Performance Benchmarking of Serverless Computing Platforms

Article *in* International Journal of Computer Trends and Technology · June 2024

DOI: 10.14445/22312803/IJCTT-V72I6P121

2 authors:

Dhruv Kumar Seth
Walmart Stores
14 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Pradeep Chintale
SEI Investment Company
42 PUBLICATIONS **455** CITATIONS
SEE PROFILE

International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I6P121

Volume 72 Issue 6, 160-167, June 2024

Original Article

Performance Benchmarking of Serverless Computing Platforms

Dhruv Seth[1], Pradeep Chintale[2]

[1]*Solution Architect, Walmart Global Tech, California, USA.*
[2]*Enterprise Cloud Platform, SEI Investment Company, Pennsylvania, USA.*

[1]*Corresponding Author : er.dhruv08@gmail.com*

Received: 18 April 2024 Revised: 25 May 2024 Accepted: 16 June 2024 Published: 30 June 2024

Abstract *- Performance benchmarking of serverless computing platforms helps to determine the most appropriate serverless platforms for running backend services for web applications. Performance benchmarking focuses on attributes such as the Central Processing Unit (CPU) performance, network speed or performance, and the memory capacity of the server. The performance benchmarking tools include micro-benchmarking and application benchmarking tools. The micro-benchmarking tools are the focus of this paper, with Amazon Web Service (AWS) as the epitome of a cloud serverless computing platform. The performance benchmarking tools indicate that the AWS is not perfect, but it is a reliable starting point for the development and advancement of serverless computing platforms. Among the performance benchmarks reviewed, the ServerlessBench stood out due to its results that showed decreased tail latency, enhanced bursty behavior, improved image fetch speed, and improved capacity for function transfer.*

Keywords *- Serverless computing, Micro-benching tools, ServerlessBench, Amazon Web Service, Benchmarking.*

1. Introduction

The development of the Internet of Things (IoT) has led to an increase in demand for computing infrastructure. One of these computing infrastructures is serverless computing, which is a method of computing that involves the provision of backend services to a developer or user on an as-used basis [1]. Traditionally, backend server providers would provide a fixed amount of computing space, which the developers overpaid. With serverless computing, developers pay only for the computing space used [3]. This means developers do not have to worry about the computing infrastructure because they are not required to pay a fixed amount for the bandwidth needed to run the application.

Historically, developers were required to have physical hardware resources to run a server, but this was expensive. Afterwards, a cloud computing platform was developed that allowed developers to rent a fixed number of servers [13]. However, the problem arose when the developers could rent the fixed servers at a high price to ensure that the servers could handle an increase in traffic [2]. This means developers or companies paid for a server space that could not be used because the spike in traffic could not be experienced within the rented period. Consequently, cloud vendors introduced auto-scaling to address the issue of a spike in traffic. However, in the event of a Distributed Denial of Service (DDoS) attack, the developer or company ends up paying much more than the initial price.

Serverless computing, therefore, enables developers to pay as they use it, meaning they will only pay for the services used. Serverless computing can be compared to the use of a data plan that charges for each byte of data used rather than a fixed limit whereby a user may not exhaust the allocated monthly data plan [13]. The term serverless can also be understood in the sense that the developer does not need to handle the server issues because the server vendor handles them.

The advantages of serverless computing include, first, lower costs- serverless computing is less costly than traditional cloud computing, whereby the developers are allocated a fixed number of servers or server space, which may not be used within the allocated period [14]. This means the developer pays for a service that is not used. Second, serverless computing has simplified scalability because the server vendors address the scaling when required. Third, serverless computing has a simplified backend code- for example, Function as a Service (FaaS) enables developers to build functions that perform a single purpose independently, such as the Application Programming Interface (API) call. Fourth, serverless computing has a quicker turnaround than traditional cloud computing [15]. For example, developers can

get around bugs by adding and modifying code quicker than in traditional computing, which requires a complicated deployment process.

Conducting performance benchmarking of serverless computing platforms can help in understating and improving the performance of serverless applications on the cloud providers. The widely used performance benchmarking tools include microbenchmarks, which measure the aspects that affect serverless computing performance- this includes the memory, CPU, and network. Some benchmarking tools are application-centric [3]. This review, therefore, focuses on a detailed analysis of the performance benchmarking of serverless computing platforms. The research gap addressed in this article is the limited research on the impact of serverless computing platforms in the IoT. The gap between traditional cloud computing and serverless computing platforms that are also addressed in the article.

