Comparative Study on Load Balancing Google Cloud Platform vs. Amazon Web Service *

Piyush Thummar
Master of Applied Computer Science
Concordia University
Montreal, QC, Canada
piyushthummar305@gmail.com

Tirth Shah
Master of Applied Computer Science
Concordia University
Montreal, QC, Canada
tirthshah510@gmail.com

Jasmine Kaur
Master of Applied Computer Science
Concordia University
Montreal, QC, Canada
jasmine14concordia@gmail.com

ABSTRACT

As the traffic volume and client requests surge over the electronic highway, www sites and other software applications, the need for flexible, secure networks and applications becomes indispensable. Load Balancing promises to overcome these issues. The load balancers ameliorate responsiveness and increase the availability of applications. They ensure efficient information transfer over the networks and optimize application delivery and usage. In this study, we have tried to delineate and implement the techniques/algorithms used for balancing the load on Google Cloud Platform (GCP) and Amazon Web Service (AWS) and analyze their performance. Lastly, we'll be enumerating a few visualizations depicting the comparison study.

Keywords: Load Balancer, Application, GCP, AWS, Siege, Httperf.

1. INTRODUCTION

In general,*"Load Balancing"* can be viewed as efficient distribution of workloads among multiple computing resources. The purpose of a load balancer is to use multiple components or resources to avoid overload of any single resource. Load balancing aims to optimize resource utilization hence providing maximum throughput in minimum response time. Every cloud platform uses a software load balancer for distributing incoming client requests or traffic. But, all make use of different algorithms and have named and categorized load balancers following their own criteria. The idea of a comparative study is to scrutinize, relate and compare load management, efficiency of servers and these classifications of load balancers on Google Cloud Platforms and Amazon Web Service by implementing virtual machine configurations and software load balancers provided by them and representing the statistical observations like Response Time, Latency, bandwidth, throughput and more.

*World's most broadly adopted and comprehensive cloud platforms

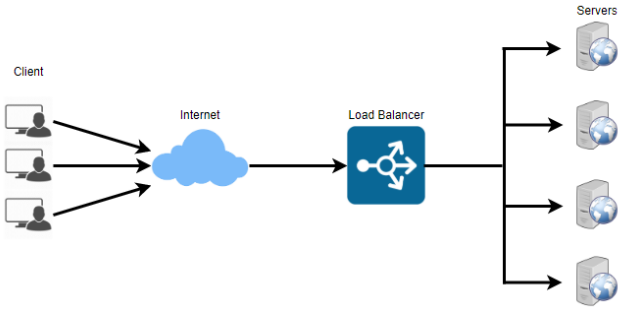

Figure 1: Load Balancing

Load Balancing can be done on the overloaded Central Processing Units, disk drives, applications, network links or computer clusters. Load balancers control the flow of data between the server and an endpoint device such as a PC, laptop, smartphone, etc. The server could be on-premises, in a data center or the public cloud.

1.1 Google Cloud Platform:

Cloud Load Balancing can put the resources behind a single anycast IP and scale resources up or down with intelligent autoscaling. Cloud Load Balancing comes in a variety of flavors and is integrated with Cloud CDN for optimal application and content delivery. With Cloud Load Balancing, a single anycast IP front-ends all your backend instances in regions around the world. It provides cross-region load balancing, including automatic multi-region failover, which gently moves traffic in fractions if backends become unhealthy. Cloud Load Balancing is built on the same frontend-serving infrastructure that powers Google.

The load balancer of GCP can be distinguished into the external and internal load balancer [4]. The traffic coming from the internet to the GCP network distributed by external load balancer whereas traffic within the GCP network handled by the internal load balancer. In the selection of the type of a load balancer for the system, monitor the type the traffic is a must. HTTP and HTTPS traffic can be handled by external HTTP(S) or Internal HTTP(S) Load Balancing. TCP/UDP traffic can be handled by Network Load

Balancing or Internal TCP/UDP Load Balancing. For this study, we are implementing HTTP(S) load balancer which supports HTTP traffic on TCP port 80 or 8080 and HTTPS traffic on port 443.

We are using Premium Tier of Network Service Tiers for the HTTP(S) load balancing which offers more benefits than the Standard Tier. It balances requests among backends in multiple regions by directing requests to the region closest to the user. If that region is at capacity, the load balancer uses a waterfall-by-region overflow model to deliver requests to the next closest region. It uses a single Anycast IPv4 or IPv6 load balancer address so that we can utilize a single DNS record, worldwide.

