International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 13 Issue 9 ǁ September 2025 ǁ PP. 09-14

A Comparative Review on Scheduling Techniques in Serverless Computing

Eunyoung Lee
*[*1]Department of Computer Science, Dongduk Women's University, Seoul, Korea*
Corresponding Author: Eunyoung Lee

Abstract
The urge of users to focus on their core application tasks without the need to manage complex virtual environments in the cloud has driven the emergence of serverless computing, a new computing model. In this model, users can delegate resource allocation and other server management tasks to the service provider. This allows users to concentrate solely on developing their application code. A serverless platform manages the cloud environment on behalf of the users and is responsible for executing the serverless functions that comprise of the application. Given the pay-per-use characteristic of serverless computing, where users are billed in proportion to the resources they consume, scheduling to provide an optimal environment for each user is crucial for both users and service providers. This paper classifies various factors that affect serverless computing performance related to scheduling and analyzes the latest research trends. With these findings, future research directions for serverless computing are also discussed.

Keywords: *Serverless Computing, Cloud Computing, Function Orchestration, Scheduling, Resource Management*

Date of Submission: 26-08-2025 Date of acceptance: 04-09-2025

I. INTRODUCTION

Cloud computing has gained immense interest from industry, academia, and government agencies since its inception with experiencing rapid growth in a short time. This swift expansion has significantly impacted both the IT infrastructure and the related software industry. Cloud computing operates like a utility service, allowing users to subscribe to IT resources rather than owning them [1].

The demand from users (i.e., developers) to focus on core application tasks, free from the complexities of managing virtual machines in the cloud, led to the birth of serverless computing in the mid-2010s [2, 3]. The term "serverless" emphasizes that the cloud service provider handles resource allocation and server management, enabling developers to concentrate solely on their application code. A serverless platform manages the virtual cloud environment for users and executes the serverless functions that constitute a user's application [4]. Since billing is based on resource consumption, scheduling is crucial for both service consumers and service providers to ensure an optimal environment.

This paper identifies factors for improving serverless computing system performance through scheduling and analyzes the latest research trends for each factor. Based on this analysis, future research directions are then explored. In Section 2, the core concepts of the serverless computing model are explained, and its unique characteristics are highlighted by comparing it to existing cloud systems. Section 3 analyzes research trends and topics aimed at improving serverless computing scheduling performance. This includes categorizing research by energy efficiency, resource usage patterns, workflows, data-oriented serverless, and packaging. Finally, Section 4 concludes with a summary of the analysis and prospects for future research.

II. SERVERLESS COMPUTING: OVERVIEW

2.1 SERVERLESS ARCHITECTURE

Cloud computing has made it easier for developers to build and operate software and services without worrying about IT infrastructure. Cloud computing provides users with virtualized hardware environments, platforms, and software services based on virtualization technology for physical hardware.

While providing an environment similar to existing computing systems lowered the initial barrier for cloud adoption, it also burdened users with managing the virtual machine's settings [5, 6]. To use cloud computing, users often had to act as system administrators or hire one to manage the hardware settings of their virtual machines, which was a significant burden for many. The demand for users to escape complex virtual environment management in the cloud and focus on their core application tasks led to the birth of serverless computing. Serverless computing is an event-driven computing model where users can define and execute their application logic as stateless functions [1, 3].

Generally, serverless computing services are defined as a combination of Function as a Service (FaaS) and Back- end as a Service (BaaS) [4, 7]. The FaaS part allows developers to implement their own application functionalities and manage their execution. With FaaS, developers can focus entirely on application logic without worrying about the underlying infrastructure. The BaaS takes the responsibility of the other part of serverless computing, where a service provider offers specific functionalities as online services. These services are typically delegated to the cloud. Common examples of BaaS are authentication and notification services. FaaS is used to execute user-defined functions, while BaaS provides pre-defined functionalities from the serverless service provider.

Regardless of whether they use FaaS or BaaS, users do not need to concern themselves with resource man- agement. In essence, serverless computing is one of the cloud computing models designed to hide the virtual environment setup of traditional server-based cloud computing, allowing users to focus more on developing and implementing their application-specific services. The first commercial serverless service was Amazon Lambda, launched in November 2014. Other major commercial serverless computing services include Google Cloud Functions,Microsoft Azure Functions, and Apache OpenWhisk.

2.2 SERVERLESS FUNCTIONS

The serverless computing model is a programming model where the service provider is responsible for all aspects of application resource management. This provides the advantage of allowing users to focus solely on their application development without the burden of complex computing infrastructure management. Once an application is registered on a serverless platform, the service provider handles the entire process, from initial resource allocation and scheduling to execution monitoring and resource scaling.

