International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 12 | Dec 2023 www.irjet.net p-ISSN: 2395-0072

Comparative studies of Serverless architecture

Munkhtsetseg Namsraidorj[1], Batsoyombo Khishigbaatar[2], Enkhtuul Bukhsuren[3], Uunganbayar Ganbold[4], Byambasuren Ivanov[5]

[1,2,3,4,5]*School of Information Technology and Electronics of National University, Mongolia*

Abstract - *In this article, we have studied how the technology implementing the "Serverless" architecture is used in modern software development by automating the tasks required for server development technical operations, making them independent of developers and how these technologies can be used in possible situations, the advantages and disadvantages of cloud technology.*
It also shows how this architectural solution supports the creation of a complex software solution that replaces the current physical and non-physical servers as well as comparative study of pricing and scalability of our testing system using AWS (Amazon Web Services), which provides 33% of the total use of "Serverless" architecture.

Key Words: Web-based software development, Serverless architecture, lambda function, cloud technology, runtime scalability

1.INTRODUCTION *(Size 11 , cambria font)*

Cloud technology refers to a technological solution based on the cloud environment. A cloud is an abstract space on the Internet where people can restore their digital resources such as software and files [1]. Cloud technology enables the resources in the cloud to be used with the help of network. Nowadays, majority of the people confuse the cloud with the internet, when in fact, the cloud is a part of the internet. In line with the development of technology, most of the software on the market uses cloud technology to some extent to suit their needs. As not everyone has the opportunity to create a physical server and network environment, choosing services based on cloud technology on the Internet is becoming a popular choice. On the other hand, even if it is possible to host an application on a physical server, it is currently not an optimal solution because maintaining the integrity of that server requires a lot of effort and money [2].

Cloud technology brings many benefits to every business applications in the market. It includes:

• Cost reduction - Creating a physical server environment and providing it 24 hours actively is costly. In addition, buying the necessary equipment and hiring an IT professional is expensive as well. By switching to cloud technology, these costs can be saved and it will even become possible to choose the service that best suits the needs and pay accordingly.

• Easy access - Most cloud-based services allow users full access from anywhere and on any device.

• Scalability of the working environment - Cloud technology can increase and decrease resources according to business requirements. In this way, the user may not worry about multiple user access to their server under any circumstances.

• Data protection - All data located in the cloud is fully secured by the system providing the service, at a high level of security.

• Unlimited storage - Multiple types of data can be stored without any capacity limitations.

2. "SERVERLESS" ARCHITECTURE

Serverless architecture is widely used in the market due to its advantages such as easy deployment, scalability, and cost savings. The possibilities of using "Serverless" architecture in software include:

- Software for web and smart devices - For example, the New York Times uses AWS Lambda, a "serverless" implementation of AWS, as the backend for their mobile app.

- IoT and Real-Time Data Processing - This architecture can be used for IoT (Internet of Things) and real-time data processing. For example, the Philips lighting system uses AWS Lambda to process sensor data for smart light bulbs [3]. Since lighting data needs to be processed in real-time environment, it is possible to use AWS Lambda without any server architecture.

- Serverless microservices - Using this architecture, it is possible to create the infrastructure of an extended working environment by responding to the requests of many users at the same time, so it is considered very suitable for executing a large number of 'microservices'. For example, the Coca-Cola Company processes its requests in real-time using the AWS Lambda service [4]. This architecture has a request management module based on the total number of requests received from at the moment and automatically increases and decreases the capacity of the operating environment.

2.1 Serverless architecture design

The most commonly used model for implementing serverless architecture is Function as a Service - FaaS. It is one version of the 3 main models mentioned in the previous section and it is the main model that uses the "Serverless"

architecture. As shown in Fig 1, developers divide the program code into specific independent functions and each of these functions performs the assigned to it with the help of an HTTP request or an event such as an email. After development phases, such as development and testing, in the deployment step features are deployed to the cloud platform along with the events. When the system is run, the event activates and the function, the cloud platform places it on an available server that has the resources required for the function, and if the server is not found, it creates a new server to run the function[4].

