future internet

MDPI

Article

An Experimental Performance Evaluation of Cloud-API-Based Applications

Yara Abuzrieq [1], Amro Al-Said Ahmad [1,2] and Maram Bani Younes [1,*]

[1] Faculty of Information Technology, Philadelphia University, P.O. Box 1, Amman 19392, Jordan; Yara.abuzrieq@gmail.com (Y.A.); asaid@philadelphia.edu.jo (A.A.-S.A.)
[2] School of Computing and Mathematics, Keele University, Newcastle-under-Lyme ST5 5BG, UK
[*] Correspondence: mbani047@uottawa.ca

Abstract: Cloud Application Programming Interfaces (APIs) have been developed to link several cloud computing applications together. API-based applications are widely used to provide flexible and reliable services over cloud platforms. Recently, a huge number of services have been attached to cloud platforms and widely utilized during a very short period of time. This is due to the COVID-19 lockdowns, which forced several businesses to switch to online services instantly. Several cloud platforms have failed to support adequate services, especially for extended and real-time-based applications. Early testing of the available platforms guarantees a level of suitability and reliability for the uploaded services. In this work, we first selected two different API-based applications from education and professional taxonomies, the two most recently used applications that have switched to the cloud environment. Then, we aimed to evaluate the performance of different API-based applications under different cloud platforms, in order to measure and validate the ability of these platforms to support these services. The advantages and drawbacks of each platform were experimentally investigated for each application.

Keywords: performance; SaaS; cloud application; cloud platform; Application Programming Interfaces (APIs)

Citation: Abuzrieq, Y.; Ahmad, A.A.-S.; Younes, M.B. An Experimental Performance Evaluation of Cloud-API-Based Applications. *Future Internet* **2021**, *13*, 314. https://doi.org/10.3390/fi13120314

Academic Editor: Joel J. P. C. Rodrigues

Received: 11 November 2021
Accepted: 8 December 2021
Published: 13 December 2021

1. Introduction

The advanced usability and popularity of cloud computing encourage developers to intensively investigate its environment. Evaluating the performance of several existing cloud platforms assists consumers' decisions regarding the optimal environment option [1]. Cloud computing has introduced scalable and elastic network resources that are offered as services through the Internet. The information, data, and shared resources are provided to the consumers upon request [2]. However, the performance evaluation of the applications uploaded to cloud computing platforms is considered highly complex. It is affected by the complexity of the infrastructure of the cloud environment where the applications are executed [3].

Application Programming Interfaces (APIs) represent a set of programming codes that permit and control the data transmission between different connected applications. Two main components are included in any API: technical specifications and the software interface. The technical specifications define the choices and rules of the data exchange among the connected applications. These specifications comprise the request for data delivery and the processing protocols. The software interface is written for the indicated specifications. When a certain application (i.e., requesting application) needs to access some operation or information from another application (i.e., providing application), it calls the corresponding API to determine how this operation or these data can be provided. The providing application, in turn, returns the requested operation or information directly to the requesting application. The API specifies the interface through which the requesting and providing applications can communicate.

APIs provide several advantages to the software development field in terms of speeding up and simplifying the development process. Developers can add functionalities and operations from other existing providers to their proposed software. The APIs can also serve as a layer of abstraction between two systems, where the working and complex details of the finalsystem are hidden [4]. Today, there are many applications and APIs that different corporations provide over the Internet. The user can access these services and applications any time and any where. Therefore, the servers that offer these applications and services must handle a massive number of simultaneously received requests. Thus, enterprises have adopted this promising cloud computing technology since not all companies have the resources to create data centers or purchase many servers for these applications.

The performance evaluation of different API-based applications uploaded to cloud platforms is an important issue. It assists in selecting the right and suitable platform to run newly developed services and applications. Several research studies have been conducted in this field considering different circumstances, cloud platforms, applications, and APIs. Here, we introduce another performance evaluation study that seeks to assess different cloud computing platforms for different API-based application categories. Amazon AWS EC2 and Microsoft Azure are considered the most commonly adopted cloud platforms [5]. We used these two platforms in our experiments since they are the top cloud service providers. Moreover, we selected web-APIs to test different mobile applications. Two taxonomies were selected and compared in this work: educational (i.e., Sebawayh) and professional (i.e., TaqTak) services. Education and professional services have witnessed a high percentage of switching to the cloud environment recently. This is due to the COVID-19 lockdowns, with schools and universities running online classes, and likewise for several businesses' activities. An experimental comparison of the performance of the API-based applications under the Amazon AWS EC2 and Microsoft Azure platforms in terms of response time, latency, processing time, and throughput is presented in this work. These experiments were performed with several users. From the experimental study, we can infer that the Microsoft Azure platform is the best choice for uploading high-computational-resource applications. On the other hand, for educational applications, Amazon AWS EC2 provides less latency compared to Microsoft Azure.

In this study, we measured the performance of cloud-API-based applications from a technical perspective, which becomes important as more API-based applications migrate to the cloud. The experimental analysis of the results considered four sets of comparisons, in order to provide a platform to construct a methodology to effectively compare the performance of cloud-API-based applications and consequently support deployment decisions with technical arguments.

The remainder of this paper is organized as follows. Section 2 reviews some of the previous studies in the literature. Section 3 presents the methodology and sets the experimental parameters of this study. Section 4 presents the obtained results and the discussion. Finally, Section 5 presents the conclusions for the entire work and the future work.

