Exploring the Serverless First Strategy in Cloud Application Development

Adriano Prado Cavalheiro, Claudio Schepke
PPGES - Graduate Program in Software Engineering
Federal University of Pampa – Alegrete, Brazil
{adrianocavalheiro.aluno,claudioschepke}@unipampa.edu.br

***Abstract*—This paper explores the Serverless First strategy in cloud application development. Serverless computing has gained popularity due to its flexibility and scalability. In our work, we provide a systematic review of the literature about the Serverless paradigm in cloud computing and an evaluation of the advantages of this approach by performing a comparative analysis among three ways for the implementation of an application: AWS Lambda, AWS Lambda with Chalice framework, and the traditional form using the Flask framework. The literature review results show the gains in scaling, cost reduction, and ease of maintenance achieved with the Serverless First strategy. However, some limitations and challenges were also highlighted, such as the greater complexity of the environment, less control over resources, resource limitations imposed by the cloud provider, and difficulties in debugging and managing the infrastructure. The case study demonstrates in practice that the Chalice framework provided the most straightforward and rapid implementation, the AWS Lambda without Chalice offered greater flexibility and control, and the Flask version allowed local testing and total control but required more manual setup and lacked automatic scalability.**

***Index Terms*—Serverless First, Function as a Service, Serverless Computing, Serverless Architecture, Serverless Applications, Software Development, Cloud Computing, Cloud Services.**

I. Introduction

Cloud computing enables access to computing resources, such as servers and storage, over the Internet. It offers Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) models [1]. This revolutionary technology has transformed how businesses manage their infrastructures and deliver services to users, offering greater agility and scalability.

Cloud computing has brought a new approach to application development, allowing developers to focus on business logic without worrying about infrastructures. One emerging strategy is *Serverless First* [2], which leverages serverless functions to develop, deploy, and manage applications in a simpler and faster manner. Serverless computing, or Function as a Service (FaaS), represents a groundbreaking approach to software development [3]. It liberates developers from the burden of managing the underlying infrastructure aspects, allowing them to focus solely on writing code and implementing application logic. AWS Lambda, a prominent serverless platform, exemplifies this paradigm, enabling event-driven computing where triggering code execution occurs in response to events or requests [4].

Serverless architecture provides significant advantages, such as seamless scalability, efficient resource utilization, and cost-effectiveness [3]. With auto-scaling capabilities, applications can handle varying workloads without manual intervention. This dynamic scaling ensures optimal resource allocation, avoiding unnecessary costs during periods of low demand.

The study of the *Serverless First* strategy is relevant due to increasing its adoption. It revolutionizes application development by prioritizing serverless architecture as the primary approach [5]. This approach promotes serverless computing platforms, like AWS Lambda, as the default choice for deploying applications.

By adopting the *Serverless First* mindset, organizations can achieve accelerated development cycles, reducing time-to-market for new features and updates. Moreover, it empowers developers to build scalable, flexible, and cost-efficient applications without worrying about infrastructure management [6]. However, while the *Serverless First* strategy offers numerous advantages, it might not be suitable for all scenarios. Careful evaluation of project requirements, such as application complexity, runtime characteristics, and compliance considerations, is necessary to determine the optimal strategy.

In this context, **the objective of this paper is to investigate the Serverless First strategy**, particularly concerning reduced development time and ease of implementation. The methodology includes a systematic literature review to identify the benefits, best practices, and challenges of the *Serverless First* strategy. Subsequently, we analyze available tools for *Serverless First* development and conduct a simulated case study to demonstrate the impact of this approach on the AWS Lambda platform.

This research **contributes (a) with a systematic literature review**, providing critically analyses, considering scalability, cost reduction, and ease of implementation, and **(b) with the identification of the benefits and limitations of the approach**, providing valuable insights for technology professionals and decision-makers interested in Serverless architectures.

The organization of this paper is as follows: Section II describes the systematic literature review conducted. Section III explores related work in the area. Section VII presents the simulated case study. Section VIII discusses the results and findings of the case study. Finally, Section IX concludes the paper and outlines future perspectives.

979-8-3503-8160-3/23/$31.00 ©2023 IEEE
DOI 10.1109/SBAC-PADW60351.2023.00023

II. Systematic Review

This chapter presents the conducted protocol for a Systematic Review (SR), following the guidelines outlined in [7]. The SR aims to address three research questions: Q1, Q2, and Q3. These questions explore the benefits, tools, and implementation challenges associated to the *Serverless First* strategy in cloud application development.

