# Implementing Vector Clocks

Vishal Vijay Devadiga (CS21BTECH11061)

## Index


- Index
- Question
- Answer
  - Vector Clocks
  - Implementation
  - Performance Comparison
    - Data Collection
    - Graph Topology 1: Sparse Graph Topology
    - Graph Topology 2: Dense Graph Topology
  - Conclusion


## Question

Implement **Vector Clocks** in a distributed system using ZeroMQ for communication between processes in C++.

Compare the performance between Vector Clocks and Singhal-Kshemkalyani's optimization of Vector Clocks.

## Answer

### Vector Clocks

**Vector Clocks** are a mechanism to capture the causality relationship between events in a distributed system. They are used to determine the order of events in a distributed system| and are used in distributed systems for various purposes.

## Implementation

I have implemented the Vector Clocks in C++ using **ZeroMQ Push and Pull sockets** for communication between processes(threads in this case).

Each process has a thread that listens for incoming messages and another thread that either executes a internal event(sleep) or sends a message to another process. So each process sends a message to another process at random intervals and also executes an internal event(sleep) at random intervals.

The Vector Clocks are implemented as a class in C++ and the class has methods to increment the clock, merge the clock and compare the clock. The class also has a method to log all the events and the clock values.

Most of the code is self-explanatory and is documented in the code itself.

## Performance Comparison

Note that here, by size of message, I mean the total space for sending messages in all processes and by number of events sent, I mean the total number of events sent by all processes.

### Data Collection

I have collected the data for the performance comparison by running the program for different number of processes. The program was run for **10, 11, 12, 13, 14, 15 processes** and for **2 graph topologies** and the **space used for messages** and **number of events sent** were recorded.

The space used for messages was recorded by size of the message sent and the number of events sent was recorded by the number of events sent by each process.

The data was collected for both Vector Clocks and Singhal-Kshemkalyani's optimization of Vector Clocks.

**Graph Topology 1: Sparse Graph Topology**

In the first graph topology, each process has a direct connection to n - 6 other processes(could have done n/2, but since I am checking for n from 10 to 15, it is almost the same) where n is the total number of processes.

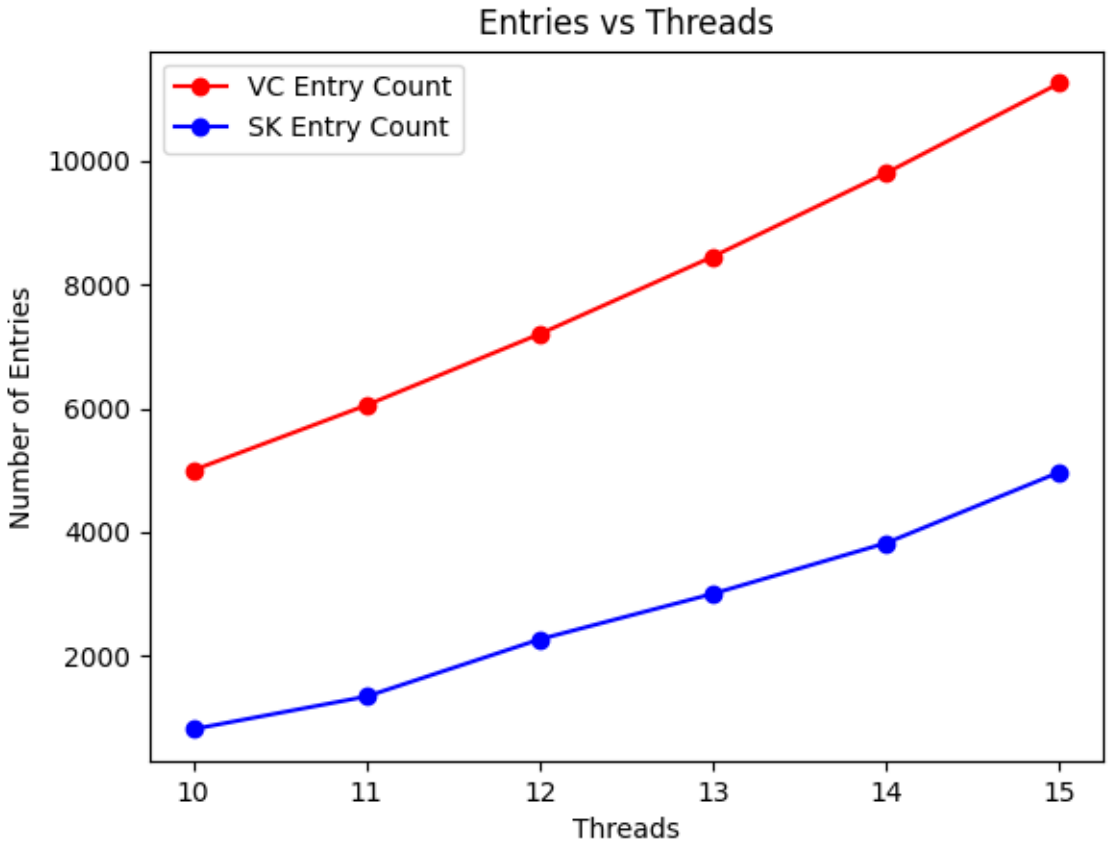
Below is the data collected(averaged out):

