Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, September 2022

Overview of Serverless Architecture

Amit Kumar Jain
Computer Science
Delhi University
Delhi, India

***Abstract*—As cloud computing evolves, serverless computing, also known as function as a service (FaaS), is viewed as the next stage of cloud computing evolution. As a logical extension of cloud computing, serverless computing is a disruptive method to application development. It is based on the developer's code for precise resource allocation, and the platform's resources are activated when a predetermined event occurs. By contrasting serverless architecture with conventional design, its benefits are demonstrated. Although serverless is a relatively new notion in software architecture, it is a highly influential technological innovation**

Keywords—Cloud computing, Serverless Architecture

I. INTRODUCTION

The ISO/IEC JTC1 and ITU-T joint working group ISO/IEC17788 defines cloud computing as "Information Computing–Cloud Computing–Overview and vocabulary" DIS version: cloud computing is a A scalable, resilient, shared pool of physical and virtual resources is provisioned and managed on an on-demand, self-service basis and provides a model for network access [1]. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service are typical cloud service models (SaaS). Depending on the circumstances, each cloud service model has unique qualities that provide consumers with specialized capabilities. IaaS primarily offers consumers with virtual computers or other resources as a service. PaaS primarily provides users with a development platform as a service. SaaS generally provides users with apps as a service. As cloud computing evolves, serverless computing, also known as function-as-a-service (FaaS), is viewed as the next stage in cloud computing evolution [2]. As a logical extension of cloud computing, serverless computing is a disruptive method to application development.

II. SERVERLESS ARCHITECTURE

A. What it means

Many believe Serverless is a combination of Back-end-as-a-Service (BaaS) and Function-as-a-Service (FaaS). The main form of serverless representation is FaaS, so serverless computing is considered a "function as a service (FaaS)" or "function-driven event" [3]. It is based on the developer's code for precise resource allocation, and the platform's resources are activated when a predetermined event occurs.

B. The advantages of a serverless architecture

The most obvious benefit of no server is that there is no need to maintain the server, which means the application team and developers can concentrate on application development, and there is no need to be concerned with infrastructure services [7].

1. Decrease operating expenses

The infrastructure does not vanish in a serverless architecture, which is effectively an outsourced solution. It is only required to pay the appropriate amount of calculation based on the size and shape of the traffic, which can significantly reduce operational expenses, particularly for early and dynamic application load requirements with varying variations.

2. Extreme Scalability

The extremely high scalability of cloud services is not new, but the no-service design takes it to an entirely new level. Use no server, eliminate the need to explicitly add and remove server instances, and allow vendors to modify the application. Since the cloud computing provider conducts extensions on a per-request basis, there is no need to calculate how many requests may be processed concurrently before memory is exhausted.

3. Separation difficulty

The serverless architecture separates the application's components so that each component solves a unique problem.

4. Isolation procedure

Each Lambda function in a serverless environment is totally isolated. If a feature is disabled, it has no effect on other features and does not cause the server to crash.

C. Hierarchical positioning of serverless architecture

FaaS and PaaS are comparable in several ways. Even people believe that FaaS is a subtype of PaaS. Vice president of engineering at Intent Media, Mike Roberts, disagrees with this statement. Roberts noted that with FaaS, the full application may be started and stopped for each request, whereas PaaS cannot. In terms of operation and maintenance, the capacity for scaling is the primary distinction between the two. Adrian Kokrov proposes a basic definition: If your PaaS can launch an instance running for half a second in 20 milliseconds, it is referred to as serverless.

The following table compares IaaS, PaaS, FaaS, and SaaS. FaaS is advantageous because to its high development efficiency, great scalability, excellent operation and maintenance, and low cost. From IaaS to PaaS to FaaS to SaaS, the control over service implementation is lowered and the emphasis is placed on business logic. In other words, the degree of abstraction increases while the degree of flexibility decreases. And FaaS is situated between PaaS and SaaS, making it incredibly versatile and user-friendly for developers. FaaS provides all resources except the application layer, developers just need to focus on the code logic, while SaaS has little flexibility and is suited for ordinary consumers, making it challenging to fulfill the unique requirements of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, September 2022

organizations. So serverless architecture will be the future trend in software development.