Applications running on a serverless platform are composed of serverless functions. A serverless function contains the application logic, and an application can consist of one or more related stateless functions. In this sense, serverless functions are the fundamental unit of serverless computing. Many researchers predict that serverless functions will become the basic unit of abstraction for general-purpose programming models in the cloud.

On a serverless platform, users develop applications by writing their desired functionality as serverless functions and selecting the events that trigger their execution. Serverless functions are written using high-level programming languages supported by the service platform. Since the serverless platform handles all other necessary cloud computing tasks, users don't need to worry about any additional environmental configurations beyond the actual application code. The tasks performed by the serverless platform include instance selection, scaling, deployment, fault tolerance, monitoring, logging, and security patching.

2.3 ORCHESTRATION

Serverless applications are built by combining multiple functions with independent functionalities in various ways. The process of combining these functions into a desired workflow is called orchestration in serverless computing. Since it is unusual for a serverless application to consist of a single function, different workflows can exist for similar applications, depending on the developer's preferences and intentions. In other words, different workflows are created based on how a developer performs orchestration.

From a service provider's perspective, having more information about an application's function call chain makes it easier to improve the performance of the serverless platform. By leveraging workflow information, a service platform can pre-warm or optimize the necessary functions, thereby improving the overall application performance and reducing costs for users.

Service providers primarily obtain serverless application workflow information in two ways. The most common method is for the developer to provide their application's workflow information using a pre-defined interface from the service provider. The platform then analyzes this user-provided workflow to make performance-enhancing decisions. This approach is used by most current serverless platforms; Amazon Step Functions [13], Azure Durable Functions [14], and Google Cloud Composer [15] are the examples of interfaces that allow developers to provide workflow information.

Even with general purpose or domain-specific workflow patterns, the burden of describing the workflow of their serverless application can still be a challenge for developers. Therefore, there is a growing need for tools that analyze the workflows of existing serverless applications or that provide developers with a means to analyze workflow attributes during the orchestration phase. These areas are considered valuable for future research.

III. SCHEDULING TECHNIQUES

Resource management is the process of allocating an appropriate amount of resources to an application and managing those allocated resources to satisfy the Quality of Service (QoS) requested by users.

Users send execution requests (invocation requests) for the functions they have written to a serverless service provider, and the service provider must process these requests within a specified deadline. Consequently, the service provider must decide which computational node and when to execute the function based on the request content. In this process, the service provider must consider the platform's overall energy consumption, resource usage, and other factors to determine the nodes for computation and their execution order [8].

This series of decision-making processes is called scheduling, and it is one of the most active research areas in serverless computing. Various techniques are being proposed for resource management and scheduling, depending on which factors are emphasized for resource allocation and scheduling. Developers or serverless platform administrators can effectively improve the performance of a serverless system by choosing resource management and scheduling techniques that are optimized for the characteristics of their applications.

In Table 1, related studies are categorized by factors which influence scheduling performance. The table shows the key concepts of each factor and major techniques proposed in the related research.

Table 1: Classification of scheduling techniques.

Factor	Key Concept	Techniques	References
Energy Efficiency	Minimizing energy consumption by using cold-state containers, which can cause warm-up delays.	• Keeping active containers to minimize latency • Using less expensive heterogeneous nodes • Minimizing additional resource consumption.	[16], [17], [18]
Resource Usage Patterns	Avoiding resource contention when applications with similar resource usage patterns are scheduled on the same physical node.	• Classifying functions by CPU/memory usage and distributing them • Predicting execution times to allocate resources efficiently • Dynamically predicting function execution for pre-scheduling.	[17], [18], [19], [20], [21], [22], [23]
Workflow	Leveraging the sequence of function calls for better scheduling	• Analyzing task graphs to identify dependencies for parallel execution • Pre-locating suitable nodes and pre-allocating resources for upcoming functions.	[24], [25], [26], [27], [28], [29], [30], [31]
Data-oriented Serverless	Optimizing performance for applications with a high dependency on external data.	• Proactively pre-fetching data and reusing runtime data • Prioritizing data locality in scheduling to reduce data overhead.	[32], [33], [34], [35]
Packaging	Reducing the delay caused by installing necessary libraries and packages before a function execution.	• Using user-provided library information to mitigate delays • Scheduling techniques to specifically address packaging-related delays.	[36], [37]

3.1 ENERGY EFFICIENCY

To minimize energy consumption, serverless platforms use methods like placing unused containers in hibernate mode or cold-state mode.

