DOI: 10.2478/seeur-2025-0062

MEASURING FUNCTION-AS-A-SERVICE PERFORMANCE IN DIFFERENT CLOUD PLATFORMS USING AN APPLICATION-LEVEL BENCHMARK

Blend Hamiti, PhD (c),
Faculty of Contemporary Sciences and Technologies
South East European University
Tetovo, North Macedonia
bh29568@seeu.edu.mk

Besnik Selimi
Faculty of Contemporary Sciences and Technologies
South East European University
Tetovo, North Macedonia
b.selimi@seeu.edu.mk

ABSTRACT

This paper extends earlier work assessing the performance of serverless computing platforms, or Function as a Service (FaaS), across major public cloud providers, Google Cloud Platform (GCP), Azure, and Amazon Web Services (AWS), using an application-level benchmark. It aims to reproduce previous results, observe possible performance changes over time, and improve the experimental set-up for a fairer comparison of the cloud service providers. The benchmark application simulates a common user account creation process, including both synchronous and asynchronous tasks. By collecting timestamps from different stages of the workflow, the study measures response times, execution durations, and variation across runs.

The findings show that function execution is generally consistent across platforms, but the triggering of functions is more variable. Azure functions, in particular, demonstrate longer and less stable trigger times compared to AWS and GCP. By contrast, AWS and GCP provide faster and more predictable triggers, though with some differences in asynchronous behavior. The results confirm the earlier study's conclusions to a large extent, while also offering new insights thanks to the adjustments in methodology. These findings are relevant for organizations that need to decide which platform to use for serverless applications and also

contribute to longer-term benchmarking of FaaS performance. Future improvements could include extending the benchmark to additional workflows and platforms, as well as increasing automation in deployment and data collection.

Keywords: Serverless Platforms, Cloud Infrastructure Evaluation, Function as a Service (FaaS), Benchmark Design, Performance Comparison.

INTRODUCTION

Cloud computing is a paradigm that enables users to access and utilize computing resources, such as applications, data storage, and communication networks, via remote cloud-based infrastructures. A 2023 report by Flexera reveals that an overwhelming 96% of surveyed organizations globally have adopted the use of a public cloud for their data storage and processing needs (Flexera, n.d.).

Cloud computing is organized into various service models, each offering a different level of service abstraction. Traditionally, these include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). More recently, a new model called Function as a Service (FaaS), or the serverless model, has emerged that allows developers to deploy lightweight, event-driven functions that run without maintaining state. This model bills users based on their actual usage instead of the resources allocated and involves packaging applications into containers that are executed on demand.

FaaS comes with its own set of performance-related challenges due to its architecture. These include issues like delays in starting up (cold starts), variability in the infrastructure across different functions, and diversity in the event sources triggering these functions. The goal of this paper is thus to conduct a comprehensive performance evaluation of FaaS across various cloud service providers (CSPs), using an application-level benchmark.

In performance evaluation, micro-benchmarks target specific facets of a platform, such as file I/O operations, CPU performance, and network speed. These benchmarks are focused and detailed. On the other hand, application-level benchmarks provide a broader evaluation, assessing overall application performance by metrics such as response times. We consider the latter approach to be more representative of real-world conditions, offering a better understanding of a platform's overall capabilities.

The paper's findings provide valuable guidance for organizations in choosing the best Cloud Service Provider (CSP) for their serverless applications. Moreover, these results serve as a reference for tracking the performance of various CSPs over time, enabling a comparative analysis.

MOTIVATION

The motivation for this paper is based on the thesis by Hamiti, which measured the performance of FaaS across three major cloud platforms: Google Cloud Platform (GCP), Azure, and Amazon Web Services (AWS) (Hamiti, 2023). In that study, Hamiti used an application-level benchmark to evaluate the consistency and performance of FaaS.

Building upon Hamiti's work, this paper aims to replicate and extend the original experiment, incorporating suggested improvements to enhance the study's accuracy. More specifically, we fix a critical issue identified in the experimental setup related to the Azure function trigger, that is expected to provide a fairer evaluation of performance across the various cloud platforms. Moreover, by revisiting the experiment, this paper seeks to verify the reproducibility of the initial findings and explore any variations in FaaS performance over time,

considering the rapid evolution of cloud technologies.

