Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 6, Issue 1, 2020
ISSN: 2455 – 7528

Development of Load balancing solution for Real-Time Communication services over AWS

Anjana Sangwan[#1], Dr. Rashid Hussain[#2]

[#1,2]*Suresh Gyan Vihar Univercity,Jaipur,India*
Address

[1]sangwan.anjana@gmail.com

[2]rashid.hussain@mygyanvihar.com

***Abstract*— The Cloud trend has gained remarkable prominence in the tech industry as well as in the field of science in recent years. The most important facet of cloud computing is the on-demand application provisioning paradigm. From the point of view of the cloud customer. This helps Cloud service services to be updated in order to accommodate more new users and to expand while the network has insufficient capacity. The effect is a more effective use of physical system and a cost-saving Cloud system. It is the most up-to-date and common IT and analysis technology with its high features like virtualization and on-demand (dynamic) allocation of resources. To have a stable service for customers with QOS the load balancing mechanism in cloud environments is necessary and an auto-scaling function must also be implemented in conjunction with the application and incoming user distribution in order to prevent overloading and crashing of the network.**

***Keywords*— Cloud, Resources, Client, Virtualization, Load Balancing, Overloading**

I. INTRODUCTION

The load balancing function allows load distribution among one or more cloud systems nodes, allowing the load balancer also to manage the excess load in an efficient service model auto-scaling feature. Auto scaling has dynamically increased and reduced the network, saving money and physical resources according to the customers entering traffic[1]. Latency-based routing is the latest cloud computing idea, which provides load balancing for the global customer based on DNS latency by mapping the domain name system (DNS) across the different host zones. We use public cloud services like Amazons' EC2 in order to achieve the above described. ELB.-ELB. This work is divided into four sections: auto scaling, load balancing, latency routing and resource surveillance. In a simple design, a load balancer or elastic load balancer is used for web servers, which distributes traffic between them. This ensures that the application remains accessible when one of the servers is not functional, the Load Balancer recognizes it and prevents the delivery of traffic to the safe instance [2]

Types of Amazon Elastic Load Balancing

As a result, Elastic Load Balancing distributes traffic inbound applications through various destinations automatically, such as Amazon EC2 instances, containers, and IP addresses [3, 4]. In a single area or in many areas, you can manage the variable load of your application traffic. Elastic Load Balancing provides three load balancing types. These are available, automatically scaled and have robust protection. Everything you need to accept the failure of your applications.

Balancing Elastic load supports three load anchor forms. Load balancers, load balancers network, and classic load balancers program. A load balancer can be selected according to the requirements of your app. In the application layer, the 7th layer of the Open Systems Interconnection Model (OSI), an Application on Load Balancer (ALB) operates. The Load Balancers support content-based and container-based applications [5].

Benefit of AWS Load balancer: for the Amazon, the definition of regions reveals sets of data centers (not just one) that run in union with each other. Amazon has many regions all over the world. There are 20 regions today. The available areas are composed of every zone. A high-tech Data Processing Center with high speed connectivity and a low latency to the rest of the available zones has a similar coverage area [6]. With which services can be built in a country, and the resources we need are available in different places to avoid a reduction in service.

Load balance: AWS has different tools, but load balancers are the most important. The Classic version (kept for previous Balancers only), HTTP (for web server requests) and TCP (to balance any resource) only exist, in three types [7].

Elasticity: There are several con-traits in the AWS case. Firstly, the Cloud Watch Service controls resources. There is a default version, which takes data every 5

minutes and a paid version which takes data every minute, which is capable of performing tasks, creating notifications or changing AWS. The second is that Amazon's management does not have all the tools that are special. They have been aimed at elastic resources,[8] that is to say, we don't build a balance, but we do make an ELB (Elastic Load Balancer): it increases according to load to accommodate all traffic. Therefore, we can build notifications that require instances to start or stop based on traffic, processing and so on.

Simplicity: A large number of services, such as RDS (Relational Databas Service), allow us to ignore the configuration of the physical machine in favor of being able to manage only the database without needing to know anything else. We can optimize the database, create copies in other availability zones or make backups in the famous S3 bucket [9].

