Microsoft Azure Cloud Computing - Server vs Serverless

Marija Taneska
Faculty of computer science and engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
marija.taneska@students.finki.ukim.mk

Aleksandar Dimkoski
Faculty of computer science and engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
aleksandar.dimkoski@students.finki.ukim.mk

***Abstract*—Cloud computing enables us to take advantage of powerful computing capabilities without having to make substantial capital investments. There is a debate on what cloud architecture should be used on what occasions, so the goal of this paper is to identify which cloud architecture is better - server or serverless, and specifically which serverless. To accomplish the goal, we created simple web applications using Django and Spring Boot frameworks, then hosted them on Microsoft Azure cloud in both way - server using Virtual machines and serverless using Azure Function and Azure Container App. In this paper, we describe the differences in cloud architectures and analyze hosting web application. This research not only highlights the importance of performance, scalability, and cost-effectiveness but also helps determine which solution is best for specific purposes.**

***Index Terms*—cloud, serverless, Microsoft Azure, Function App, Container App**

I. Introduction

Businesses now approach IT infrastructure differently because of cloud computing, which enables them to take advantage of powerful computing capabilities without having to make substantial capital investments. Server-based and serverless architectures are two well-liked methods for deploying applications in the cloud. A cloud architecture that relies on servers to host and operate applications can be virtual or physical. These servers need continual upkeep and monitoring and are often managed by the cloud provider or by the customer. A serverless design, on the other hand, frees developers from having to worry about managing the underlying infrastructure so they can concentrate on building and distributing code. Event-driven, or serverless, systems dynamically scale to meet demand and only run when triggered by specified events.
Our main research question was "Which cloud architecture server or serverless is better? " and the next question was "Which Microsoft Azure serverless solution is better to use?". We created simple web applications using Django and Spring Boot frameworks, whose specifications we describe in Section II. In our research we used Azure virtual machines for server based application deployment, while for serverless application deployment we used Function App and Container App. We provide an overview of used Azure services in Section III, and in Section IV we give the details of application deployments. In order to evaluate the performance and scalability, load testing using parallel requests was conducted. Load testing and configuration we describe in Section V, while in Section VI we present the results and analysis.

II. Related work

Cloud environment is not an easy manage, corresponding challenges as availability, load balancing, auto-scalling are addressed in [4]. Serverless cloud computing by adding an additional abstraction layer set as free from these challenges [5]. Additionaly, deploying an application will not cost the developer in the case where the application is idle, and the serverless provider will only charge whenever the application has started using resources [6]. There is common challenges for using both cloud deployment architectures, such as security, monitoring [4]. The testing and comparison of server vs. serverless Azure deployments were conducted in reference [7]. The author reached the conclusion that serverless deployments are more suitable for heavier processing and longer-running tasks.

III. Project specification

We created two simple web applications using Spring Boot and Django frameworks. The application flow is shown in Fig. 1. Both applications work in the following order: through the User Interface the user uploads a file, then a calculation is started via API web service and at the end the Processing time and Result file content are displayed on the user interface.

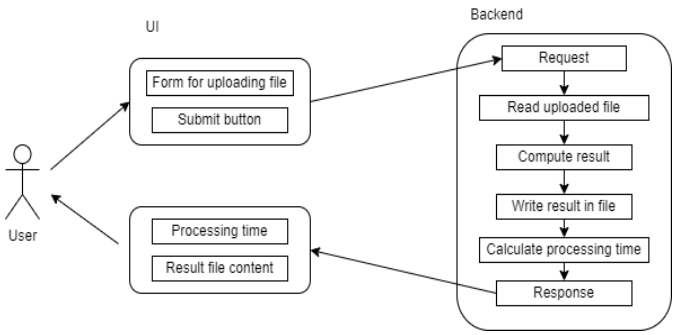

Fig. 1. Application flow

The problem specification for the Spring Boot web application is: Creating a web application and web service (API) for sorting numbers in ascending order. The input files contain

500000 integers.
The problem specification for Django web application is: Creating a web application and web service (API) for creating a file where each line represents the average between two neighboring integers. The input files contain 200000 integers.

IV. Azure Services

Microsoft Azure, sometimes known as Azure, is a cloud computing platform run by Microsoft that offers application and service access, management, and development across globally dispersed data centers.

