Distributed Systems Lab Assignment 4
Read Me
By: Sneha Potluri
19XJ1A0572

Git link: https://github.com/sneha-pots/ds_4

logical.txt and code should be in same file

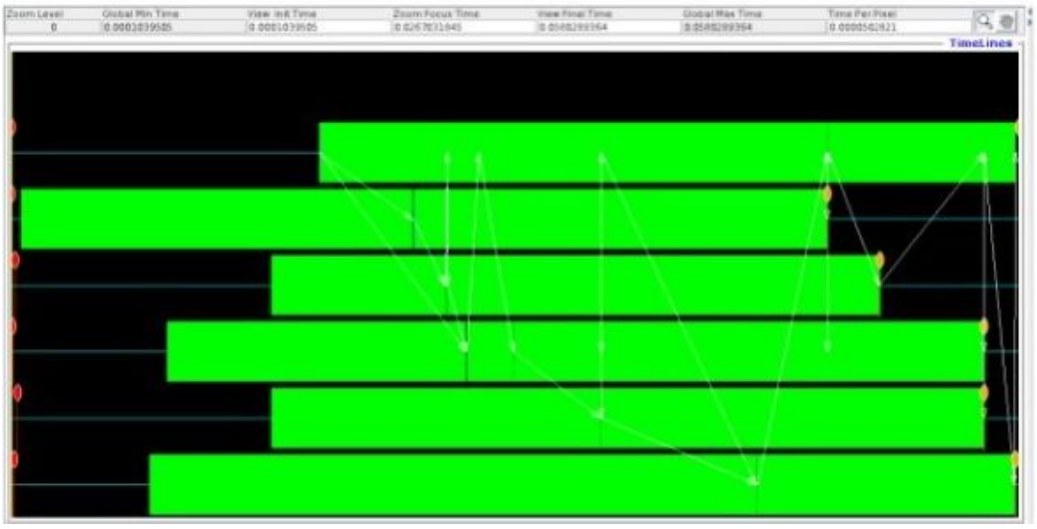
compile: mpicc logical.c -o logical
Execute: mpirun -np 4 ./logical ./logical.txt

Core Idea of Program Logic:

This implementation will run three separate processes, which can communicate with each other through messages.
Each of these processes has its own internal counter, which will be updated with each event.
The script will then print a line for each event together with the updated internal counter and the time on the machine running the processes.

One of the shortcomings of Lamport Timestamps is rooted in the fact that they only partially order events (as opposed to total order).
Partial order indicates that not every pair of events need be comparable.
If two events can't be compared, we call these events concurrent.
The problem with Lamport Timestamps is that they can't tell if events are concurrent or not.
This problem is solved by Vector Clocks.

Jumpshot:



Lamport Clock

- Process: Pi
  Event: Eij, where i is the process in number and j: jth event in the ith process.
- tm: vector time span for message m.
- Ci vector clock associated with process Pi, the jth element is Ci[j] and contains Pi's latest value for the current time in process Pj.
- d: drift time, generally d is 1.

[IR1]: If a -> b ['a' happened before 'b' within the same process] then, Ci(b) =Ci(a) + d
[IR2]: Cj = max(Cj, tm + d) [If there's more number of processes, then tm = value of Ci(a), Cj = max value between Cj and tm + d]

Vector Clock

- Initially all clocks are zero.
- Each time a process experiences an internal event, it increments its own logical clock in the vector by one. For instance, upon an event at process i, it updates .
- Each time a process sends a message, it increments its own logical clock in the vector by one (as in the bullet above, but not twice for the same event) and then the message piggybacks a copy of its own vector.

Each time a process receives a message, it increments its own logical clock in the vector by one and updates each element in its vector by taking the maximum of the value in its own vector clock and the value in the vector in the received message (for every element). For example, if process Pj receives a message m from Pi, it updates by setting