Jeff Anderson
551 - project 3
3/29/2020

SPEEDUP AND EFFICIENCY OF PARALLEL NUMERICAL INTEGRATION WITH OPEN MPI

I ran my tmin program beginning at 40,000,000 and got a tmin of **20908204** and that is the number I fed into the ptrap program.
The absolute relative true error using n = 20908204 and p = 1 was
4.933896744519773457 85e-15.

The speeds for all runs are as follows:

Cores	Run 1 (in seconds)	Run 2 (in seconds)	Run 3 (in seconds)
1	**19.06785**	19.10964	19.06888
2	16.60170	16.74140	**14.23967**
8	**2.635107**	2.921991	2.676994
14	1.665140	1.605051	**1.558292**
20	**1.155780**	1.161628	1.163234

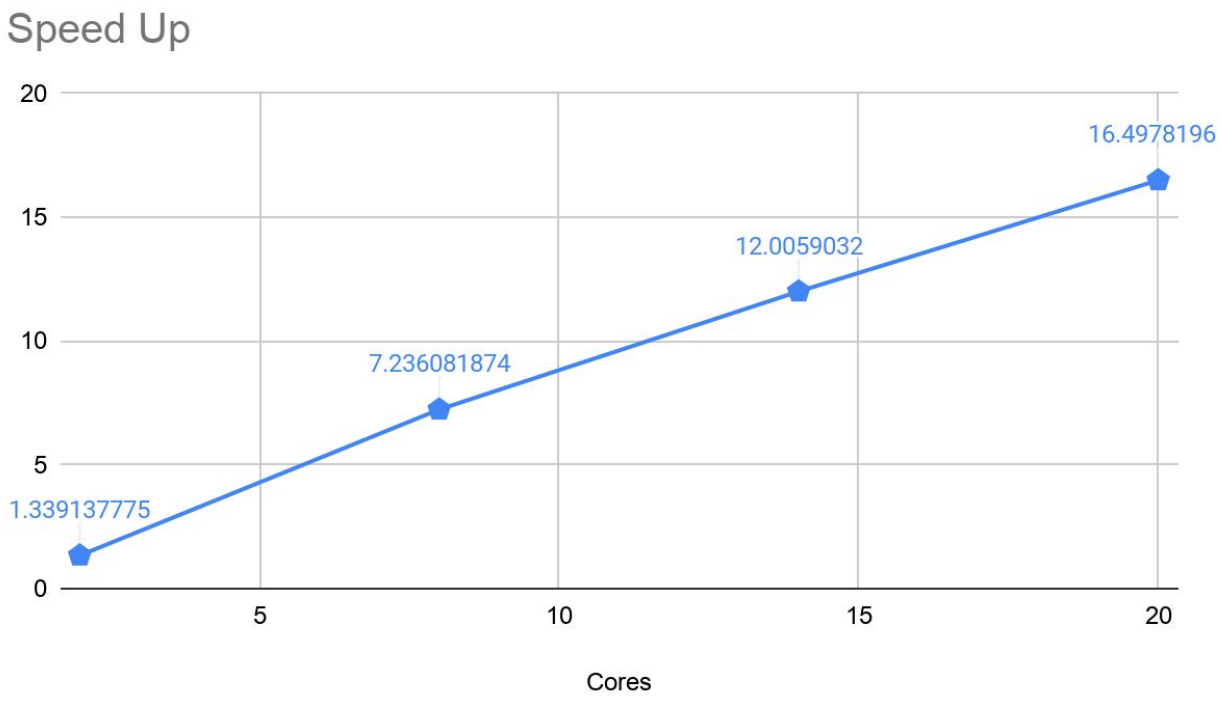
* a bolded value denotes the minimum time