A. Azure Virtual Machine

Azure Virtual Machine is a cloud computing option that allows users to deploy and manage virtual machines on the Microsoft Azure cloud platform. Azure Virtual Machine enables users to create and administer virtual machines in the cloud without having to worry about the supporting infrastructure since virtual machines are essentially simulations of physical computers. The versatility of Azure Virtual Machine is one of its main advantages. Depending on their particular needs, users can select from a range of pre-configured virtual machine sizes or build new virtual machines with the exact specs they need. If their processing demands change, users can simply scale up or down thanks to this, all without having to buy and maintain real gear.

B. Azure Function

One of the various cloud computing choices that aims to make scalability less painful is Azure Function. A serverless app has a different strategy than a continuous operation: it waits for requests to arrive, then starts as many instances as necessary to deal with them, shutting down once the job is finished. There is no fee if no one uses the service. Scalability, on the other hand, is already covered in the event that your software becomes an overnight success. In a serverless framework, there is also the additional benefit of not having to worry about operating systems; the platform is abstracted in a way that allows developers to concentrate on creating their web apps.

C. Azure Container App

Another cloud computing choice that provides a scalable and adaptable method for delivering and managing containerized applications is Azure Container App. In contrast to conventional server deployments, the Azure Container App abstracts away the underlying infrastructure and operating system to free developers to concentrate entirely on their application code. As a result, the application may be developed more quickly and managed more easily throughout its lifecycle.

V. Application deployment

This is an essential part of the project - application deployment on a cloud provider, Microsoft Azure.

A. Server hosting

We set up Virtual Machines using Ubuntu as the operating system and configured them with a 64-bit (x86) architecture. As part of the setup process, we generated a new key pair to facilitate remote SSH access. Additionally, we configured the firewall rules to allow for SSH, HTTP, and HTTPS access. To deploy our Spring application, we installed JDK and Tomcat web server since our application is built with Java. Additionally, we added a firewall rule to allow incoming traffic on port 8080. Next, we created a '.war' package and uploaded it to the manager page of the Tomcat server client portal.
To deploy our Django application, we installed the necessary dependencies including python3-django, python3-pip, and python3-venv, as well as Nginx. We then configured Nginx to act as a reverse proxy for our application.

B. Serverless hosting - Azure Functions

To deploy our application on Azure Functions, we'll need to make some modifications. Firstly, we'll need to split the application into two parts: the first being an API Azure handler that returns a sorted array of numbers, and the second being a REACT application that renders the data received from the AJAX API call. As our application is built with Java and Springboot, we'll use the Spring Cloud Function module to implement the backend. Once we're done coding, we can easily deploy the function to Azure using the command line interface.

C. Serverless hosting - Container App

We created a Docker image of our application in order to containerize it. Then, on Microsoft Azure, we created a Container Registry and pushed the image there. We proceeded to build the Azure Container App using the previously pushed Docker image. Additionally, we manually activated Azure Container Instances for our research and found that they can be integrated with Azure Functions.

VI. Testing methodology

After completing the application deployment phase, we moved on to performance testing by writing the request generator. The pseudo code for the generator is as follows:

Result: Total processing time
Function `upload(`*url, req*`)`**:**
  create a new executor service with a fixed thread pool of size req create an empty list for processing times increase the maximum number of retries for HTTPS connections repeat req times do create a file with random numbers using createFileWithRandomNumbers function create a new task to upload the file submit the task to the executor service shut down the executor service wait for all submitted tasks to finish executing before proceeding sum up the processing times in the list and print the size of the list and the sum return the sum as a string
end
Result: Absolute path of the file

Function `createFileWithRandomNumbers(`*num*`)`**:**
create a random number generator create an empty string builder repeat 200000/500000 times do append a random number between 1000 and 10000 to the string builder append a newline character to the string builder create a file create a buffered writer for the file write the contents of the string builder to the file close the buffered writer return the absolute path of the file
end

Pseudo code for creating a file with random numbers and uploading it to a given URL using multiple threads

VII. Results

We have reached the part of the analysis of the results which is the purpose of this paper.

