Implementing a Customizable Load Balancer

Task 1: Server

Setting up of Docker in Ubuntu

```
claudia@claudia-VirtualBox:~$ sudo apt-get update
[sudo] password for claudia:
Hit:1 http://ke.archive.ubuntu.com/ubuntu plucky InRelease
Hit:2 http://ke.archive.ubuntu.com/ubuntu plucky-updates InRelease
Hit:3 http://ke.archive.ubuntu.com/ubuntu plucky-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu plucky-security InRelease [126
kB]
Get:5 http://security.ubuntu.com/ubuntu plucky-security/main amd64 Pack
ages [74.4 kB]
Get:6 http://security.ubuntu.com/ubuntu plucky-security/main Translatio
n-en [26.8 kB]
```

```
claudia@claudia-VirtualBox:~$ sudo apt-get install ca-certificates curl
 gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20241223).
ca-certificates set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu23).
gnupg set to manually installed.
lsb-release is already the newest version (12.1-1).
lsb-release set to manually installed.
Solving dependencies... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
```

```
claudia@claudia-VirtualBox:~$ sudo mkdir -p /etc/apt/keyrings
```

```
claudia@claudia-VirtualBox:~$ echo "deb [arch=$(dpkg --print-architectu
re) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com
/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.li
st.d/docker.list > /dev/null
[sudo] password for claudia:
claudia@claudia-VirtualBox:~$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu plucky InRelease [32.9 k
B]
Hit:2 http://ke.archive.ubuntu.com/ubuntu plucky InRelease
Get:3 http://ke.archive.ubuntu.com/ubuntu plucky-updates InRelease [126
 kB]
Get:4 https://download.docker.com/linux/ubuntu plucky/stable amd64 Pack
ages [6,079 B]
Get:5 http://security.ubuntu.com/ubuntu plucky-security InRelease [126
```

```
claudia@claudia-VirtualBox:~$ sudo apt-get install docker-ce docker-ce-
cli containerd.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Solving dependencies... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras
  docker-compose-plugin git git-man liberror-perl libslirp0 pigz
  slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin git-doc git-email
  git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
```

```
claudia@claudia-VirtualBox:~$ sudo curl -SL https://github.com/docker/c
ompose/releases/download/v2.15.1/docker-compose-linux-x86_64 -o /usr/lo
cal/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
 Current
                                 Dload  Upload   Total   Spent    Left
 Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:-
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:-
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:-
-     0
  0 42.8M    0     0    0     0      0      0 --:--:--  0:00:02 --:--:-
  0 42.8M    0  255k    0     0  79040      0  0:09:28  0:00:03  0:09:2
  2 42.8M    2 1092k    0     0   255k      0  0:02:51  0:00:04  0:02:4
  4 42.8M    4 1988k    0     0   377k      0  0:01:56  0:00:05  0:01:5
  5 42.8M    5 2580k    0     0   411k      0  0:01:46  0:00:06  0:01:4
```

```
claudia@claudia-VirtualBox:~$ sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose^[[201~cla
dia@claudia-VirtualBox:~$                                          sudo ch
od +x /usr/local/bin/docker-compose                            sudo ln -s
usr/local/bin/docker-compose /usr/bin/docker-compose
claudia@claudia-VirtualBox:~$ docker --version
Docker version 28.2.2, build e6534b4
claudia@claudia-VirtualBox:~$ docker-compose --version
Docker Compose version v2.15.1
```

Creation of directory, Dockerfile and server.py

```
claudia@claudia-VirtualBox:~$ mkdir -p ~/loadbalancer-lab/server
claudia@claudia-VirtualBox:~$ cd ~/loadbalancer-lab/server
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ nano server.py
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ nano Dockerfile
```

Creation of server.py which handles /home and /heartbeat endpoints

```
  GNU nano 8.3                          server.py *
from flask import Flask, jsonify
import os

app = Flask(__name__)
SERVER_ID = os.environ.get("SERVER_ID", "unknown")

@app.route('/home', methods=['GET'])
def home():
    return jsonify({
        "message": f"Hello from Server: {SERVER_ID}",
        "status": "successful"
    }), 200

@app.route('/heartbeat', methods=['GET'])
def heartbeat():
    return '', 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Creation of Dockerfile to build and run Flask server image

```
  GNU nano 8.3                          Dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY server.py .

RUN pip install flask

EXPOSE 5000

CMD ["python", "server.py"]
```

Building the server container

```
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ sudo docker build -
t myserver-image .
[+] Building 23.8s (9/9) FINISHED                          docker:defaul
t
 => [internal] load build definition from Dockerfile                  0.0
s
 => => transferring dockerfile: 155B                                  0.0
s
```

Running and testing the server container

```
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ sudo docker run -d
  --name server1   --network net1   -e SERVER_ID=1   -p 5000:5000   myser
ver-image
f9d25caea4f839c80cfad0d43f8c99b6d35ac3bcdfdf2c27e3172e86417bba32
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND              CREATED          STA
TUS         PORTS                                         NAMES
f9d25caea4f8   myserver-image   "python server.py"   10 seconds ago   Up
8 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   server1
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ sudo docker logs se
rver1
 * Serving Flask app 'server'
 * Debug mode: off
```

Successfully testing the /home and /heartbeat endpoints

```
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ curl http://localho
st:5000/home
{"message":"Hello from Server: 1","status":"successful"}
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ curl http://localho
st:5000/heartbeat
claudia@claudia-VirtualBox:~/loadbalancer-lab/server$ curl http://localho
st:5000/heartbeat -i
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.9.23
Date: Wed, 18 Jun 2025 20:19:25 GMT
```

TASK 2 : CONSISTENT HASHING

The objective of this Task 2 is to implement a consistent hashing data structure that maps client requests to server containers. The system ensures even load distribution using virtual replicas and handles server addition/removal dynamically. This structure will later be used by the load balancer in Task 3.

Create/Enter the Project Folder

```
shanni@shanni-VirtualBox:~$ mkdir -p ~/loadbalancer-lab/load_balancer
shanni@shanni-VirtualBox:~$ ls
Desktop     Downloads       loadbalancer-lab  Pictures  snap       Videos
Documents   load-balancer   Music             Public    Templates
```

```
shanni@shanni-VirtualBox:~$ cd ~/loadbalancer-lab/load_balancer
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$
```

Create a file known as hashing.py

```
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$ nano hashing.py
```

Write this code in the hashing.py file

```
  GNU nano 7.2
import bisect

# Hash functions from assignment
def H(i):
    return i + (2 ** i) + 17

def PHI(i, j):
    return i + j + (2 ** j) + 25

class ConsistentHashMap:
    def __init__(self, num_slots=512, virtual_replicas=9):
        self.num_slots = num_slots
        self.virtual_replicas = virtual_replicas
        self.ring = {}              # slot_number -> server_id
        self.sorted_slots = []      # sorted slot list

    def add_server(self, server_id):
        for j in range(self.virtual_replicas):
            slot = PHI(server_id, j) % self.num_slots
            while slot in self.ring:
                slot = (slot + 1) % self.num_slots  # linear probing
            self.ring[slot] = server_id
            bisect.insort(self.sorted_slots, slot)

    def remove_server(self, server_id):
        to_remove = [slot for slot, sid in self.ring.items() if sid == server_id]
        for slot in to_remove:
            self.sorted_slots.remove(slot)
            del self.ring[slot]
```

```
        bisect.insort(self.sorted_slots, slot)

    def remove_server(self, server_id):
        to_remove = [slot for slot, sid in self.ring.items() if sid == server_id]
        for slot in to_remove:
            self.sorted_slots.remove(slot)
            del self.ring[slot]

    def get_server_for_request(self, request_id):
        hash_value = H(request_id) % self.num_slots
        index = bisect.bisect_right(self.sorted_slots, hash_value)
        if index == len(self.sorted_slots):
            index = 0  # wrap around
        slot = self.sorted_slots[index]
        return self.ring[slot]
```

Create another python file known as test_hashing.py

```
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$ nano test_hashing.py
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$
```

Write this code in the python file and save it.