2. Performance Benchmarking

2.1. Background

Benchmarking in computing is used to make a comparison in the performance of computing tools, techniques, and computer systems [13]. A benchmark, therefore, is a test of a system to determine its performance. The following characteristics and criteria are required for successful benchmarking. First, relevance- the benchmark should be relevant and applicable to a specific computing area. The benchmark should also consider the context of the consumer of the results [5]. Studies show that scalability is often a challenge for relevance- this is because scalability can only be achieved if a benchmark runs on a broader computing system and runs simulations with real applications. Therefore, developers usually focus on a narrow applicable area.

Another characteristic of benchmarking is reproducibility, which is the notion that a benchmark should produce consistent results. However, modern software systems are varying, which makes it difficult, if not impossible, to achieve perfect reproducibility [1]. To improve reproducibility, developers should run a benchmark for a relatively long time to allow the inclusion of all the variable behaviors. Also, developers are required to run the systems many times, which contributes to an improvement in consistency.

Moreover, benchmarking of computing systems requires verifiability, which is the ability to use the benchmark for verification purposes Verification of benchmark results helps in improving the trustfulness of the benchmark results [20]. This means developers must provide detailed information about the computing system to enable the provision of accurate, verifiable data. Lastly, benchmarking should be fair- this means artificial constraints should not interfere with the comparison of the systems on their metrics [13]. To improve fairness, developers should use consensus to design benchmarks. A panel of experts should come up with the benchmarks instead of sourcing them from individual parties.

This section covers the application characteristics of a serverless application and examines the number of functions that an application is built and the cloud services that the application runs. Currently, AWS Lambda dominates the platform for serverless applications. Currently, about 80 percent of the developers have adopted the AWS Lambda [13]. Therefore, AWS Lambda is widely studied because it is the first serverless vendor.

The traffic patterns of a serverless function determine the workload characteristics. For example, the number of times that functions run and the volume of the workload. For example, if the workload is bursty, the functions will take considerable time to run. If there is a high volume per request, the resource use will be stretched. The researchers surveyed 67% of the serverless functions and found that they are running short, which means they are not able to handle the data volume per request efficiently [13]. The documented running time is milliseconds or seconds

3. Serverless Benchmarking

3.1. Micro-Benchmarking

Micro-benchmarking involves the use of a single function to measure the individual characteristics of a server function. These characteristics or attributes include the Central Processing Unit (CPU), memory, disk Input-Output (I/O), and network performance [16]. For instance, a single Amazon Web Services (AWS) Lambda function can be used to implement a handler that generates a parameter and calculates the latency using a floating point [13]. Consequently, the micro-benchmark helps a developer determine the latency for an operation that consumes too much of the CPU resources. Moreover, a function-bench is also a micro-bench function that uses a single function to download and upload objects to measure the performance of the network.

3.2. Application Benchmarking

Application benchmarking involves the use of applications to measure the end-to-end response time on serverless components. For instance, the use of an e-commerce benchmark such as the BeFaaS to implement a webshop [16]. Furthermore, an Image Processing application can also form an application benchmark- this involves fetching an image from a storage area and applying varied effects on it then uploading it to another storage area. The latency time is calculated to determine the effectiveness of the image upload, filtering, and sharing on the serverless storage.

3.3. Benchmarkers in Serverless Computing

Serverless computing requires standardization of benchmarking tests to make them reducible and automatic. The following are the core elements for the serverless benchmark frameworks: first, built-in benchmarks- these are

used to demonstrate features of the tool and to provide examples of creating and integrating benchmarks. Second, the deployment tool is used for making a standard benchmark to be used by network cloud providers. Lastly, the load generation automation which helps developers in load configuration [6].

The SPEC-RG CLOUD created the serverless benchmarker (SB) to arrange serverless benchmarking that is reducible. In this regard, the users of a serverless benchmark can design a workload regardless of the complexity of the commands executed [8]. The following figure (1) shows an overview of high-level serverless benchmarker architecture.