A. Spring Boot application

Scalabilitity on Virtual machine
The testing phase involved uploading a CSV file with 500,000 numbers 100 times and calculating the average processing time for all requests. We conducted this test as the processing time for each request can vary due to several factors such as cold or warm start. Our results indicate that the average processing time for the application hosted on an Azure VM is 343.1 milliseconds, with most requests taking between 250-500ms. To further evaluate the system, we conducted an experiment where we sent 10 requests simultaneously to the virtual machine API handler. However, the server crashed and was unable to handle all the requests at once. Scalability on Function App
During the testing phase, we uploaded a CSV file with 500,000 numbers 100 times and calculated the average processing time of the application. The results showed that the average processing time of the Function App was 880.43 milliseconds. Most of the requests had an execution time in the range of 500-1000ms. However, when we sent 10 requests simultaneously, we saw a different result. Unlike the VM, the Function App was able to handle the requests and return the proper response. If we sent more than 10 requests, the Function App would manage to process them all. The Function App waits for requests to come in and fires up as many instances as needed to handle the requests, shutting down once the work is complete. The following chart shows the average processing time as we send multiple parallel requests to the Function App.
If we do the same experiment as we did on the Virtual machine with sending 10 request simultaneosly, we will see a completely different results. The results are that the function app unlike the VM, handles the requests and returns the proper response. If we send more than 10 requests the function app will manage to process all the requests. Azure function app waits for requests to come in and fires up as many instances as needed to handle the requests, shutting down once the work is complete. In Table I we present processing times of Azure Function, and in Fig 2 we visualize them.

Requests	5	10	15	20	25	30	35	40
Processing time	2171	3077	3245	2642	2361	2158	2671	1897

TABLE I
Azure Function from Spring Boot application processing time

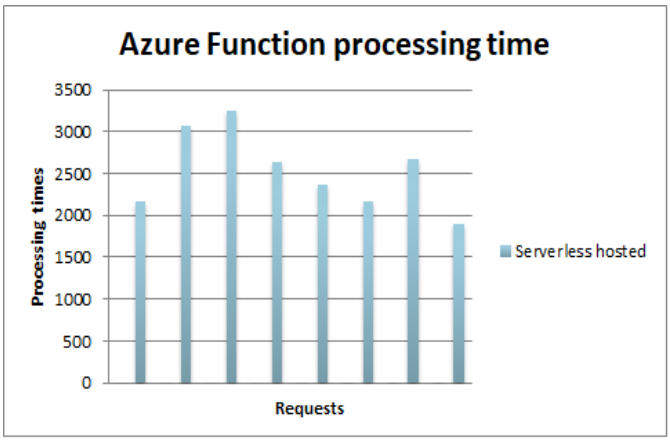

Fig. 2. Azure Function processing time

B. Django application

In this section, we analyze the processing times of a deployed application in both server and serverless environments. At the outset, we observe a slight difference in processing times for a single file upload: the server takes 279ms, while the serverless approach takes 249ms. The Table I. shows the that as the number of requests increases, the difference in processing times becomes more pronounced.

Requests	10	20	30	40	50
Type of service	Processing time in milliseconds				
Server	3403	7986	12647	23005	25024
Serverless	2564	5075	7577	10221	12730

TABLE II
Django application processing time comparison up to 50 different request loads

While testing the serverless hosted application, some issues were encountered where not all requests were successful for requests above 50. To address this, for a maximum of 100 requests, the failed requests were sent while maintaining a similar level of parallelization. Despite this issue, the processing times for the serverless hosted application were consistently lower compared to the server-hosted application. From the initial testing stage with a single file upload, there was only a minimal difference in processing times, with the server processing at an average of 279ms and the serverless hosted application processing at an average of 249ms. However, as the number of requests increased, as we can see from Table II. the processing time difference between the two became more noticeable, with the serverless hosted application consistently outperforming the server-hosted application in terms of processing time.

Time performance analysis showed that serverless hosted applications performed better. From Fig. 3 we can see that

Requests	60	70	80	90	100
Type of service	Processing time in milliseconds				
Server	26509	29395	38962	42629	41525
Serverless	15357	17914	20368	22868	25356

TABLE III
DJANGO APPLICATION PROCESSING TIME COMPARISON FROM 50 UP TO 100 DIFFERENT REQUEST LOADS

Serverless hosted applications are sometimes even (roughly) twice as fast as server hosted applications.