2. Related Work

Several studies have assessed and evaluated cloud computing applications by considering several techniques [6–8]. Researchers have investigated the capacity of cloud servers and the performance of the applications uploaded to these cloud environments. Jackson et al. [6] presented an efficient assessment study that compared HPC and the Amazon AWS EC2 cloud platform. The experiments employed real applications that represented the workload of a super-computing center. From the experimental results, it was shown that using a Linux cluster was six-times faster than Amazon AWS EC2, and modern HPC systems were found to be twenty-times faster than Amazon AWS EC2. The performance of the tested applications was limited by interconnecting with the Amazon AWS EC2 platform. Moreover, the Amazon AWS EC2 platform introduces substantial variability.

Moreover, Bautista et al. [7] used "ISO Quality Characteristics" to evaluate the performance of the applications uploaded to the cloud computing environment. The main consideration regarding this experimental study was the high complexity of the cloud environment, which is directly inherited from the complex infrastructure. In this experimental study, they adopted a new measurement framework and applied it to log data obtained from the data center. This was performed to map and examine particular quality characteristics of the ISO (i.e., the behavior during a period of time). A particular industrial private cloud was used to conduct this experimental study. The experimental results confirmed the effectiveness of the proposed framework at measuring and evaluating the performance of the uploaded applications. Ravanello et al. [3] investigated the performance of big data applications, which combine the concepts of software quality from the ISO25010 model. Their experimental study aimed to fill the gap in the numerical representation of these concepts and the measurement of big data applications' performance.

New models and frameworks have been proposed, aiming to facilitate the experimental study of applications uploaded to the cloud environment. Addamani and Basu [8] presented a new model to analyze the performance of web applications in the cloud computing environment. They modeled the IaaS platform as multiple queues and "Virtual Machines" (VMs) as service centers. This model depends on the application behavior of the reverse instances on Amazon EC2. On the other hand, Vasar et al. [9] proposed a new framework that combines different monitoring and benchmarking tools. Performance engineers can use this framework to test the applications under different loads and configurations. Moreover, the dynamic allocation of the sever is supported through this framework depending on the incoming load by employing response-time-aware heuristics.

Many researchers have investigated different performance metrics of the cloud environment for several applications. An experimental assessment of the performance of an analytic application that is run through different configurations of the storage service was presented in [10]. The authors analyzed the performance achieved through analytic workloads and explained the problems that occurred due to impedance mismatch, which occurs in some configurations. Marian et al. [11] tried to help the users choose the optimal configuration of each application by creating a prediction model using the cloud providers for the offered system. This model of prediction was created by applying machine-learning methods. They used the OpenStack cloud system to ensure the validity of the suggested methodology. Kotas et al. measured the performance of the most commonly used cloud platforms (i.e., Amazon AWS EC2 and Microsoft Azure). The experiments were run by generating clusters of Azure and AWS instances. Then, they executed the benchmarks related to the several instances. Another recent work investigated the performance of real-time-stream-processing systems for Internet of Things (IoT) applications [12]. The paper measured and compared the performance of a number of IoT applications based on response time, throughput, jitter, and scalability. Moreover, Ismail and Materwala [13] evaluated and compared the performance of the blockchain and the client/server cloud paradigms for healthcare applications.

Furthermore, scalability has been investigated in the cloud environment to determine the capacity of the uploaded applications. Expósito et al. [14] proposed methods to reduce the virtualization overhead effect on the scalability of communication-intensive HPC codes. A method that combined multiple threading and message-passing techniques to run HPC applications was presented. The scalability and the cost efficiency were the major metrics used to evaluate the performance. An experimental evaluation of the auto-scaling policies' performance was performed in [15] as well. These experiments performed different group and pairwise comparisons to recognize the performance dissimilarities among the seven strategies. A recent work by Potodar et al. [16] presented a performance evaluation of cloud docker containers and VMs in terms of CPU performance, throughput, disk input/output, load testing, and operation speed measurement. The work used multiple benchmarking tools in order to compare docker containers and VMs, and the comparisons showed that dockers performed better in terms of the chosen technical measurements. Al-Said Ahmad and Andras [17] also used a technical scalability measurement for cloud-based software services. They were inspired by the technical scalability and elasticity metrics. They employed two different cloud-based systems, Media-Wiki and OrangeHRM, to illustrate their metrics' usefulness and compared the performance of these systems in terms of scalability on two platforms: Amazon AWS EC2 and Microsoft Azure.

In this work, we aim to continue the work presented in [17]. Thereby, we investigated the performance of two popular and commonly used API-based applications for different numbers of users on cloud platforms.

3. Methodology and Experimental Setting

As discussed earlier, cloud computing is one of the most common promising technologies. Individuals, organizations, and enterprises can focus on developing their main activities while leaving the IT services' maintenance and development to the cloud providers. APIs act as the interfaces that connect the service providers to the customers or other developed services. The most recent services provided on cloud platforms are performed through APIs. These APIs are used to provide services to the consumers in addition to managing and monitoring them. Investigating the performance of different API-based applications on different cloud platforms is considered an important issue. It facilitates the process of selecting the most suitable platform for each developed application and/or service and the main effects or considered parameters.

3.1. Testing Process and Approach

In this work, we introduce a performance evaluation study that aims to assess different cloud computing platforms for different API-based application categories (i.e., educational and professional). These categories were selected because they have been the ones most commonly deployed in the cloud environment during the COVID-19 lockdowns. The entire world suddenly switched to online studies, and most professional businesses went online as well.

