Efficient Load Balancing in Distributed Systems

CS262 Final Project

Ye Joo Han & Evan Howard

Introduction

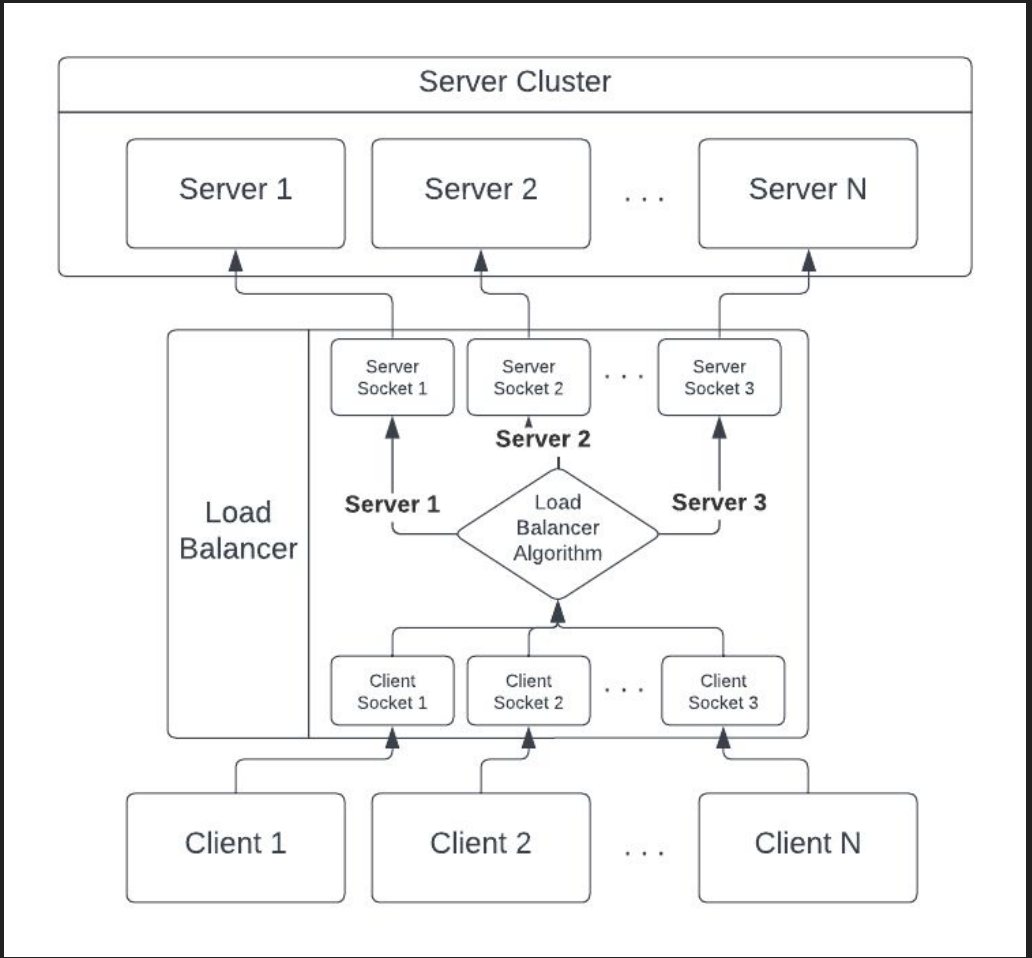
- What is a load balancer?
  - Acts as an intermediary between clients and servers
  - Receives incoming client requests and distributes them among available servers
  - Uses load balancing algorithms to determine the best server for each request
  - Monitors the health and availability of servers to maintain efficient distribution
- Very important tool in modern distributed systems!

Load Balancing Algorithms

- Variety of load balancing algorithms used in practice
- Two main classes: Static and Dynamic
  - Static: Round Robin
  - Dynamic: Least Connections
- We've implemented one of each class to explore the differences in two widely used load balancing algorithms, along with a baseline that just connects to a single server

Project Components

- Load Balancer
- Server
- Client
- Stress Tester



Load Balancer Implementation

- Written in python
- Keeps a list of all available servers
- When a client connects, selects a server according to an algorithm and sets up mappings from a Server-LoadBalancer socket to a LoadBalancer-Client socket and vice versa
- When any data is received from those sockets, it forwards it to the corresponding socket