Serverless architecture offers many opportunities to users, but it also has several disadvantages. For example, the cold start latency (when the server is turned on for the first time), the infrastructure depends on others due to limited resource availability, debug and control, limited working environment and security issues need to be calculated in advance for the development of "Serverless" software. Developers should plan the software structure well and take full advantage of "Serverless" architecture to reduce the above problems as much as possible.

• Cold start latency - The time it takes until any "Serverless" function is activated for the first time and starts working is called cold start latency. This will give the user a feeling that it is running slowly at first, and this affects the overall performance of the software to some extent. One way to prevent this is to keep the function "warm" by running it for a certain period of time.

• Limited resource availability - Platforms that offer a “serverless” architecture impose certain limits on the amount of memory and capacity required to run each function. This poses a problem for computationally intensive software. Developers should improve the code they write to avoid using more resources than allocated.

• Infrastructure dependence (Vendor lock-in) - When choosing one of the companies that offer the implementation of "Serverless" architecture, such as Amazon, Microsoft and Google, the software can only work on that platform and infrastructure, and therefore very complicated to move to other platforms.

• Debug and control - During the development of "Serverless" functions, it is quite difficult to debug or develop step-by-step and to make certain controls. For example, using traditional debug and monitoring tools is not appropriate. Developers can only monitor and edit their usage in the cloud using platform-specific tools.

• Limited Runtime Environments - Most infrastructure platforms providing Serverless architectures offer a limited number of running environments. This causes considerable difficulty in software development because it makes it impossible for developers to use recently released programming languages or runtime environments.

3. AWS SERVERLESS INFRASTRUCTURE

To implement "Serverless", some platform or infrastructure is required. "Serverless" architecture has been in use for 15 years, but has been intensively used in recent years. Several major infrastructure solutions implementing this architecture are operating on the market[5].

"Serverless" requires a large infrastructure organization and capacity to support its continuous operation, so there is a lot of opportunity only for large cloud technology providers. Currently, there are 3 most used main infrastructures in the market: Amazon's AWS Lambda, Microsoft's Azure Functions and Google's Cloud Functions. Among these infrastructures, we chose AWS Lambda as the main technology for our research work. AWS is the largest cloud technology provider on the market and has implemented Serverless architecture most effectively with its Lambda service. Although people think that the AWS term "Serverless" is directly related to AWS Lambda, it is actually a part of the "Serverless" architecture[6]. Developing a complete software requires not only Lambda, but also many other services needed for databases and resources. AWS offers specific services for specific software requirements and we selected and implemented the technologies shown in Table 1.

Table -1: Components needed to implement “Serverless” on AWS used for test

Dedicate for	AWS services	Explanation
Computing	Lambda	business logic and calculations of software
Router	API Gateway	HTTP requests are directed, along with their associated data, to the Lambda function for processing.
Database	DynamoDB	NoSQL database
Storage	S3	File storage

3.1 AWS Lambda function working principle

Lambda acts as the most important part of the framework because it performs the logic and computations of the main operations of AWS "Serverless". Lambda executes a code function written by the developer as instructed by creating the appropriate computing environment for the function when an event occurs. It is fully capable of handling high traffic and capacity management automatically.

When an event occurs, such as an HTTP request or a file backup, the Lambda function is called along with the data relevant to the event and information about how the result should be returned. Lambda functions are code containing specific instructions that can be written in several programming languages supported on AWS. The following programming languages are currently supported on Lambda:

Node.js, Python, Java, Go, Ruby, .NET

In Node.js, event data, context objects and callback functions are taken as its parameters. A context object contains information about the Lambda function and its implementation, such as when it started running and how long it has been active. As for the third parameter, the callback function takes the result of the Lambda function as its argument and if there is an error in that data, it returns the error information to the source that called the event. Below is a small piece of Lambda function written in Node.js that returns a result of text type 'Hello from AWS Lambda'.

```
function lambdaFunction(event, context, callback) {
    // This is a success repsonse for the lambda function
    callback(null, 'Hello from AWS Lambda')
}
// Here the 'lambdaFunction' is defined as the handler of the lambda
   function
exports.handler = lambdaFunction
```

Fig -1: Lambda function written in Node.js

As mentioned in the previous section, the event passed to the function contains the data passed from the "trigger" of any other AWS service that called the function. For example, Lambda functions can be called from many places, such as HTTP requests from API Gateway services, file backups to S3, or when data is added to a DynamoDB database. Some of the most common services that invoke AWS Lambda functions include:

- HTTP requests from API Gateway
- When adding or deleting files in S3
- Modify data in DynamoDB
- AWS SNS (Simple Notification Service) to send any notification
- Voice command from Amazon Alexa

Lambda has some specific runtime and memory limitations. For example, by default, the Lambda function's runtime is set to a maximum of 3, so when the time expires, the function will "timeout" and cannot continue to run. In addition, memory is limited to 128MB, which indicates that it is not intended for large-scale computing.

