Jessica Spokoyny
April 18, 2017
Project #1
Numeric Integration with OpenMP

1. My Machine:
   I ran this program on the flip server in linux from a windows 10 laptop. My main function is contained in a file called project0.cpp which I compiled by typing:

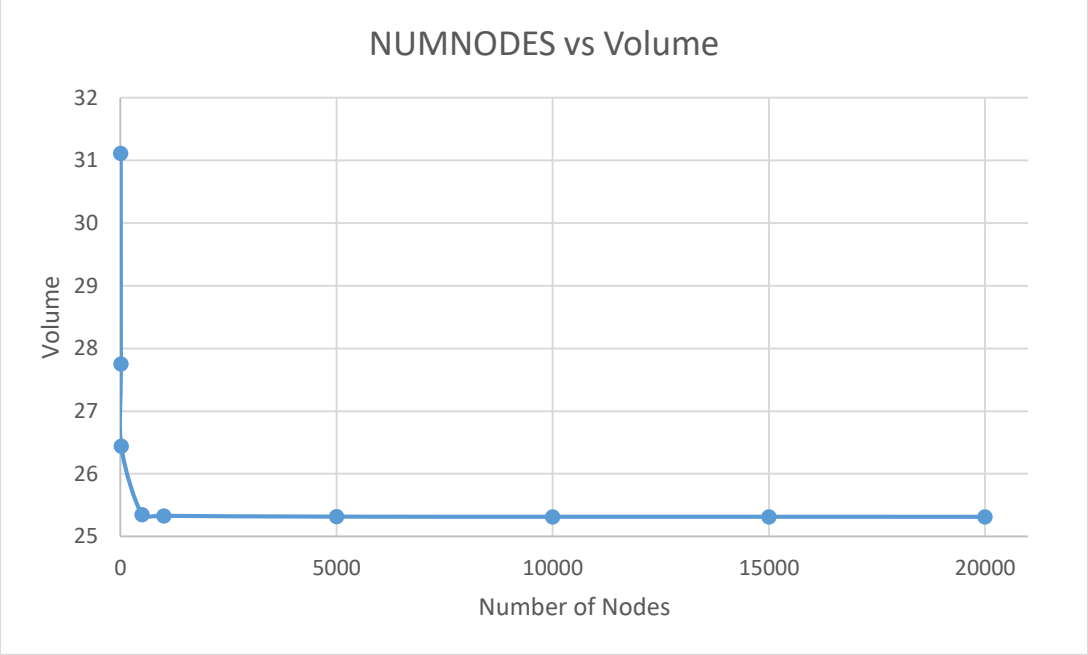
   % g++ -I/usr/local/common/gcc-5.4.0/ project1.cpp –o proj1 –O3 –lm –fopenmp

   Then I executed the program with:

   % ./proj1

2. Volume:
   I think the actual volume is **25.313**

   As can be seen in the graph below, when we increase the number of nodes used in the calculation, the resulting volume becomes closer and closer to the actual volume of the space.



3. My performance results:

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
4	1	10.34	31.111		
	2	6.64		0.642166344	-1.114457831
	4	5.84		0.564796905	-1.02739726
	8	5.59		0.540618956	-0.971121901
	16	2.78		0.268858801	-2.900719424
	30	0.39		0.037717602	-26.39257294

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
8	1	6.37	27.753		
	2	10.73		1.684458399	0.812674744
	4	19.14		3.004709576	0.88958551
	8	14.75		2.315541601	0.649297821
	16	9.54		1.497645212	0.354437456
	30	1.53		0.240188383	-3.272481406

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
16	1	12.84	26.443		
	2	13.8		1.074766355	0.139130435
	4	22.2		1.728971963	0.562162162
	8	30.99		2.413551402	0.66934034
	16	28.62		2.228971963	0.588120196
	30	6.4		0.498442368	-1.040948276

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
500	1	12.93	25.346		
	2	25.8		1.995359629	0.997674419
	4	49.35		3.816705336	0.983991895
	8	98.6		7.625676721	0.99298754
	16	115.27		8.914926527	0.947017148
	30	129.34		10.00309358	0.931066475

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
1000	1	12.9	25.329		
	2	25.63		1.986821705	0.993367148
	4	49.38		3.827906977	0.985014176
	8	98.65		7.647286822	0.993411049
	16	119.1		9.23255814	0.951133501
	30	132.69		10.28604651	0.933911294

