**Load balancer in HTTP traffic**

# Shashank Sharma\*1, Ms. Shagufta 2

\*1UG Student of Bachelor of Computer Applications, Shri Ramswaroop Memorial College of Management, Lucknow, Uttar Pradesh, India.

\*2Assistant Professor, Department of Bachelor of Computer Applications, Shri Ramswaroop Memorial College of Management Lucknow, Uttar Pradesh, India.

[shashanksharma500v@gmail.com](mailto:shashanksharma500v@gmail.com)

**Abstract:**

In this study, we explore diverse methodologies and advancements within the realm of load balancer using amazon web services.This investigation examines methods for increasing the speed and reliability of websites under a heavy load. Various strategies, such as Round Robin, Least Connections, and IP Hash, will be considered in terms of their effectiveness at directing requests to appropriate servers. Additionally, we shall ascertain the wellbeing of these machines so that all things can run well. In addition to this, we want to find out how users can stay connected when moving from one page to another within the same domain without having to re-enter their login credentials every time. The intention is that through practical testing coupled with performance analysis; useful recommendations on enhancing website quality while ensuring its stability shall be provided

**Introduction:**

In today’s fast-paced online world, where the speed of websites is pivotal and they have to be able to accommodate a huge amount of users, good load balancing becomes a highly valuable feature. Nowadays, as more people consume and generate content on the Web or in the cloud, the responsiveness and reliability of websites becomes a rather crucial point. Load balancing is dividing all incoming requests to a website equally among different servers to improve its performance and reliability, thus keeping users happy. This project will delve into this by looking at how to accomplish this specifically for handling web traffic; we'll inspect how to balance the load between web servers using methods like Round Robin, Least Connections, and IP Hash; and we'll make sure to check in on the web servers to see how they're doing and to adjust the load accordingly. Furthermore, users on our site can be automatically logged in without having to re-enter their password as they navigate around the site. By running these little experiments and seeing how much effectiveness increases through these little attentions, we can get some very practical tips for improving websites and making users happy.

**Literature Review:**

Load balancing is quite a hot topic these days with our ever increasing expectations from the web. We all want our web applications to be lightning speed and scale well. I will be listing below what some researchers around the world have been saying with respect to different policies. Some policies followed are: Round Robin, Least Connections, and IP Hash. Round Robin: It distributes the requests in the order they arrive to the servers in a circular order. So request1 gets forwarded to the server 1 and request 2 gets forwarded to the server 2 and when request N comes in, it goes to server N, server 1 when N is multiple of no. of servers and so on.. Least Connections: It routes the request to the server with least established connections. And, IP Hash: It routes to a specific server based on the source IP.

Dynamic methods like Weighted Round Robin and Least Response Time modify server selection on the fly depending on current server states. They account for attributes like server health, server load, and response times in their selections. This is a broad area that impacts the overall system performance and availability, and needs to be used to the best effect. Server health monitoring is a vital area to ensure the system remains reliable at all times. This is primarily done through active checks, which fire requests to servers at a particular interval and verify their health (whether the server responds or not) and passive checks, which monitor stats like the number of USB clients, CPU utilization, memory usage, network latency, so on so forth. Efficient health monitoring may well prevent a potential issue and does a good job at keeping the system up and running most of the times.

Session Persistence This is about making sure that a user remains connected to the same server throughout the duration of their visit to the site. This is generally practised by using cookies or tracking IP addresses. It is important to make sure that users do not lose their session data when moving around a site. This means that when a user visits a site, finds something interesting and tries to share it with someone else the link should hopefully still work. To test different techniques a whole variety of metrics could be considered: How much data can be transferred between the user and the server. How long does it take for the server to process the request and return the right information. How well is the server being utilised. There is a whole branch of research simply designed to answer these questions and then some. A simple google search should return a number of academic papers and other reading on the topic. It takes a lot of work to figure out what the best technique is for each situation and much like the issue of load balancing there is quite a bit of research going into making it work well. Hopefully, we can build on their work and help make web applications faster and more reliable in the real world.

