TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 1, February 2020, pp. 530~537
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i1.12169

A review on serverless architectures - function as a service (FaaS) in cloud computing

Arokia Paul Rajan R
Department of Computer Science, CHRIST (Deemed to be University), Bengaluru

Article Info

Article history:

Received Dec 30, 2018
Revised Nov 5, 2019
Accepted Nov 30, 2019

Keywords:

AWS lambda
Cloud computing
FaaS
Google cloud function
Microsoft Azure function
Serverless computing

ABSTRACT

Emergence of cloud computing as the inevitable IT computing paradigm, the perception of the compute reference model and building of services has evolved into new dimensions. Serverless computing is an execution model in which the cloud service provider dynamically manages the allocation of compute resources of the server. The consumer is billed for the actual volume of resources consumed by them, instead paying for the pre-purchased units of compute capacity. This model evolved as a way to achieve optimum cost, minimum configuration overheads, and increases the application's ability to scale in the cloud. The prospective of the serverless compute model is well conceived by the major cloud service providers and reflected in the adoption of serverless computing paradigm. This review paper presents a comprehensive study on serverless computing architecture and also extends an experimentation of the working principle of serverless computing reference model adapted by AWS Lambda. The various research avenues in serverless computing are identified and presented.

Corresponding Author:

Arokia Paul Rajan R,
Department of Computer Science,
CHRIST (Deemed to be University),
Bengaluru, India.
Email: arokia.rajan@christuniversity.in

1. INTRODUCTION

Cloud computing is the on-demand consumption of compute power, storage, database, applications, and any IT resources through the Internet following pay-as-you-go pricing model [1]. The most basic way to define what the 'Cloud' is that it is a computer located somewhere else that is accessed via the Internet and utilized in some way. Web services is also another name for what people call the cloud. The cloud is comprised of server computers located in different locations around the world [2]. When we use a cloud service like Amazon web services (AWS) or Google Cloud architecture or Microsoft Azure, we are actually utilizing the computers belonging to these cloud service providers [CSPs].

The principle of cloud computing is remaining as it is, but, the need for shared working principle, enhancements in fast response of the services, agility of resource provisioning, and minimized management hurdles has been the targets of hyper scale CSPs [3]. Inspecting the level of capital investment and management involved to reach the above objectives, there are many researches proposed several reference models. Hyper scale data centers will grow from 338 in number at the end of 2016 to 628 by 2021 [4]. They will represent 53 percent of all installed data center servers by 2021. This kind of drastic increase in the computational requirements, there is a need for transforming the ‘traditional data centers’ into ‘hyper scale datacenters’ sophisticated with high levels of abstraction and virtualization. Recent technical advancement in

the virtualization technologies which are very promising to achieve hyper scale data centres easily.

Figure 1 represents the evolution stages of cloud computing. In the initial stages virtualization used as the mean for software and service consolidation by which attained the maximum utilization of the resources and easy management. During the initial phase, there was a common sharing of hardware. In the next phase, there was a pool of virtual machines (VMs) created on a server and each VM carry a copy of an operating system. Later, it advanced into the concept of containers where it included, OS level virtualization [5]. The containers are the platform sufficient enough to hold the resources needed for running a specific application. It achieved higher abstraction of resources comparing with VMs. In containerization, resource provisioning is much faster than VMs.

Apart from the efficiencies and faster rate of provisioning of resources through containerization, further enhancements are constrained with the basic infrastructural elements called servers. Serverless computing is a model of pooling and utilizing the resources which includes OS, runtime environments and hardware [6]. Figure 2 presents the evolution of serverless computing from containerization.

Serverless computing or function-as-a-service (FaaS) is defined as a software architecture where an application is decomposed into 'triggers' (events) and 'actions' (functions), and there is a platform that provides a seamless hosting and execution environment [7]. The application developer's concern only for light weighted and stateless functions that can be executed through an API based on the on-demand principle. The application consumes the resources to the point of execution and later the resources are released. The price model includes only the amount of time in which the resources were in use and the application developer need not to pay for resources until they are executed, thus it is referred to as 'serverless'.

In serverless computing, the responsibilities of the cloud service provider include the management of the data centre, server and the runtime environment. A contrast to the other cloud models, the more responsibility is vested on the shoulder of cloud service provider and the developer is relieved with the management and maintenance complications any further [8].