RESEARCH QUESTIONS

In this study, we re-state the hypotheses from the thesis of Hamiti to maintain continuity and enable direct comparison (Hamiti, 2023):

- **H1**: Performance of each CSP varies across successive runs
- **H2**: Performance of CSPs differs from one another.

RELATED WORK

In the thesis by Hamiti, they use an application that simulates the processes of creating a user account in a typical system to measure performance of different platforms. The application was deployed in GCP, Azure, and AWS, and made use of various cloud services to perform its functionalities (Hamiti, 2023). In their study, they measured both the consistency of performance within each CSP and the performance variations between CSPs, using FaaS performance as a focus point. They found that while execution times of functions were highly consistent across all platforms, the consistency of function triggers varied, being moderate to low. In terms of performance variation, they observed similar execution durations for functions across platforms, but trigger performance differed significantly. Overall, the study noted that Azure's performance was two to four times inferior compared to AWS and GCP, particularly in trigger response times.

McGrath and Brenner's tool evaluated platform overhead for functions that return immediately (McGrath & Brenner, 2017). The study found similar overhead for Azure Functions and AWS Lambda, but significantly higher overhead for GCP Cloud Functions. Their method involved invoking functions synchronously through HTTP triggers using virtual machines within the same geographic region, aiming to minimize network latency. Nevertheless, it remains unclear why they opted not to leverage platform logs for their analysis.

Zhang et al. investigated the impact of memory configurations on the execution duration and cost of FaaS functions, focusing on AWS and GCP (Zhang, Zhu, Zhang, & Liu, 2019). Their findings revealed that AWS consistently had shorter execution times across various memory settings. On the other hand, GCP demonstrated more consistent function execution times, regardless of the memory configuration. The authors also noted that the execution duration for AWS varied throughout the day, attributing it to the heterogeneous nature of the underlying infrastructure.

In the study by Ivan et al., they explored performance, cost, and scalability by deploying a complete Web application in both virtual machines and FaaS environments (Ivan, Vasile, & Dadarlat, 2019). This application involved several endpoints and types of persistent storage and was implemented on AWS and Azure using their respective FaaS services. Their findings highlight that AWS's response times were consistently half as long as those observed in Azure.

Manner et al. investigated function cold start times in Azure and AWS, focusing on the overhead difference between cold and warm executions (Manner, Endreß, Heckel, & Wirtz, 2018). Their study found AWS functions to have a consistently lower cold start overhead compared to Azure functions.

Lee et al. conducted a study to assess the elasticity of different FaaS platforms, including GCP, Azure, and AWS. (Lee, Satyam, & Fox, 2018). The research involved distributed data processing tasks that were invoked concurrently. The findings indicated that

AWS exhibited the least concurrency overhead, particularly in terms of CPU usage, file input/output, and network bandwidth.

RESEARCH METHOD

Our study closely follows the methodology used by Hamiti (Hamiti, 2023), employing the same application for benchmarking. The application is designed to mimic the user account creation process in a standard application. It involves two main procedures. The first is a synchronous process where a user's submission of a web form leads to a database entry and an image uploaded to persistent storage. This process concludes with confirmation for the account creation. The second, an asynchronous process, involves creating a thumbnail for the newly uploaded image and storing it as well.

The implementation uses services like API management, FaaS, object storage, and RDBMS. The application structure is depicted in Figure 1. The benchmark is organized around two FaaS functions, namely CreateUser, which is a synchronous function, and ProcessImage, an asynchronous function. Moreover, the implementation spans GCP, Azure, and AWS. The study continues with these three cloud platforms due to their sustained prominence in the field (Flexera, n.d.).

