DOI 10.36074/logos-26.05.2023.040

THE MAIN METHODS OF LOAD BALANCING ON THE NGINX WEB SERVER

Chyrvon Andrii
Candidate on bachelor's degree Department of Telecommunications Educational and research Institute of Telecommunication Systems
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

Kostiantyn Lisovskyi
Master's Student. Department of Information Communication Technologies and Systems Educational and research Institute of Telecommunication Systems
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

Kyryndas Nikita
Candidate on bachelor's degree Department of Telecommunications Educational and research Institute of Telecommunication Systems
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

UKRAINE

Abstract. *We've seen an unprecedented spike in website visitors as online activity continues to soar. This surge inevitably causes an increase in server traffic. Yet when theres too much demand and the server is at maximum capacity it can potentially crash. To address this issue, efforts have been made to improve the efficacy of the web server by utilizing a load balancing system to manage a large number of service access requests. Server load balancing can effectively prevent server failures by distributing workloads across multiple servers based on their individual capacities.*

Six servers were utilized in a recent demonstration utilizing the Apache software. 4,000, 8,000, 12,000, and 16,000 requests were sent to each server, respectively. To attain stability and balance in the web server, Nginx and three distinct load balancing algorithms were implemented: RoundRobin, Least Connection, and IP Hash. The "Least Connection" algorithm proved to be the most effective of these methods. It obtained the highest response speed by processing 2300.96 requests per second and exhibited remarkable response time stability of 116ms.

INTRODUCTION

The rapid expansion of the Internet has led to an increase in traffic, frequently resulting in server overflow. When a server becomes overloaded, it cannot process queries and goes offline. Utilizing traffic balancers helps prevent server downtime. Load balancers distribute duties across multiple servers according to their capacities, ensuring that no single server is overloaded. This allows requests from clients to be distributed evenly across the cluster nodes, preventing overloads and allowing the web server to manage 10,000 requests without encountering any errors.

If you're looking for ways to optimize your internet bandwidth while avoiding any imbalances load balancing techniques are an excellent option to consider. With tools like the Network Load Balancer (NLB) available, managing up to 10.82 requests per second is possible - well beyond what typical servers could handle at just 6.25 requests per second. By implementing these strategies you can make sure everything runs smoothly and efficiently at all times! In addition, the NLB system has speedier response times than a single server system, particularly when 1000 requests are generated. The response time for the NLB system is 333,716.915 milliseconds, while the response time for the single server system is 58,809.28 milliseconds. Therefore, the NLB system experiences significantly less request queuing than a single server.

Web server Nginx is known for its lightweight nature and high performance, according to previous research. Nginx is able to efficiently manage multiple requests and accommodate simultaneous access from multiple customers. This makes Nginx an ideal choice for load balancing, as it can manage large volumes of traffic efficiently.

METHOD

Popular web server and reverse proxy server Nginx is frequently employed for load balancing. It provides a variety of load balancing strategies or categories to distribute incoming requests across multiple backend servers. Here are some of the most common forms of load balancing in Nginx:

Round Robin

Round Robin is Nginx's default load-balancing algorithm. With Round Robin, incoming queries are distributed evenly and cyclically across the backend servers. Over time, each server receives the same number of requests, regardless of its current traffic or response time.

Here are some instances in which Round Robin load balancing can be advantageous:

• Round Robin ensures that incoming queries are evenly distributed across all backend servers. When you have a cluster of servers with comparable configurations and capacities, this can be useful. By distributing the load uniformly across servers, you can maximize resource utilization.
• Stateless Applications: If your application is stateless, meaning that it does not rely on maintaining session information or storing user-specific data on a specific server, Round Robin may be an appropriate choice. Since Round Robin distributes requests without regard to the current state of the server, it functions well for stateless applications that can independently process requests on any server.
• Round Robin is a straightforward and easy-to-implement method for load balancing. It does not require sophisticated configuration or server condition monitoring. If you have a small number of backend servers and desire a simple load balancing configuration, Round Robin can be a useful option.
• Fault Tolerance: Round Robin is capable of providing some fault tolerance. If one of the backend servers becomes inaccessible or encounters issues, the other servers in the rotation can continue to process queries. This ensures that the service's overall availability is maintained even if individual servers experience issues.

It is essential to observe that Round Robin may not be appropriate for all situations. If your backend servers have varying capacities, response times, or load requirements, other load balancing strategies, such as Least Connections or Least Time, may be more suitable. In addition, if your application relies on sustaining session data or requires session affinity, you may want to consider IP Hash or Weighted Load Balancing as alternative load balancing strategies.

