

Name: Bhaskar Adhikary

Roll No: G24AI2035

g24ai2035@iitj.ac.in

Indian Institute of Technology Jodhpur

---

## Question 1: Vector Clocks and Causal Ordering

### 1. Introduction

This section covers the implementation of a causally consistent distributed key-value store using vector clocks.

The system is built using Python Flask microservices and orchestrated with Docker Compose across three nodes. Each node maintains a vector clock for version control and causal message handling.

### 2. Step-by-Step Process

Step 1: Create the project directory structure with folders and files: node.py, client.py, Dockerfile, docker-compose.yml

Step 2: Implement node.py with logic to handle writes, reads, vector clock updates, and buffer synchronization.

Step 3: Create client.py to simulate causally related writes and reads across the three nodes.

Step 4: Write a Dockerfile to containerize the Flask node service.

Step 5: Create docker-compose.yml to define three separate nodes: node0, node1, node2 with different ports.

Step 6: Run the containers using the command: docker compose up --build

Step 7: Use python client.py to verify causal consistency in the terminal output.

Step 8: Use browser to open localhost:5000/status, 5001/status, 5002/status to check node states.

### 3. Final Output

All nodes successfully synchronized their vector clocks. The final state for key 'x' was 'B' with vector clock [1, 0, 1] and all buffers were empty, confirming causal consistency.

link- <https://github.com/SUMMERWHEEL11/vector--clock--kv---store>

## 4. Screenshots

```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bhaskar Adhikary>cd C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store\src
C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store\src>python client.py

--- Testing Causal Consistency ---

Step 1: Writing 'A' to key 'x' at node0
Response: {'status': 'written', 'vector_clock': [1, 0, 0]}

Step 2: Reading key 'x' at node1
Response: {'value': 'A', 'vector_clock': [1, 0, 0]}

Step 3: Writing 'B' to key 'x' at node2 (causal dependency)
Response: {'status': 'written', 'vector_clock': [1, 0, 1]}

Step 4: Final status of all nodes

Node0 /status:
{'buffer': [], 'store': {'x': 'B', 'vector_clock': [1, 0, 1]}

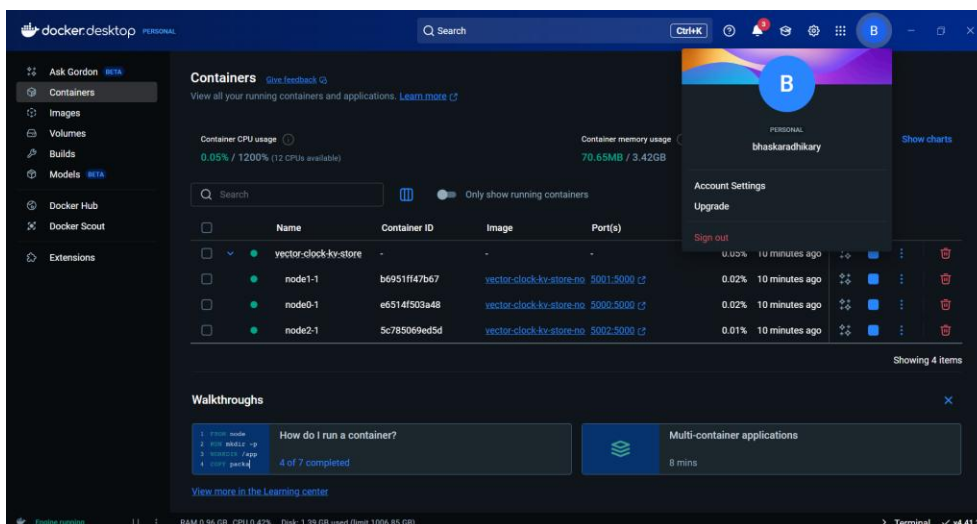
Node1 /status:
{'buffer': [], 'store': {'x': 'B', 'vector_clock': [1, 0, 1]}

Node2 /status:
{'buffer': [], 'store': {'x': 'B', 'vector_clock': [1, 0, 1]}

C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store\src>
```

```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bhaskar Adhikary>cd C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store
C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store>docker compose up
time="2025-06-24T23:38:39+05:30" level=warning msg="C:\Users\Bhaskar Adhikary\Desktop\vector -clock -kv - store\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
  ✓ Container vector-clock-kv-store-node0-1 Created 0.8s
  ✓ Container vector-clock-kv-store-node2-1 Created 0.8s
  ✓ Container vector-clock-kv-store-node1-1 Created 0.8s
Attaching to node0-1, node1-1, node2-1
node2-1 | * Serving Flask app 'node'
node2-1 | * Debug mode: off
node1-1 | * Serving Flask app 'node'
node1-1 | * Debug mode: off
node0-1 | * Serving Flask app 'node'
node0-1 | * Debug mode: off
node2-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node2-1 | * Running on all addresses (0.0.0.0)
node1-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node1-1 | * Running on http://172.0.0.1:5000
node0-1 | * Running on all addresses (0.0.0.0)
node0-1 | * Running on http://172.0.0.2:5000
node1-1 | * Running on http://172.0.0.1:5000
node0-1 | * Running on http://172.0.0.1:5000
node2-1 | Press CTRL+C to quit
node1-1 | * Running on http://172.0.0.4:5000
node0-1 | * Running on http://172.0.0.3:5000
node2-1 | Press CTRL+C to quit
node1-1 | Press CTRL+C to quit
node0-1 | Press CTRL+C to quit
node1-1 | 172.18.0.3 - - [24/Jun/2025 18:01:56] "POST /put HTTP/1.1" 200 -
node2-1 | 172.18.0.3 - - [24/Jun/2025 18:01:56] "POST /put HTTP/1.1" 200 -
node0-1 | 172.18.0.1 - - [24/Jun/2025 18:01:56] "POST /write HTTP/1.1" 200 -
node1-1 | 172.18.0.1 - - [24/Jun/2025 18:01:58] "GET /get/x HTTP/1.1" 200 -
node0-1 | 172.18.0.2 - - [24/Jun/2025 18:01:58] "POST /put HTTP/1.1" 200 -
node2-1 | 172.18.0.1 - - [24/Jun/2025 18:01:58] "POST /write HTTP/1.1" 200 -
```