The rest of the paper is structured as follows: Section II presents the detailed study of related works on the conceptualization of serverless computing. Section III presents a few enterprise use cases fitting to the serverless computing. Section IV shows the comparison of features by the top FaaS service providers. Section V presents a demonstration to understand the working principle of serverless computing using AWS Lambda. Section VI presents a few technical difficulties and research gaps in FaaS. Section VII concludes by reinstating the significance of the serverless computing paradigm.

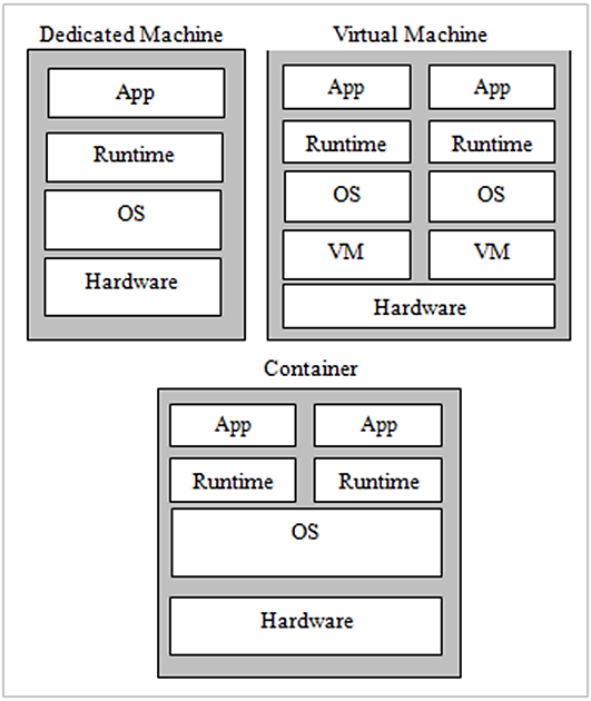

Figure 1. Evolution of sharing resources

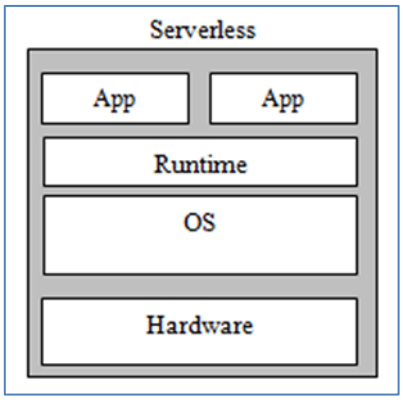

Figure 2. Sharing resources in serverless computing

2. RELATED WORKS

Born from a need to make platform as a service (PaaS) more accessible, fine-grained, and affordable, serverless computing has garnered interest from both industry and academia. The work [9] aims to give an understanding of these early days of serverless computing: what it is, where it comes from, what is the current status of serverless technology, and what are its main obstacles and opportunities. T. Lynn, et al., [10] provides a review and multi-level feature analysis of seven enterprise serverless computing platforms. It reviews extant research on these platforms and identifies the emergence of AWS Lambda as an actual base platform for research on enterprise serverless cloud computing.

A novel design of performance-oriented serverless computing platform deployed in Microsoft Azure, and utilizing Windows containers as function execution environments [11]. There are metrics proposed to evaluate the execution performance of serverless platforms and conduct tests with the proposed prototype. The measurements showed significant improvement in achieving greater throughput than other platforms at most concurrency levels. The other platform alternatives [12] to AWS Lambda, no discrete academic researches using Azure Functions, Google Cloud Functions, IBM Bluemix OpenWhisk, Iron.io Ironworker, Webtask, Galactic Fog Gestal Laser were identified.

J. Short, et al., [13] presented three demonstrators for IBM Bluemix OpenWhisk. They demonstrate event-based programming triggered by weather forecast data, Apple WatchOS2 application data, and speech utterances. It also demonstrated a chatbot using IBM Bluemix OpenWhisk that calls on IBM Watson services including news, jokes, dates, weather, music tutor and an alarm service. [14] conducted a survey on the existing serverless platforms from industry, academia, and open source projects, key characteristics and use cases, and describe technical challenges and open problems. This work presented a hands-on experience of using the serverless technologies available from different cloud providers such as IBM, Amazon, Google and Microsoft.

