Testing Analysis

1. **/rep endpoint `c`**

2. **/heartbeat endpoint e.g for server at `http://localhost:5002/hearbeat` server_2**

```
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ curl -X GET http://localhost:5002/heartbeat
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ curl -X GET http://localhost:5000/rep
{
  "message": {
    "N": 3,
    "replicas": [
      "server_2",
      "server_1",
      "distributedsysserver-load_balancer-1"
    ]
  },
  "status": "successful"
}
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$
```

3. **/home endpoint e.g for server at `http://localhost:5001/home` server_1**

```
}
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ ls
Makefile  code_test  docker-compose.yml  load_balancer_requirements  server_requirements
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ curl -X GET http://localhost:5001/home
{"message":"Hello from Server: 1","status":"successful"}
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$
```

4. **/add endpoint**

Provide the n field and a list of hostnames e.g adding 4:

curl -X POST http://localhost:5000/add -H "Content-Type: application/json" -d '{

"n": 4,

"hostnames": ["server_4", "server_5", "server_6", "server_7"]

}'

```
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ curl -X POST http://localhost:5000/add -H "Content-Type: application/json" -d '{
    "n": 4,
    "hostnames": ["server_4", "server_5", "server_6", "server_7"]
}'
{
nugi@DESKTOP-QVVGJA3:~/DistributedSysServer$ docker ps -a
CONTAINER ID   IMAGE                   COMMAND                CREATED        STATUS        PORTS
         NAMES
e2d3ab48640c   myproject_server        "python server.py"     22 hours ago   Created
         server_8
4a0c956f46e1   myproject_server        "python server.py"     22 hours ago   Created
         server_3
fa866b3bd2ec   myproject_server        "python server.py"     22 hours ago   Created
         server_7
e01eec4aeef9   myproject_server        "python server.py"     22 hours ago   Created
         server_6
79aed9ac6a8a   myproject_server        "python server.py"     22 hours ago   Created
         server_5
2ad3bbf4d9e7   myproject_server        "python server.py"     22 hours ago   Created
         server_4
2e1b05f7515d   myproject_server        "python server.py"     22 hours ago   Up 22 hours   0.0.0.0:5002->5000/tcp, :::5002->500
0/tcp   server_2
f5de625ceafa   myproject_server        "python server.py"     22 hours ago   Up 22 hours   0.0.0.0:5001->5000/tcp, :::5001->500
0/tcp   server_1
58747acebe25   myproject_load_balancer "python load_balance…" 22 hours ago   Up 22 hours   0.0.0.0:5000->5000/tcp, :::5000->500
0/tcp   distributedsysserver-load_balancer-1
```

Testing Load balancing

Load Distribution Among 3 Servers**

Observations

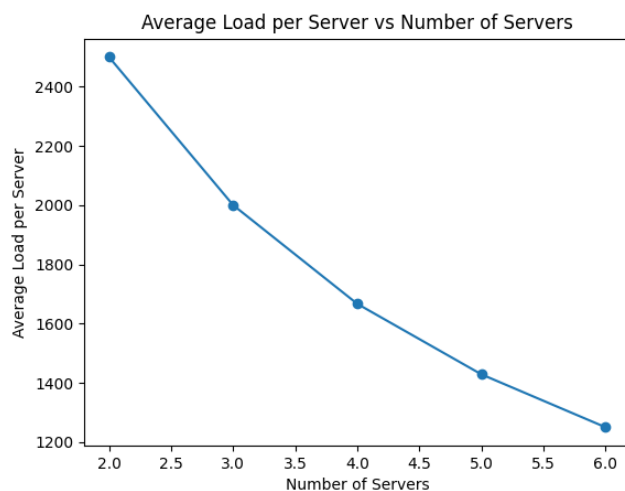Requests Handled by Each Server (N=3)
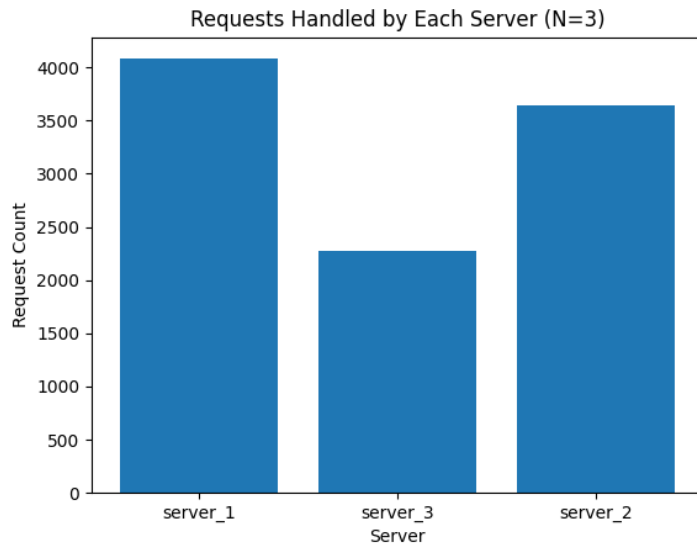
Analysis

- The load distribution is uneven, with `server_1` handling the most requests and `server_3` handling the least.

