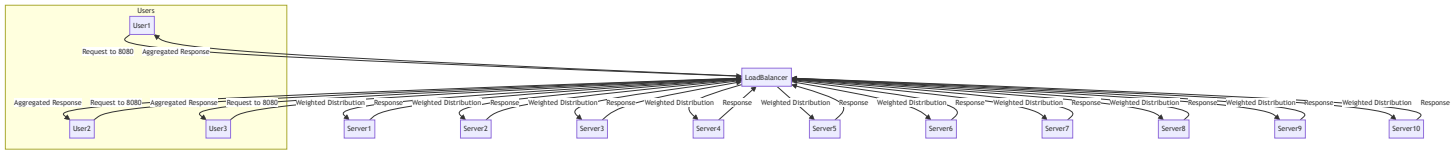
ExpressBalancify

This project implements a simple but quite extensive Express.js-based load balancer that uses a **weighted round-robin algorithm** to distribute incoming requests across multiple servers. It also includes logging functionality to track the details of each request and a Python script for analyzing the logs.

> Weighted Round-Robin is a load balancing algorithm that assigns a weight to each server based on its performance capabilities. The higher the weight assigned to a server, the more requests it will receive. When a server goes down, its weight is reassigned to other servers. This algorithm is used to achieve optimal resource utilization and avoid overloading a single server. Source

Explanation

The load balancer is implemented using the Round-Robin algorithm. The algorithm is implemented in the `index.js` file. It use 10 servers from port 5073 to 5973. Each server has



Usage

Prerequisites

- Node.js
- npm
- Python 3
- npx
- pm2

Installation

1. Clone the repository

```
git clone https://github.com/hendurhance/express-balancify.git
```

2. Change directory

```
cd express-balancify
```

3. Install dependencies

```
npm install
```

4. Start the servers

```
npm run start
```

5. Start the load balancer

```
npm run start:balancer
```

6. Test the load balancer

> The available routes are `/`, `/api`. You can access the load balancer at `http://localhost:8000` health check route and `http://localhost:8000/api` for the API route.

Route	Description
`/`	It is an health check route that returns a JSON object with the message
`/api`	It is a route that returns a JSON object of a sales sample data of up to 3000 records

To test the load balancer, you can use the following command:

```
# Health check route
npm run test:load:health

# API route
npm run test:load:api
```

This will send 1200 requests to the load balancer and 400 concurrent requests at a time. You can change the number of requests and concurrent requests by changing the values of in the `package.json` file.

Logging

Requests and their details, such as timestamp, IP address, server port, and original URL, are logged to a CSV file (`request_logs.csv`). The log file contains the following information:

```
# Log file content
Timestamp,IP Address,Server Port,Original URL
2023-12-17T17:39:46.250Z,::1,5673,/
```

You can analyze the log file using the run the following command:

```
npm run analyze
```

The script generates a report and visualizations, including bar charts for the distribution of requests between server ports, the highest amount of requests handled by each server, and a pie chart showing the percentage distribution of requests between server ports.

Contributing

Feel free to contribute to this project by opening issues or submitting pull requests. Your feedback and contributions are highly appreciated.

License

This project is licensed under the MIT License. See the LICENSE file for details.

Author

- Endurance - Github - Twitter - LinkedIn

Support

If you found this project useful, please consider starring it. You can also buy me a coffee ☕ or become a sponsor using the links below ➡️ github sponsors. Thanks a bunch for your support!