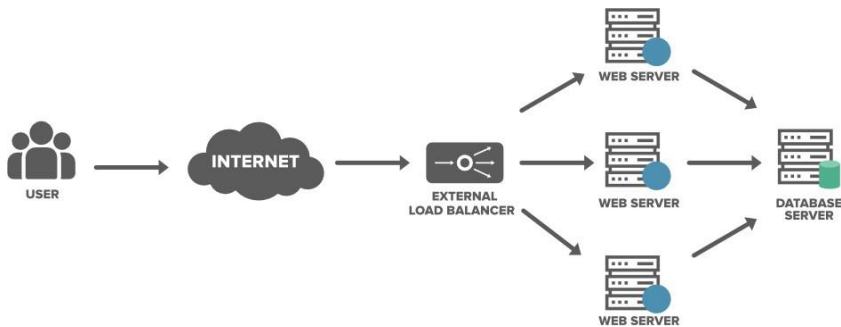




BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

Name	Mayur Solankar, Manish Jadhav, Vishesh Savani, Shreyansh Salvi
UID	2023301018, 2023301005, 2022300100, 2022300091
Subject	Distributed Computing
Experiment No.	4
Project title	Social Media System
Problem Statement	To implement load balancing in social media platform.
Objectives	Make sure everyone gets to see and interact with social media posts quickly and without any problems by sharing the work among different computers, so none of them get too busy or slow down.
Theory	<p>What is load balancing ?</p> <p>A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load adjusting is the approach to conveying load units (i.e., occupations/assignments) across the organization which is associated with the distributed system. Load adjusting should be possible by the load balancer. The load balancer is a framework that can deal with the load and is utilized to disperse the assignments to the servers. The load balancers allocate the primary undertaking to the main server and the second assignment to the second server.</p>  <p><u>Purpose of Load Balancing in Social media platform :</u></p> <ul style="list-style-type: none">• Even Distribution of User Requests: Social media platforms face heavy traffic with millions of users accessing content concurrently. Load balancing helps distribute this load evenly across servers, ensuring no single server gets overwhelmed, preventing slowdowns or crashes.



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

- **Enhanced Performance and Responsiveness:** By spreading the load, servers can handle incoming requests more efficiently. This leads to faster response times, quicker content delivery, and an overall smoother user experience.
- **Optimized Resource Utilization:** Load balancing ensures that resources, such as server capacity and bandwidth, are used effectively. It helps avoid underutilization or overloading of specific servers, maximizing the platform's efficiency.
- **Fault Tolerance and High Availability:** Load balancing can redirect traffic away from failed or underperforming servers to healthy ones. This helps in maintaining the platform's availability and minimizes disruptions in case of server failures or issues.
- **Scalability:** As social media platforms grow, load balancing facilitates easy scaling by allowing new servers to be added to the system. It ensures that the increased traffic is efficiently distributed across the expanded infrastructure.
- In essence, load balancing in social media platforms is vital for ensuring consistent, fast, and reliable access to content while efficiently managing the considerable traffic and user interactions these platforms experience.

Code:

Load balancing:-

```
import http.server
import socketserver
from http import HTTPStatus
import urllib.parse
import requests
import time
from threading import Thread, Lock
import json
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
class DynamicLoadBalancer:
    def __init__(self, servers):
        self.servers = servers
        self.server_health = {server: True for server in servers}
        self.server_connections = {server: 0 for server in servers}
        self.lock = Lock()

    def get_least_busy_server(self):
        with self.lock:
            healthy_servers = [server for server in self.servers if self.server_health[server]]
            if not healthy_servers:
                return None
            return min(healthy_servers, key=lambda s: self.server_connections[s])

    def increment_connections(self, server):
        with self.lock:
            self.server_connections[server] += 1

    def decrement_connections(self, server):
        with self.lock:
            self.server_connections[server] = max(0, self.server_connections[server] - 1)

    def check_server_health(self):
        while True:
            for server in self.servers:
                try:
                    response = requests.get(f"{server}/health", timeout=5)
                    with self.lock:
                        self.server_health[server] = response.status_code == 200
                except:
                    with self.lock:
                        self.server_health[server] = False
            time.sleep(10) # Check health every 10 seconds

class LoadBalancerHandler(http.server.BaseHTTPRequestHandler):
```

