Serverless Architecture

Ankita Tripathy

Cloud Computing
Karim Djemame
March 2022

Chapter 1 : Introduction

1.1 Context

Serverless Computing is a service that allows code to be triggered without the requirement of a physical platform to do so. It provides a cloud based architecture for the code to run. The code is triggered on demand without the explicit need of storing it on a server. Azure Function Apps provides us with a functionality where in we can directly write our code on the portal and can also use various triggers such as HTTP Trigger, Time Trigger, Event Trigger, etc.

Azure Function Apps is a service (Open Service) to run small pieces of code on the cloud system. This runs on Pay-per-use model which is an extension of the Azure App Service. Not only do they support multiple languages including but not limited to C#, Java, Python, Powershell but also supports integrated security services. Pieces of code hosted on the cloud can also be integrated with external tools. Additionally, integration with GitHub, DevOps is also supported.

To run a function app, we would need a trigger where it would run the code in order to produce an output. A function has only one trigger but can have multiple inputs and outputs. Since the model pricing is based on its use, major disadvantages to server less computing include accidental overload, lack of operational tools, limited API resources.

Chapter 2: The Paradigm

2.1 Server based computing vs. Serverless computing

Server based computing typically follows a 3 tier architecture consisting of a database layer, application layer and presentation layer where as server less computing follows a similar architecture in the form of micro services in the application layer and is independent of a physical infrastructure and hosted on the cloud. To be able to perform server based computing one would have to follow various installations and setup which could potentially take up a lot of space where as serverless computing allows a functionality to develop functions on web.

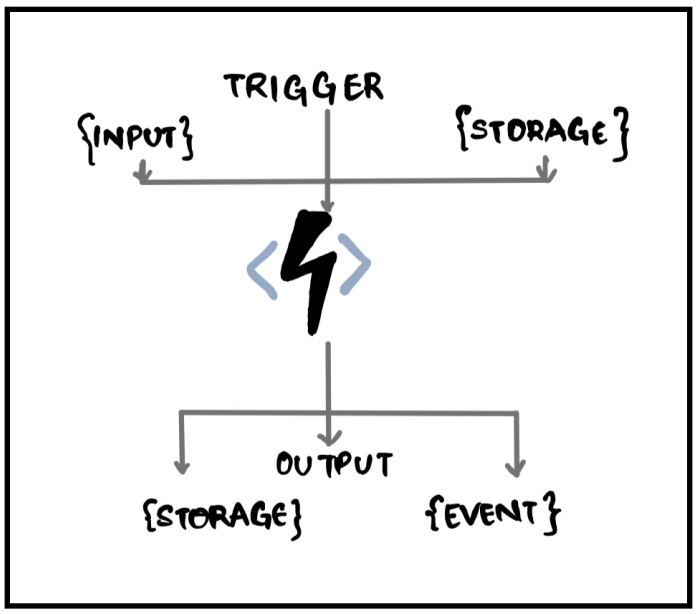


Fig 2a : What triggers a function app?

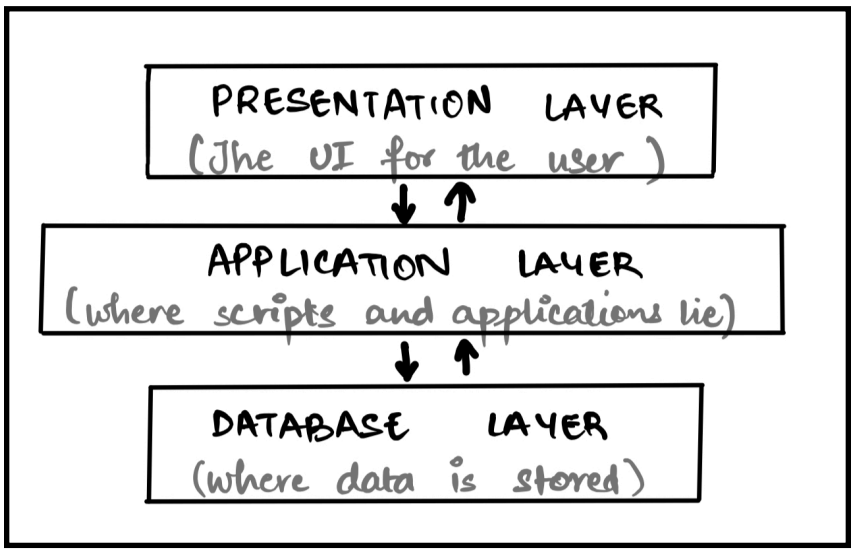


Fig 2b: Architecture: Server based computing

2.2 Commercial use : Serverless Computing

Aside from the now established fact that Serverless Computing requires zero servers, operating systems or related software/hardware, one of the major benefits it has to provide is the built in high availability and fault tolerance irrespective of the scale of business.

