*Grenze International Journal of Engineering and Technology, Jan Issue*

# Triggering Issues Related to Serverless Function in Serverless Computing Environment

Shashank Srivastava[1] and Dr. Santosh Kumar[2]
[1]Research Scholar,Shri Ramswaroop Memorial University, Lucknow, Uttar Pradesh,India,225003
Email: shivam.shashank@gmail.com
[2]Associate Professor, Shri Ramswaroop Memorial University, Lucknow, Uttar Pradesh, India,225003
Email: sant7783@hotmail.com

***Abstract*—In today's global scenario, cloud technologies are leading the world by facilitating the end users by providing the flawless infrastructure, platforms and services to every community. These features of Cloud are known as IaaS[1],PaaS[2]and SaaS[3] but recent days one more model of cloud technology is in its execution, which is known as function as a service [4] or serverless computing. The term serverless was firstly coined by Amazon in 2014 started with AWS[5] Lambda, after 2014 few other vendors also start thinking on the same, where Google started with Google cloud functions [GCF][6], Microsoft started with Microsoft Azure [7]Functions and IBM Open Whisk [8,9]. However, the serverless approach to computing is not completely new.**

**In this paper we have included the role and importance of serverless computing model, discussed various serverless computing platforms and prepare a comparatively study of various serverless[10] platforms on different parameters, This will help to the developers in selecting better cloud platforms to run their codes.**

***Index Terms*— Serverless Computing, Serverless Functions, Cloud Computing, Lambda, Azure, Open Whisk, Cloud Computing.**

## I. INTRODUCTION

In general developers spend uncounted hours in solving business problems with code. These uncounted hours are basically based on two important tasks:

- Firstly in getting the code written by developers and to make it available for any computers.
- Secondly for making sure that these available computers work smoothly. Practically this task is a very long process.

Then to get rid of these long processes, there was a term coined "Serverless" in earlier of 2012, which given the concept to leave this task on serverless model providers and from there the journey of serverless computing model came into existence.

In "server awake"technology adhere the services where developers have to think from selection of operating systems to deployment of their code and its execution by their own, but in serveless computing developers have to just put or upload their piece of code to the interface of the serveless computing vendor's application and these vendors having the responsibility to run or execute these code not by the developers.

In Serverless computing users have to just "pay per use" not for storing their codes on serverless platforms. The

*Grenze ID: 01.GIJET.8.1.502_1*
*© Grenze Scientific Society, 2022*

charges are only applicable only when user's code executed each time. This saves lot to money of the users and motivate the users to choose serverless platforms to run their application because in server awake modeling, users have to pay per second or pay per hour for using server related services, either they are using the server services or not.
Serverless computing model provides the automatic provisioning and maintenance of users' code without any interference of its users, which makes the serverless platforms highly available for all.
Serverless computing model never enforced any non-availability of servers, there are servers but as developers have not to interact or think about servers, makes this term as serverless.
Serverless platforms are generally provide the facilities of Platform-as-a- service where Server awake computing models are based on Infrastructure-as-a-service because in server awake computing model, developers can interact with the server related hardware and software. Sereverless[10] computing provides the robust, scalableservices. The codes written by the developers are called as server-lessfunction. These functions are entitled to respond to externalevent, when they are required to invoke. In serverless computing developers can set the instructions for triggering the execution of its code; but there are lots of consequences behind this triggering of these code because these codes are not designed to run on serverless platforms.

## II. Commercialized Cloud Platforms

Amazon's AWS Lambda [15] was the first serverless platform and includes various Important features including cost, programming model, deployment, resource limits, Security, and monitoring. Supported languages include Node.js, Java, Python, and#. Initial versions had limited compos ability but this has been addressed recently.
The platform takes advantage of a large AWS[5] ecosystem of services and makes it easy to use Lambda functions as event handlers and to provide glue code when composing services.
Currently available as an Alpha release, Google Cloud Functions [6,15] provides basic FaaS[4] functionality to run serverless functions written in Node.js in response to HTTP calls or events from some Google Cloud services. The functionality is currently limited but expected to grow in future versions.
Microsoft Azure [7,15] Functionsprovides HTTP web hooks and integration with Azure services to run user provided functions. The platform supports C#, F#, Node.js, Python, PHP, bash, or any executable. The runtime code is open-source and available on GitHub under an MIT License. To ease debugging, the Azure Function.CLI provides a local development experience for creating, developing, testing, running, and debugging Azure Functions.
IBM Open Whisk [8,9,15] provides event-based serverless programming with the ability to chain serverless functions to create composite functions. It supports Node.js, Java, Swift, Python, as well as arbitrary binaries embedded in a Docker container. Open Whisk is available on GitHub under an Apache open source license.

