International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-7, Issue-4, July 2020
www.ijirem.org

Perceptions of Client and Server Side Load Balancing in Microservices

V. Indrani
Department of CSE,
JNTUH/Vignan Institute of Management and Technology for Women, Hyderabad, INDIA,
vmtw.csehod@gmail.com

D. Swaroopa
Department of CSE,
JNTUH/Vignan Institute of Management and Technology for Women, Hyderabad, INDIA,

D.Deepthisri
Department of CSE,
JNTUH/Vignan Institute of Management and Technology for Women, Hyderabad, INDIA,

ABSTRACT
Microservices is a collection of small individual services of a single functional module or an application. Microservices address the challenges in monolithic and provides best services like loosely coupled, independently deployed, highly maintainability and testable, owned by small teams. It had address the challenges in monolithic but has to address the challenges with the own like deployment complexity, distributed network complexity. One of the challenges of Microservices is Deployment complexity which has to consider various parameters load balancing, virtual networks, memory storage, firewalls and auto scaling. In order to scale the client and microservices independent of each other load balancing is used. Load of microservices can be handled with the help of load balancing, security and remains available. Load balancing is defined as to efficiently distribute network traffic and computing properties across a group of backend servers. A load balancer performs the following functions like Distributes client requests and network load efficiently across multiple servers.

Keywords
Microservices, Load balancing, Client side load balancing,Server side load balancing,EC2(Amazon Elastic Compute Cloud).

1. INTRODUCTION
Microservices or Microservices architecture constitute a software architectural style that approaches a single applications as a suite of small services, each running its own process and communicating with lightweight mechanisms using API.The services are small, highly decoupled and focus on providing a single "useful" business capability.Genrally Microservices involve very small centralized management.This could be written in different languages and data storage technologies. It enhances rapid ,frequent and reliable delivery fo large complex applications and it also enables an organization to evolve its technology stack.

1.2 Purpose of Microservices

1.2.1 Monolithic
A traditional way of building applications is Monolithic architecture. A monolithic application is built as a single and indivisible unit. Usually, such a solution comprises a client-side user interface, a server side-application, and a database. It is unified and all the functions are managed and served in one place. Despite having different components/modules/services, the application is built and deployed as one Application for all platforms (i.e. desktop, mobile and tablet) using a data source. This type of architecture suits for small application which does not require any changes in future. But it is a drawback for lager application which requires modification based client requirement.

1.2.2 Microservices
Monolithic is not adoptable to the technology advancements where as Microservices have this with its extensible features like scalability, flexibility and agility[6].

1.3 The Six Characteristics of Microservices

1.3.1 Multiple Components
Microservices can provide the service by dividing the software into multiple component services. Each component of Microservices can be deployed, tweaked, and then redeployed independently without compromising the integrity of an application. Individual independent services can be redeployed instead of deploying entire applications. This feature have its downsides, complexity will increase when redistributing responsibilities between components and expensive remote calls, coarser-grained remote APIs

1.3.2 Built For Business
The microservices style, unlike monolithic approach, it is organized based on business capabilities and priorities. Microservices architecture utilizes cross-functional teams. The responsibilities of each team say UIs, database, Technology layers or server-side logic are to make specific products based on one or more individual services communicating via message bus.The other side this feature restricts teams working on other teams and also suits only for larger application where maintaining multiple teams for small application is expensive

1.3.3 Simple Routing
Microservices- simply receive request, process them, and generate a response accordingly .It is similarly like classical UNIX system. Microservices have smart endpoints that process information and apply logic, and dumb pipes through which the data flows.

1.3.4 Decentralized
Centralized governance isn't optimal for Microservices as it involves a variety of technologies and platforms. So

Decentralized governance is favored by the microservices community. In a Microservice application, each service usually manages its unique database and its developers strive to produce useful tools that can be used by others to solve the same problems.

1.3.5 Failure Resistant

Like a well-rounded child, microservices are designed to deal with failure. microservices involves several unique services and they are communicating each other ,so failure of service can be shared by other service. In these cases, the client should accept its neighboring services to work. Risk of failure can be reduced by monitoring microservices but it increases complexity as it involves several components together,

1.3.6 Evolutionary

Microservices architecture is an evolutionary design and, is right for evolutionary systems where you can't fully anticipate the kinds of devices which will at some point be accessing your application.Many monolithic architecture applications are migrated to Microservices that interact over an older monolithic through APIs .

