Implementing reverse proxy mechanism (load balancer - HTTP) alongside data encryption (TLS/SSL)

Nithin Veer Reddy Kankanti
University of Colorado Boulder
Boulder CO, USA
nithin.kankanti@colorado.edu

Srinivas Baskar
University of Colorado Boulder
Boulder CO, USA
srinivas.baskar@colorado.edu

Sitesh Ranjan
University of Colorado Boulder
Boulder CO, USA
sitesh.ranjan@colorado.edu

Chanheum Park
University of Colorado Boulder
Boulder CO, USA
chpa8095@colorado.edu

ABSTRACT

Load balancers are often used for balancing the heavy amount of load received on a public facing server which further divides into the servers beneath it. Technically it is defined as the methodical and efficient distribution of network or application traffic across multiple servers in a server farm. Each load balancer sits between client devices and application or backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them. There are many cloud providers that offer load balancer as a managed service but failed to provide a solution which can be transferred from one to another cloud provider or may be to a local data center. This solution would aim to offer a solution where the load balancer application can be deployed on any cloud provider or even on your own data centre. In other words, this will be a cloud platform agnostic solution. To improve the security an end to end encryption on the requests is also being offered with this solution. This would further improve the security of the whole application.

KEYWORDS

load balancer, reverse proxy, encryption, server side, client side, public subnets, private subnets, internet gateway, SSL/TLS, cloud providers

1 INTRODUCTION

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.

Modern high-traffic websites must serve hundreds of thousands, if not millions, of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. To cost-effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.

A load balancer acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

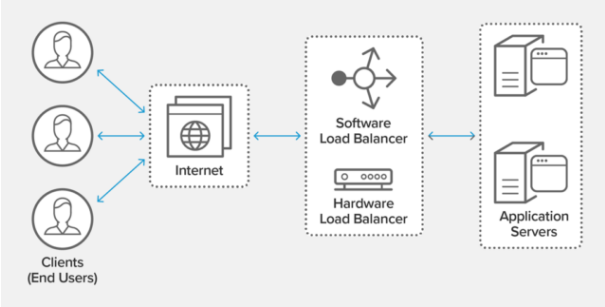

Figure 1: Load balancing diagram

2 PROPOSED WORK

2.1 Architecture

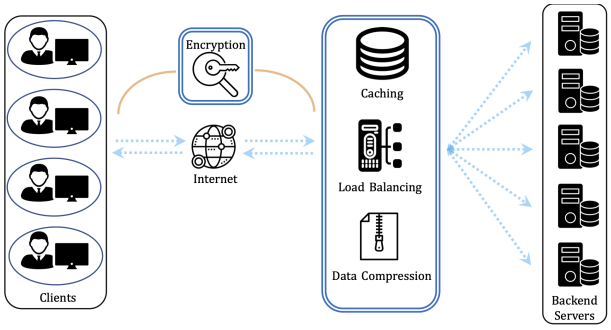

Figure 2: Diagram of our Proposed Work

At upper level, the application server acting as load balancer is the only public facing server on the server side. Internally there would be a series of backend/application servers which are in private subnet and responds to the requests raised by the clients. Load balancer also aims to provide compression and caching static data. Caching would help respond the requests faster by saving the data on the

load balancer server itself. This would greatly help the servers by offloading humongous amounts of requests in certain scenarios. Compression would save the bandwidth thereby allowing more room for new requests and responses. Another important feature is encryption where an encrypted request from the client is made and the request can be only decrypted on the backend servers which are beneath the load balancers. This would greatly increase the security of the request/responses originating from both clients and servers.

2.2 Components

Load Balancing - Our reverse proxy server sits in front of our backend servers and distributes clients requests in a manner that it maximizes speed and bandwidth utilization while mitigating server overloads which is the cause of poor performance. When a server goes down or suffers a catastrophic failure, then our load balancer will redirect the traffic to another server.