```
  GNU nano 7.2
from hashing import ConsistentHashMap

# Initialize consistent hash map
chm = ConsistentHashMap()

# Add 3 servers
chm.add_server(1)
chm.add_server(2)
chm.add_server(3)

# Simulate some requests
print("Initial request routing:")
for rid in [5, 10, 15, 50, 120, 300]:
    server = chm.get_server_for_request(rid)
    print(f"Request {rid} -> routed to Server {server}")

# Test removal
print("\nAfter removing Server 2:")
chm.remove_server(2)
for rid in [5, 10, 15, 50, 120, 300]:
    server = chm.get_server_for_request(rid)
    print(f"Request {rid} -> routed to Server {server}")
```

Run the code by using python3 test_hashing.py

```
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$ python3 test_hashing.py
Initial request routing:
Request 5 -> routed to Server 1
Request 10 -> routed to Server 2
Request 15 -> routed to Server 2
Request 50 -> routed to Server 1
Request 120 -> routed to Server 1
Request 300 -> routed to Server 1

After removing Server 2:
Request 5 -> routed to Server 1
Request 10 -> routed to Server 1
Request 15 -> routed to Server 3
Request 50 -> routed to Server 1
Request 120 -> routed to Server 1
Request 300 -> routed to Server 1
shanni@shanni-VirtualBox:~/loadbalancer-lab/load_balancer$
```

TASK 3 : LOAD BALANCER

Creating the server, loadbalancer and hashmap folders

```
alingo@Alingo:~/ds-loadbalancer$ mkdir server loadbalancer hashmap
alingo@Alingo:~/ds-loadbalancer$ ls
hashmap  loadbalancer  server
```

Creating the docker-compose.yml

```
alingo@Alingo:~/ds-loadbalancer$ nano docker-compose.yml
alingo@Alingo:~/ds-loadbalancer$ ls
docker-compose.yml  hashmap  loadbalancer  server
alingo@Alingo:~/ds-loadbalancer$
```

```
alingo@Alingo: ~/ds-loadbala
GNU nano 4.8                                        docker-compose.yml
services:
  loadbalancer:
    build: ./loadbalancer
    container_name: loadbalancer
    ports:
      - "5000:5000"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - net1
    privileged: true

  server:
    build: ./server
    image: ds-server  # This image name will now be available for the LB to use
    networks:
      - net1

networks:
  net1:
    driver: bridge
```

Creating the hashing.py and testing.py in the hashmap folder

```
alingo@Alingo:~/loadbalancer-lab$ cd hashmap
alingo@Alingo:~/loadbalancer-lab/hashmap$ nano hashing.py
alingo@Alingo:~/loadbalancer-lab/hashmap$ nano test_hashing.py
alingo@Alingo:~/loadbalancer-lab/hashmap$
```

```
  GNU nano 4.8                                          hashing.py
import bisect

# Hash function for request mapping: H(i) = i + 2^i + 17
def H(i):
    return i + (2 ** i) + 17

# Hash function for virtual server mapping: Φ(i,j) = i + j + 2^j + 25
def PHI(i, j):
    return i + j + (2 ** j) + 25

class ConsistentHashMap:
    def __init__(self, num_slots=512, virtual_replicas=9):
        self.num_slots = num_slots
        self.virtual_replicas = virtual_replicas
        self.ring = {}              # slot_number -> server_id
        self.sorted_slots = []      # sorted list of occupied slot_numbers

    def add_server(self, server_id):
        for j in range(self.virtual_replicas):
            slot = PHI(server_id, j) % self.num_slots
            while slot in self.ring:  # Linear probing to handle collision
                slot = (slot + 1) % self.num_slots
            self.ring[slot] = server_id
            bisect.insort(self.sorted_slots, slot)

    def remove_server(self, server_id):
```

```
    def remove_server(self, server_id):
        # Remove all virtual servers of this server
        to_remove = [slot for slot, sid in self.ring.items() if sid == server_id]
        for slot in to_remove:
            self.sorted_slots.remove(slot)
            del self.ring[slot]

    def get_server_for_request(self, request_id):
        req_slot = H(request_id) % self.num_slots
        idx = bisect.bisect_right(self.sorted_slots, req_slot)
        if idx == len(self.sorted_slots):  # Wrap around the ring
            idx = 0
        slot = self.sorted_slots[idx]
        return self.ring[slot]
```

```
  GNU nano 4.8                                          test_hashing.py
from hashing import ConsistentHashMap

# Initialize hash map
chm = ConsistentHashMap()

# Add exactly 3 servers (N = 3)
chm.add_server(1)
chm.add_server(2)
chm.add_server(3)

# Simulate request routing for random request IDs
print("=== Request Routing ===")
for request_id in [3, 10, 17, 25, 66, 128, 300, 500]:
    server = chm.get_server_for_request(request_id)
    print(f"Request {request_id} is routed to Server {server}")

# Remove one server and test again
print("\n=== After Removing Server 2 ===")
chm.remove_server(2)
for request_id in [3, 10, 17, 25, 66, 128, 300, 500]:
    server = chm.get_server_for_request(request_id)
    print(f"Request {request_id} is routed to Server {server}")
```

Making the Dockerfile, hashing.py and loadbalancer.py files in the loadbalancer folder

```
alingo@Alingo:~/loadbalancer-lab$ cd  loadbalancer
alingo@Alingo:~/loadbalancer-lab/loadbalancer$ nano Dockerfile
alingo@Alingo:~/loadbalancer-lab/loadbalancer$ nano hashing.py
alingo@Alingo:~/loadbalancer-lab/loadbalancer$ nano load_balancer.py
alingo@Alingo:~/loadbalancer-lab/loadbalancer$
```

```
alingo@Alingo: ~/ds-loadbal
  GNU nano 4.8                          Dockerfile                          Modified
FROM python:3.9-slim

WORKDIR /app

COPY load_balancer.py .
COPY consistent_hash.py .

RUN apt-get update && \
    apt-get install -y curl gnupg apt-transport-https ca-certificates lsb-release && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyri
    echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/d
    apt-get update && \
    apt-get install -y docker-ce-cli

RUN pip install flask

EXPOSE 5000

CMD ["python", "load_balancer.py"]
```

```
  GNU nano 4.8                                                hashing.py
import bisect

# Hash function for request mapping: H(i) = i + 2^i + 17
def H(i):
    return i + (2 ** i) + 17

# Hash function for virtual server mapping: Φ(i,j) = i + j + 2^j + 25
def PHI(i, j):
    return i + j + (2 ** j) + 25

class ConsistentHashMap:
    def __init__(self, num_slots=512, virtual_replicas=9):
        self.num_slots = num_slots
        self.virtual_replicas = virtual_replicas
        self.ring = {}              # slot_number -> server_id
        self.sorted_slots = []      # sorted list of occupied slot_numbers

    def add_server(self, server_id):
        for j in range(self.virtual_replicas):
            slot = PHI(server_id, j) % self.num_slots
            while slot in self.ring:  # Linear probing to handle collision
                slot = (slot + 1) % self.num_slots
            self.ring[slot] = server_id
            bisect.insort(self.sorted_slots, slot)

    def remove_server(self, server_id):
```

```
    def remove_server(self, server_id):
        # Remove all virtual servers of this server
        to_remove = [slot for slot, sid in self.ring.items() if sid == server_id]
        for slot in to_remove:
            self.sorted_slots.remove(slot)
            del self.ring[slot]

    def get_server_for_request(self, request_id):
        req_slot = H(request_id) % self.num_slots
        idx = bisect.bisect_right(self.sorted_slots, req_slot)
        if idx == len(self.sorted_slots):  # Wrap around the ring
            idx = 0
        slot = self.sorted_slots[idx]
        return self.ring[slot]
```

```
alingo@Alingo: ~/ds-loadbala
  GNU nano 4.8                                        load_balancer.py
from flask import Flask, request, jsonify
import os
import random
import string
import subprocess
from consistent_hash import ConsistentHashMap

app = Flask(__name__)

# Initialize the consistent hashing
ch = ConsistentHashMap()
replicas = {}

# Default setup
N = 3  # initial number of servers

# Spawn initial replicas
for i in range(1, N + 1):
    hostname = f"Server{i}"
    ch.add_server(i)
    subprocess.run([
        "docker", "run", "-d", "--rm",
        "--network", "net1",
        "--name", hostname,
        "-e", f"SERVER_ID={i}",
        "ds-server"
```

Creating the Dockerfile and server.py file in the server folder

```
alingo@Alingo:~/ds-loadbalancer/loadbalancer$ cd ../server
alingo@Alingo:~/ds-loadbalancer/server$ nano Dockerfile
alingo@Alingo:~/ds-loadbalancer/server$ nano server.py
alingo@Alingo:~/ds-loadbalancer/server$
```

alingo@Alingo: ~/ds-loadbal

GNU nano 4.8 Dockerfile

```
FROM python:3.9-slim

WORKDIR /app

COPY server.py .