Z. Al-Ali [15] designed ServerlessOS, comprised of three key components: (a) a new desegregation model, which leverages desegregation for abstraction, but enables resources to move fluidly between servers for performance, (b) a cloud orchestration layer which manages fine-grained resource allocation and placement throughout the application's lifetime via local and global decision making, and (c) an isolation capability that enforces data and resource isolation. [16] proposed an efficient resource management system for serverless cloud computing frameworks with the goal to enhance resource with a focus on memory allocation among containers. The design added a layer top of an open-source serverless platform, OpenLambda. It is based upon application workloads, and serverless function’s memory needs events are triggered. The memory limits also lead to variations in the number of containers spawned on OpenLambda.

3. ENTERPRISE USECASES FOR SERVERLESS COMPUTING

Proper investigation of the nature and need of recent use cases of the enterprises, serverless computing becomes inevitable move to fulfill the requirements. The scenarios include event processing with big data, API orchestration among the vendors, consolidation of APIs to minimize API calls, process monitoring, and execution control for tracking the issues. The following are some of the best fit use cases for serverless computing [17]:

– Use case 1: event-triggered computing

In multimedia processing business applications, huge volumes of files are frequently uploaded to Object Storage Services [OSS] for processing. The requirements may be such as transcoding, watermarking, fetching the data. This business scenario involves a variety of devices like desktop computers or PDAs or mobile phones accessing different file types of uploading multimedia content such as images, videos, and text files. Event-triggered computing will be a solution for addressing many technical difficulties by event-triggered computing.

- Use case 2: live video broadcasting
  In live video broadcasting scenarios, the broadcasting synthesizing node receives audio and video streams from the hosts. The collected data can be synthesized based on the function computing. Finally, the synthesized video stream needs to be pushed to Content Delivery Network [CDN].
- Use case 3: IoT data processing
  IoT framework needs an efficient function computing design that can receive status data from a variety of connected smart devices. Also, it needs an efficient event-based computing architecture to transmit the processed data to other devices or storing into the database [18].
- Use case 4: shared delivery system
  A global group of restaurants or a product-based company may need an event-based notification system to the nearest delivery personnel to pick up from the nearest seller for the product delivery. Though event-based computing is applicable in many such use cases, but it is not a one-size-fits-all solution. If the requests are not having significant fluctuations in the use case 4, then function computing may be a wrong choice of solution design [19].

4. COMPARISON OF TOP SERVERLESS COMPUTING CLOUD SERVICE PROVIDERS

In a short period, the serverless technology gained a lot of momentum in the industry. Table 1 presents the comparison of various features provided by the pioneers of FaaS providers [7, 20-23].

Table 1. Comparison of features

	AWS Lambda	Google Cloud Function	Microsoft Azure Function
Introduction	2015	2016	2016
Scalability	Automatic	Automatic	Automatic
Max functions	Unlimited	20 per project	Depends on the trigger & available resources
Supported languages	Javascript, Java, Python, NodeJS	Javascript	C#, F#, NodeJS, Python, PHP, Bash
Concurrent execution	100 parallel executions per account	Unlimited	Based on App service
Deployment	ZIP uploads	ZIP uploads, Cloud storage	Git integrated, REST API
Memory allocation	Per function	Not specific	Per App service
Licensing	Closed source	Open source	Open source
Pricing model	Pay as code executes	Pay as code executes	Pay as code executes
Event driven architecture	S3, SNS, Dynamo DB, Kinesis, Cloud Watch	Cloud Pub, Cloud storage objects	Azure and third-party services

5. DEMONSTRATION OF SERVERLESS COMPUTING USING AWS LAMDA

The objective of the experiment carried out in this paper is to demonstrate the method of configuring AWS Lambda for responding the notifications from the Auto Scaling Group. Table 2 presents the glossary of AWS services used to accomplish the objective of the experiment [24]. Figure 3 presents the demonstration scenario which aims to create an event-driven computing function.

Table 2. AWS services

S.No.	Service	Purpose
1	Auto Scaling Group	Logical grouping of EC2 instances with same features used for the scale management.
2	Simple Notification Service [SNS]	Service used for delivery of messages in bulk, especially for the mobile users.
3	IAM	Identity and Access Management is a service which provides a secure way of assessing AWS resources.
4	AWS Lambda	Event-driven, serverless computing platform service that runs code in response to events and automatically manages the computing resources required by that code [20].
5	Cloud Watch	It is a service used to receive and monitor log files, set alarms, and automatically respond to changes in AWS resources [20].

