Serverless Architecture: the Future of Cloud Computing

Rajni Jha
M.Tech Student
Department of Computer Science
BIT, Mesra(Noida Campus)
Noida , India

Dayanand
Assistant Professor
Department of Computer Science
HMR Institute of Technology & Management,
New Delhi, India

***Abstract*-The strategy of software developer to develop software architecture lies in the range of its blueprint or plan via conceptual model to achieving desired target. It is assuredly true that the lack of architecture can make software failure. Good architecture may help to progress a web or mobile application and poor architecture may cause serious issues leads to highly costly to recover again. Understanding the signification of choice regarding architecture and making plan in advance is paramount to creating effective, high performing and basically successful software systems. . The existing software architecture is typically three tier architecture, with heavy lifting weight residing in the back end. The user interface really just serves as a convenient way for the user to drive functions in the back-end. But this is inefficient to build because there is a huge amount of duplication. In this architecture there are lots of layers to touch for adding new features. The vision for future that sees a generation of light weight cloud connected devices is based on server less architecture. With the release of function execution as service technologies such as AWS Lambda developers are now building entirely server less platform. In these new architectures, traditional backend servers are replaced with cloud functions acting as discrete single purpose services. These are stateless functions are used to perform protected actions, execute business logic and set up interaction with vast array of powerful cloud services. This is characterized by rich thick client applications talking directly to cloud data stores and small cloud based micro services for protected workloads and service orchestration. This article introduces server less architecture, key services such as AWS Lambda and describes the principles and benefits of server less architecture system.**

Key Words: Software Architecture, Server less Architecture,

INTRODUCTION

Server less computing is the next layer of abstraction in cloud computing. It does not mean that there are no servers, but rather the underlying infrastructure (physical and virtual hosts, virtual machines, containers), as well as the operating system, is abstracted away from the developer. Applications are run in stateless compute containers that are event triggered (e.g. a user uploading a photo which triggers notifications to his/her followers). Developers create functions and depend on the infrastructure to allocate the proper resources to execute the function. If the load on the function grows, the infrastructure will create copies of the function and scale to meet demand. Server less computing supports multiple languages so developers can choose the tools they are most comfortable with. Users are only charged for runtime and resources (e.g. RAM) that the function consumes; thus there is no longer any concept of under or over provisioning. For example, if a function runs for 500ms and consumes 15 MB of RAM, the user will only be charged for 500 ms of runtime, and the cost to use 15 MB of RAM.

Server less architectures are extension of micro services. Similar to micro services, components. While micro services may group similar server less architecture applications are broken down into specific core functionality into one service, server less application depicts functionality into finer grained components. Custom code is developed and executed as isolated, autonomous, granular functions that run in a stateless compute service. To illustrate this point, let’s look at a simple example of how a micro service and server less architecture differ. In Figure 1, a client interacts with a user microservice. A container is pre-provisioned with all of the functionality of the user service residing in the container. The service consists of different functions (update_user, get_user, create_user, delete_user) and is scaled based on the overall load across the service. The service will consume hardware resources even while idle, and the user will still be charged for the underutilized resources.

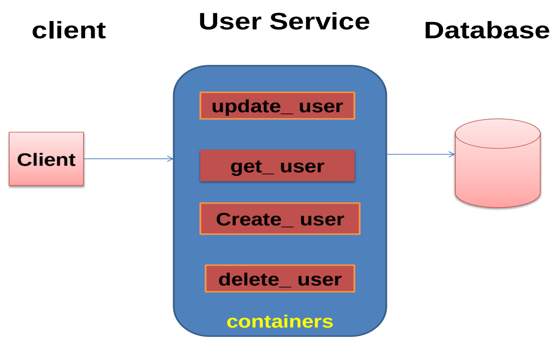

Figure 1: Micro services Architecture

For a server less architecture, the “User” service would be separated into more granular functions. In Figure 2, each API endpoint corresponds to a specific function and file. When a “create user” request is initiated by the client, the entire codebase of the “User” service does not have to run; instead only create_user.js will execute. There is no need to pre-provision containers, as standalone functions only consume resources when needed, and users are only charged on the actual runtime of their functions. This granularity also facilitates parallel development work, as functions can be tested and deployed independently.

