Vector Clocks and Causal Ordering

Title: Causally Consistent Key-Value Store using Vector Clocks

Name: Samir Kumar Jyotishi
Assignment : 1 (Vector Clocks and Causal Ordering)
Roll No : G24AI2047
Date : 25-June-2025

System Architecture:

Nodes: 3 nodes (A, B, C) hosted as separate Docker containers

Framework: Python Flask for each node

Orchestration: Docker Compose

Clock Mechanism: Vector Clocks

Communication: REST over HTTP via /write, /receive, and /read routes

Implementation

Each node:

- Stores a key-value map (store)
- Maintains a vector clock tracking logical time from each node
- Buffers received writes if causal dependencies aren't met
- Applies buffered writes when ready

Vector Clock Semantics

- **Local Write**: Increments its own clock

- **Send**: Includes current vector clock
- **Receive**: Merges clocks using max() per dimension
  *(does **not** increment own clock on receive)*

Causality Condition

For a node to apply a received message from sender S with VC V, the condition is:

V[S] == local[S] + 1

V[other] <= local[other] for all other nodes

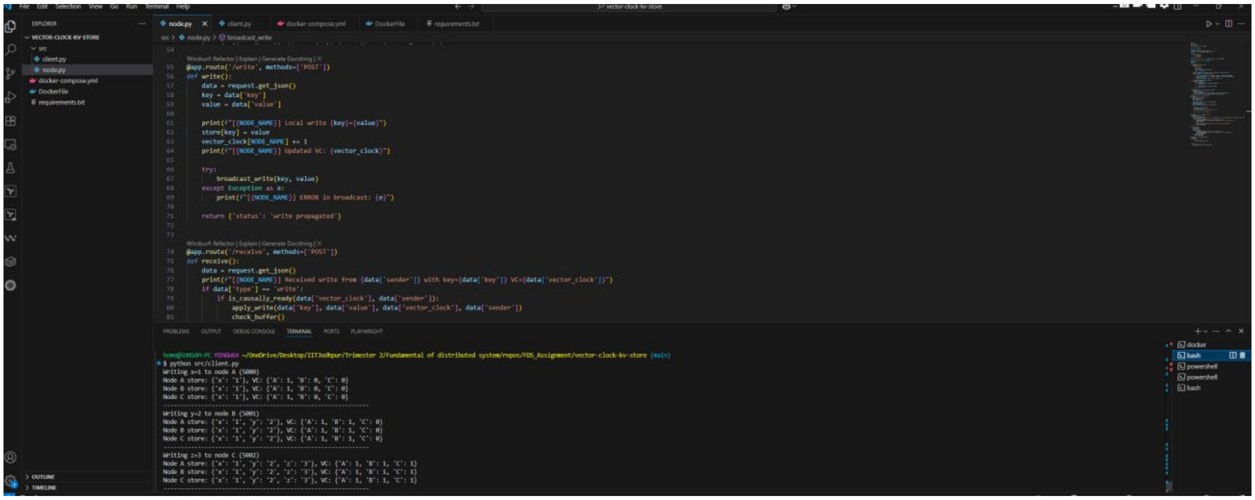
Steps to execute:

- In shell run docker-compose up
- Check for containers are running in Docker Desktop / Logs
- In another shell run “python src/client.py” or “python3 src/client.py”

"We first write x=1 to Node A. It updates its vector clock and broadcasts it.
Then we write y=2 to Node B.
Then z=3 to Node C.

Screenshots:

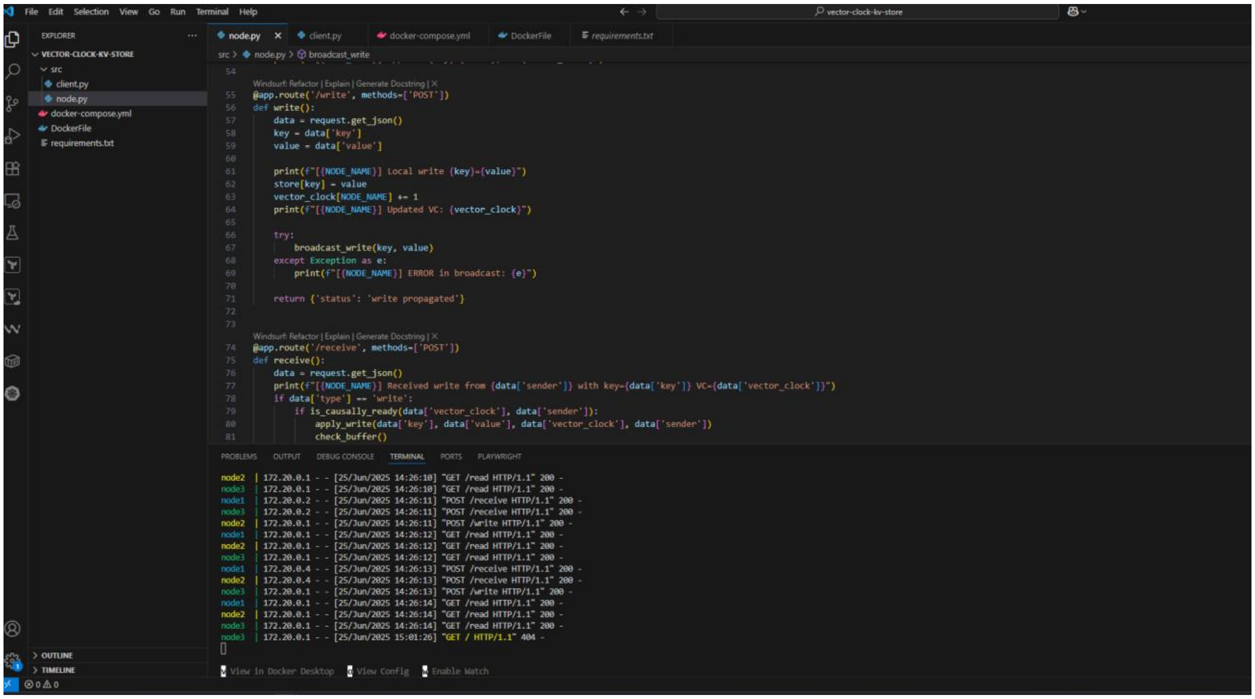
After running python src/client.py



```
@app.route('/write', methods=['POST'])
def write():
    data = request.get_json()
    key = data['key']
    value = data['value']

    print(f"[{NODE_NAME}] Local write {key}={value}")
    store[key] = value
    vector_clock[NODE_NAME] += 1
    print(f"[{NODE_NAME}] Updated VC: {vector_clock}")

    try:
        broadcast_write(key, value)
    except Exception as e:
        print(f"[{NODE_NAME}] ERROR in broadcast: {e}")

    return {'status': 'write propagated'}


@app.route('/receive', methods=['POST'])
def receive():
    data = request.get_json()
    print(f"[{NODE_NAME}] Received write from {data['sender']} with key={data['key']} VC={data['vector_clock']}")
    if data['type'] == 'write':
        if is_causally_ready(data['vector_clock'], data['sender']):
            apply_write(data['key'], data['value'], data['vector_clock'], data['sender'])
            check_buffer()
```

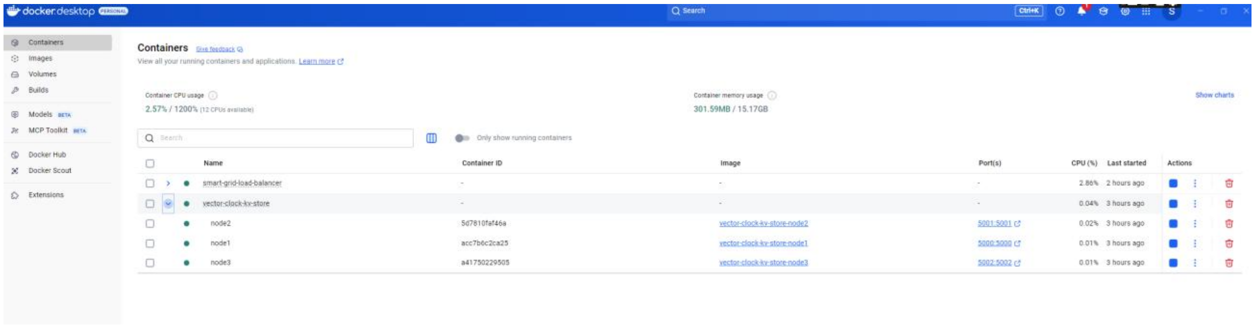
```
home@SUNSAM-PC MINGW64 ~/OneDrive/Desktop/IITJodhpur/Trimester 2/Fundamental of distributed system/repos/FDS_Assignment/vector-clock-kv-store (main)
$ python src/client.py
Writing x=1 to node A (5000)
Node A store: {'x': '1'}, VC: {'A': 1, 'B': 0, 'C': 0}
Node B store: {'x': '1'}, VC: {'A': 1, 'B': 0, 'C': 0}
Node C store: {'x': '1'}, VC: {'A': 1, 'B': 0, 'C': 0}
------------------------------------------------------------
Writing y=2 to node B (5001)
Node A store: {'x': '1', 'y': '2'}, VC: {'A': 1, 'B': 1, 'C': 0}
Node B store: {'x': '1', 'y': '2'}, VC: {'A': 1, 'B': 1, 'C': 0}
Node C store: {'x': '1', 'y': '2'}, VC: {'A': 1, 'B': 1, 'C': 0}
------------------------------------------------------------
Writing z=3 to node C (5002)
Node A store: {'x': '1', 'y': '2', 'z': '3'}, VC: {'A': 1, 'B': 1, 'C': 1}
Node B store: {'x': '1', 'y': '2', 'z': '3'}, VC: {'A': 1, 'B': 1, 'C': 1}
Node C store: {'x': '1', 'y': '2', 'z': '3'}, VC: {'A': 1, 'B': 1, 'C': 1}
------------------------------------------------------------
```