While this method can reduce energy usage, the delay that occurs when a container transitions from a cold state to an active state carries the risk of not meeting the user-specified deadline. Therefore, even when using cold-state mode, scheduling must be designed to minimize warm-up delays. Ensure [16] proposed a technique to keep a few containers in an active state, even if they are not in use, to minimize latency. Fifer [17] adopted a similar approach to avoid cold states as much as possible.

However, keeping more containers in an active state than necessary leads to a waste of energy and resources. It is essential to minimize this additional resource consumption while maintaining a sufficient number of active containers to minimize latency. Ensure [16] also presented a theoretical model to minimize the total amount of resources. Roy [18] proposed a technique to minimize warm-up costs by utilizing less expensive heterogeneous nodes. Further research on scheduling that considers both energy efficiency and cost is expected to become more active in the future.

3.2 RESOURCE USAGE PATTERNS

Serverless applications often exhibit consistent resource usage patterns depending on their domain. The resources that serverless functions compete for include not only CPU but also memory, disk, and network resources. For example, applications that require many arithmetic operations tend to have high CPU usage, while applications that analyze given data tend to have high memory usage.

If functions with similar resource usage patterns are allocated to the same physical node, resource contention among them will inevitably occur, which can lower overall performance. Therefore, scheduling which allocates resources to balance the use of a given physical node's resources and minimizes competition among functions with similar usage patterns is highly effective in improving the performance of a serverless service.

FnSched [19] is an early study that focused on the relationship between resources and performance in serverless services. This research proposed a method to classify serverless functions based on their CPU usage and distribute them among physical nodes to minimize competition. Fifer [17] adopted an offiine profiling method to predict function execution times. Akhtar [20] proposed a method to allocate resources by predicting execution time before a function runs and then scheduling it to a physical node using statistical techniques. Roy [18] also proposed a method to improve system efficiency by dynamically predicting whether a function will be executed and pre-scheduling the node for its execution. Hoseinyfarahabady also proposed a predictive model for anticipating application execution times and satisfying QoS [21–23].

As more diverse domains are expected to use the serverless computing model, research on scheduling techniques that leverage resource usage patterns is expected to continue for each domain.

3.3 WORKFLOW

Serverless applications are composed of independent, stateless functions, and their actual execution consists of a series of calls to these constituent functions. This sequence of function calls is called a workflow.

The completeness of a serverless application's workflow and orchestration greatly impacts the performance of the serverless platform that executes it. This analysis is discussed in detail in Section 2.3. Workflow information can also be effectively used to improve resource management and scheduling perfor- mance.

Service platforms can receive information about the call chain of individual applications in the form of a directed task graph. Within this graph, each node represents a function, and the edges represent dependencies or the execution order between functions. By analyzing the task graph, service providers can identify features such as cycles, self-loops, or conditional branches, and based on this, they make scheduling decisions to enhance efficiency, such as parallel execution. Lin [24] discussed how to improve overall system performance using task graphs. WiseFuse [25] is another system that uses task graphs to optimize execution order, considering both user-defined latency and cost constraints simultaneously.

A scheduler can use information about the function call order to perform more efficient scheduling and ultimately improve the overall performance of the serverless system. If the scheduler knows which function will be executed next or can predict a branch, it can pre-locate a suitable physical node for the function's execution and pre-allocate the necessary resources within that node. Pre-securing the resources required for function execution can also reduce the latency associated with waking up from a cold state.

Research on scheduling techniques that leverage function call chain information is also active. Xanadu [26] used workflow structure information to reduce warm-up latency. Archipelago [27] demonstrated an attempt to use a Directed Acyclic Graph (DAG) structure to predict the function pool that will be used. The Sequoia framework [28] utilized function call chain information to design a scheduler that considers QoS.

Burckhardt [29] and Wen [30] conducted research to analyze the workflows of serverless applications and derive domain-specific characteristics. John [31] proposed a framework that facilitates the development of domain-specific applications by supporting workflow composition that reflects domain characteristics.

Even with general-purpose languages or domain-specific workflow patterns, the burden on serverless ap- plication developers to manually describe their application's workflow remains. Therefore, there is a need for more research into tools that analyze the workflows of already developed serverless applications or that analyze workflow properties for developers during the orchestration phase. These are considered valuable future research areas.