**Architecture of the load balancer in http traffic:**

Here are the three main parts of the setup:

1. **The Internet**: Essentially, this is where everything comes from. People use the Internet to access websites and services by sending requests.

2. **Load Balancer**: Imagine the Load Balancer as a traffic cop. It's the server that sits between the Internet and the servers hosting websites or services. When someone sends a request over the Internet, the Load Balancer determines which server is best equipped to handle the request and routes them accordingly. It distributes out requests to servers to make sure no one server gets too many and everyone gets a fair amount of the work.

3. **Backend Servers**: These are the servers that actually host the websites or services you are trying to access. When the Load Balancer gets a request, it sends them to one of these servers. From there, the servers do what they need to do and respond. The Load Balancer keeps an eye on these servers. If one of these servers goes down or has too much traffic, it will not send requests to them until they get back up or traffic slows down. Sometimes, there is a feature called session persistence. This makes sure if you are doing something on a website and you move to another page that you still have your "session" with the same server. Essentially, it remembers you, which is very important for certain kinds of websites and their uses.

So, basically, this setup makes sure that websites and services run smoothly, handle lots of people at once, and don't crash even if one of the servers has a problem. It's all about making the Internet a better place for everyone

**Technologies used:**

1. Amazon Web Services (AWS) Elastic Load Balancing (ELB) is your personal traffic director for the web. It ensures that incoming traffic gets distributed across various targets, such as virtual servers or EC2 instances, containers or IP addresses. ELB allows your site to remain live and responsive even if there is an influx of visitors. In order to cater for different requirements, load balancers come in three types according to ELB.
2. Behind the scenes of this serverless architecture are the virtual servers known as Amazon EC2 instances which are responsible for processing all requests made by website users. These servers can be scaled up and down depending on traffic fluctuations and can also support different web servers such as Apache, Microsoft IIS or Nginx where you store your website files before they get served.
3. Auto Scaling acts as a self-regulating system that adds more EC2 instances when necessary based on demand signals coming from CloudWatch metrics like CPU utilization or request count per minute (RPM). This means it brings in extra hands during peak hours and dismisses them at off-peak hours hence allowing your site handle huge volumes of visitors without any hiccups along the way
4. Amazon Route 53 functions, as your guide on the internet directing traffic to your load balancer using intelligent routing strategies like weighted routing or latency based routing. It also monitors the health of your servers. Can redirect traffic from any servers experiencing issues.
5. AWS Certificate Manager (ACM) handles all security matters ensuring that communication between your website visitors and the load balancer is encrypted and secure. It manages the complexities of SSL/TLS certificates relieving you of this concern.
6. AWS WAF (Web Application Firewall) acts as a security guard, at your websites entrance shielding it from web attacks and detecting any activity to maintain the safety and security of your site.
7. AWS CloudWatch acts as a control panel to monitor the performance of your website. It keeps tabs on metrics such, as the volume of requests managed by your load balancer, the processing time for those requests and the presence of any errors. This feature allows you to oversee your websites well being and implement adjustments.
8. When combined these AWS services form an dependable framework for handling your websites traffic maintaining its operation and delivering an exceptional experience, to your site visitors.