A. Research Questions

- **Q1:** What are the benefits of the *Serverless First* strategy in cloud application development?
- **Q2:** What are the most efficient and suitable tools for developing applications using the *Serverless First* strategy?
- **Q3:** What are the challenges and limitations associated with adopting the *Serverless First* strategy in application development?

B. Search Strategy

We formulate a search string with related keywords and synonyms to identify relevant studies, as presented in Table I. We search in academic databases, including Scopus, IEEE Xplore, and ACM Digital Library. An initial screening was based on the title and abstract, followed by reading the selected papers.

C. Inclusion Criteria

We include studies if they met the following criteria:

1) Published in English between 2018 and 2023, emphasizing recent developments.
2) Address topics related to *Serverless First* approach, cloud application development, and software programming.

D. Exclusion Criteria

We exclude the studies if they met any of the following criteria:

1) Published in languages other than English to maintain language consistency for analysis.
2) Not fully accessible or unavailable for analysis due to the need for comprehensive evaluation.
3) Published before 2018 to prioritize recent research and align with current industry practices.
4) Lack relevant empirical evaluation related to the *Serverless First* strategy, ensuring the inclusion of substantive findings
5) Do not contain any terms related to the research topic in the title or abstract, ensuring the study's direct relevance to our investigation.

TABLE I
Terms in search string

Term	Synonym
Serverless First	Serverless approach, serverless architecture
Benefits	Advantages, positive impact, improvements
Challenges and Limitations	Obstacles, problems, restrictions, disadvantages
Efficient Tools	Effective resources, productive solutions, suitable tools
Application Development	App development, software programming

E. Quality Assessment

We evaluate selected articles for their contribution to answering the research questions. We use three labels (Yes, Partially, No) to assess the quality of the studies based on criteria related with clarity, relevance, and up-to-dateness.

F. Data Analysis and Synthesis

During the data analysis phase, we retrieve 65 articles from different sources. After applying inclusion and exclusion criteria, we select 23 papers for further examination. The extraction phase resulted in 6 relevant articles addressing the research questions in-depth. We analyse these studies thoroughly to provide insights into the *Serverless First* strategy for application development.

G. Limitations and Ethical Considerations

The SR has limitations, including the restriction to consider only English articles and, eventually, the exclusion of relevant studies from other sources. We follow ethical principles to respect copyright and provide proper citations for selected studies.

In conclusion, the SR serves as a crucial foundation for exploration of the *Serverless First* strategy, enabling us to synthesize relevant research findings and draw insights to address our research questions effectively.

III. Related Work

In this section we present the related works identified during the literature review. We briefly describe each study, highlighting its title, main topics, and original source. Table II provides the list of these studies.

A. Paper [8]: Serverless Computing

This study reviews Serverless computing, highlighting its benefits, challenges, and applications. It suggests that Serverless computing has potential to transform application development and deployment but acknowledges ongoing evolution and challenges in its adoption [8].

TABLE II
Related Work

Paper	Title
[8]	Serverless Computing: A Survey of Opportunities, Challenges, and Applications
[9]	Understanding the Challenges and Novel Architectural Models of Multi-Cloud Native Applications – A Systematic Literature Review
[10]	A Survey and Implementation on Using a Runtime Overhead to Enable Serverless Deployment
[11]	Fine-Grained Performance and Cost Modeling and Optimization for FaaS Applications
[12]	Serverless on Machine Learning: A Systematic Mapping Study
[13]	Serverless Cloud Computing: State of the Art and Challenges

B. Paper [9]: Multi-Cloud Native Applications

This study explores multi-cloud native architectures, discussing challenges into development and deployment. It favors event-based architectural models for scalable, modular development, and multi-cloud deployment [9].

C. Paper [10]: Using Runtime Overhead for Serverless Deployment

This research investigates using runtime overheads for Serverless development, addressing critical challenges. It concludes that this overheads can enhance Serverless performance and reduce development costs [10].

D. Paper [11]: Fine-Grained Performance and Cost Modeling

This study introduced a fine-grained performance and cost model for Serverless apps based on resource analysis. It demonstrates that this model can optimize Serverless app performance and reduce costs [11].

E. Paper [12]: Serverless on Machine Learning

This study discusses challenges and proposes solutions, examining Serverless computing for machine learning. It concludes that Serverless can lower costs and enhance performance for machine learning apps [12].