The internal HTTP(S) load balancer performs proxy-based load balancing of Layer 7 application data which we mention with URL maps. It uses a private IP address that acts as a frontend to your backend instances. Internal HTTP(S) Load Balancing comes with the functionalities of Health checks, Autoscaling without prewarming and Regional load balancing.

1.2 Amazon Web Service:

Amazon Web Service handles the load by Elastic Load Balancing [2]. It automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault-tolerant. For this study, we are using an application load balancer.

Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and container-based applications. Operating at the individual request level (Layer 7), Application Load Balancer routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) based on the content of the request. Application Load Balancer improves the security of the application, by ensuring that the latest SSL/TLS ciphers and protocols are used at all times. For the load balance of HTTP/HTTPS application, it uses layer 7- specific features, such as X-Forwarded-For headers.

We can load balance any application hosted in AWS or on-premises using IP addresses of the application backends as targets. This allows load balancing to an application backend hosted on any IP address and any interface on an instance. An Application Load Balancer requires you to specify more than one Availability Zone. We can distribute incoming traffic across your targets in multiple Availability Zones. We can use the Access Logs feature to record all requests sent to our load balancer and store the logs in Amazon S3.

Also, we can use AWS WAF to protect web applications on Application Load Balancers. AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources.

2. RELATED WORK

Several alternative studies for comparison of load balancing algorithms have been proposed. Almost all of them are extended to a comparison of algorithms. However, they have never explicitly covered or evaluated the application outcomes of their utilization.

In [6], The purpose of this study is to compare different load balancing algorithms based on identified qualitative parameters. It carried out the analysis of different load balancing algorithms, various parameters are used to check the results. This comparison shows that static load balancing algorithms are more stable than dynamic. But dynamic load balancing algorithms are always better than static as per overload rejection, reliability, adaptability, cooperativeness, fault-tolerant, resource utilization, response and waiting time and throughput is a concert.

Another VM Load Balancing Algorithm is proposed in [5]. It is implemented in the Cloud Computing environment using the CloudSim toolkit, in java language. In this algorithm, the VM assigns a distinct amount of processing power to the individual application services. These VMs of different processing powers, the requests are assigned or allocated to the VM which has the highest processing power and then to the lowest and so on. Hence the study optimized the given performance parameters such as response time and data processing time, giving an efficient VM Load Balancing algorithm i.e. Weighted Active Load Balancing Algorithm in the Cloud Computing environment.

In the practice of [7], load balancing algorithms work on the principle that in which situation workload is assigned, during compile time or at runtime. The comparison in this study shows that static load balancing algorithms are more reliable in comparison to dynamic and it is also easy to predict the behavior of static, but at the same time, dynamic distributed algorithms are always considered better than static algorithms. The dispatcher based approach is a centralized scheduling method in which the dispatcher has the overall control over the balancing process and achieves a fine-grained balancing. In the study, they simulated four different dispatcher-based scheduling algorithms: random, round-robin, least connection scheduling and the least loaded server. The random algorithm performs the worst in some of the conditions but at others, it seemed to work well. The random algorithm gives unpredictable results.

All the existing studies compare the load balancing algorithm but they have never shown different scenarios in which, how a particular load balancer would behave. Load balancing could be implemented in a single region and in multiple regions. Based on that, the comparison could be possible among them as well as we can monitor the performance of the server having a load balancer in the communication and without the load balancer.

In this study, we are going to create an environment in which, we can possibly get the performance efficiency of the servers in different conditions and can do a comparison between Google Cloud Platform and Amazon Web Service.

3. PROPOSED SOLUTION

We have a myriad of cloud platforms performing similar services but in distinct ways. The key point of comparison between these cloud platforms that we are studying is load balancing. This study will give insights about the performance of these platforms in different scenarios. We can perform the load test on the servers and evaluate the performance of the servers based on the parameters such as bandwidth, latency and throughput. To make a fair comparison, we need to create the same testing environment for all the cloud platforms. The key point of comparison will be their load balancing capabilities under extreme conditions. The obtained results can then be evaluated and studied to know the strengths and weaknesses of each cloud platform, and perhaps think about potential solutions to solve the arising problems.

To begin with the comparison, we first need to have a service on each of these cloud platforms that users will interact with. Further, this interaction will be tested under the following two conditions:

1) Without load balancer in cloud platforms.
2) With load balancer and servers possibly distributed across multiple zones.