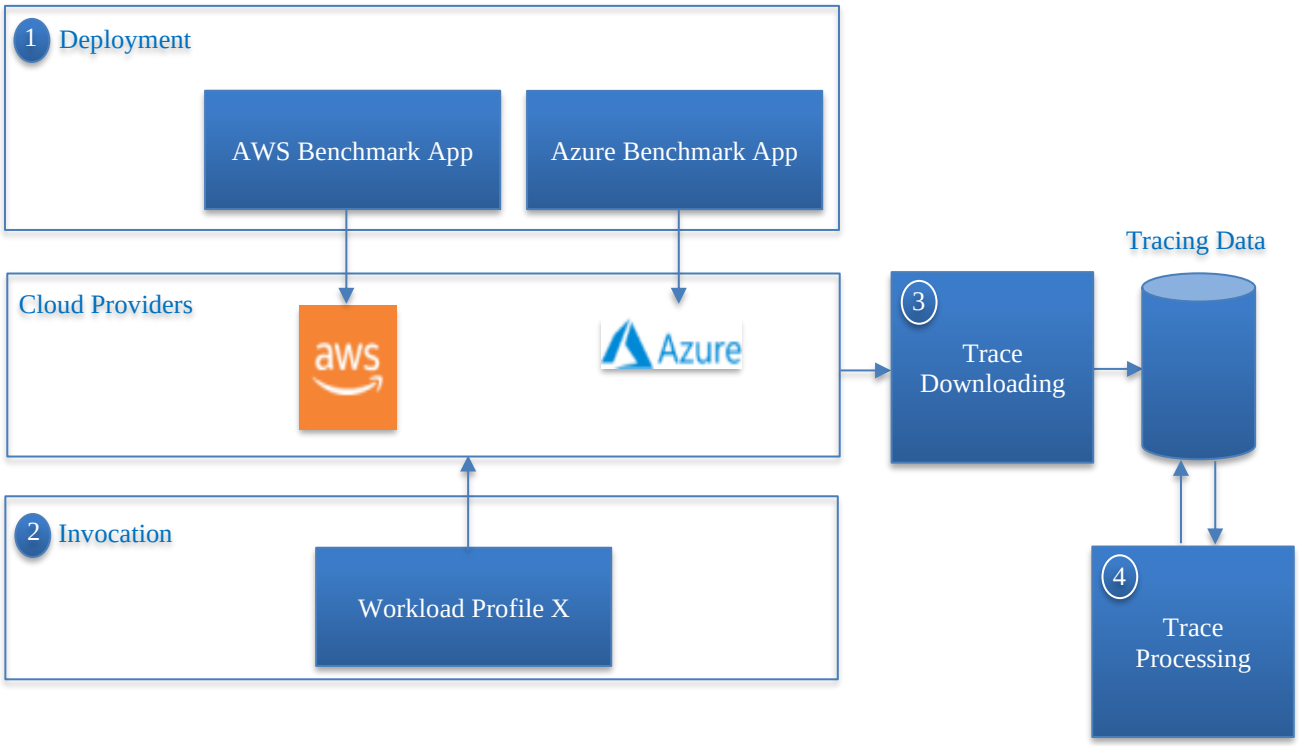

Fig. 1 An overview of a high-level serverless benchmarker architecture [13]