**Evaluation and Testing:**

1. **Performance Testing:** Subject the load balancer to realistic levels of traffic in order to determine if it is capable of handling peak demand without a significant degradation in response times. Look at things such as throughput, latency, response times and request processing times.
2. **Scalability Testing:** Take your load balancer for a spin! Assign it more backend servers to accommodate a higher volume of traffic requests. Run tests with different numbers of concurrent connections and requests to get a sense of its performance and stability (ie, how well it scales horizontally under extreme loads).
3. **Fault Tolerance Testing:** Assess your load balancer’s responsiveness to server failures or other networking glitches, by shutting down backend servers, or by inducing them to become sluggish in processing traffic, and examining the load balancer’s ability to react by re-routig traffic to remaining healthy servers..
4. **Health Checking Evaluation:** We'll simulate server issues by taking some backend servers offline temporarily. This helps us see how quickly the load balancer identifies these problems and stops sending traffic their way.
5. **Session Persistence Testing:** If our system remembers users between visits (session persistence), we'll test if it correctly directs each user to the same server. Imagine remembering your shopping cart items even after a quick website detour!
6. **Algorithm Comparison:** We'll compare different traffic distribution methods (like Round Robin or Least Connections) used by the load balancer. We'll see which one keeps all servers busy without overloading any, ensuring smooth traffic flow.
7. **Security Testing:** We'll make sure the load balancer acts like a secure tunnel, encrypting communication between visitors and our servers. This keeps everyone's information safe and sound.
8. **Integration Testing:** We'll see if the load balancer works perfectly alongside other Amazon Web Services (AWS) tools we use, like Auto Scaling and EC2 instances. It's like making sure all the instruments in a band are in tune for a harmonious performance!

This test plan ensures the load balancer efficiently distributes traffic and keeps things running smoothly.

**Results:**

Load balancing undoubtedly improves website and application performance. Performance benchmarks demonstrate that in high traffic loads, the response time will be significantly lower, and the throughput – higher than in the case of a single server. Real websites that have improved their scalability and reduced downtime are using the technology, therefore real-life case analysis can be conducted. The cost analysis oftentimes shows that load balancing implementation is worth it: the loss of traffic and decreased performance downtime will surely exceed the amount of money spent on it. In case a customer satisfaction survey is conducted, the reports will show if the website started to load faster and if the user experience became more pleasant.

**Conclusion:**

In conclusion, load balancers are essential parts of the contemporary web architecture. They enable businesses to offer their applications to the users in different parts of the world in a fast, scalable, and reliable manner. If they plan and implement load balancing in the right manner and use the features of AWS effectively, they can get the best performance, scalability, and reliability for their web applications. When the performance of the web applications improves and they begin to run reliably, the users are happier, and the business outcomes are generally improved.

**Future Scope:**

Going forward, the project "Social Media Platform Using Cloud Computing" is poised to grow and develop further. One possible direction of development is the addition of artificial intelligence and machine learning algorithms, which can greatly improve the user experience, enhance recommendation systems, and availability of reliable data analytics services. Security, as a matter of fact, is growing by leaps and bounds, and particularly because of the concerns of privacy and trust on linking social media networks. So, there is a huge potential to integrate blockchain, which indeed enhances transparency, security, and data ownership.

**References:**

* NGINX offers a comprehensive guide on HTTP load balancing that provides a deep dive into load balancing and its role in managing HTTP traffic, which makes it worth reading if you want to learn more.
* Amazon Web Services (AWS) also offers a simple description of what load balancing is and why it is so beneficial for HTTP-based applications. It provides a rundown of the benefits of load balancers in an easily understandable manner. Amazon Web Services offers several services crucial for managing load balancers and optimizing HTTP traffic.
* Elastic Load Balancing is just as it sounds, it is like a traffic manager for your website. It is like having someone that ensures there are no traffic jams when your website is busy.
* Amazon EC2 is virtual servers that run your website, similar to a leasing office space. Auto Scaling is like a magical helper that increases the size of the servers, similar to how you would consider taking on more office space as your business grows.
* In Amazon Route 53, it is your personal GPS to the internet, helping people find your website. WAF is a shield that protects your web applications from being attacked while online. Load balancing involves distributing incoming network traffic across a group of backend servers, which increases site or app performance.
* The method enhances responsiveness and increases capacity, leading to optimal resource utilization business continuity.
* You need to know various methods for load balancing, based on HTTP header information about the client, or the application's session, or data about the application message like payload, URL, or parameters.
* Similarly, you balance by using IP and port of the client device, or server that handles requests from those devices or computing resources available with each type of storage in your data center plus geographical location factors.