MPI Trapezoid Project

CSCI 551

Aaron McIntyre

Problem: Write a program using MPI to calculate the integral of the given function using the trapezoid rule. Find an N number of trapezoids that gives the correct answer up to 14 significant digits. Calculate performance.

Value of N = 2422000

I spent many hours trying to find out a good way to find n. I first tried guessing a good n. Then I tried a version of a binary search. It did not work because the absolute relative true error does not decrease linearly with respect to n. I eventually ended up using a brute force method to find my n. I ran the integral algorithm and compared the true error with the criteria of 5.0 e^-13. Originally I got a relatively low value for n but not enough significant figures. After the announcement I multiplied my absolute relative true error by 100 to convert it to a percentage before I compared it with the criteria. I incremented n by 1000 on every run until I found an n that fit the criteria. It is very likely that there is an N that is smaller than the one that I found. I was incrementing by 1000 so I was actually skipping 1000 values that could have produced the an n that fit the criteria. As an experiment I modified my program to increment by 20 and start at 1,000,000. The program is still running and has not found a suitable n. It is important to note that even incrementing by 20 is still skipping many values of n. The relative true error cannot be traced and is always jumping around. The only idea I could come up with to find the true min N would be to brute force on every value on n until you found one to match the criteria. I would be very interested to know if there is a better way to find the minimum number of trapezoids for 14 significant digits.

Program output:

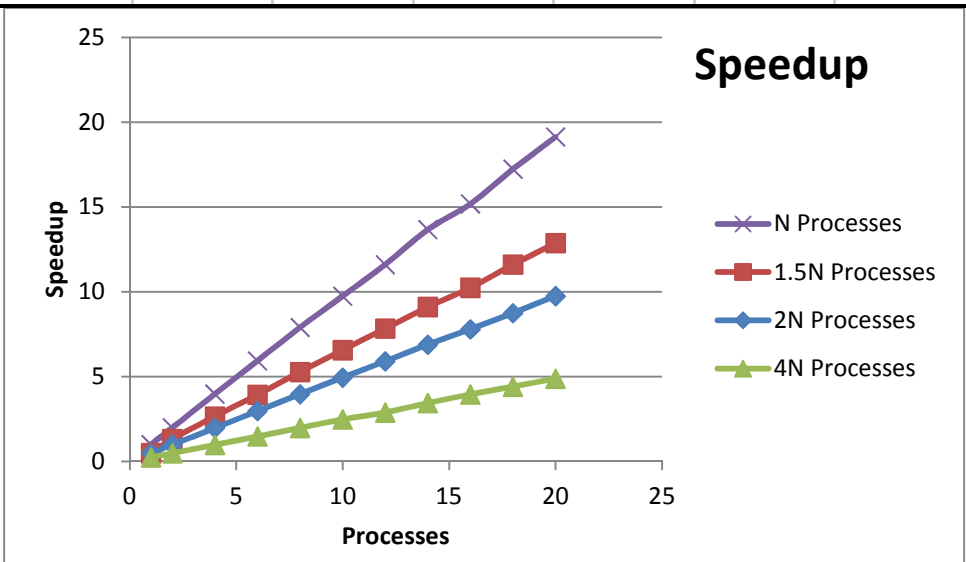
Elapsed Time = 7.213020e-02

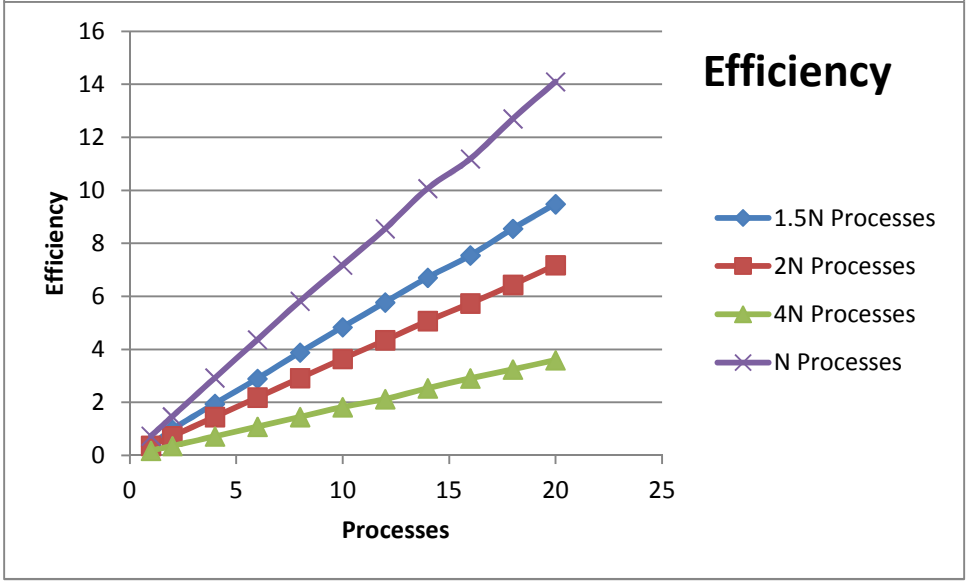
With n = 2422000 trapezoids, our estimate

of the integral from 100.000000 to 600.000000 = 4003.720900151336082

Absolute Relative True Error using N trapezoids = 2.271624e^-13 < 5.0e^-13

Times for each run n=N										
1 Proc	2 Proc	4 Proc	6 Proc	8 Proc	10 Proc	12 Proc	14 Proc	16 Proc	18 Proc	20 Proc
1.363311	0.682026	0.342876	0.230087	0.171727	0.140723	0.119272	0.101467	0.092172	0.078827	0.070925
1.3631	0.681762	0.34173	0.22886	0.172235	0.139226	0.118335	0.099505	0.089698	0.079678	0.071449
1.363211	0.683366	0.34145	0.229188	0.172196	0.139304	0.116919	0.099331	0.089369	0.078928	0.071206
1.363133	0.681708	0.341821	0.229444	0.172232	0.139441	0.118756	0.099703	0.089376	0.079521	0.073426
1.362911	0.681703	0.341631	0.228981	0.172183	0.139418	0.117654	0.099271	0.089842	0.078729	0.07213





Conclusions: The results from my program produce the expected graphs of speed up and efficiency. As you add processes the program speeds up and is more efficient. I was not able to show where speed up and efficiency slow down with respect to processes because it required many more processes to graph. My results from the scaled speedup experiments produced great speed up and efficiency lines that follow Gustafson's law. Modifying N and increasing processes shows diminished returns as the number of trapezoids increases.