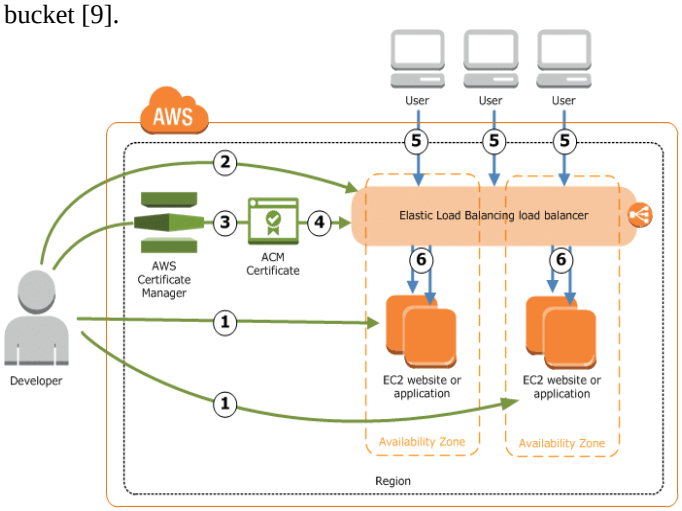

Fig. 1. 01 AWS Load Balancer Configuration

2. Accepted protocols in Amazon Elastic Load Balancing

They endorse many mainstream industry protocols (WebSocket and HTTP 2). They also give additional views of the status and containment of objective instances. The benefits of websites and mobile phones in containers or EC2 cases. This is suitable for advanced load balancing of HTTP and HTTPS traffic with managed application load balancer service. It routing supports modern application architecture, including micro services and applications built on containers[10], in advanced request routing. The Network Load Balancer (NLB) is the other al-ternouncer to deal with tens of millions of requests per second. At a very low latency, it retains high efficiency without any effort. Accepting incoming customer traffic and spreading the traffic within the Availability Zone. Routing of link to destinations: the Amazon EC2 instances, the containers, IP adresses based on IP protocol data. The network is load balancer based at the interface (Layer 4). The Load Balancer Network supports the Balancer API User Load. This provides complete programmatic control of target group and destination. The Load Balancer Network is ideal for managing TCP traffic load. With extremely low latency, NLB is able to process millions of requests per second. NLB is designed to manage traffic conditions that are abrupt and unpredictable. Availability zone uses only one static IP address[11,12].

2.1 Amazon Elastic Load Balancing in legacy mode

Classical Load Balancer (CLB) for multiple Amazon EC2 instances that offers simple load balance and operates both at the request level and the link level. For applications built on the EC2-Classic network an ancient network alternative is planned that is no longer being used, so this choice is not recommended even when applications built in the EC2-Classic network are still in use [13,14].

3. MongoDB load balancing across multiple AWS instances

Amazon web service for a commercial application that uses the node.js server and mongodb as the database. Currently the node.js server is running on a middle EC2 instance. And keep our mongodb database in a separate micro instance. Now we want to deploy a set of replicas to our mongodb database, so if the mongodb crashes or cannot be enabled then we can still run our database and get data from it. So to keep each member of the replica set in separate instances, so that we can fetch data from the database even if the instance of the main memeber goes down[15]. Now, want to add the load balancer to the database, so that the database works well even with high traffic load at once. In that case, I can read the balance of the database by adding the slave OK configuration in the replica set[16]. But it will not balance the load on the database if there is a high traffic load for the write operation on the database.

3.1 To solve this problem, two options so far.

1: Copy the database and keep each fragment in a separate instance. And under each fragment there will be a reapplication set in the same instance. But there is a problem, since the fragment divides the database into several parts, so each fragment will not save the same data within it. So if an instance goes down, we won't be able to access the fragment data within that instance. To solve this problem, I try to divide the database into fragments and each fragment will have a set of replicas in separate instances. So even if an instance goes down, we won't face any problems. But if we have 2 shards and each shard has 3 members in the replica set then I need 6 Aws instances. So I think it is not the optimal solution[17,18].

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 6, Issue 1, 2020
ISSN: 2455 – 7528

2. We can create a master-master configuration in mongodb that means the entire database will be primary and everyone will have read / write access, but I would also like to auto-sync every so often, so everyone they end up being clones of each other. And all of these primary databases will be in a separate instance[19].

4. Implementation of Load balancing

4.1 Load balancing with auto-scaling

In the Load Balancing menu there is a Load Balancers option. In there you must choosethe button create load balancer once inside the assistant is displayed to create letter balancers. You can select between Application Load Balancer and Classic Load Balancer[20]. For our case we are going to use Classic Load Balancer.

