Oregon State University

CS 475 - Spring 2021

Prof. Mike Bailey
mjb@cs.oregonstate.edu

Project #2

Numeric Integration with OpenMP Reduction

by
Abdullah Saydemir
saydemia@oregonstate.edu
April 21, 2021

Setup:

This experiment is run on an Ubuntu laptop with the following specifications.

- CPU: Intel i5-7300HQ @ 2.50GHz 4 cores
- RAM: 8 GB with 2400 MHz
- GPU: NVIDIA GeForce GTX 1050

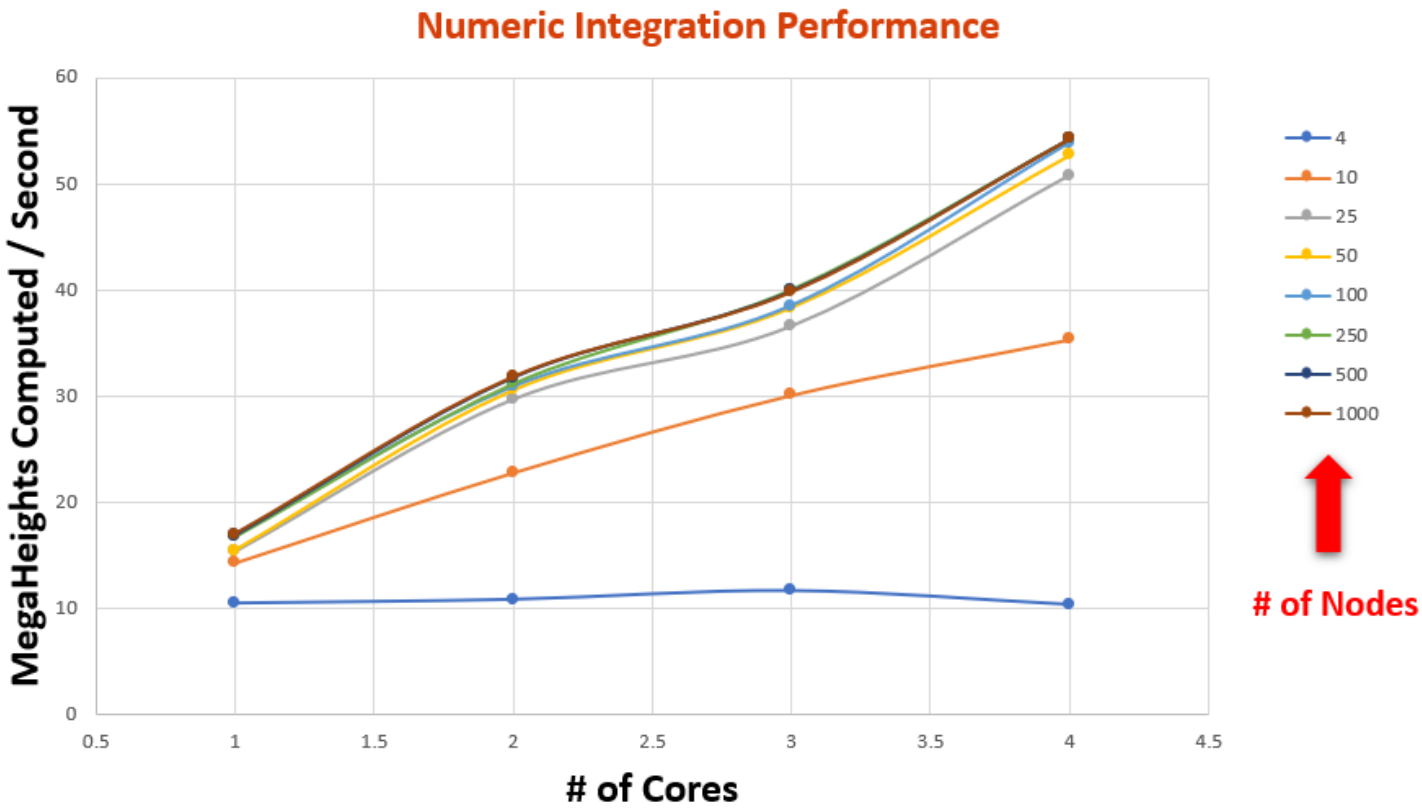
NUMT = [1, 2, 3, 4]
NUMNODES = [4, 10, 25, 50, 100, 250, 500, 1000]
# of Pins = NUMNODES^2
MegaHeights Calculated / sec = NUMNODES^2 / second

Results:

I think the actual volume is pretty close to *0.4358*. I get this value using 42K nodes. Here is the screenshot.

