

# Causally Consistent Distributed Key-Value Store using Vector Clocks

## Assignment 1 – Fundamentals of Distributed Systems

Name: Mohamed Bin Badhusha E B

Roll No: g24ai2026

### 1. Introduction

Distributed systems face a major challenge in maintaining consistency across multiple nodes, especially when events happen concurrently. Traditional Lamport clocks can only establish a partial order, but they fall short in capturing causal relationships between events. To address this, this project implements Vector Clocks for tracking causality and ensures that a key-value store remains causally consistent across three nodes.

### 2. Objective

The goal of this assignment is to:

- Understand causal ordering in distributed systems.
- Implement a multi-node key-value store using Vector Clocks.
- Maintain causal consistency by buffering out-of-order writes.
- Test and verify the correctness using a controlled scenario.

### 3. Technologies Used

Tool	Purpose
Python (3.10)	Node logic, vector clock implementation
Flask	API server for each node
Docker	Containerizing each node
Docker Compose	Orchestrating all 3 nodes together
Requests lib	Used in client to send HTTP POST/GET

### 4. System Architecture

- Three independent nodes (X, Y, Z), each runs a Flask server and maintains:
  - A local key-value store
  - A Vector Clock
  - A message buffer for undeliverable updates
- Nodes communicate over HTTP and apply causal rules before accepting writes.
- Writes with missing dependencies are buffered and processed later.

## 5. Implementation Overview

### ► Vector Clock Logic:

Each node has a clock like {X: 0, Y: 0, Z: 0}.

On every write:

- It increments its own entry.
- Sends the entire clock with the update.

On receiving:

- It checks causal readiness using `is_causally_ready()`.
- If not ready, it buffers the message.

### ► Node Logic (service\_node.py):

- Implements PUT endpoint to accept writes.
- Checks causal conditions.
- Starts a background thread to retry buffered writes.

### ► Dockerization:

Each node is containerized using a custom Dockerfile, and launched via `compose-run.yml`.

## 6. Screenshots

### 📸 Screenshot 1 – System Build and Startup using Docker Compose

```
C:\Users\badhu\OneDrive\Desktop\FDS Assignment 1\causal-kv-store>docker-compose -f compose-run.yml up --build
time="2025-06-25T09:01:12+05:30" level=warning msg="C:\\\\Users\\\\badhu\\\\OneDrive\\\\Desktop\\\\FDS Assignment 1\\\\causal-kv-store\\\\compose-run.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 13.5s (20/24)
ker:desktop-linux
=> [node_x internal] load build definition from SetupFile
    0.0s
=>> transferring dockerfile: 324B
    0.0s
=> [node_y internal] load build definition from SetupFile
    0.1s
=>> transferring dockerfile: 324B
    0.0s
=> [node_z internal] load build definition from SetupFile
    0.0s
=>> transferring dockerfile: 324B
    0.0s
=> [node_x internal] load metadata for docker.io/library/python:3.10
    4.1s
=> [node_z internal] load .dockerignore
    0.0s
=>> transferring context: 2B
    0.0s
=> [node_y internal] load .dockerignore
    0.0s
=>> transferring context: 2B
    0.0s
=> [node_x internal] load .dockerignore
    0.1s
=>> transferring context: 2B
    0.0s
=> [node_z internal] FROM docker.io/library/python:3.10@sha256:33f72df2ad8c9f777bf0adb35b9d89c5d62935cee2af1f9c3224fb6f7daldc6b
    0.0s
=> [node_z internal] load build context
    0.0s
=>> transferring context: 4.58kB
    0.0s
=> [node_y internal] load build context
    0.0s
=>> transferring context: 4.58kB
    0.0s
=> [node_x internal] load build context
    0.0s
=>> transferring context: 4.58kB
    0.0s
=> CACHED [node_z 2/4] WORKDIR /app
    0.0s
=> [node_y 3/4] COPY src/ .
    0.0s
=> [node_y 4/4] RUN pip install flask requests
    8.7s
=> [node_z] exporting to image
    0.3s
=>> exporting layers
    0.2s
=>> writing image sha256:6fcbb6ba3926969491c2bfc0d6673604cd6b54429deba3ad6862fd3da42263207
    0.0s
=>> naming to docker.io/library/causal-kv-store-node_z
    0.0s
=> [node_x] exporting to image
    0.3s
=>> exporting layers
    0.2s
=>> writing image sha256:f8dba9ff0b771bc840e04f2f257b19a48fc292949bd9c8b611e1305aa0abe27e
    0.0s
=>> naming to docker.io/library/causal-kv-store-node_x
    0.0s
```

```

=> => exporting layers
    0.2s
=> => writing image sha256:78fdb61dc7e9b1c1fffb52dbf4fa1b4e0fd44e5846e133b6454e004fe01de0f2
    0.0s
=> => naming to docker.io/library/causal-kv-store-node_y
    0.0s
=> [node_x] resolving provenance for metadata file
    0.0s
=> [node_x] resolving provenance for metadata file
    0.0s
=> [node_y] resolving provenance for metadata file
    0.0s
=> [node_z] resolving provenance for metadata file
    0.0s
[+] Running 7/7
  ✓ node_x           Built
  ✓ node_y           Built
  ✓ node_z           Built
  ✓ Network causal-kv-store_default   Created
  ✓ Container causal-kv-store-node_y-1 Created
  ✓ Container causal-kv-store-node_z-1 Created
  ✓ Container causal-kv-store-node_x-1 Created
Attaching to node_x-1, node_y-1, node_z-1
node_y-1 | * Serving Flask app 'service_node'
node_y-1 | * Debug mode: off
node_y-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node_y-1 | * Running on all addresses (0.0.0.0)
node_y-1 | * Running on http://127.0.0.1:6000
node_y-1 | * Running on http://172.21.0.2:6000
node_z-1 | * Serving Flask app 'service_node'
node_y-1 | Press CTRL+C to quit
node_x-1 | * Serving Flask app 'service_node'
node_z-1 | * Debug mode: off
node_x-1 | * Debug mode: off
node_z-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node_x-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node_z-1 | * Running on all addresses (0.0.0.0)
node_x-1 | * Running on all addresses (0.0.0.0)
node_z-1 | * Running on http://127.0.0.1:6000
node_x-1 | * Running on http://127.0.0.1:6000
node_z-1 | * Running on http://172.21.0.4:6000
node_x-1 | * Running on http://172.21.0.3:6000
node_z-1 | Press CTRL+C to quit
node_x-1 | Press CTRL+C to quit
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:33:08] "POST /store HTTP/1.1" 200 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:33:10] "POST /store HTTP/1.1" 202 -
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:33:12] "GET /retrieve?key=x HTTP/1.1" 200 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:33:12] "GET /retrieve?key=x HTTP/1.1" 200 -
node_z-1 | 172.21.0.1 - - [25/Jun/2025 03:33:12] "GET /retrieve?key=x HTTP/1.1" 200 -
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:36:03] "POST /store HTTP/1.1" 202 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:36:04] "POST /store HTTP/1.1" 200 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:36:05] "POST /store HTTP/1.1" 202 -
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:36:07] "GET /retrieve?key=x HTTP/1.1" 200 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:36:07] "GET /retrieve?key=x HTTP/1.1" 200 -
node_z-1 | 172.21.0.1 - - [25/Jun/2025 03:36:07] "GET /retrieve?key=x HTTP/1.1" 200 -
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:38:11] "POST /store HTTP/1.1" 202 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:38:12] "POST /store HTTP/1.1" 202 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:38:13] "POST /store HTTP/1.1" 200 -
node_x-1 | 172.21.0.1 - - [25/Jun/2025 03:38:15] "GET /retrieve?key=x HTTP/1.1" 200 -
node_y-1 | 172.21.0.1 - - [25/Jun/2025 03:38:15] "GET /retrieve?key=x HTTP/1.1" 200 -
node_z-1 | 172.21.0.1 - - [25/Jun/2025 03:38:15] "GET /retrieve?key=x HTTP/1.1" 200 -