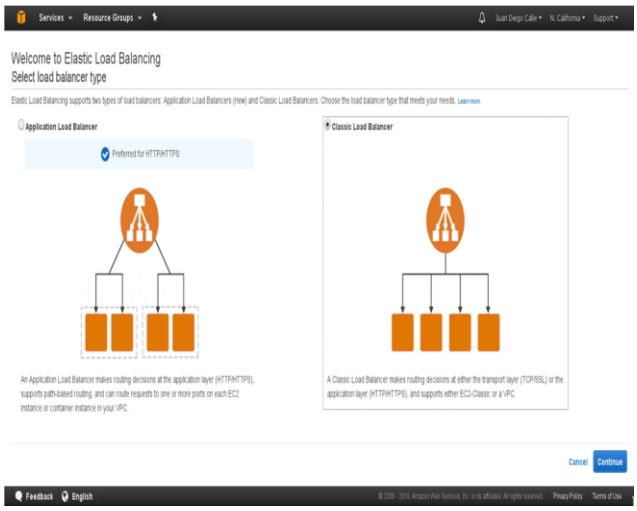

Figure: 02: Types of load balancing.

Then you must select a name for the load balancer and also the ports with their protocols to be used.

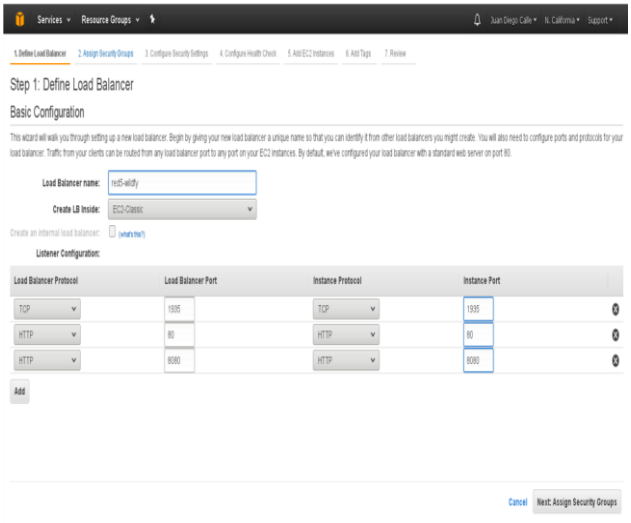

Figure 03: Ports for configuration of load balancer

In the next step you need to add the load balancer to the security group that was created earlier. Or you can create a new group if you want, but you need to have access to the same ports (1935, 80, and 8080).

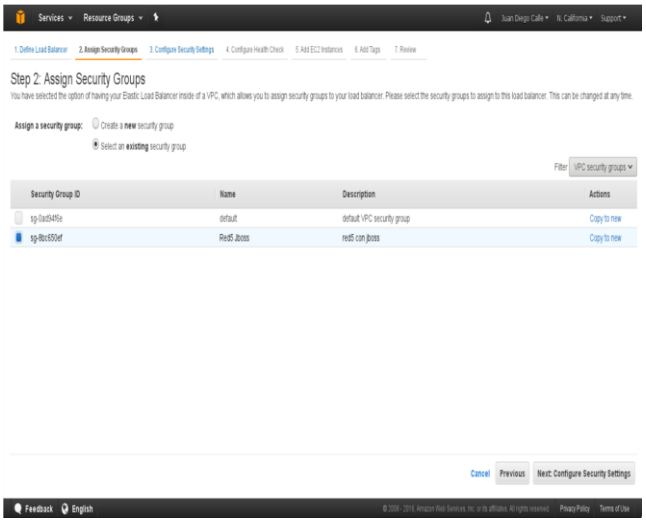

Figure: 04 Safety groups for load balancing.

You can also select the already created instance directly to the load balancer. If necessary, more instances can be added.

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 6, Issue 1, 2020
ISSN: 2455 – 7528

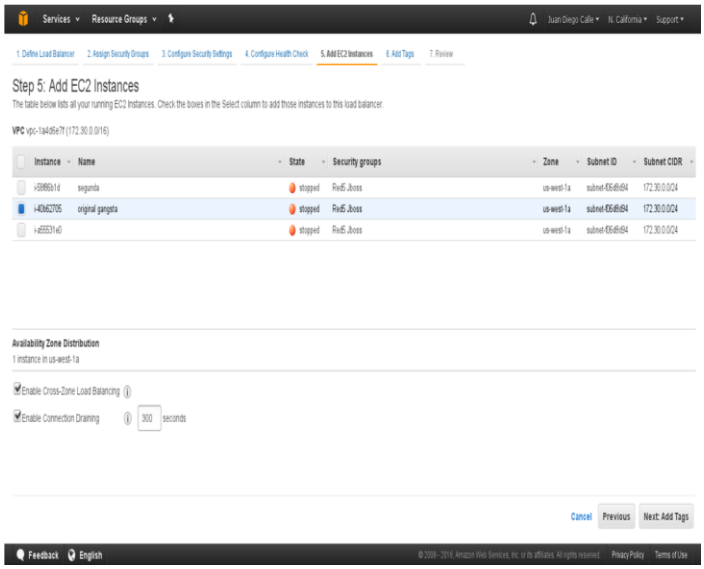

Figure: 05: Add instances directly to the load balancer.

