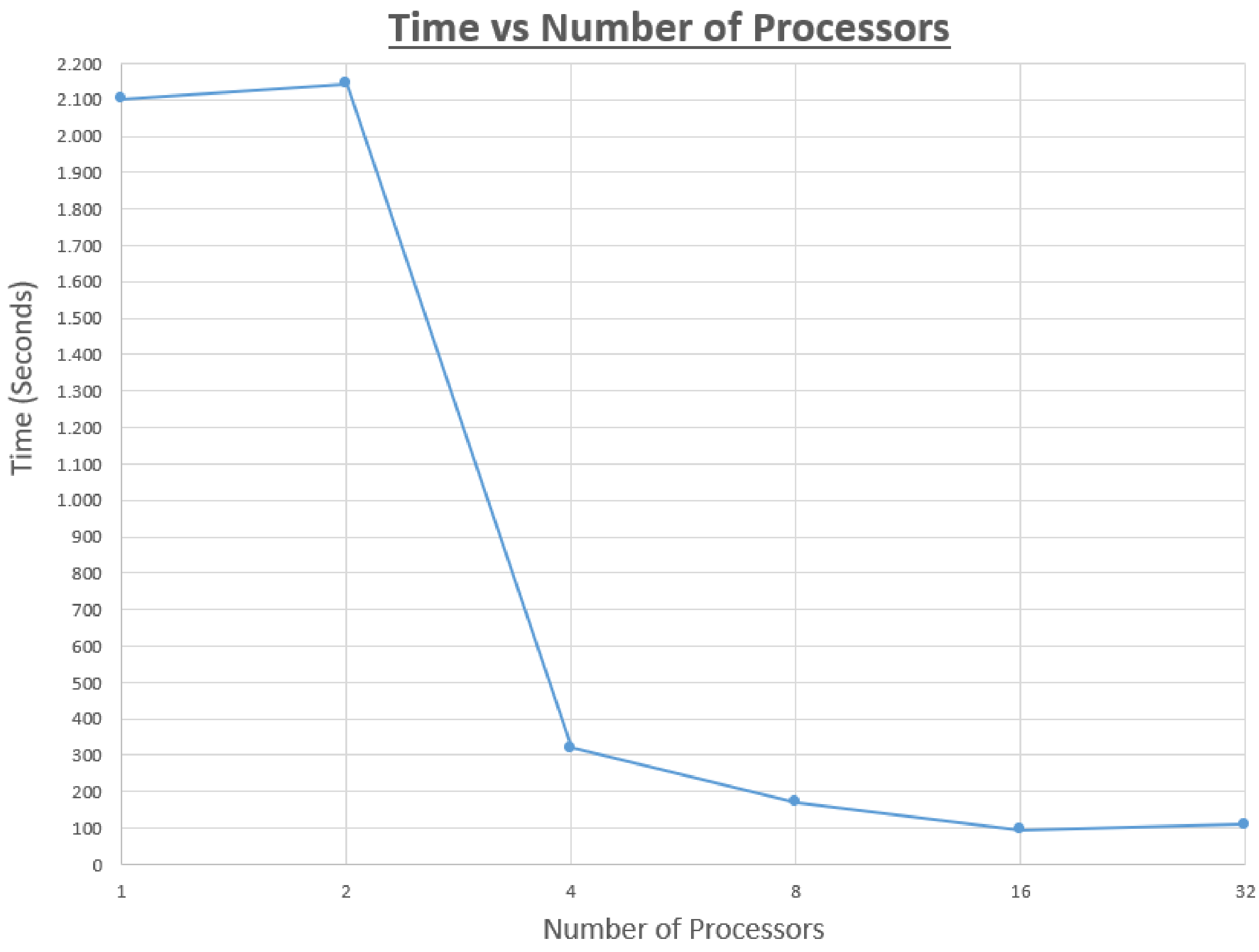
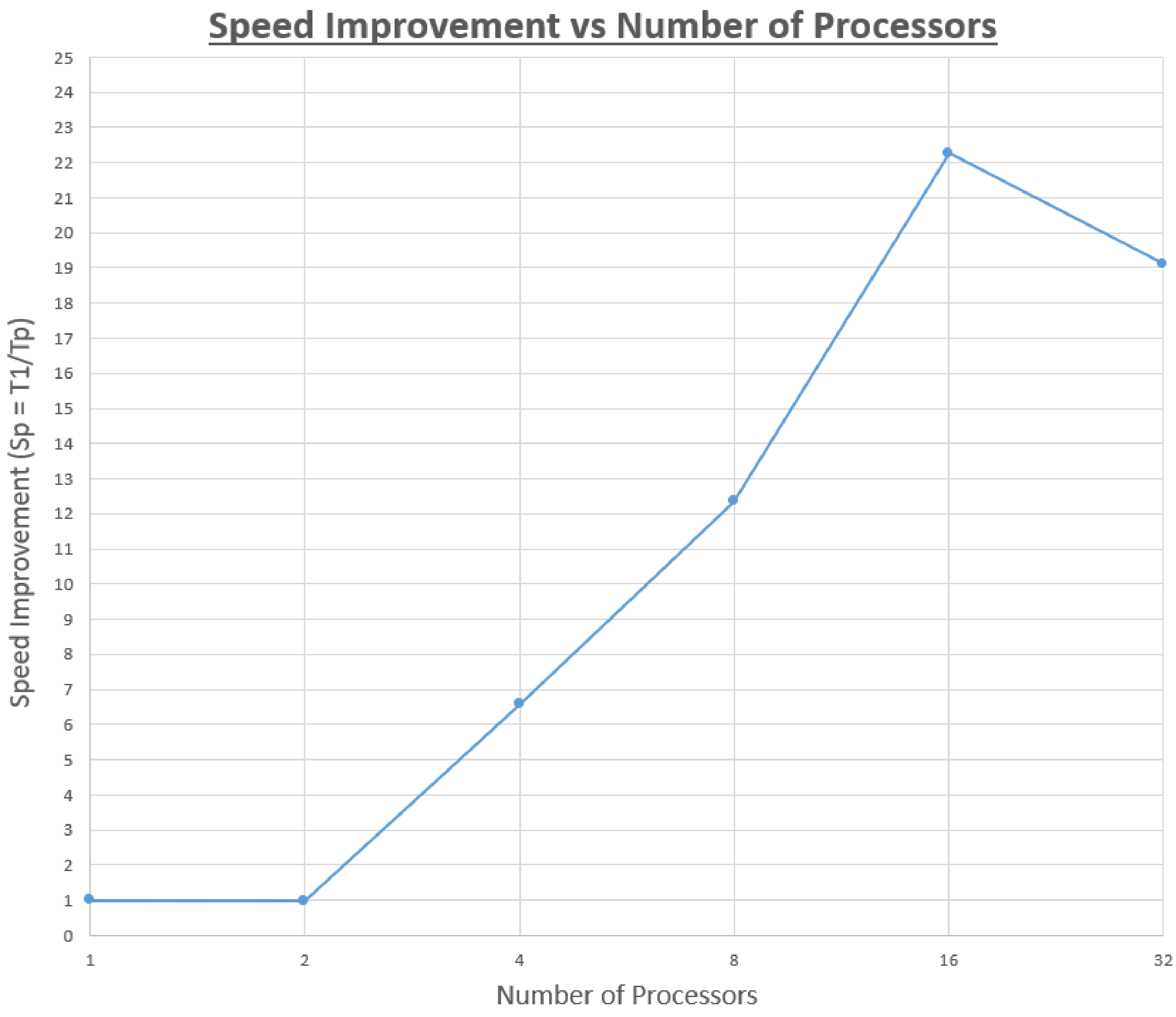
CENG478 - HW3 - Tests and Report

Full Name : Orçun Başşimşek
Id Number : 2098804



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 the process that finishes its task latest for each different execution(I used MPI_Reduce for taking maximum elapsed time by the latest processor). As seen in the above graph, increasing the number of processors does not always decrease the consumed time properly. For example, while working with 2 processor, much more time is consumed even if it is compared with 1 processor case. I think the cause of this increase is parallel overhead. However, after 2 processor case, when I increased the number of processors, consumed time decreased so much. I understand that computation time was smaller and smaller for these cases, and it affects parallel running time positively for us. Actually, this is expected scenario for my simple implementation. I just partitioned the rows of first matrix (matrix A), and assigned these smaller partitions to each processor. Then, each processor realized multiplication between its special partition of matrix A and whole matrix B. Hence, when the number of processors was increased, each processor did less job. Parallel communication was created only while summing the Frobenius Norm results of each resulting partition. If I interpret why 32 processors case is slower than 16 processors case, it is possible that this parallel communication time could play an important role for these two cases. As a result, for my implementation, best parallel running time is acquired when working with 16 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 graph, execution with 16 parallel processors 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 processors, and then execution with 8 processors gives better parallel efficiency compared to other executions. About speculation of performance improving, I think for my simple implementation, increasing the number of processors could be beneficial to improve performance due to my implementation related reasons that I mentioned before. If we increase the number of processors, each processor will handle less data, and also complexity of calculations will not tend to increase. Of course, this definitely causes additional parallel overhead. However, optimal point can be found to gain from parallelization much more.