. 69, doi: 10.1186/s13677-022-00347-w.

[51] Bardsley, D.—Ryan, L.—Howard, J.: Serverless Performance and Optimization Strategies. 2018 IEEE International Conference on Smart Cloud (SmartCloud), 2018, pp. 19–26, doi: 10.1109/SmartCloud.2018.00012.

[52] Rajan, R. A. P.: Serverless Architecture – A Revolution in Cloud Computing. 2018 Tenth International Conference on Advanced Computing (ICoAC), 2018, pp. 88–93, doi: 10.1109/ICoAC44903.2018.8939081.

[53] Grogan, J.—Mulready, C.—McDermott, J.—Urbanavicius, M.—Yilmaz, M.—Abgaz, Y.—McCarren, A.—MacMahon, S. T.—Garousi, V.—Elger, P.—Clarke, P.: A Multivocal Literature Review of Function-as-a-Service (FaaS) Infrastructures and Implications for Software Developers. In: Yilmaz, M., Niemann, J., Clarke, P., Messnarz, R. (Eds.): Systems,

Software and Services Process Improvement (EuroSPI 2020). Springer, Communications in Computer and Information Science, Vol. 1251, 2020, pp. 58–75, doi: 10.1007/978-3-030-56441-4_5.

[54] Vahidinia, P.—Farahani, B.—Aliee, F. S.: Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach. IEEE Internet of Things Journal, Vol. 10, 2023, No. 5, pp. 3917–3927, doi: 10.1109/JIOT.2022.3165127.

[55] Liu, X.—Wen, J.—Chen, Z.—Li, D.—Chen, J.—Liu, Y.—Wang, H.—Jin, X.: FaaSLight: General Application-Level Cold-Start Latency Optimization for Function-as-a-Service in Serverless Computing. ACM Transactions on Software Engineering and Methodology, Vol. 32, 2023, No. 5, Art. No. 119, doi: 10.1145/3585007.

[56] Fuerst, A.—Sharma, P.: FaasCache: Keeping Serverless Computing Alive with Greedy-Dual Caching. Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21), 2021, pp. 386–400, doi: 10.1145/3445814.3446757.

[57] Kumari, A.—Sahoo, B.—Behera, R. K.: Workflow Aware Analytical Model to Predict Performance and Cost of Serverless Execution. Concurrency and Computation: Practice and Experience, Vol. 35, 2023, No. 22, Art. No. e7743, doi: 10.1002/cpe.7743.

[58] Aljicevic, Z.—Kasapovic, S.—Hivziefendic, J.—Kevric, J.—Mujkic, S.: Resource Allocation Model for Cloud-Fog-Based Smart Grid. Science and Technology for Energy Transition, Vol. 78, 2023, Art. No. 28, doi: 10.2516/stet/2023030.

[59] Kaur, S.—Bala, A.—Parashar, A.: A Multi-Step Electricity Prediction Model for Residential Buildings Based on Ensemble Empirical Mode Decomposition Technique. Science and Technology for Energy Transition, Vol. 79, 2024, Art. No. 7, doi: 10.2516/stet/2024001.

[60] Hussain, A.—Aleem, M.—Iqbal, M. A.—Islam, M. A.: Investigation of Cloud Scheduling Algorithms for Resource Utilization Using CloudSim. Computing and Informatics, Vol. 38, 2019, No. 3, pp. 525–554, doi: 10.31577/cai_2019_3_525.

[61] Pérez, A.—Moltó, G.—Caballer, M.—Calatrava, A.: Serverless Computing for Container-Based Architectures. Future Generation Computer Systems, Vol. 83, 2018, pp. 50–59, doi: 10.1016/j.future.2018.01.022.

[62] Lin, C.—Khazaei, H.: Modeling and Optimization of Performance and Cost of Serverless Applications. IEEE Transactions on Parallel and Distributed Systems, Vol. 32, 2021, No. 3, pp. 615–632, doi: 10.1109/TPDS.2020.3028841.

[63] Suo, K.—Son, J.—Cheng, D.—Chen, W.—Baidya, S.: Tackling Cold Start of Serverless Applications by Efficient and Adaptive Container Runtime Reusing. 2021 IEEE International Conference on Cluster Computing (CLUSTER), 2021, pp. 433–443, doi: 10.1109/Cluster48925.2021.00018.

[64] McGrath, G.—Brenner, P. R.: Serverless Computing: Design, Implementation, and Performance. 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 405–410, doi: 10.1109/ICDCSW.2017.36.

