https://doi.org/10.46335/IJIES.2022.7.8.8 **e-ISSN: 2456-3463**
Vol. 7, No. 8, 2022, PP. 42-47

International Journal of Innovations in Engineering and Science, www.ijies.net

Effective Triggering Mechanism for Server less Environment

Shashank Srivastava[1], Dr. Bineet Kumar Gupta[2]

[1]Research Scholar,ShriRamswaroop Memorial University, Lucknow, Uttar Pradesh,India,225003
[2]Associate Professor, Shri Ramswaroop Memorial University, Lucknow, Uttar Pradesh, India,225003

shivam.shashank@gmail.com

***Received on**: 11 June ,2022* ***Revised on**: 29 July ,2022,* ***Published on**: 01 August,2022*

Abstract – *Cloud technologies are dominating the globe in today's worldwide environment by enabling end users by delivering faultless infrastructure. Every community will benefit from platforms and services. The Server less approach, also known as Function as a Service[FaaS], is now the most popular cloud technology paradigm,. Because of this, the server less approach is in high demand among numerous companies and individual stakeholders. No organization or individual is responsible for supplying and monitoring needed resources. It also works with a variety of programming languages. That is why the server less paradigm is so popular. However, the server less computing model is not entirely new. There are several options available nowadays.AWS Lambda, Microsoft Azure, Google Cloud Functions, and IBM Open Whisk are examples of server less service providers These server less solutions make it possible to individuals to execute their programmers in environments supplied. The state of code execution changes as individuals trigger the code to execute in a server less environment. Because the code operates in an isolated environment of the server less platforms, which is the biggest challenge of these platforms, it is unknown to the individual.*

In server less environment triggering of server less function is an important & crucial aspect because all the server less activities in server less environment depend on it. As in current server less environments there are many challenges which are directly or indirectly affect the behavior of server less function which may cause the major deviations in results and processing of developer's code in the serverless environment.

Therefore, in this paper we are proposing the possible challenges in serverless environment with their proposed solutions to overcome with those challenges.

.

Keywords- Serverless Computing, Serverless Functions, Cloud Computing, Lambda, Azure, Open Whisk, Cloud Computing, Serverless Computing

I – INTRODUCTION

Generally, programmers spend endless hours using code to solve business challenges. These many hours are mostly spent on two tasks: Firstly, finding out how to get the code that developers create running on whatever machines are available; secondly ensuring that those computers work smoothly. This is a task that will never cease.

To get rid of these lengthy procedures, a word called "Serverless"**[1,2]** was coined in early 2012, with the idea of delegating these tasks to serverless model providers, and so the serverless computing model was born.

In "server awake" technology, developers must think about everything from operating systems to code deployment and execution on their own, whereas in serverless computing, developers must simply put or upload their code to the interface of the serverless

https://doi.org/10.46335/IJIES.2022.7.8.8 **e-ISSN: 2456-3463**
Vol. 7, No. 8, 2022, PP. 42-47

International Journal of Innovations in Engineering and Science, www.ijies.net

computing vendor's application, and these vendors are responsible for running or executing the code, not the developers.

Users of server less computing must only "pay per usage," rather than saving their code on server less platforms. The fees are only applied when the user's code is run each time. This saves consumers a lot of money and encourages them to utilize server less platforms to execute their applications because in server awake modeling, users must pay per second or pay per minute.

II –MAJOR CHALLENGES IN SERVERLESS ENVIRONMENT

The main challenges and Problems in triggering theserverless functions as Follows:

State of flux: In serverless systems[14], any container can exist in two states. The container will be in one of two states: idle or active. The idle state of a container shows that it is not currently performing anything, whereas the active state indicates that it is actively executing server less operations.

The first obstacle appears after a few minutes of inactivity; at this point, all global variable information is misidentified or lost. This problem arises when serverless services are run in a container that may keep a short-term state during execution and then abruptly shut down when the container reaches the idle state. Because developers are unaware of when a container shuts down by reaching its idle state, this information is not available to them.

Indirect Execution. Implicit parallel execution is another problem in activating serverless functions. When a single serverless function gets several events in a short period of time, this problem develops. In this circumstance, the functions are overloaded, thus the server starts a new instance to process the requests in order. Each request in line is performed by the server in a new allocated instance, and this execution takes place in an isolated environment that is likewise unknown to the developers.
As a result, it's possible that these inline functions are dependent on one other's outputs, making measuring the total right output a difficult task due to request concurrent execution.