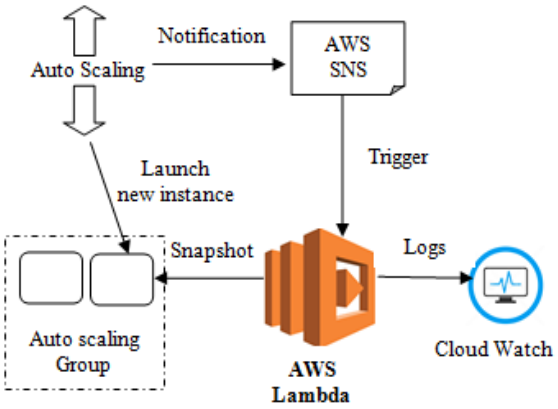

Figure 3. Scenario of the demonstration

5.1. Demonstration Setup

Some of the AWS services can automatically generate notifications when an event occurs. Such notifications can be used as a trigger to automate actions without requiring human intervention. In is section, an illustration has been presented to understand how a serverless architecture is working [25]. The scenario is to use AWS Lambda function that will automatically snapshot and attach a new AWS EC2 instance launched by the auto scaling group. In this illustration, an auto scaling group has been already configured. Table 2 shows the variety of AWS services incorporated to accomplish the objective of the demonstration. The following steps are carried out in order to achieve the objective of this illustration:

- Login into AWS account and open the console
- Create an SNS topic
  a. Click 'Get started'
  b. Create 'Topic' with the topic name.
  c. Click 'create topic'
- Configure Auto Scaling to send Event
  a. On the 'Services', click EC2
  b. Select 'Auto Scaling Groups'
  c. Click 'Notifications' tab
  d. Click 'Create notification'
  e. Confirm 'ScaleEvent' is selected in 'Send a notification to'
  f. In 'Whenever Instances' select 'launch'.
  g. Save.
- Create an IAM role for the Lambda function
  a. In 'Services', click 'IAM'
  b. Click 'Roles'.
  c. Click 'Create Role'
  d. Select 'AWS Service'
  e. Select use case as 'Lambda'.
  f. Give the permission as 'AmazonEC2FullAccess'.
  g. Give a Role Name.
  h. Click 'Create role'.
- Create a Lambda Function
  a. Select 'Lambda' from 'Services'.
  b. Create a 'Function'
  c. Configure the Name, Runtime and Role.
  d. Add the Function code using Python.
  e. Add 'Triggers' on 'Scale Event'
- Auto Scaling Group Scaling out to trigger the Lambda function.
  a. Select 'EC2' in 'Services'.
  b. Edit Scaling Group with desired instances as 2.
  c. Create the 'Snapshots' which is created by the Lambda function.

Figure 4 shows the Python code which creates trigger on the event of scaling out. Figure 5 shows the creation of a trigger (event) for the Lambda function.

```
# Snap_and_Tag Lambda function
#
# This function is triggered when Auto Scaling launches a new instance.
# A snapshot of EBS volumes will be created and a tag will be added.

from __future__ import print_function

import json, boto3

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    # Extract the EC2 instance ID from the Auto Scaling event
notification
    message = event['Records'][0]['Sns']['Message']
    autoscalingInfo = json.loads(message)
    ec2InstanceId = autoscalingInfo['EC2InstanceId']

    # Snapshot all EBS volumes attached to the instance
    ec2 = boto3.resource('ec2')
    for v in ec2.volumes.filter(Filters=[{'Name': 'attachment.instance-
id', 'Values': [ec2InstanceId]}]):
        description = 'Autosnap-%s-%s' % ( ec2InstanceId, v.volume_id )

        if v.create_snapshot(Description = description):
            print("\t\tSnapshot created with description [%s]" %
description)

    # Add a tag to the EC2 instance: Key = Snapshots, Value = Created
    ec2 = boto3.client('ec2')
    response = ec2.create_tags(
        Resources=[ec2InstanceId],
        Tags=[{'Key': 'Snapshots', 'Value': 'Created'}]
    )
    print ("***Tag added to EC2 instance with id: " + ec2InstanceId)

    # Finished!
    return ec2InstanceId
```

