Distributed Computing: Autumn 2018
Programming Assignment 1: Implementing Vector Clocks

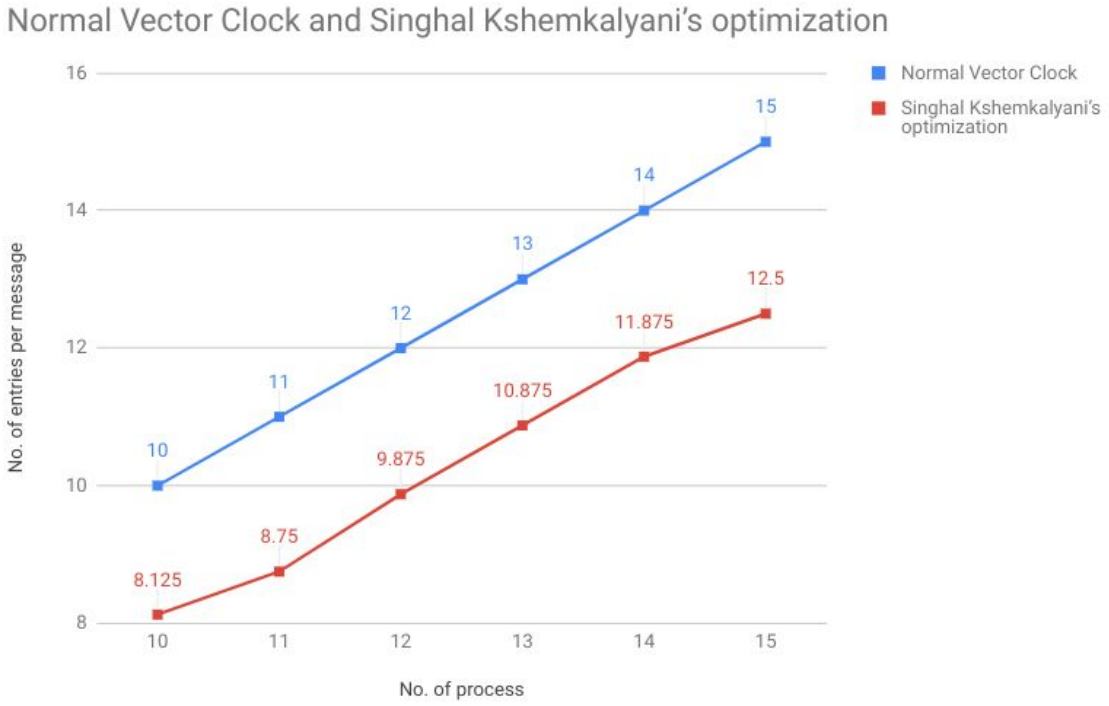
Ganesh Vernekar - CS15BTECH11018

Graphs (lambda=2, alpha=2, M=50)

Topology: Every process has every other process as its neighbour.

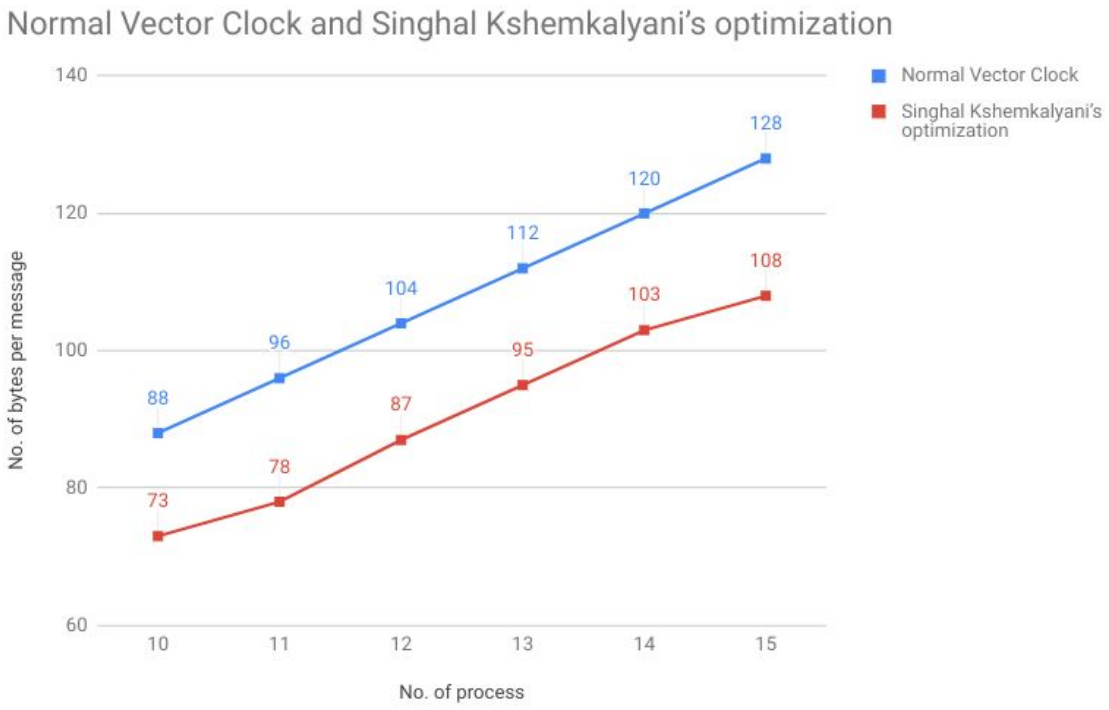
1. Number of entries per message.

