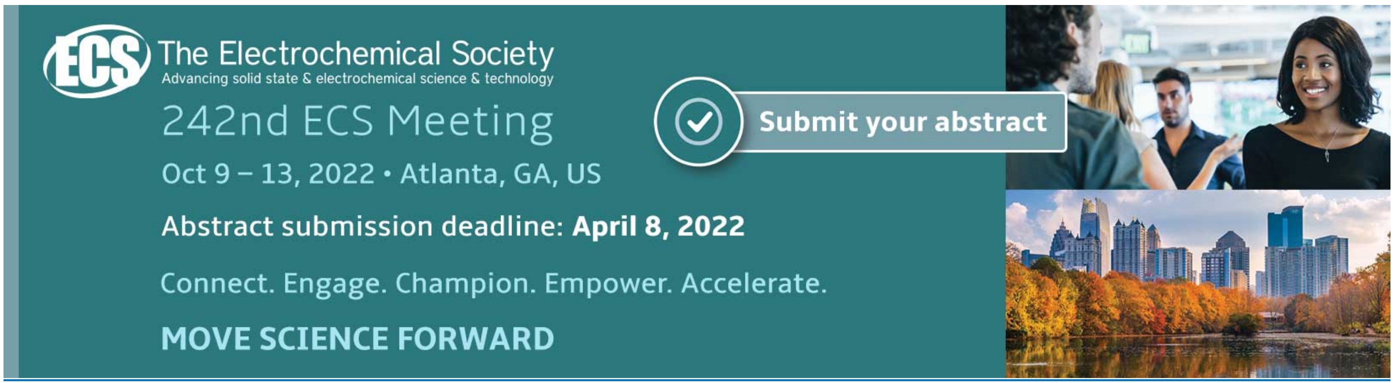
Overview Of Serverless Architecture Research


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
242nd ECS Meeting
Oct 9 – 13, 2022 • Atlanta, GA, US
Submit your abstract
Abstract submission deadline: April 8, 2022
Connect. Engage. Champion. Empower. Accelerate.
MOVE SCIENCE FORWARD

Overview Of Serverless Architecture Research

Lizheng Jiang[1], Yunman Pei[2] and Jiantao Zhao[3]

[1]School of Control and Computer Engineering, North China Electric Power University, Beijing, Beijing, 102206, China

[2]School of Control and Computer Engineering, North China Electric Power University, Beijing, Beijing, 102206, China

[3]School of Control and Computer Engineering, North China Electric Power University, Beijing, Beijing, 102206, China

[*]Corresponding author's e-mail: peiyunman@eplian.com

Abstract. With the development of cloud computing, serverless computing, that is, function as a service (FaaS) is considered to be the next stage of cloud computing development. Serverless computing can be seen as a logical extension of cloud computing, a disruptive approach to application development. It is based on the code written by the developer for accurate resource allocation, and the resources on the platform are started when a predefined event is triggered. By comparing with the traditional architecture, the advantages of serverless architecture are highlighted. Although serverless is a relatively new concept in the field of software architecture, it is also a technological innovation with great influence.

1. Introduction

Cloud computing is defined by the ISO/IEC JTC1 and ITU-T joint working group ISO/IEC17788 "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]. Typical cloud service models include: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each cloud service model has different characteristics that provide users with specific functionality, depending on the particular situation. IaaS mainly provides virtual machines or other resources as services to users. PaaS mainly provides a development platform as a service to users. SaaS primarily provides applications as a service to users. With the development of cloud computing, serverless computing, namely function-as-a-service (FaaS), is considered to be the next stage of cloud computing development [2]. Serverless computing can be seen as a logical extension of cloud computing, a disruptive approach to application development.

2. Serverless architecture

2.1. Concept

Mike Roberts believes that 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 code written by the developer for accurate resource allocation, and the resources on the platform are started when a predefined event is triggered.

2.2. The advantages of a serverless architecture

The most obvious advantage of no server is that there is no need to maintain the server, no server can focus on application development, no need to worry about infrastructure services [7].

(1)Reduce operating costs

A serviceless architecture is essentially an outsourced solution and the infrastructure does not disappear. It only needs to pay the required amount of calculation according to the size and form of the traffic, which may greatly save operating costs, especially for early and dynamic application load requirements with different changes.