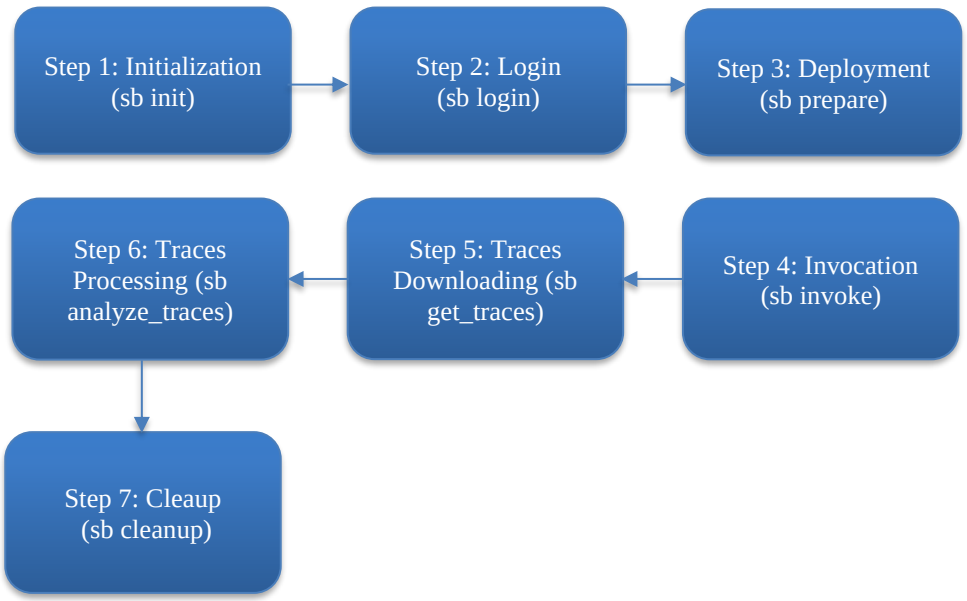

Fig. 2 Step-by-step workflow in the SB [13]

The figure 1 has four main features: deployment component, cloud providers, traces, invocation, and trace processing. The function of the deployment feature is to automate the benchmark apps- from the figure, examples of the deployment apps include the Amazon Web Service (AWS) app and the Azure Benchmark app. The serverless benchmark creates reproducible deployment using Docker, a containerization technology- this is achieved by diverting the dependencies required for the deployment of the application benchmark [13]. The serverless benchmark, through Docker technology, can also mount application codes automatically, thus simplifying the deployment.

The function of the invocation component is to provide an interface required to configure the workload and generate the load. The serverless benchmark enables the generation of the load automatically by integrating the K6 and a load testing tool. The tool undergoes the optimization process to invoke less consumption of resources and generate a good or enjoyable experience for the developer [12].

The function of the trace downloading feature is to provide a template and integrate tracing tools to enhance the downloading of data in the cloud providers. The trace downloading ensures that the required software development kit (SDK) and the application programming interface (API) are downloaded. Finally, the trace processing features are used to handle custom logic [21].

The figure 2 shows a step-by-step workflow in the SB. In Figure 2, the step-by-step workflow in the serverless benchmark is illustrated in conjunction with the command line interface. During the initialization phase, the serverless benchmark installs the required features. The login phase involves user authentication. The deployment phase involves the preparation of the serverless benchmark for deployment [19]. The deployment of the benchmark leads to the invocation step, followed by data tracing and downloading then the serverless benchmark analyzes the traces. Finally, the serverless benchmark initiates data cleanup.

The first step is the initialization which involves the serverless benchmark- this involves the initialization of default configuration and packages that had been pre-installed. When required, developers can use the initialization step to install custom packages [18]. However, developers should execute this step only if they do not see any change in the dependent packages.

The second step is the log in whereby the serverless benchmark authenticates the users in the cloud platforms. This involves temporary storage of the credentials. For example, the Amazon Web Service (AWS) and the Azure login are supported with a single sign-on (SSO) out-of-box. This means users can log in to several applications with one set of credentials.

However, since the credentials are supported out-of-box, they can be accessed by other unauthorized users. Therefore, during the serverless benchmarking process, the SSO ensures that the authentication process for users is streamlined. A perfect example of SSO would be once a user logs into Gmail, and the user is automatically authenticated into other Google services such as AdSense and Google Analytics [15]. Back to the serverless benchmarking activity in the second step, the Amazon Web Service credentials expire in twelve hours, while the Azure logins expire in 60 minutes. Therefore, serverless benchmarking should consider the login expiration of each application and perform the benchmarking process before the expiry of the session.

The third step is the development which involves the deployment configuration of the benchmark application. However, the technologies used can lead to different configuration sessions. For example, the runtime can be different for different benchmark applications. Also, different infrastructures can lead to different run time. In step three, the serverless benchmark enables the developer to build Docker images, and container loading with variables [17].