Figure 4. Creating AWS Lambda using Python in the code window

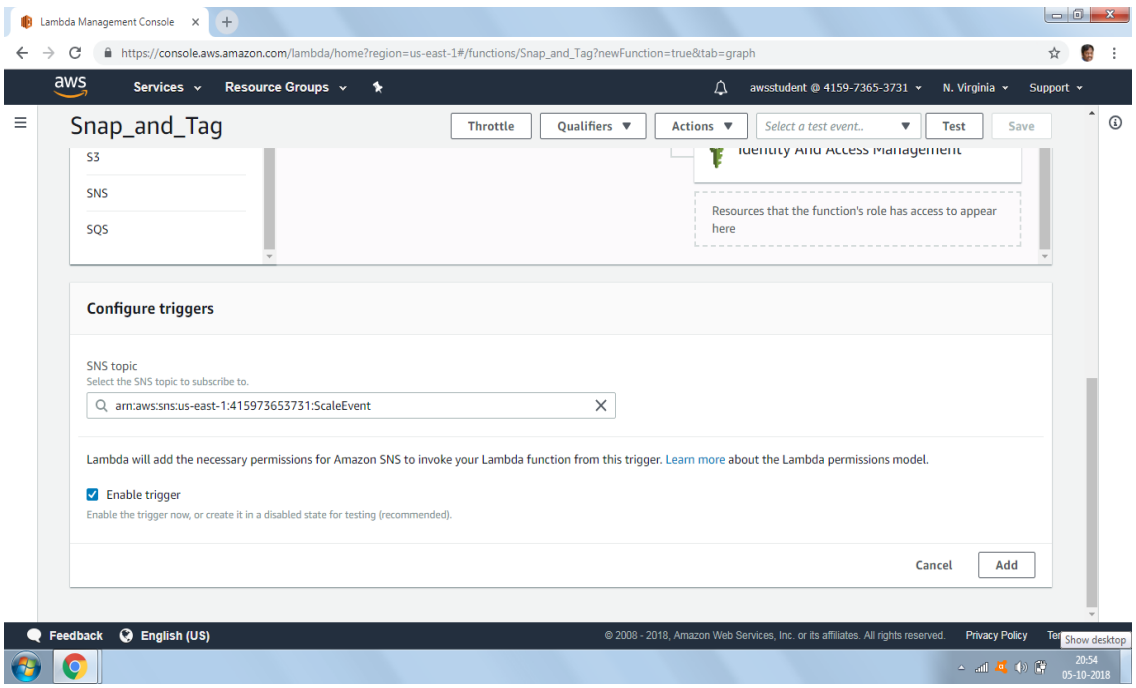

Figure 5. Creating trigger for the Lambda

5.2. Results

On successful completion of the steps 1 to 6 will create two snapshots that were created by Lambda function. It is the representation of successful execution of Lambda function of the auto scaling group. If the snapshots were not created, then the Lambda function either had a failure or was not triggered.

6. RESEARCH AVENUES IN SERVERLESS COMPUTING

Extending the commercial use cases that are presented in Section II would be a better way to explore new challenges put forth by this new computing architecture. Careful investigation of the literature in the form of researches and white papers suggests that there are a variety of unattended challenges hidden despite of its promising predictions. The following are the major categories of challenges identified by [26, 27]:

- Hardware-level challenges: virtualization of servers, distributed storage and their levels, interoperability of supporting heterogeneous hardware of vendors, cold starts, optimization of resources, and designing fault tolerant system architectures.
- Developer-level challenges: lack of tracking and debugging tools, declarative deployment, the expertization of a programming caliber to cater refactoring of existing systems, ability to integrate and compose, managing and maintaining, designing stateful and stateless functions, transaction and concurrency management, optimized code granularity, recovery system design, and adapting DevOps principles.
- Management-level challenges: fixing the resource limits, resource provisioning and load balancing, dynamic scheduling, launch overheads, legacy system migration, predictable scalability, and security mechanisms.
- Business-level challenges: cost estimation, pricing model, managing hybrid cloud, and non-cloud systems utilizing the serverless architectures.

7. CONCLUSION

Serverless computing is in the stage of conceptualization by the researchers and experimentation by the industry. It is predicted that the evolution of this new computing paradigm in the Cloud will definitely lead to a simpler, cheaper and more efficient resource management. It is to be acknowledged that these promises are based on however such propositions are based on a specific use case with the small scale of deployment by the industry. The review presented in section 2 and section 3 provides a futuristic dimension for serverless architectures as a new era of computation which can be adapted for a broader use case. Section 2 also implies the possibilities of exploring more research avenues for the academic and research community in the arena of serverless computing.

The principle of serverless computing when deployed in non-cloud systems lead to a new computing technology known 'deviceless edge computing'. Serverless and deviceless computing are the new buzz words in the industry, which paves a way to new research opportunities in the Cloud as well as non-cloud systems. Since serverless computing is in its infant stage, there are a variety of technical difficulties and challenges that are unaddressed, such as smooth scaling with a tolerance of network hassles and secured resource provisioning. Also, section 6 identified a few research directions.