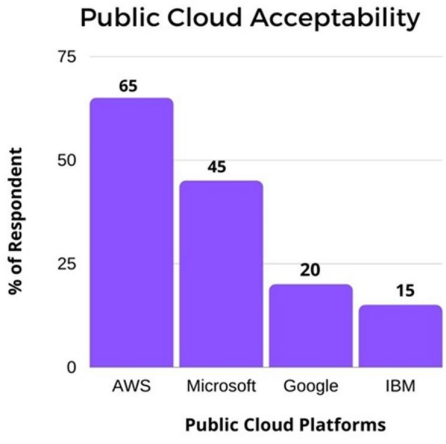


Figure 1: Public Cloud Acceptability

## III. Problem Definition

Serverless[1] computing provides the robust, scalable services. The codes written by the developers are called as serverless function. These functions are entitled to respond to external event, when they are required to invoke.

In serverless computing developers can set the instructions for triggering the execution of its code.But there are lots of consequences behind this triggering of this code because these codes are not designedto run on serverless platforms.

The main challenges and Problems in triggering the serverless functions **[14]** are as Follows:

**Transient state:** Basically, there are two states of any container in serverless platforms. The container will be either in idle state or in active state**.**Idle state of container indicates that container is not involve in any execution whereas Active state represents that container is somehow busy in executing the serverless functions**.**

The first challenge evolves after a few minutes of idleness, at this time all the related information of global variables are unidentified or lost. This challenge generated when the serverless functions are executed in the container which can hold short term state in its execution and suddenly close down when container reached to the idle state. When container shut down itself by reaching its idle state, this information is not known to the developers because developers do not get any notification regarding that.

When any new request to execute the serverless function arrives to the container, the serverless platform automatically invoke new container to run that serverless function but because all previous information related to developers code execution is lost therefore function must have to serialize all its previous state update to a persistent store.

**Indirect Execution**. Another challenges in triggering the serverless functions is implicit parallel execution. This challenge arises when one serverless function receives more than one events in short span of time. In this case functions exhibits high load and to manage this server invokes new instance to execute the requests in line. Each requests in line are executed in new allotted instance by the server and this execution is done in an isolated environment which is not known to the developers also.

Therefore there is a possibility that is these inline functions are dependent on each other's outputs, in that case to measure the overall correct output is a big challenge because of parallel execution of requests.

**Just One Time Execution.** This challenge arises when any failure occurs in infrastructure of the serverless service provider. In serverless there is a possibility that one function needs to be executed multiple times to complete any single task and each function executed at least one time in this execution lifecycle so it may be possible that if one function becomes faulty during its execution than it may influence other functions also, because all functions can run parallel which we already discussed in our challenge of Parallel execution and it don't have any log records so to identify the faulty function is also a big challenge.

Our research work will focus on to develop the better mechanism to address above mentioned challenges in a more effective way with the better security model.

## IV. Related approaches to solve the Problems/Challenges

Large number of authors proposed different methods for solving the problem of triggering serverless function in Serverless environment.

**Jangda, Abhinav**"et.al;"**[14]** given the solution by using key value store for persistence, transactions to address concurrency and unique identifiers to support safe re-invocation using naïve bayes theorem.

We can use the following methods to enhance the triggering of serverless functions in serverless environment.

## V. Proposed Methodology

**Methodology 1:**Our method is based on checkpoint methodology, where all states of serverless function can involve with one checkpoint. With this checkpoint insertion, in case of ideal stage of serverless container, the all states of serverless function will not be diluted we can roll back to the previous successful state of the serverless function and resume its execution. This will save the time and efforts of the developers.

**Methodology 2:**Our second method is based on QUEUE principal, when server will receive multiple requests at a time, all the incoming request will be arrange in Queue principal means FIFO (First In First Out), First incoming request will be executed first in the container.

**Methodology 3:**Our Third method is based on by fixing the maximum execution time of any function in container. If any function execution will take more than the maximum allotted time, Every time container will stop its execution and will notify the developers to recheck the serverless function and marked as suspicious function

**Methodology 4:**Our Fourth method is based on Probability Distribution Function (PDF) by which we can identify the occurrences of any serverless function, retries and failure of any function so that developer can easily get to know that on which serverless function he/she has to give proper attention.