Figure 1 summarizes the sequential steps of the proposed experimental study. The first step was to select applications for the testing process and investigate their characteristics and the parameters affected. In our work, two mobile applications based on web-APIs were selected from two different categories. These applications were Sebawayh (i.e., educational application) and TaqTak (i.e., professional application). Sebawayh is an educational web-API and smartphone app for Arabic language learning. It has three main users: students, teachers, and system administrators. It has multiple learning levels and categories. The system has a registration form for the students before starting any level of learning on the system. It has a smartphone interface that sends HTTP requests to the web-API. On the other hand, TaqTak is a solar power renewable energy design, monitoring, and management application. It is utilized to monitor power system usage and to show reports about the system. It can also be utilized by engineers to calculate the voltage requirements of any system. It calculates the voltages and battery power and monitors consumption. It can be connected to a third-party application to receive power usage data for the monitoring process.

The next step was to lease cloud servers and configure them according to the tested applications. The most popular and commonly used platforms were selected in this study (i.e., Amazon AWS EC2 and Microsoft Azure). The third step was to deploy the APIs on the configured servers. A load test method was used to compare the two applications on the two cloud platforms. To test these web services/APIs, an open-source load-testing framework was utilized, named Apache JMeter [18]. This framework is used for application and web service testing, as well as for measuring and analyzing the performance of a variety of web services including web-based/cloud applications. JMeter, considered as a multi-threading framework, allows concurrent workload sampling [19]. It can be configured to automate web services' testing regarding different parameters. Finally, the collected results were analyzed and compared according to the considered parameters, as illustrated in Figure 1. In this study, we focused on collecting technical performance parameters. All monitoring data were extracted from JMeter after each test completed [20]. The investigated parameters of this study were:

- **Response time**: measures the required time for handling the request and returning the results for the request;
- **Latency**: measures the required time for the request to reach the server. This time is used to measure the network quality between the server location and clients. It also can be impacted by the load that hits the server;
- **Processing time**: measures the required time of the operating system to work or process a request;
- **Throughput**: measures the number of tasks that have been completed during a certain period of time. In the cloud environment, throughput measures how fast the application can transfer the data between a client and a server.

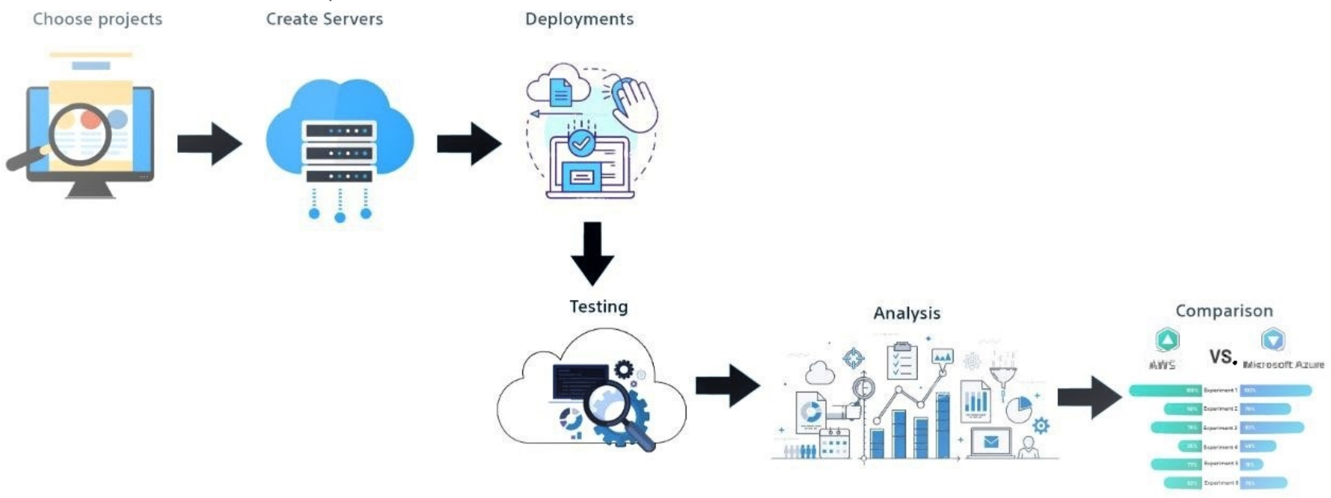


Figure 1. Steps of the experimental study.

3.2. Cloud Platforms' Configuration

In order to perform the comparison study, we leased two cloud servers in the same geographical location. The first server was leased and configured for Amazon AWS. The other server was configured for the Microsoft Azure platform. The two web-API (i.e., Sebawayh and TaqTak) were deployed on these servers. These applications adopt the Django web framework. The configuration parameters of these two servers are given in Table 1. The Ubuntu operating system was utilized on both servers with two CPUs, with four gigabytes of RAM memory and five gigabytes assigned as a secondary storage for each server in these experiments.

Table 1. Servers' configuration parameters.

Parameters	Amazon AWS EC2	Microsoft Azure
Server Type	Ubuntu	Ubuntu
Instant Type	T2 medium	T2 medium
Number of CPUs	2	2
RAM	4	4
Storage	5	5

This framework was configured to automate the performance tests with different parameters, such as the number of requests, the number of users, and the minimum and maximum delay time between the different requests. Apache JMeter was configured two times for the different tested applications. The load test was also performed using these configurations on both leased servers. Table 2 shows the pages that were requested by the Sebawayh application. Two main methods were tested in this set of experiments: "Post" and "Get". Different HTTP requests were applied for the different APIs/paths as well. Moreover, Table 3 gives the pages requested by the TaqTak application and their paths.