In order to provide a better service that adjusts to the load of the users and allows lower costs using only the resources that are needed, self-escalation policies must be established [21].

4.2 Auto escalation groups

To create an auto scaling group it is necessary to go to the Auto Scaling menu and select Auto Scaling Groups. Inside is the button create launch configuration. Clicking on that button should select it in themenu on the left where it says My AMIs[22] where the created images will be.

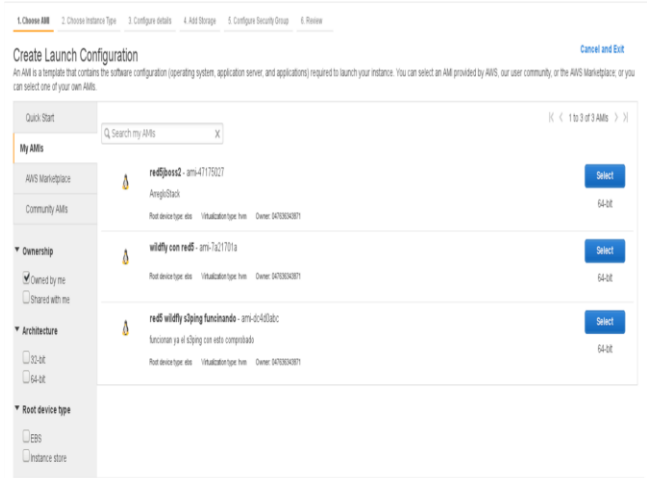

Figure 06: Create Launch Configuration window, My AMIs option.

And then you need to select the type of instance to use, you should use at least one instance similar or with better requirements than the one we use to create the image.Then it is necessary to create self-escalation policies.

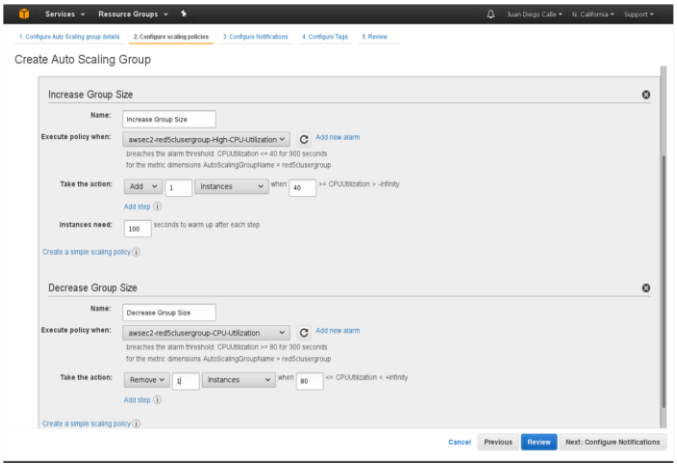

Figure 08: Auto escalation policies

You can choose the CPU or RAM saturation for when to create a new instance and how long the instance is deprecated to shut it down[23].

4.3 Load Computation

Load in Amazon Web Services can vary according to the area and not all types of instances serve everything.

Instance Type	Software	EC2	Total
m1.large	$ 0.23 / hr	$ 0.175 / hr	$ 0.405 / hr
m1.xlarge	$ 0.970 / hr	$ 0.350 / hr	$ 1.320 / hr
m2.xlarge	$ 0.230 / hr	$ 0.2450 / hr	$ 0.4750 / hr
m2.2xlarge	$ 3.760 / hr	$ 0.490 / hr	$ 4.250 / hr
m2.4xlarge	$ 3.760 / hr	$ 0.980 / hr	$ 4.740/ hr
m3.large	$ 0.230 / hr	$ 0.1330 / hr	$ 0.3630/ hr
m3.xlarge	$ 0.460 / hr	$ 0.2660 / hr	$ 0.7260 / hr
c1.xlarge	$ 0.890 / hr	$ 0.520 / hr	$ 1.410 / hr

Conclusions

This work focused on the solution by the server to obtain a high availability solution achieved the general objective of studying the load balancing of a free software system for high availability streaming using cluster with Red5 in the

cloud with Amazon Web Services (AWS). Implementation of the proposal is the main contribution of this work due to the adaptations that had to be made and that leave the way open for new research. It is possible to create a high availability streaming with high speed with JBoss clusters with Red5 that can be applied for video and audio essentially although it could be used for more complex programs that use real time. In this sense, for subsequent studies, the client side could be considered, which will surely enrich the solutions when carrying out their own projects, because of the topic discussed, it is open for further study.