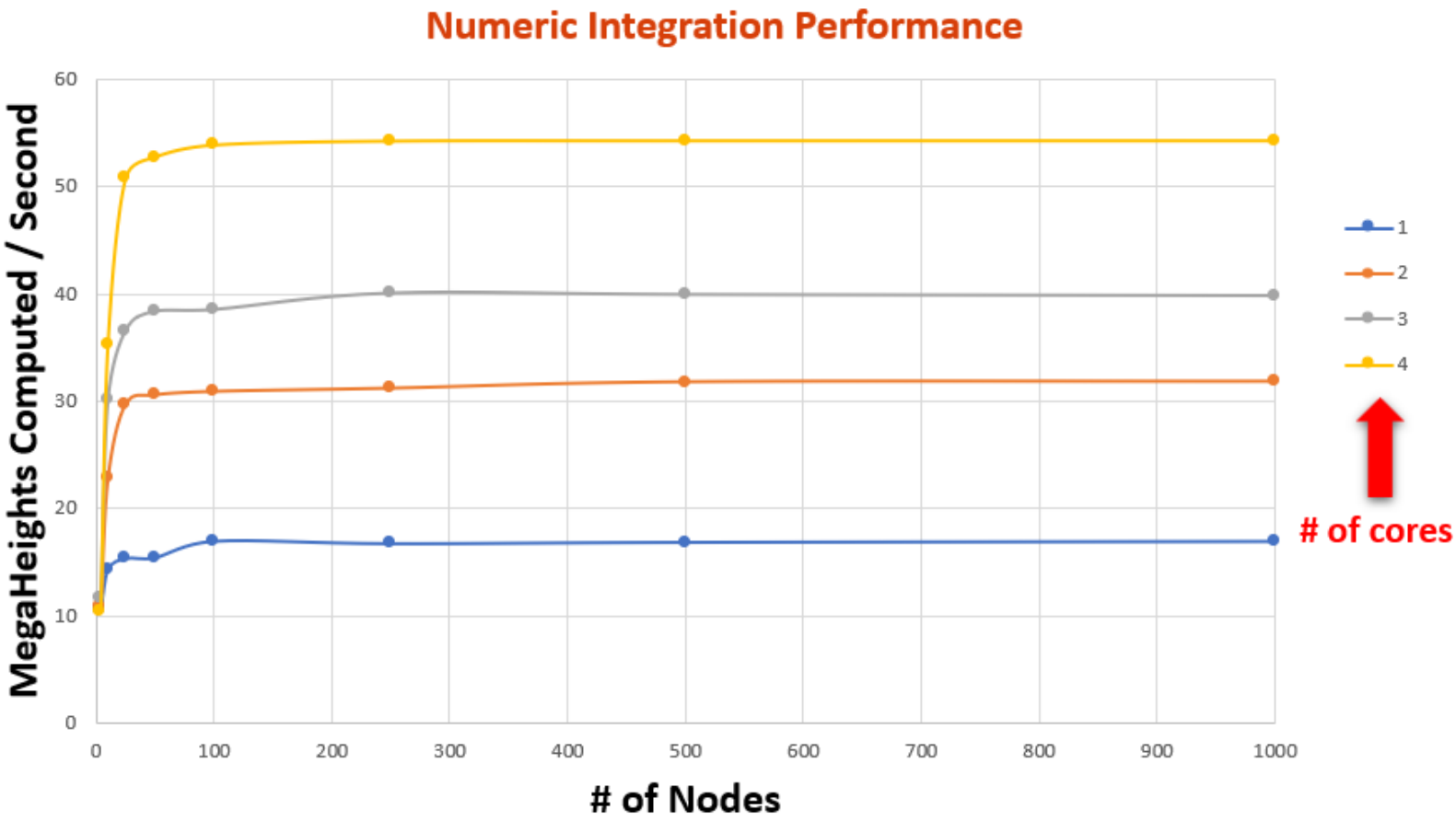
Here is the table based on collected data from the experiment.

	# of NODES							
# of Cores	4	10	25	50	100	250	500	1000
1	10.53	14.32	15.38	15.45	16.98	16.76	16.88	16.97
2	10.88	22.81	29.68	30.55	30.87	31.17	31.77	31.82
3	11.72	30.09	36.56	38.34	38.51	40.04	39.91	39.80
4	10.40	35.32	50.73	52.70	53.83	54.20	54.23	54.23



The first thing I noticed is that these graphs are pretty similar to those in the first project. However, I am also suspicious about the performance of 3 cores. For any other number of cores the graph is pretty much linear but 3 cores somehow breaks the pattern.

For one core there is not much difference in performance between different numbers of pins. For 625 pins and more the performance is pretty much similar since the number of cores is small. To see the performance gap between higher numbers of pins we need more cores. If we put some more cores we will also see that somewhere the lines will top, slightly fall and continue flat as lines for 16 and 100 pins did. Because after some point there will be empty threads waiting for jobs and the efficiency of the for loop will fall.



I think the problem with 3 cores can be seen more clearly in this graph. The performance gap between 2 - 3 cores and 3 - 4 cores is massively different and the gap is closer to the 2 core line. So, I don't want to include values for 3 cores to my calculations and I will not test 3 cores in my future projects.

The performance is poor and inconsistent for less number of cores and less number of pins. After a reasonable amount of pins the ratio of the performance is pretty close to the ratio of the number of cores. This suggests that the parallel fraction should be high. *(Please ignore 3 cores)*

Parallel Fraction *(F_P)*, **Speedup** *(S)* **and Maximum Speedup** *(S_{max})*

I can use performance values from the table to calculate *S* and using Inverse Amdahl, I can get F_P.

$$S = Performance\ with\ n\ threads\ /\ Performance\ with\ one\ thread$$
$$Fp = (n / (n - 1)) \ x \ (1 - (1 / S))$$

- 1 to 2 cores : $S = 31.87 / 16.97 = 1.8780$
  $Fp = (2 / (2 - 1)) \ x \ (1 - (1 / 1.8780)) = 0.9350$

- 1 to 4 cores : $S = 54.23 / 16.97 = 3.1956$
  $Fp = (4 / (4 - 1)) \ x \ (1 - (1 / 3.1956)) = 0.9161$

Then we can calculate F_P by taking the average of resulting values.

$$Fp = (0.9350 + 0.9161) / 2$$
$$= 0.9256$$

Using this parallel fraction, the maximum speedup we could ever get can be calculated using Amdahl's Law when $n \to \infty$.

$$S\,max = 1 / (1 - Fp)$$
$$= 1 / (1 - 0.9256)$$
$$= 13.44$$

It is also important to note that S_{max} is computed based on the parallel fraction that I calculated. S_{max} and F_P may slightly change if you run the code on some other systems.