The biggest advantage of serverless architecture is payment flexibility. Amazon rates its virtual server EC2 (Elastic Computing Cloud) by the hour. In terms of time estimation, Lambda is better than EC2, but Lambda does not charge at all if the function is not called. Therefore, Lambda is more beneficial than EC2. A total of 1 million Lambda function calls cost a total of 20 cents, or 706 MNT, and $0.000016 is paid per 1GB of memory used.

Table -2: Fees for running functions on x86 architecture [8]

Feature	Cost	Number of requests
First 6 billion requestн GB/sec	$0.0000166667 1 GB/per sec	$0.20 per million request
Next 9 billion requestн GB/sec	$0.000015 1 GB/per sec	$0.20 per million request
Up to 15 billion request GB/sec	$0.0000133334 1 GB/per sec	$0.20 per million request

Table -3: Fees for running functions on ARM architecture [8]

Feature	Cost	Number of requests
First 6 billion request GB/sec	$0.0000133334 1 GB/per sec	$0.20 per million request
Next 9 billion request GB/sec	$0.0000120001 1 GB/per sec	$0.20 per million request
Up to 15 billion request GB/sec	$0.0000106667 1 GB/per sec	$0.20 per million request

It also offers 1 million functions per month and 400,000 GB of storage space for free in a free trial [6]. More detailed payment information can be found in tables 2 and 3.

3.2 API Gateway

Amazon's API Gateway offers a comprehensive set of services for developers to easily create and manage the API they need. APIs act as the bridge between backend software services and the core business logic and operations needed to perform them. By using API Gateway, you can create Websocket APIs, that can create 2-way real-time communication, and RESTful APIs. Also, the service fully supports "containerized" and "serverless" architecture.

API Gateway handles all the work, including receiving and processing hundreds of thousands of API requests simultaneously. For example, it will be possible to distribute and manage simultaneous requests, CORS (Cross Origin Resource Sharing) settings, throttling, and control all at once.

Fees vary depending on the number of API calls and how much data is transferred.
More detailed payment information can be found in tables 4 and 5.

Table -4: Payment information of HTTP API on Gateway [8]

Number of requesto by month	Payment (million)
First 300 million	$1.00
+300 million	$0.90

Table- 5: Payment information of REST API on Gateway

Number of request by month	Payment (by million)
first 333 million	$1.00
next 667 million	$0.90
next 19 billion	$2.38
Up to 20 billion	$1.51

3.3 DynamoDB - NoSQL Database

Using DynamoDB, users can create database tables to store and retrieve any amount of data, and can handle large volumes of access. It is completely possible to adjust the capacity of the "tables" you have created without losing any time. With the help of AWS Management Console (console.aws.amazon.com), it is possible to monitor the use and capacity of databases, to make backups of tables at any time, and to restore the history of operations performed on table data for a period of 35 days. . DynamoDB offers built-in security protection, batch process or periodic data backup, automatic multi-region storage, in-memory caching and data I/O.

4.IMPLEMENTATION

In this section, we will show how to develops a small-scale or low-performance machine evaluation web software is and implement it in a completely "Serverless" environment and the performance and cost of it are compared. This system processes 2 main requests: processing machine data and calculating machine evaluation.