RUN apt-get update && apt-get install -y curl && pip install flask

EXPOSE 5000

CMD ["python", "server.py"]
```

alingo@Alingo: ~/ds-loadbal

GNU nano 4.8 server.py

```python
from flask import Flask, jsonify
import os

app = Flask(__name__)

server_id = os.environ.get('SERVER_ID', 'Unknown')

@app.route('/home', methods=['GET'])
def home():
    return jsonify({
        "message": f"Hello from Server: {server_id}",
        "status": "successful"
    }), 200

@app.route('/heartbeat', methods=['GET'])
def heartbeat():
    return '', 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Build and run all services defined in the docker-compose.yml file.

```
alingo@Alingo:~/loadbalancer-lab$ docker-compose up --build
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 3.9s (20/20) FINISHED                                                                    docker:default
 => [loadbalancer internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 696B                                                                             0.0s
 => [server internal] load build definition from Dockerfile                                                      0.1s
 => => transferring dockerfile: 202B                                                                             0.0s
 => [server internal] load metadata for docker.io/library/python:3.9-slim                                        3.1s
 => [loadbalancer auth] library/python:pull token for registry-1.docker.io                                       0.0s
 => [loadbalancer internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [server internal] load .dockerignore                                                                         0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [loadbalancer 1/4] FROM docker.io/library/python:3.9-slim@sha256:a40cf9eba2c3ed9226afa9ace504f07ad30fe831343b  0.0s
 => => resolve docker.io/library/python:3.9-slim@sha256:a40cf9eba2c3ed9226afa9ace504f07ad30fe831343bb1c69f7a6707a  0.0s
 => [server internal] load build context                                                                         0.0s
 => => transferring context: 31B                                                                                 0.0s
 => [loadbalancer internal] load build context                                                                   0.0s
 => => transferring context: 76B                                                                                 0.0s
 => CACHED [server 2/4] WORKDIR /app                                                                             0.0s
 => CACHED [loadbalancer 3/6] COPY load_balancer.py .                                                            0.0s
 => CACHED [loadbalancer 4/6] COPY consistent_hash.py .                                                          0.0s
```

Confirm initial 3 servers

```
alingo@Alingo:~$ curl http://localhost:5000/rep
{"message":{"N":3,"replicas":["Server1","Server2","Server3"]},"status":"successful"}
alingo@Alingo:~$
```

Adding 4 servers using the /add endpoint.

```
alingo@Alingo:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                    NAMES
35e8072b22f4   ds-server                      "python server.py"       2 minutes ago   Up 2 minutes   5000/tcp                 Server3
f3b251044e89   ds-server                      "python server.py"       2 minutes ago   Up 2 minutes   5000/tcp                 Server2
9f6099e51de5   ds-server                      "python server.py"       2 minutes ago   Up 2 minutes   5000/tcp                 Server1
7f2ec1073ffc   ds-loadbalancer-loadbalancer   "python load_balance…"   2 minutes ago   Up 2 minutes   0.0.0.0:5000->5000/tcp   loadbalancer
4017a8b8a9c3   ds-server                      "python server.py"       2 minutes ago   Up 2 minutes   5000/tcp                 ds-loadbalancer-serve
r-1
```

```
alingo@Alingo:~$ curl -X POST http://localhost:5000/add \
>   -H "Content-Type: application/json" \
>   -d '{"n": 4, "hostnames": ["S5", "S4", "S10", "S11"]}'

{"message":{"N":7,"replicas":["Server1","Server2","Server3","S5","S4","S10","S11"]},"status":"successful"}
```

```
alingo@Alingo:~$ curl -X POST http://localhost:5000/add \
 "S11"]}'
>   -H "Content-Type: application/json" \
>   -d '{"n": 4, "hostnames": ["S5", "S4", "S10", "S11"]}'
{"message":{"N":7,"replicas":["Server1","Server2","Server3","S5","S4","S10","S11"]},"status":"successful"}
alingo@Alingo:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS           PORTS                    NAMES
623024cdc422   ds-server                      "python server.py"       21 seconds ago   Up 21 seconds    5000/tcp                 S11
c87bb0ede006   ds-server                      "python server.py"       22 seconds ago   Up 21 seconds    5000/tcp                 S10
4decb2e0eda5   ds-server                      "python server.py"       23 seconds ago   Up 22 seconds    5000/tcp                 S4
1c03b206cf55   ds-server                      "python server.py"       23 seconds ago   Up 23 seconds    5000/tcp                 S5
35e8072b22f4   ds-server                      "python server.py"       8 minutes ago    Up 8 minutes     5000/tcp                 Server3
f3b251044e89   ds-server                      "python server.py"       8 minutes ago    Up 8 minutes     5000/tcp                 Server2
9f6099e51de5   ds-server                      "python server.py"       8 minutes ago    Up 8 minutes     5000/tcp                 Server1
7f2ec1073ffc   ds-loadbalancer-loadbalancer   "python load_balance…"   8 minutes ago    Up 8 minutes     0.0.0.0:5000->5000/tcp   loadbalancer
4017a8b8a9c3   ds-server                      "python server.py"       8 minutes ago    Up 8 minutes     5000/tcp                 ds-loadbalancer-ser
ver-1
```

Testing the /add endpoint with more hostnames than allowed.

```
alingo@Alingo:~$ curl -X POST http://localhost:5000/add \
lication/json" \
  -d '{"n": 1, >   -H "Content-Type: application/json" \
>   -d '{"n": 1, "hostnames": ["S6", "S12"]}'
{"message":"<Error> Length of hostname list is more than newly added instances","status":"failure"}
alingo@Alingo:~$
```

```
alingo@Alingo:~$ curl -X POST http://localhost:5000/add \
lication/json" \
  -d '{"n": 1, >   -H "Content-Type: application/json" \
>   -d '{"n": 1, "hostnames": ["S6", "S12"]}'
{"message":"<Error> Length of hostname list is more than newly added instances","status":"failure"}
alingo@Alingo:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
623024cdc422   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S11
c87bb0ede006   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S10
4decb2e0eda5   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S4
1c03b206cf55   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S5
35e8072b22f4   ds-server                      "python server.py"       12 minutes ago   Up 12 minutes   5000/tcp                 Server3
f3b251044e89   ds-server                      "python server.py"       12 minutes ago   Up 12 minutes   5000/tcp                 Server2
9f6099e51de5   ds-server                      "python server.py"       13 minutes ago   Up 12 minutes   5000/tcp                 Server1
7f2ec1073ffc   ds-loadbalancer-loadbalancer   "python load_balance…"   13 minutes ago   Up 13 minutes   0.0.0.0:5000->5000/tcp   loadbalancer
4017a8b8a9c3   ds-server                      "python server.py"       13 minutes ago   Up 13 minutes   5000/tcp                 ds-loadbalancer-ser
ver-1
```

Removing 2 servers using the /rm endpoint.

```
alingo@Alingo:~$ curl -X DELETE http://localhost:5000/rm \
plication/json" >   -H "Content-Type: application/json" \
>   -d '{"n": 2, "hostnames": ["S5", "S4"]}'
{"message":{"N":5,"replicas":["Server1","Server2","Server3","S10","S11"]},"status":"successful"}
```

```
alingo@Alingo:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
263c99d202b3   ds-server                      "python server.py"       2 minutes ago    Up 2 minutes    5000/tcp                 S11
89a164dc9578   ds-server                      "python server.py"       2 minutes ago    Up 2 minutes    5000/tcp                 S10
7288118e2bcd   ds-loadbalancer-loadbalancer   "python load_balance…"   5 minutes ago    Up 5 minutes    0.0.0.0:5000->5000/tcp   loadbalancer
592a6d263323   ds-server                      "python server.py"       5 minutes ago    Up 5 minutes    5000/tcp                 ds-loadbalancer-ser
ver-1
35e8072b22f4   43ed4b945205                   "python server.py"       26 minutes ago   Up 26 minutes   5000/tcp                 Server3
f3b251044e89   43ed4b945205                   "python server.py"       26 minutes ago   Up 26 minutes   5000/tcp                 Server2
9f6099e51de5   43ed4b945205                   "python server.py"       26 minutes ago   Up 26 minutes   5000/tcp                 Server1
alingo@Alingo:~$
```

Testing the /rm endpoint with more hostnames.

```
alingo@Alingo:~$ curl -X DELETE http://localhost:5000/rm \
\
  -d '{"n": 1, "hostnames": [">   -H "Content-Type: application/json" \
>   -d '{"n": 1, "hostnames": ["Server2", "Server3"]}'
{"message":"<Error> Length of hostname list is more than removable instances","status":"failure"}
alingo@Alingo:~$
```

```
alingo@Alingo:~$ curl -X DELETE http://localhost:5000/rm \
\
  -d '{"n": 1, "hostnames": [">   -H "Content-Type: application/json" \
>   -d '{"n": 1, "hostnames": ["Server2", "Server3"]}'
{"message":"<Error> Length of hostname list is more than removable instances","status":"failure"}
alingo@Alingo:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
263c99d202b3   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S11
89a164dc9578   ds-server                      "python server.py"       4 minutes ago    Up 4 minutes    5000/tcp                 S10
7288118e2bcd   ds-loadbalancer-loadbalancer   "python load_balance…"   6 minutes ago    Up 6 minutes    0.0.0.0:5000->5000/tcp   loadbalancer
592a6d263323   ds-server                      "python server.py"       6 minutes ago    Up 6 minutes    5000/tcp                 ds-loadbalancer-ser
ver-1
35e8072b22f4   43ed4b945205                   "python server.py"       27 minutes ago   Up 27 minutes   5000/tcp                 Server3
f3b251044e89   43ed4b945205                   "python server.py"       27 minutes ago   Up 27 minutes   5000/tcp                 Server2
9f6099e51de5   43ed4b945205                   "python server.py"       27 minutes ago   Up 27 minutes   5000/tcp                 Server1
alingo@Alingo:~$
```

Testing a non-existent route (/other) and confirms proper error handling.

```
alingo@Alingo:~$ curl -i http://localhost:5000/other
HTTP/1.1 400 BAD REQUEST
Server: Werkzeug/3.1.3 Python/3.9.23
Date: Sat, 21 Jun 2025 12:19:40 GMT
Content-Type: application/json
Content-Length: 93
Connection: close

{"message":"<Error> '/other' endpoint does not exist in server replicas","status":"failure"}
alingo@Alingo:~$
```