[65] Ishakian, V.—Muthusamy, V.—Slominski, A.: Serving Deep Learning Models in a Serverless Platform. 2018 IEEE International Conference on Cloud Engineering

(IC2E), 2018, pp. 257–262, doi: 10.1109/IC2E.2018.00052.

[66] Van Eyk, E.—Iosup, A.—Abad, C. L.—Grohmann, J.—Eismann, S.: A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures. Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18), 2018, pp. 21–24, doi: 10.1145/3185768.3186308.

[67] Cicconetti, C.—Conti, M.—Passarella, A.: A Decentralized Framework for Serverless Edge Computing in the Internet of Things. IEEE Transactions on Network and Service Management, Vol. 18, 2021, No. 2, pp. 2166–2180, doi: 10.1109/TNSM.2020.3023305.

[68] Wu, C.—Sreekanti, V.—Hellerstein, J. M.: Transactional Causal Consistency for Serverless Computing. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20), 2020, pp. 83–97, doi: 10.1145/3318464.3389710.

[69] Balla, D.—Maliosz, M.—Simon, C.: Open Source FaaS Performance Aspects. 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2020, pp. 358–364, doi: 10.1109/TSP49548.2020.9163456.

[70] Khatri, D.—Khatri, S. K.—Mishra, D.: Potential Bottleneck and Measuring Performance of Serverless Computing: A Literature Study. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), IEEE, 2020, pp. 161–164, doi: 10.1109/ICRITO48877.2020.9197837.

[71] Henschel, J.: Dimensioning, Performance and Optimization of Cloud-Native Applications. Master Thesis. 2021, `https://urn.fi/URN:NBN:fi:aalto-202108298554`.

[72] Masanet, E.—Shehabi, A.—Lei, N.—Smith, S.—Koomey, J.: Recalibrating Global Data Center Energy-Use Estimates. Science, Vol. 367, 2020, No. 6481, pp. 984–986, doi: 10.1126/science.aba3758.

[73] Shehabi, A.—Smith, S.—Sartor, D.—Brown, R.—Herrlin, M.—Koomey, J.—Masanet, E.—Horner, N.—Azevedo, I.—Lintner, W.: United States Data Center Energy Usage Report. Technical Report No. LBNL–1005775; ir:1005775, 2016, doi: 10.2172/1372902.

[74] Denninnart, C.—Salehi, M. A.: Efficiency in the Serverless Cloud Computing Paradigm: A Survey Study. CoRR, 2021, doi: 10.1002/spe.3233.

[75] Aslanpour, M. S.—Toosi, A. N.—Cheema, M. A.—Gaire, R.: Energy-Aware Resource Scheduling for Serverless Edge Computing. 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2022, pp. 190–199, doi: 10.1109/CCGrid54584.2022.00028.

[76] Byrne, A.—Pang, Y.—Zou, A.—Nadgowda, S.—Coskun, A. K.: MicroFaaS: Energy-Efficient Serverless on Bare-Metal Single-Board Computers. 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022, pp. 754–759, doi: 10.23919/DATE54114.2022.9774688.

[77] Hassan, H. A.—Salem, S. A.—Saad, E. M.: A Smart Energy and Reliability Aware Scheduling Algorithm for Workflow Execution in DVFS-Enabled Cloud Environment. Future Generation Computer Systems, Vol. 112, 2020, pp. 431–448, doi: 10.1016/j.future.2020.05.040.

[78] Jindal, A.—Chadha, M.—Gerndt, M.—Frielinghaus, J.—Podolskiy, V.—Chen, P.: Poster: Function Delivery Network: Extending Serverless to Heterogeneous Computing. 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021, pp. 1128–1129, doi: 10.1109/ICDCS51616.2021.00120.

[79] Guo, L.—Zhang, Y.—Zhao, S.: Heuristic Algorithms for Energy and Performance Dynamic Optimization in Cloud Computing. Computing and Informatics, Vol. 36, 2017, No. 6, pp. 1335–1360, doi: 10.4149/cai_2017_6_1335.

[80] Kallam, S.—Patan, R.—Ramana, T. V.—Gandomi, A. H.: Linear Weighted Regression and Energy-Aware Greedy Scheduling for Heterogeneous Big Data. Electronics, Vol. 10, 2021, No. 5, Art. No. 554, doi: 10.3390/electronics10050554.

[81] Aslanpour, M. S.—Toosi, A. N.—Gaire, R.—Cheema, M. A.: WattEdge: A Holistic Approach for Empirical Energy Measurements in Edge Computing. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. Y. (Eds.): Service-Oriented Computing (ICSOC 2021). Springer, Cham, Lecture Notes in Computer Science, Vol. 13121, 2021, pp. 531–547, doi: 10.1007/978-3-030-91431-8_33.