F. Paper [13]: Serverless Cloud Computing

This literature review covers Serverless cloud computing, discussing its benefits, challenges, and applications. It suggests that Serverless cloud computing has the potential to revolutionize app development and deployment but recognizes ongoing evolution and challenges [13].

These studies provide valuable insights, contributing to the comparative analysis of the *Serverless First* strategy's benefits, identifying efficient tools, and understanding associated challenges and limitations.

IV. Benefits of the *Serverless First* Strategy

In this section, we analyze the benefits of adopting the *Serverless First* strategy in cloud application development, drawing from the referenced studies. Table III summarizes these benefits.

A. Cost Reduction

Serverless services are generally cheaper than traditional servers, as you only pay for the resources you use. Previous studies like [8], [9], and [11] have addressed this advantage of the Serverless paradigm.

B. Scalability Enhancement

Serverless services are easily scalable up or down, allowing applications to be always available even during peak traffic. Studies like [9], [10], and [12] have discussed scalability as one of the benefits of the Serverless paradigm.

C. Development Simplification

Serverless services simplify application development by eliminating the need for server and infrastructure management. Studies like [10], [11], and [13] have addressed this advantage.

TABLE III
RQ1: Benefits of the Serverless First Strategy in Cloud Application Development

Benefits	Referenced Studies
Cost reduction	[8], [9], [11]
Scalability enhancement	[9], [10], [12]
Development simplification	[10], [11], [13]
Security improvement	[11], [12], [13]

D. Security Improvement

Serverless services improve application security as the shared responsibility between the cloud provider and the developer. Studies like [11], [12], and [13] have explored this advantage.

V. Efficient and Suitable Tools for Developing *Serverless First* Applications

In this section, we explore the tools and platforms that are efficient and suitable for developing applications following the *Serverless First* strategy, drawing from the referenced studies. Table IV summarizes these tools.

A. Amazon Web Services (AWS) Lambda

AWS Lambda is the pioneering Serverless computing platform that allows you to run code without provision or manage servers necessity. Previous studies like [8], [9], [10], [11], [12], and [13] mention AWS Lambda as an efficient and suitable tool for developing *Serverless First* applications.

B. Microsoft Azure Functions

Microsoft Azure Functions is a Serverless computing service in the Microsoft cloud that enables you to build scalable and highly available applications. Previous studies like [8], [9], [10], [11], [12], and [13] highlight Azure Functions as a relevant tool for developing Serverless applications.

C. Google Cloud Functions

Google Cloud Functions is a Serverless computing service from Google that allows you to run code in response to specific events such as database changes, file uploads, or API calls. Previous studies like [8], [9], [10], [11], [12], and [13] indicate Google Cloud Functions as a viable option for developing applications in the Serverless paradigm.

D. Serverless Framework

The Serverless Framework is an open-source framework that simplifies the deployment and management of Serverless applications. Studies like [9], [10], [11], and [12] mention the Serverless Framework as an efficient tool for developing applications in the Serverless paradigm.

E. AWS Serverless Application Model (SAM)

AWS Serverless Application Model (SAM) is an extension of AWS CloudFormation that makes it easier to create, deploy, and manage Serverless applications. Studies like [8], [9], [10], [11], and [12] highlight AWS SAM as a suitable tool for developing Serverless applications.

TABLE IV
RQ2: EFFICIENT AND SUITABLE TOOLS FOR DEVELOPING *Serverless First* APPLICATIONS

Mentioned Tools in the Studies	Referenced Studies
AWS Lambdas	[8], [9], [10], [11], [12], [13]
Azure Functions	[8], [9], [10], [11], [12], [13]
Google Cloud Functions	[8], [9], [10], [11], [12], [13]
Serverless Framework	[9], [10], [11], [12]
AWS Serverless Application Model (SAM)	[8], [9], [10], [11], [12]

TABLE V
RQ3: CHALLENGES AND LIMITATIONS ASSOCIATED WITH ADOPTING THE *Serverless First* STRATEGY IN APPLICATION DEVELOPMENT

Challenges and Limitations	Referenced Studies
Increased complexity	[9], [10], [11], [12]
Reduced control over resources	[8], [9], [10], [11]
Resource limitations	[8], [9], [10], [11], [12]
Dependency on cloud providers	[8], [9], [10], [11], [12]
Debugging difficulty	[8], [9], [10], [11], [12]

VI. CHALLENGES AND LIMITATIONS OF THE *Serverless First* STRATEGY

In this section, we discuss the challenges and limitations associated with adopting the *Serverless First* strategy in application development, drawing from the referenced studies. Table V summarizes these challenges.