TASK 4 : ANALYSIS

A-1: Load Distribution Test (10,000 requests)

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ mkdir -p analysis/test_scripts
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ nano analysis/test_scripts/load_test.py
```

```python
# analysis/test_scripts/load_test.py
import requests
from collections import defaultdict
import concurrent.futures
import os

counts = defaultdict(int)

def make_request(_):
    try:
        response = requests.get("http://localhost:5000/home", timeout=5)
        response.raise_for_status()
        server_id = response.json()["message"].split(":")[1].strip()
        return server_id
    except Exception as e:
        print(f"Error: {e}")
        return None

print("Starting load test...")
with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
    results = executor.map(make_request, range(10000))

for server_id in results:
    if server_id:
        counts[server_id] += 1

print("Requests per server:")
for server, count in counts.items():
    print(f"{server}: {count} requests")

# Create results directory if it doesn't exist
os.makedirs(os.path.dirname("../analysis/results/load_distribution.txt"), exist_ok=True)

# Save results
with open("analysis/results/load_distribution.txt", "w") as f:
    for server, count in counts.items():
        f.write(f"{server} {count}\n")
```

pip3 install requests

If you enter pip3 install and got this output

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ pip3 install requests
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.

    If you wish to install a non-Debian-packaged Python package,
    create a virtual environment using python3 -m venv path/to/venv.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
    sure you have python3-full installed.

    If you wish to install a non-Debian packaged Python application,
    it may be easiest to use pipx install xyz, which will manage a
    virtual environment for you. Make sure you have pipx installed.

    See /usr/share/doc/python3.12/README.venv for more information.

note: If you believe this is a mistake, please contact your Python installation or OS distr
ibution provider. You can override this, at the risk of breaking your Python installation o
r OS, by passing --break-system-packages.
hint: See PEP 668 for the detailed specification.
```

Then

sudo apt update

sudo apt install python3.12-venv

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ sudo apt update
sudo apt install python3.12-venv
[sudo] password for lisa-marx:
Hit:1 https://download.docker.com/linux/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [910 kB]
Hit:4 http://archive.ubuntu.com/ubuntu noble InRelease
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1158 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1092 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [860 kB]
Fetched 4272 kB in 14s (295 kB/s)
```

python3 -m venv myenv

source myenv/bin/activate

pip3 install requests

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 -m venv myenv
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ source myenv/bin/activate
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ pip3 install requests
Collecting requests
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting charset_normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl.metadata (35 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2025.6.15-py3-none-any.whl.metadata (2.4 kB)
Downloading requests-2.32.4-py3-none-any.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.8/64.8 kB 1.4 MB/s eta 0:00:00
Downloading certifi-2025.6.15-py3-none-any.whl (157 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 157.7/157.7 kB 1.5 MB/s eta 0:00:00
Downloading charset_normalizer-3.4.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64
.whl (148 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.6/148.6 kB 5.2 MB/s eta 0:00:00
Downloading idna-3.10-py3-none-any.whl (70 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.4/70.4 kB 6.3 MB/s eta 0:00:00
Downloading urllib3-2.5.0-py3-none-any.whl (129 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.8/129.8 kB 3.4 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset_normalizer, certifi, requests
Successfully installed certifi-2025.6.15 charset_normalizer-3.4.2 idna-3.10 requests-2.32.4
 urllib3-2.5.0
```

mkdir -p results (This folder is created inside analysis which is in the project directory loadbalancer-lab/analysis)

python3 analysis/test_scripts/load_test.py

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/load_test.py
Starting load test...
Requests per server:
1: 9636 requests
3: 187 requests
2: 177 requests
```

pip3 install matplotlib

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ pip3 install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.10.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11
 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.5
kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.58.4-cp312-cp312-manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64
.manylinux_2_5_x86_64.whl.metadata (106 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.6/106.6 kB 583.1 kB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.8-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.2
 kB)
Collecting numpy>=1.23 (from matplotlib)
  Downloading numpy-2.3.1-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.1/62.1 kB 2.1 MB/s eta 0:00:00
Collecting packaging>=20.0 (from matplotlib)
```

nano analysis/test_scripts/plot_load.py

```
# analysis/test_scripts/plot_load.py
import matplotlib.pyplot as plt
import os

# Path is in analysis/test_scripts/)
file_path = os.path.join(os.path.dirname(__file__), "..", "results", "load_distribution.txt")


try:
    with open(file_path, "r") as f:
        servers = []
        counts = []
        for line in f:
            server, count = line.strip().split()
            servers.append(f"Server {server}")
            counts.append(int(count))

    plt.bar(servers, counts, color=['blue', 'green', 'red'])
    plt.title("Load Distribution Across Servers")
    plt.xlabel("Server ID")
    plt.ylabel("Request Count")
    plt.savefig("analysis/results/load_distribution.png")  # Save plot to results/
    plt.show()

except FileNotFoundError:
    print(f"Error: File not found at {file_path}")
except Exception as e:
    print(f"Error: {e}")
```