Docker compose up



```
@app.route('/write', methods=['POST'])
def write():
    data = request.get_json()
    key = data['key']
    value = data['value']

    print(f"[{NODE_NAME}] Local write {key}={value}")
    store[key] = value
    vector_clock[NODE_NAME] += 1
    print(f"[{NODE_NAME}] Updated VC: {vector_clock}")

    try:
        broadcast_write(key, value)
    except Exception as e:
        print(f"[{NODE_NAME}] ERROR in broadcast: {e}")

    return {'status': 'write propagated'}


@app.route('/receive', methods=['POST'])
def receive():
    data = request.get_json()
    print(f"[{NODE_NAME}] Received write from {data['sender']} with key={data['key']} VC={data['vector_clock']}")
    if data['type'] == 'write':
        if is_causally_ready(data['vector_clock'], data['sender']):
            apply_write(data['key'], data['value'], data['vector_clock'], data['sender'])
            check_buffer()
```

```
node2  | 172.20.0.1 - - [25/Jun/2025 14:26:10] "GET /read HTTP/1.1" 200 -
node3  | 172.20.0.1 - - [25/Jun/2025 14:26:10] "GET /read HTTP/1.1" 200 -
node1  | 172.20.0.2 - - [25/Jun/2025 14:26:11] "POST /receive HTTP/1.1" 200 -
node3  | 172.20.0.2 - - [25/Jun/2025 14:26:11] "POST /receive HTTP/1.1" 200 -
node2  | 172.20.0.1 - - [25/Jun/2025 14:26:11] "POST /write HTTP/1.1" 200 -
node1  | 172.20.0.1 - - [25/Jun/2025 14:26:12] "GET /read HTTP/1.1" 200 -
node2  | 172.20.0.1 - - [25/Jun/2025 14:26:12] "GET /read HTTP/1.1" 200 -
node3  | 172.20.0.1 - - [25/Jun/2025 14:26:12] "GET /read HTTP/1.1" 200 -
node1  | 172.20.0.4 - - [25/Jun/2025 14:26:13] "POST /receive HTTP/1.1" 200 -
node2  | 172.20.0.4 - - [25/Jun/2025 14:26:13] "POST /receive HTTP/1.1" 200 -
node3  | 172.20.0.1 - - [25/Jun/2025 14:26:13] "POST /write HTTP/1.1" 200 -
node1  | 172.20.0.1 - - [25/Jun/2025 14:26:14] "GET /read HTTP/1.1" 200 -
node2  | 172.20.0.1 - - [25/Jun/2025 14:26:14] "GET /read HTTP/1.1" 200 -
node3  | 172.20.0.1 - - [25/Jun/2025 14:26:14] "GET /read HTTP/1.1" 200 -
node3  | 172.20.0.1 - - [25/Jun/2025 15:01:26] "GET / HTTP/1.1" 404 -

v View in Docker Desktop   o View Config   w Enable Watch
```



Containers

View all your running containers and applications.

Container CPU usage
2.57% / 1200% (12 CPUs available)

Container memory usage
301.59MB / 15.17GB

		Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□	> ●	smart-grid-load-balancer	-	-	-	2.86%	2 hours ago	
□	●	vector-clock-kv-store	-	-	-	0.04%	3 hours ago	
□	●	node2	5d7810faf46a	vector-clock-kv-store-node2	5001:5001	0.02%	3 hours ago	
□	●	node1	acc7b6c2ca25	vector-clock-kv-store-node1	5000:5000	0.01%	3 hours ago	
□	●	node3	a41750229505	vector-clock-kv-store-node3	5002:5002	0.01%	3 hours ago	

Github Link:

https://github.com/samirkj/Vector_Clocks_And_Causal_Ordering_G24AI2047.git

Video:

Attached in the GitHub