The serverless, infrastructure independent architecture provides the option to a developer to execute their scripts and code as and when required. Serverless functions are triggered on specific events based on the kind of function triggers allocated.

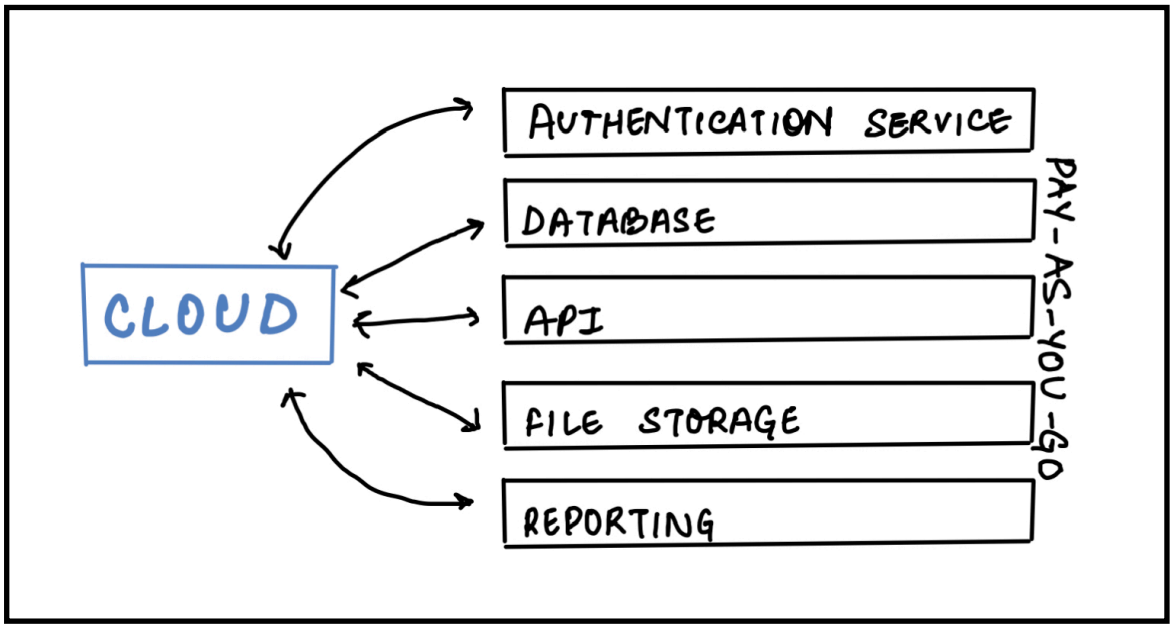


Fig 2c: Serverless model and its functionalities

Choosing to work with Function As A Service (FanS) would be helpful to a commercial entity because of its sheer ease of development and integration with an existing or a new CI/CD pipeline. Transitioning from one cloud service to another is also not a complex process which could pose to be an easy trump for anybody willing to switch to a serverless model.

In its best, business applications to Cloud computing include

A. Integration with IDEs and IoT based devices including circuits, sensors, phones, etc.

B. Serverless structure enabling no idle capacity cuts back on latency and enables cost effective solutions.

C. A piece of code and its resources on the cloud can be scaled up and down as and when required and provides for a comfortable mode of deployment that is hassle free and easy for a novice.

Chapter 3: Experimental Design

3.1 Function Apps

To compare and evaluate the performance of Azure function apps, I have created two function apps *viz.* sc20at and sc20atsecond. The details have been tabulated below :

	Sc20at	Sc20atsecond
Runtime stack	.NET	Node.js
Number of functions	2	2
Function #1 definition	Displays the value of the factorial of 10.	Displays the value of the factorial of 10.
Function #2 definition	Counts the number of palindromes between 1 and 2000	Counts the number of palindromes between 1 and 2000
URL	https://sc20at.azurewebsites.net/	https://sc20atsecond.azurewebsites.net/