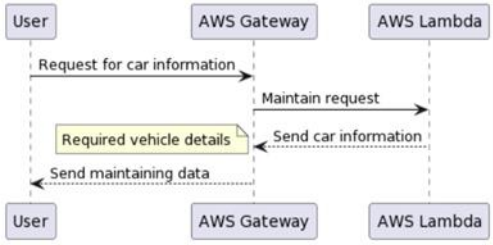

Fig-2: Sequence diaugram for maintaining car information

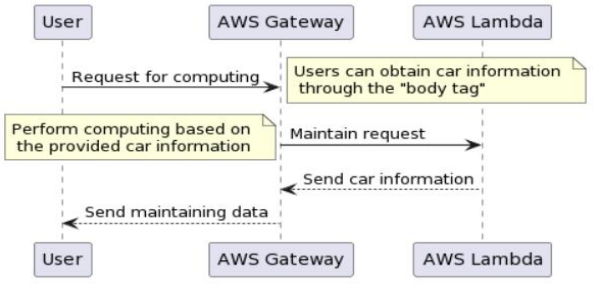

Fig-3: Sequence diagram for calculating car information

When a user sends a request, the API Gateway receives the request and calls the Lambda function associated with that API. A Lambda function can access a database or file and perform the necessary operations. The data is then processed and sent back to the API Gateway, which returns the processed data to the user. We show the technologies of the car cost calculation system in Fig 4.

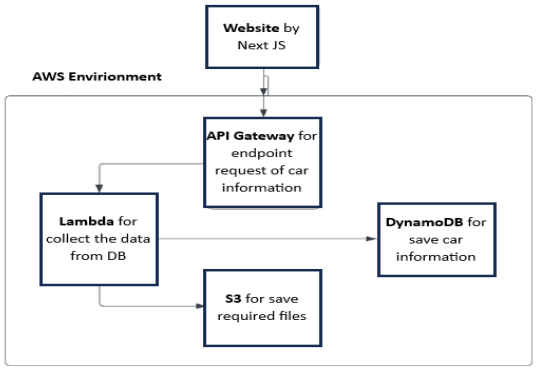

Fig-4: Technology options for the car cost computing system

The following steps were taken to make this system. It includes:

1. Designing software
2. Organize the DynamoDB database
4. Create S3 file storage, create IAM

IAM, or Identity Access Management, defines access rights for all services on AWS. For example, an IAM user has certain rights and with the help of those rights, it is possible to access services other than services like Lambda.

5. Create API Gateway and Lambda functions

When creating a function, the location of the function is very important. For example, in the case of accessing a function from the geographic location of Mongolia, the speed of access can be increased if the location of the function is as close as possible. The function will be placed in the location of Hong Kong (ap-east-1), because Hong Kong is considered to have the fastest network in Mongolia's Internet gateway. Lambda functions can be created using AWS' own web console (https://ap-east1.console.aws.amazon.com/lambda/home). The function will be created by filling in information such as the name of the function, which programming language the code will be developed in, and which server architecture it will work on. This feature will be used for the latest 'stable' version of Node.JS and for running on x86_64 or Intel CPUs.

As for the machine evaluation API, the main purpose of the function is to calculate using a machine learning model, and it is necessary to install large-scale Python libraries such as sckit-learn and numpy to perform large-scale calculations, so for the Lambda function, it is specified that the maximum size of the function can be 10MB, which means it cannot be uploaded directly with these libraries. Therefore, the Lambda function is created and deployed in a Docker container using the 'container image' option. In order to connect the 'docker container' to Lambda, it is necessary to upload it to AWS's ECR (Elastic Container Registry), a service that stores containerized images[7]. The following steps were taken.

Step 1: Set ECR
Step 2: Create a Docker container
Step 3: Deploy the Docker container
Step 4: Connect the container image created in ECR to the Lambda function

Returning to the Lambda creation section and selecting the container image in the new function section, the container placed on the ECR can be chosen. With this, the 2 main APIs to be used are ready. These APIs can be used by sending requests using any 'frontend' technology. The code snippet below shows how this function is called for a website developed in Next JS.

```
const handleFetch = useCallback(() => {
    ...
    axios
        .get(
            `https://iic8d24ha3.execute-api.ap-east-1.amazonaws.com
              /dev/data/${item.id}?` +
               params
        )
        .then((res) => {
            setData(res.data.data);
            setFilteredData(res.data.data);
            setLoading(false);
        });
}, [item, currentSlide, selectedItems]);
```

