Week 5 (Introduction to Concurrent Programming) Report

-By Tirth Nandha (25535018)

Graphs generated:

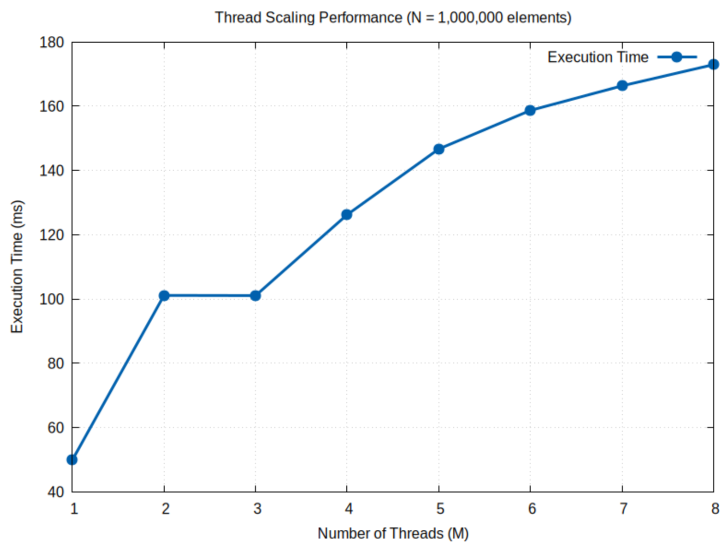

Figure 1: Thread Scaling Performance (fixed N = 1,000,000 elements, varying number of threads M).

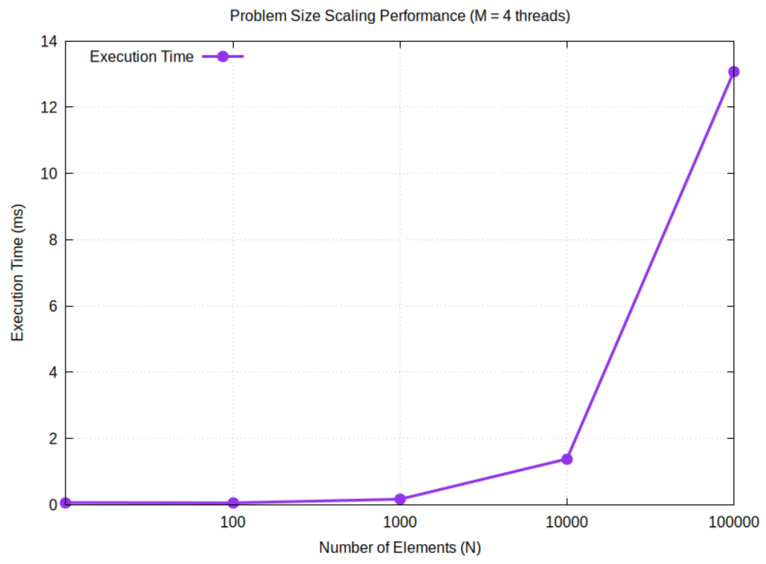

Figure 2: Problem Size Scaling Performance (fixed M = 4 threads, varying number of elements N).

Obervations:

Figure 1:

- As the number of threads increases, the time taken also increases which is opposite to the ideal case in which the time should decrease as the number of threads increases.
- For small values of M there are some ups and down in time but beyond a specific no of threads like 2 in our case the time increases steadily.

Figure 2:

- The execution time increase as no of elements N increases.
- For smaller values of N like upto 1000, the time is almost negligible.
- Beyond 1000 the time grows exponentially.

Explanation of results:

Figure 1:

- Our main concern is the maintain the same order as vector in the list so if multiple threads operate on vector at the same time and insert on its own when its comes then the order will of Linked List and vector will not be maintained. So to maintain the order we implemented mutex lock on the shared index. So each thread has to wait till the previous thread process its element. With more threads there is more fight for the lock, leading to increase in wait time. Due to this the synchronization time dominates the runtime.

Figure 2:

- Here the algorithm processes each element once, so the complexity is O(N). So, As the number of elements increases, the runtime increases proportionally.