Table 2. Pages requested by the Sebawayh application.

No.	HTTP Request	Method	API/Path
1	authentication	Post	/api/v1/auth/authenticating/
2	user	Get	/api/v1/auth/user/2/
3	language	Get	/api/v1/language/
4	language info	Get	/api/v1/language_levels/2/
5	language level	Get	/api/v1/language_levels/
6	language to learn	Get	/api/v1/lang_to_learn/
7	setting	Get	/api/v1/setting/
8	notification	Get	/api/v1/notification/
9	to learn	Get	/api/v1/why_To_Learn/
10	why to learn post	Post	/api/v1/why_To_Learn/

Table 3. Pages requested by the TaqTak application.

No.	HTTP Request	Method	API/Path
1	authentication	Post	/api/v1/auth/authenticating/
2	user	Get	/api/v1/auth/user/2/
3	country	Get	/api/v1/country/
4	city	Get	/api/v1/city/
5	notification	Get	/api/v1/notification/
6	role	Get	/api/v1/role/
7	project	Get	/api/v1/project/
8	category	Get	/api/v1/category/
9	ticket	Get	/api/v1/ticket/
10	setting	Get	/api/v1/setting/1/

During the time of the experimental test, all network applications were disabled on the computer used. Moreover, all other devices in the lab where we ran our experiments were disconnected from the Internet to reduce the latency and the delay of the experiments. This set of experiments required around sixty hours of execution time on the leased servers. Furthermore, the tested experiments were run for different numbers of users: 500 users, 1000 users, 2000 users, and 4000 users. Table 4 illustrates the parameter setting for the different numbers of tested users, including the experiment time, number of iterations, number of experiments, and total running time.

Table 4. Experimental parameter setting.

No. of Users	Experiment Time	No. of Iterations	No. of Experiments	Total Time
500	500 s	10	40	5 h
1000	1000 s	10	40	11 h
2000	2000 s	10	40	22 h
4000	4000 s	5	20	22 h

4. Experimental Results and Analysis

To compare the applications from the two different API-based categories on both cloud platforms, four performance metrics were selected: response time, latency, processing time, and throughput.

4.1. The Experimental Scenarios

Table 5 gives the scenarios that were utilized in this work. The four performance metrics were measured for both applications on the two different investigated cloud platforms. Then, the outputs of these four experiments were compared. We emphasize that we compared the two application categories (i.e., education and professional) in the cloud environment and how they behaved in the different cloud paradigms.

Table 5. The experimental scenarios.

Platform	**Scenario1**	**Scenario2**	**Scenario3**	**Scenario4**
Amazon AWS EC2	Sebawayh	TaqTak	Both APIs	-
Microsoft Azure	Sebawayh	TaqTak	-	Both APIs

4.2. Results for Different Cloud-API-Based Applications on the Amazon AWS EC2 Platform

We investigated the four measured parameters for the selected applications on the AWS EC2 platform. Figure 2a shows the response time of both cloud-API-based applications. We can see from the figure that the response time of each application was stable for the different numbers of tested users. The number of users never affected the system's performance or its ability to handle the requests. However, we can observe that Sebawayh had a faster response time than the TaqTak application. The reason behind this is the size of the data transmitted in each application. A larger number of request and response messages of the applications increased the response time parameter on the tested cloud platform. TaqTak requires transferring more parameters between the two end nodes compared to Sebawayh. The black interval lines in all performance evaluation figures illustrate the variation of the response times of the different pages requested for each application to provide accurate results.

The latency parameter of these APIs on the Amazon AWS EC2 platform is illustrated in Figure 2b. It was approximately the same for both applications for the different investigated numbers of users. However, the latency of the two applications was different. The latency of Sebawayh was less than that of TaqTak. This was due to the size of the transmitted data and the complexity of the investigated applications.

Moreover, Figure 2c shows the processing time of these two applications. We can observe that Sebawayh required more processing time than the TaqTak service application. Sebawayh's algorithm depends on image processing and the different database transactions of the users' levels, grades, and exams. Under a high processing time, the response time of this application became lower, as seen in Figure 2c, since most of the work was processed on the servers and did not involve the network. Another observation from Figure 2c is that the variation of the processing time results of TaqTak was greater than the variation results of Sebawayh. This was affected by the required services of the different pages in these applications.

Finally, Figure 2d shows the throughput of the network for these applications in the Amazon AWS EC2 environment. We can observe that both of these applications obtained a 100% throughput in all tested scenarios, which means that no packets were dropped and no requests were delayed or ignored.