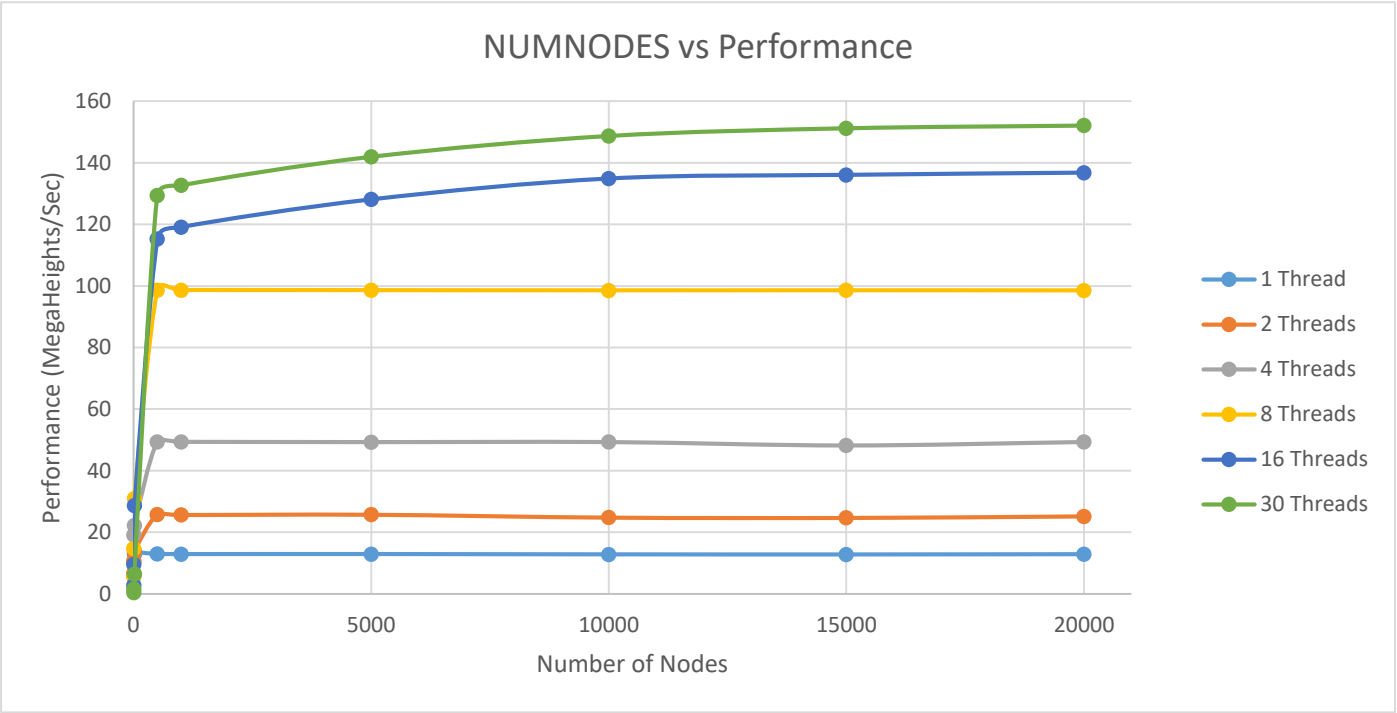
NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
5000	1	12.91	25.316		
	2	25.71		1.991479473	0.995721509
	4	49.29		3.817970565	0.984107662
	8	98.6		7.637490318	0.993219357
	16	128.1		9.922540666	0.959167317
	30	141.95		10.99535244	0.940399121

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
10,000	1	12.81	25.314		
	2	24.77		1.933645589	0.965684296
	4	49.3		3.848555816	0.986883029
	8	98.54		7.692427791	0.994288034
	16	134.87		10.52849336	0.965354292
	30	148.71		11.6088993	0.945371575

