Project 2: Numeric Integration with OpenMP

Joel Huffman huffmajo@oregonstate.edu

Tables and Graphs

Table 1: Performance and Calculated Volume Across Thread and Node Sizes

Number of Threads	Number of Nodes	Average MegaHeights per second	Peak MegaHeights per second	Calculated Volume
1	100	16.06631	17.052453	28.68762
1	1000	17.458072	17.462539	28.6875
1	2000	17.10499	17.441793	28.6875
1	4000	17.071047	17.241434	28.6875
1	6000	17.069376	17.141106	28.6875
1	8000	17.075379	17.194886	28.6875
1	10000	17.062789	17.06473	28.6875
1	12000	17.066996	17.111174	28.6875
2	100	32.783559	33.899037	28.68762
2	1000	34.767693	34.830669	28.6875
2	2000	34.207649	34.732454	28.6875
2	4000	34.100598	34.183499	28.6875
2	6000	34.101996	34.224118	28.6875
2	8000	34.125003	34.379323	28.6875
2	10000	34.111331	34.28513	28.6875
2	12000	34.091789	34.098712	28.6875

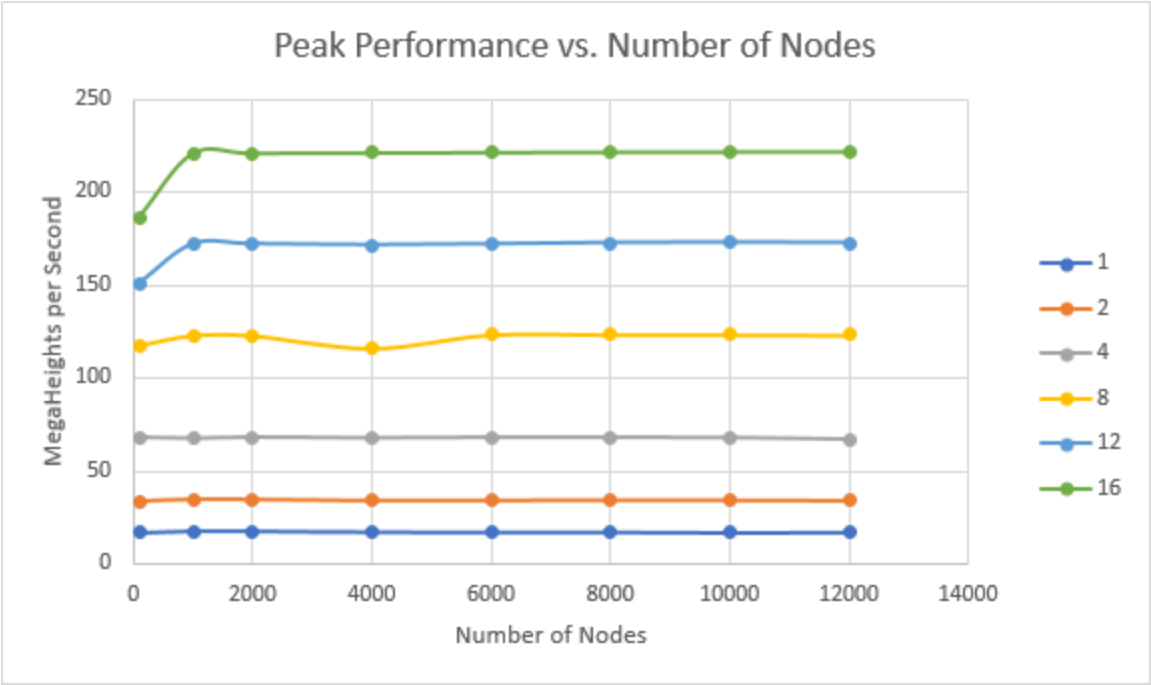
4	100	58.160236	68.211763	28.68762
4	1000	66.760357	67.690081	28.6875
4	2000	66.992941	68.196001	28.6875
4	4000	67.006333	67.902278	28.6875
4	6000	67.220256	68.129265	28.6875
4	8000	67.252117	68.141521	28.6875
4	10000	66.703007	67.999184	28.6875
4	12000	66.679537	67.2561	28.6875
8	100	103.946746	117.666468	28.68762
8	1000	122.611662	123.262893	28.6875
8	2000	122.88748	123.187143	28.6875
8	4000	115.767176	115.915463	28.6875
8	6000	123.598183	123.654382	28.6875
8	8000	123.290355	123.699278	28.6875
8	10000	123.364476	123.6363	28.6875
8	12000	123.146241	123.430227	28.6875
12	100	130.593291	151.061033	28.68762
12	1000	170.991619	172.147338	28.6875
12	2000	171.707693	172.207609	28.6875
12	4000	171.619711	171.720717	28.6875
12	6000	172.051527	172.130468	28.6875
12	8000	172.450526	172.805682	28.6875
12	10000	172.467171	173.066386	28.6875
12	12000	172.357116	172.848037	28.6875
16	100	165.044804	186.663043	28.68762
16	1000	218.788043	221.193592	28.6875

16	2000	220.1252	221.275631	28.6875
16	4000	221.215288	221.451814	28.6875
16	6000	221.503386	221.653851	28.6875
16	8000	221.744124	221.882864	28.6875
16	10000	221.728887	222.032866	28.6875
16	12000	221.564553	222.036276	28.6875