3.4 DATA-ORIENTED SERVERLESS

In an ideal serverless computing environment, serverless applications are composed solely of stateless functions and do not depend on any external data sources. However, in reality, it is almost impossible to build an application that completely excludes external data. For example, machine learning applications, which have recently received significant attention, are categorized as software with a very high dependency on externally stored data.

For applications with a high dependency on external data where external data plays a crucial role, scheduling must consider dataflow, and related research is active. Freshen [32] proposed a method to reduce data overhead during the execution of serverless functions by allowing developers or service providers to proactively pre-fetch necessary data along with runtime reuse capabilities. Cloudburst [33] also adopted a method to reduce data overhead by prioritizing data locality in its scheduling. Rausch [34] showed an attempt to consider both dataflow and the characteristics of domain applications in edge computing scheduling. Kaffes [35] also implemented a technique to reduce waiting time before a function's execution by considering data locality, and they experimented with its performance using Apache OpenWhisk.

3.5 PACKAGING

In recent serverless techniques, packaging has emerged as a crucial factor that determines the performance of a serverless platform. When a function invocation request occurs, the serverless platform must first install the necessary libraries and related packages on the computation node. This process of setting up the execution environment on the computation node inevitably causes a certain amount of delay. OpenLambda [36] is a representative system where execution delays related to packaging occur. Amumala [37] proposed a scheduling technique to mitigate the execution delays caused by packaging in OpenLambda.

Most serverless platforms currently ask developers to register information about related libraries and packages during the application development process, and they attempt to reduce packaging-related delays using the information provided by the user. While research in this area is not yet very active, it is a topic worthy of consideration for future researchers, as it aligns with the core purpose of serverless computing: to reduce the burden of environment setup and resource management on the users.

IV. CONCLUSION

A new computing model, serverless computing, has emerged to meet the demands of users who want to focus on their core application tasks without the complexities of managing virtual environments in the cloud. Using this model, a user can delegate resource allocation and intricate server management to a service provider. This allows the user to concentrate solely on application development. By reducing the burden on cloud service users, serverless computing has enhanced the utility of cloud computing and is expected to become a foundational model for future cloud environments.

This paper identifies the factors that must be considered during the scheduling process to improve the performance of serverless computing systems. It analyzes current research trends for each factor and, based on these findings, explores future research directions for serverless computing. The serverless computing model is expected to become the foundational computing model for the cloud. The number of users has increased dramatically, and new domains are continuously adopting the serverless model. Although research in this field has begun, it is evident that additional research and attention are needed to keep pace with its rapid growth and evolving applications.

REFERENCES

[1]. P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," Communications of the ACM, vol. 62, no. 12, pp. 44–54, 2019.
[2]. B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," Future Generation Computer Systems, vol. 79, pp. 849–861, 2018.
[3]. J. Wen, Z. Chen, X. Jin, and X. Liu, "Rise of the planet of serverless computing: A systematic review," ACM Transactions on Software Engineering and Methodology, 2023.
[4]. H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: a survey of opportunities, challenges, and applications," ACM Computing Surveys, vol. 54, pp. 1–32, 2022.
[5]. C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware cluster management," ACM SIGPLAN Notices, vol. 49, pp. 127–144, 2014.
[6]. Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of internet services and applications, vol. 1, pp. 7–18, 2010.
[7]. Z. Li, L. Guo, J. Cheng, Q. Chen, B. He, and M. Guo, "The serverless computing survey: A technical primer for design architecture," ACM Computing Surveys (CSUR), vol. 54, pp. 1–34, 2022.
[8]. "AWS Step Functions." https://aws.amazon.com/step-functions/. Online available; accessed at 2025/08/28.
[9]. "Azure Durable Functions." https://docs.microsoft.com/en-us/azure/azure-functions/durable/. Online available; accessed at 2025/08/28.
[10]. "Google Cloud Composer." https://cloud.google.com/composer. Online available; accessed at 2025/08/28.
[11]. A. Mampage, S. Karunasekera, and R. Buyya, "A holistic view on resource management in serverless computing environments: Taxonomy and future directions," ACM Computing Surveys (CSUR), vol. 54, pp. 1–36, 2022.
[12]. A. Suresh, G. Somashekar, A. Varadarajan, V. R. Kakarla, H. Upadhyay, and A. Gandhi, "Ensure: Efficient scheduling and autonomous resource management in serverless environments," in 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), pp. 1–10, 2020.
[13]. J. R. Gunasekaran, P. Thinakaran, N. Chidambaram, M. T. Kandemir, and C. R. Das, "Fifer: Tackling underutilization in the serverless era," arXiv preprint arXiv:2008.12819, 2020.
[14]. R. B. Roy, T. Patel, and D. Tiwari, "IceBreaker: Warming serverless functions better with heterogeneity," in Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 753–767, 2022.
[15]. A. Suresh and A. Gandhi, "Fnsched: An efficient scheduler for serverless functions," in Proceedings of the 5th international workshop on serverless computing, pp. 19–24, 2019.
[16]. N. Akhtar, A. Raza, V. Ishakian, and I. Matta, "Cose: Configuring serverless functions using statistical learning," in IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 129–138, 2020.
[17]. M. HoseinyFarahabady, Y. C. Lee, A. Y. Zomaya, and Z. Tari, "A QoS-aware resource allocation controller for function as a service (FaaS) platform," in Service-Oriented Computing: 15th International Conference, ICSOC 2017, Malaga, Spain, November 13–16, 2017, Proceedings, pp. 241–255, 2017.