Fig-5: Call APIs created using Lambda and API Gateway

4.1 System cost comparison

This section compares how serverless architectures cost developers to develop using their own physical and non-physical servers. As of March 2023, the API Gateway received a total of 3,092 requests and 3,092 Lambda function calls were made, with a total of 263,472 seconds of activity. As for the DynamoDB database, a total of 15,090 readings were performed. Let's look at this in a table for the total amount of charges related to the access made in 1 month:

Services	Usage	Cost
Lambda	Active count 3.092	$5.12
	Active time: 263.472 секунд	
DynamoDB	Total reading number: 15.090	$0.00
	Total storage: 0.01GB	
API Gateway	Request number: 3.092	$0.01
S3	Total number of file reads:569	$0.00

Table-6: Billing data for "Serverless" implementations on AWS

Services Usage Cost

These prices are included in the 'free tier' set by AWS because the price is considered to be 0.0$ if it does not exceed a certain usage base, so it is quite possible to pay for medium and small software.

If always active and working server is needed to be developed, it can be hosted using the EC2 virtual server service on AWS. The method accepts any number of accesses and the monthly payment is always fixed. For example, receiving 1 request per second and receiving 1000 requests will be charged exactly the same. However, since the server will have limited resources, it will be necessary to calculate the capacity resources when receiving 1000 requests. The server receives less traffic at night most of the time, but has the disadvantage of paying a certain amount for it.

EC2 сервер	CPU	RAM	Cost a hour	Cost per month
t3.micro	2	1GB	$0.0104	$8.35
t3.small	2	2GB	$0.0208	$14.976
t3.medium	2	4GB	$0.0416	$29.952
t3.large	2	8GB	$0.0832	$59.904
t3.xlarge	4	16GB	$0.1664	$119.808
t3.2xlarge	8	32GB	$0.3328	$239.6.16

Table-7: Common server types on EC2 [8]

As shown in Table 7, if a serverless implementation is implemented on a server, it will be necessary to organize retrieving of the main access and main database servers. In creating a virtual server using EC2 service, the cheapest option is a machine with 1 CPU and 1GB of RAM, but it is impossible to make many accesses at the same time to RAM and CPU for calculations on the machine learning model. It is because considering that 1 calculation takes about 10 seconds, 1 CPU itself is a heavy load, so processing accesses at the same time will not be possible. Therefore, it is necessary to have at least 2 CPUs with 2GB of RAM. There is a server called t3.small that meets this requirement and the total payment will be $14.976 per month or $0.0201 per hour. In addition to the server, if we assume that the database server will also be included, it will be necessary to pay a regular monthly fee of $29,952. Payment details can be found in the table above.

4.2 Comparation of scalability

The most important and major advantage of serverless software is the ability to automatically scale up to handle large amounts of traffic without developer control. For example, the function can be run any number of times depending on how many requests there are. Then, starting from 1 user request, 1000 and 10000 user requests will be

processed in the same way. So when implementing these on AWS Lambda, Lambda is typically conFigd to receive a total of 1000 concurrent requests. This means that the function will run simultaneously in 1000 separate 'execution environments'. If you want to increase the limit, you can make an additional request to AWS. AWS Lambda runs the function in an independent standalone environment. Before starting to run the function, it is necessary to load the function into the operating environment and then the function is called and executed. As shown in the diagram below, one row represents the runtime environment itself, while the green square represents the loading time of the function and the orange represents the execution time when it was called.

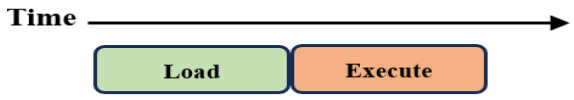

Fig-5: Call the Lambda function

After a function is executed in a runtime environment, the same function that is called by another request in that runtime environment can be called directly and used without reloading.

