CENG478 - HW1 - Tests and Report

Full Name : Orçun Başşimşek
Id Number : 2098804

a)

Theoretical result of this inner product = 7700000

- For 1 process: Calculated Result = 7700000.000132 (taken from 1_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7700000.000132}{7700000}\right| = 17.143 \times 10^{-12}$$

- For 2 processes: Calculated Result = 7700000.000079 (taken from 2_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7700000.000079}{7700000}\right| = 10.259 \times 10^{-12}$$

- For 4 processes: Calculated Result = 7699999.999992 (taken from 4_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7699999.999992}{7700000}\right| = 1.039 \times 10^{-12}$$

- For 8 processes: Calculated Result = 7700000.000020 (taken from 8_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7700000.000020}{7700000}\right| = 2.597 \times 10^{-12}$$

- For 16 processes: Calculated Result = 7699999.999980 (taken from 16_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7699999.999980}{7700000}\right| = 2.597 \times 10^{-12}$$

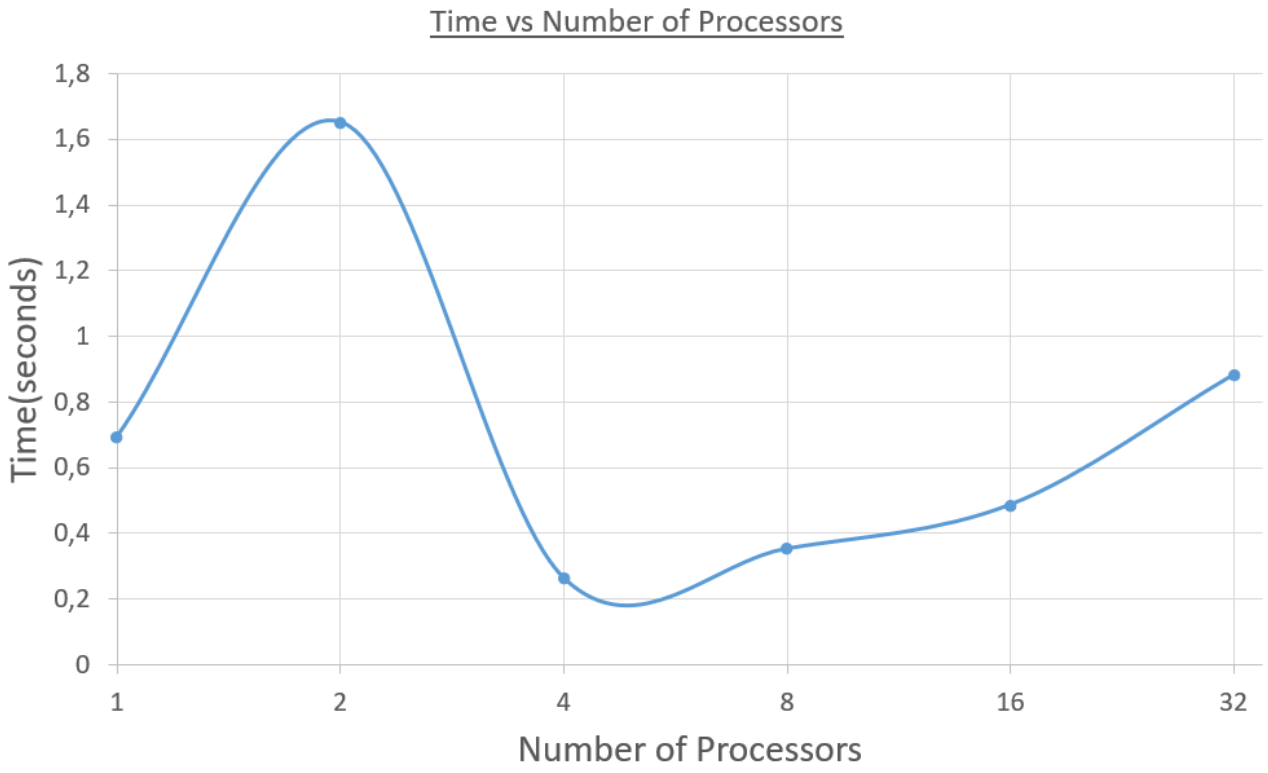
- For 32 processes: Calculated Result = 7700000.000001 (taken from 32_out.txt)

$$Relative\ Error = \left|\frac{7700000 - 7700000.000001}{7700000}\right| = 0.129 \times 10^{-12}$$

b)