(2)Unlimited scalability

Extremely high scalability is not new in cloud services, but the no-service architecture takes it to a whole new level. Use no server, no need to explicitly add and remove instances to the server array, and let vendors extend the application. Since the cloud computing provider performs extensions on a per-request basis, there is no need to consider the issue of how many concurrent requests can be processed before the memory is low.

(3)Separation problem

Serverless divides the application into different parts by separating them so that each part solves a single problem.

(4)Isolation process

In a serverless environment, each Lambda function is completely isolated. If one of the features is turned off, it does not affect other features, it does not cause the server to crash.

2.3. Hierarchical positioning of serverless architecture

FaaS and PaaS have many similarities in some respects. Even people think that FaaS is another form of PaaS. Mike Roberts, vice president of engineering at Intent Media, disagrees with this statement. Roberts pointed out that FaaS can start and stop the entire application for each request, but PaaS can't. And the biggest difference between the two in terms of operation and maintenance is the scaling capability. Adrian Kokrov gives a simple definition: If your PaaS can efficiently start an instance running for half a second in 20ms, it is called serverless.

Table 1 below shows the comparison of IaaS, PaaS, FaaS and SaaS. It can be seen that FaaS has the advantages of high development efficiency, strong scalability, good operation and maintenance, and low cost. From IaaS to PaaS to FaaS to SaaS, the control of service implementation is reduced, and the focus on business logic is increased. In other words, the level of abstraction is getting higher and higher, and the degree of flexibility is getting lower and lower. And FaaS is between PaaS and SaaS, which is highly flexible and convenient for developers. Figure 1 shows a comparison of the four cloud service model deployment levels. It can be seen that FaaS provides all resources except the application layer, developers only need to pay attention to the code logic, and SaaS has low flexibility and is suitable for ordinary users, which is difficult to meet the special needs of enterprises. So the serverless architecture will be the future development trend.

Table 1. IaaS, PaaS, FaaS and SaaS comparison table.

	IaaS	*PaaS*	FaaS	*SaaS*
Development efficiency	Low	Middle	High	High
scalability	Low	Middle	High	High
Operational maintenance	Low	High	High	High
Cost	High	High	Low	High
Applicable people	System administrator	Developer	Developer	Ordinary user

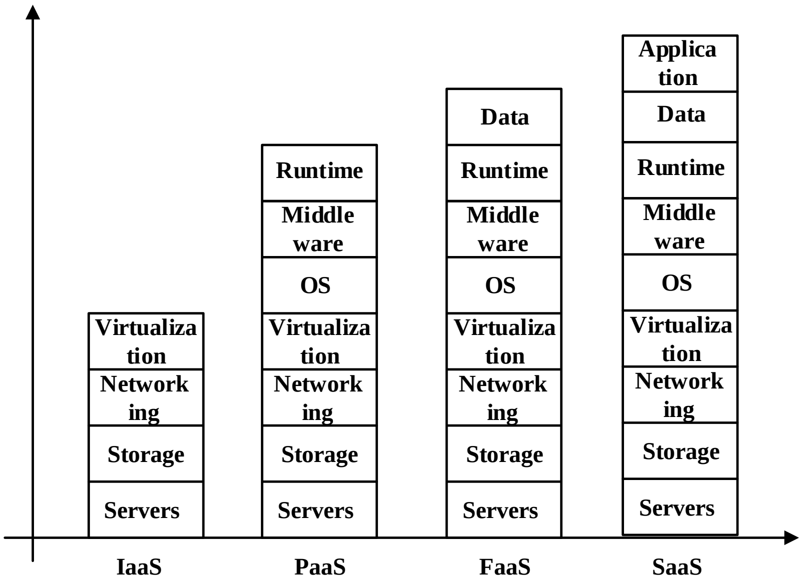


Figure 1 Cloud service model deployment level comparison chart

3. Comparison of serverless architecture and traditional architecture

3.1. Serverless architecture vs. SOA architecture

In traditional web applications, servers are an integral part of the system. Although sometimes there are load balancers or dedicated web servers in front of the server, the application server is mostly done. It performs all the necessary functions of an application, including storing user data, performing security authentication, controlling processes, and so on. Most of the application's pages only provide an interface for the backend, although it also involves some of the functions of controlling navigation. This is the traditional way that many people call a multi-tier architecture. The system is generally composed of browsers, application servers, and multiple post-systems. End service composition.