## VI. Future Scope

Effective methodology for Triggering of serverless function is very important in the in serverless environment as many developers use the serverless platforms to execute their program function due the distinct features of the serverless computing. After the process of proposed methods, we will be able to find the most appropriate methods so that serverless functions and serverless environment can work in most effective way.
we can provide the environment where triggering of serverless functions can be more compatible with these platforms easily and can address the challenges related to serverless function execution in serverless platform because demand and acceptability of Public Cloud platforms are growing very fast with time. The notification about each state of the function execution can also be built by the serverless service providers for the developers. This can be more beneficial for both the developers and service providers.

## VII. Conclusion

The conclusion of our research proposal is as follows:

- To provide the proper notification to the developers about each state of serverless function execution under serverless container
- To provide the better mechanism to address the serverless container when it reaches to the Ideal state and provide better methods to persist all its execution states
- To generate more effective methodology to identify the occurrences of serverless functions, its failure, re-tries
- To develop better security solution for serverless functions in serverless environment

## References


[1] Panda, Surya & Mehta, Ashima. (2018). Design of Infrastructure as a Service (IAAS[1]) Framework with Report Generation Mechanism.

[2] Cloud Computing Platform as Servicel, InformationWeek 16 Oct. 2, 2009

[3] Espadas, J., Concha, D., and Molina, A., "Application Development over Software-as-a-Service platforms,"In Proceedings of Software Engineering Advances(ICSEA 2008), IEEE, pp.97 - 104, October, 2008.

[4] Theo Lynn, PierangeloRosati, Arnaud Lejeune, and Vincent C. Emeakaroha. A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. In IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2017, Hong Kong, December 11-14, 2017, pages 162–169, 2017.

[5] Choudhary, Brijesh&Pophale, Chinmay&Gutte, Aditya &Dani, Ankit&Sonawani, Shilpa. (2020). Case Study: Use of AWS[5] Lambda for Building a Serverless Chat Application. 10.1007/978-981-15-0790-8_24.

[6] Moroney, Laurence. (2017). Cloud Functions for Firebase. 10.1007/978-1-4842-2943-9_8.

[7] D. Chappell. Introducing the Azure services platform. White paper, Oct. 2008.

[8] Kuntsevich, Aleksandr&Nasirifard, Pezhman& Jacobsen, Hans-arno. (2018). A Distributed Analysis and Benchmarking Framework for Apache OpenWhisk Serverless Platform. 3-4. 10.1145/3284014.3284016.

[9] Sampé, Josep&Vernik, Gil & Sánchez-Artigas, Marc &López, Pedro. (2018). Serverless Data Analytics in the IBM Cloud. 1-8. 10.1145/3284028.3284029.

[10] Shafiei, Hossein&Khonsari, Ahmad &Mousavi, P. (2019). Serverless Computing: A Survey of Opportunities, Challenges and Applications. 10.13140/RG.2.2.32882.25286.

[11] Heydari, Atefeh&Tavakoli, Mohammadali&Riazi, Mohammad. (2014). An Overview of Public cloud[11] Security Issues. International Journal of Management Excellence. 3. 10.17722/ijme.v3i2.166.

[12] Yunxia, Jiang & Bowen, Zhao &Shuqi, Wang &Dongnan, Sun. (2014). Research of Enterprise Private cloud[12] Computing Platform Based on OpenStack. International Journal of Grid and Distributed Computing. 7. 171-180. 10.14257/ijgdc.2014.7.5.16.

[13] Aryotejo, Guruh&Kristiyanto, Daniel &Mufadhol, Mufadhol. (2018). Hybrid cloud[14]: bridging of private and public cloud[11] computing. Journal of Physics: Conference Series. 1025. 012091. 10.1088/1742-6596/1025/1/012091.

[14] Jangda, Abhinav & Pinckney, Donald & Baxter, Samuel & Devore-McDonald, Breanna &Spitze, Joseph &Brun, Yuriy&Guha, Arjun. (2019). Formal Foundations of Serverless Computing,ACM Journal

[15] Baldini, Ioana& Castro, Paul & Chang, Kerry & Cheng, Perry & Fink, Stephen &Isahagian, Vatche& Mitchell, Nick &Muthusamy, Vinod&Rabbah, Rodric&Slominski, Aleksander&Suter, Philippe. (2017). Serverless Computing: Current Trends and Open Problems.

[16] Dutta, Pranay& Dutta, Prashant. (2019). Comparative Study of Cloud Services Offered by Amazon, Microsoft and Google. International Journal of Trend in Scientific Research and Development. Volume-3. 981-985. 10.31142/ijtsrd23170.