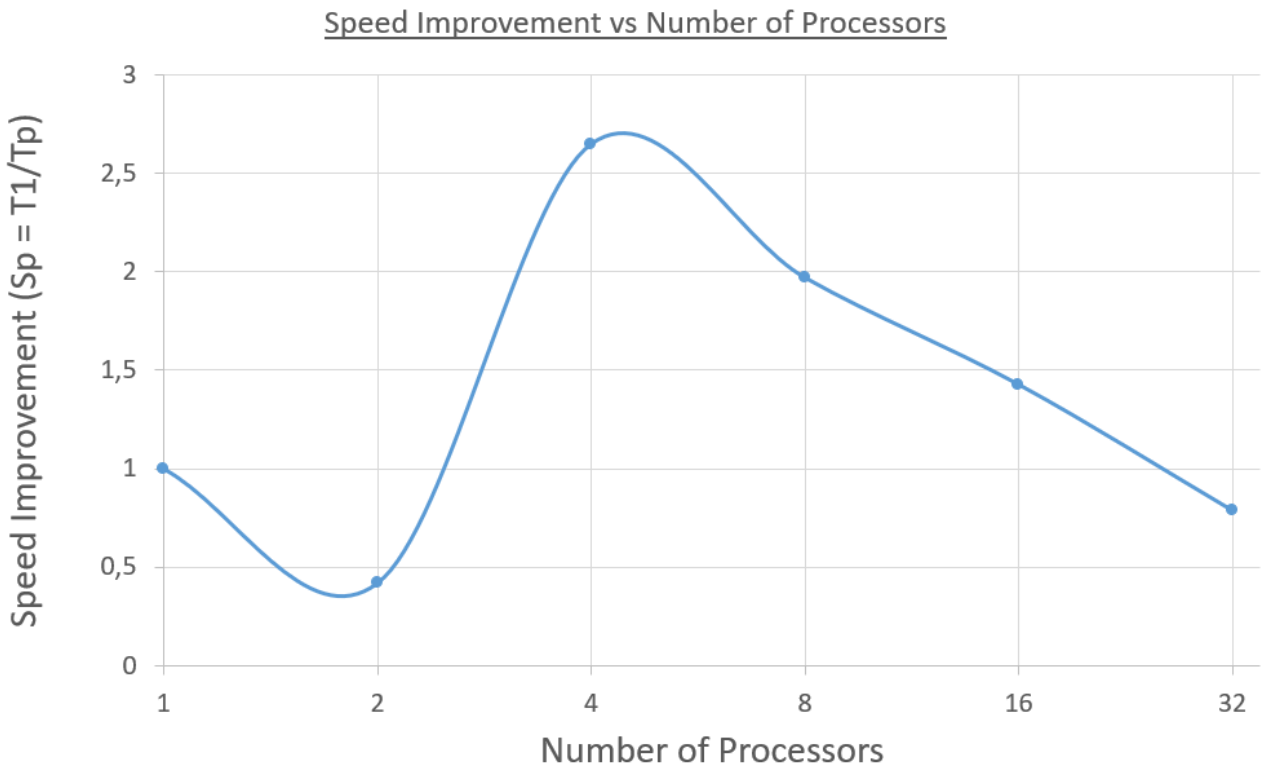
According to my implementation, for each execution that runs different number of processes in parallel, each process is responsible for different parts of the vectors to apply dot product to corresponding vector elements. The size of the vector part that processes are responsible for depends on the current number of processes working in parallel. When the size of the vector part increases, possibility of round-up errors due to floating point addition and multiplication tends to increase also, and this causes relative errors. On the other hand, if we increase number of processes and shrink the size of vector part of for each process, while adding all parts' results by MPI_Reduce(), again we can encounter round-up errors due to floating point arithmetic, and again we will see different results. Thus, with my implementation, the calculated result will be slightly different than actual(theoretical) result all the time.

c)



Consumed time data for the above graph is taken from my experimental results (from #_out.txt files, # = number of processors). Time axis shows the consumed time from my root process(process which has rank 0) for each different execution. As seen in the above graph, increasing the number of processors did not decrease the consumed time correspondingly for our case. It can be logical because parallel running time does not depend only the computation time. Communication and

idle times should also be thought while evaluating these results. The flactuations may also be depend on my algorithm, the network or operating system. For my implementation, best parallel running time is acquired when working with 4 processors.



I calculated speed improvement(speedup) for each execution as following:

$$S_p = \frac{T_1}{T_n}$$

S_p = Speed Improvement(Speedup)
T_1 = Consumed time with 1 processor
T_n = Consumed time with n processors

As seen in the above figure, execution with 4 parallel processor has the most speedup. If we think that the parallel efficiency is equal to:

$$Parallel\ Efficiency = \frac{S_p}{n} = \frac{Speedup}{Number\ of\ processors}$$

Execution with 4 processor again gives better parallel efficiency(which is close to 1) compared to other executions.