This will give a detailed overview of how each of these cloud services performs under different conditions.

We have created a testing environment for Google Cloud Platform and Amazon Web Service as follows:

3.1 Google Cloud Platform:

First, set the default to compute engine zone, for the future actions and set this as an environment variable for future use. Then create and configure Virtual Private Cloud (VPC). VPC gives us the flexibility to scale and control how workloads connect regionally and globally. With the usage of VPC, we have global access without needing the replication of connectivity and administrative policies in each region whereas in AWS we do not have global access across all regions. Post VPC creation, we need to create a firewall for further communication.

Google Cloud Platform (GCP) firewall rules let us allow or deny traffic to and from your virtual machine (VM) instances based on a configuration we specify. Enabled GCP firewall rules are always enforced, protecting our instances regardless of their configuration and operating system, even if they have not booted up. For this study, we allow internal and SSH traffic to communicate with the VPC network.

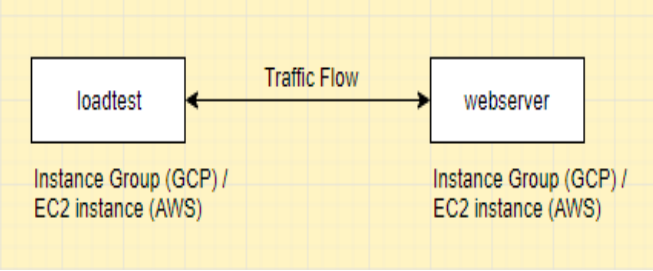

Figure 2: Without load balancer

For the testing on the VM instances, we have set up the load test VM instance in the same zone, environment, and configuration as the first VM instance. With the help of this VM instance, we measure the performance of VM instances and load balancers using performance measurement tools. These measurement tools help to define the correct load balancing capacity setting for the instance group.

Set up the webserver VM instance as a 4-core VM instance. We have set up machine type for this VM instance is n1- highcpu-4 and image family is debian-10. Make it a template for the future creation of servers.

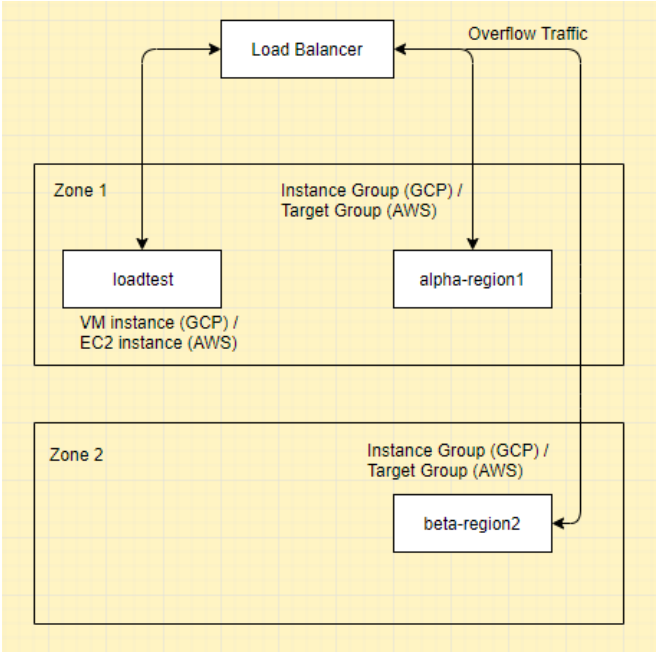

Figure 3: With load balancer and servers possibly distributed across multiple zones

For measuring overload effects with a different region we have to establish a global application with HTTP(S) Load Balancing and should have backend deployed in multiple regions. When a capacity overload occurs in a single region, traffic automatically flows to another region. We can validate this by adding a second VM instance group in another region and configure it with the load balancer.

Creation of backend service for load balancer: include the instance groups on which load balancer will redirect the load given by the clients. A health check is necessary to set up because it ensures that requests are sent only to instances that are up and running. For HTTP load balancer the suitable protocol for a health check is HTTP. In health check, we have set the health check interval of 60 seconds. As the load will be generated from the same machine only, specify a port tag as 80.

For the testing purpose, this VM instance uses a Python script to create a CPU-intensive task by calculating and displaying a picture of the Mandelbrot set on each request to the root (/) path. The result is not cached. During the tutorial, the system gets the Python script from the GitHub repository used for this solution [1].

