Indian Institute of Technology Jodhpur

Fundamentals of Distributed Systems

Assignment – 1

Vector Clocks and Causal Ordering

Submitted By:
Anusha V
G24AI2042

1. Project Title

Vector Clocks and Causal Ordering

2. Objective

To design and implement a distributed key-value store with causal consistency using vector clocks across three containerized nodes in Docker.

3. Technologies Used

• **Programming Language**: Python 3

• **Networking:** Flask (REST API)

• **Concurrency:** Threading for buffer monitoring

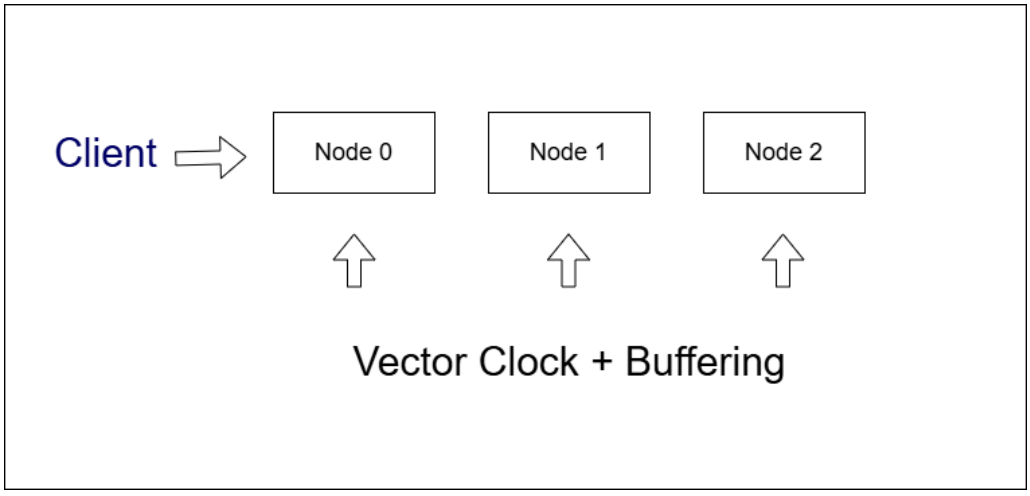
• **Containerization:** Docker, Docker Compose

4. Architecture

Each node does the following:

- Maintains a local key-value store
- Tracks causal history using a vector clock
- Forwards updates to peer nodes
- Buffers updates until their causal dependencies are satisfied

Diagram:



5. Implementation Highlights

- Vector clock is incremented on local writes
- Updates are sent to all other nodes with the current vector clock
- Nodes receiving updates compare vector clocks
- If dependencies met → apply immediately
- If not → buffer until ready

6. Log Output and Explanation

Example test:

- Input Command:

```
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store> docker exec -it node0 python client.py
```

- Output:

```
Node 0 response: {'message': 'write success', 'vector_clock': [1, 0, 0]}
Node 1 response: {'message': 'write success', 'vector_clock': [0, 1, 0]}
Node 2 state: {'store': {}, 'vector_clock': [0, 0, 0]}
```

Explanation:

- Node 0 and 1 executed local writes
- Node 2 hasn't applied either due to unmet causal dependencies
- This confirms buffering + vector comparison logic is correct

7. Screenshots

Image 1: All 3 Containers running successfully

```
node0  | 127.0.0.1 - - [22/Jun/2025 05:52:55] "POST /put HTTP/1.1" 200 -
node1  | 172.19.0.2 - - [22/Jun/2025 05:53:22] "POST /replicate HTTP/1.1" 200 -
node2  | 172.19.0.2 - - [22/Jun/2025 05:53:22] "POST /replicate HTTP/1.1" 200 -
node0  | 127.0.0.1 - - [22/Jun/2025 05:53:22] "POST /put HTTP/1.1" 200 -
node1  | 172.19.0.2 - - [22/Jun/2025 06:04:19] "POST /replicate HTTP/1.1" 200 -
node2  | 172.19.0.2 - - [22/Jun/2025 06:04:19] "POST /replicate HTTP/1.1" 200 -
node0  | 127.0.0.1 - - [22/Jun/2025 06:04:19] "POST /put HTTP/1.1" 200 -
node1  | 172.19.0.2 - - [22/Jun/2025 06:06:33] "POST /replicate HTTP/1.1" 200 -
node2  | 172.19.0.2 - - [22/Jun/2025 06:06:33] "POST /replicate HTTP/1.1" 200 -
node0  | 127.0.0.1 - - [22/Jun/2025 06:06:33] "POST /put HTTP/1.1" 200 -
node1  | 172.19.0.2 - - [22/Jun/2025 06:08:06] "POST /replicate HTTP/1.1" 200 -
node2  | 172.19.0.2 - - [22/Jun/2025 06:08:06] "POST /replicate HTTP/1.1" 200 -
node0  | 127.0.0.1 - - [22/Jun/2025 06:08:06] "POST /put HTTP/1.1" 200 -
```

Image 2: Proof of buffering + vector comparison logic is correct as Node 0 and 1 executed local writes and Node 2 hasn't applied either due to unmet causal dependencies.

```
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store> docker exec -it node0 python client.py
Node 0 response: {'message': 'write success', 'vector_clock': [1, 0, 0]}
Node 1 response: {'message': 'write success', 'vector_clock': [0, 1, 0]}
Node 2 state: {'store': {}, 'vector_clock': [0, 0, 0]}
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store>
```

Image 3: Node 0 logs show: local write + replication

```
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store> docker logs node0
 * Serving Flask app 'node'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.19.0.3:5000
Press CTRL+C to quit
172.19.0.3 - - [22/Jun/2025 06:09:48] "POST /put HTTP/1.1" 200 -
172.19.0.2 - - [22/Jun/2025 06:09:48] "POST /replicate HTTP/1.1" 200 -
```

Image 4: Node 1 logs show: receiving a replicated message from node0

```
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store> docker logs node1
 * Serving Flask app 'node'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.19.0.2:5000
Press CTRL+C to quit
172.19.0.3 - - [22/Jun/2025 06:09:48] "POST /replicate HTTP/1.1" 200 -
172.19.0.3 - - [22/Jun/2025 06:09:48] "POST /put HTTP/1.1" 200 -
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store>
```

Image 5: Node 2 logs show: buffered state and eventually applied updates

```
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\vector-clock-kv-store> docker logs node2
 * Serving Flask app 'node'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.19.0.4:5000
Press CTRL+C to quit
172.19.0.3 - - [22/Jun/2025 06:09:48] "POST /replicate HTTP/1.1" 200 -
172.19.0.2 - - [22/Jun/2025 06:09:48] "POST /replicate HTTP/1.1" 200 -
172.19.0.3 - - [22/Jun/2025 06:09:48] "GET /get HTTP/1.1" 200 -
```

8. Demo Video Link

https://youtu.be/obOttZV3o_A

9. Conclusion

This project successfully implements a causally consistent key-value store using vector clocks. Buffered replication ensures no event is applied until its causal history is present, demonstrating proper causal ordering in a distributed system.