Sanyam Agrawal SE21UCSE192 CSE3

DS - Assignment 2: Implementing Lamport Logical Clocks in a Distributed System

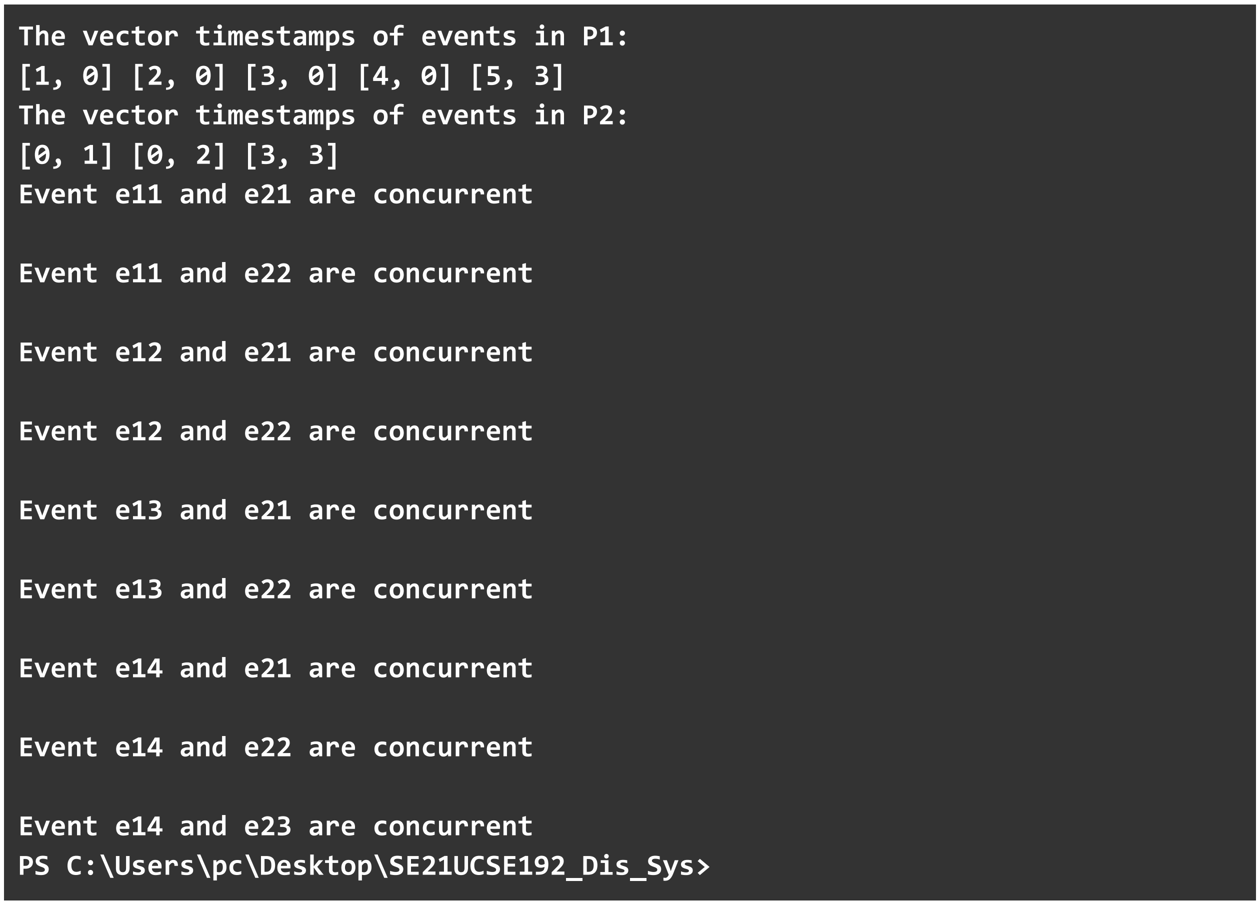
Code: lamport_vc_conc.py->

```
def display_vector(e1, e2, p1, p2):
    print()
    print("The vector timestamps of events in P1:")
    for i in range(e1):
        print(p1[i], end=" ")

    print()
    print("The vector timestamps of events in P2:")
    for i in range(e2):
        print(p2[i], end=" ")

def is_concurrent(v1, v2):
    return not all(v1[i] <= v2[i] for i in range(len(v1))) and not all(v2[i] <=
v1[i] for i in range(len(v2)))

def vector_clock_with_concurrency(e1, e2, m):
    p1 = [[0, 0] for _ in range(e1)]
    p2 = [[0, 0] for _ in range(e2)]

    for i in range(e1):
        p1[i][0] = i + 1
        p1[i][1] = 0

    for i in range(e2):
        p2[i][0] = 0
        p2[i][1] = i + 1

    for i in range(e1):
        for j in range(e2):
            if m[i][j] == 1:  # Message sent
                p2[j] = [max(p2[j][0], p1[i][0] + 1), max(p2[j][1], p1[i][1] +
1)]
                for k in range(j + 1, e2):
                    p2[k] = [p2[k - 1][0] + 1, p2[k - 1][1] + 1]
```

```python
            elif m[i][j] == -1:  # Message received
                p1[i] = [max(p1[i][0], p2[j][0] + 1), max(p1[i][1], p2[j][1] +
1)]
                for k in range(i + 1, e1):
                    p1[k] = [p1[k - 1][0] + 1, p1[k - 1][1] + 1]

    display_vector(e1, e2, p1, p2)

    # Detect and report concurrent events
    for i in range(e1):
        for j in range(e2):
            if is_concurrent(p1[i], p2[j]):
                print(f"\nEvent e1{i+1} and e2{j+1} are concurrent")

# Driver Code
if __name__ == "__main__":
    e1 = 5
    e2 = 3
    m = [[0]*3 for _ in range(5)]

    m[0][0] = 0
    m[0][1] = 0
    m[0][2] = 0
    m[1][0] = 0
    m[1][1] = 0
    m[1][2] = 1
    m[2][0] = 0
    m[2][1] = 0
    m[2][2] = 0
    m[3][0] = 0
    m[3][1] = 0
    m[3][2] = 0
    m[4][0] = 0
    m[4][1] = -1
    m[4][2] = 0

    vector_clock_with_concurrency(e1, e2, m)
```