- Possible reasons for this discrepancy could include the network latency, or environmental factors.

Scalability with Incrementing Servers N from 2 to 6**

Observations

Average Load per Server vs Number of Servers

Analysis

- The average load per server decreases as the number of servers increases.

- The load balancer scales efficiently with more servers.

Load Balancer Recovery from Server Failure

Observations

Initial Requests with addition of 4 Servers

server_4: 1 request

server_6: 3 requests

server_1: 1 request

server_2 : 2 requests

server_5: 1 request

server_7: 2 requests

Post-Failure Requests with deletion of 2 Servers

server_1: 2 requests

server_2: 3 requests

server_7: 4 requests

server_4: 1 request

Observations:

- The load balancer quickly detected the removal of 2 servers and redistributed the load.

- The response times remained stable, indicating efficient handling of server failures.

Scaling Down Test

Initial Requests with addition of 6 Servers

server_8: 2 requests

server_4: 1 request

server_10: 2 requests

server_9: 1 request

server_2: 4 requests

Post-Scaling Requests with removing 2 Servers

server_10: 2 requests

server_12: 2 requests

server_11: 2 requests

server_9: 2 requests
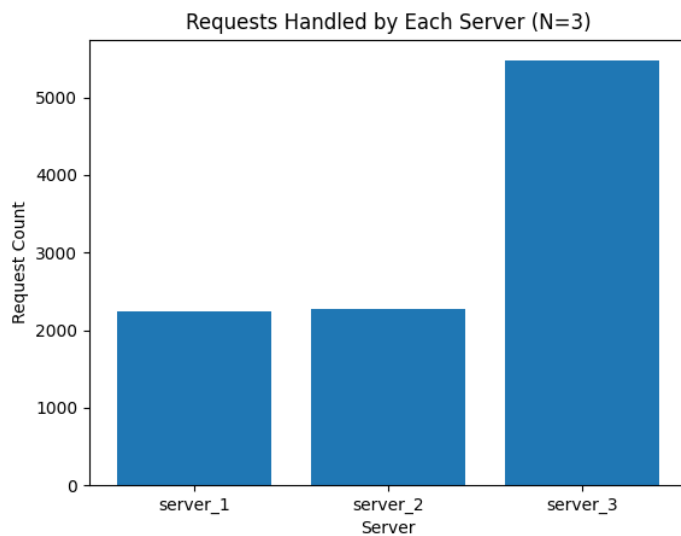
server_2: 3 requests

Observations:

- The load balancer effectively scaled down when deletion of 2 servers.

- The load distribution post-scaling was balanced, and the system maintained the performance.

Testing Load balancing for A-4 modifing the hash and virtual server functions using md5

Load Distribution Among 3 Servers

Observations

Requests Handled by Each Server (N=3)

Analysis

- The MD5 hash function resulted in an imbalanced load distribution among the servers, with one server handling a significantly higher number of requests.

- The load distribution with the MD5 hash function is less balanced compared to the original SHA-256 based hash function.

- This imbalance indicates that the MD5 hash function might not be as effective in distributing requests evenly across servers.

A-2 Scalability with Incrementing Servers N from 2 to 6

Observations

Average Load per Server vs Number of Servers
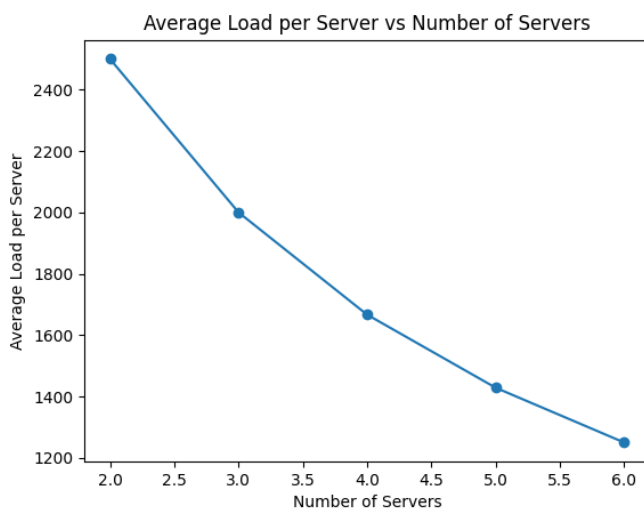
Analysis

- The average load per server decreases as the number of servers increases, indicating good scalability.

- Despite the imbalances observed in A-1, the system scales well with the MD5 hash function, distributing the load across an increasing number of servers.