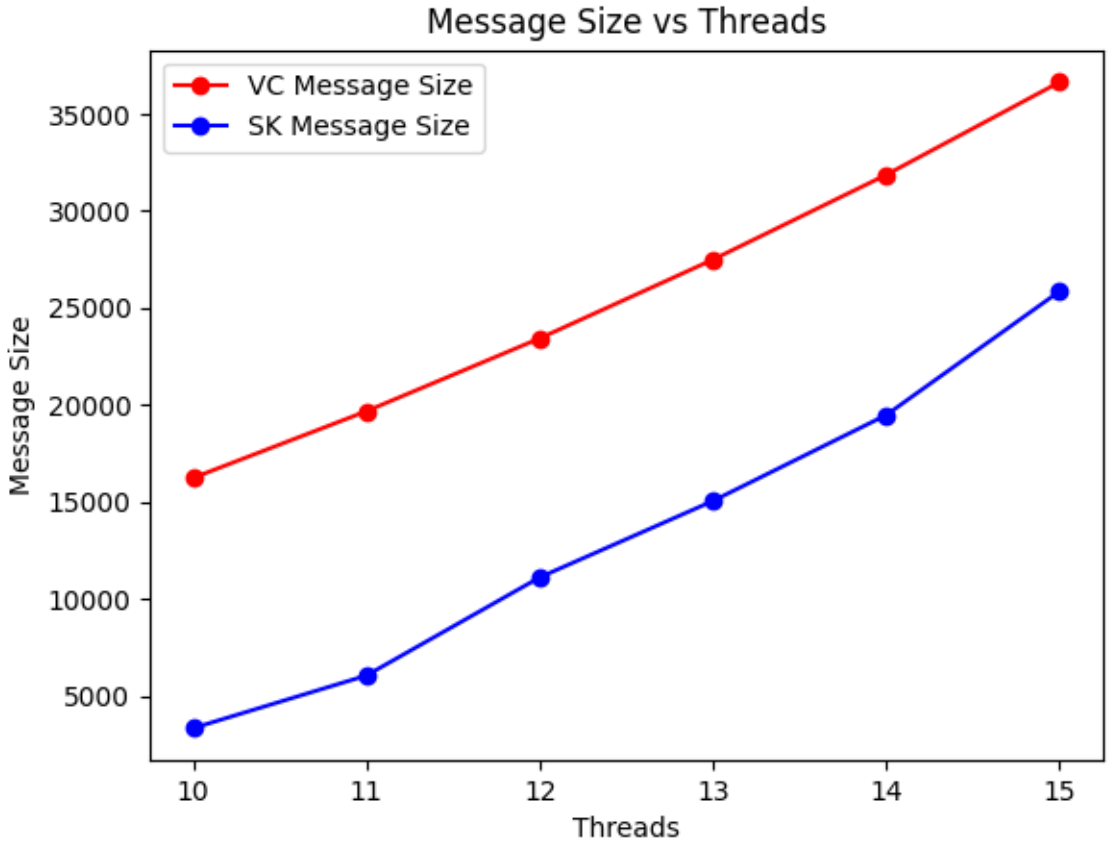
| Threads | Event Count (VC) | Space Used (VC) | Event Count (SK) | Space Used (SK) |
|---|---|---|---|---|
| 10 | 5000.0 | 16248.0 | 821.2 | 3350.8 |
| 11 | 6050.0 | 19685.4 | 1348.2 | 6064.2 |
| 12 | 7200.0 | 23440.8 | 2271.4 | 11121.6 |
| 13 | 8450.0 | 27501.4 | 3007.4 | 15063.4 |
| 14 | 9800.0 | 31872.8 | 3826.8 | 19492.2 |
| 15 | 11250.0 | 36649.0 | 4971.2 | 25870.2 |

The data shows that the space used for messages is much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks. The number of events sent is also much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks.