3.2 Amazon Web Service:

Firstly, we need to create a Virtual Private Cloud (VPC) for communication among the cloud platform. For communication with the servers, we have to create at least one subnet per availability zone. Then, we can create a route table to redirect the incoming traffic from VPC to the servers as per their availability and health that is followed by making internet gateway, which works as the bridge between two networks to redirect the incoming traffic from clients. And then, the security group and ACL (Network Access Control List) is being set up for the additional security against the incoming traffic as per the requirement, but for this study, we only performed basic and default set-ups. After this, we can create the servers under that VPC across different availability zones before we create the load balancer.

For the configuration of the load balancer, include the target groups on which load balancer will redirect the load given by the clients. The health check is necessary to set up because it ensures that requests are sent only to instances that are up and running. In health check, we have set the health check interval of 60 seconds. Specify the port tag with 80 because we are going to apply the load from our own machine.

For setting up the servers, configure ec2 linux instances with Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-00bf61217e296b409 (64-bit x86) / ami-0c3e7919416671c83 (64-bit Arm) type. After creation of instances, register these instances with the VPC. We have set up the ubuntu ec2 instance with Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami- 0d5d9d301c853a04a (64-bit x86) / ami-0fb0129cd568fe35f (64-bit Arm) type. After this, we have to install siege and httperf tools in this instance to generate the load and test the load balancer.

We have set up the ubuntu ec2 instance with Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0d5d9d301c853a04a (64-bit x86) / ami-0fb0129cd568fe35f (64-bit Arm) type. After this, we have to install siege and httperf tools in this instance to generate the load and test the load balancer.

We start measuring network capacity using load-testing tools (siege and httperf).

Siege:
Siege is an open-source regression test and benchmark utility. It can stress test a single URL with a user-defined number of simulated users. Siege allows you to load a web server with n number of users t number of times, where n and t are defined by the user. It records the duration time of the test as well as the duration of every single transaction. It reports the number of transactions, elapsed time, bytes transferred, response time, transaction rate, concurrency and the number of times the server responded OK, that is status code 200.

Httperf:
Httperf is a tool for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. The focus of httperf is not on implementing one particular benchmark but on providing a robust, high-performance tool that facilitates the construction of both micro- and macro-level benchmarks. The three distinguishing characteristics of httperf are its robustness, which includes the ability to generate and sustain server overload, support for the HTTP/1.1 and SSL protocols, and its extensibility to new workload generators and performance measurements.

After performing the load test on the servers, we have required attributes of all three scenarios for both GCP and AWS. We will compare these outcomes and conclude the efficient cloud platform in the aspect of load management.

4. EXPERIMENTAL EVALUATION

We have created the same kind of environment and executed similar commands on both platforms to get performance parameters. With the use of the siege tool, we get the response time, transaction rate, throughput and latency of the server.

The first attribute that we used for comparison is the response time. Response time is the total amount of time it takes to respond to a request for service. Here, the ideal case is one with minimum response time, the system is that much faster to send a response. It ultimately depicts that users will not have to wait for long to get the response.

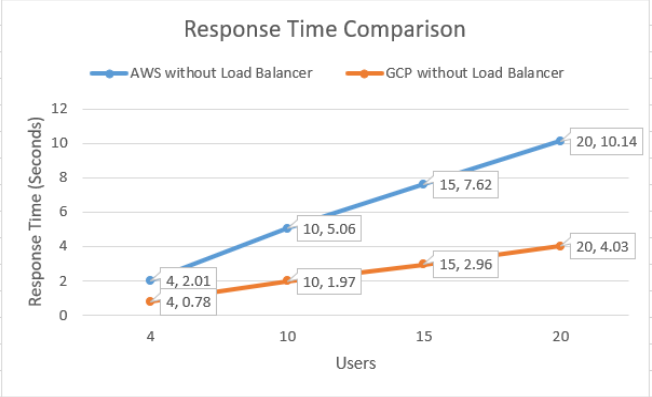

Figure 4: Response Time Comparison - Without Load Balancer

As mention in the solution, we have measured the performance of the server with or without having the role of a load balancer. Figure 4 indicates the outcomes of the response time for GCP and AWS without setting up the load balancer. We have observed that for 4 concurrent users the response time in GCP is 0.78 seconds whereas for AWS it was 2.01 seconds. As the number of users increases, the response is also increased gradually. Servers are performing well in GCP as compared to AWS. However, we can improve this outcome by having a load balancer. The increment in the response time is much faster in AWS than GCP. The performance of AWs degrades rapidly than the GCP.