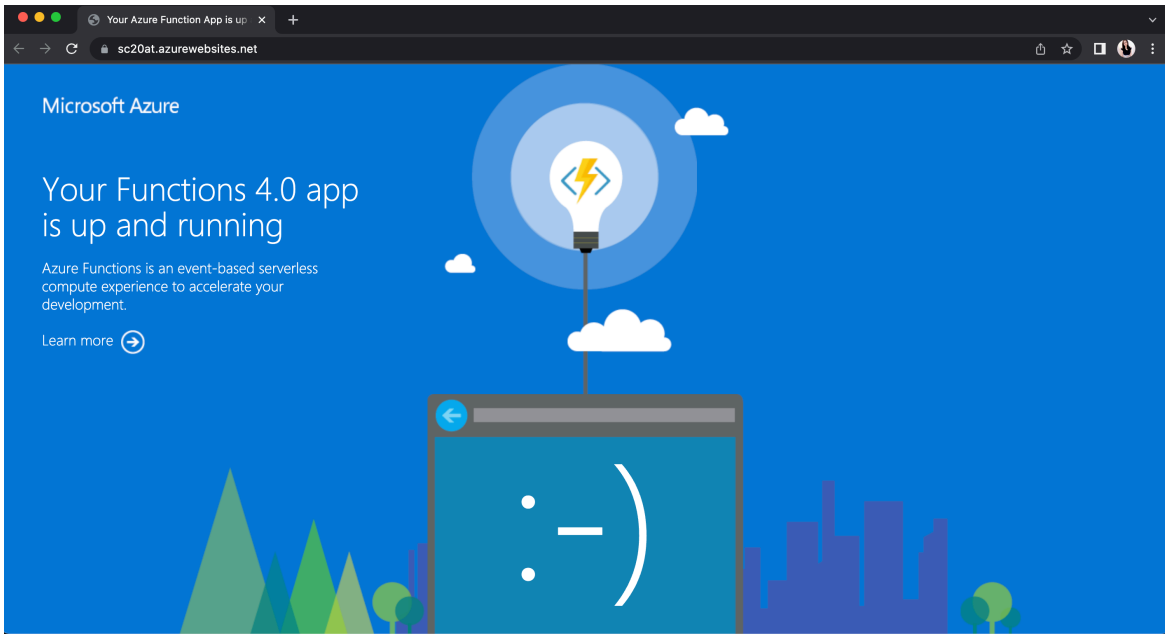


Fig 3a: Function app url : sc20at

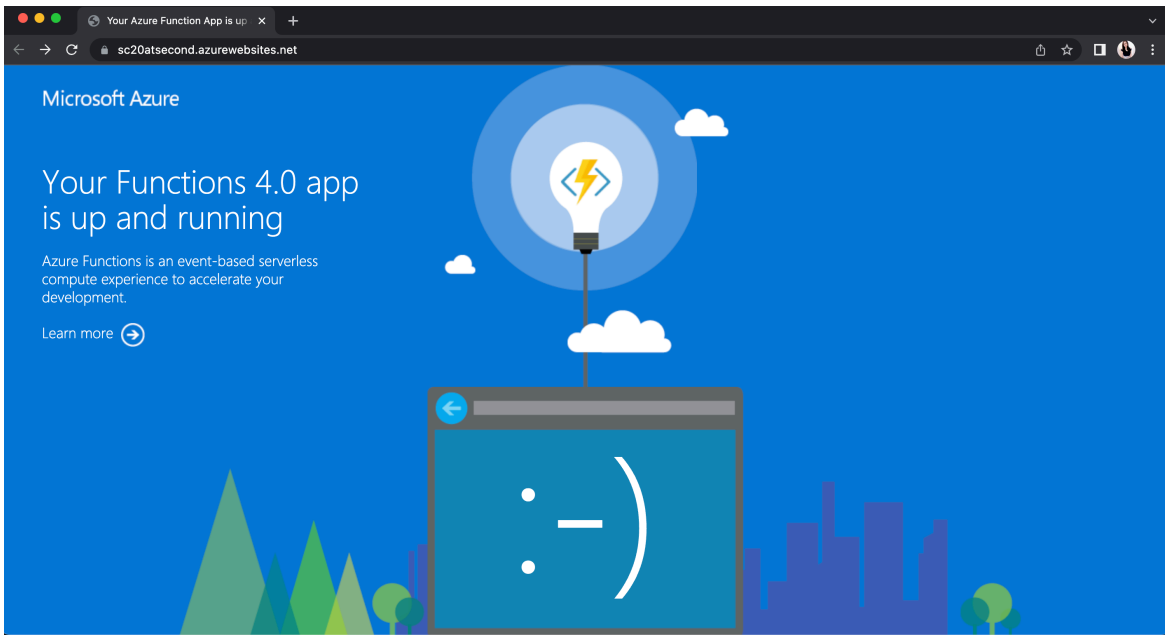


Fig 3b: Function app url : sc20atsecond

3.2 Function Apps - Output

3.2.1 Function App : sc20at

Fig 3c: HTTPTrigger function to print the factorial of 10 triggered using the function URL.