The graphs show that the number of events sent and the space used for messages is much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks.

Below are the graphs for the data collected:


Entries vs Threads
VC Entry Count
SK Entry Count
Number of Entries
10000
8000
6000
4000
2000
10
11
12
13
14
15
Threads


Message Size vs Threads
VC Message Size
SK Message Size
Message Size
35000
30000
25000
20000
15000
10000
5000
10
11
12
13
14
15
Threads

**Graph Topology 2: Dense Graph Topology**

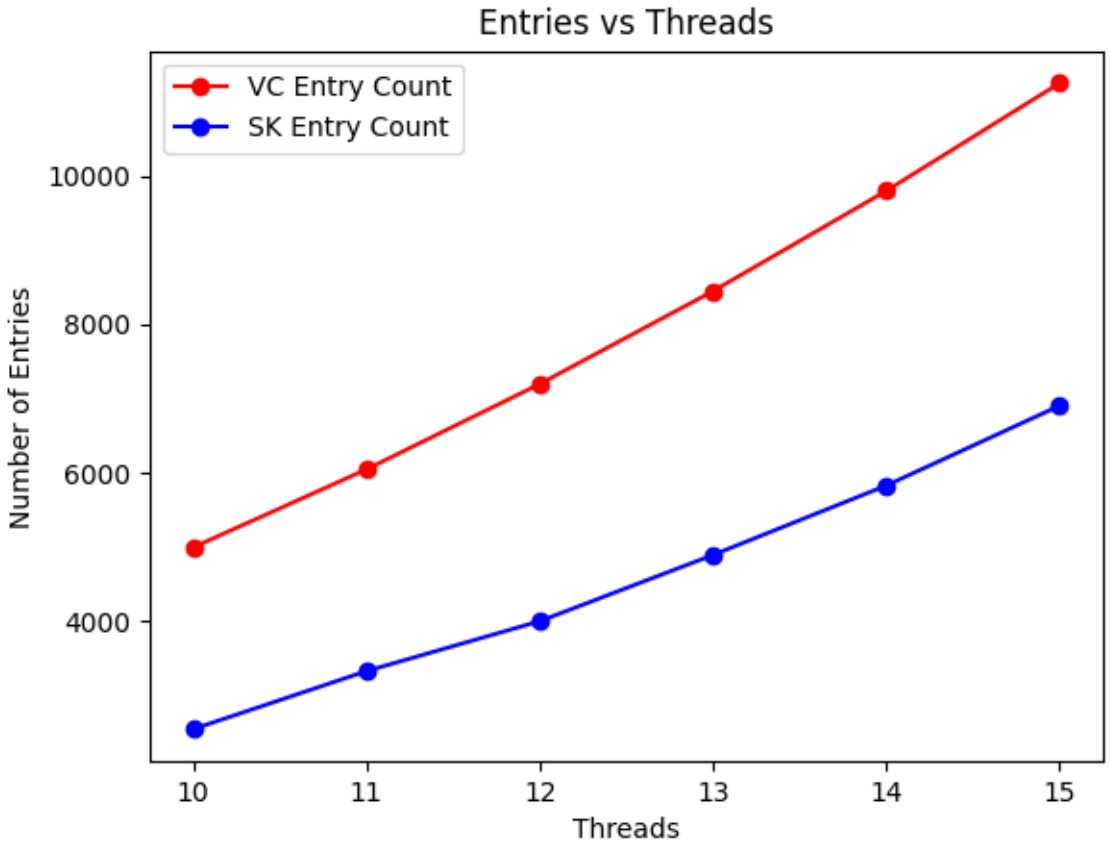
In the second graph topology, each process has a direct connection to all other processes.