The fourth step is invocation, which involves the K6 configuration- this is a default configuration file that allows a developer to edit or create a new file. The K6 configuration often overrides the configuration flag options. The fifth step is to trace downloading- this involves the standardization of the operation of download traces from the available cloud providers. The default implementation exists, but the differences in the instrumentation and application can force the users or developers to modify the downloading logic. For example, for the Amazon Web Service (AWS), developers unify the tracing data into a single JavaScript Object Notation (JSON) file, which is an effective way of transmitting data in web applications. For instance, the JSON file enables a user to send data from the server to the client. A user requires AWS X-Ray SDK to download X-Ray traces. This is because the AWS X-Ray SDK is used to fetch the JSON file. In the implementation of the Azure app, developers trace data through three categories- this is request, trace, and dependences [11]. Fetching data through Azure is implemented in various ways- this includes the Continuous Export and the representational state transfer (REST) API. While the web APIs rely on multiple communication and the JSON, REST APIs apply the hypertext transfer protocol (HTTP) to send and receive data.

The fifth step is the traces processing which involves pre-processing of data from step 4 (traces downloading). In step 5, the final trace breakdown is generated and extracted before being applied to analysis during post-experiments. The sixth step is the cleanup which involves the destruction of all the resources in the cloud platform.

4. Analysis of the Benchmarking Tools

The reviewed studies show that the current serverless computing does not meet the required demand. For example, the run time is relatively low and the data volume requests are overwhelming. This calls upon software developers to consider more efficient avenues. For example, studies show the constraint of serverless computing in the manner that the lifetimes of the functions are limited and the hardware resources are not specialized. This means developers are gambling with serverless computing and benchmarking would help in improving the performance of these servers. Also, research indicates that the processing of data in serverless platforms is inefficient- this is attributed to the input-output problems that arise from the low network bandwidth and low Central Processing Unit (CPU) storage.

In addition, research shows that wrong policies hinder the full adoption of serverless computing platforms. For example, the policies lead to the incorporation of request queuing that is inadequate. Consequently, several serverless application developers have resorted to the measurement of the performance of the serverless computing platforms. The performance measures include cold start latency [2], lifetime of the function [6], idle time reached before the server shuts down [7], and the usage of the Central Processing Unit [11]. However, since the experiments did not have control over other circumstances or variables, the results cannot be replicated and are therefore irrelevant in the benchmarking of the current serverless computing platforms.

Furthermore, studies show that breadth is an issue of concern in the s computing platforms. For instance, the study in [1] examined the function images of up to 15MB, while the study in [13] went up to 230MB and found different results. Similarly, in [14], the researchers evaluated the delay in the image fetch function for images of up to 70MB and found a significant image delay in serverless bench. Moreover, there is a lack of depth for the analysis of the serverless bench inter-function transmission latency [13]. In the previous studies, the payloads of up to 50KB are considered normal, which is inconsistent with other studies which considered normal payloads to be 1GB. [14].

In another experiment, the researchers explored cold latency, network, and CPU performance [2]. Researchers in [13] used a similar methodology but focused on tail latency, variability in the upload of images of various sizes, and serverless bench behavior. While the experiments were identical, the authors in [13] failed to replicate the same results as in [3], indicating that the previous studies are outdated. For example, the experiment in [17] showed an increase in the slowdown of CPU in Amazon Web Service (AWS) since the year 2018 [18]. The slowness of the CPU performance in the Amazon Web Service is attributed to the update in vendor policies. The outdated of some research papers in the serverless computing platforms is reflected in the tenancy of a virtual machine (VM), which is not currently applied because of the exclusion of the VM co-residency from the AWS. This means the vendors assigned only one function to the micro-virtual machine [19].