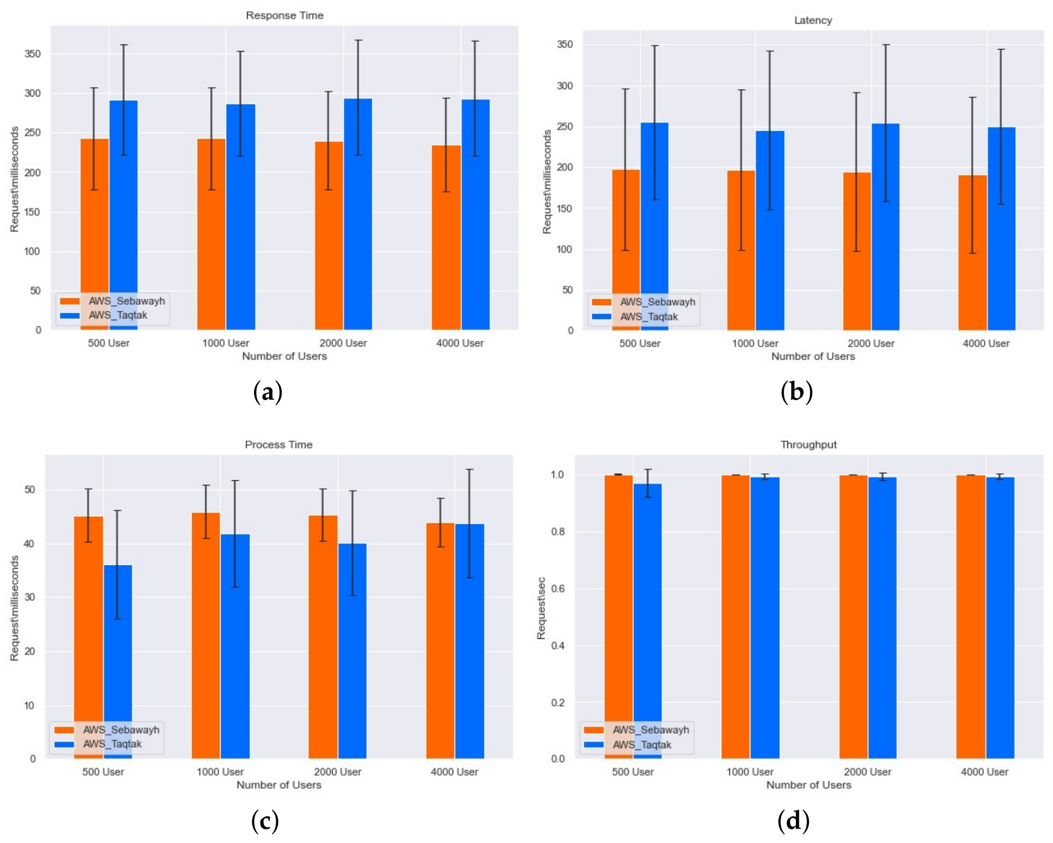


Figure 2. Results for different cloud-API-based applications on Amazon AWS EC2. (**a**) Response time of both APIs on AWS EC2. (**b**) Latency of both APIs on AWS EC2. (**c**) Processing time of both APIs on AWS EC2. (**d**) Throughput of both APIs on AWS EC2.

4.3. Results for Different Cloud-API-Based Applications on Microsoft Azure

The same experiments were performed on the Microsoft Azure cloud platform. Figure 3a shows the response time of these applications on Azure. The same as the results on AWS EC2, the response time did not depend on the number of users or the number of requests. The response times for the different numbers of users were the same. We can also observe that the response time of the TaqTak application was greater than that of the Sebawayh application. This depends on the complexity of the request/response activities and the size of the data in the request and the response messages of these categories.

Figure 3b illustrates the latency of both applications on Microsoft Azure. We can infer from the figure that both obtained similar results, with a higher latency for TaqTak compared to the Sebawayh application. We can also observe that the number of users did not affect the latency parameter for either application since the results in all four cases (i.e., 500 users, 1000 users, 2000 users, 4000 users) were approximately the same.

Figure 3c shows the processing time performance of these investigated applications on the Azure platform. The processing time of the Sebawayh application was higher than the processing time of TaqTak, the same as on the AWS EC2 environment. The number of users also did not impact the processing time parameter on this cloud platform. Finally, Figure 3d shows the throughput parameter of these two applications on Azure. We can observe from this figure that both applications obtained approximately 100% efficiency regarding the throughput.