A. Increased Complexity

Serverless applications can present greater complexity compared to traditional applications. Studies [9], [10], [11], and [12] highlight this complexity as a challenge in the development and maintenance of applications in the *Serverless First* strategy.

B. Reduced Control Over Resources

When using Serverless services, as mentioned in studies [8], [9], [10], and [11], you have less control over the resources you use. This limitation can be a challenge for applications that require more precise control over resources.

C. Resource Limitations

Serverless services may impose restriction on available resources, such as runtime and memory. These limitations, mentioned in studies [8], [9], [10], [11], and [12], can pose a challenge for applications that require more robust resources.

D. Dependency on Cloud Providers

Adopting the *Serverless First* strategy implies a dependence on cloud providers. Studies [8], [9], [10], [11], and [12] highlight that this dependency can be a challenge as provider policies and prices can influence the development and operation of applications.

E. Debugging Difficulty

Serverless applications can present additional challenges in the debugging process. Identifying and resolving errors in this context can be more challenging than in traditional applications. The mention of this difficulty is in the studies [8], [9], [10], [11], and [12] as a challenge associated with the development of applications in the *Serverless First* strategy.

VII. CASE STUDY

This section presents a simulated case study comparing three approaches to implement a REST API using the *Serverless First* strategy on AWS Lambda. The objective is to explore the advantages and limitations of each approach and highlight the importance of the *Serverless First* strategy in creating scalable, flexible, and easily manageable APIs.

A. AWS Lambda

The *Serverless First* strategy, utilizing AWS Lambda, presents an opportunity to explore the advantages and limitations of Serverless development in building scalable and cost-effective applications. Embracing cloud computing, particularly the Serverless paradigm with AWS Lambda, empowers organizations to build agile, efficient, and cost-effective applications that can scale effortlessly to meet the challenges of today's dynamic digital landscape.

As a leading Serverless platform, AWS Lambda plays a pivotal role in the *Serverless First* strategy [4]. It allows developers to deploy code in function forms, which automatically scale to meet demand. AWS Lambda integrates seamlessly with other AWS services, such as Amazon S3, Amazon DynamoDB, and Amazon API Gateway, enabling the creation of robust and feature-rich applications.

With AWS Lambda, developers can build highly responsive and event-driven applications. Its event-driven nature ensures that functions precisely execute when triggered, allowing applications to respond instantaneously to user interactions or system events.

B. Implementation

The implementation scenario involves developing a REST API to manage a to-do list. The requirements for the API are:

1) List all available tasks;
2) Create a new task;
3) Update an existing task;
4) Delete a task;
5) Retrieve a specific task by its identifier;

These requirements are popular in many applications that require the management of tasks or activities.

The three approaches analyzed for the REST API are:

1) Implementation using the Chalice framework (Serverless approach)
2) Implementation using only AWS Lambda (Serverless approach)

3) Implementation using Flask in a local environment (Traditional approach)

We conduct the implementation following the steps provided for each approach and analyze mainly aspects such as ease of use, deployment speed, flexibility, scalability, and cost. Additionally, we conduct a cost analysis, considering both the free period of AWS and after the 12-month free time. The *Serverless First* strategy, especially during the free period, allows for low-cost initial development and testing. In the long term, the pay-as-you-go model and automatic scalability contribute to sustained cost savings.

To view the full implementation of this case study, visit the GitHub repository[1]. The repository contains detailed documentation, including code, tables, and further analysis, providing valuable insights for researchers and professionals seeking to utilize Serverless technologies. It also highlights the importance of carefully choosing the implementation approach based on specific project needs. The analysis encourages further research and improvements in development methodologies.

VIII. Results and Discussion

A. Responses for the Research Questions

In the Systematic Review conduction, we select six relevant articles addressing the *Serverless First* strategy in cloud application development. We obtain answers to the questions, as shown in Table VI, from the analysis of these articles.

Research Question Q1, which investigates the benefits of the *Serverless First* strategy in cloud application development, had its answers identified in two of the six selected articles. These articles highlighted benefits such as cost reduction, scalability, development simplification, and improved security.