1.4 Examples of Microservices

Many of applications and large scale website have changed their trend from a monolithic to microservices. Some of the examples are which adopt the monolithic are Netflix, eBay, Amazon, the UK Government Digital Service, Twitter and PayPal etc.

1.4.1 Netflix

Netflix is the king of microservices. In 2015, JAX unanimously selected Netflix for the Special Jury award, citing the developer team's huge influence on innovation and IT. Today Netflix services 93.8 million users globally, streaming more than ten billion hours of movies and shows.

1.4.2 Amazon

Microservices architecture allowed Amazon to transition to continuous deployment, and now Amazon engineers deploy code every 11.7 seconds and able to handle countless calls from a variety of applications.

1.4.3 eBay

eBay knew that in order to stay competitive, they had to deliver quality features and innovations at an accelerating pace. Making a Multiple components like database,application tiers , search engine and implementing microservice architecture allowed eBay to answer the arising challenges of the growing complexity of the codebase, enabling a faster time-to-market while maintaining the site stability and improving developers productivity.

1.4.4 Uber

Uber also transformed from monolithic to microservices as it is expanding services to different locations with a numerous features .Uber introduced an API Gateway and independent services that have individual functions and can be deployed and scaled separately[8].

1.5 Challenges in Microservices

Microservices have numerous benefits of a loosely coupled design of a Microservices even though the challenges like how to quickly roll out, troubleshoot and mange these Microservices remains unavailable. Distributed data management is demanding feature of the microservices even though it is facing the problem like implementing distributed transactions is much more complex.Automated Tests are important in microservices but it is time consuming and more complex when the integration test to be done and also it is harder to find which functionality is broken down when integration test fails.It is harder to find Service faults in a distributed system than a monolithic.Remote Procedure Calls (RPC) may become a challenge when developing applications under microservices because RPC are expensive and take much longer than local calls[5].

2. LOAD BALANCING

Load balancing is defined as to efficiently distribute network traffic and computing properties across a group of backend servers. A load balancer performs the following functions like distributing client requests and network load efficiently across multiple servers.

In order to scale the client and microservices independent of each other, a client interacting with a microservices based application does not need to know about the instances that are serving it. This is provided by reverse proxy or a load balancer which is known as de-coupling. Load balancer, which balances the payload among multiple instances of services[9].

The four challenges to be considered in loadbancing are:

- Understanding Your Load.
- Dealing with Dynamic Changes.
- Knowing When There's an Issue.
- Preventing an Emerging Issue.

2.1 Brief History of Load Balancing

Load Balancing has introduced in 1990s as hardware appliances distributing traffic across a network. Later with ADC(Application Delivery Controllers) load balancer manages many responsibilities in order to improve the accessibility, security and seamless access of the application running on servers at peak times. ADCs are classified as hardware appliances, virtual appliances (essentially the software extracted from legacy hardware) and software-native load balancers[7].

In the cloud, software ADCs perform similar tasks to hardware. They also come with added functionality and flexibility. They let an organization quickly and securely scale up its application services based on demand in the cloud. Modern ADCs allow organizations to consolidate network-based services. Those services include SSL/TLS offload, caching, compression, intrusion detection and web application firewalls. This creates even shorter delivery times and greater scalability[2].

2.2 Forms of Load balancing

- Load balancer generally group in to two categories. They are
- Layer 4: Act in network and transport layer protocols (IP, TCP, FTP, and UDP).
- Layer 7: Requests are distributed based upon data found in application layer protocols such as HTTP.
- Also Load Balancer reside in two types. They are
- Hardware Load Balancer: F5 BIG-IP, Cisco, Citrix
- Software Load Balancer: NGINX, HAProxy, LoadMaster

2.3 Types of Load balancers in AWS ELB(Amazon web services Elastic Load Balancers

Load Balancing can be of two types: Client side Load balancing and Server side load balancing[4].