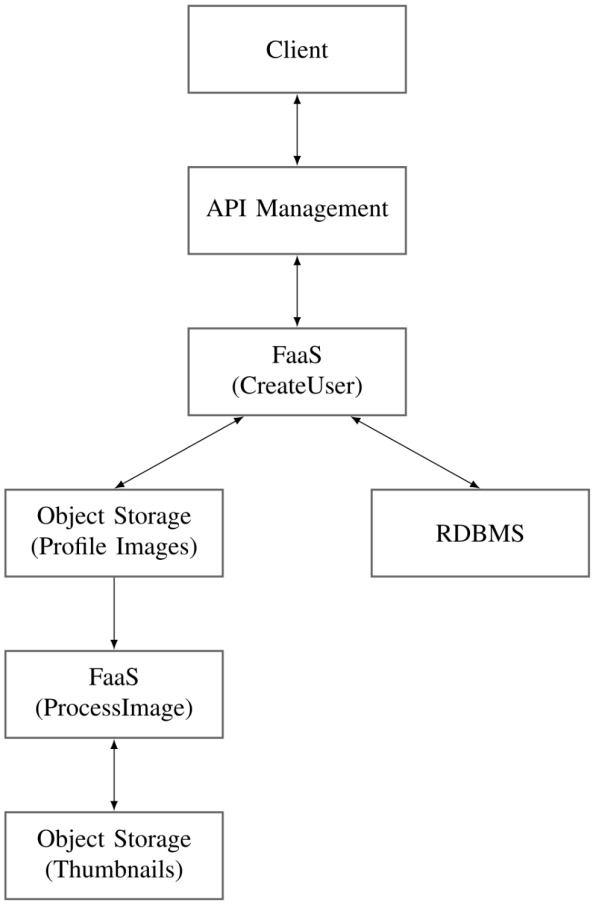

Figure 1: Application structure.

The benchmarking approach involves generating 100 measurements for each application process. These measurements are derived from a series of key timestamps representing different stages in both synchronous and asynchronous processes. They are as follows:

- t_1: Client sends a request to the API, starting their synchronous process.
- t_2: API gateway receives the request, initiating the platform's synchronous process.
- t_3: Execution of CreateUser function begins.
- t_4: Profile image upload.
- t_5: CreateUser function execution completes.
- t_6: API gateway receives a response, ending its synchronous process.
- t_7: Client receives a response, concluding their synchronous process.
- t_8: ProcessImage function starts, beginning the asynchronous process.
- t_9: Thumbnail image upload.
- t_{10}: ProcessImage function execution ends, concluding the asynchronous process.

We then calculate specific time intervals t_{ab} between these timestamps to analyze different aspects of the processes. These intervals capture the duration of the synchronous process from the platform's perspective, the total time observed by the client, and the individual times for sending the request, executing the function, and receiving the response. Additionally, we examine the time intervals related to the asynchronous process, such as the duration between image upload and processing.

A simulated client application is employed to initiate the application processes. To mitigate the impact of concurrent executions on our measurements, we schedule benchmark runs with a 10-second interval. Finally, we extract timestamps from the logging services of each platform and store them into a CSV file, formatted similarly to the results in Hamiti's study (Hamiti, 2022).

RESULTS

We calculate the mean, standard deviation, and coefficient of variation (COV) for various durations derived from the timestamps, as defined in the Research Method section, for each platform. These values are shown in Table 1. The mean and standard deviation values are in seconds, while the COV is shown as a percentage.

Table 1: Mean, standard deviation (STD), and coefficient of variation (COV) of the durations obtained from the timestamps described in the Research Method section, presented per platform, in seconds.

	AWS			Azure			GCP		
	Mean	STD	COV	Mean	STD	COV	Mean	STD	COV
t_1t_7	2.39	0.75	31.5	1.81	0.29	16.2	1.48	0.12	8.6
t_2t_6	0.56	0.06	11.5	1.35	0.29	21.4	0.47	0.05	11.3
t_1t_2	1.75	0.74	42.3	0.45	0.02	4.8	0.93	0.07	7.9
t_6t_7	0.06	0.07	117.9	-0.00	0.01	1248.8	0.07	0.08	108.5
t_2t_3	0.06	0.01	17.5	1.14	0.27	24.4	0.02	0.00	31.0
t_3t_5	0.51	0.06	12.4	0.20	0.06	32.7	0.45	0.05	12.0
t_5t_6	-0.00	0.00	80.8	0.01	0.09	871.3	0.00	0.00	45.8
t_4t_8	1.14	0.29	25.7	1.36	4.22	309.9	0.41	1.21	291.2
t_8t_{10}	0.42	0.05	11.9	0.11	0.03	29.1	0.98	0.12	12.4
t_4t_9	1.56	0.30	19.6	1.47	4.22	285.7	1.40	1.24	88.1

In addition, we compute the cumulative distribution function (CDF) of the defined durations. The results are divided into three figures for clarity: synchronous process (Fig. 2), triggering and execution of the CreateUser function (Fig. 3), and triggering and execution of

the ProcessImage function (Fig. 4). In each graph, time is plotted along the x-axis in seconds, and the cumulative probability is plotted on the y-axis, with the title indicating the duration that the plot represents.