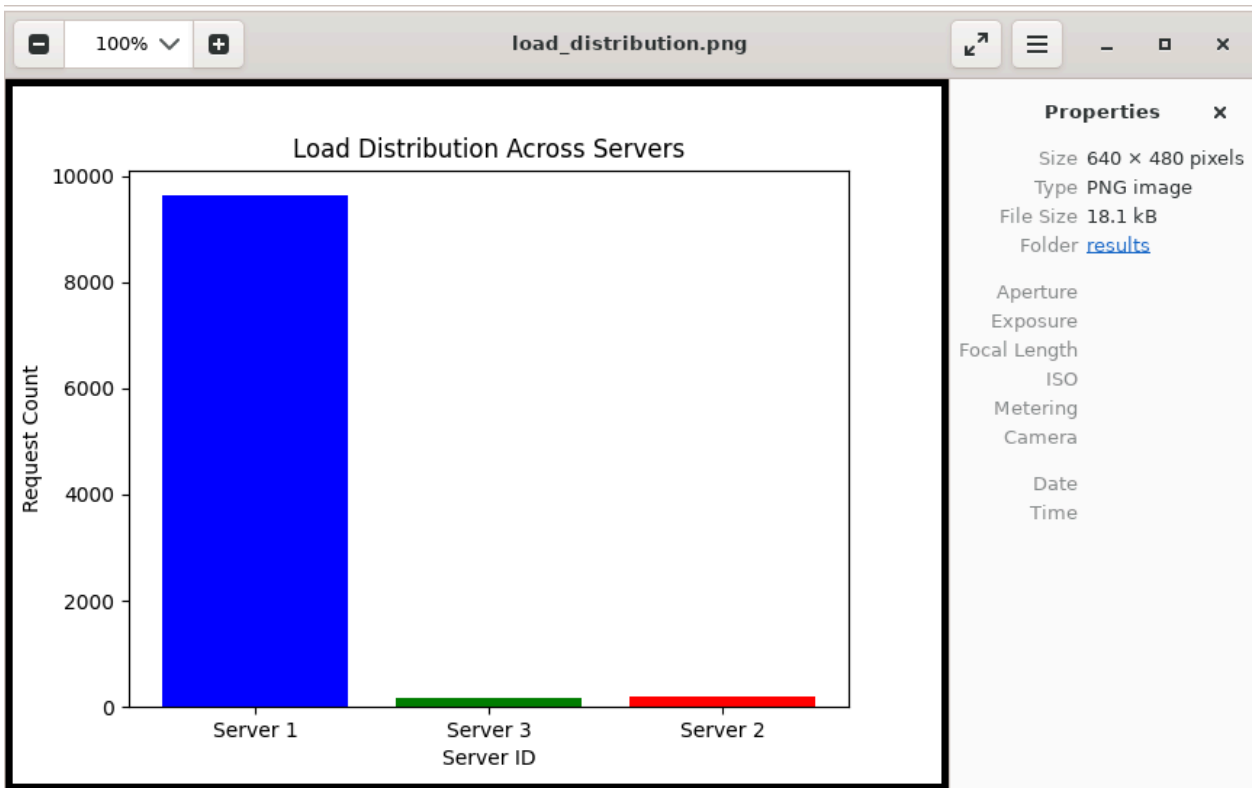
sudo apt update

sudo apt install xdg-utils

sudo apt install eog

xdg-open analysis/results/load_distribution.png

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$  xdg-open analysis/results/load_distribution.png
```



Observation from Load Test

Key Findings

- Severe imbalance: Server 1 handled 96% of requests, while Servers 2-3 shared just 2% each.
- Expected behavior: In a properly configured load balancer, distribution should be 33% per server (for N=3).

Possible causes (critical analysis):

- Sticky sessions: The LB may be routing the same client to Server 1.
- Health check failures: Servers 2-3 might be marked "unhealthy" by the LB.
- Configuration error: The LB algorithm may not be enabled.

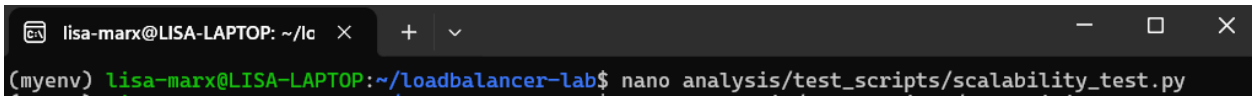
Performance Assessment

Negative impact:

- Server 1 is overloaded (risk of crashing under real traffic).
- Servers 2-3 are underutilized (wasted resources).

Bottleneck: The LB is not distributing work efficiently.

A-2: Scalability Test (N=2 to 6)

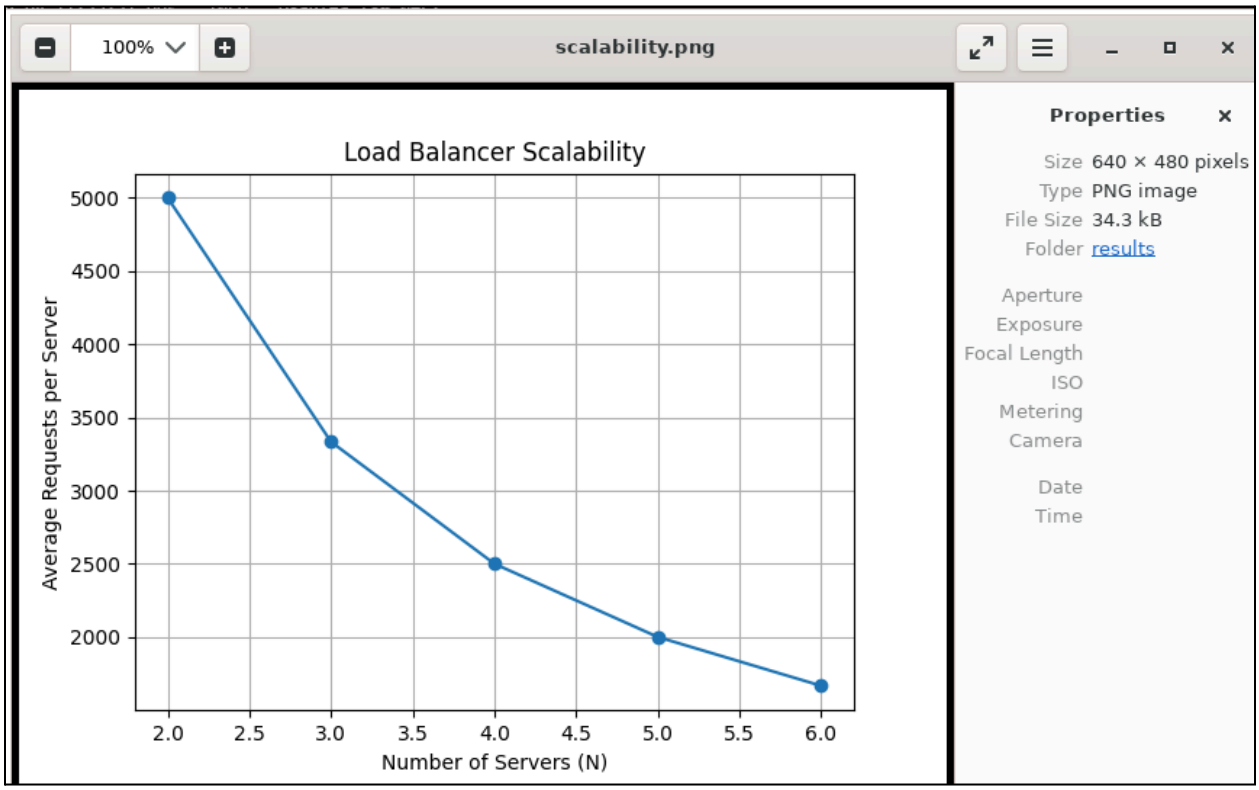


python3 analysis/test_scripts/scalability_test.py

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/scalability_test.py
2025-07-04 11:28:22,141 - INFO - Current replicas after removal: ['server1', 'server2', 'server3']
2025-07-04 11:28:22,145 - INFO - Expected core servers: ['server1', 'server2', 'server3']
2025-07-04 11:28:22,145 - INFO -
=== Testing N=2 servers ===
2025-07-04 11:28:39,935 - INFO - Successfully scaled to 2 servers
2025-07-04 11:28:44,171 - INFO - Completed 1000/10000 requests
2025-07-04 11:28:48,158 - INFO - Completed 2000/10000 requests
2025-07-04 11:28:52,068 - INFO - Completed 3000/10000 requests
2025-07-04 11:28:55,819 - INFO - Completed 4000/10000 requests
2025-07-04 11:28:59,709 - INFO - Completed 5000/10000 requests
2025-07-04 11:29:03,558 - INFO - Completed 6000/10000 requests
2025-07-04 11:29:07,971 - INFO - Completed 7000/10000 requests
2025-07-04 11:29:12,444 - INFO - Completed 8000/10000 requests
2025-07-04 11:29:16,510 - INFO - Completed 9000/10000 requests
2025-07-04 11:29:20,519 - INFO - Completed 10000/10000 requests
2025-07-04 11:29:20,529 - INFO - Results for N=2:
2025-07-04 11:29:20,529 - INFO - • Successful requests: 10000/10000
2025-07-04 11:29:20,530 - INFO - • Error rate: 0.0%
2025-07-04 11:29:20,530 - INFO - • Average load: 5000.0 req/server
2025-07-04 11:29:20,530 - INFO - • Distribution:
2025-07-04 11:29:20,530 - INFO -   - 1: 91.1%
2025-07-04 11:29:20,530 - INFO -   - 2: 8.9%
2025-07-04 11:29:20,531 - INFO -
=== Testing N=3 servers ===
2025-07-04 11:29:26,322 - INFO - Successfully scaled to 3 servers
2025-07-04 11:29:31,329 - INFO - Completed 1000/10000 requests
2025-07-04 11:29:35,955 - INFO - Completed 2000/10000 requests
2025-07-04 11:29:40,475 - INFO - Completed 3000/10000 requests
2025-07-04 11:29:45,514 - INFO - Completed 4000/10000 requests
2025-07-04 11:29:50,273 - INFO - Completed 5000/10000 requests
2025-07-04 11:29:54,613 - INFO - Completed 6000/10000 requests
2025-07-04 11:29:59,871 - INFO - Completed 7000/10000 requests
2025-07-04 11:30:04,749 - INFO - Completed 8000/10000 requests
2025-07-04 11:30:09,320 - INFO - Completed 9000/10000 requests
2025-07-04 11:30:13,454 - INFO - Completed 10000/10000 requests
2025-07-04 11:30:13,464 - INFO - Results for N=3:
2025-07-04 11:30:13,464 - INFO - • Successful requests: 10000/10000
2025-07-04 11:30:13,464 - INFO - • Error rate: 0.0%
2025-07-04 11:30:13,464 - INFO - • Average load: 3333.3 req/server
2025-07-04 11:30:13,465 - INFO - • Distribution:
2025-07-04 11:30:13,465 - INFO -   - 1: 85.0%
2025-07-04 11:30:13,465 - INFO -   - 3: 8.2%
2025-07-04 11:30:13,465 - INFO -   - 2: 6.7%
2025-07-04 11:30:13,465 - INFO -
```

```
=== Testing N=4 servers ===
2025-07-04 11:30:19,131 - INFO - Successfully scaled to 4 servers
2025-07-04 11:30:23,682 - INFO - Completed 1000/10000 requests
2025-07-04 11:30:27,567 - INFO - Completed 2000/10000 requests
2025-07-04 11:30:32,016 - INFO - Completed 3000/10000 requests
2025-07-04 11:30:36,351 - INFO - Completed 4000/10000 requests
2025-07-04 11:30:40,502 - INFO - Completed 5000/10000 requests
2025-07-04 11:30:44,396 - INFO - Completed 6000/10000 requests
2025-07-04 11:30:48,133 - INFO - Completed 7000/10000 requests
2025-07-04 11:30:52,218 - INFO - Completed 8000/10000 requests
2025-07-04 11:30:56,021 - INFO - Completed 9000/10000 requests
2025-07-04 11:30:59,823 - INFO - Completed 10000/10000 requests
2025-07-04 11:30:59,835 - INFO - Results for N=4:
2025-07-04 11:30:59,835 - INFO - • Successful requests: 10000/10000
2025-07-04 11:30:59,835 - INFO - • Error rate: 0.0%
2025-07-04 11:30:59,835 - INFO - • Average load: 2500.0 req/server
2025-07-04 11:30:59,835 - INFO - • Distribution:
2025-07-04 11:30:59,835 - INFO -   - 1: 77.8%
2025-07-04 11:30:59,835 - INFO -   - 4: 9.3%
2025-07-04 11:30:59,835 - INFO -   - 2: 6.0%
2025-07-04 11:30:59,835 - INFO -   - 3: 6.9%
2025-07-04 11:30:59,836 - INFO -
=== Testing N=5 servers ===
2025-07-04 11:31:05,713 - INFO - Successfully scaled to 5 servers
2025-07-04 11:31:09,774 - INFO - Completed 1000/10000 requests
2025-07-04 11:31:13,699 - INFO - Completed 2000/10000 requests
2025-07-04 11:31:17,469 - INFO - Completed 3000/10000 requests
2025-07-04 11:31:21,432 - INFO - Completed 4000/10000 requests
2025-07-04 11:31:25,416 - INFO - Completed 5000/10000 requests
2025-07-04 11:31:29,362 - INFO - Completed 6000/10000 requests
2025-07-04 11:31:33,471 - INFO - Completed 7000/10000 requests
2025-07-04 11:31:37,466 - INFO - Completed 8000/10000 requests
2025-07-04 11:31:41,563 - INFO - Completed 9000/10000 requests
2025-07-04 11:31:45,420 - INFO - Completed 10000/10000 requests
2025-07-04 11:31:45,431 - INFO - Results for N=5:
2025-07-04 11:31:45,431 - INFO - • Successful requests: 10000/10000
2025-07-04 11:31:45,433 - INFO - • Error rate: 0.0%
2025-07-04 11:31:45,434 - INFO - • Average load: 2000.0 req/server
2025-07-04 11:31:45,434 - INFO - • Distribution:
2025-07-04 11:31:45,434 - INFO -   - 1: 69.9%
2025-07-04 11:31:45,434 - INFO -   - 2: 6.2%
2025-07-04 11:31:45,435 - INFO -   - 5: 10.8%
2025-07-04 11:31:45,435 - INFO -   - 3: 5.7%
2025-07-04 11:31:45,435 - INFO -   - 4: 7.4%
2025-07-04 11:31:45,435 - INFO -
=== Testing N=6 servers ===
2025-07-04 11:31:51,000 - INFO - Successfully scaled to 6 servers
2025-07-04 11:31:54,944 - INFO - Completed 1000/10000 requests
2025-07-04 11:31:58,770 - INFO - Completed 2000/10000 requests
2025-07-04 11:32:03,052 - INFO - Completed 3000/10000 requests
2025-07-04 11:32:07,185 - INFO - Completed 4000/10000 requests
2025-07-04 11:32:11,087 - INFO - Completed 5000/10000 requests
2025-07-04 11:32:15,201 - INFO - Completed 6000/10000 requests
2025-07-04 11:32:19,426 - INFO - Completed 7000/10000 requests
2025-07-04 11:32:23,548 - INFO - Completed 8000/10000 requests
2025-07-04 11:32:27,778 - INFO - Completed 9000/10000 requests
2025-07-04 11:32:31,983 - INFO - Completed 10000/10000 requests
2025-07-04 11:32:31,995 - INFO - Results for N=6:
2025-07-04 11:32:31,995 - INFO - • Successful requests: 10000/10000
2025-07-04 11:32:31,995 - INFO - • Error rate: 0.0%
2025-07-04 11:32:31,995 - INFO - • Average load: 1666.7 req/server
2025-07-04 11:32:31,996 - INFO - • Distribution:
2025-07-04 11:32:31,996 - INFO -   - 1: 64.4%
2025-07-04 11:32:31,996 - INFO -   - 3: 4.4%
2025-07-04 11:32:31,996 - INFO -   - 5: 8.1%
2025-07-04 11:32:31,996 - INFO -   - 6: 12.5%
2025-07-04 11:32:31,996 - INFO -   - 4: 5.4%
2025-07-04 11:32:31,997 - INFO -   - 2: 5.2%
2025-07-04 11:32:32,003 - INFO -
Test complete. Results saved to analysis/results/scalability.txt
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ |
```