One-Time Execution.
When a failure in the serverless service provider's infrastructure happens, this problem develops. Because all functions can run parallel, as we discussed in our challenge of Parallel execution, and there are no log records, it's possible that one function needs to be executed multiple times to complete any single task, and each function is executed at least once in this execution lifecycle, it's possible that if one function becomes faulty during its execution, it could affect other functions as well.
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.

III- FEATURES OF SERVERLESS COMPUTING MODEL

Serverless [10,15] computing models offer a variety of distinct advantages for developers, making them suited for many types of developer communities. In this part, we'll look at several key characteristics of serverless computing to assist developers in selecting serverless platforms.

- **Financial**
  This characteristic distinguishes the serverless paradigm from all other server-aware systems. Developers must pay per usage only in the serverless paradigm, not for keeping their code on the server; scripts must either execute or not. Users that utilise serverless must only pay while their code is running, which is highly inexpensive and efficient.
- **High Availability**
  The scalability of resources is controlled automatically by the vendor, not by the developers, in serverless models. Serverless suppliers will control the selection of servers, RAM capacity, CPU utilization, and operating system based on the necessity to run developer code. In any case, this makes it extremely accessible.
- **Broad Spectrum of Programming Languages**
  Serverless models are compatible with a broad range of programming languages, from Java Script to Google's Go. NODE.js, Java, JavaScript #, Ruby, Python, and Go are also supported. Some serverless systems allow for code developed in any programming language to be extended using well-defined APIs bundled in a Docker image.
- **Model of programming**
  Currently, serverless systems run a single main function that accepts a dictionary JSON object as input and outputs another dictionary.
- **Integration**

https://doi.org/10.46335/IJIES.2022.7.8.8 e-ISSN: 2456-3463
Vol. 7, No. 8, 2022, PP. 42-47

Various models allow for the invocation of one serverless function into another, as well as the integration of these functions to handle complicated development.

- **Implementation**
  Any serverless service provider's goal is to make deployment as simple as possible. Normally, only developers require function source code in a file. Aside from that, code can be packaged as an archive with numerous files within or as a binary coded Docker image, among other choices.
- **Accounting**
  Serverless systems are multi-user and run in a separate environment. The associated detail of performed functions is presented to the user; this detail includes how many times the user's functions have been executed and how much they must pay for it.
- **Troubleshooting**
  Basic debugging is supported on all platforms via print statements that are captured in the execution logs. Additional features might be added to aid developers in locating bottlenecks, tracing problems, and so on**.**

IV-MONETIZED CLOUD PLATFORMS

Commercialized Cloud Platforms are those that may be utilized by any company or organization in exchange for a fee based on how much they use cloud platforms. There are a variety of commercialized cloud platforms available, some of which are listed below.

- **AWS Lambda**
  Amazon's AWS Lambda [15] was the first serverless platform. It has a number of useful features for resource deployment, price, security, and monitoring. It supports Node.js, Java, Python, and C# as mainframe languages. AWS Lambda [15] started out with modest capabilities, but it has now grown to become the most popular serverless service provider on the planet. It has the entire potential of lambda functions, making it more powerful. The code in AWS Lambda is performed in reaction to events in AWS services such as adding and deleting files in an S3 bucket, making an HTTP call to the Amazon API gateway, and so on. Amazon Lambda, on the other hand, can only be used to perform background operations.
- **Google Cloud Function**
  Google Cloud Functions [6,15] is a serverless execution environment that allows you to construct and link cloud services. We may develop simple, single-purpose functions that are linked to events broadcast by your cloud infrastructure and services using Cloud Functions. Google Cloud Functions provides rudimentary FaaS [4] capabilities to run serverless functions written in Node.js in response to HTTP calls or events from various Google Cloud services. It is currently available as an Alpha version. The present functionality is limited, but it is intended to expand in future editions.
- **Microsoft Azure**
  Microsoft Azure Functions [7,15] enables HTTP web hooks and Azure service connectivity to run user-provided code or functions. C#, F#, Node.js, Python, PHP, shell, and any other executable are supported. The runtime code is open-source and licenced under the MIT License on GitHub. The Azure Functions CLI provides a local development environment for creating, developing, testing, running, and debugging Azure Functions.
- **IBM Open Whisk**
  IBM Open Whisk [8,9,15] enables event-driven serverless programming and the creation of integrated functions by connecting serverless services. Node.js, Java, Swift, Python, and arbitrary binaries placed in a Docker container are all supported. Open Whisk is licensed under the Apache open-sourcelicense and is accessible on GitHub.

