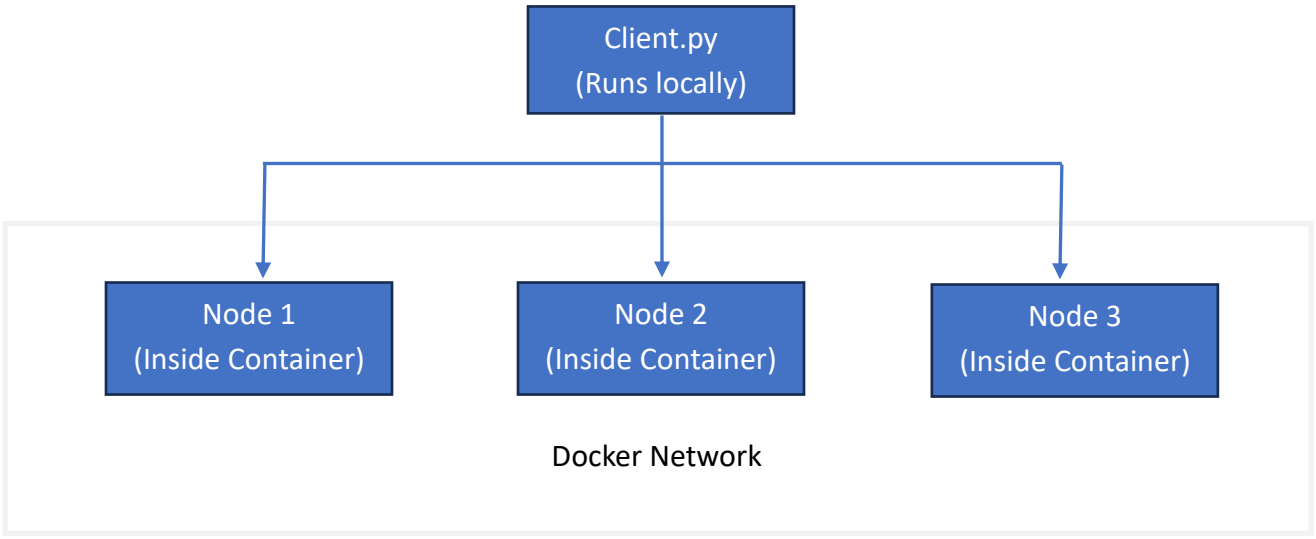
Project Report

Part 1

Vector Clocks and Causal Ordering

In this assignment the objective is to implement a distributed key-value store using Vector Clocks to enforce causal consistency across 3 or more nodes.

The assignment follows the below overall architecture:



Below are the descriptions of each file that are used to implement Vector Clocks to capture the causal relationships between events in a distributed system:

File	Logic(process)
node.py	This file contains key-value store with vector clock logic/distributed node logic
client.py	This file sends reads/writes to different nodes
Dockerfile	Used for containerizing node.py
docker-compose.yml	This file sets up 3-node system with networking

Below are the steps followed to complete the assignment:

- After preparing all the files I ran the command “docker-compose up --build” which created and started all the services defined in the docker-compose.yml file.

```
[+] Running 7/7
 ✓ node1                                      Built
           0.0s
 ✓ node2                                      Built
           0.0s
 ✓ node3                                      Built
           0.0s
 ✓ Network vector-clock-kv-store_default      Created
           0.1s
 ✓ Container vector-clock-kv-store-node2-1    Created
           0.1s
 ✓ Container vector-clock-kv-store-node1-1    Created
           0.1s
 ✓ Container vector-clock-kv-store-node3-1    Created
           0.1s
```