2.3.1 Client side Load balancing

In this type of load balancing it maintains an algorithm like round robin or zone specific, by which it can invoke instances of calling services. The client side load balancing can be done with programmtically with the help of algorithms which adds aditional feature to client side load balncer.. With client load balancing, a WLAN distributes new connections across multiple Access Points in order to make the best use of the network and radio spectrum[2]. Client will take the decision to forward the request to the server. How it works is very simple: Client holds the list of server IPs that it can deliver the requests. Client selects an IP from the list randomly and forwards the request to the server.All these different load balancers uses different algorithms to distribute the load among the application pool. Some of the industry standard Load Balancing algorithms are:

2.3.1.1 Round Robin

This method continuously rotates a list of services that are attached to it. It assigns the connection to the first service in the list, and then moves that service to the bottom of the list while virtual server receives a request[6].

2.3.1.2 Least connections

The default method, it selects the service with the fewest active connections, when a virtual server is configured to use the least connection

2.3.1.3 Least response time

This method selects the lowest average response time and the service with the fewest active connections.
With the microservices Client side load balancing plays a vital role where Services like Netflix, Ribbon and Eureka components helps client side load balancing have similar features to server side load balancing like fault tolerance, caching and batching[3].

2.3.2 Server side Load Balancing

A Server side load balancing was the most common method used in past to mange our application load .it locates in the middle of client and server to farm accepting incoming network and application traffic . Mostly load balancer will check the health of the server pool underneath and use the algorithm which we discussed earlier to distribute the load[7].
AWS ELB available in three versions which perform different tasks.

- The Version 1 provides detailed instructions for using Classic Load Balancers.
- 2nd version provides detailed instructions for using Application Load Balancers.
- 3rd provides detailed instructions for using Network Load Balancers.
- These are the unique Versions of AWS ELB; let's discuss them one by one:

2.3.2.1 Classic Load Balancers

Classic Load Balancers distribute upcoming traffic to different EC2 instances in multiple Availability Zones. During this process, there is a chance of the fault tolerance of your application. These Load Balancers detect healthy and unhealthy instance and direct the traffic towards only healthy ones. It also helps in a way such that without disrupting the flow of requests to your application you can add or remove instances from your load balancers as your need changes. AWS ELB calculates the majority of workloads automatically. Protocol and port which a person configures are used to detect the connection requests from clients it also forwards requests to one or more registered instances. The number of instances can be modified. Health checks can be monitored so that the load balancer only sends requests to the healthy instances.

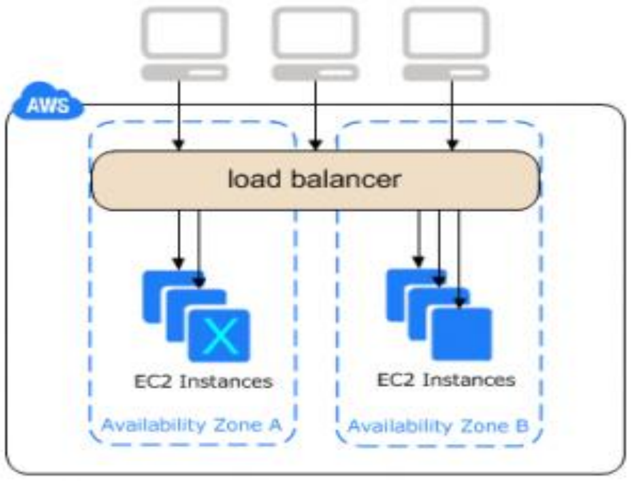

Figure 1: Classic Load Balancers

Benefits of Classic Load Balancers-

- Provides Support to TCP and SSL listeners.
- Support to Sticky Session.
- Support to EC2- Classic.

2.3.2.2 Application Load Balancers

After receiving the request Application Load Balancer analyzes the rules provided by the listener in priority order and determines the rule which has to apply. After that, it selects a target from the target group for the rule action. An Application Load Balancer functions at the Open Systems Interconnection (OSI) model which is the seventh layer of the OSI model[2].
The User can analyze the rules of the listener and can modify it by sending it to different target groups based on the content of the application traffic even when the target is associated with multiple target groups. Addition and Deletion of tags can do from the load balancers as per your needs. This can done without breaking the flow of your requests of the application.