Research Question Q2, which focuses on identifying the most efficient and suitable tools for developing *Serverless First* applications, did not receive direct answers from the selected articles. However, we found relevant insights related to the tools used in Serverless Computing in all six papers. These insights include the usage of AWS Lambda, Azure Functions, Google Cloud Functions, Serverless Framework, and AWS Serverless Application Model (SAM) in various contexts of Serverless Computing. While the articles did not specifically address the concept of *Serverless First*, the tools mentioned can be relevant for developers adopting this approach in their application development process.

Research Question Q3, which addresses the challenges and limitations associated with adopting the *Serverless First* strategy in application development, had its answers mentioned in all six selected articles. The identified limitations and challenges include increased complexity, control over resources reduced, resource limitations, dependency on cloud providers, and debugging difficulties.

Furthermore, to complement the SR, we conducted a simulated case study comparing three approaches to implementing a REST API using the *Serverless First* strategy. The case study explored the advantages and limitations of each approach, including the use of the Chalice framework, AWS Lambda without Chalice, and Flask in a local environment. We analyzed aspects such as ease of use, deployment speed, flexibility, scalability, and cost for each approach.

B. Case Study Analyses

Table VII shows a comparative analyses of the three adopted approaches. Results and analysis highlight that the *Serverless First* strategy offers significant advantages, such as automatic scalability, simplified management, and cost optimization based on actual usage. The Chalice framework provides the most straightforward and rapid implementation, while the AWS Lambda approach without Chalice offers greater flexibility and control. The traditional Flask implementation allows for local testing and total control but requires more manual setup and lacks automatic scalability.

The approach using the Flask framework in a local environment offers complete control over the application, allowing testing and development of functionalities in an isolated environment before deployment to the cloud. So, developers have complete control over the implementation and can customize the application according to specific project needs. It is also possible test and debug the API locally before deploying it to the cloud. That can accelerates the development process. However, this approach also requires more effort for configuration and resource management related to Chalice and AWS Lambda, especially when the application needs to be deploy.

The case study results highlighted the benefits of the *Serverless First* strategy, such as automatic scalability, simplified management, and cost optimization based on actual usage. The Chalice framework demonstrated the most straightforward and rapid implementation, while AWS Lambda without Chalice offered greater flexibility and control. The traditional Flask

[1]https://github.com/apcavalheiro/impl_use_case_serverless_first

TABLE VI
Number of papers answering the research questions

Question	Number of Articles with Answers
Q1	2
Q2	0
Q3	6

TABLE VII
Comparison Among Approaches

Aspect	Chalice	AWS Lambda	Flask (Local)
Ease of Use	High	Medium	Medium
Rapid Deployment	High	Medium	Medium
Flexibility	Low	High	High
Automatic Scalability	Yes	Yes	No (depends on configuration)
Native AWS Integration	Yes	Yes	No
Learning Curve	Low	Medium	Medium
Simplified Management	Yes	Yes	No
Full Resource Control	No	Yes	Yes
Local Testing	No	No	Yes
Consumption-Based Pricing	Yes	Yes	No (only in production)

approach allowed for local testing and total control but required more manual setup and lacked automatic scalability.

Based on the obtained results, we can conclude that the *Serverless First* strategy offers significant benefits for cloud application development. The serverless approach allows for automatic scalability, simplified management, native integration with cloud services, and cost reduction. However, for developers it is essential to carefully consider the specific needs of their projects and select the most appropriate implementation approach based on factors such as project complexity, development team expertise, and expected scalability.

The case study also emphasizes the importance of further research and improvements in Serverless development methodologies to address challenges and limitations effectively. The SR and the case study combination contributes valuable insights for researchers and professionals seeking to leverage Serverless technologies effectively while considering the benefits and trade-offs involved.

C. Cost Analysis

After the 12 month free period, features and services used in the use case API will be subject to actual usage charges. It's important to note that the cost structure may change after the initial 12 months. Based on previous estimates of 1 million incoming API calls, 1 million incoming HTTP API calls, 1 million messages processed, and 750,000 minutes of WebSocket API connection, the total cost after 12 months would be approximately $8.95 a month. The total cost after 24 months would be approximately $17.91 per month while maintaining constant resource usage. This cost increase after 24 months is due to potential changes in pricing and discounts. It highlights the sustained savings using an efficient implementation of the Serverless pattern in the AWS framework over time.