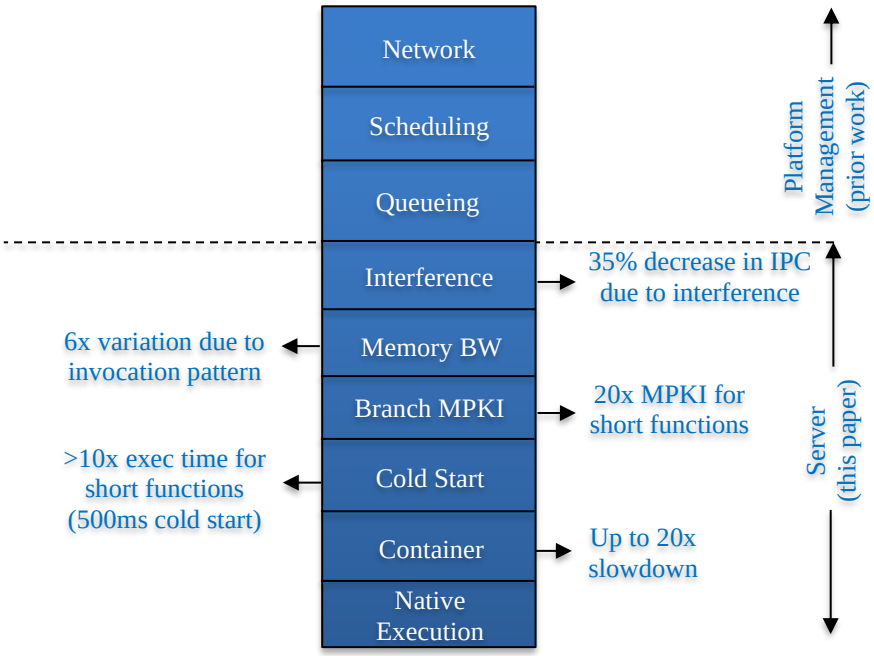

Fig. 3 Server-level overhead of FaaSProfiler [12]

The previous studies' methodologies focused on end-to-end issues. In [20], the function-as-a-service profiler (FaaS) is emphasized, which is an open-source. This means it is possible to perform a higher degree of introspection and interference with inter-function [8]. The studies also found consistent results concerning the cold starts in functions and slowdowns in containers.

The Figure 3 illustrates the server-level overhead of FaaSProfiler. In Figure 3 above, the serverless benchmark shows inter-function interference, which negatively affects its performance. In addition, the figure under-performance of other parameters in the serverless benchmark. For example, there was a 35% decrease in the Inter-Process Communication (IPC), a 6 times variation in the memory bandwidth due to the invocation pattern, a 20 times Managed Public Key Infrastructure (MPKI) for short functions, more than 10 times execution time in the cold start for the short functions, and up to 20 times slowdown in container performance [13].

In [16], the researchers considered statistical accuracy like other studies that prioritized statistical soundness. For example, LANCET is a self-correcting tool used to measure latency [15]. LANCET employs statistical measuring techniques such as Pearson and Anderson Darling Test [16]. Other tools, such as the ServerlessBench [17], offer insights into how the developer can derive economic benefits from the serverless computing platform.

The researchers in [13] used a micro-bench framework to examine the aspects of the function-as-a-service (FaaS) platform. However, a developer can adopt the BeFaaS which is specialized and has in-built benchmarks such as the IoT application and e-commerce. Another applicable framework is the SeBS, which includes image recognition engines. Nonetheless, the platforms that emphasize the use of visuals and the analysis of programming languages include the FaaSDom, which is built in and supports seven languages. However, since the FaaSDom outsources its latency from the wrk2, there is a compromise on the precision [15].

WRK2 is a concurrency model that distributes the number of connections evenly. This means the number of CPUs should equal the number of connections.

For example, thread 1 is created for clients 0, 1, and 2. Thread 2 is created for clients 3, 4, and 5.

In [13], the authors examined the benchmarking tools against their characteristics and summarized them as shown in the figure below.

Figure 5 above shows the micro-benching contributions to the serverless computing platform. The ticks where the benchmarking tool is effective, while the x indicates where the benchmarking tool is ineffective.

Benchmarking Tool	Tail Latency	Bursty Behavior	Image Fetch	Function Transfer
Research by [13]	✓	500 reqs.	230MB	1GB
Serverless-Bench [1]	✓	X	72.6MB	50KB
FaaSprofile [4]	X	X	X	X
FaaSDom [19]	X	X	X	X
BeFaaS [13]	X	X	X	X
Function-Bench [17]	X	X	X	X

Fig. 4 Micro-benching contributions to the serverless computing platform

The review of existing literature on the performance benchmarking of serverless computing platforms shows a lack of tail latency in most research papers. Also, the review of the literature shows that the experiments are too narrow and lack the statistical soundness to make informed decisions about the effectiveness of the benchmarking tools. In [13], the researchers analyzed tail latency on 99^{th} percentile server latencies. The bursty behavior examined the workloads considered bursty, but the researchers did not consider the incidences of concurrent requests. The researchers also analyzed the image delay by examining the cold start performance. Finally, the study examined a function transfer to determine the transfer speeds in serverless computing platforms.