Figure 5 indicates the response time of the server in the presence of a load balancer. A load balancer distributes the traffic evenly so that we have better results for the response time in both the platforms. Specifically, it can be said that for all the times, the response time for the GCP load balancer is always less than the AWS load balancer. From the graph, it can be illustrated that the GCP load balancer is faster than the AWS load balancer. It can be stated that for 4 users, the GCP load balancer has 0.46 seconds of response time, which is lesser than 0.66 seconds of AWS. the difference between both types is increasing gradually. For 20 users, the difference is almost 0.45 seconds; where, the

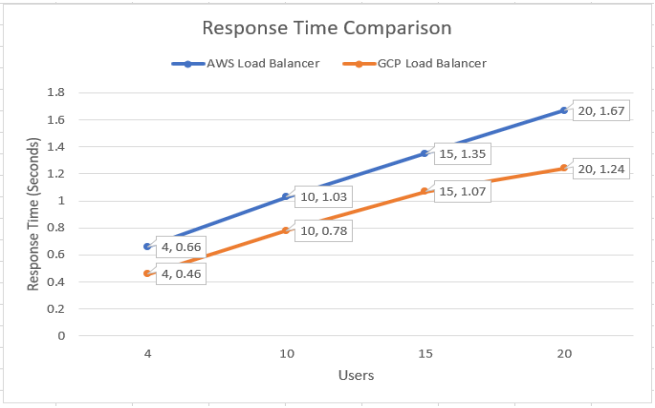

Figure 5: Response Time Comparison - With Load Balancer

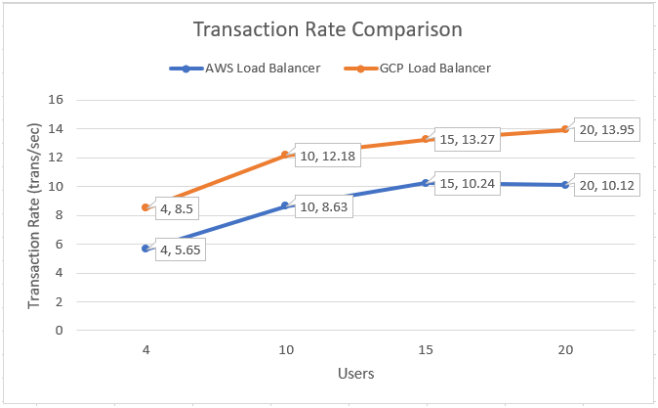

Figure 7: Transaction Rate Comparison - With Load Balancer

response time for the AWS load balancer is 1.67 seconds, and for the GCP load balancer, it is 1.24 seconds, which is lesser than the other one. Overall, the increase in response time for the GCP load balancer is lesser than the AWS load balancer, which is 0.46 to 1.24 and 0.66 to 1.67 accordingly. In a nutshell, for both, the response time is increased as per the rise in several users, but the rise for AWS load balancer is higher than the GCP load balancer.

The second attribute to compare the platforms is the Transaction rate. The transaction rate is the data transfer per second from servers of the load balancer to requesters. It is a fact that the maximum transaction rate leads to better system performance. This transaction rate is measured in transaction/sec, specifically for the given application.

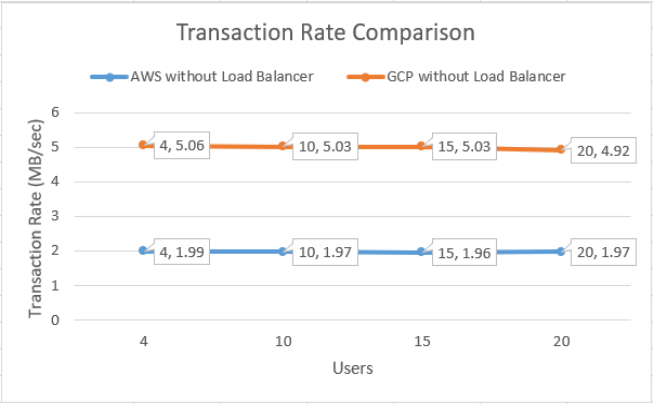

Figure 6: Transaction Rate Comparison - Without Load Balancer

Similar kind of environment we have set up for the examination of the transaction rate. As we can see from figure 6, the transaction rate in the AWS for a different number of users is quite constant. But in GCP, there are slightly better results of the transaction rate. In GCP transaction rate for 4 concurrent users is transaction/second which decreased to 4.92 transaction/second for 20 concurrent users.