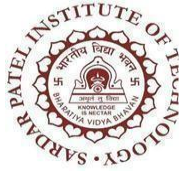


BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
def do_GET(self):
    if self.path == '/health':
        self.send_response(200)
        self.end_headers()
        return

    if self.path == '/stats':
        self.send_response(200)
        self.send_header("Content-type", "application/json")
        self.end_headers()
        stats = {
            "health": load_balancer.server_health,
            "connections": load_balancer.server_connections
        }
        self.wfile.write(json.dumps(stats).encode())
        return

    server_url = load_balancer.get_least_busy_server()
    if server_url:
        load_balancer.increment_connections(server_url)
        try:
            response = requests.get(f"{server_url}{self.path}", timeout=30)
            self.send_response(response.status_code)
            for header, value in response.headers.items():
                self.send_header(header, value)
            self.end_headers()
            self.wfile.write(response.content)
        except:
            self.send_response(HTTPStatus.BAD_GATEWAY)
            self.end_headers()
            self.wfile.write(b"Error communicating with backend server")
        finally:
            load_balancer.decrement_connections(server_url)
    else:
        self.send_response(HTTPStatus.SERVICE_UNAVAILABLE)
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
self.end_headers()
self.wfile.write(b"No healthy servers available")
```

```
servers = ["http://localhost:8001", "http://localhost:8002", "http://localhost:8003"]
load_balancer = DynamicLoadBalancer(servers)
```

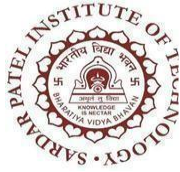
```
# Start health check thread
health_thread = Thread(target=load_balancer.check_server_health)
health_thread.daemon = True
health_thread.start()
```

```
PORT = 8000
with socketserver.TCPServer(("", PORT), LoadBalancerHandler) as httpd:
    print(f"Load balancer serving at port {PORT}")
    httpd.serve_forever()
```

Server 1:-

```
import http.server
import socketserver
import time
import random
```

```
class ServerHandler(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/health':
            self.send_response(200)
            self.send_header("Content-type", "text/plain")
            self.end_headers()
            self.wfile.write(b"OK")
        else:
            # Simulate varying processing time
            processing_time = random.uniform(0.1, 2.0)
            time.sleep(processing_time)
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
self.send_response(200)
self.send_header("Content-type", "text/html")
self.end_headers()
response = f"Hello from Server on port {PORT}! (Processing time:
{processing_time:.2f}s)"
self.wfile.write(response.encode())
```

PORT = 8002 # Change this for each server instance

with socketserver.TCPServer(("", PORT), ServerHandler) as httpd:

```
    print(f"Server serving at port {PORT}")
```

```
    httpd.serve_forever()
```

Server 2:-

```
import http.server
```

```
import socketserver
```

```
import time
```

```
import random
```

```
class ServerHandler(http.server.BaseHTTPRequestHandler):
```

```
    def do_GET(self):
```

```
        if self.path == '/health':
```

```
            self.send_response(200)
```

```
            self.send_header("Content-type", "text/plain")
```

```
            self.end_headers()
```

```
            self.wfile.write(b"OK")
```

```
        else:
```

```
            # Simulate varying processing time
```

```
            processing_time = random.uniform(0.1, 2.0)
```

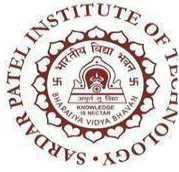
```
            time.sleep(processing_time)
```

```
            self.send_response(200)
```

```
            self.send_header("Content-type", "text/html")
```

```
            self.end_headers()
```

```
            response = f"Hello from Server on port {PORT}! (Processing time:
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
{processing_time:.2f}s"}  
self.wfile.write(response.encode())
```

```
PORT = 8001 # Change this for each server instance  
with socketserver.TCPServer(("", PORT), ServerHandler) as httpd:  
    print(f"Server serving at port {PORT}")  
    httpd.serve_forever()
```

Output:

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\experiment 4> python load_balancing.py  
Load balancer serving at port 8000  
127.0.0.1 - - [18/Sep/2024 19:51:34] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:36] "GET /favicon.ico HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:47] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:50] "GET /favicon.ico HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:56] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:00] "GET /favicon.ico HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:45] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:47] "GET /favicon.ico HTTP/1.1" 200 -  
█
```

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\experiment 4> python server1.py  
Server serving at port 8002  
127.0.0.1 - - [18/Sep/2024 19:51:20] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:26] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:28] "GET /favicon.ico HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:38] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:51:56] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:15] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:33] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:52:51] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:53:09] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:53:27] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:53:46] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:54:04] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:54:22] "GET /health HTTP/1.1" 200 -  
127.0.0.1 - - [18/Sep/2024 19:54:40] "GET /health HTTP/1.1" 200 -  
█
```




BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\experiment 4> python server2.py
Server serving at port 8001
127.0.0.1 - - [18/Sep/2024 19:51:18] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:24] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:34] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:36] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:36] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:50] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:54] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:51:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:00] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:13] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:31] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:47] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:52:49] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:53:07] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:53:25] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:53:44] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:54:02] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:54:20] "GET /health HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 19:54:38] "GET /health HTTP/1.1" 200 -
█
```

Conclusion:

Hence by completing we came to about implementation of Load Balancing in Distributed computing.