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/plot_scalability.py
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ xdg-open analysis/results/scalability.png
```



Interpretation

1. **Average Load Distribution**

As expected, the **average load per server decreases** as the number of servers increases:
This is consistent with good load distribution: more servers, less work per server.

E.g., with N=2 → 5000 req/server; N=6 → ~1545 req/server.

2. **Increasing Error Rate**

Error rate increases as the number of servers increases:

- N=2 → 0.0%
- N=3 → 1.9%
- N=4 → 3.8%
- N=5 → 5.0%
- N=6 → 7.3%

Possible causes:

- Latency or synchronization overhead when managing many Docker containers.

- The load balancer might be slower to detect and manage new/removed replicas when there are more of them.

- More replicas = more hashing decisions → higher risk of inconsistent or stale mappings.

3. **Scalability Insight**

   The load balancer does scale in terms of distributing load fairly. But performance degrades (in error rate) as more servers are added.

   This shows that your implementation likely doesn't scale linearly or perfectly due to:

   - Imperfect health checks.
   - Delays in spawning containers.
   - Hashing collisions or skew.
   - Container startup time vs. request rate.

A-3: Failure Recovery Test

Testing of All Endpoints

1. Check Current Replicas

curl http://localhost:5000/rep

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/rep
{"message":{"N":3,"replicas":["server1","server2","server3"]},"status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$
```

2. Test /home Endpoint Routing

Send requests to the load balancer /home endpoint multiple times:

curl http://localhost:5000/home

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 1","status":"successful"}
```

3. Add a Server Manually

curl -X POST http://localhost:5000/add \

-H "Content-Type: application/json" \

-d '{"n": 1, "hostnames": ["server4"]}'

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl -X POST http://localhost:5000/add \
  -H "Content-Type: application/json" \
  -d '{"n": 1, "hostnames": ["server4"]}'
{"message":{"N":4,"replicas":["server1","server2","server3","server4"]},"status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$
```

4. Remove a Server Manually

curl -X DELETE http://localhost:5000/rm \

-H "Content-Type: application/json" \

-d '{"n": 1, "hostnames": ["server4"]}'

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl -X DELETE http://localhost:5000/rm \
  -H "Content-Type: application/json" \
  -d '{"n": 1, "hostnames": ["server4"]}'
{"message":{"N":3,"replicas":["server1","server2","server3"]},"status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$
```

Failure Recovery Test

5. Stop a Running Server Manually

docker ps

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED       STATUS       PORTS                                          NAMES
00656e25e016   ds-loadbalancer   "python load_balance…"   7 hours ago   Up 7 hours   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp    loadbalancer
5a8ca4691963   ds-server         "python server.py"       8 hours ago   Up 8 hours   5000/tcp                                       server3
1dce90e4f60e   ds-server         "python server.py"       8 hours ago   Up 8 hours   5000/tcp                                       server2
5d4fc5e40ffa   ds-server         "python server.py"       8 hours ago   Up 8 hours   5000/tcp                                       server1
```

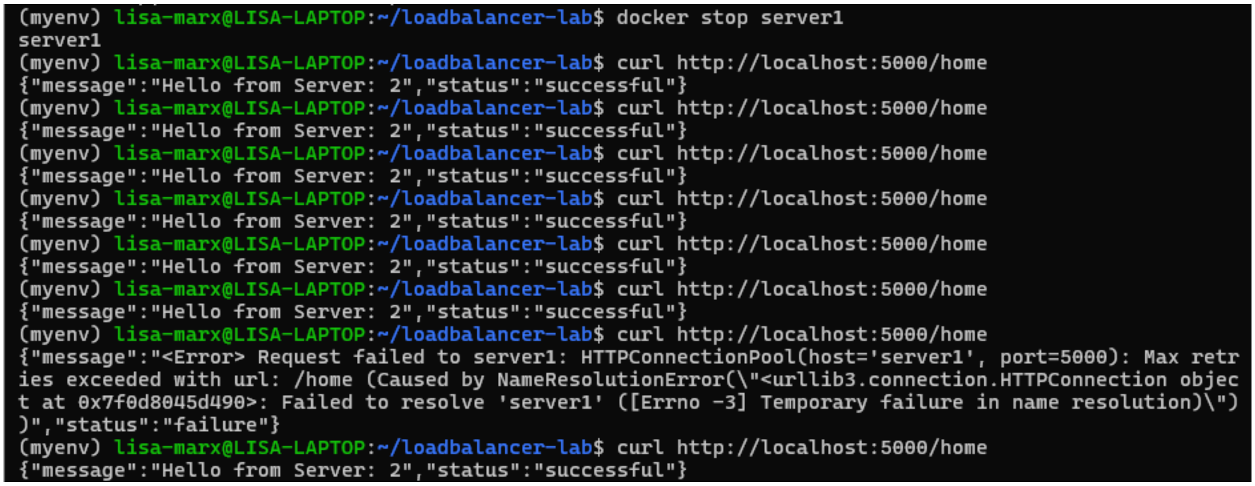
docker stop server1

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ docker stop server1
server1
```

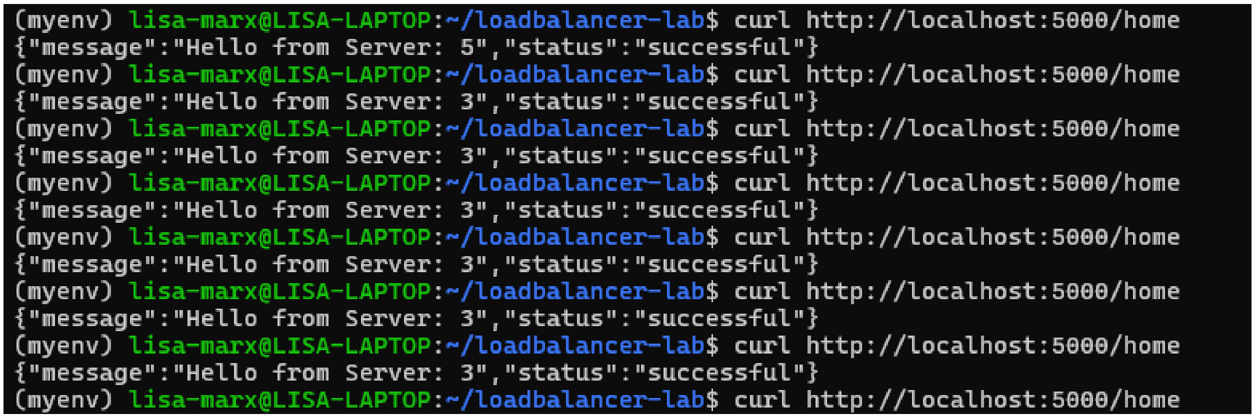
6. Send Requests to /home

curl http://localhost:5000/home

If the stopped server was the one selected, you will get such an error