[82] Paul, J. J.: Serverless Through the AWS Well-Architected Framework. Distributed Serverless Architectures on AWS: Design and Implement Serverless Architectures, Apress, Berkeley, CA, 2023, pp. 131–141, doi: 10.1007/978-1-4842-9159-7_8.

[83] El Ioini, N.—Hästbacka, D.—Pahl, C.—Taibi, D.: Platforms for Serverless at the Edge: A Review. In: Zirpins, C., Paraskakis, I., Andrikopoulos, V., Kratzke, N., Pahl, C. et al. (Eds.): Advances in Service-Oriented and Cloud Computing (ESOCC 2020). Springer, Cham, Communications in Computer and Information Science, Vol. 1360, 2021, pp. 29–40, doi: 10.1007/978-3-030-71906-7_3.

[84] Li, J.—Kulkarni, S. G.—Ramakrishnan, K. K.—Li, D.: Analyzing Open-Source Serverless Platforms: Characteristics and Performance. CoRR, 2021, doi: 10.48550/arXiv.2106.03601.

[85] Kuntsevich, A.—Nasirifard, P.—Jacobsen, H. A.: A Distributed Analysis and Benchmarking Framework for Apache OpenWhisk Serverless Platform. Proceedings of the 19th International Middleware Conference (Posters) (Middleware '18), 2018, pp. 3–4, doi: 10.1145/3284014.3284016.

[86] Habala, O.—Bobák, M.—Šeleng, M.—Tran, V.—Hluchý, L.: Architecture of a Function-as-a-Service Application. Computing and Informatics, Vol. 42, 2023, No. 4, pp. 878–895, doi: 10.31577/cai_2023_4_878.

[87] Djemame, K.—Datsev, D.—Kelefouras, V.: Evaluation of Language Runtimes in Open-Source Serverless Platforms. Proceedings of the 12th International Conference on Cloud Computing and Services Science CLOSER – Volume 1, SciTePress, 2022, pp. 123–132, doi: 10.5220/0010983000003200.

[88] Kurniawan, A.—Lau, W.: Introduction to Azure Functions. Apress, Berkeley, CA, 2019, pp. 1–21, doi: 10.1007/978-1-4842-5067-9_1.

[89] Lyu, X.—Cherkasova, L.—Aitken, R.—Parmer, G.—Wood, T.: Towards Efficient Processing of Latency-Sensitive Serverless DAGs at the Edge. Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '22), 2022, pp. 49–54, doi: 10.1145/3517206.3526274.

[90] Malawski, M.—Gajek, A.—Zima, A.—Balis, B.—Figiela, K.: Serverless Ex-

ecution of Scientific Workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. Future Generation Computer Systems, Vol. 110, 2020, pp. 502–514, doi: 10.1016/j.future.2017.10.029.

[91] Wang, A.—Chang, S.—Tian, H.—Wang, H.—Yang, H.—Li, H.—Du, R.—Cheng, Y.: FaaSNet: Scalable and Fast Provisioning of Custom Serverless Container Runtimes at Alibaba Cloud Function Compute. 2021 USENIX Annual Technical Conference (USENIX ATC 21), 2021, pp. 443–457, `https://www.usenix.org/conference/atc21/presentation/wang-ao`.

[92] Pérez, A.—Moltó, G.—Caballer, M.—Calatrava, A.: A Programming Model and Middleware for High Throughput Serverless Computing Applications. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), 2019, pp. 106–113, doi: 10.1145/3297280.3297292.

[93] Mohanty, S. K.—Premsankar, G.—Di Francesco, M.: An Evaluation of Open Source Serverless Computing Frameworks. 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom, Vol. 2018, 2018, pp. 115–120, doi: 10.1109/CloudCom2018.2018.00033.

Jasmine Kaur is a Research Scholar in the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India. She received her M.Tech. from the Punjabi University, Patiala and is pursuing her Ph.D. in the research area of serverless computing from Thapar University, Patiala.

Inderveer Chana has Ph.D. in computer science with specialization in grid computing and her M.Eng. in software engineering from the Thapar Institute of Engineering and Technology, Patiala, India and her B.Eng. in computer science and engineering. She is presently serving as Professor in the Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala. Her research interests include grid and cloud computing and other areas of interest are software engineering and software project management. She has more than 100 research publications in reputed journals and conferences.

Anju Bala is working as Professor in the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India. She received her B.Eng. in computer science and engineering and her M.Tech. from the Punjabi University and her Ph.D. in the research area of cloud computing from the Thapar University, Patiala. She has more than 100 research publications in reputed journals and conferences.