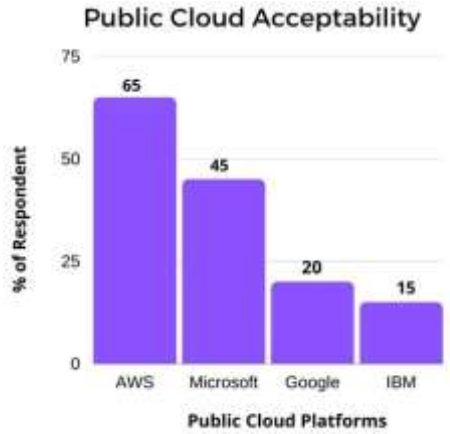

Fig.1-Different Cloud Environments Acceptability

V- DEMAND OF CLOUD ENVIRONMENTS

In general, there are three sorts of cloud environments that control practically all forms of web services.

Personal Cloud

The private cloud **[12]** encompasses all web-related services purchased by a single developer or company. It

https://doi.org/10.46335/IJIES.2022.7.8.8 **e-ISSN: 2456-3463**
Vol. 7, No. 8, 2022, PP. 42-47

International Journal of Innovations in Engineering and Science, www.ijies.net

might be housed on the organization's facilities or provided by a third-party provider. In a private cloud [12], the organization or owner owns all of the hardware and software. In this approach, managing and monitoring available resources in a private cloud[12] environment is always simple.

Public Cloud

All web-related resources are maintained by third-party suppliers in a public cloud **[11]** or serverless environment. Individuals can use the internet to access web services. Any other individual or business can rent services in the public cloud [11], and each individual must only pay for the services that they have used at any given moment.

Hybrid Cloud

A hybrid cloud system [14] serves as a link between public and private clouds [12]. It encompasses both public and private cloud [12] services. Businesses may seamlessly scale up their on-premises infrastructure to the public cloud[11] to manage any overflow without allowing third-party databases access to all of their data using hybrid cloud[14]computing.

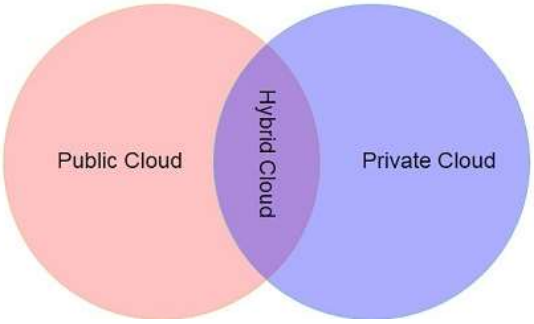

Fig.2 – Public, Private & Hybrid Cloud

VI-RELATED APPROCHES

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

Computing without servers**[3]** provide a novel basic for serverless function sequencing and introduce the challenge of serverless function composition. Following that, **Rabbit 2017** developed serverless state machines (Conductor) and a dsl (Composer) to make state machines easier to create. Trapeze **[16]** introduces dynamic ifc for serverless computing and sandboxes serverless functions to help them communicate with shared storage. Their formalization of Coq Article 149 in Proc. ACM Program. Lang., Vol. 3, No. OOPSLA. Date of publication: October 2019.

Several initiatives **[4]** use serverless computing to achieve elastic parallelization. Computing to achieve elastic parallelization.

Cloud orchestration frameworks**[5]** is a language for managing cloud environments; **Engage [Fischer et al. 2012]** is a deployment manager that supports inter machine dependencies;**[6]** is an embedded dsl for writing cloud-configurable programs; and CPL **[7]** is a unified language for writing distributed cloud programs and their distribution routines. In contrast, is a serverless computing semantics

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.

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.

VII- PROPOSED METHDOLOGY

Our proposed method can be 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.

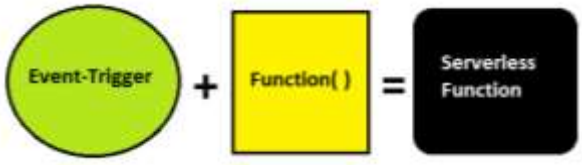

Fig.3- Formation of Serverless Function

In the fig 3, we have shown the internal structure of the serverless function in serverless container. A serverless container is basically the environment where any serverless function performs its execution. The formation of any serverless function comprises of two segments i.e Event triggering segment and user defined functions.