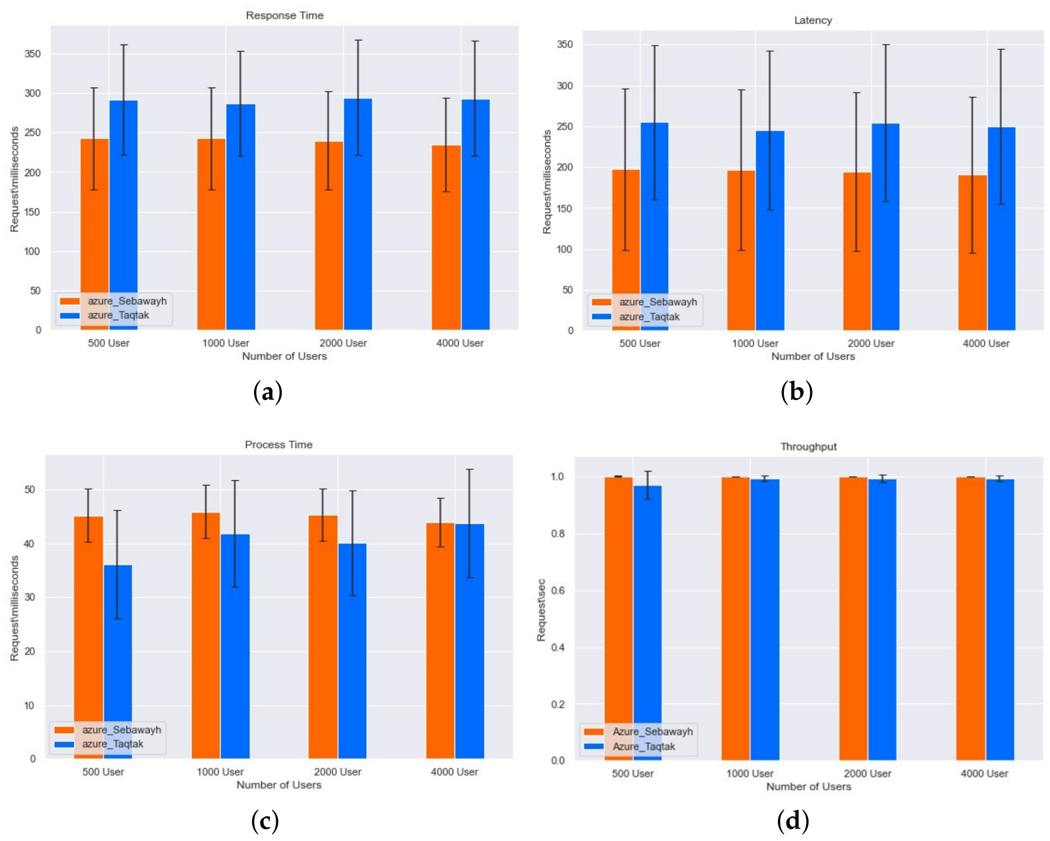
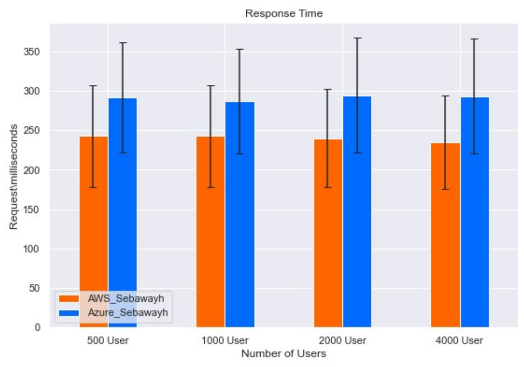


Figure 3. Results for different cloud-API-based applications on Microsoft Azure. (**a**) Response time of both APIs on Azure. (**b**) Latency of both APIs on Azure. (**c**) Processing time of both APIs on Azure. (**d**) Throughput of both APIs on Azure.

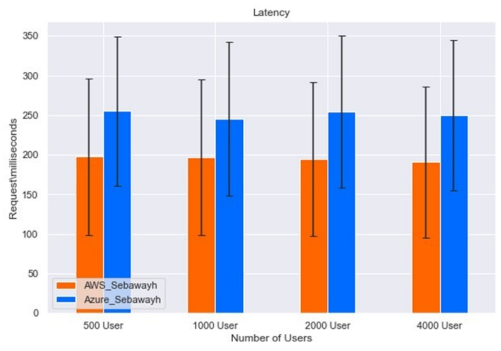
4.4. Comparison of the Educational Category API on the Two Platforms

In this set of experiments, we aimed to evaluate the performance of the Sebawayh application on the two different cloud platforms. This would help in recommending the most suitable platform for the educational category. First, Figure 4b shows the response time of this application on the two cloud platforms. AWS EC2 had, on average, a 20% slower response time than Azure. The same was true for the latency parameter. Figure 4a shows the latency obtained for the educational application on AWS EC2 and Azure. We can observe that the performance on AWS EC2 had lower latency in all scenarios compared to that on Azure. The average difference under all scenarios was approximately 2%. This does not mean that AWS is better than Azure all the time. However, for the educational application and the location of our clients, AWS EC2 offered lower latency than Azure.

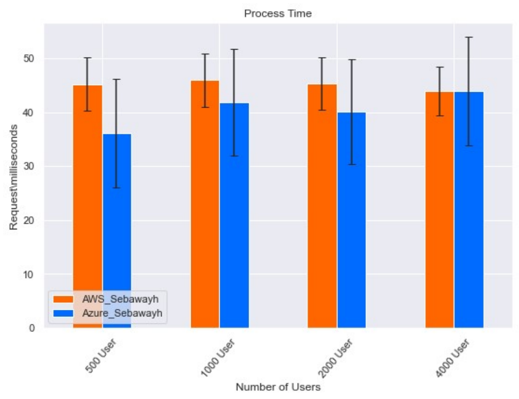
Figure 4c shows the processing time of this application on both cloud platforms. We can observe from the figure that the processing time of AWS EC2 was much greater than that of Azure for this application. This means that for other application categories that depend heavily on processing time, Azure offers a better choice with respect to the computational delay. Another observation in this figure is that the processing time on Azure changed when the number of users changed. It slightly increased when increasing the number of users. It is worth mentioning that our leased server crashed for 4000 users with Azure, and we repeated the experiment several times to confirm this. This is considered a serious issue for a salable environment where the number of users is expanding. Figure 4d shows the throughput for Sebawayh in these environments. The throughput was identical for this application on the two cloud platforms, as it only depends on the number of transferred messages, and this was the same in all experiments.