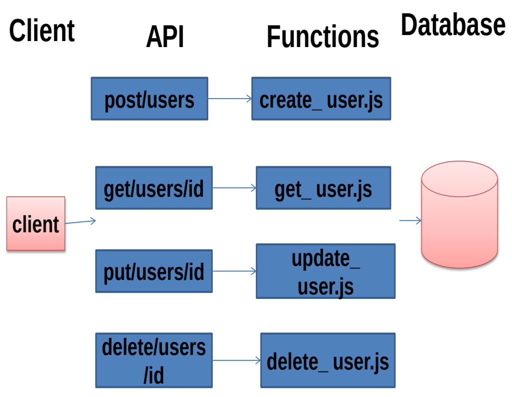

Figure 2:Server less Architecture

The evolution of the term server less can be attributed to the growing popularity of compute technologies like AWS Lambda and architectural patterns like crafting systems without traditional backend servers generally this includes integration of rich array of services in AWS or 3rd party web services. The word server less is a bit of misunderstanding that whether users use a compute services such as AWS Lambda to execute code or interact with application program interface, there are still servers running in the background. The difference is that in this architecture servers are hidden from users. Someone else takes care of core detail of infrastructure management so that users saving time for other things. Therefore, server less is about running code in a computing service and interacting with services and APIs to get the result. The difference is that in this architecture servers are hidden from users. Someone else takes care of core detail of infrastructure management so that users saving time for other things. Therefore, server less is about running code in a computing service and interacting with services and APIs to get the result.

1	User interface
2	Client side model binding
3	Client side service layer
4	Server side service layer
5	Server side model binding
6	server side database mapping
7	Database storage

Figure 3.3-tier architecture, lot of layers to touch, just to add a new feature

Let us consider user interface application which contains all of the set up logic that currently sits in the backend, and can talk directly to a range of cloud-enabled services to perform unique functions, such as authentication, storage, notifications etc. Now, adding new feature, it is adding to front-end code then few layers are required to touch as shown in figure 2

1	User interface
2	Client side model binding
3	Client side service layer
4	Database storage

Figure 4.Layers of Server less architecture- cut out the middleman and brings the UI closer to the data store

Lambda is a computing service from Amazon Web Services. Lambda can execute code in a massively parallelized way in response to events. Lambda takes users code and runs it without any need to provision servers, install software, deploy containers, or worry about low-level detail. AWS takes care of provisioning and management of their Elastic Compute Cloud (EC2) servers that run the actual code and provides a high-availability compute infrastructure including capacity provisioning and automated scaling that the developer does not need to think about. By taking Lambda and making use of various powerful single purpose APIs and web services, developers can build loosely coupled, scalable, and efficient architectures quickly. Figure 3 shows server less architecture there is no single traditional back end.
The front end of the application communicates directly with services, the database, or compute functions via an API gateway. Some services, however, must be hidden behind compute service functions where additional security measures and validation can take place. The user's browsers can send credentials to interact directly with the different cloud services such as authentication, protected file download / upload, credit card payments, notifications (email, SMS, push) and real time streaming database access (with offline sync capability). Mobile hardware these days is powerful and people prefer a richer and faster user experience. Ubiquitous, always reliable internet hasn't yet arrived. There is still suffering degraded connections and loss of connection regularly. So, general preference is for richer, more powerful applications on our mobile devices that can deal with offline scenarios and sync data when returning online.

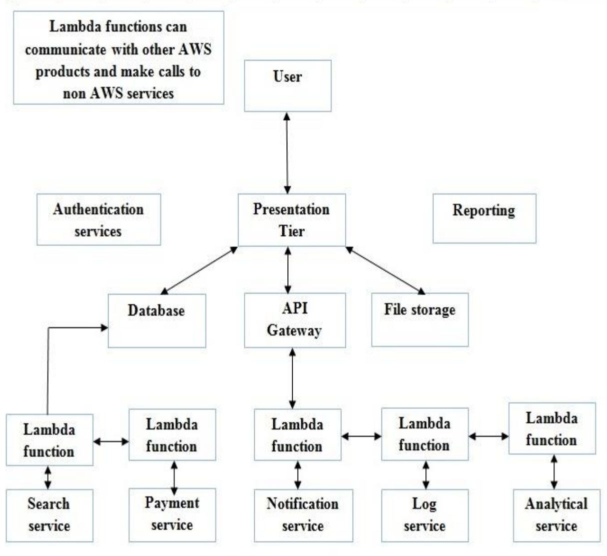

Figure 5.Serverless architecture

Figure 5.Serverless architecture

PRINCIPLES OF SERVER LESS ARCHITECTURES

There are five principles of server less architecture that describe how an ideal serverless system should be built. Use these principles to help in creating serverless architecture.