Output Screenshot ->

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys> python -u
"c:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab2\lamport_vc_conc.py"
```



```
The vector timestamps of events in P1:
[1, 0] [2, 0] [3, 0] [4, 0] [5, 3]
The vector timestamps of events in P2:
[0, 1] [0, 2] [3, 3]
Event e11 and e21 are concurrent

Event e11 and e22 are concurrent

Event e12 and e21 are concurrent

Event e12 and e22 are concurrent

Event e13 and e21 are concurrent

Event e13 and e22 are concurrent

Event e14 and e21 are concurrent

Event e14 and e22 are concurrent

Event e14 and e23 are concurrent
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys>
```

<u>Brief Report on Event Ordering using Vector Clocks:</u>

Vector Timestamps:

- **Process P1:** The vector timestamps for events in P1 are `[1, 0], [2, 0], [3, 0], [4, 0], [5, 3]`.
- **Process P2:** The vector timestamps for events in P2 are `[0, 1], [0, 2], [3, 3]`.

Each entry in the vector timestamp represents the knowledge of that process about its own events and the events in the other process. For example, the timestamp `[5, 3]` for the last event in P1 indicates that P1 is aware of five events in P1 and three events in P2.

Event Ordering:

- The code identifies concurrent events by comparing vector timestamps of events in P1 and P2. Two events are concurrent if their vector timestamps are neither entirely less than nor entirely greater than each other.

For example:

- **Event e1 (in P1) with timestamp [1, 0]** and **Event e21 (in P2) with timestamp [0, 1]** are concurrent because neither `[1, 0] <= [0, 1]` nor `[0, 1] <= [1, 0]`.

Concurrency Analysis:

The code correctly identifies all pairs of concurrent events:

- **Concurrent Events:**
  - Events e11 and e21
  - Events e11 and e22
  - Events e12 and e21
  - Events e12 and e22
  - Events e13 and e21
  - Events e13 and e22
  - Events e14 and e21
  - Events e14 and e22
  - Events e14 and e23

These results indicate that the majority of events between P1 and P2 are concurrent, meaning they are independent of each other and do not have a causal relationship.