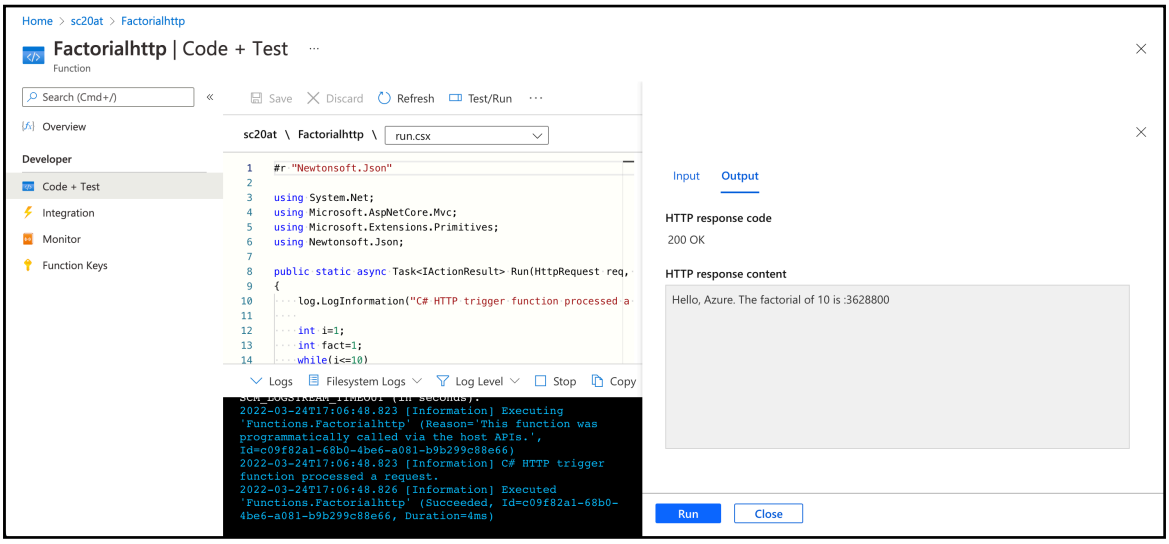


Fig 3d: HTTPTrigger function tested on the Azure Web Portal to prints the factorial of 10

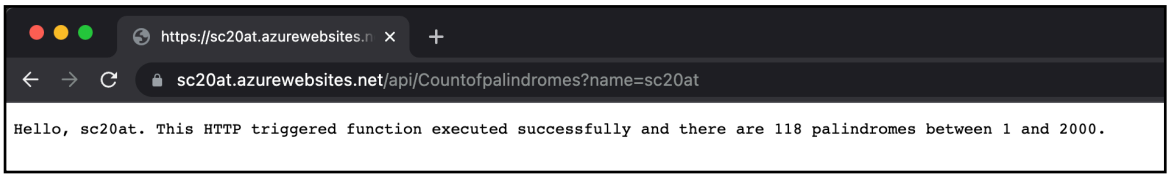


Fig 3e: HTTPTrigger function to print the count of palindromes between 1 and 2000 triggered using the function URL.