DISCUSSION

We examine our study's results in relation to each hypothesis, and compare our results with those achieved by Hamiti, as well as other studies that offer a direct comparison for our results from the Related Work section (Hamiti, 2023). This comparison not only serves to validate and contrast our findings but also provides an opportunity to identify any emerging trends or insights in FaaS performance. In addition, we discuss the process and challenges faced in replicating Hamiti's experiment, and further potential improvements.

In our discussion, we refer to CreateUser as the function with an HTTP trigger, and ProcessImage as the function with an event-based trigger.

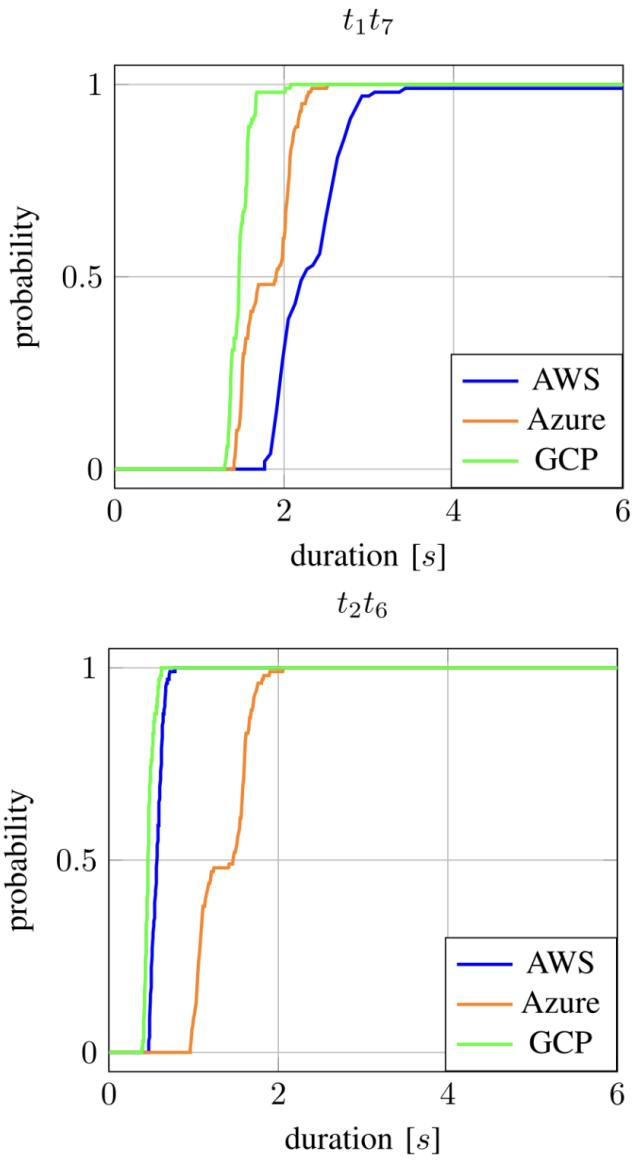

Figure 2: CDF plots of the durations related to the whole synchronous process.