REFERENCES

[1] B. Sosinsky, "Cloud Computing Bible," 1st ed, Wiley Publishing, 2011.
[2] R A. P. Rajan, "Service Request Scheduling based on Quantification Principle using Conjoint Analysis and Z-score in Cloud," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 1238-1246, 2018.
[3] L. A. Barroso, et al., "The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines," *Synthesis Lectures on Computer Architecture*, vol. 8, pp. 1-154, 2013.
[4] Cisco, "Cisco global cloud index: Forecast and methodology, 2015- 2020," CA: Cisco Public, 2018.
[5] Matt Soucoup, "Introduction to Serverless Computing," Telerik, [Online], Available: https://www.telerik.com/blogs/introduction-to-serverless-computing, 2018.
[6] Abrams, H., "The Evolution of Serverless Computing," [Online], Available: https://www.ca.com/us/modern-software-factory/content/the-evolution-of-serverless-computing.html, 2017.
[7] Amazon, "Building Applications with Serverless Architectures," [Online], Available: https://aws.amazon.com/lambda/serverless-architectures-learn-more/.
[8] Ivan Dwyer, "Serverless Computing Developer Empowerment Reaches New Heights," Iron, [Online], Available: https://www.iron.io/docs/Whitepaper_Serverless_Final_V2.pdf.

[9] E. van Eyk, et al., "Serverless is More: From PaaS to Present Cloud Computing," *IEEE Internet Computing*, vol. 22, pp. 8-17, 2018.
[10] T. Lynn, et al., "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms," *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2017, pp. 162-169.
[11] G. McGrath, P. R. Brenne, "Serverless Computing: Design, Implementation, and Performance," *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops*, pp. 405-410, 2017.
[12] P. Castro, et al., "Serverless Programming (Function as a Service)," *2017 IEEE 37th International Conference on Distributed Computing Systems*, pp. 2658-2659, 2017.
[13] J. Short, et al., "Cloud Event Programming Paradigms: Applications and Analysis," in *Proceedings of the 9th IEEE International Conference on Cloud Computing (CLOUD)*, pp. 400-406, 2017.
[14] I. Baldini, et al., "Serverless Computing: Current Trends and Open Problems," *Research Advances in Cloud Computing, Springer*, pp. 1-20, 2017.
[15] Z. Al-Ali, "Making Serverless Computing More Serverless," *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 456-459, 2018.
[16] A. Saha and S. Jindal, "EMARS: Efficient Management and Allocation of Resources in Serverless," *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 827-830, 2018.
[17] Eric Jonas, et al., "Cloud Programming Simplified: A Berkeley View on Serverless Computing," *Technical Report No. UCB/EECS-2019-3*, University of California at Berkeley, 2019.
[18] Adhitya Bhawiyuga, et al., "Architectural design of IoT-cloud computing integration platform," *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 17, no. 3, pp. 1399-1408, 2017.
[19] Fan and L. Liu, "A Survey of Challenging Issues and Approaches in Mobile Cloud Computing," *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Guangzhou, pp. 87-90, 2016.
[20] K. Kritikos and P. Skrzypek, "Simulation-as-a-Service with Serverless Computing," *2019 IEEE World Congress on Services*, Italy, pp. 200-205, 2019.
[21] Leona Zhang, "4 Use Cases of Serverless Architecture," [Online], Available: https://dzone.com/articles/4-use-cases-of-serverless-architecture, 2018.
[22] Microsoft, "Azure Functions Documentation," [Online], Available: https://docs.microsoft.com/en-us/azure/azure-functions/.
[23] Google, "Google Cloud Functions Documentation," [Online], Available: https://cloud.google.com/functions/.
[24] Amazon Web Services, "AWS Lambda Developer Guide," [Online], Available: http://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf.
[25] S. Hendrickson, et al., "Serverless computation with openlambda," Hotcloud'16, *2016 USENIX Annual Technical Conference*, 2016.
[26] Data Center Frontier, "Data Center Developers: Meeting the Challenges of Today's Requirements," [Online], Available: https://datacenterfrontier.com/data-center-developers-meeting-challenges/
[27] D. Gannon, R. Barga and N. Sundaresan, "Cloud-native applications," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 16–21, 2017.