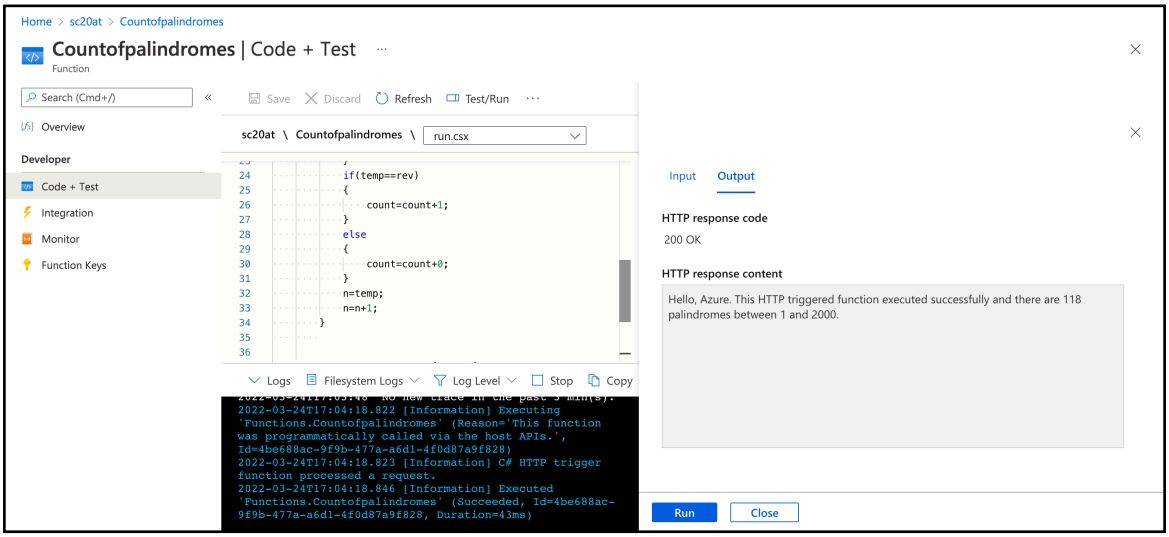


Fig 3f: HTTPTrigger function run on the Azure Web Portal to print the number of palindromes between 1 and 2000.

3.2.2 Function App : sc20atsecond

Fig 3g: HTTPTrigger function to print the factorial of 10 triggered using the function URL.