We can see that number of entries per message was less in the optimized version though the volume of message was more. So we can infer that when the number of messages sent are less, then optimized version would work even better with lot fewer entries per message.



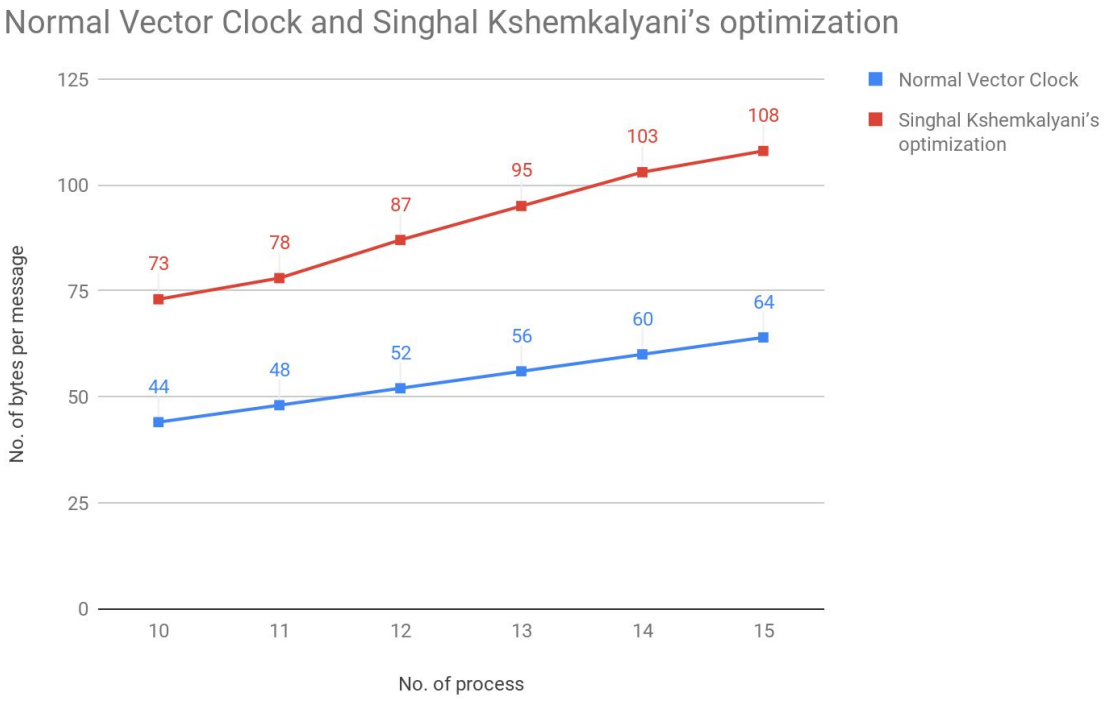
2. Number of bytes per message, when <u>same format of message</u> is used in both algorithms.
In this case, the format of message was: `[ sender_id, num_tuples, tuples… ]`
Where each tuple is `process_id, its_time`.

Inference: As both followed same format, we can see the reduction of number of bytes send in the optimized algorithm as we are not sending the entire vector. But in case of the normal method, we would ideally not like to use the same format, hence look at the 2nd graph to know when does the optimisation method work well.



3. Number of bytes per message, hen <u>different format is used for normal vector clock method</u>.
In this case for format for the optimized version stays the same as above. But for the normal vector clock algorithm, we just send the array of the times, where the index in the array tells the process id: `[ sender_id, clock_of_1, clock_of_2, ……, clock_of_N ]`.

Inference: Here you can see that, for <u>less number of processes and frequent messages</u>, the normal implementation would work better, as frequent messages doesn't reduce the number of vector clock entries in optimized version to a great extent.



When will optimized work best?

When there are a lot of update messages sent and received, every time we would be sending majority of the processes from the vector. Also if the number of processes are less, the overhead added by tuples in the optimized version can become a lot more than the unoptimized version.

Hence it will work best when number of update messages sent and received are less, and the number of processes is big.