```

☰ View in Docker Desktop   
 ☰ View Config   
 ☰ Enable Watch

 Screenshot 2 – Client Test Output & Causal Consistency

```

PS C:\Users\badhu\OneDrive\Desktop\FDS Assignment 1\causal-kv-store\src> python test_client.py
● Writing x=100 from Node X

✉ Sending to Node X at http://localhost:6000/store
✓ Node X says: {'incoming_clock': {'X': 1, 'Y': 0, 'Z': 0}, 'local_clock': {'X': 2, 'Y': 0, 'Z': 0}, 'reason': 'causality conflict', 'status': 'delayed'}

✉ Syncing Node X's write to Node Y

✉ Sending to Node Y at http://localhost:6001/store
✓ Node Y says: {'incoming_clock': {'X': 1, 'Y': 0, 'Z': 0}, 'local_clock': {'X': 1, 'Y': 1, 'Z': 0}, 'reason': 'causality conflict', 'status': 'delayed'}

● Writing x=200 from Node Y (causally dependent)

✉ Sending to Node Y at http://localhost:6001/store
✓ Node Y says: {'clock': {'X': 1, 'Y': 3, 'Z': 0}, 'key': 'x', 'node': 'Y', 'status': 'saved', 'value': '200'}

● Final key state from all nodes:
↳ Node X → {'key': 'x', 'local_clock': {'X': 2, 'Y': 0, 'Z': 0}, 'stored_clock': {'X': 1, 'Y': 0, 'Z': 0}, 'value': '100'}
↳ Node Y → {'key': 'x', 'local_clock': {'X': 1, 'Y': 3, 'Z': 0}, 'stored_clock': {'X': 1, 'Y': 2, 'Z': 0}, 'value': '200'}
↳ Node Z → {'clock': {'X': 0, 'Y': 0, 'Z': 0}, 'message': "Key 'x' not found", 'value': None}

PS C:\Users\badhu\OneDrive\Desktop\FDS Assignment 1\causal-kv-store\src>

```

## 7. Observations

- When Node Y attempts a write dependent on Node X's earlier write, it gets buffered until Node X's update is applied.
- All final values are consistent once causal order is respected.
- Demonstrated that messages arriving out of order are safely handled and applied in correct order.

## 8. Challenges Faced

- Docker Desktop didn't start initially — solved by rebooting and running as administrator.
- Buffer logic and `is_causally_ready()` conditions needed several test runs to get right.
- Port forwarding in `docker-compose.yml` required precise configuration for testing.

## 9. Learnings

- Gained deep understanding of causal ordering vs total ordering.
- Learned to write and test vector clocks from scratch.
- Understood how to containerize distributed applications.
- Practiced real-world debugging in Docker environments.

## 10. Demo Video Link (Max 3 Minutes)

Video Link:

<https://drive.google.com/file/d/1TpY0GuT9O97JrcOD1dShu-9ipUCPlDfs/view?usp=sharing>

## File Structure Followed

```

```
causal-kv-store/
|
|--- src/
|   |--- service_node.py
|   |--- causal_clock.py
|   |--- test_client.py
|
|--- SetupFile (Dockerfile)
|--- compose-run.yml
|--- project_report.pdf
```
```