After the load balancer comes into action, the transaction rate improved for both the platforms. Transaction improves gradually as the number of users increases. As figure 7 depicts, the transaction rate for the AWS load balancer is 5.65 transactions/second, and for GCP load balancer is 8.5 transactions/second in the scenario of 4 users. Overall, it can be illustrated that the transaction rate for both the platforms increases as the user count increases. Also, the increasing ratio for both platforms is almost the same. In addition to this, the transaction rate for AWS load balancer is increased from 5.65 to 10.12, which has risen almost 4.5 transactions per second for AWS, that is lesser than the rise of GCP load balancer from 8.5 to 13.95 transactions/sec, that means the rise is almost 5.5 transactions per second. It can be generalized that the transaction rate for the GCP load balancer is always higher than the AWS load balancer for the given scenario, which depicts that the GCP load balancer is again stronger than the AWS load balancer for the comparison of attribute transaction rate.

The third attribute that is considered for this comparative study is Throughput. Throughput is the average amount of data that is transferred during the transactions in our case study. Here, throughput is measured in KB/sec (kbps). As maximum as the throughput is, the system is considered that much better or stronger. For our study, we selected this attribute because we are using the application which involves the data transfer. Also, throughput is one of the principal attributes in the load balancer.

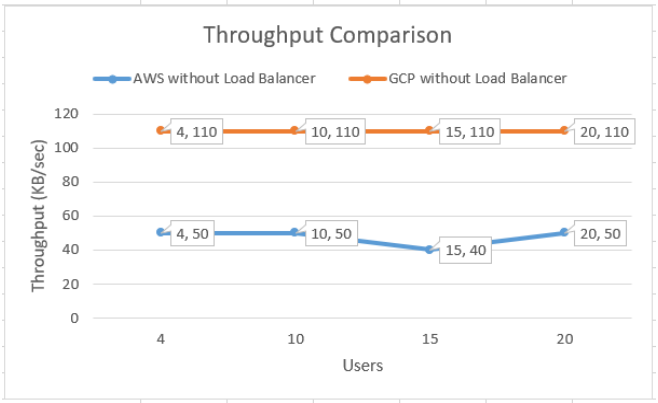

Figure 8: Throughput Comparison - Without Load Balancer

In figure 8, we can see that without having a configuration

of a load balancer, the throughput of the servers for both platforms is not good as it would be in the presence of the load balancer. For GCP, the throughput seems constant (110 kb/s) for all the number of users whereas in AWS it is 50kb/s. there is a minor decrement in the throughput (40 kb/s) of the EC2 instance when the concurrent user is 15. We can see the refinement in the throughput by having a load balancer in the network.

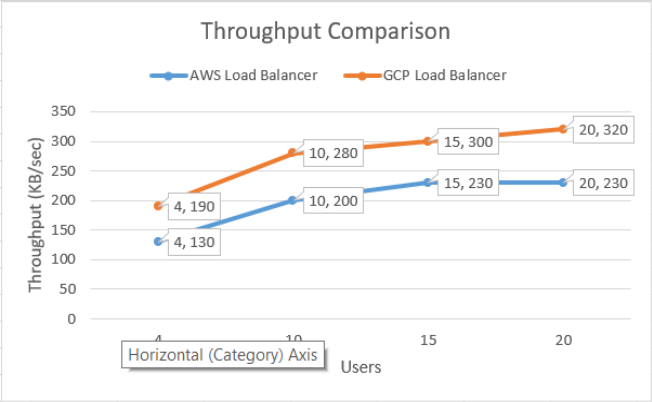

Figure 9: Throughput Comparison - With Load Balancer

It is shown from the figure 9 that with the increasing number of users, throughput increases for both of the platforms GCP and AWS. For 4 users, the AWS load balancer shown the throughput of 130 kbps; whereas, the GCP load balancer is efficient than the AWS load balancer with the throughput of 190 kbps. As the number of users increases from 4 to 20, the rise in both the platforms can be seen, which are 190 kbps to 320 kbps and 130 kbps to 230 kbps for the GCP load balancer and AWS load balancer accordingly. However, the rise of GCP load balancer throughput with the increasing users is higher than AWS load balancer throughput, the difference is almost 30 kbps. The difference between throughputs of both the platforms with the growing number of users is almost the same for all the given scenarios, which is 60 kbps for 4 users and 90 kbps for 20 users.