The choice of load balancing method ultimately depends on the characteristics of your application, the configuration of your backend servers, and the intended behavior of your system under load.

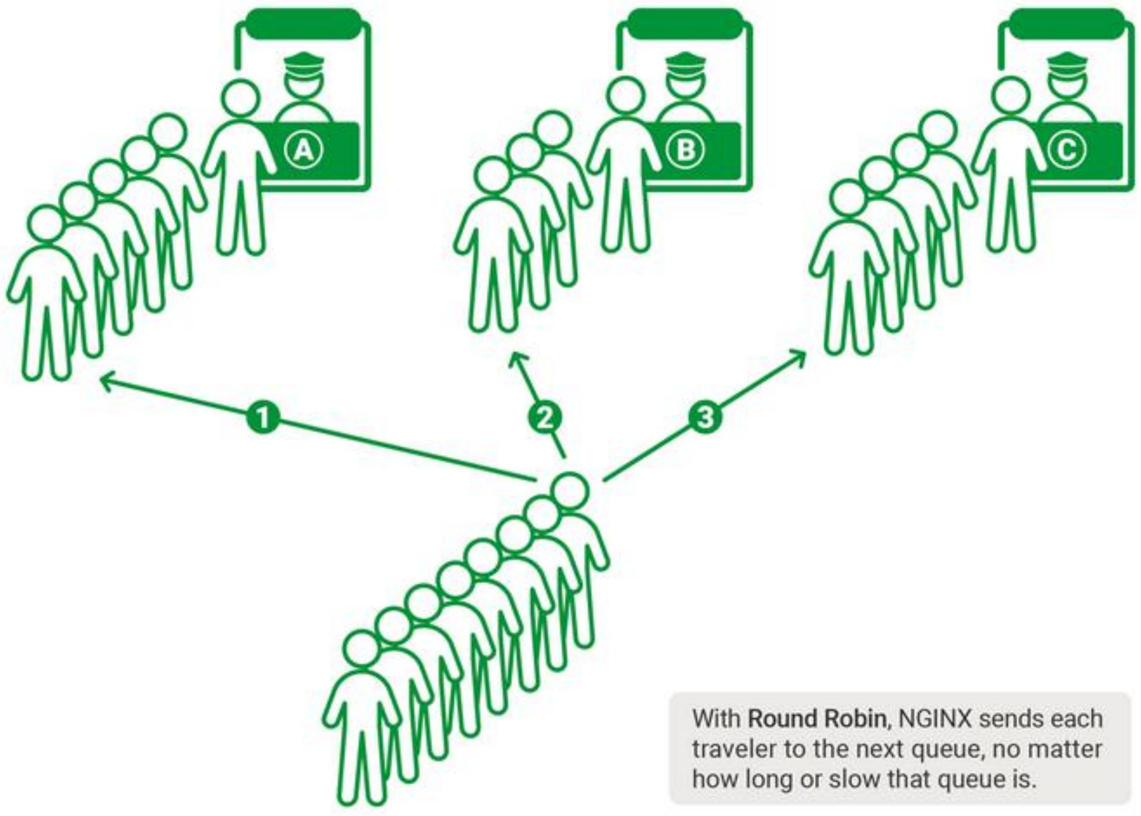
A
B
C
1
2
3
With Round Robin, NGINX sends each traveler to the next queue, no matter how long or slow that queue is.

Figure above: l**oad balancing method**

Least Connections Load Balancing

Least Connections load balancing is a technique in which Nginx directs incoming requests to the backend server with the smallest number of active connections at the time of the request. It attempts to distribute the burden based on the server's current workload, ensuring that requests are distributed evenly across all servers.

Here are a few instances in which Least Connections load balancing can be advantageous:

1. Variable Workload: If your backend servers experience fluctuating levels of traffic or load, Least Connections can help more efficiently distribute the burden. It dynamically adapts to each server's current state by routing requests to the server with the fewest active connections. This serves to prevent server overload and maximizes resource utilization across the entire server pool.

2. If your backend servers have varying capacities or capabilities, Least Connections can be helpful. By directing queries to the server with the fewest active connections, regardless of the server's capacity, the workload can be distributed more evenly. This can prevent a single server from becoming overloaded while others remain idle.

3. Least Connections can be advantageous if your application manages long-lived connections, such as persistent HTTP or WebSocket connections. As these connections can hold up server resources for extended periods of time, distributing them evenly across servers can prevent resource exhaustion on particular servers.

4. Response Time Considerations If some backend servers have longer response times due to network latency, hardware constraints, or other factors, Least Connections can mitigate the effect. By diverting requests to servers with fewer active

connections, you increase the likelihood of routing requests to more responsive and faster-handling servers.