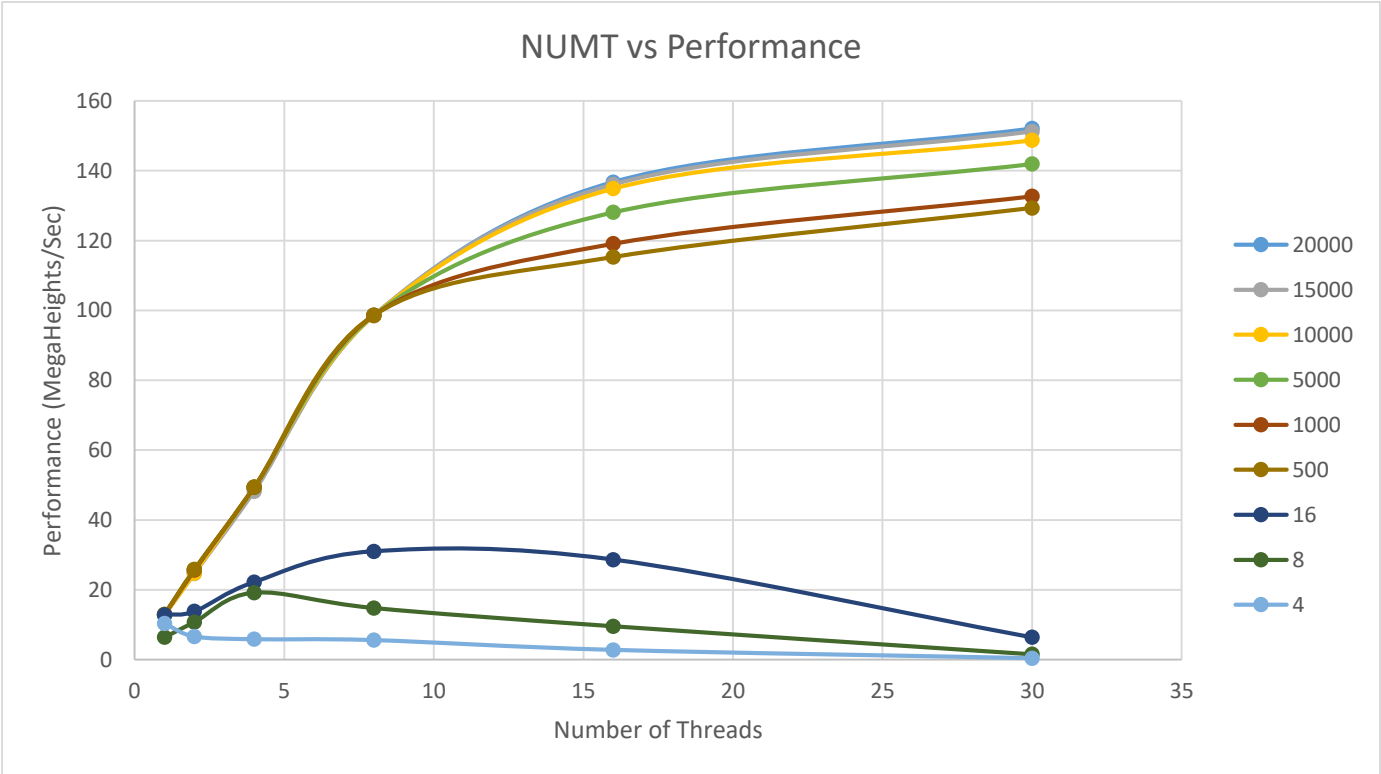
NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
15,000	1	12.78	25.314		
	2	24.67		1.930359937	0.963923794
	4	48.2		3.771517997	0.979806362
	8	98.56		7.712050078	0.994666048
	16	136.05		10.64553991	0.96646821
	30	151.21		11.83176839	0.947050118

NUMNODES	NUMT	PERFORMANCE	VOLUME	S	Fp
20,000	1	12.86	25.313		
	2	25.17		1.957231726	0.97814859
	4	49.32		3.835147745	0.985671803
	8	98.53		7.661741835	0.993693001
	16	136.77		10.63530327	0.966371768
	30	152.09		11.82659409	0.947011865

In the graph below, each line represents the performance given n number of threads. The x-axis shows the number of nodes used, and the y-axis shows the performance.



In the graph below, each line represents the performance given n number of nodes used to calculate the volume. The x-axis shows the number of threads available, and the y-axis shows the performance.



4. Speed Patterns I Noticed:

   Taking a look at the NUMT vs. Performance graph:
   We can see that for all values of NUMNODES > 16, the more threads added, the better the performance. For all of these values of NUMNODES > 16, the performances with 1, 2, 4 and 8 threads are almost identical. The differences arise when we use 16+ threads because the performance will vary depending on the size of the problem and how many calculations are to be performed. However, when NUMNODES is 16 or fewer, the more threads added, the worse the performance.

   If we look at the NUMNODES vs. Performance graph:
   We can see that for each NUMT, the performance first rises rapidly, then stabilizes to a constant. In other words, adding more nodes has no effect on the performance values given a certain number of threads.

5. Why These Behaviors Take Place:

   When the problem to be calculated is greater than 16 nodes x 16 nodes, adding more threads will increase the performance as there is still a sufficient portion of the problem that can be parallelized. Thus, adding more threads will make it run faster and more efficiently.
   When the problem to be calculated is less than 16 nodes x 16 nodes, adding more threads will not increase the performance because the problem cannot be parallelized any further. The multithreading setup time and resolution of the clock overshadow the value of adding more threads.

   Also, I believe that the performance converges to a constant for each NUMT value, regardless of the number of nodes added because the performance value is growing in a constant and consistent way. The performance figure consists of NUMNODES*NUMNODES as the numerator and elapsed time as the denominator, then this number is divided by 1,000,000 to get MegaHeights/Sec. As we increase the numerator, the denominator is also increasing by the same proportion.

   For example, if we set NUMNODES = 5000 and elapsed time = 24, our performance = 1.04 MegaHeights/Sec.
   Now, we can take NUMNODES = 10000 and elapsed time = 96, our performance is still 1.04 MegaHeights/Sec because both the numerator and denominator increased by a factor of 4.

6. Parallel Fraction for this application (using the Inverse Amdahl equation)

   To calculate each speed-up: $S = P_n/P_1$ where P = performance and n = 2, 4, 8, 16, 30

   To calculate each Parallel fraction: $F_p = (n/(n-1))*(1-(1/S)$

   The overall Parallel Fraction for this application = **0.9721**

   *Both the speed-ups and F_p's can be seen in the tables above.

**Because the performance values (and thus speed-up values) for NUMNODES <=16 displayed idiosyncrasies due to their small size, I did not include those in my calculations. These issues are discussed in question 5.

7. Maximum Speed-up:

   The Maximum speed-up: Ms = $1/(1- F_p)$ = 1/0.0279 = **35.824**