Fig 3h: HTTPTrigger function tested on the Azure Web Portal to print the factorial of 10

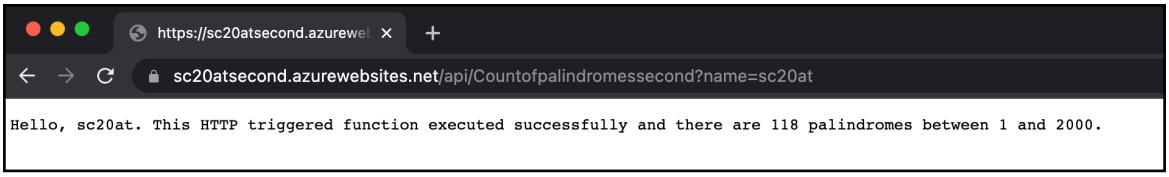


Fig 3i: HTTPTrigger function to print the count of palindromes between 1 and 2000 triggered using the function URL.

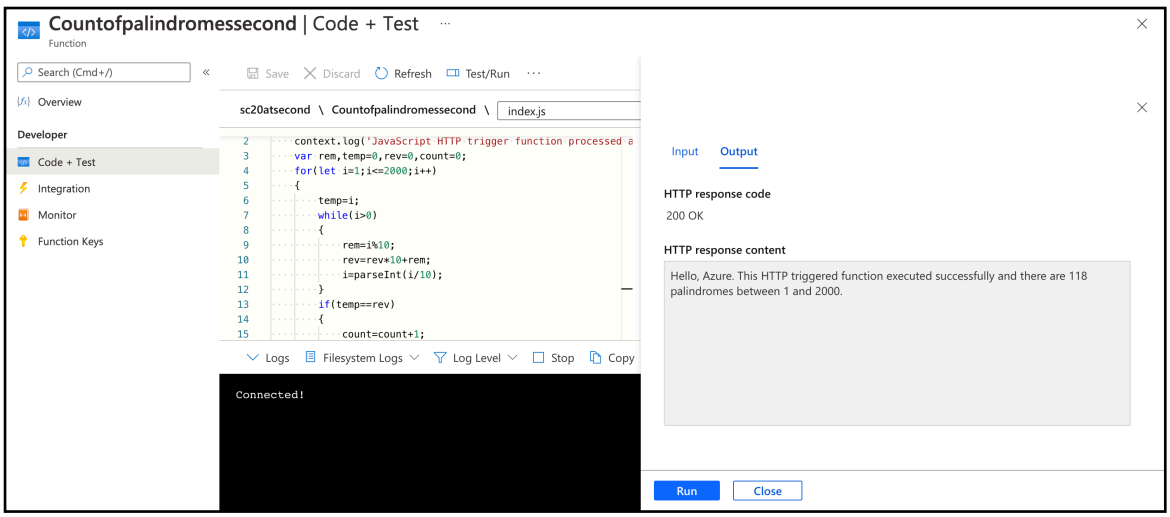
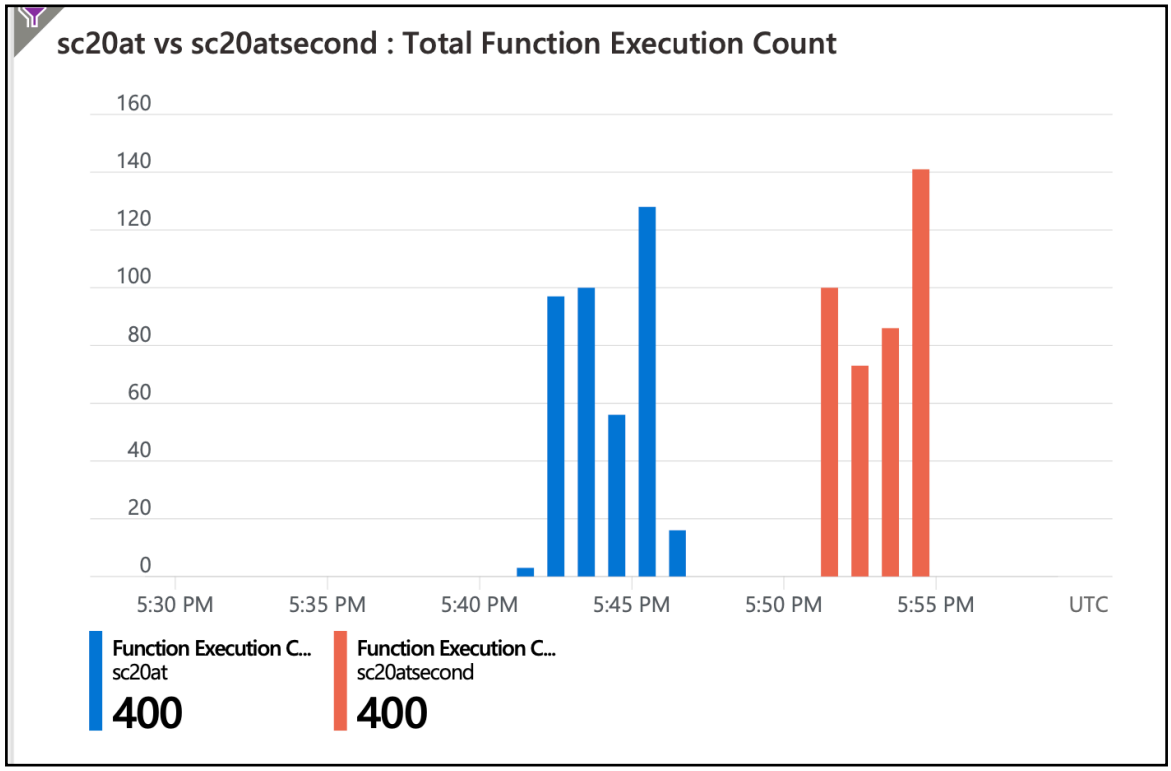