5. Auto-Scaling Scenarios: Least Connections is beneficial in environments where servers are dynamically added or removed based on the current load. Achieving proportional allocation of incoming requests amongst new servers via load balancing is fundamental in minimizing the time it takes for these servers to attain full capacity.

Nonetheless, deciding on an appropriate load balancing method entails taking into account specific application traits and backend server capabilities as well as expected workload patterns. As effective as Least Connections can be in many instances, it might not suffice if there are requirements like session affinity or unique needs that can be best catered for through other options such as Round Robin, IP Hash or Weighted Load Balancing.

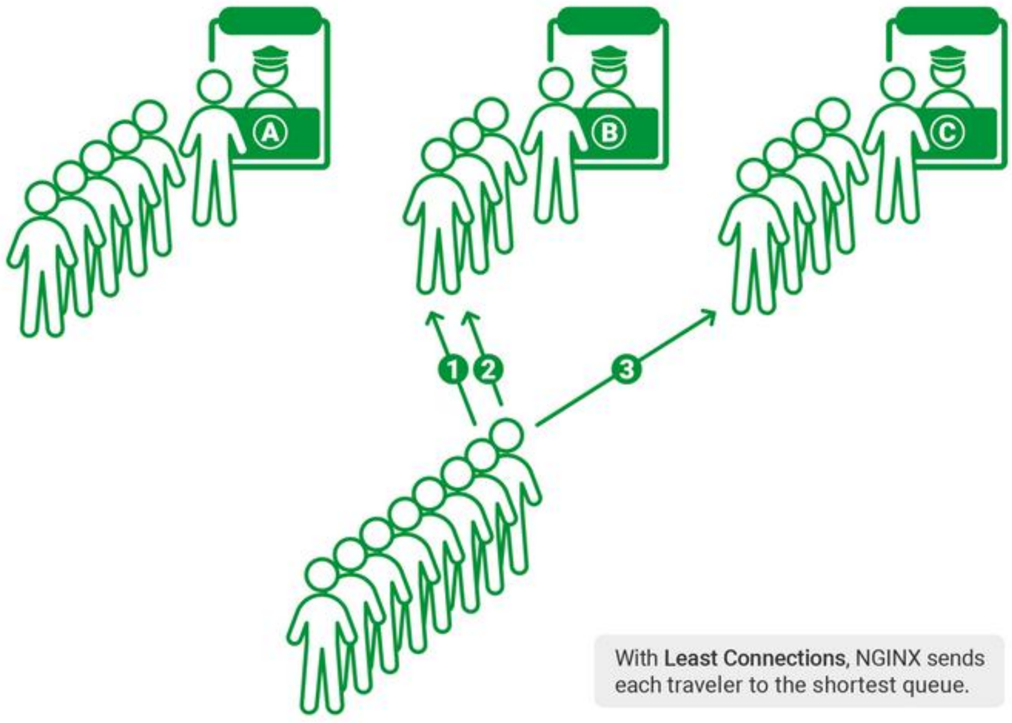
A
B
C
1
2
3
With Least Connections, NGINX sends each traveler to the shortest queue.

Figure above: **Least Connections load balancing method**

Least Response Time

Load balancing is the process of routing incoming queries to the backend server with the quickest response time. The objective is to distribute the duty so as to maximize overall performance and minimize client response times.

Here are a few instances in which Least Response Time load balancing can be advantageous:

If your backend servers have varying response times due to differences in hardware, network latency, or other factors, Least Response Time can help ensure that queries are routed to the server with the fastest performance. By minimizing the impact of slower servers on response times, this strategy can contribute to improved overall performance.

Least Response Time load balancing can be effective in situations where the burden on the backend servers fluctuates over time. It dynamically adapts to changes in server response times and routes new queries to the fastest-responding servers.

This ensures requests are managed by the most responsive servers and contributes to an efficient load distribution.

Least Response Time can be advantageous for applications that are susceptible to latency, such as real-time applications, gaming servers, and streaming services. By routing requests to the server with the quickest response time, you can reduce the overall latency experienced by consumers and improve the user experience.

Least Response Time load balancing can assist in optimizing the overall performance of your application. By continually routing requests to the server with the quickest response time, you can guarantee that client requests are always handled by the most capable and efficient server.