Using a serverless architecture, all of these hierarchies can be removed for a more straightforward implementation [4]. Instead of just using the web client as the interface of the application server, it is better to build a single-page web application to implement the application logic in the browser. This means that you only need a simple static web server. All interactions are just the transfer of application content. The browser is like an application container. In this way, the final design is to remove all intermediate layers in the traditional web application architecture, allowing the browser to connect directly to the services it needs. The comparison with the traditional application mode is shown in Table 2 below.

Table 2. Serverless architecture and SOA architecture comparison table.

	SOA architecture	*Serverless architecture*
Focus angle	Various parts	Single business logic layer
Management server	Enterprise related technical personnel	Cloud service provider
Expansion capability	Requires receiver queue and display capacity management to scale	Automatically expand according to actual situation
Usability and fault tolerance	Need to code, configure, and manage	Built-in usability and fault tolerance
Payment mode	Pay by purchase amount	Pay by request

3.2. Serverless architecture vs. microservice architecture

Although Serverless is not as hot as microservices, it does not lose microservers in terms of its practicability, reliability and future potential. Serverless is not directly related to microservices, but there are similarities between the two, such as business splitting, statelessness, and agile features.

Serverless is much smaller and more demanding than microservices in many ways. For example, microservices split services by service, and Serverless splits services by function. Microservices can share memory state across calls, and Serverless requires calls to be completely stateless. In addition, Serverless relies on BaaS to provide third-party dependencies, while microservices are free to choose third-party dependencies, such as using locally built traditional middleware stacks (such as local MySql and message buses) [6].

The idea of microservices is to decouple complex monolithic applications into multiple independent services. The purpose of microservices is to simplify the process of building complex applications by means of microservices [5]. But microservices and Serverless are compatible, all emphasize the decoupling of the system. It is not new to split the implementation of business logic into the granularity of Function. This method is not new: Microservice itself is a more widely used model, and has already had successful experiences from companies such as Netflix and Uber. Users can implement a microservice as a Function. An important goal of Alibaba Cloud's computational design is also to make it the best platform for building applications in microservices. Microserviced architecture is actually very challenging. After breaking into hundreds of services, it requires a highly automated release deployment system to manage the dependencies between microservices. In a way, Serverless and microservices are different levels, but they promote each other: microservices is the development model, and Serverless is the computing platform. The comparison table between serverless architecture and microservice architecture is shown in Table 3 below.

Table 3. 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 dependence	Free choice of third party dependence	Relying on third-party dependencies provided by BaaS
Level positioning	Development model	Computing platform

3.3. Serverless architecture vs. container

Serverless is an ideal, predictable workload with small resource requirements and short-term transactions. Containers have advantages for long-running processes and predictable workloads. Containers also provide more flexibility in application design, but require more management of the underlying infrastructure.

Serverless and the container are not on a single plane. Serverless is a software design architecture, and the container is the carrier of the software architecture. Although there is no public information, we can speculate that a serverless framework like AWS Lambda uses some degree of container technology, and it is difficult to implement language-independent and millisecond-level startup. Although there are already some open source projects that use Docker to implement the FaaS part of Serverless, we don't think that the public Serverless framework like AWS Lambda uses Docker directly. It must be a more lightweight and smaller container technology. We may be able to call it the Nano-Container [8]. The relationship between the serverless architecture and the container is complementary to each other rather than overlapping, and the container comparison table is shown in Table 4 below.

Table 4. Serverless architecture and container comparison table.

	Container	*Serverless architecture*
Level positioning	Software architecture bearer	Software architect
Workload	Long-running processes and unpredictable workloads	Ideal, predictable workload with small resource requirements and short-term transactions

4. Serverless architecture design

The Serverless architecture has three main levels: the scheduling layer, the computing layer and the base layer. The architecture design of the Serverless architecture is shown in Figure 2.