Fig 3j: HTTPTrigger function run on the Azure Web Portal to print the number of palindromes between 1 and 2000.

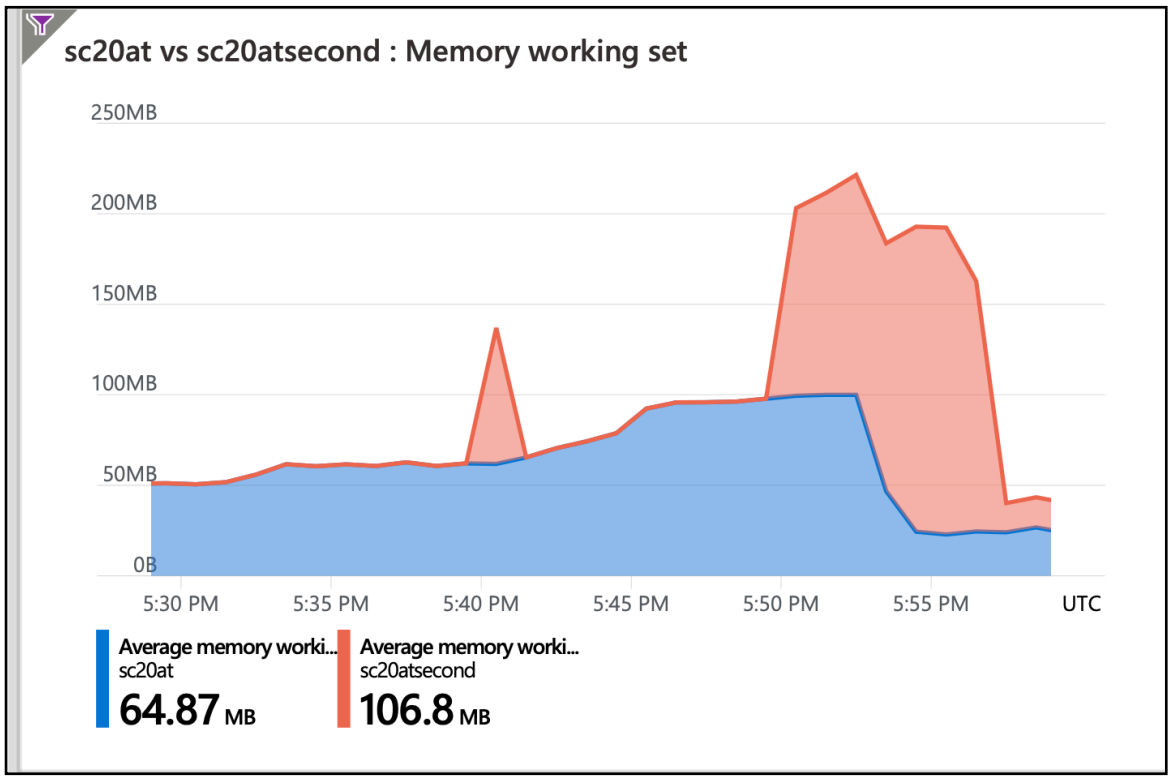
Chapter 4 : Experiment Outcome

4.1 Function Execution Count



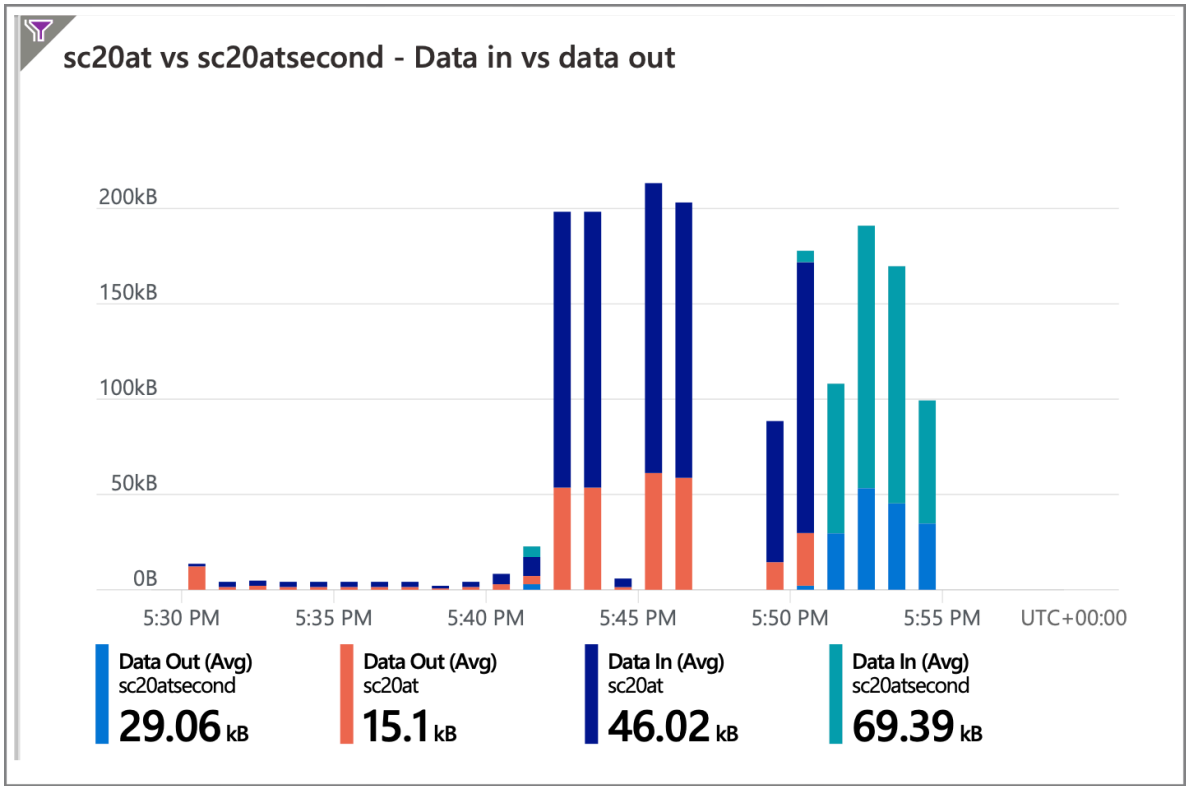
To measure the execution performance of the implementation, I wrote a python function that invoked each function 200 times by trigger the URL via a GET method. The graph depicts every the total function execution count of both the function apps to be a total of 400 each.

