CS3543: Computer Networks - II
Professor : Kotaro Kataoka
Department of Computer Science
Indian Institute of Technology Hyderabad

Load Balancer

R Gokul Kannan - CS19BTECH11048

Danda Sarat Chandra - CS19BTECH11003

Veeramalli Sathwik - CS19BTECH11022

Pochetti Rashmitha - CS18BTECH11019

L Harsha Vardhan - CS18BTECH11023

K Lakshmi Sravya - CS19BTECH11017

May 3, 2022

Contents

1 Introduction

- Load balancer plays an important role in Cloud Infrastructure. It improves efficiency and saves a lot of money for security by hiding servers from internet.
- A load balancer is a piece of hardware (or virtual hardware) that acts like a reverse proxy to distribute network and/or application traffic across different servers.
- A load balancer is used to improve the concurrent user capacity and overall reliability of applications.
- A load balancer helps to improve these by distributing the workload across multiple servers, decreasing the overall burden placed on each server.
- Physical load balancing appliances are similar in appearance to routers. They are connected to your network infrastructure in the same manner as a router or another server.
- In contrast, virtual load balancing hardware is a program that emulates the addition of a new hardware solution. These virtual load balancers work in a manner similar to a virtual server or a virtual computer on your network.
- They act as a physical load balancer and distribute requests accordingly, saving you space and offering greater flexibility than many hardware balancers.

2 Software Overview

2.1 Load Balancer

Load balancer sits in between servers and clients. It communicates with them using sockets. After initializing all servers, we start load balancer. It listens for clients and based on whether its query or file transfer it transmits the request to the server and sends back the response to clients.

2.2 Client

Generally the client requires some files (typically front-end files like html). If the demand increases, clients or end users increase load on the server. In case of a single server it is impossible to keep up.This is where load balancer comes into picture.

2.3 Server

Server has the files and database in it. Upon client querying for database or requesting for some files, server sends them to load balancer and load balancer sends files to the client.

2.4 Algorithm

Load balancer assigns an idle server based on round robin method. In this the list of idle servers are maintained ordered according to their power (the lesser the processing time, the higher the computing power).

2.5 Features

The main features are:

- **Database management and Coherence:** All servers are connected to the same database and this ensures all of them are updated correctly. There are various online database available, like Firebase, which can be used if servers are across multiple networks.
- **File transfer:** Transfer of any type of files from server to client through Load balancer.
- **Load balancing:** Allocating servers in round robin method which is efficient.

2.6 Rules/Protocols

- If the message from the client starts with **"file . . . "**, it is considered to be a file request.Any other message is considered to be a database query.
- TCP (Transmission Control Protocol) sockets are used for all the sending and receiving in the network (client - load balancer - server) to ensure reliability.

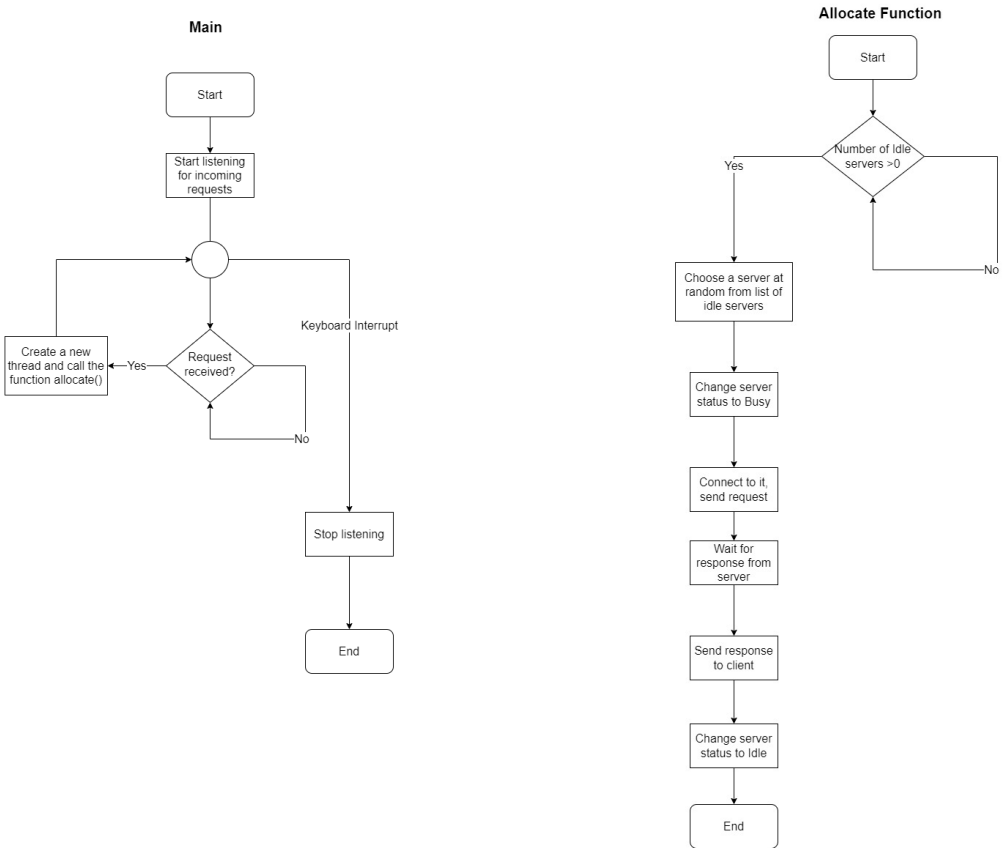
2.7 Difficulties Faced

- We tried to load balance the servers dynamically i.e, if the server tries to connect with load balancer then load balancer adds that to list of active server. But it was difficult, and with more time, we may be able to do it.
- Assigning different processing powers for different servers.
- File transmission was a bit difficult as we involved another load balancer in middle.