TABLE I. IAAS, PAAS, FAAS AND SAAS COMPARISON TAB HIGH

	IaaS	PaaS	FaaS	SaaS
Development efficiency	Low	Middle	High	High
Scalability	Low	Middle	High	High
Cost	High	High	Low	High
Operational maintenance	Low	High	High	High

III. COMPARISON OF SERVERLESS ARCHITECTURE AND TRADITIONAL ARCHITECTURE

A. Serverless architecture vs. Traditional monolithic application architecture

In conventional online applications, servers are a vital component. Despite the presence of load balancers or dedicated web servers in front of the server on occasion, the application server is largely complete. It provides all the necessary application functions, such as storing user data, providing security authentication, controlling processes, etc. The majority of the application's pages just provide an interface for the backend, albeit with some navigation control functions.

This is the conventional method that many individuals refer to as a multi-tier architecture. The system consists of browsers, application servers, and numerous post-systems. Conclusion of service component All of these levels can be eliminated for a more straightforward solution using a serverless architecture [4]. Instead of using the web client as the application server's interface, it is preferable to create a single-page web application that implements the application logic in the browser. This means that a simple static web server is sufficient. All interactions consist solely of the conveyance of application data. The browser functions similarly to an application container. Thus, the final design will eliminate all intermediate layers from the conventional web application architecture, enabling the browser to connect directly to the required services

B. Serverless architecture vs. microservice architecture

Although Serverless is not as popular as microservices, it surpasses microservices in terms of usability, dependability, and future potential. There are several parallels between serverless and microservices, such as business segmentation, statelessness, and agile characteristics.

In many ways, serverless is more compact and demanding than microservices. Microservices divide services by service, whereas Serverless divides services by function. Microservices can exchange memory state between calls, whereas Serverless calls must be stateless. In addition, Serverless depends on BaaS to offer third-party requirements, whereas microservices are free to choose their own third-party dependencies, such as locally built traditional middleware stacks (such as local MySQL and message buses) [6].

Microservices are intended to divorce complex monolithic applications into numerous separate services. Microservices are designed to simplify the process of developing complicated applications [5]. However, microservices and Serverless are compatible; both encourage system decoupling. It is not novel to separate the implementation of business logic into Functions. This technique is not novel: Microservice is a more prevalent paradigm, and firms like Netflix and Uber have already achieved success with it. As a Function, users can implement a microservice. A key objective of the computational design of Alibaba Cloud is to make it the ideal platform for developing microservices applications. Microservice architecture is quite difficult. Managing the interdependence between microservices necessitates a highly automated release distribution strategy following the division of a system into hundreds of services. Microservices is a programming approach, whereas Serverless is a computing platform.

TABLE II. SERVERLESS ARCHITECTURE AND MICROSERVICE ARCHITECTURE COMPARISON TABLE.

	Microservice architecture	Serverless architecture
Business Split	Service bound	Function bound
Server Request	Memory state sharing across calls	Completely stateless
Third Party dependency	Free choice of third-party dependence	Relying on third-party dependencies provided by BaaS
Level Positioning	Development model	Computing platform