4.2 Average memory working Set



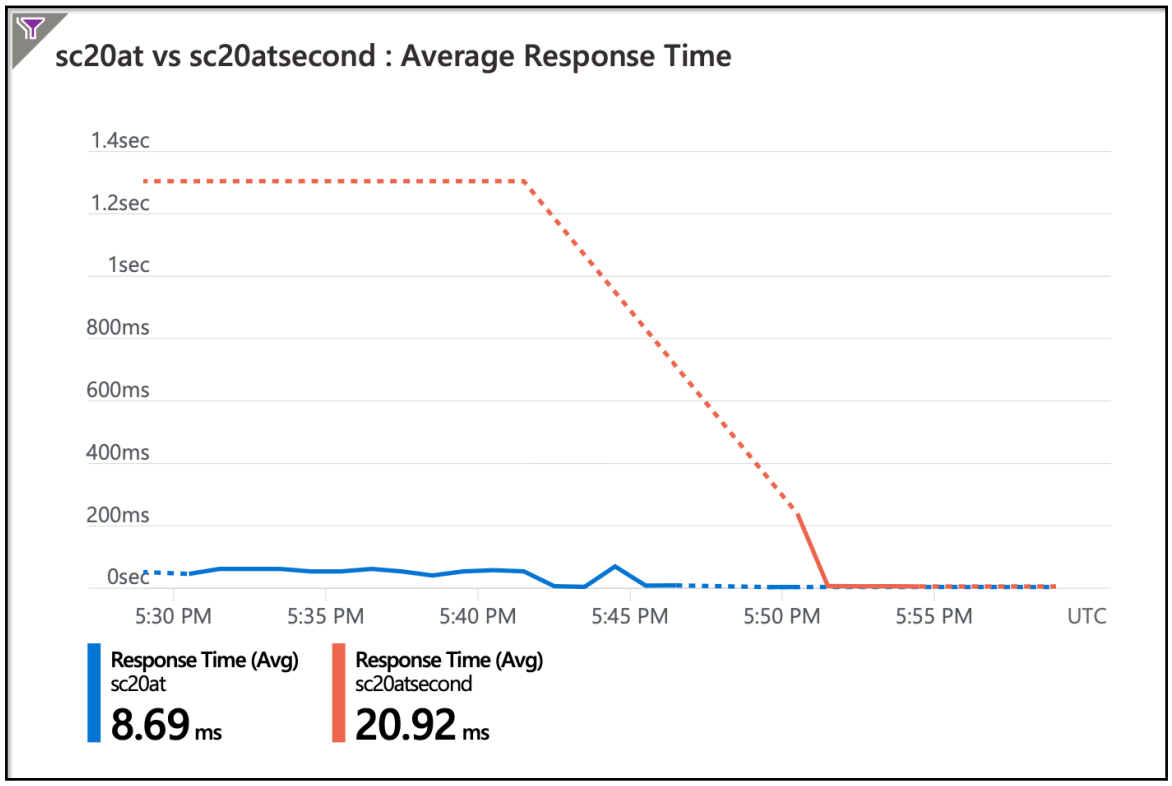
Having invoked the functions, that follow the same algorithm and adhere to the same time and space complexity, for an equal number of 200 times each, the average memory working set reserved by each function app is of remarkable difference. The function app that uses .NET as a framework utilises almost half as much as that of node.js. This could be one of the major reasons why .NET is a supported runtime stack in all the Cloud Services present in the industry today.

4.3 Data in vs Data out



Neither of the functions require an input per se, except for the name of the user. It is observed that the Data In value for sc20at is lower than that of sc20atsecond and the same trend has been observed in terms of Data out value. However, the ratio of Data in to Data out is similar for both the function apps. It has been observed that, there is a flux in data in and out value even when the functions have not been triggered explicit which could pose as an anomaly .

4.4 Response Time



The average response time for sc20atsecond is again observed to be more than double as that of sc20at. Additionally, latency of very minimal amplitude was observed for the stack that used Node.js. Overhead experienced by both of the function apps were considerably different even though the number of triggers per function were the same. Despite the synchronous HTTP Triggers, there were minimal but prevalent latencies.

Chapter 5 : Conclusion

Serverless computing is a technology driven solution that eradicates the woes of space and extensive server installation processes. Due to the numerous benefits, the utilisation of cloud services by commercial organisations will see a substantial increase. The performance owing to the fact that it is flexible, scalable and cost effective is an optimal solutions for organisations today in the world of space deprivation.

References

- *Citated in Paper: G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 405-410, doi: 10.1109/ICDCSW.2017.36.*

- *S. Xu, S. M. Ghazimirsaeed, J. M. Hashmi, H. Subramoni and D. K. Panda, "MPI Meets Cloud: Case Study with Amazon EC2 and Microsoft Azure," 2020 IEEE/ACM Fourth Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware (IPDRM), 2020, pp. 41-48, doi: 10.1109/IPDRM51949.2020.00010.*

Link to GitHub repository : Click here to navigate to GitHub repository