IX. Conclusion

In this article, we've explored the *Serverless First* strategy in cloud application development, aiming to understand its benefits, challenges, and efficient tools. The *Serverless First* approach prioritizes Serverless solutions from the start of development, leveraging cloud computing platform resources. Through a systematic review (SR) II, we distilled insights from six selected articles, addressing questions about the merits, suitable tools, and hurdles of Serverless computing.

Our analysis revealed significant benefits of the *Serverless First* strategy, including cost reduction, scalability, streamlined development, and enhanced security. We identified suitable tools like AWS Lambda, Azure Functions, Google Cloud Functions, Serverless Framework, and AWS Serverless Application Model (SAM).

Despite these advantages, challenges persist, such as increased complexity, limited resource control, dependence on cloud providers, and debugging intricacies. In a case study comparing REST API implementation approaches, we confirmed the benefits of the *Serverless First* strategy, showcasing automatic scalability, simplified management, and cost optimization.

Collectively, our SR and case study provide valuable insights for researchers and professionals interested in the *Serverless First* strategy. Further research to refine Serverless development methodologies is encouraged.

In conclusion, the *Serverless First* strategy is promising for cloud application development, particularly for projects with dynamic workloads and a focus on scalability and cost efficiency. While this study offers valuable insights, it could benefit from more precise data, which we aim to address in future research. Understanding Serverless architectures' opportunities and challenges remains crucial for successful cloud application deployments as cloud technologies evolve.

We recommend in-depth investigations into the Serverless First strategy's implications in projects of varying scales and complexities, as exploring best practices for deploying and managing Serverless applications in future endeavors. Additionally, extending the analysis to other cloud providers is an attractive avenue for research, although results may vary depending on the specific provider and its offerings.

References

[1] H. Cloud, "The nist definition of cloud computing," *National Institute of Science and Technology, Special Publication*, vol. 800, no. 2011, p. 145, 2011.

[2] D. Anderson, "What is serverless-first in 2021?" The New Stack, March 2021, [Accessed 05-May-2023]. [Online]. Available: https://thenewstack.io/what-is-serverless-first-in-2021/

[3] M. Roberts, "Serverless architectures - martinfowler.com," https://martinfowler.com/articles/serverless.html, 2018, [Accessed 05-May-2023].

[4] P. Sbarski and S. Kroonenburg, *Serverless architectures on AWS: with examples using Aws Lambda*. Simon and Schuster, 2017.

[5] J. Short, C. Caron, M. Cohn, and M. Snover, "The journey to serverless first - enterprise use cases show the way," *AWS for Business*, 2021. [Online]. Available: https://d1.awsstatic.com/analyst-reports/The%20Journey%20to%20Serverless%20First%20-%20Enterprise%20Use%20Cases%20Show%20the%20Way.pdf

[6] J. Short, "The serverless first mindset," 2023. [Online]. Available: https://serverless-architecture.io/serverless-architecture-design/the-serverless-first-mindset/

[7] C. Okoli and K. Schabram, "A guide to conducting a standalone systematic literature review," *Communications of the Association for Information Systems*, vol. 37, no. 43, pp. 879–910, 2015.

[8] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: A survey of opportunities, challenges, and applications," *ACM Comput. Surv.*, vol. 54, no. 11s, nov 2022. [Online]. Available: https://doi.org/10.1145/3510611

[9] J. Alonso, L. Orue-Echevarria, V. Casola, A. I. Torre, M. Huarte, E. Osaba, and J. L. Lobo, "Understanding the challenges and novel architectural models of multi-cloud native applications – a systematic literature review," *Journal of Cloud Computing*, vol. 12, no. 1, p. 6, 2023. [Online]. Available: https://doi.org/10.1186/s13677-022-00367-6

[10] N. Saravana Kumar and S. Selvakumara Samy, "A survey and implementation on using a runtime overhead to enable serverless deployment," in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, March 2023, pp. 497–501.

[11] C. Lin, N. Mahmoudi, C. Fan, and H. Khazaei, "Fine-grained performance and cost modeling and optimization for faas applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 180–194, Jan 2023.

[12] A. Barrak, F. Petrillo, and F. Jaafar, "Serverless on machine learning: A systematic mapping study," *IEEE Access*, vol. 10, pp. 99 337–99 352, 2022.

[13] V. Lannurien, L. D'Orazio, O. Barais, and J. Boukhobza, *Serverless Cloud Computing: State of the Art and Challenges*. Cham: Springer International Publishing, 2023, pp. 275–316. [Online]. Available: https://doi.org/10.1007/978-3-031-26633-1_11