Encryption and Privacy - Reverse proxy server protects the client identities by intercepting requests headed to the backend servers. We encrypt our plain text client requests by using a combination of symmetric and asymmetric encryption. SSL/TLS encryption is applied by performing a TLS handshake where we can verify the clients and confirm their identities and use symmetric encryption to achieve better performance in terms of speed in larger data.

Compression - By compressing inbound and outbound data, we can speed up the traffic flows by reducing the load off each flows.

Caching - We can also enhance the overall performance by caching the frequently requested files instead of forwarding the requests to the backend server.

2.3 Load Balancing Algorithm

Load balancing involves distribution of jobs/requests to multiple servers. The important problem here is to ensure a balanced distribution of load. This is important to improve the user experience, as an efficient load distribution will result in low latency for the user, improving the user experience.
A dynamic load balancing technique reacts to the state of each server when distributing the load, whereas a static load balancing technique only depends on the average behaviour of the servers [1]. Due to this, a dynamic load balancing technique fares better.
In this project, we propose to implement a load balancer using the Rate Adjustment Policy [2]. In this policy, the scheduler keep the health of each of the servers in check. Using this information, the scheduler distributes the load. This information may include the remaining processing capacity (the number of process threads the server has remaining), the response time of the server, etc. Each server has a manually set parameter indicating the maximum number of processes that can be run on that server. Each server can only accept a maximum of this number of requests at a time. Besides this, the load balancer would also try to send the request to a server with the lowest response time (if available). The load balancer could also act as a firewall to the servers it is connected to, dropping any malicious requests.

3 EVALUATION

To evaluate the efficiency of our system, we will use a single server to accept the requests and compare this to our load balanced system. To be fair to a single server system, we will keep the per server requests constant. We will then compare the average per request time between the two.
Besides, this we will compare our system to the ones provided by amazon, google, etc. Our objective here is to achieve a performance close enough to these state-of-the-art systems.
We will also test the caching efficiency of our system by sending requests which should have been cached. This should result in significantly smaller response time.

4 MILESTONES

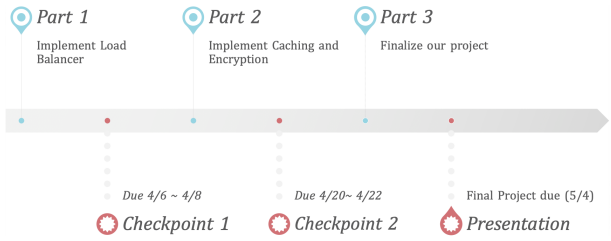

Figure 3: Project Milestone

A step-by-step roadmap to implementing reverse proxy mechanism alongside data encryption includes the following milestones.

- Week 1, Checkpoint 1:
  Implement load balancer with a dynamic load balancing technique that can efficiently divide the load to multiple predefined serves.
- Week 2:
  Implement caching and security on the reverse proxy load balancer. The load balancer should be able to decrypt messages before sending the request to the server. And any request that the server has already seen should be responded by the reverse proxy.
- Week 3, Checkpoint 2:
  Implement compression for the responses the reverse proxy sends to the client.
- Week 4, Final submission:
  Evaluate the system and report the results.

REFERENCES

[1] Zeng Zeng and Bharadwaj Veeravalli. *Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems, Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS'04) 1521-9097/04*
[2] Zahra Mohammed Elngomi, Khalid Khanfar. A Comparative Study of Load Balancing Algorithms: A Review Paper, IJCSMC, Vol. 5, Issue. 6, June 2016, pg.448–458
[3] Peter Sommerlad. Reverse Proxy Patterns
[4] Jiang Du, GuoXin Nie. Design and Implementation of Security Reverse Data Proxy Server Based on SSL, ICCIC 2011: Information and Management Engineering pp 523-528
[5] Sommerlad, P. Reverse Proxy Patterns. EuroPLoP (2003)