The latency is considered as the fourth attribute for comparing two platforms for load balancing. The latency is the time that is spent on transferring the response. That means, if processing time is subtracted from the response time, what we get is called the latency. Here, processing time indicates the time to process the request. As similar to response time, as minimum as the latency is, the better the system is. In other words, with the minimum latency, the system is considered to be better than the system with higher latency [3]. Latency can be computed or measured in milliseconds or seconds. For this study, latency is measured in seconds. The latency is measured with several hits that are 500 requests per loop for this study.

The following command is used for detection of the latency for VM instance in the GCP and EC2 instance in AWS. In command "webserver indicates the name of the server.

for ((i=0;i¡500;i++)); do curl -w "time$_t$otal" / $-$ o/dev/null $-$ s webserver; **done**

Whereas, for measuring latency in the presence of load balancer, we have used this command. In which, web-map indicates a load balancer.

for ((i=0;i¡500;i++)); do curl -w "time$_t$otal" / $-$ o/dev/null $-$ sweb $-$ map; **done**

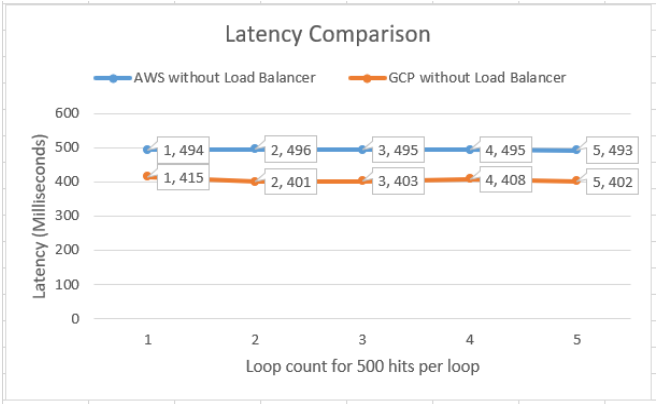

Figure 10: Latency Comparison - Without Load Balancer

Figure 10 indicates the latency of the network for both cloud platforms in the absence of a load balancer in the network. We can observe the better latency in GCP as compared to AWS. We have used a loop to test the latency of the network. So in the first loop, the latency for GCP is 415 milliseconds(ms) which decreased gradually and become 402 ms. The same behavior is shown by the AWS. Firstly it is 494 ms and slightly reduced on the last loop. In between that, it has increased to 496 ms.

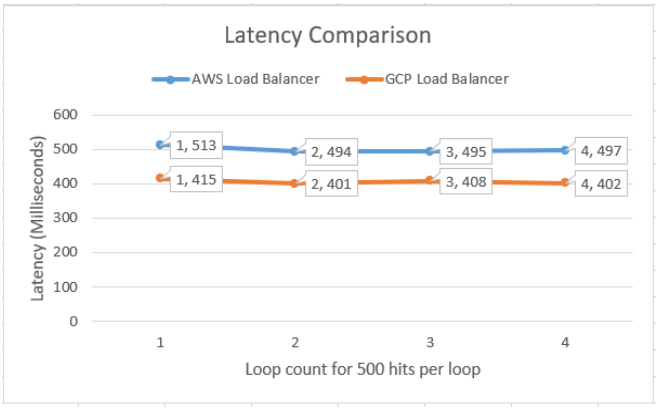

Figure 11: Latency Comparison - With Load Balancer

As the figure 11 illustrates, it can be generalized that latency for both the platforms has been almost the same as the loop increases. Initially, latency for the AWS load balancer is 513 ms, which is higher than the latency of the GCP load balancer that is 415 ms. This depicts the GCP load balancer again beats the AWS load balancer in the comparison of attributes. For continuous 4 loops, it can be seen from the chart that latency is almost the same. However, the difference between the latency of the GCP load balancer and the AWS load balancer is from 90 ms to 100 ms. For all the 4 loops, the latency of the AWS load balancer has remained

almost the same as 490 ms to 510 ms; whereas, the latency for GCP load balancer has remained near 400 ms to 410 ms.