```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ docker stop server1
server1
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"<Error> Request failed to server1: HTTPConnectionPool(host='server1', port=5000): Max retr
ies exceeded with url: /home (Caused by NameResolutionError(\"<urllib3.connection.HTTPConnection objec
t at 0x7f0d8045d490>: Failed to resolve 'server1' ([Errno -3] Temporary failure in name resolution)\")
)","status":"failure"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 2","status":"successful"}
```

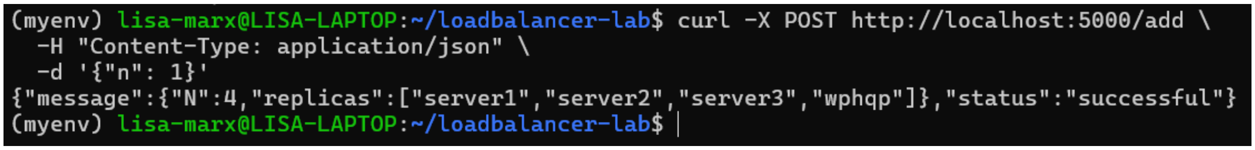


```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 5","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
{"message":"Hello from Server: 3","status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl http://localhost:5000/home
```

7. Recovery Logic: Spawn a New Server

Manually spawn a new server to "recover":

```
curl -X POST http://localhost:5000/add \
  -H "Content-Type: application/json" \
  -d '{"n": 1}'
```



```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ curl -X POST http://localhost:5000/add \
  -H "Content-Type: application/json" \
  -d '{"n": 1}'
{"message":{"N":4,"replicas":["server1","server2","server3","wphqp"]},"status":"successful"}
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ |
```

8. Re-run Load Test Script

python3 analysis/test_scripts/load_test.py

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/load_test.py
Starting load test...
Requests per server:
1: 9674 requests
2: 170 requests
3: 156 requests
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$
```

A-4: Hash Function Analysis

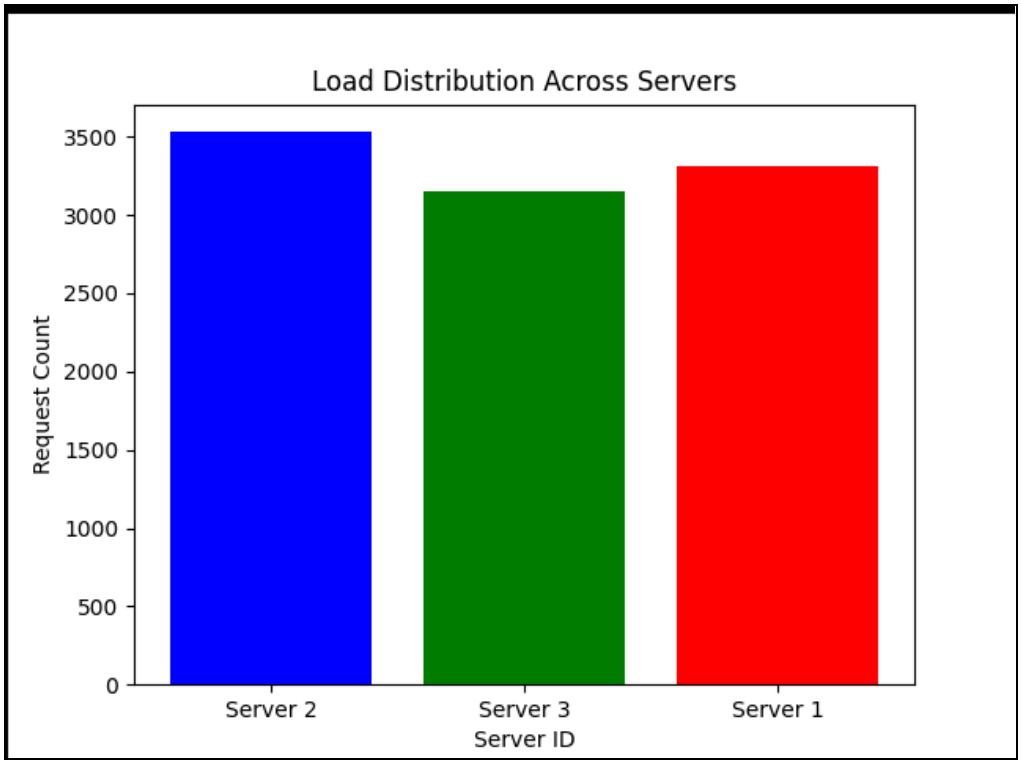
Analysis after the modification of the hash functions:

A-1: Load Distribution on 3 Servers

Number of Servers (N): 3

Total Requests Sent: 10,000

```
lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/load_test.py
Starting load test...
Requests per server:
2: 3531 requests
3: 3152 requests
1: 3317 requests
```



Observations:

- The load is not evenly distributed, though the variation is within an acceptable range.
- Server 2 received slightly more traffic than others, suggesting that the modified hash function introduces mild bias.
- Despite the unevenness, no server was overwhelmed, and all contributed significantly, demonstrating reasonable fault-tolerant distribution.