C. Serverless architecture vs. container

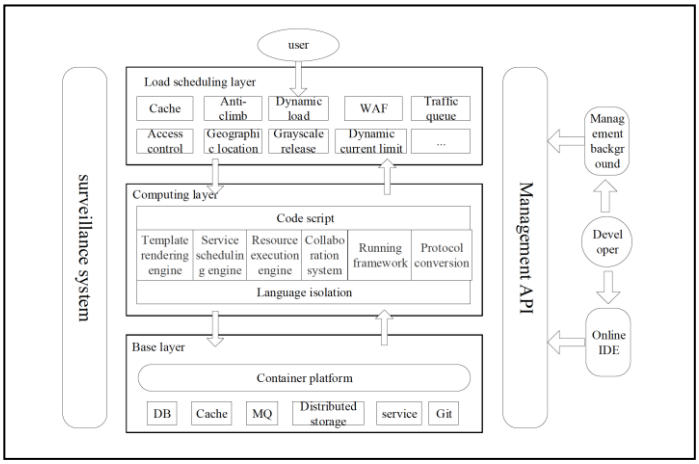
Serverless workloads are perfect in terms of predictability, resource requirements, and transaction duration. Containers have benefits for long-running processes and workloads that are predictable. Containers offer allow greater architectural flexibility for applications but require more infrastructure administration. Containers and serverless computing are not on the same plane. Serverless is a software design architecture, and the software architecture is carried by the container. Although there is no publicly available information, we can assume that a serverless framework such as AWS Lambda utilizes container technology, and that it is challenging to perform language-independent and millisecond-level launch.

Although several open-source projects already use Docker to implement the FaaS component of Serverless, we do not believe that the public Serverless framework, such as AWS Lambda, employs Docker directly. It must be a more compact and lightweight container technology. It may be referred to as the Nano-Container [8]. The container and serverless architecture have a complementary rather than overlapping relationship.

IV. SERVERLESS ARCHITECTURE DESIGN

Three tiers comprise the Serverless architecture: the scheduling layer, the computation layer, and the base layer. Figure 1 depicts the architecture design of the Serverless architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, September 2022



The scheduling layer is the initial layer. It appears simple, however there are numerous functions to implement. For instance, how to mount and expand the following enhanced services in a single second. Therefore, dynamic load balancing of hot loads is crucial and may be observed at every stage. In addition to the exterior traffic of the front end, internal traffic must also enter.

The second layer is the computer layer, which focuses mostly on how to swiftly open resources for job scheduling. There are two primary approaches. The first option is to directly utilize a Docker container. Each expansion is a container, which is stacked one container on top of the next. However, this strategy wastes considerable resources. The advantage of using dynamic language is that the runtime is segregated from each other, like that of a standalone container.

The infrastructure layer provides users with the required infrastructure, such as virtual computers and storage resources, to schedule and manage physical resources more effectively.

The serverless architecture is not yet ideal, and there are still issues. Serverless is in its infancy, is not yet ideal, and there is much work to be done in the future. At this moment, Serverless's next objective consists mostly of four points:

1) *Add additional languages*
2) *The capabilities of Web-based IDEs have improved.*
3) *More competencies are configured*
4) *Implement an automated testing mechanism.*

V. CONCLUSION

As the company's system and services continue to expand, an increasing number of businesses are transitioning from a traditional architecture to a microservices architecture and then to a serverless design. Serverless is a relatively new concept in software architecture, however it is a highly influential technological innovation. Companies who adopt a serverless architecture and a culture that embraces the technology will lead us into the future. The serverless architecture is in its infancy, not yet ideal, and there is much work to be done in the future.

REFERENCES

[1] (2014) White Paper on Cloud Computing Standardization. China Electronics Technology Standardization Institute.
[2] Clint B. (2019) Will serverless computing be the future of cloud computing. Computer world, 2019-03-25(005).
[3] Brandon B. (2017) Serverless Computing: Next Generation Cloud Infrastructure. Computer world, 2017-05-15(003).
[4] Carter G. (2018) Serverless Architecture. Mechanical Industry Press. Beijing.
[5] Su J., Tian J.B. (2019) Microservice Architecture of Web Application in Cloud Environment. Environment. Electronic Technology and Software Engineering, (15):131-132.
[6] Xiang Xu. (2019) Serverless serverless architecture is the manifestation of microservice architecture. https://msd.misuland.com/pd/3255818238113618692.
[7] Zhi Yun. (2018) Read the advantages and disadvantages of serverless architecture, and use cases. http://www.sohu.com/a/234002113_100159565.
[8] Lingming Xia. (2018) Deep understanding of serverless architecture. https://blog.csdn.net/xialingming/article/details/81369624.