## Question 2: Dynamic Load Balancing for a Smart Grid

### 1. Introduction

The goal of this project was to simulate a smart grid infrastructure where electric vehicle (EV) charging requests are distributed evenly among multiple substations.

The system comprises multiple containerized services that communicate via REST APIs and includes observability via Prometheus and Grafana.

### 2. Step-by-Step Process

Step 1: Create project directories for each service: `charge_request_service`, `load_balancer`, `substation_service`, `load_tester`, `monitoring`.

Step 2: Implement `main.py` for each service:

- `charge_request_service`: Accepts external requests and forwards them to the load balancer.
- `load_balancer`: Forwards requests to the least-loaded substation based on `/metrics`.
- `substation_service`: Handles the actual charging logic and exposes `/metrics` endpoint.
- `load_tester`: Simulates 20 EV charge requests.

Step 3: Create a Prometheus config file to scrape metrics from substations.

Step 4: Set up Grafana to visualize load trends using dashboards.

Step 5: Write Dockerfiles for each service and create `docker-compose.yml` to orchestrate all services.

Step 6: Start the entire system using `docker compose up --build`.

Step 7: Run `python test.py` from `load_tester` directory to generate load.

Step 8: Open `localhost:9090` for Prometheus and `localhost:3000` for Grafana dashboards.

### 3. Final Output

The requests were evenly distributed across `substation1` and `substation2`. Each handled exactly 10 out of 20 requests. Metrics were visible in Prometheus and Grafana.

Link- <https://github.com/SUMMERWHEEL11/smart-grid-load-balancer>

## 4. Screenshots

```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bhaskar Adhikary>cd C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer

C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer>docker compose up --build
time="2025-06-25T08:13:15+05:30" level=warning msg="C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[*] Running 13/22
- grafana [ ] Pulling 7.6s
  ✓9183b65e98ee Pull complete 2.6s
  ✓7899d5891b77 Download complete 2.6s
  ✓4ae098080e7 Download complete 1.8s
  ✓f18232178bc9 Pull complete 2.5s
  ✓538de38e89c Pull complete 0.8s
  ✓784d5c988dc Downloading [=====] 1.849MB/3.524MB 3.2s
  ✓38bb92ff8688 Downloading [=====] 5.243MB/8.735MB 1.2s
  ✓04f6155c873d Downloading [>] 2.607MB/107.3MB 3.2s
  ✓85de74c6bda Downloading [==>] 2.607MB/63.4MB 3.2s
[*] Running 22/22 Pulling fs layer 3.2s
- grafana Pulled 42.6s
  ✓9183b65e98ee Pull complete 2.6s
  ✓7899d5891b77 Pull complete 1.8s
  ✓4ae098080e7 Pull complete 2.5s
  ✓f18232178bc9 Pull complete 0.8s
  ✓538de38e89c Pull complete 5.8s
  ✓38bb92ff8688 Pull complete 6.8s
  ✓04f6155c873d Pull complete 38.1s
  ✓85de74c6bda Pull complete 18.2s
  ✓897a2e813ecd Pull complete 1.2s
- prometheus Pulled 33.5s
  ✓f389c5827777 Pull complete 25.2s
  ✓1617c25568b2 Pull complete 2.5s
  ✓b778c5107ab5 Pull complete 1.7s
  ✓488812a7b118 Pull complete 1.7s
  ✓499862318b99 Pull complete 0.9s
  ✓7df673c7855d Pull complete 1.8s
  ✓6ac6e9ad315 Pull complete 25.6s
  ✓722165d8ba9 Pull complete 9.5s
  ✓9fa9226be834 Pull complete 2.8s
```

```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bhaskar Adhikary>cd C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer\load_tester

C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer\load_tester>python test.py
Request 0: {'result': {'current_load': 1, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 1: {'result': {'current_load': 1, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 2: {'result': {'current_load': 2, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 3: {'result': {'current_load': 2, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 4: {'result': {'current_load': 3, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 5: {'result': {'current_load': 3, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 6: {'result': {'current_load': 4, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 7: {'result': {'current_load': 4, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 8: {'result': {'current_load': 5, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 9: {'result': {'current_load': 5, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 10: {'result': {'current_load': 6, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 11: {'result': {'current_load': 6, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 12: {'result': {'current_load': 7, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 13: {'result': {'current_load': 7, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 14: {'result': {'current_load': 8, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 15: {'result': {'current_load': 8, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 16: {'result': {'current_load': 9, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 17: {'result': {'current_load': 9, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}
Request 18: {'result': {'current_load': 10, 'status': 'charging'}, 'routed_to': 'http://substation1:7000'}
Request 19: {'result': {'current_load': 10, 'status': 'charging'}, 'routed_to': 'http://substation2:7000'}

C:\Users\Bhaskar Adhikary\Desktop\smart-grid-load-balancer\load_tester>
```