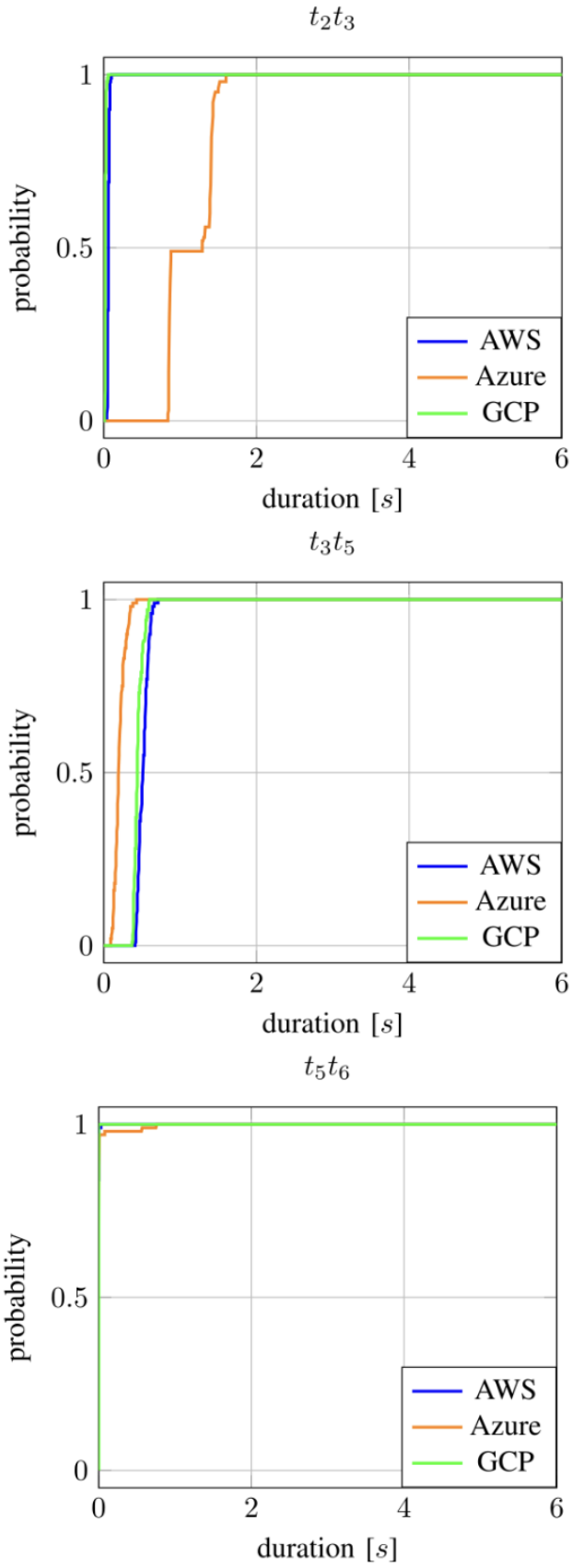

Figure 3: CDF plots of the durations related to the triggering and execution of the CreateUser function.

