



Load Balancer

Lalita Panika¹, Kiran Dhirani², S Shubham³, Rishi Nirmalkar⁴, Tanya Wadhvani⁵

¹ Assistant Professor, Bhilai Institute of Technology, Raipur, Chhattisgarh, India

^{2,3,4,5} Student, Bhilai Institute of Technology, Raipur, Chhattisgarh, India

ABSTRACT

This here abstract introduces a Java based load balancer designed to optimize resource allocation and enhance the performance of distributed systems. Our Java based load balancer leverages sophisticated algorithms and dynamic routing techniques to evenly distribute workload among servers, and prevent overloading on any single node, and ensuring optimal utilization of resources. Implemented in Java, the loadbalncer demonstates flexibility, scalability, robustness and all, catering to diverse application requirements in distributed environments.

The significance of load balancer is to achieve high availability, scalability, and reliability in distributed environments, which emphasizes its pivotal role in modern computing infrastructures ya know. The core functionalities of this load balancer thingy include real-time monitoring server health, dynamic adjustment of traffic distribution based on server loads, and seamless scaling to accommodate varying workloads .

Keywords: load balancer , algorithms , flexibility , scalability

1. Introduction

A Java-based load balancer is serving as a pivotal intermediary between clients, you know, and a pool of backend servers. Its smartness lies in its ability to route incoming requests to the most suitable server based on various factors. These factors include the health of the server, the ongoing load, and even the proximity. The primary advantage here is Java. It's such a versatile language that it can run on various operating systems with minimal modifications. The whole portability aspect makes Java-based load balancers super attractive for enterprises. They are just so adaptable to infrastructure needs. Java-based load balancers are pivotal. I mean, they play a role, an important one, in ensuring efficient and reliable distribution of network traffic across servers. They contribute significantly to scalability, performance, and resilience of modern web applications.

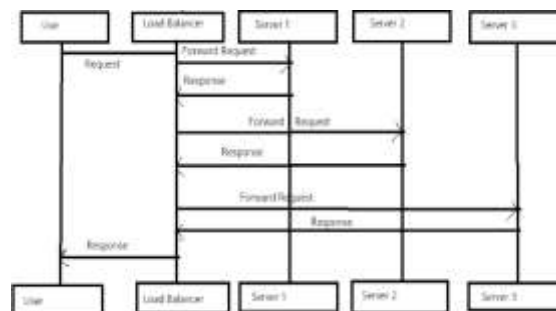


Fig.1-(a). Load Balancer

2. METHODOLOGY

The methodology involved in this process includes intercepting incoming requests, which kind of means catching them as they come, and then placing them into a queue. After that, each request is dynamically assigned to a server based on factors like server availability, load, or performance metrics, which makes everything very smooth. A queue, which is utilized by a queue-based load balancer, playfully distributes incoming requests among multiple servers. This distribution happens by queuing up the request s, and then assigning them to available servers based on some predefined algorithms. This approach is quite efficient and ensures that resources are utilized in the most efficient way possible. Additionally, it also helps prevent server overload, which is always a good thing. It also has the ability to enhance system scalability and resilience. This means that the system is able to handle more and more requests and still maintain its stability.

2.1 Load Balancing Algorithms

a) Round Robin;

- Distribute requests equally among available servers in a circular manner, like going around and around in a merry-go-round.
- Implement logic to cycle, bike, or maybe even skateboard through servers in a round-robin fashion.

b) Weighted Round Robin:

- Assign different weights or sizes to servers based on their capabilities (CPU, memory, etc.), like comparing apples and oranges .
- Distribute requests in proportion to server weights to optimize and make the most of resource utilization, utilizing them all to max.

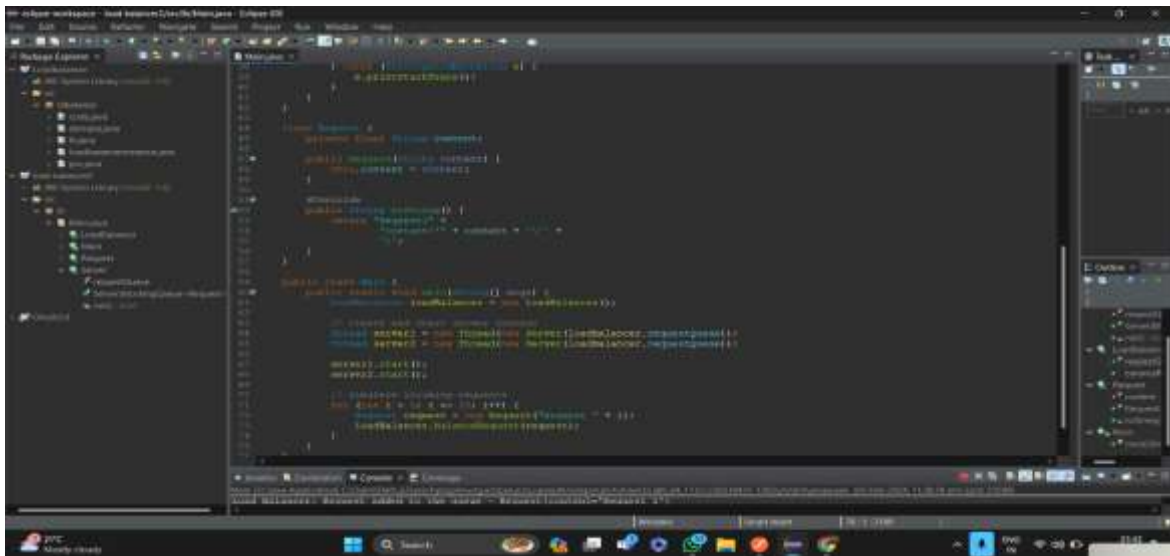
c) Least Connections:

- Track the number of active connections on every single server, like keeping count of how many friends you have on social media.
- Route requests to the server with the fewest active connections to evenly distribute the load, just like sharing a slice of chocolate .

d) Random Selection:

- Randomly select a server from the pool to handle every single new request, like pulling a name out of a hat.
- Implement logic to generate random selection within the server pool, just like rolling dice or flipping a coin. It's all up to chance

2.2 Load Balancer Code



```
package main

import (
    "fmt"
    "log"
    "net/http"
    "net/http/httputil"
    "os"
    "os/signal"
    "strings"
    "sync"
    "time"
)

const (
    port = 8080
)

var (
    mu sync.Mutex
    servers []string
)

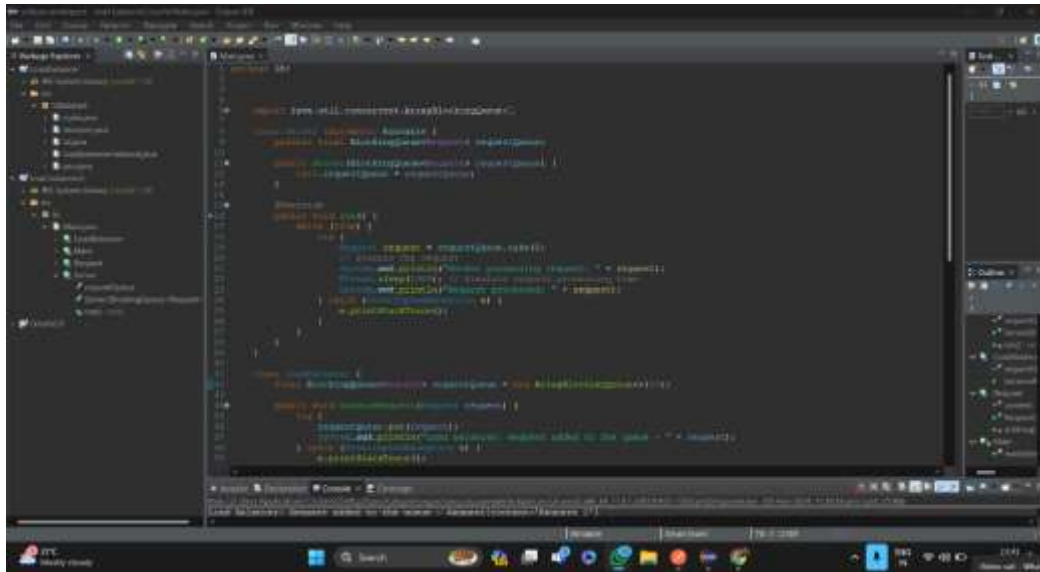
func main() {
    servers = []string{"http://127.0.0.1:8081", "http://127.0.0.1:8082"}

    http.HandleFunc("/", proxy)

    go func() {
        log.Println("Listening on port", port)
        http.ListenAndServe(":"+port, nil)
    }()

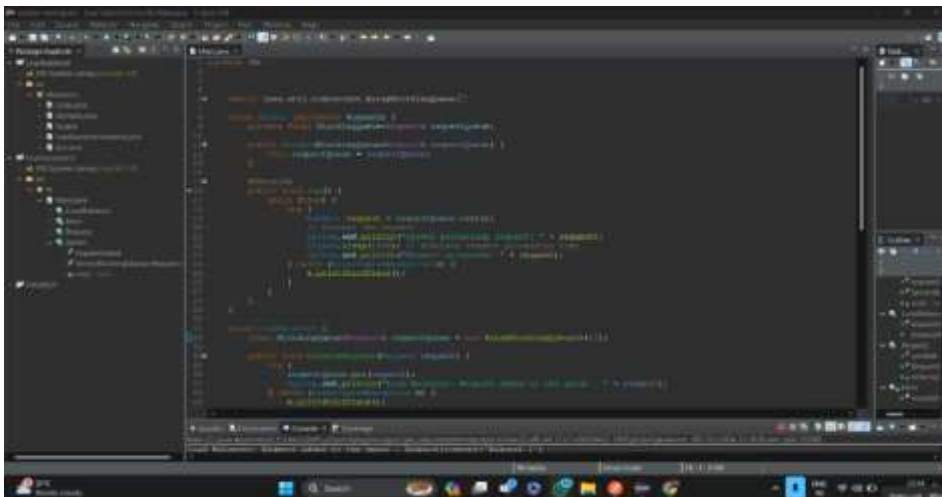
    signal.Notify(os.Stderr, os.Interrupt)
    log.Println("Press Ctrl+C to exit")
}
```

2.2 Code for Server



A screenshot of the Visual Studio Code editor displaying a server-side code file. The code is written in a dark theme and includes various comments and function definitions. The code appears to be a server-side implementation, possibly for a web application, involving database connections and data processing. The interface shows the file explorer on the left, the code editor in the center, and the output console at the bottom.

2.3 QUEUE IMPLEMENTATION



A screenshot of the Visual Studio Code editor showing the implementation of a queue. The code is written in a dark theme and includes various comments and function definitions. The code appears to be a server-side implementation, possibly for a web application, involving database connections and data processing. The interface shows the file explorer on the left, the code editor in the center, and the output console at the bottom.

-
- [5] S.K. Vasudevan, S. Anandaram, A.J. Menon, A. Aravinth A novel improved honey bee based load balancing technique in cloud computing environment.
- [6] Research on the Cloud Computing Load Balance Degree of Priority Scheduling Algorithm based on Convex Optimization Theory'.