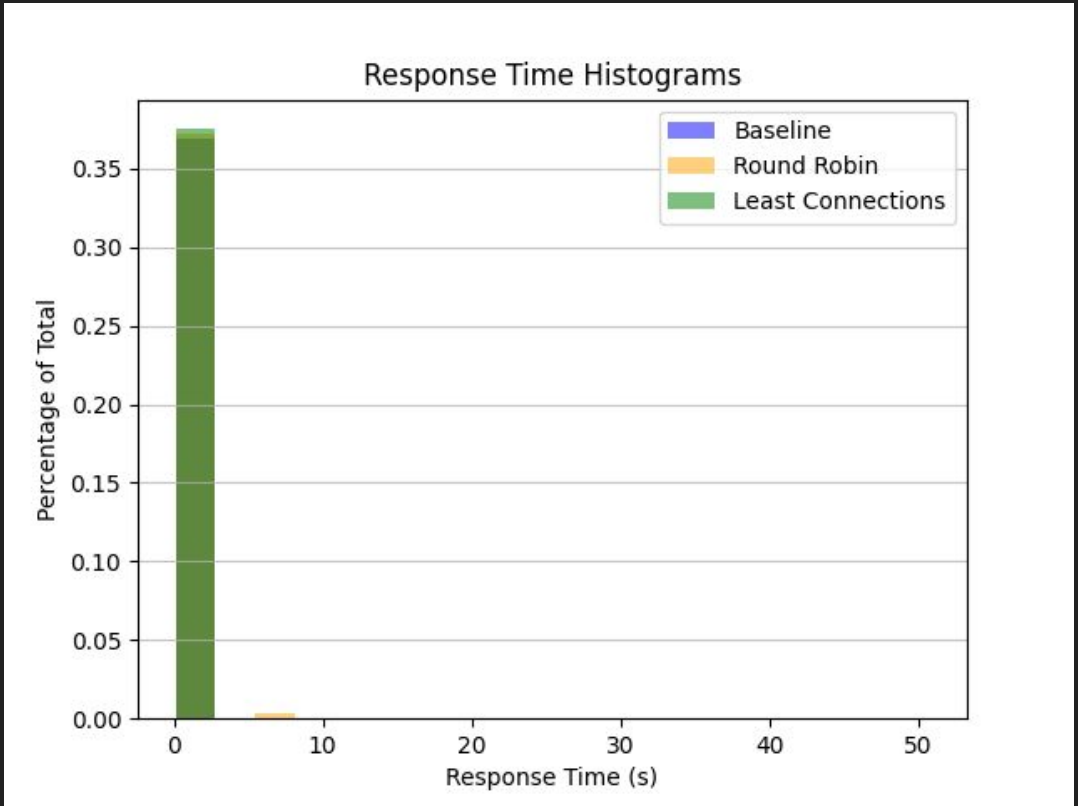
Client and Server Implementation

- Typical python implementation of client and server, from their perspectives the load balancer doesn’t exist
- In our experiments, server and client send simple ping and pong response
- Server has various features to simulate real world conditions and get better comparisons between algorithms
    - Simulated server processing time using a sleep statement on a normal distribution
    - Simulated server overloading with a maximum number of active connections
- Client keeps track of response time for use in experiments

