German Razo
CSCI551

Report

Value of N used: 44083498
Absolute Relative True Error using N trapezoids for p = 1: 4.93447066404711111563e-15

Table of timings for all runs

	Runs		
	1	**2**	**3**
comm_sz (number of cores)			
2	14.7742	14.7787	14.7778
8	3.7365	3.7128	3.7253
14	2.1261	2.1856	2.1291
20	1.5069	1.5085	1.5311

Speedup for each configuration

	Runs		
	1	**2**	**3**
comm_sz (number of cores)			
2	1.8329	1.8323	1.8324
8	7.2473	7.2936	7.2691
14	12.7366	12.3897	12.7185
20	17.9703	17.9508	17.6855

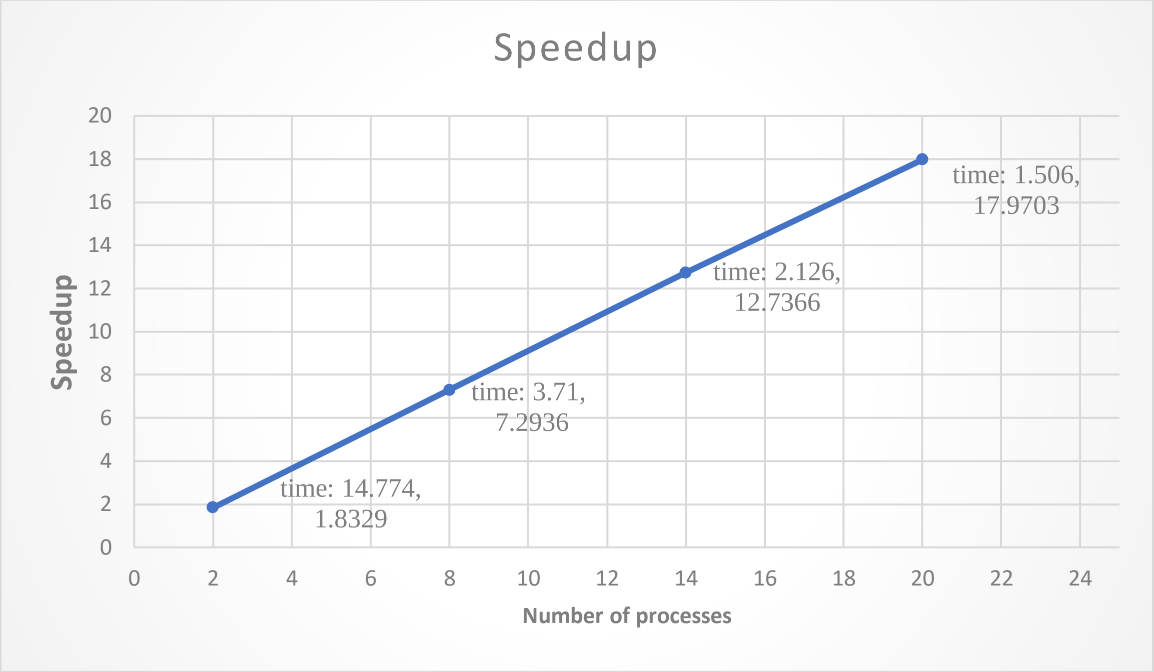
Efficiency for each configuration

	Runs		
	1	**2**	**3**
comm_sz (number of cores)			
2	.91645	.916175	.9162
8	.90592	.911711	.9086
14	.90976	.884979	.9084
20	.89851	.897541	.8842