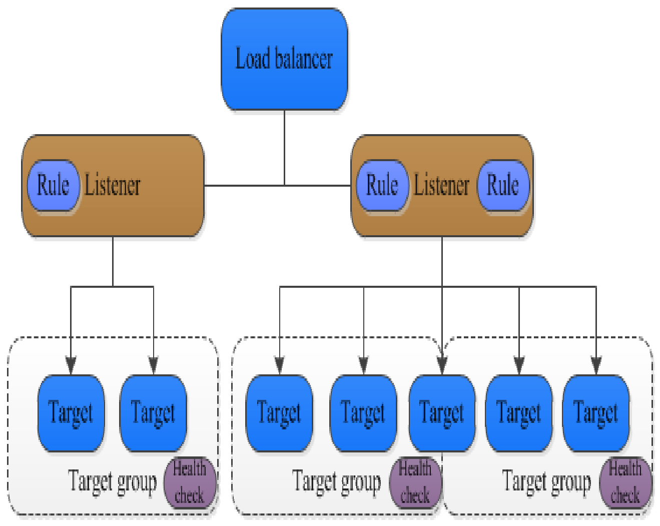

Figure 2: Application Load Balancers

Benefits of Application Load balancers-
• Load Balancer's performance improves in Application Load Balancer.
• Access logs containing Data compress such that they may not require the additional space.
• Provides benefit for registering targets by IP address, including targets outside the VPC for the load balancer.

2.3.2.3 Network Load Balancers

It is the fourth layer of the Open System Interconnection Model. After the load balancer receives a connection request, it selects a target from the group which targets for the default rule.

2.3.2.4 Elastic Load Balancer

creates the load balancer node in the availability zone After enabling the availability Zone and Each load balancer node automatically distributes traffic across the registered targets in its Availability Zone only.

2.3.2.5 Cross-zone Load

Balancing distributes the traffic across registered targets in all enabled Availability Zones.Increase in the fault tolerance of the applications Enables Multiple Availability Zone to cause harm and it will happen only if every target group has at least one target in each enabled Availability Zone. The problem can overcome in such a way that if one or more target groups do not have a healthy target in an Availability Zone, the IP address for the corresponding subnet from DNS is removed. If a person attempts again the request fails.

2.3.2.6 Ribbon: Load Balancer Without Eureka

Ribbon acts as a client-side load balancer, which gives you a lot of control over the behavior of HTTP and TCP clients.Connection timeouts, retries, retry algorithm (exponential, bounded back off) etc are provided by Ribbon's Client component. Ribbon comes with a pluggable and customizable Load Balancing component[1].

2.3.2.3 The Envoy Proxy

configuration consists of listeners, filters and clusters,it is an L7 proxy and communication bus designed for large modern service-oriented architectures.Layer 7 load balancers operate at the highest level in the OSI model, the Application layer (on the Internet, HTTP is the dominant protocol at this layer).

3. CONCLUSION

In this paper it describes efficiency of Microservices rather than monolithic like deployment complexity, distributed and network complexity. It also addresses load balancing at the client as well as severs side. Hence load balancing provides the solution how a load can be handled in microservices without changing security and integrity.

REFERENCES

[1] P.Geetha, C.R. Rene Robin,A comparative-study of load-cloud balancing algorithms in cloud environments, in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)

[2.] R. Li, Baker street: avoiding bottlenecks with a client-side load balancer for microservices (2015), https://thenewstack.io/baker-street-avoiding-bottlenecks-with-a-client-side-load-balancer-for-microservices/. Accessed Feb 2019

[3].H. Liu, R. Zhang-Shen, On direct routing in the valiant load-balancing architecture, in Proceedings of the Global Telecommunications Conference (GLOBECOM), vol. 2 (2005), p. 6

[4] D. Malavalli, S. Sathappan, Scalable microservice based architecture for enabling DMTF profiles, in 2015 11th International Conference on Network and Service Management (CNSM) (2015), pp. 428–432

[5] A. Messina, R. Rizzo, P. Storniolo, M. Tripiciano, A. Urso, The database-is-the-service pattern for microservice architectures, in International Conference on Information Technology in Bio- and Medical Informatics, vol. 9832 (2016)

[6] T. Deepa , Dhanaraj Cheelu, A comparative study of static and dynamic load balancing algorithms in cloud computing, 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)

[7] Hamid Shoja, Hossein Nahid, Reza Azizi, A comparative survey on load balancing algorithms in cloud computing, Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)

[8] Veerawali Behal ,Anil Kumar,Cloud computing: Performance analysis of load balancing algorithms in cloudheterogeneous environment, 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)

[9] Hong Zhu∗ , Hongbo Wang†, and Ian Bayley∗ Formal Analysis of Load Balancing in Microservices with Scenario Calculus, 2018 IEEE 11th International Conference on Cloud Computing