- ➢ Use a compute service to execute code on demand (no servers).
- ➢ Write single-purpose stateless functions.
- ➢ Design push-based, event-driven pipelines.
- ➢ Create thicker, more powerful front ends.
- ➢ Embrace third-party services.

Benefits of Server less Computing

Costs Scale With Usage: One of the biggest benefits of server less computing is that you only pay for the runtime of your function. With server less computing, enterprises could build out an entire infrastructure and not pay for any compute resources until customers start using the application.

Elastic Scalability: Elastic scalability is also simple with a server less architecture. If a function needs to scale, the infrastructure will make copies of the function to handle the load. An example of this could be a chat bot that responds to weather requests. In a server less architecture, a chatbot function would handle the response by retrieving the user's location and responding back with the temperature. For a few requests this is not a problem, but what happens if the chatbot service is flooded with thousands of request a second. For this scenario, the chatbot function would automatically scale by instantiating thousands of copies of the function. Once the requests have subsided, the environment would terminate the idle instances and scale down, allowing costs to scale proportionally with user demand.

Rapid Development and Iteration: Server less computing is ideal for companies that need to quickly develop, prototype, and iterate. Development is quicker since there are not any dependencies on IT Ops. Functions are single threaded, which makes debugging and deploying functions simpler. The build process is also broken down into smaller and more manageable chunks. This increases the number of changes that can be pushed through the continuous delivery pipeline, resulting in rapid deployment and more iterative feedback and iterations are fast as the architecture is conducive to making large code changes quickly, resulting in more customer feedback and better product market fit.

Less System Administration: Serverless does not mean that you completely obviate the operational element of your infrastructure, but it does mean that there is less system administration.

There are no servers to manage, provision, and scale, as well as no patching and upgrading.

Developer Productivity: By using a serverless architecture, developers can focus more on writing code than having to worry about managing the operational tasks of the application. This allows them to develop innovative features and focus on the core business logic that matters most to the business.

CONCLUSION

Cloud continues to be a game changer for IT infrastructure and software development. Software developers need to think about the way they can maximize use from cloud platforms to gain a competitive advantage. Serverless architectures are the latest advance for developers and organizations to think about, study, and adopt. It is an exciting new shift in architecture that will grow quickly as developers embrace compute services such as AWS Lambda. Today there are serverless applications that support thousands of users and carry out complex operations including heavy-duty tasks such as video encoding and data processing. In many cases, serverless architectures can achieve a better outcome than traditional models and are cheaper, and faster to implement. There is also a need to reduce complexity and costs associated with running infrastructure and carrying out development of traditional software systems. The reduction in cost and time spent on infrastructure maintenance, and the benefits of scalability are good reasons for organizations and developers to consider serverless architectures. It is likely that the push for serverless back ends will accelerate over the coming years.

REFERENCES

[1] Ken Fromm, 3x tech co-founder, " Why the future of software and apps is serverless", original posted on read write on octber 15, 2012.
[2] Danilo Poccia," AWS Lambda in action event driven serverless application", pages 384, November 2016.
[3] Danilo Poccia, "Agile Development for serverless platforms", page no 124 Manning publication April 2017.
[4] Peter Sbarski and Sam Kroonenburg's book," Serverless Architectures on AWS".
[5] Cagatay Gurturk, Martin Linderberg, software engineer Home 24 AG,"The journey to Serverless", 2016.
[6] Drew Dennis, Maitreya Ranganath, Ajoy kumar,"Serverless architectural pattern and best practices", ARC 402, November 30, 2016.
[7] Jeremy Edberg, Founder of CEO, MinOPS,"Getting started with serverless architectures", CMP 211, December 2016.
[8] Ben Kehoe, cloud robotics research scientist, "FaaS is stateless, and AWS step functions provides state as a service", 5 december 2016.
[9] Wayne Scarano," Prepare for the Next Disruptive Waves: Serverless & InterCloud",AWS-SA, CISSP, CCSK, SABSA Cloud/Cybersecurity Analyst wscarano@sga.com ©2016 SGA Business Systems, Inc.
[10] Mike Roberts, "Serverless architectures", 4 august 2016.