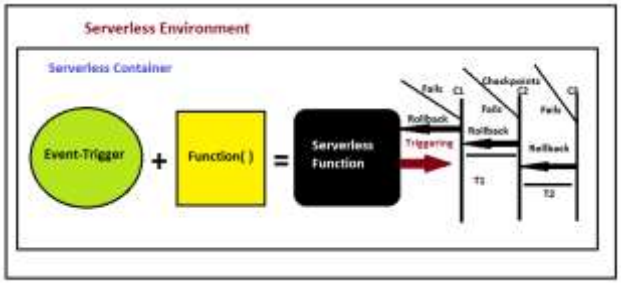

Fig. 4- Implementation of Checkpoint Methodology

https://doi.org/10.46335/IJIES.2022.7.8.8 **e-ISSN: 2456-3463**
Vol. 7, No. 8, 2022, PP. 42-47

International Journal of Innovations in Engineering and Science, www.ijies.net

In fig 4, we have shown the working of our proposed methodology to provide the better environment to the user to assess the current execution status of its code using checkpoint methodology.
In each execution of user's code, we will create checkpoints.Everytime when code will complete its execution on checkpoint, its completion log will be stored at that Checkpoint (In Fig. 4, C1,C2,C3 are those Checkpoints), so that if any time serverless function fails during its complete execution, it would not be required to that function to start from initial stage, It has to be just rollback to its previous checkpoint and can resume its execution form there only.
This will save the time and efforts of user to analysis its code better.

VIII- SIGNIFICANCE OF RESEARCH

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 more effective 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.

IX- CONCLUSION

The following is the conclusion of our Research paper:

- Provide a better mechanism to address the serverless container when it reaches the Ideal state and provide better methods to persist all of its execution states • Provide a better mechanism to notify developers about each state of serverless function execution under serverless container
- Provide a better mechanism to address the serverless container when it reaches the Ideal state and provide better methods to persist all of its execution states
- To develop a more effective system for detecting serverless function occurrences, failures, and retries.
- To provide a more secure serverless environment by developing better security solutions for serverless services.

X-FUTURE SCOPE

Methodology that works for Because of the special properties of serverless computing, many developers choose serverless platforms to perform their program functions. Triggering of serverless functions is particularly crucial in the serverless environment. Following the implementation of the recommended techniques, we will be able to identify the most suited ways for serverless functions and serverless environments.

We can create an environment where serverless function triggering is more easily compatible with these platforms, and we can address the challenges associated with serverless function execution in a serverless platform, because the demand for and acceptance of Public Cloud platforms is rapidly increasing. The serverless service may also generate notifications for each state of the function execution.

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 Service‖, InformationWeek 16 Oct. 2, 2009*
[3] *Ioana Baldini, Perry Cheng, Stephen J. Fink, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Philippe Suter, and Olivier Tardieu. 2017. The Serverless Trilemma: Function Composition for Serverless Computing. In ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!).*
[4] *KalevAlpernas, Cormac Flanagan, SadjadFouladi, Leonid Ryzhyk, MoolySagiv, Thomas Schmitz, and Keith Winstein. 2018. Secure Serverless Computing Using Dynamic Information Flow Control. Proceedings of the ACM on Programming Languages 2, OOPSLA (Oct. 2018).*
[5] *SanjivaWeerawarana, Chathura Ekanayake, Srinath Perera, and Frank Leymann. 2018. Bringing Middleware to Everyday Programmers with Ballerina. In Business Process Management*
[6] *Pulumi 2018. Pulumi. Cloud Native Infrastructure as Code. https://www.pulumi.com/. Accessed Oct 12 2019.*
[7] *Oliver Bračevac, Sebastian Erdweg, Guido Salvaneschi, and Mira Mezini. 2016. CPL: A Core Language for Cloud*

Computing. In Proceedings of the 15th International Conference on Modularity.

[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] *Srivastava, Shashank and Bahadur, Promila, Metaphorical Analysis of Serverless Computing Platforms and Related Challenges (May 10, 2021). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021,*

[16] *KalevAlpernas, Cormac Flanagan, SadjadFouladi, Leonid Ryzhyk, MoolySagiv, Thomas Schmitz, and Keith Winstein. 2018. Secure Serverless Computing Using Dynamic Information Flow Control. Proceedings of the ACM on Programming Languages 2, OOPSLA (Oct. 2018)*