.

3 Software Design

3.1 Baseline Implementation



3.1.1 Load Balancer

- Sockets and threading are used.
- The server reads from the file "server.json" and connects to all the active servers at the time of starting the load balancer.
- Once a client connects to the load balancer, it chooses a server from the list of idle servers, creating a new thread.
- The request is sent to the server, it waits till the response is received.
- Once the response is received, the load balancer sends it back to the client that requested it.

3.1.2 Client

- The client is given an ID.
- The user is prompted to enter the request, which is sent to the load balancer.
- `{(file \<filename\> - to receive a file, query - to execute a database query)}.`
- Once the response is received, it is displayed, and the program terminates.

3.1.3 Server

- The server starts listening for incoming connections, and writes its IP address and port to the file "servers.json"
- The server connects to the database using the database credentials in the file "database.json".
- The load balancer connects to the server, and if a request is received, it processes the request and sends the response back to loadbalancer.Then it keeps listening for more incoming requests.

4 Software Functioning

4.1 File Transfer

- First the size of file is sent to the load balancer, and it will send it to client.
- Client prepares to receive the file as bytes, creates a new file, opens it in write bytes mode, writes incoming bytes onto it, and saves the file. The bytes are received in the loadbalancer first, and it sends them to the client.
- If the file requested by the client is not in the server, an error message is sent to the client.

4.2 Database Management

- The client sends the query to the load balancer, it will allocate a server.
- The server executes the query, and returns the result.
- If there is no result, a "No output" message is returned.
- The load balancer receives the output, and returns it to the client.

4.3 Load Balancer

- The load balancer connects to all the servers that are active at the time of starting the load balancer.
- When a client connects to the load balancer, and sends a request, a server is allocated for it, and it waits till the server responds.
- The load balancer sends the response back to the client that sent the request.

5 Procedure

First, make sure the contents of the file-

- "servers.json" is "[]" only.
- Input the database credentials into the file - database.json
- To run the Server open a terminal for each server, and type :

```
python server.py <server_id> <Server_port> <Processing_time>
```

- Open another terminal and run the loadbalancer, by using the command python `loadbalancer.py`.
- Open another terminal for each client, and type `"python client.py <client_id>"` to run each client.
- When prompted to enter a request, either enter a query to be processed or `"file <filename>"` to request for a file from the server.
- The response will be displayed/the requested file will be downloaded in the same directory as the client.py file.

6 Database Query

6.1 Client Side

```
PS D:\IITH\sem 6\Computer Networks 2\project\codes\client> python client.py 2
Enter query: SELECT * FROM COLORS LIMIT 3;
Sending request from client 2
Name: White
HEX: #FFFFFF
RGB: rgb(100,100,100)

Name: Silver
HEX: #C0C0C0
RGB: rgb(75,75,75)

Name: Gray
HEX: #808080
RGB: rgb(50,50,50)


Time taken: 10.027992725372314
PS D:\IITH\sem 6\Computer Networks 2\project\codes\client>
```

6.2 Server Side

```
PS D:\IITH\sem 6\Computer Networks 2\project\codes> python server.py 2 5072 6
Connected to databse successfully
Server 2 is starting...
Server is listening on ('192.168.1.206', 5072).
Connected with ('192.168.1.206', 55987)
<socket.socket fd=548, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('192.168.1.206', 5072), raddr=('192.168.1.206', 55987)>
SELECT * FROM COLORS;
Received request. Processing...
Response sent!
```

7 File Transfer

7.1 Client Side

```
PS D:\IITH\sem 6\Computer Networks 2\project\codes\client> python .\client.py 1
Enter query: file colors.csv
Sending request from client 1
Preparing to receive the file: colors.csv
File colors.csv received successfully!
Time taken: 10.018753051757812
PS D:\IITH\sem 6\Computer Networks 2\project\codes\client>
```

7.2 Server Side

```
PS D:\IITH\sem 6\Computer Networks 2\project\codes> python server.py 1 5071 10
Connected to databse successfully
Server 1 is starting...
Server is listening on ('192.168.1.206', 5071).
Connected with ('192.168.1.206', 55986)
<socket.socket fd=564, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('192.168.1.206', 5071), raddr=('192.168.1.206', 55986)>
file colors.csv
Received request. Processing...
Starting to send file
File colors.csv sent successfully
```

8 Load Balancer for both Functionalities

```
Connected to server 1
Connected to server 2
Connected with ('192.168.1.206', 55995)
file colors.csv
calling function
Choosen server: 1
Connected with ('192.168.1.206', 56000)
SELECT * FROM COLORS;
calling function
Choosen server: 2
Got response from server!
Sent response to client!
Sent requested file from server 1!
Sent response to client!
Connected with ('192.168.1.206', 56041)
SELECT * FROM COLORS LIMIT 3;
calling function
Choosen server: 1
Got response from server!
Sent response to client!
```

9 Improvements

- Dynamically connecting to new servers when they are added (Scalability)
- Include some form of security in the loadbalancer, Include more functionalities other than File transfer and Database queries Support for load balancing when servers are running in different networks/different computers.
- We can use GUI for visualizing the working of the load balancer.
- We can try using locks for the list of idle servers, as that list is accessed by multiple threads, maybe at the same time, which may lead to race conditions.

10 References

- reference 1
- reference 2
- reference 3

THE END!
LaTeX generated document