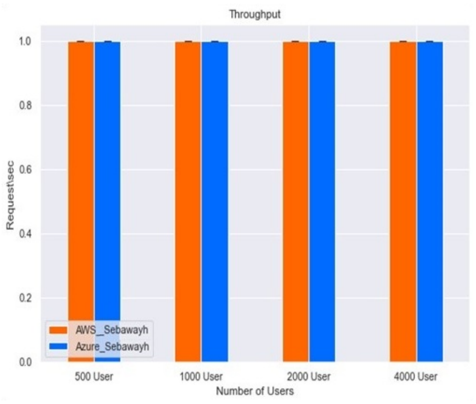
(**a**)



(**b**)



(**c**)



(**d**)

Figure 4. Comparison of the educational category API under the two platforms. (**a**) Response time of the Sebawayh application. (**b**) Latency of the Sebawayh application. (**c**) Processing time of the Sebawayh application. (**d**) Throughput of the Sebawayh application.

4.5. Comparison of the Professional Service Category API on the Two Cloud Platforms

This last set of experiments aimed to determine the best cloud platform for the professional services category. Figure 5a shows the response time of the professional category application on the two cloud platforms. We can observe from the figure that the response time was approximately the same for both cloud environments. It was slightly higher for AWS EC2 than Azure. This was not the same as for the educational category. This shows that the selected cloud service depends heavily on the application category that will be operated or offered.

On the other hand, Figure 5b shows the latency of both platforms. We can observe that the latency of AWS was lower than that of Azure. Moreover, the number of users on both platforms did not impact the latency. Figure 5c shows the processing time of both cloud platforms. We can observe that Azure obtained a lower processing time than AWS EC2, the same as for the educational category. This means that for higher computational and processing power, Azure should be selected. Finally, Figure 5d shows the throughput of this application on both platforms. Again, a 100% throughput was obtained for both platforms.

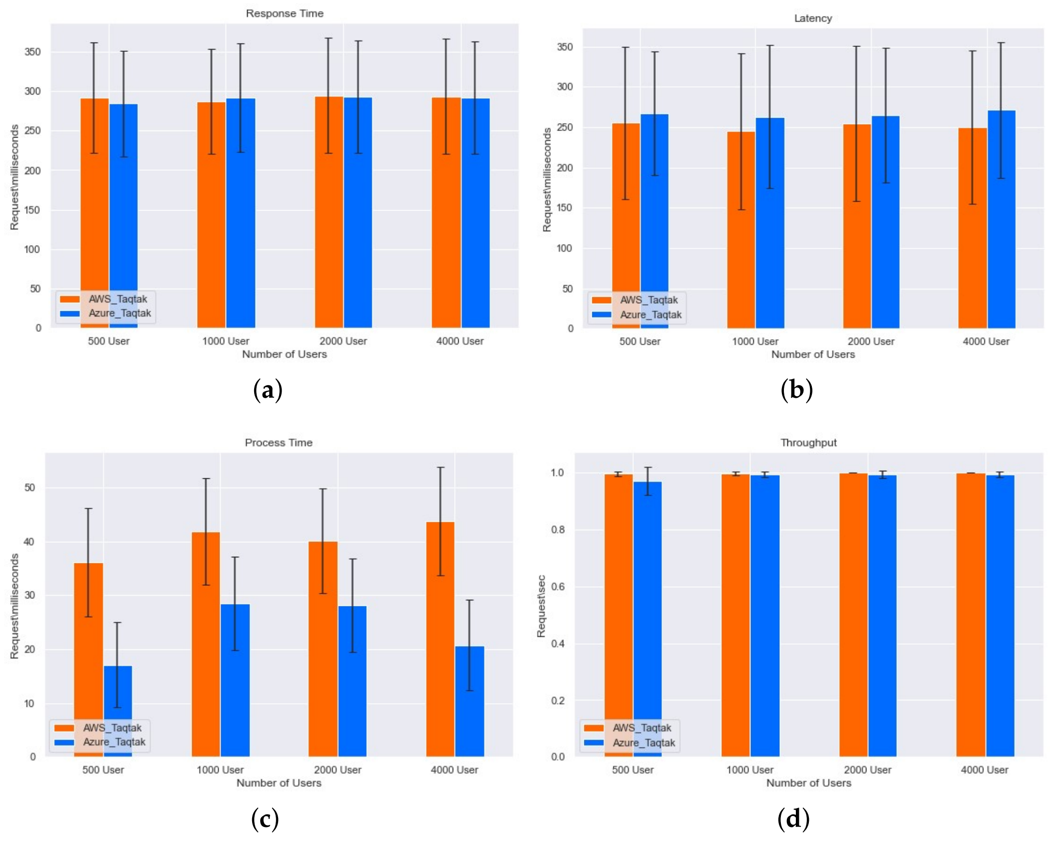


Figure 5. Comparison of the professional service category API on the two cloud platforms. (**a**) Response time of the TaqTak API. (**b**) Latency of the TaqTak API. (**c**) Processing time of the TaqTak API. (**d**) Throughput of the TaqTak API.

5. Conclusions and Future Work

In this work, we evaluated the performance of two different categories of API-based applications on the Amazon AWS EC2 and Microsoft Azure cloud platforms. A performance evaluation study was conducted to compare the performance of Sebawayh and TaqTak on these two most commonly adopted cloud platforms. The experimental results showed that: the performance metric selected depended mainly on the API category rather than the number of users. The educational API (i.e., Sebawayh) had less response time and less latency than the professional application (i.e., TaqTak) on the leased cloud platforms. The processing time of Azure was lower than that of AWS EC2. The throughput of these applications did not change or decreased when moving from one cloud platform to the other. This means that the Azure platform is the best choice for API-based applications that require heavy computations.