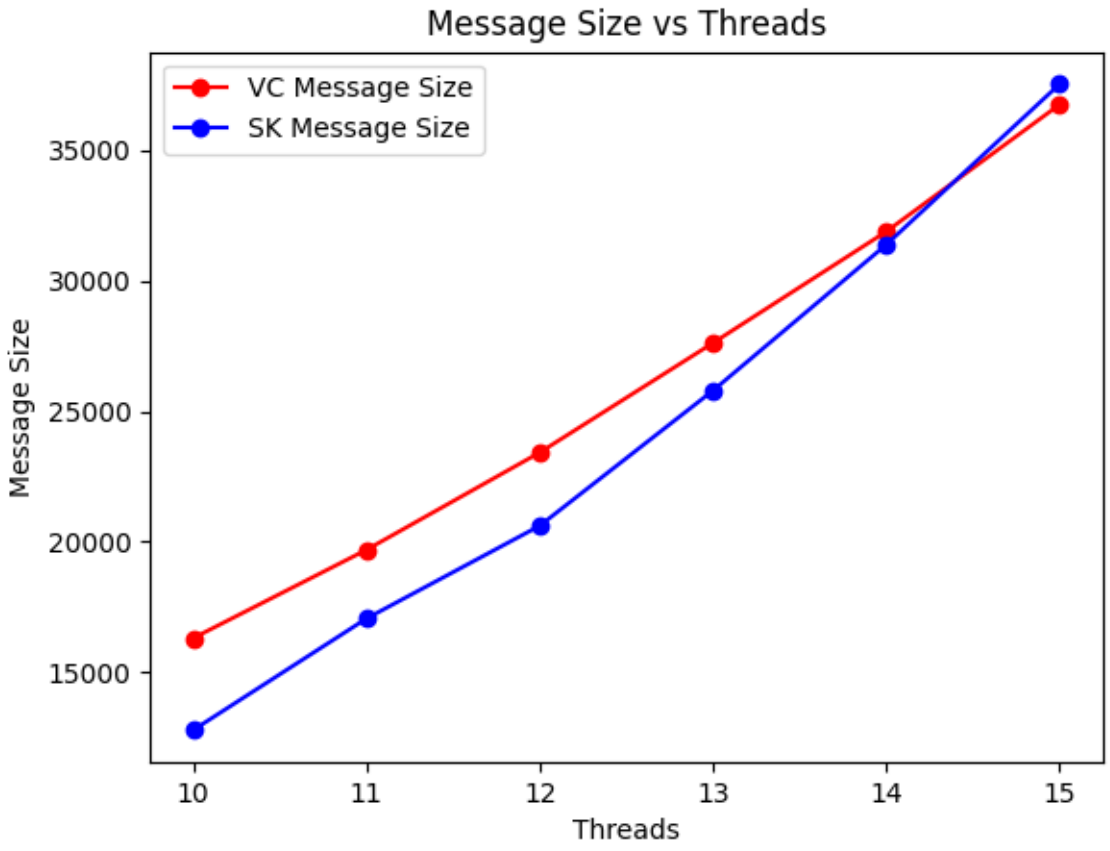
Below is the data collected(averaged out):

| Threads | Event Count (VC) | Space Used (VC) | Event Count (SK) | Space Used (SK) |
|---|---|---|---|---|
| 10 | 5000.0 | 16314.6 | 2552.8 | 12797.0 |
| 11 | 6050.0 | 19694.6 | 3334.4 | 17065.4 |
| 12 | 7200.0 | 23418.0 | 4006.2 | 20616.2 |
| 13 | 8450.0 | 27611.2 | 4898.0 | 25789.8 |
| 14 | 9800.0 | 31873.2 | 5832.8 | 31374.0 |
| 15 | 11250.0 | 36732.2 | 6907.4 | 37514.2 |

Both the graph and data shows that while number of events sent is much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks, the space used for messages is quite similar for both Vector Clocks and Singhal-Kshemkalyani's optimization of Vector Clocks.

Below are the graphs for the data collected:


Entries vs Threads
VC Entry Count
SK Entry Count
Number of Entries
10000
8000
6000
4000
10
11
12
13
14
15
Threads


Message Size vs Threads
VC Message Size
SK Message Size
Message Size
35000
30000
25000
20000
15000
10
11
12
13
14
15
Threads

## Conclusion

The data collected shows that:

- There is a **linear relation** between the number of events sent and the number of threads, and the space used for messages and the number of threads.
- For a sparse graph topology, the space used for messages and the number of events sent is much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks.
- For a dense graph topology, the space used for messages is quite similar for both Vector Clocks and Singhal-Kshemkalyani's optimization of Vector Clocks, but the number of events sent is much higher for Vector Clocks compared to Singhal-Kshemkalyani's optimization of Vector Clocks.

The reason why is that in Singhal-Kshemkalyani's optimization of Vector Clocks, **the message size is increased by sending both the index and the value of the clock, but the number of entries sent is reduced, thus cancelling out the increase in message size.**

This shows that Singhal-Kshemkalyani's optimization of Vector Clocks is better than Vector Clocks in terms of space used for messages and number of events sent in a sparse graph topology and is quite similar to Vector Clocks in terms of space used for messages in a dense graph topology but is much better than Vector Clocks in terms of number of events sent in a dense graph topology.