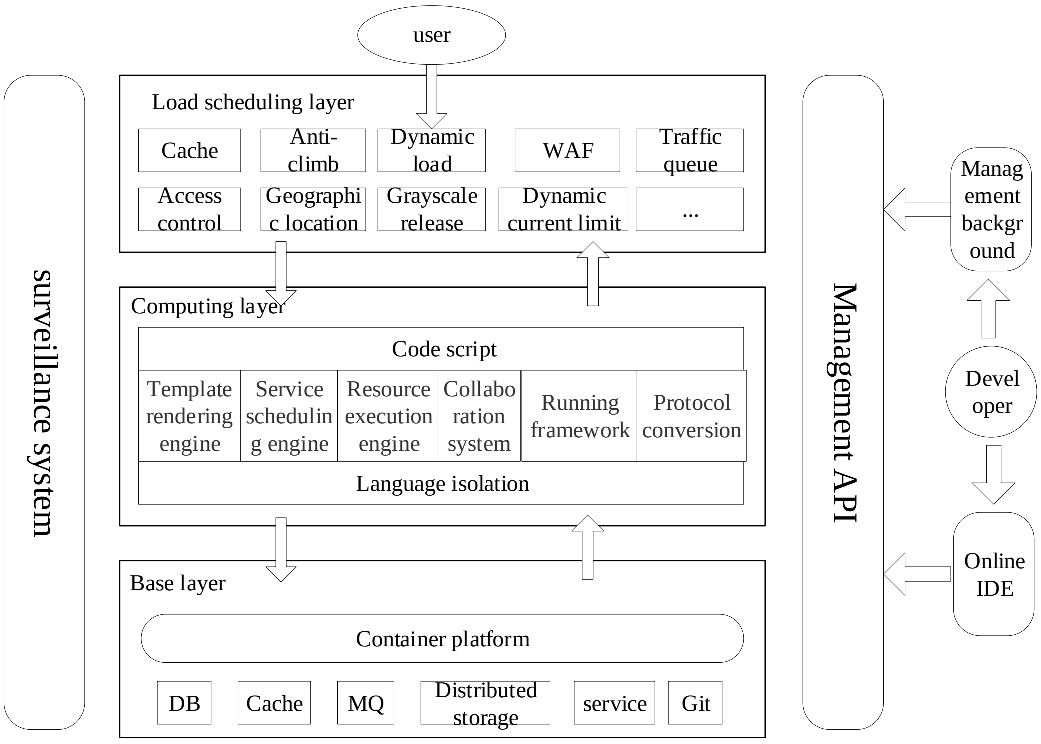


Figure 2 Architecture design of the Serverless architecture

The first layer is the scheduling layer. It seems simple, but it has many functions to implement. For example, how to quickly mount and expand the following expanded services in one second. Therefore, the dynamic load balancing of hot load is very important, and it can be found in every stage. And not only must the external traffic of the front end come in, but also the internal traffic.

The second layer is the computing layer, mainly how to quickly open resources for scheduling work. There are two main methods. The first one is to use the Docker container directly. Each expansion is a container, which is piled up by one container. But this approach wastes a lot of resources. The second is to use dynamic language to achieve, the advantage is that the runtime is isolated from each other, equivalent to a stand-alone container.

The third layer is the infrastructure layer to provide users with the necessary infrastructure such as virtual machines or storage resources to better schedule and manage physical resources.

At present, the serverless architecture is still not perfect, and there are still big problems. For example, the Serverless architecture has no problem in local deployment, but it can't be done online or in a test environment. Serverless is just getting started, not perfect enough, and there is still a lot of work to do in the future. At this stage, Serverless's next goal is mainly four points:

(1)Add more languages.

(2)Web-based IDE capabilities have improved.

(3)More skills are configured

(4)Incorporate an automated test system.

5. summary

As the company's system continues to grow and services continue to grow, more and more companies are undergoing a transition from a traditional architecture to a microservices architecture and a microservice architecture to a serverless architecture. Serverless is a relatively new concept in the field of software architecture, but it is a technological innovation with great influence. Adapting to a serverless and adapting culture that supports the technology, such companies will lead us forward in the future. The serverless architecture is just getting started, not perfect enough, and there is still a lot of work to do 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. 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.