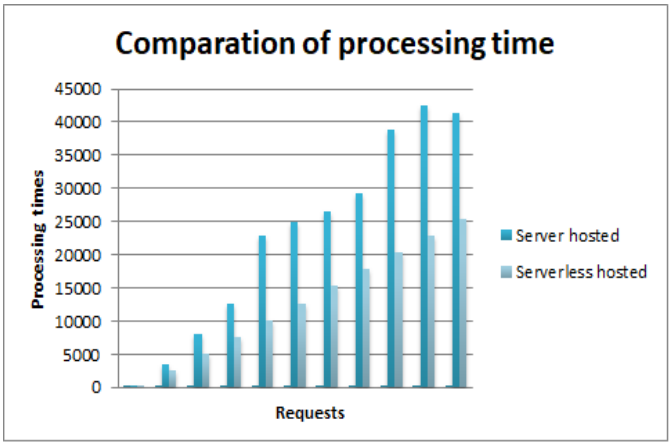

Fig. 3. Django application processing time comparison up to 50 different request loads

VIII. FUTURE WORK

Our future plans involve transforming a Spring Boot application into an Azure Function and a Django application into a Container App. We aim to conduct a comprehensive test to compare the results between these different deployment approaches. Additionally, we plan to explore the reverse scenario, where we will develop Django and Spring Boot applications based on specific project requirements.

IX. CONCLUSION

In this paper, we compared the performance and key differences between Server architecture and Serverless architecture. We observed that the Spring Boot server hosted application is sometimes twice as fast compared to the serverless application, but it is less scalable. In contrast, the Django application serverless hosted application (Container App) is twice as fast as server hosted application but has problems with resubmitting requests. Therefore, we conclude that serverless hosting may not be suitable for mission-critical applications. Organizations should carefully consider the trade-offs between speed, scalability, and reliability when choosing between server architecture and serverless architecture.

Server based solutions that use virtual machines offer better control over the environment, are better suited for long-running processes, provide more predictable performance, and can be more cost-effective for certain workloads. While serverless solutions are advantageous in many situations, virtual machines may still be a better choice in certain circumstances, such as applications that require specific software or hardware configurations, have long-running processes, strict performance requirements, or consistent and predictable resource usage. If stability is a priority and there is a good team to manage server infrastructure, then server-based architecture is the logical choice. Otherwise, if speed is a priority and it is not a mission-critical application, then serverless architecture using Container App is the better choice. Serverless computing is a growing trend and will likely be adopted more in the future. However, serverless computing also has its challenges. Since the cloud provider manages the underlying infrastructure, developers have less control over the runtime environment, making debugging and optimization more difficult. Additionally, serverless computing can introduce new security concerns, as code and data are stored and executed on third-party servers. Despite these challenges, serverless computing continues to be a growing trend in the world of cloud computing, with many organizations adopting it to increase efficiency, reduce costs, and improve their overall agility. As serverless technology continues to evolve, we can expect to see even more innovation in this space and new use cases emerging.

X. ACKNOWLEDGEMENT

This research was conducted as students projects for Cloud Computing course at Faculty of Computer Science and Engineering at ”Ss Cyril and Methodius” University in Skopje.

REFERENCES

[1] Virtual machines in Azure, available at https://learn.microsoft.com/en-us/azure/virtual-machines/
[2] Azure Functions documentation, available at https://learn.microsoft.com/en-us/azure/azure-functions/
[3] Azure Container Apps documentation, available at https://learn.microsoft.com/en-us/azure/container-apps/?ocid=AID3042118
[4] Jonas E, Schleier-Smith J, Sreekanti V, Tsai C-C, Khandelwal A, Pu Q, Shankar V, Carreira J, Krauth K, Yadwadkar N, Gonzalez JE, Popa RA, Stoica I, Patterson DA (2019) Cloud Programming Simplified: A Berkeley View on Serverless Computing.
[5] Baldini I, Castro P, Chang K, Cheng P, Fink S, Ishakian V, Mitchell N, Muthusamy V, Rabbah R, Slominski A, Suter P (2017) Serverless Computing: Current Trends and Open Problems In: Research Advances in Cloud Computing, 1–20.. Springer, Singapore.
[6] Hassan B. Hassan, Saman A. Barakat, Qusay I. Sarhan (2021) Survey on serverless computing. In: Journal of Cloud Computing
[7] Server vs. Serverless: A Performance Test, available at https://www.singlemindconsulting.com/blog/server-vs-serverless-a-performance-test/