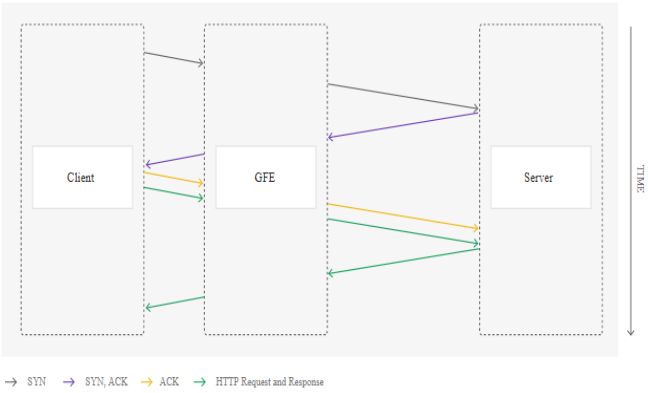

Figure 12: Initial HTTP request via GFE (Google Front End)

In terms of latency, GCP outperformed AWS. When a user makes an HTTP request, unless load balancing is configured, the traffic flows directly from the users network to the virtual machine (VM) hosted on Google Compute Engine. Traffic then enters Google's network at an edge point of presence (POP) close to the user's location. When pinging the VM IP address, the response comes directly from the web server. This difference is because a new TCP connection is opened for every HTTP request, and an initial three-way handshake is needed before the HTTP response is sent.

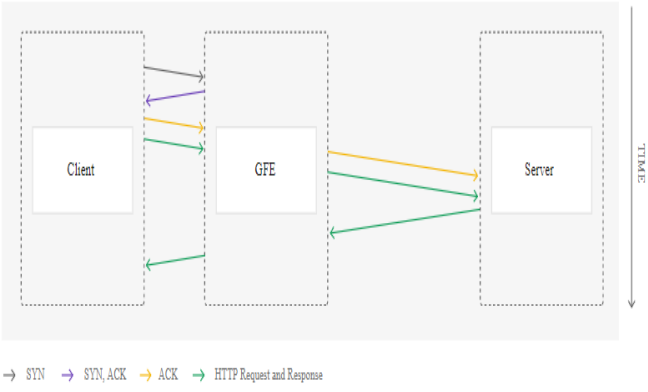

Figure 13: Subsequent HTTP request via GFE (Google Front End)

With HTTP Load Balancing, traffic is proxied through GFEs, which are typically located at the edge of Google's global network. The GFE terminates the TCP session and connects to a backend in the closest region that can serve the traffic. The reason for the lower latency is that traffic is proxied through GFEs, which keep persistent connections open to the backend VMs. Therefore, only the first request from a specific GFE to a specific backend needs a three-way handshake.

5. FUTURE WORK

In this study, we have not covered all the functionalities of GCP and AWS for comparison. For the enhancement of this study, we would consider network security measures of both platforms. Any developer who wants to host a system on the cloud platform could benefit from our study. One big open question is to find an even better way to perform the above tests. For the load test, the environment which has the role of load balancer when the load limit exceeds on the server. At the moment of an overflow of requests, a load balancer comes into action and keeps a balance of requests among all the servers.

Hence, it might improve the efficiency of the servers. In this paper, we focused only on a comparison of GCP and AWS. Microsoft Azure is also one of the prominent cloud platforms. For further study, we would consider Microsoft Azure for the various examination of load management.

6. CONCLUSION

In conclusion, it can be evaluated that after observing and analyzing all the four attributes and its values for different scenarios, GCP's load balancer is better than the AWS load balancer. To sum up the experimental evaluation, the GCP load balancer has been proven to be better than the AWS load balancer in the comparison of cardinal attributes, which are Response Time, Throughput, Transaction Rate and Latency.

7. REFERENCES

[1] Optimizing Application Capacity with Global Load Balancing. `https://github.com/GoogleCloudPlatform/lb-app-capacity-tutorial-python#optimizing-application-capacity-with-global-load-balancing`, 2018. [Online].

[2] AWS. Elastic Load Balancing. `https://aws.amazon.com/elasticloadbalancing/`. [Online].

[3] Google. Optimizing application latency with load balancing. . `https://cloud.google.com/load-balancing/docs/tutorials/optimize-app-latency`.

[4] Google. Google Cloud Load Balancing. `https://cloud.google.com/load-balancing`, 2019. [Online].

[5] J. James and B. Verma. Efficient vm load balancing algorithm for a cloud computing environment. *International Journal on Computer Science and Engineering*, 4(9):1658, 2012.

[6] A. A. Rajguru and S. Apte. A comparative performance analysis of load balancing algorithms in distributed system using qualitative parameters.

[7] J. Vashistha and A. K. Jayswal. Comparative study of load balancing algorithms. *IOSR Journal of Engineering*, 3:45–50, 2013.