In [8], they evaluated the performance of the serverless platforms of Amazon, Google, Microsoft, and IBM using a benchmarking test suite. The authors developed seven tests to benchmark the cloud serverless computing platforms- this includes (1) scalability, also known as latency and throughput, (2) memory, (3) CPU performance, (4) payload size, (5) programming language, (6) resource management and (7) the use of platform overhead. The researchers developed software that deployed the test code and concluded that the benchmark tools help developers identify the serverless aspects that need improvement.

Analytical performance models can be used for benchmarking the performance of serverless computing platforms [13]. The primary purpose of these analytical models is to determine the strengths and weaknesses of the serverless computing platforms. For example, the developers can determine whether the serverless computing platform can

handle the volume of requests submitted or the time taken for a function to be executed. While a serverless computing platform cannot meet 100% of the strengths required, it can have enough tools to handle a peak in traffic. Furthermore, serverless computing platforms are flexible in the manner that their resources can increase to meet increased demand.

The greatest strength of serverless computing platforms is the ability to detect the workload and deploy the required resources. This provides a better cost and performance for the user, unlike traditional cloud computing, which offers a fixed amount of space at a certain cost. The researchers also emphasized the need for users or developers to have extensive knowledge of serverless computing platforms to execute deployments effectively. The study showed that the analytical performance model can calculate the response time, chances of cold start, and the number of functions in a steady state.

5. Summary of the Findings

Cloud serverless computing is a growing technology that has not yet been embraced by a majority of cloud vendors, developers, or users. The serverless computing platform works on the idea that the developers do not need to pay for the cloud computing services they do not use. For example, the purchase of the bandwidth that one does not use is not only a waste of resources but also a waste of time. Therefore, the cloud serverless computing platform offers developers solutions by handling the backend services depending on the resources needed. This means developers will pay only for the resources used. This means the payment for the serverless computing platform will vary depending on the needs of the users. For example, in case there is a traffic increase to a web application, the serverless cloud computing platform will adjust to accommodate the traffic rather than shouting down as is the case of the cloud server computing platforms.

The advantages of cloud serverless computing platforms include: 1) cost efficiency- this means the developer or company will pay for the resources used instead of the resources purchased, as is the case for the traditional cloud server computing platforms. Also, the developer saves the cost of hosting the backend because there is no payment for the server space that is idle. 2) the cloud serverless computing platform has operational efficiency by simplifying him management of tasks. 3) The cloud serverless computing platforms are scalable. The disadvantages of serverless computing, however, include performance issues whereby a function enters a dormant state when it is not used for a certain period. Another disadvantage of serverless cloud computing includes limited flexibility and control due to the used infrastructure, and Operating System.

The review of literature addressed the problems or disadvantages of cloud serverless computing platforms by use of benchmarking tools. Several benchmarking tools were identified. This includes micro-benchmarking and application benchmarking tools [12]. Micro-benchmarking tools focus on serverless attributes such as the CPU, memory, disk I/O, and network performance [13]. The application benchmarkers include the BeFaaS.

6. Conclusion

The analysis of benchmarking tools shows that serverless computing is far from meeting the required demand. While serverless computing is cost-effective, it is replete with performance problems which include low run time and overwhelming data volume requests. Moreover, the I/O problems contribute to the inefficiency in the processing of data by serverless computing platforms. However, the ServeerlessBench in [13] showed that the cloud serverless computing platform is effective due to the reduced tail latency, enhanced bursty behavior whereby the server can handle 500 requests in a millisecond, a maximum image fetch of 230MB, and the function transfer of 1GB. The findings indicate that for developers to optimize the performance of serverless computing platforms, they must have the technical knowledge of the architecture and the performance benchmarks required to test the reliability of the platform. The serverless computing platform is the future of computing. It is only a matter of time before all developers will migrate from the traditional cloud servers to the serverless computing framework.