References

[1.] Pradhan P, Behera PK, Ray BNB (2016) Modified round Robin algorithm for resource allocation in cloud computing. Proced Comp Sci 85:878–890

[2.] Mishra SK, Sahoo B, Parida PP (2018) Load balancing in cloud computing: a big picture. J King Saud Univ Comp Infor Sci:1–32

[9.] Alouane M, El Bakkali H (2016, May) Virtualization in cloud computing: no hype vs HyperWall new approach. In: 2016 International Conference on Electrical and Information Technologies (ICEIT), pp 49–54

[10.] Rimal BP, Choi E, Lumb I (2009, August) A taxonomy and survey of cloud computing systems. In: Fifth international joint conference on INC, IMS and IDC, 2009. NCM'09, pp 44–51

[11.] Afzal S, Kavitha G (2018, December) Optimization of task migration cost in infrastructure cloud computing using IMDLB algorithm. In: 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), pp 1–6

[12.] Achar R, Thilagam PS, Soans N, Vikyath PV, Rao S, Vijeth AM (2013, December) Load balancing in cloud based on live migration of virtual machines. In: 2013 annual IEEE India Conference (INDICON), pp 1–5

[13.] Magalhães D, Calheiros RN, Buyya R, Gomes DG (2015) Workload modeling for resource usage analysis and simulation in cloud computing. Comp Elect Eng 47:69–81

[14.] Dam S, Mandal G, Dasgupta K, Dutta P (2015, February) Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing. In: Proceedings of the 2015 third international conference on computer, communication, control and information technology (C3IT), pp 1–7

[15.] Dave A, Patel B, Bhatt G (2016, October) Load balancing in cloud computing using optimization techniques: a study. In: International Conference on Communication and Electronics Systems (ICCES), pp 1–6

[16.] Gupta H, Sahu K (2014) Honey bee behavior based load balancing of tasks in cloud computing. Int J Sci Res 3(6)

[17.] Mishra SK, Puthal D, Sahoo B, Jena SK, Obaidat MS (2017) An adaptive task allocation technique for green cloud computing. J Supercomp 405:1–16

[18.] Ibrahim AH, Faheem HEDM, Mahdy YB, Hedar AR (2016) Resource allocation algorithm for GPUs in a private cloud. Int J Cloud Comp 5(1–2):45–56

[19.] Jebalia M, Ben Letafa A, Hamdi M, Tabbane S (2015) An overview on coalitional game-theoretic approaches for resource allocation in cloud computing architectures. Int J Cloud Comp 4(1):63–77

[20.] Noshy M, Ibrahim A, Ali HA (2018) Optimization of live virtual machine migration in cloud computing: a survey and future directions. J Netw Comput Appl:1–10

[21.] Gkatzikis L, Koutsopoulos I (2013) Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems. IEEE Wirel Commun 20(3):24–32

[22.] Jamshidi P, Ahmad A, Pahl C (2013) Cloud migration research: a systematic review. IEEE Trans Cloud Comp 1(2):142–157

[23.] Kanakala VR, Reddy VK, Karthik K (2015, March) Performance analysis of load balancing techniques in cloud computing environment. In: 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp 1–6

[3.] Reddy VK, Rao BT, Reddy LSS (2011) Research issues in cloud computing. Glob J Comp Sci Technol 11(11):70–76

[4.] Bohn RB, Messina J, Liu F, Tong J, Mao J (2011) NIST cloud computing reference architecture. In: Proceedings of IEEE 7th world congress on services (SERVICES'11), Washington, DC, USA, Jul. 2011, pp 594–596

[5.] Bokhari MU, Shallal QM, Tamandani YK (2016, March) Cloud computing service models: a comparative study. In: 3rd international conference on computing for sustainable global development (INDIACom), 16–18, March 2016, pp 890–895

[6.] Mahmood Z (2011, August) Cloud computing: characteristics and deployment approaches. In: 2011 IEEE 11th international conference on Computer and Information Technology (CIT), pp 121–126

[7.] Buyya R, Vecchiola C, Selvi ST (2013) Mastering cloud computing: foundations and applications programming. Morgan Kaufmann, USA, 2013

[8.] Jain N, Choudhary S (2016, March) Overview of virtualization in cloud computing. In: Symposium on colossal data analysis and networking (CDAN), pp 1–4