As future works, other API categories/taxonomies will be investigated, tested, and compared to show the differences among them. Then, a full guideline for deployment in all categories can be constructed. Other parameters could be tested on cloud environments for different loads and basic parameters. Moreover, the usability of API-based applications can be further investigated, also considering the user's quality of experience.

Author Contributions: Conceptualization, Y.A. and M.B.Y.; methodology, Y.A. and M.B.Y.; software, Y.A. and M.B.Y.; validation, Y.A. and M.B.Y.; investigation, Y.A. and M.B.Y.; resources, Y.A. and M.B.Y.; data curation, Y.A. and M.B.Y.; writing—original draft preparation, Y.A.; writing—review and editing, A.A.-S.A. and M.B.Y.; visualization, Y.A.; supervision, A.A.-S.A. and M.B.Y.; project administration, A.A.-S.A.; funding acquisition, A.A.-S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the scientific research Deanship of Philadelphia University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work has been done in Philadelphia University, Amman, Jordan as a part of master thesis.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khanghahi, N.; Ravanmehr, R. Cloud Computing Performance evaluation: Issues and Challenges. *Int. J. Cloud Comput. Serv. Archit.* **2013**, *5*, 29–41. [CrossRef]
2. Khalid, R.; Abdullah, T.; Rashid, I. Performance Degradation Factors in Cloud Computing. *Int. J. Sci. Eng. Res.* **2016**, *7*, 384–394.
3. Ravanello, R.; Desharnais, J.; Villalpando, L.; April, A.; Gherbi, A. Performance meas-urement for cloud computing applications using ISO 25010 standard characteristics. In Proceedings of the 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, Rotterdam, The Netherlands, 6–8 October 2014; pp. 41–49.
4. Altexsoft: *What Is API: Definition, Types, Specifications, Documentation*. Available online: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/ (accessed on 10 April 2021).
5. ZDNET Homepage. Available online: https://www.zdnet.com/article/the-top-cloud-providers-of-2021-aws-microsoft-azure-google-cloud-hybrid-saas/ (accessed on 29 April 2021).
6. Jackson, K.; Muriki, K.; Canon, S.; Cholia, S.; Shalf, J. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In Proceedings of the 2nd IEEE Interna-tional Conference on Cloud Computing Technology and Science, Indianapolis, IN, USA, 30 November–3 December 2010; pp. 159–168.
7. Bautista, L.; Abran, A.; April, A. Design of a Performance Measurement Framework for Cloud Computing. *J. Softw. Eng. Appl.* **2012**, *5*, 69–75. [CrossRef]
8. Addamani, S.; Basu, A. Performance Analysis of Web Applications on IaaS Cloud Computing Platform. *Int. J. Comput. Appl.* **2013**, *64*, 0975–8887. [CrossRef]
9. Vasar, M.; Srirama, S.; Dumas, M. Framework for Monitoring and Testing Web Application Scalability on the Cloud. In Proceedings of the WICSA/ECSA '12: WICSA/ECSA 2012 Proceedings Companion Volume, Helsinki, Finland, 20–24 August 2012.
10. Pace, F.; Milanesio, M.; Venzano, D.; Carra, D.; Michiardi, P. Experimental Performance Evaluation of Cloud-Based Analytics-as-a-Service. In Proceedings of the 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 27 June–2 July 2016.
11. Mariani, G.; Anghel, A.; Jongerius, R.; Dittmann, G. Predicting Cloud Performance for HPC Applications: A User-oriented Approach. In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Madrid, Spain, 14–17 May 2017; pp. 524–533.
12. Mishra, L.; Varma, S. Performance evaluation of real-time stream processing systems for Internet of Things applications. *Future Gener. Comput. Syst.* **2020**, *113*, 207–217.
13. Ismail, L.; Materwala, H. Blockchain paradigm for healthcare: Performance evaluation. *Symmetry* **2020**, *12*, 1200. [CrossRef]
14. Expósito, R.; Taboada, G.; Ramos, S.; Touriño, J.; Doallo, R. Performance analysis of HPC applications in the cloud. *Future Gener. Comput. Syst.* **2013**, *29*, 218–229. [CrossRef]
15. Ilyushkin, A.; Ali-Eldin, A.; Herbst, N. An Experimental Performance Evaluation of Au-toscaling Policies for Complex Workflows. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, L'Aquila, Italy, 22–26 April 2017.
16. Potdar, A.M.; Narayan, D.G.; Kengond, S.; Mulla, M.M. Performance Evaluation of Docker Container and Virtual Machine. *Procedia Comput. Sci.* **2020**, *171*, 1419–1428. [CrossRef]
17. Al-Said Ahmad, A.; Andras, P. Scalability analysis comparisons of cloud-based software services. *J. Cloud Comput.* **2019**, *8*, 10. [CrossRef]
18. Nevedrov, D. Using JMeter to Performance Test Web Services. *dev2dev* **2006**, 1–11. Available online: https://loadstorm.com/files/Using-JMeter-to-Performance-Test-Web-Services.pdf (accessed on 10 April 2021)..
19. JMeter: *Apache JMeter*. Available online: https://jmeter.apache.org/ (accessed on 10 April 2021).
20. JMeter Glossary: *JMeter Glossary*. Available online: https://jmeter.apache.org/usermanual/glossary.html (accessed on 10 April 2021).