References

[1] Jithin Jude Paul, "Serverless Data Platforms," *Distributed Serverless Architectures on AWS*, pp. 75–93, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Maddie Stigler, "Understanding Serverless Computing," *Beginning Serverless Computing*, pp. 1–14, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[3] N. Saravana Kumar, and Samy S. Selvakumara, "Serverless Computing Platforms Performance and Scalability Implementation Analysis," *2022 International Conference on Computer, Power and Communications (ICCPC)*, Chennai, India, pp. 598 – 602, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Victor Juan Exposito Jimenez, and Herwig Zeiner, "Serverless Cloud Computing: A Comparison Between 'Function as a Service' Platforms," *7th International Conference on Information Technology Convergence and Services*, Vol. 5, 2018. [CrossRef] [Google Scholar]

[5] Francisco Carpio, Marc Michalke, and Admela Jukan, "Engineering and Experimentally Benchmarking a Serverless Edge Computing System," *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021, pp. 1-6. [CrossRef] [Google Scholar] [Publisher Link]

[6] Karim Djemame, Daniel Datsev and Vasilios Kelefouras, "Evaluation of Language Runtimes in Open-source Serverless Platforms," *Proceedings of the 12th International Conference on Cloud Computing and Services Science*, vol. 1, pp. 123-132, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jorn Kuhlenkamp et al., "Benchmarking Elasticity of FaaS Platforms as a Foundation for the Objective-Driven Design of Serverless Applications," *SAC '20: Proceedings of the 35th Annual ACM Symposium on Applied Computing*, New York, USA, pp. 1576-1585, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Pérez González, Alfonso María, "*Advanced Elastic Platforms for High Throughput Computing on Container-Based and Serverless Infrastructures*," PhD Thesis, Polytechnic University of Valencia, pp. 1-161, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[9] Sumanth Tatineni, "Performance Evaluation of Serverless Computing Platforms in Cloud Environments," *International Journal of Science and Research (IJSR)*, vol. 12, no. 11, pp. 1013–1020, 2023. [CrossRef]

[10] Nima Mahmoudi, and Hamzeh Khazaei, "Temporal Performance Modelling of Serverless Computing Platforms," *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*, pp. 1-6, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Junfeng Li et al., "Understanding Open Source Serverless Platforms," *Proceedings of the 5th International Workshop on Serverless Computing*, pp. 37-42, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] D.M. Naranjo Delgado, "*Serverless Computing Strategies on Cloud Platforms*," Thesis, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Horacio Martins, Filipe Araujo, and Paulo Rupino da Cunha, "Benchmarking Serverless Computing Platforms," *Journal of Grid Computing*, vol. 18, pp. 691–709, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Junfeng Li et al., "Understanding Open Source Serverless Platforms: Design Consideration and Performace," *Proceedings of the 5th International Workshop on Serverless Computing*, pp. 37-42, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] Bernd Strehl, The Largest Benchmark of Serverless Providers, Medium, 2018. [Online]. Available: https://medium.com/elbstack/the-largest-benchmark-of-serverless-providers-ac19b55750f4

[16] SPCL/Serverless-Benchmarks, SeBS: Serverless Benchmarking Suite. [Online]. Available: https://github.com/spcl/serverless-benchmarks

[17] Hima Govind, and Horacio GonzaleznVelez, "Benchmarking Serverless Workloads on Kubernetes," *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021. doi:10.1109/ccgrid51090.2021.00085 [CrossRef] [Google Scholar] [Publisher Link]

[18] Devesh Tiwari, "Bringing Serverless Computing to the HPC Community," *Proceedings of the 2nd Workshop on High Performance Serverless Computing*, pp. 1-2, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Mubashra Sadaqat, Mary Sánchez-Gordón, and Ricardo Colomo-Palacios, "Benchmarking Serverless Computing: Performance and Usability," *Journal of Information Technology Research*, vol. 15, no. 1, 2022. [CrossRef] [Publisher Link]

[20] Samuel Ginzburg, and Michael J. Freedman, "Serverless Isn't Server-less," *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*, pp. 43-48, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Surya Chaitanya Palepu et al., "Benchmarking the Data Layer Across Serverless Platforms," *Proceedings of the 2nd Workshop on High Performance Serverless Computing*, pp. 3-7, 2022. [CrossRef] [Google Scholar] [Publisher Link]