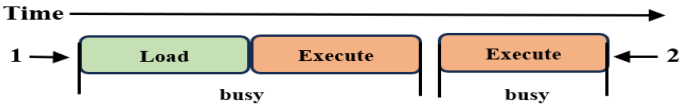

Fig-6: Lambda function re-calling

Fig 5 shows that Lambda does not need to reload a function that is called after a function is loaded in a single runtime. Currently we are only considering lambda working with one environment. In a real environment Lambda can run 1000 environments at the same time as its typical settings as mentioned in the previous section[9]. So, when there is more than 1 environment, Lambda follows 2 basic rules for handling multiple requests. It includes:

1. If there is an environment with already loaded in function, call the function in that environment to run it

2. If such an environment does not exist, create a new working environment

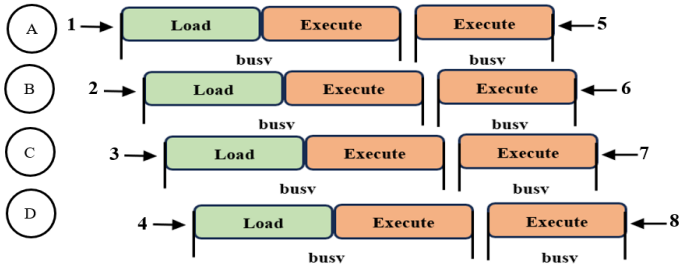

Fig-7: Lambda function invocation in a multi-threaded environment

As you can see in Fig 7, each row represents an environment and the table shows how the environment handles a total of 8 requests.

Req no.	Procedure of lambda	Reason
1	Create A working envirionment	Initiating a new environment with the first request due to the absence of a pre-existing one
2	Create B working envirionment	Establishing a new environment due to the current unavailability of environment A.
3	Create C working envirionment	Establishing a new environment due to the current unavailability of environment A and B.
4	Create D working envirionment	Establishing a new environment due to the current unavailability of environment A,B and C.
5	Environment A is currently used	Environment A is being reused because function 1 that was running in environment A has ended
6	Environment B is currently used	Environment B is being reused because function 2 that was running in environment B has ended
7	Environment C is currently used	Environment C is being reused because function 3 that was running in environment C has ended
8	Environment D is currently used	Environment D is being reused because function 4 that was running in environment D has ended

Table-8: Diagram of activity of functions

However, in the case of development using physical servers and virtual machines, the developer needs to organize multi-level configuration and proper use of resources to ensure continuous availability. For example, one software server will be divided into many parts, all will receive a request through one main input and run it in turn with the help of an algorithm to find a more specific order for the servers that have divided the request into several parts.

5. CONCLUSIONS

In this article, we carried out a theoretical study of "Serverless" architecture and then developing "Serverless" in a practical environment using the AWS cloud platform according to the research, studying the AWS environment, getting to know the application services and performing practical implementations, experimenting in real life conditions and learning the use, advantages and disadvantages of "Serverless" architecture. watched. From this it shows that Serverless architecture fully enables developers to develop scalable software as a working environment in a short period of time with fewer human

resources and lower costs. According to the results of the study, the significant cost of implementing Serverless can also be reduced by the development of major software available in the current market. Not only that, it allows developers to focus only on the quality of their own code and leave the infrastructure and operating environment issues to the platform, creating a more productive and high-quality product. It is concluded that the use of "Serverless" architecture in software production will greatly contribute to the speed and quality of development, as infrastructure problems will be solved by third parties.

REFERENCES

[1] What is Cloud Technology, and How Does It Work?, https://dynamixsolutions.com/what-is-cloud-technology-and-how-does-it-work/
[2] 7 Most Popular Applications of Cloud Computing : All You Need to Know, https://www.simplilearn.com/applications-of-cloud-computing-article
[3] On the use of IoT and Big Data Technologies for Real-time Monitoring and Data Processing, Y. Nait Maleka,b, A. Kharbouchb,c, H. El Khoukhib,c, M. Bakhouyaa,*, V. De Floriod,c, D.
[4] Vaishnavi Kulkarni, 2022, A Research Paper on Serverless Computing, international journal of engineering research & technology (ijert) Volume 11, Issue 09 (September 2022),
[5] Jiang, Lizheng & Pei, Yunman & Zhao, Jiantao. (2020). Overview Of Serverless Architecture Research. Journal of Physics: Conference Series. 1453. 012119. 10.1088/1742-6596/1453/1/012119.
[6] Ortiz, Ariel. (2019). Architecting Serverless Microservices on the Cloud with AWS. SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education. 1240-1240. 10.1145/3287324.3287533.
[7] Slobodan Stojanovic ´ Aleksandar Simovic (2019) Serverless Application with Node.JS
[8] AWS Lambda Pricing https://aws.amazon.com/lambda/pricing/
[9] Api Gateway Guide https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html