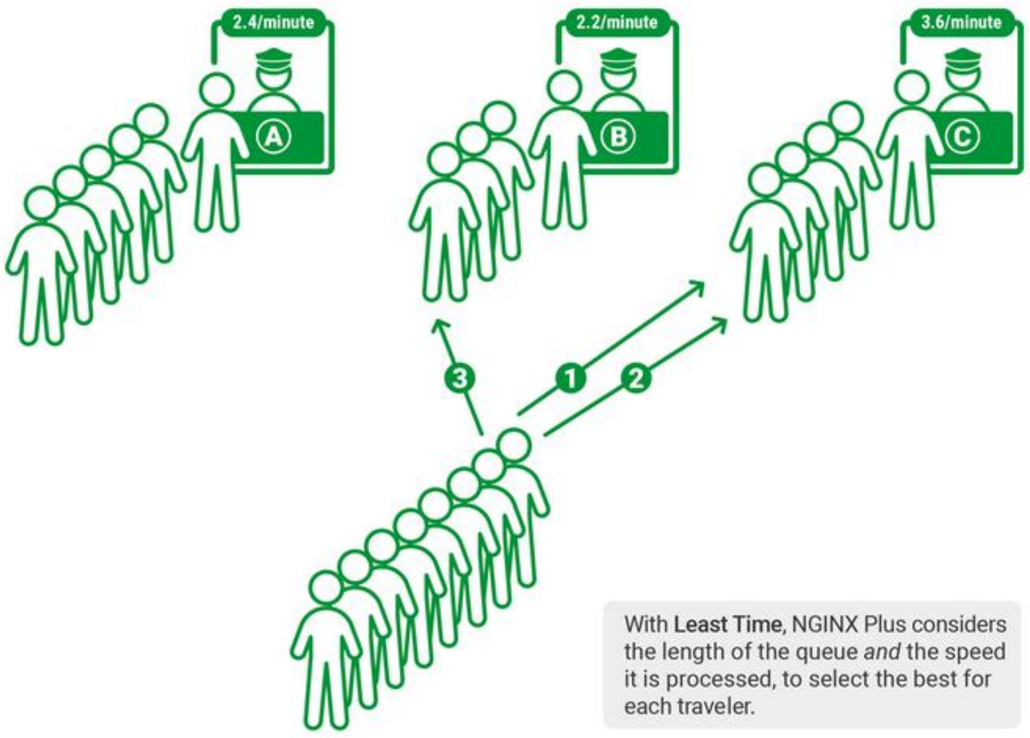
2.4/minute
2.2/minute
3.6/minute
A
B
C
3
1
2
With Least Time, NGINX Plus considers the length of the queue and the speed it is processed, to select the best for each traveler.

Figure above: **Least Response Time load balancing method**

CONCLUSION

Load balancing is essential for optimizing performance, improving scalability, and ensuring high availability of applications by distributing incoming requests across multiple backend servers. Least Time, Least Connections, and Round Robin are some of the load balancing methods offered by Nginx.

Least Time Load Distribution: In situations where server response times vary, it may be beneficial to direct requests to the server with the quickest response time. This can optimize overall efficacy and reduce client response times.

Least Connections Load Balancing: Least Connections load balancing is an important Nginx method. It directs incoming requests to the server with the fewest active connections, based on the server's current demand. It is especially advantageous in scenarios with dynamic workloads, heterogeneous server performance, or when attempting to balance the burden across servers efficiently.

Round Robin (default) Load Balancing: Round Robin is Nginx's default method for load balancing. It distributes incoming requests equitably across backend servers in a cyclical fashion, ensuring an equal number of requests over time. Round Robin is simple to configure and appropriate for situations in which servers have comparable capacities and load distribution is the primary objective.

References:

[1] *F5 NGINX* (By F5, Inc.). (2023). F5 NGINX. Retrieved from https://www.nginx.com/

[2] Rahmatulloh, A., & Lisa, F. M. R. C. (2017). Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi. *Jurnal Nasional Teknologi Dan Sistem Informasi (TEKNOSI) Universitas Andalas, 3(2),* 241–248. https://doi.org/10.25077/teknosi.v3i2.2017.241-248

[3] DeJongh, D. (2019). *The Complete NGINX Cookbook - 2019 Edition*. https://www.nginx.com/resources/library/complete-nginx-cookbook/

[4] Garrett, O., & Smith, F. (2022). *High-Performance Caching with NGINX and NGINX Plus.* https://www.nginx.com/resources/library/high-performance-caching-with-nginx-and-nginx-plus/

[5] High Availability Server Implementation Using Load Balancing and Failover Methods on Virtual Web Server Clusters. (2022). E-Proceedings of Engineering, 4496–4503.

[6] Riskiono, S. D., Rosalia, M., Munadi, R., & Mayasari, R. (2016). Implementation of Load Balancing Methods in Supporting Server Cluster Systems. SEMNAS RISTEK, 455–460.