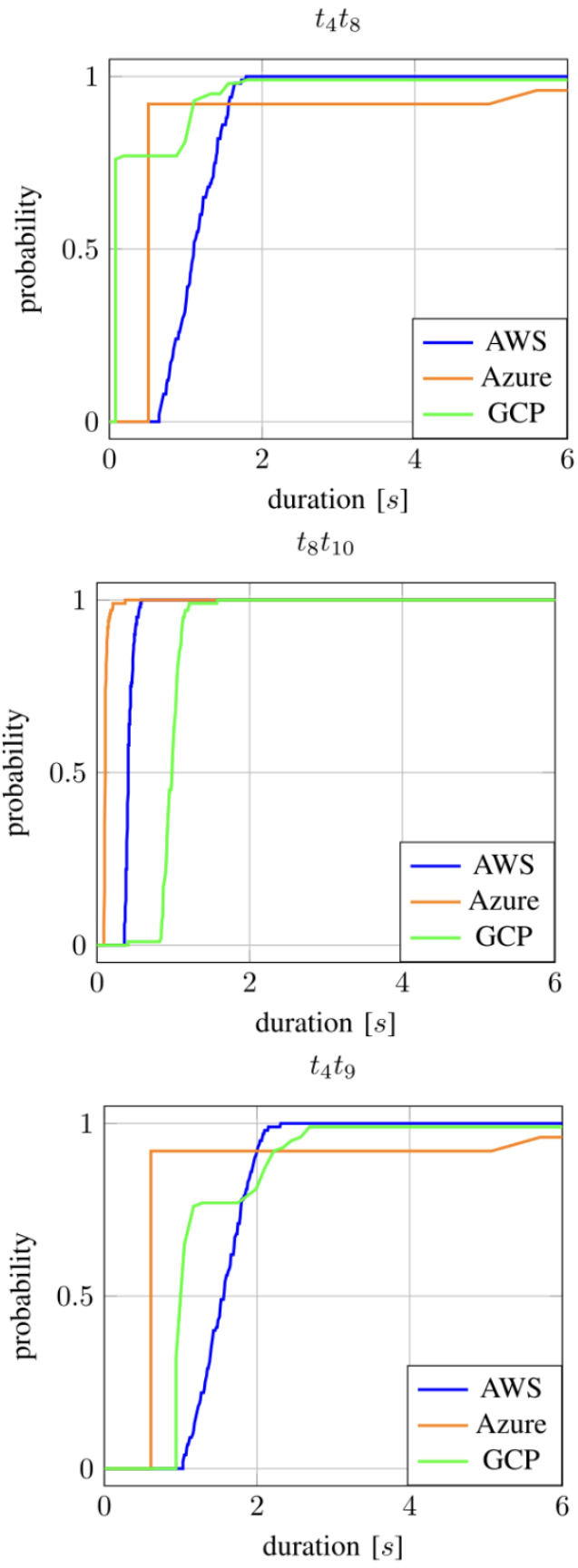

Figure 4: CDF plots of the durations related to the triggering and execution of the ProcessImage function.

HYPOTHESIS 1

We assess the consistency of the platforms using the COV values derived from the calculated durations of the CreateUser and ProcessImage functions. The COV values help us evaluate our first hypothesis, where a lower COV indicates better consistency. However, it's important to note that COV is not an absolute measure; its interpretation depends on the mean value's magnitude, and that means that smaller mean values tend to make achieving a low COV more challenging.

Our results indicate that for the HTTP trigger, AWS and GCP exhibit high consistency, while Azure's performance is significantly less consistent, considering its mean trigger duration. In the case of event-based triggers, AWS shows moderate consistency. GCP, although less consistent, has a smaller mean trigger duration. Azure, on the other hand, is notably less consistent, with a significant variation in performance, especially at the extremes. As for the function execution durations, we see a high level of consistency across all platforms.

Compared to Hamiti's results (Hamiti, 2023), we see a similar level of consistency in function execution durations. In the case of HTTP trigger durations, we observe minor variations in the COV, though these are not significant. For the event-based triggers, there is a significant increase in GCP's coefficient, which is almost three times higher than what was observed in the thesis.

Contrasting our findings with Zhang et al.'s study, which suggested higher consistency in GCP's function execution duration compared to AWS, our results do not support such a conclusion (Zhang, Zhu, Zhang, & Liu, 2019). It's important to note, however, that Zhang et al. conducted their tests across a broader range of memory configurations, whereas our study focused on a single configuration.

HYPOTHESIS 2

For platform performance comparison, we analyze the mean and standard deviation values from the table, as well as the CDF plots. We examine both the triggering and execution durations of the CreateUser and ProcessImage functions, as well as the overall synchronous and asynchronous processes. While network latency durations are present in our results, they are not discussed as part of our analysis.

We see that the HTTP trigger in AWS and GCP is almost instantaneous, while in Azure it takes longer than a second. The performance of the event-based trigger on each platform is comparable, although each platform seems to have its advantages, i.e. GCP has the lowest mean value for trigger duration, Azure has a strictly equal duration for 95% of cases, and AWS has the lowest standard deviation. The functions execution duration is consistently lower in Azure compared to the other platforms, though it is worth noting that functions in Azure can use more memory as they cannot be strictly configured to use a specific amount of memory. Considering the synchronous process as a whole, we see that overall, Azure performs twice as worse than AWS and GCP, while the asynchronous process seems to complete in an equal amount of time on all platforms.