[18]. M. R. HoseinyFarahabady, A. Y. Zomaya, and Z. Tari, "A model predictive controller for managing qos enforcements and microarchitecture- level interferences in a lambda platform," IEEE Transactions on Parallel and Distributed Systems, vol. 29, pp. 1442–1455, 2017.

[19]. Y. K. Kim, M. R. HoseinyFarahabady, Y. C. Lee, and A. Y. Zomaya, "Automated fine-grained CPU cap control in serverless computing platform," IEEE Transactions on Parallel and Distributed Systems, vol. 31, pp. 2289–2301, 2020.

[20]. C. Lin and H. Khazaei, "Modeling and optimization of performance and cost of serverless applications," IEEE Transactions on Parallel and Distributed Systems, vol. 32, pp. 615–632, 2020.

[21]. A. Mahgoub, E. B. Yi, K. Shankar, E. Minocha, S. Elnikety, S. Bagchi, and S. Chaterji, "WiseFuse: Workload characterization and DAG transformation for serverless workflows," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 6, no. 2, pp. 1–28, 2022.

[22]. N. Daw, U. Bellur, and P. Kulkarni, "Xanadu: Mitigating cascading cold starts in serverless function chain deployments," in Proceedings of the 21st International Middleware Conference, pp. 356–370, 2020.

[23]. A. Singhvi, K. Houck, A. Balasubramanian, M. D. Shaikh, S. Venkataraman, and A. Akella, "Archipelago: A scalable low-latency serverless platform," arXiv preprint arXiv:1911.09849, 2019.

[24]. A. Tariq, A. Pahl, S. Nimmagadda, E. Rozner, and S. Lanka, "Sequoia: Enabling quality-of-service in serverless computing," in Proceedings of the 11th ACM symposium on cloud computing, pp. 311–327, 2020.

[25]. S. Burckhardt, C. Gillum, D. Justo, K. Kallas, C. McMahon, and C. S. Meiklejohn, "Serverless workflows with durable functions and netherite," arXiv preprint arXiv:2103.00033, 2021.

[26]. J. Wen and Y. Liu, "An empirical study on serverless workflow service," arXiv preprint arXiv:2101.03513, 2021.

[27]. A. John, K. Ausmees, K. Muenzen, C. Kuhn, and A. Tan, "Sweep: accelerating scientific research through scalable serverless workflows," in Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, pp. 43–50, 2019.

[28]. E. Hunhoff, S. Irshad, V. Thurimella, A. Tariq, and E. Rozner, "Proactive serverless function resource management," in Proceedings of the 2020 Sixth International Workshop on Serverless Computing, pp. 61–66, 2020.

[29]. V. Sreekanti, C. Wu, X. C. Lin, J. Schleier-Smith, J. M. Faleiro, J. E. Gonzalez, J. M. Hellerstein, and A. Tumanov, "Cloudburst: Stateful functions-as-a-service," arXiv preprint arXiv:2001.04592, 2020.

[30]. T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," Future Generation Computer Systems, vol. 114, pp. 259–271, 2021.

[31]. K. Kaffes, N. J. Yadwadkar, and C. Kozyrakis, "Hermod: principled and practical scheduling for serverless functions," in Proceedings of the 13th Symposium on Cloud Computing, pp. 289–305, 2022. no pdf.

[32]. S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with OpenLambda," in 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16), 2016.

[33]. G. Aumala, E. Boza, L. Ortiz-Avilés, G. Totoy, and C. Abad, "Beyond load balancing: Package-aware scheduling for serverless platforms," in 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 282–291, 2019.