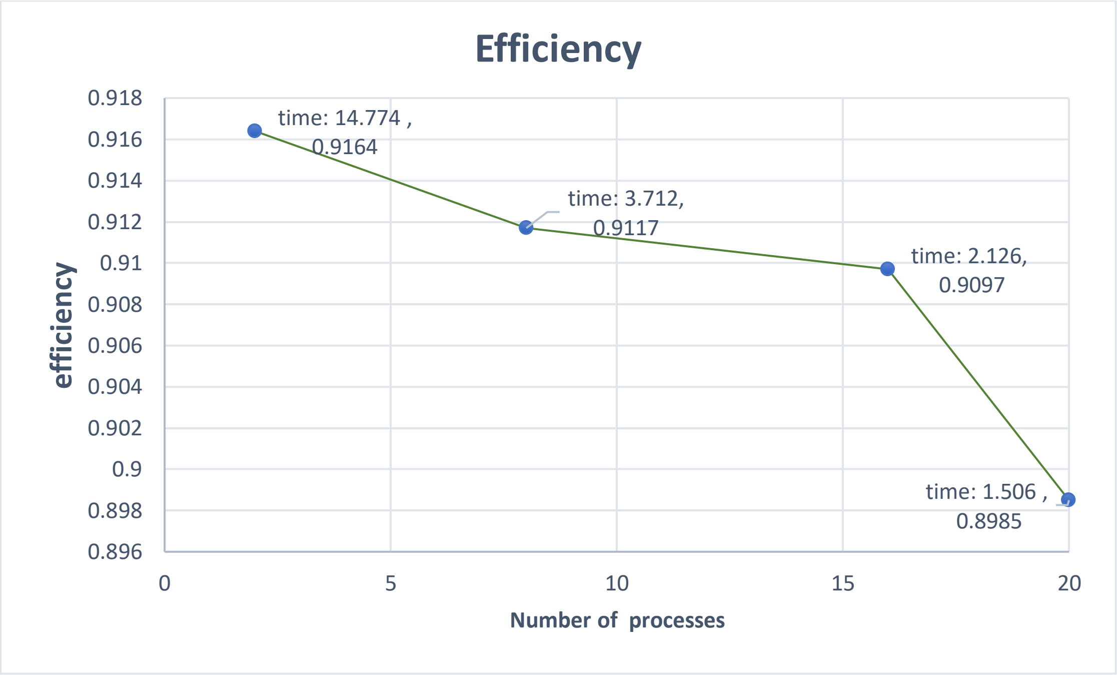
Table of minimum times with estimated integral, absolute relative true error,
And calculated speedup and efficiency.

comm_sz (number of cores)	**Minimum times**	**Estimated integral**	**ABSRTE (absoule relative true error)**	**Speedup**	**Efficiency**
2	14.774259	4.7540192288588e+03	5.0344230268089317103e-15	1.8329	.9164
8	3.712802	4.7540192288588e+03	5.0129379394862973857e-15	7.2936	.9117
14	2.126143	4.7540192288588e+03	5.0439511959694043240e-15	12.7366	.9097
20	1.506926	4.7540192288588e+03	5.0396541785048774588e-15	17.9703	.8985

Speedup Graph, minimum times



Efficiency Grpah, minimum times



Discussion of results and conclusion:

I observed that as number of processes increases, time of my program to finish decreases. In fact, if the size of the processes increases by almost doubling the number of processes, times also decreases by more than half the previous times, for instance, when running the program with 2 cores and 8 cores there's a huge improvement on times, for instance, times decrease from 14.772 to 3.7 seconds for minimum times. On the other hand, from 8 to 20 processes there is not a big difference as observed from 2 to 8, and 14 to 20 processes. However, it is still a big improvement that time decrease from 3.73 to 1.5 seconds. The common pattern that it is observed from the timings is the decrease of time as number of processes increases, and this is something that is expected because the work to calculate the approximate integral is shared among more processes, however, the more processes the less decrease of time is also observed as I mentioned from 2 to 8 and 14 to 20 processes.

To obtain perfect parallelization T1 (time to complete the task on processor) / TP (time for p processors) must equal number of processors and this is rarely achieved and sometimes is not possible to get this linear speedup. Having sad that, the speedups obtained by all the runs in my program were approximate to the number of processors and this is a good result. The number of processes that were the closest were 2 and 8, for 2 processes speedup was 1.8, and for 8 processes speedup was 7.2, so this is approximate to number of processors, which is favorable. On the other hand, as number of processors increases, speedup was close, but not as much as less number of processors, so for number of processes 14 and 20, speedup obtained for 14 was 12.73 and for process 20 speedup was 17.97. So, as the number of processes increases, speedup is not approximate to number of processors and the less processes the better speedup was achieved in my program.

When achieving an efficiency of 1 is considered perfect parallelization and is not likely to happen, however, most of the efficiency obtained for each run in my program was close to 1.0, so all the runs that were made by each processor resulted in a good efficiency. Like speedup, the less processors the better efficacy because with two processors an efficiency of .91 was reached and for 20 processors a efficiency of .89 was reach, so efficiency was near linear. the results of efficiency and speedup are represented on my graphs, and as I mentioned less processors have better results than more processors. The speedup's graph increases and the efficiency's graphs decreases.

In conclusion, I believe every run achieved a good result on efficiency and speedup, and it is unlikely to achieve perfect speedup and efficiency because of the communication between processes or some portions might not be fully parallelized. However, the results and the graph gave a better representation of how speedup and efficiency is achieved and how the number of processors affected these results.