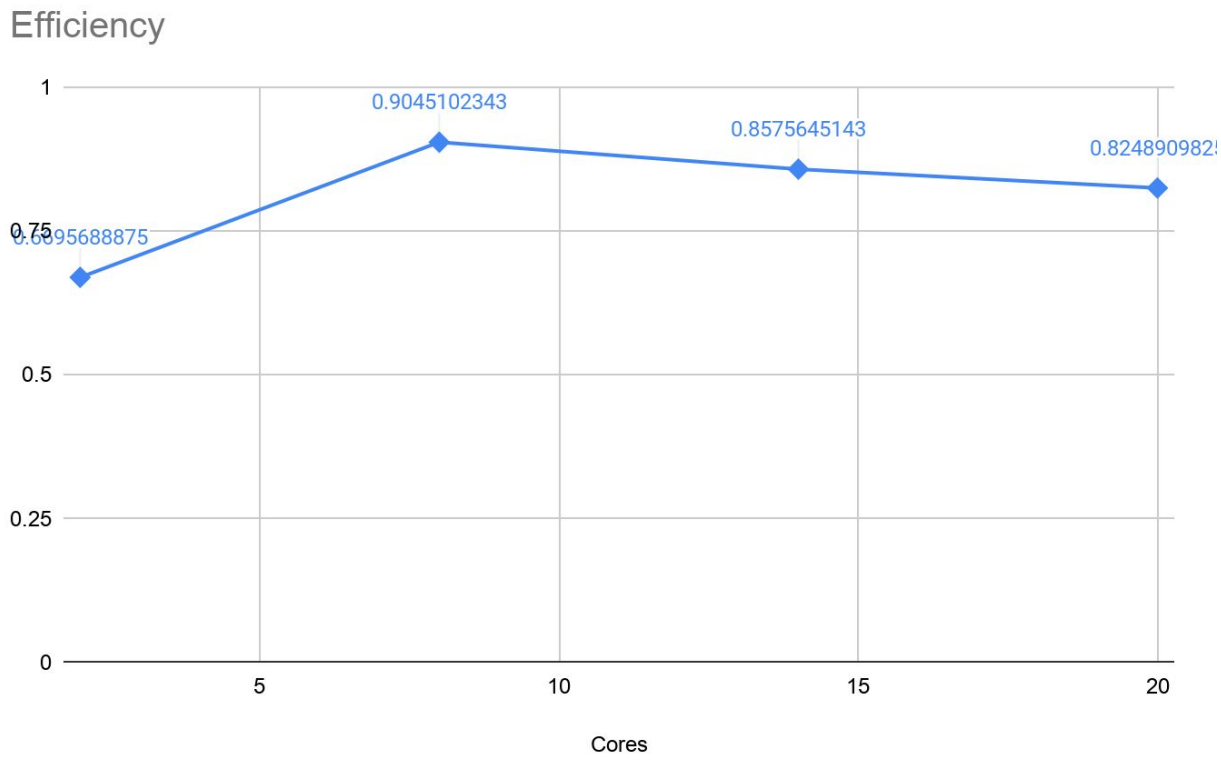
The table below shows minimum times, the estimated integral (%.13e), absolute relative true error (%.19e), and calculates speedup and efficiency.

Cores	**Minimum Time**	**Estimated Integral**	**ARTE**	**Speedup**	**Efficiency**
1	19.06785	6.7459948824470e+03	4.9338967445197735785e-15	1	1
2	14.23967	6.7459948824470e+3	4.9695108039306422695e-15	1.339137775	0.669568887 5
8	2.635107	6.745994882447007512 4e+03	2.4967361782377590486e-08	7.236081874	0.904510234 3
14	1.558292	6.7459947982329e+03	1.12483564181579138897e-0 8	12.0059032	0.857564514 3
20	1.155780	6.74599471040173e+03	2.4967361775202114560e-08	16.49781965	0.824890982 5

Jeff Anderson
551 - project 3
3/29/2020

Graphs for the Speedup and Efficiency





Jeff Anderson
551 - project 3
3/29/2020

Method for calculating Speedup:
$SU(p) = T_1 / T_p$, where T_1 is the time to complete the task on one processor, p is the number of processors, and T_p is the time for p processors to complete.

Method for calculating Efficiency:
$Ef(p) = SU(p) / p$

I had to really rely on the textbook for reference, and especially when it came to learning and writing with MPI. I also relied on this page https://computing.llnl.gov/tutorials/mpi/ for clarification.
As you can see from the graphs, I have somewhat generic speedup. It looks much like what we were shown in class, like Amdahl's Law.
My Efficiency graph resembles the graph in the notes if n=64, where it increases in efficiency between n=2 and n=8, but then declines from n=8 to n=14 and n=14 to n=20. I am still pleased with my efficiency graph. Since 1 is pure efficiency (and therefore really only possible in theory), I am glad that I got the results that I did.
Overall, the project was very difficult, it took a lot of self-learning and reading and studying Pacheco's examples, and I'm just happy to have something to turn in.