- Then I ran the python file “client.py” which performed a series of write and read operations on the distributed nodes and checked if vector clocks and causal relationships are properly maintained.

```
PS C:\Users\Kush\Downloads\vector-clock-kv-store> cd src
PS C:\Users\Kush\Downloads\vector-clock-kv-store\src> python client.py

Step 1: Node1 writes x=5
Node1 put x=5: {'message': 'Value stored with causal consistency', 'vc': [1, 0, 0]}

Step 2: Node2 writes x=10
Node2 put x=10: {'message': 'Value stored with causal consistency', 'vc': [1, 1, 0]}

Step 3: Node3 writes y=15 (independent write)
Node3 put y=15: {'message': 'Value stored with causal consistency', 'vc': [1, 1, 1]}

Step 4: Read 'x' and 'y' from all nodes
node1 stores:
  x: {'key': 'x', 'value': '10', 'vc': [1, 1, 0]}
  y: {'key': 'y', 'value': '15', 'vc': [1, 1, 1]}
node2 stores:
  x: {'key': 'x', 'value': '10', 'vc': [1, 1, 0]}
  y: {'key': 'y', 'value': '15', 'vc': [1, 1, 1]}
node3 stores:
  x: {'key': 'x', 'value': '10', 'vc': [1, 1, 0]}
  y: {'key': 'y', 'value': '15', 'vc': [1, 1, 1]}
```

```
node2-1  | 172.18.0.4 - - [22/Jun/2025 11:37:31] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.4 - - [22/Jun/2025 11:37:31] "POST /replicate HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:31] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.3 - - [22/Jun/2025 11:37:33] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.3 - - [22/Jun/2025 11:37:33] "POST /replicate HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:33] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.2 - - [22/Jun/2025 11:37:35] "POST /replicate HTTP/1.1" 200 -
node2-1  | 172.18.0.2 - - [22/Jun/2025 11:37:35] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:35] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=x HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=y HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=x HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=y HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=x HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:37] "GET /get?key=y HTTP/1.1" 200 -
```

- Again, on executing the same client.py file I got below results:

```
PS C:\Users\Kush\Downloads\vector-clock-kv-store\src> python client.py

Step 1: Node1 writes x=5
Node1 put x=5: {'message': 'Value stored with causal consistency', 'vc': [2, 1, 1]}

Step 2: Node2 writes x=10
Node2 put x=10: {'message': 'Value stored with causal consistency', 'vc': [2, 2, 1]}

Step 3: Node3 writes y=15 (independent write)
Node3 put y=15: {'message': 'Value stored with causal consistency', 'vc': [2, 2, 2]}

Step 4: Read 'x' and 'y' from all nodes
node1 stores:
  x: {'key': 'x', 'value': '10', 'vc': [2, 2, 1]}
  y: {'key': 'y', 'value': '15', 'vc': [2, 2, 2]}
node2 stores:
  x: {'key': 'x', 'value': '10', 'vc': [2, 2, 1]}
  y: {'key': 'y', 'value': '15', 'vc': [2, 2, 2]}
node3 stores:
  x: {'key': 'x', 'value': '10', 'vc': [2, 2, 1]}
  y: {'key': 'y', 'value': '15', 'vc': [2, 2, 2]}
PS C:\Users\Kush\Downloads\vector-clock-kv-store\src>
```

```
node2-1  | 172.18.0.4 - - [22/Jun/2025 11:37:42] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.4 - - [22/Jun/2025 11:37:42] "POST /replicate HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:42] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.3 - - [22/Jun/2025 11:37:44] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.3 - - [22/Jun/2025 11:37:44] "POST /replicate HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:44] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.2 - - [22/Jun/2025 11:37:46] "POST /replicate HTTP/1.1" 200 -
node2-1  | 172.18.0.2 - - [22/Jun/2025 11:37:46] "POST /replicate HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:46] "POST /put HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=x HTTP/1.1" 200 -
node1-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=y HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=x HTTP/1.1" 200 -
node2-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=y HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=x HTTP/1.1" 200 -
node3-1  | 172.18.0.1 - - [22/Jun/2025 11:37:48] "GET /get?key=y HTTP/1.1" 200 -
```

Video Link:

https://drive.google.com/file/d/1ARjxgrpfWjqTOvF1XS84Qyn4UYN3gGao/view?usp=sharing

Conclusion:

Executing the files shows that causal consistency is maintained by using vector clocks. Updates like x=10 is applied only after x=5 while y=15 is applied without any delay as it's an independent write, shows that correct order is maintained across all nodes & repeated run shows that clocks are getting added cumulatively preserving the prior state.

By,

Keshab Garg

G24AI2021