Our results are in agreement with Hamiti's observations for the most part. We see that Azure function duration is considerably lower in our study, and we see that using an event-based trigger has made a considerable positive impact in the triggering of the ProcessImage function in Azure. In contrast to McGrath and Breener's findings that AWS and Azure have Superior HTTP trigger performance compared to GCP (McGrath & Brenner, 2017), our results differ. We found GCP to have the fastest HTTP trigger, followed closely by AWS, while Azure

lagged significantly, being around 20 to 50 times slower. Our results are in agreement with Zhang et al.'s observation of differences in AWS and GCP execution times (Zhang, Zhu, Zhang, & Liu, 2019). However, for the execution of the ProcessImage function, we see that AWS's performance nearly doubles that of GCP. Lastly, our findings perfectly match the examination of Ivan et al. on the duration of the synchronous process. We both find Azure to take twice as long as AWS to complete the same process.

EXPERIMENT REPLICATION AND FUTURE IMPROVEMENTS

To deploy Hamiti's application, we made several modifications to the infrastructure code, which has now been updated to reflect the latest changes (Hamiti, 2023). These changes addressed issues such as expired hardcoded tokens, inconsistencies in the cloud services used, and necessary updates to the function code dependencies. Cost-wise, re-running the experiment proved to be more economical, likely due to the minimal trial and error required to re-deploy an already defined infrastructure, resulting in no costs for AWS, while GCP incurred around EUR 6 and Azure approximately EUR 17.

Future research should aim to expand the benchmark to include more workflows and utilize a broader range of cloud services. Enhancing the automation level of the deployment, authentication, and data collection scripts could significantly improve the efficiency of the experimental process. Furthermore, incorporating additional cloud platforms in the benchmark should be considered, as it would provide a more comprehensive understanding of the FaaS landscape.

CONCLUSION

The study aims to assess the performance of various Function as a Service (FaaS) platform, building on the framework established in Hamiti's thesis (Hamiti, 2023). We re-execute the benchmark from their study with several enhancements. The benchmark is an application-level evaluation conducted on Google Cloud Platform (GCP), Azure, and Amazon Web Services (AWS). It comprises two processes reflecting real-world use cases: a synchronous process for creating a user account and uploading a profile picture, and an asynchronous process that generates a thumbnail from the uploaded picture. Our approach involved recording timestamps for each step in the workflow and thus measuring the performance consistency within each Cloud Service Provider (CSP) and comparing the overall performance across the CSPs. Our findings indicate that while function execution shows a high degree of consistency, function triggering is less so. In terms of speed, Azure's trigger durations were significantly longer compared to those in GCP and AWS, though function execution durations did not show marked differences across the platforms.

When compared with Hamiti's findings (Hamiti, 2023), the study confirms the repeatability of their measurements to a considerable extent. There are minor variations over time, but there are not any changes in ranking among CSPs. We see that the enhancements made to the experimental set-up yielded new insights, which are valuable for a fair comparison between CSPs. Lastly, we found Hamiti's reproducibility package to be practical and effective, but we see a lot of opportunities for further improvement.

REFERENCES

1. Ivan, C., Vasile, R., & Dadarlat, V. (2019). Serverless Computing: An Investigation of Deployment Environments for Web APIs. Computers, 8(2), 50. doi:10.3390/computers8020050
2. Manner, J., Endreß, M., Heckel, T., & Wirtz, G. (2018, December). Cold Start Influencing Factors in Function as a Service. 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 181–188. doi:10.1109/UCC-Companion.2018.00054
3. McGrath, G., & Brenner, P. R. (2017, June). Serverless Computing: Design, Implementation, and Performance. 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 405–410. doi:10.1109/ICDCSW.2017.36
4. Zhang, M., Zhu, Y., Zhang, C., & Liu, J. (2019, June). Video processing with serverless computing: a measurement study. Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, 61–66. doi:10.1145/3304112.3325608
5. Lee, H., Satyam, K., & Fox, G. (2018, July). Evaluation of Production Serverless Computing Environments. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 442–450. doi:10.1109/CLOUD.2018.00062
6. Flexera 2023 State of the Cloud Report. (n.d.). Retrieved from https://info.flexera.com/CM-REPORT-State-of-the-Cloud
7. Hamiti, B. (2022). Resources for 'Function-as-a-Service Performance Evaluation with Application-level Workflows' (Version 0.1) [Data set]. doi:10.5281/zenodo.7112754
8. Hamiti, B. (2023). Function-as-a-Service Performance Evaluation with Application-level Workflows (Master's thesis). South East European University.