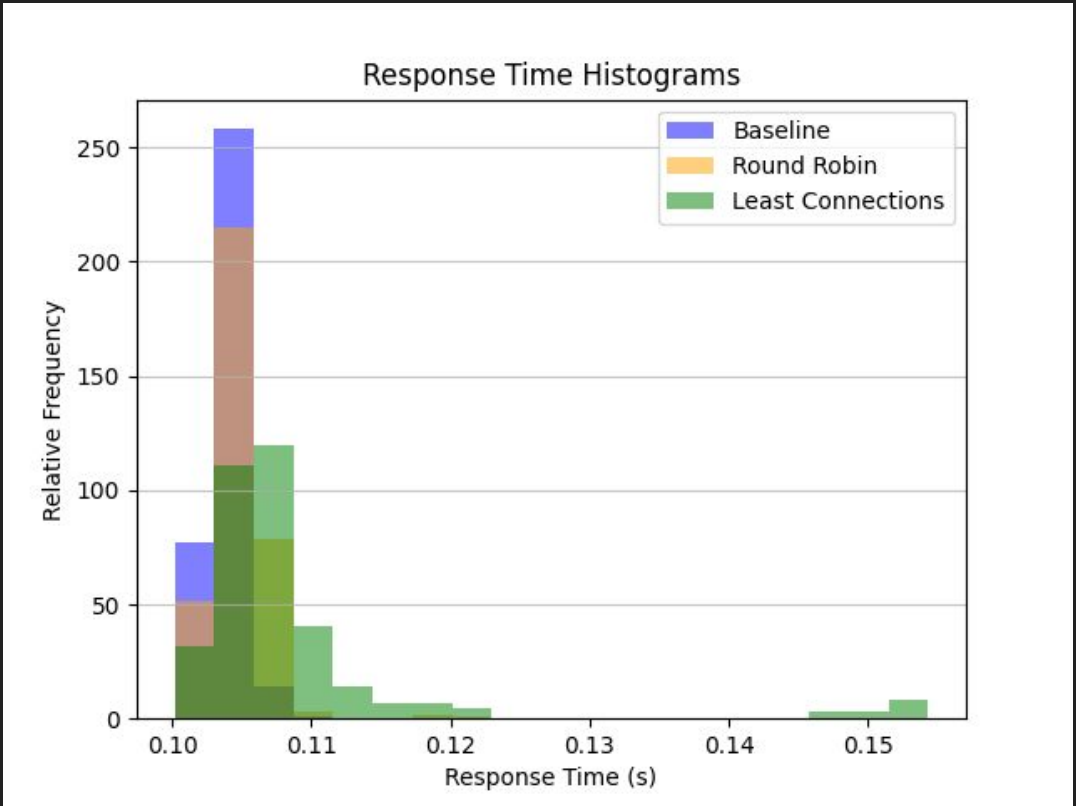
Experiments and Evaluation

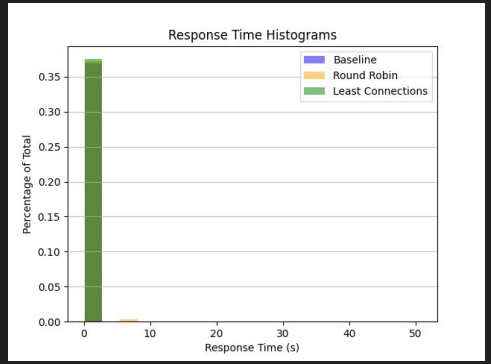
- Using our stress tester, we ran an experiment with 100 parallel clients, all making a random number of requests between 1 and 100, with 5 servers each having a maximum number of 10 active connections
- We ran this experiment with a single server (i.e. no load balancer), with a round robin load balancer, and with a least connections load balancer

Results and Graphs



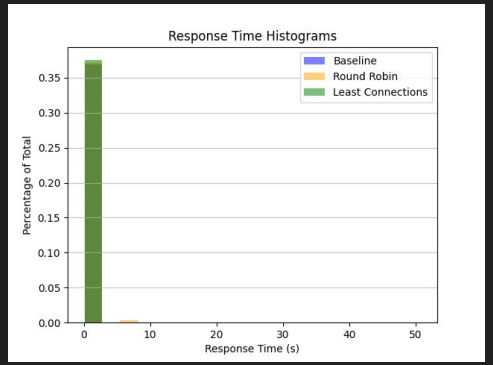
Results and Graphs

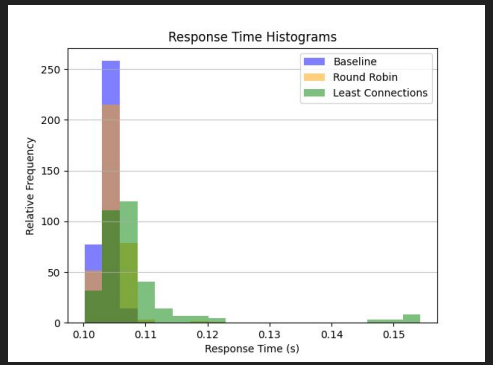




Results and Graphs

- Baseline: 56.3 seconds
- Round Robin: 13.6 seconds
- Least Connections: 2.63 seconds
- For the majority of requests, the load balancer algorithm doesn’t affect response time
- When the server is overloaded, the load balancer is very important, and thus contributes greatly to the total time
- Without the simulated server load, we largely got similar results. This may indicate that a good load balancer becomes more and more important as the scale increases





Conclusion

- Load balancing is critical in distributed systems, particularly under high server load
- A well-designed load balancer can significantly improve performance and ensure fairness among servers
- Least Connections algorithm proved to be the most effective in our project

Future Work

- Investigate other load balancing algorithms (e.g., Weighted Round Robin, Consistent Hashing, etc.)
- Conduct more in-depth analysis of the impact of server processing time and variability on load balancing performance
- Test the performance of the load balancer under different network conditions (e.g., high latency, packet loss)
- Increase reliability through replication and support more complex client operations that may involve server communication and state

Thank You!