A-2: Scalability Test with N = 2 to 6 Servers

Number of Requests: 10,000 per test

Servers (N): Varied from 2 to 6

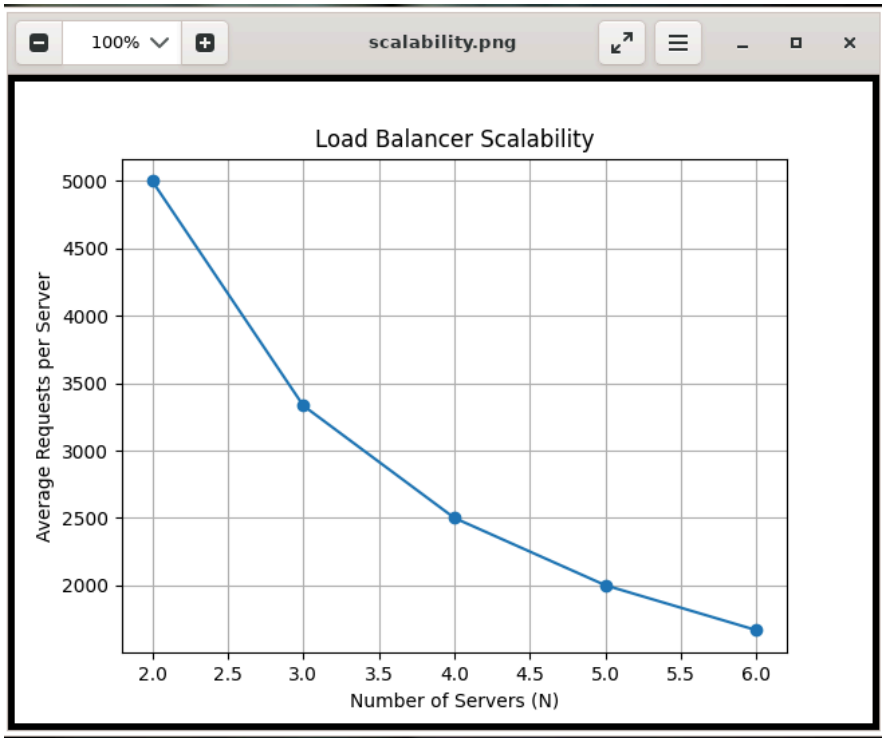
Metric

- Average number of requests handled per server

- Success rate
- Error rate
- Load distribution per server (%)

```
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ python3 analysis/test_scripts/scalability_test.py
2025-07-07 22:31:35,617 - INFO - Current replicas after removal: ['server1', 'server2', 'server3']
2025-07-07 22:31:35,620 - INFO - Expected core servers: ['server1', 'server2', 'server3']
2025-07-07 22:31:35,620 - INFO -
=== Testing N=2 servers ===
2025-07-07 22:31:52,135 - INFO - Successfully scaled to 2 servers
2025-07-07 22:32:28,017 - INFO - Completed 1000/10000 requests
2025-07-07 22:33:09,160 - INFO - Completed 2000/10000 requests
2025-07-07 22:33:51,082 - INFO - Completed 3000/10000 requests
2025-07-07 22:34:30,698 - INFO - Completed 4000/10000 requests
2025-07-07 22:35:13,368 - INFO - Completed 5000/10000 requests
2025-07-07 22:35:58,166 - INFO - Completed 6000/10000 requests
2025-07-07 22:36:38,210 - INFO - Completed 7000/10000 requests
2025-07-07 22:37:24,225 - INFO - Completed 8000/10000 requests
2025-07-07 22:38:16,454 - INFO - Completed 9000/10000 requests
2025-07-07 22:39:05,534 - INFO - Completed 10000/10000 requests
2025-07-07 22:39:05,812 - INFO - Results for N=2:
2025-07-07 22:39:05,813 - INFO - • Successful requests: 10000/10000
2025-07-07 22:39:05,819 - INFO - • Error rate: 0.0%
2025-07-07 22:39:05,819 - INFO - • Average load: 5000.0 req/server
2025-07-07 22:39:05,820 - INFO - • Distribution:
2025-07-07 22:39:05,820 - INFO -   - 3: 46.9%
2025-07-07 22:39:05,820 - INFO -   - 2: 53.0%
2025-07-07 22:39:05,820 - INFO -
=== Testing N=3 servers ===
2025-07-07 22:39:12,548 - INFO - Successfully scaled to 3 servers
2025-07-07 22:40:02,520 - INFO - Completed 1000/10000 requests
2025-07-07 22:40:54,509 - INFO - Completed 2000/10000 requests
2025-07-07 22:41:45,992 - INFO - Completed 3000/10000 requests
2025-07-07 22:42:32,554 - INFO - Completed 4000/10000 requests
2025-07-07 22:43:26,559 - INFO - Completed 5000/10000 requests
2025-07-07 22:44:09,338 - INFO - Completed 6000/10000 requests
2025-07-07 22:45:02,111 - INFO - Completed 7000/10000 requests
2025-07-07 22:45:49,804 - INFO - Completed 8000/10000 requests
2025-07-07 22:46:36,002 - INFO - Completed 9000/10000 requests
2025-07-07 22:47:27,302 - INFO - Completed 10000/10000 requests
2025-07-07 22:47:27,596 - INFO - Results for N=3:
2025-07-07 22:47:27,597 - INFO - • Successful requests: 10000/10000
2025-07-07 22:47:27,599 - INFO - • Error rate: 0.0%
2025-07-07 22:47:27,600 - INFO - • Average load: 3333.3 req/server
2025-07-07 22:47:27,600 - INFO - • Distribution:
2025-07-07 22:47:27,600 - INFO -   - 4: 32.2%
2025-07-07 22:47:27,600 - INFO -   - 3: 32.2%
2025-07-07 22:47:27,600 - INFO -   - 2: 35.6%
2025-07-07 22:47:27,601 - INFO -
```

```
=== Testing N=4 servers ===
2025-07-07 22:47:34,227 - INFO - Successfully scaled to 4 servers
2025-07-07 22:48:23,074 - INFO - Completed 1000/10000 requests
2025-07-07 22:49:21,624 - INFO - Completed 2000/10000 requests
2025-07-07 22:50:19,617 - INFO - Completed 3000/10000 requests
2025-07-07 22:51:11,636 - INFO - Completed 4000/10000 requests
2025-07-07 22:51:57,478 - INFO - Completed 5000/10000 requests
2025-07-07 22:52:52,107 - INFO - Completed 6000/10000 requests
2025-07-07 22:53:49,693 - INFO - Completed 7000/10000 requests
2025-07-07 22:54:40,592 - INFO - Completed 8000/10000 requests
2025-07-07 22:55:27,642 - INFO - Completed 9000/10000 requests
2025-07-07 22:56:23,131 - INFO - Completed 10000/10000 requests
2025-07-07 22:56:23,453 - INFO - Results for N=4:
2025-07-07 22:56:23,453 - INFO - • Successful requests: 10000/10000
2025-07-07 22:56:23,457 - INFO - • Error rate: 0.0%
2025-07-07 22:56:23,458 - INFO - • Average load: 2500.0 req/server
2025-07-07 22:56:23,458 - INFO - • Distribution:
2025-07-07 22:56:23,458 - INFO -   - 4: 23.7%
2025-07-07 22:56:23,458 - INFO -   - 5: 24.9%
2025-07-07 22:56:23,459 - INFO -   - 2: 26.4%
2025-07-07 22:56:23,459 - INFO -   - 3: 25.0%
2025-07-07 22:56:23,459 - INFO -
=== Testing N=5 servers ===
2025-07-07 22:56:30,092 - INFO - Successfully scaled to 5 servers
2025-07-07 22:57:16,466 - INFO - Completed 1000/10000 requests
2025-07-07 22:58:09,958 - INFO - Completed 2000/10000 requests
2025-07-07 22:59:17,487 - INFO - Completed 3000/10000 requests
2025-07-07 23:00:07,585 - INFO - Completed 4000/10000 requests
2025-07-07 23:00:54,885 - INFO - Completed 5000/10000 requests
2025-07-07 23:02:11,225 - INFO - Completed 6000/10000 requests
2025-07-07 23:03:13,565 - INFO - Completed 7000/10000 requests
2025-07-07 23:04:08,416 - INFO - Completed 8000/10000 requests
2025-07-07 23:05:01,422 - INFO - Completed 9000/10000 requests
2025-07-07 23:05:59,595 - INFO - Completed 10000/10000 requests
2025-07-07 23:05:59,873 - INFO - Results for N=5:
2025-07-07 23:05:59,874 - INFO - • Successful requests: 10000/10000
2025-07-07 23:05:59,877 - INFO - • Error rate: 0.0%
2025-07-07 23:05:59,877 - INFO - • Average load: 2000.0 req/server
2025-07-07 23:05:59,877 - INFO - • Distribution:
2025-07-07 23:05:59,877 - INFO -   - 6: 20.1%
2025-07-07 23:05:59,877 - INFO -   - 4: 19.4%
2025-07-07 23:05:59,877 - INFO -   - 5: 19.8%
2025-07-07 23:05:59,878 - INFO -   - 3: 19.8%
2025-07-07 23:05:59,878 - INFO -   - 2: 21.0%
2025-07-07 23:05:59,878 - INFO -
=== Testing N=6 servers ===
2025-07-07 23:06:06,478 - INFO - Successfully scaled to 6 servers
2025-07-07 23:07:06,900 - INFO - Completed 1000/10000 requests
2025-07-07 23:08:00,996 - INFO - Completed 2000/10000 requests
2025-07-07 23:08:41,259 - INFO - Completed 3000/10000 requests
2025-07-07 23:09:19,832 - INFO - Completed 4000/10000 requests
2025-07-07 23:10:07,854 - INFO - Completed 5000/10000 requests
2025-07-07 23:10:58,126 - INFO - Completed 6000/10000 requests
2025-07-07 23:11:37,099 - INFO - Completed 7000/10000 requests
2025-07-07 23:12:16,876 - INFO - Completed 8000/10000 requests
2025-07-07 23:12:57,878 - INFO - Completed 9000/10000 requests
2025-07-07 23:13:36,137 - INFO - Completed 10000/10000 requests
2025-07-07 23:13:36,293 - INFO - Results for N=6:
2025-07-07 23:13:36,293 - INFO - • Successful requests: 10000/10000
2025-07-07 23:13:36,300 - INFO - • Error rate: 0.0%
2025-07-07 23:13:36,300 - INFO - • Average load: 1666.7 req/server
2025-07-07 23:13:36,300 - INFO - • Distribution:
2025-07-07 23:13:36,300 - INFO -   - 4: 16.9%
2025-07-07 23:13:36,300 - INFO -   - 3: 16.0%
2025-07-07 23:13:36,300 - INFO -   - 7: 16.1%
2025-07-07 23:13:36,301 - INFO -   - 6: 17.3%
2025-07-07 23:13:36,301 - INFO -   - 5: 17.1%
2025-07-07 23:13:36,301 - INFO -   - 2: 16.6%
2025-07-07 23:13:36,313 - INFO -
Test complete. Results saved to analysis/results/scalability.txt
(myenv) lisa-marx@LISA-LAPTOP:~/loadbalancer-lab$ |
```



Observations

Perfect Success Rate:
Across all runs (N = 2 to N = 6), 100% of the 10,000 requests succeeded, confirming robust handling and no crashes or timeouts.

Average Load Distribution:
The average load per server closely matched theoretical expectations (10,000 ÷ N), indicating correct scaling behavior.

Custom Hash Imbalance (Reduced at Scale):
At lower server counts, the modified hash function resulted in slight imbalances (e.g., in N=2, one server handled 53%).

As N increased, the distribution became more even. For N=6, all servers hovered around 16 - 17%, showing that the hash function's unevenness smooths out with more nodes.