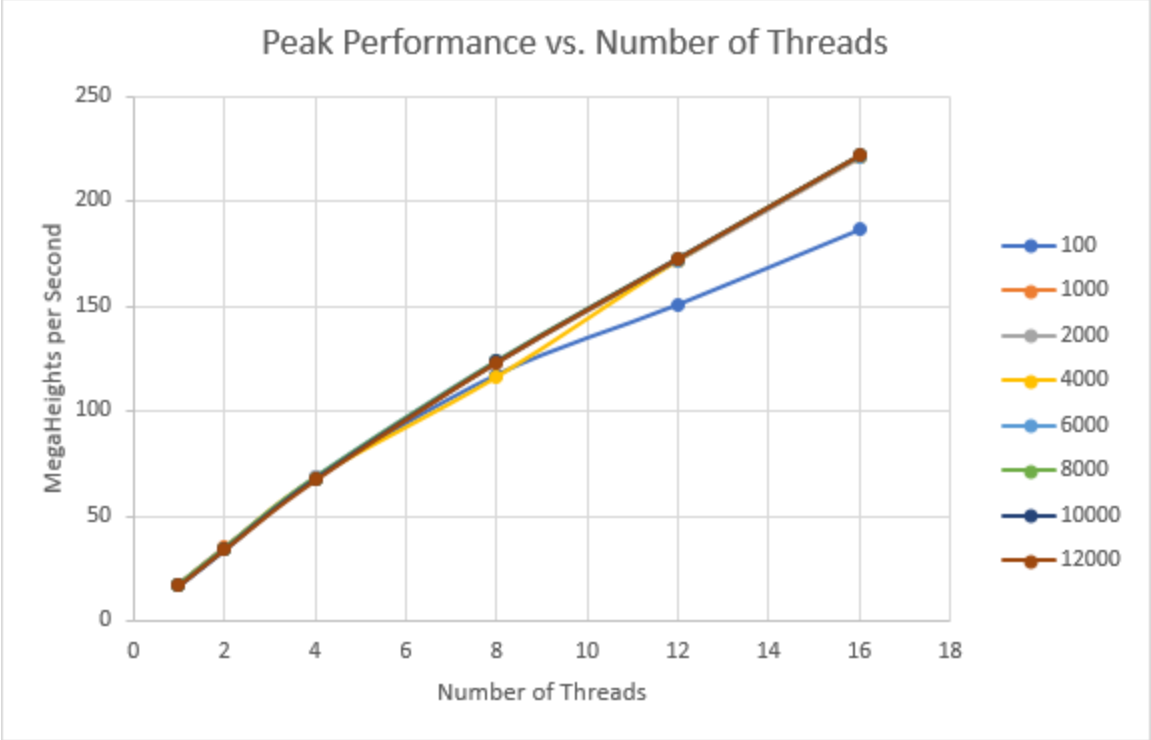
Table 2: Peak Performance Across Number of Threads and Number of Nodes

		Number of Threads					
		1	**2**	**4**	**8**	**12**	**16**
Number of Nodes	**100**	17.05	33.90	68.21	117.67	151.06	186.66
	1000	17.46	34.83	67.69	123.26	172.15	221.19
	2000	17.44	34.73	68.19	123.19	172.21	221.28
	4000	17.24	34.18	67.90	115.92	171.72	221.45
	6000	17.14	34.22	68.13	123.65	172.13	221.65
	8000	17.19	34.38	68.14	123.70	172.81	221.88
	10000	17.06	34.29	68.00	123.64	173.07	222.03
	12000	17.11	34.10	67.26	123.43	172.85	222.03

Graph 1:



Graph 2:



Explanation and Analysis

1. **Tell what machine you ran this on**

All testing/benchmarking was done on OSU's rabbit server.

2. **What do you think the actual volume is?**

I think the actual volume between the two surfaces is 28.69. I consistently got that value regardless of the number of threads or number of nodes (as seen in the Table 1).

3. **Show the performances you achieved in tables and graphs as a function of NUMNODES and NUMT**

See the above tables and graphs for information on performance.

4. **What patterns are you seeing in the speeds?**

Performance increases as the number of threads increases as seen in Graph 1, but the peak MegaHeights/second remain mostly unchanged as the number of nodes increases. In Graph 2 we see that same effect, but also that performance starts to dip with higher core counts in the lower numbers of nodes. We see a dip by nearly 50 megaHeights/second across when 16 cores either process fewer than 1000 nodes.

5. **Why do you think it is behaving this way?**

We are seeing performance dips when we use smaller numbers of nodes (especially on larger numbers of threads) because the serial part of our code is taking up a larger portion of our program when we only have a limited quantity of numbers to crunch. Performance is higher with the same number of threads but a higher number of nodes because we have more parallelizable code (and thus a higher parallel fraction) that can be utilized by the multiple threads.

What is the Parallel Fraction for this application, using the Inverse Amdahl equation?

Speedup calculations:

$S_2 = \frac{P_2}{P_1} = \frac{34.098712}{17.111174} = 1.99$

$S_4 = \frac{P_4}{P_1} = \frac{67.2561}{17.111174} = 3.93$

$S_8 = \frac{P_8}{P_1} = \frac{123.430227}{17.111174} = 7.21$

$S_{12} = \frac{P_{12}}{P_1} = \frac{172.848037}{17.111174} = 10.10$

$S_{16} = \frac{P_{16}}{P_1} = \frac{222.036276}{17.111174} = 12.98$

Fp calculations:

$Fp_2 = \frac{n}{n-1}\left(1 - \frac{1}{Fp}\right) = \frac{2}{1}\left(1 - \frac{1}{1.99}\right) = 0.99$

$Fp_4 = \frac{4}{3}\left(1 - \frac{1}{3.93}\right) = 0.99$

$Fp_8 = \frac{8}{7}\left(1 - \frac{1}{7.21}\right) = 0.98$

$Fp_{12} = \frac{12}{11}\left(1 - \frac{1}{10.10}\right) = 0.98$

$Fp_{16} = \frac{16}{15}\left(1 - \frac{1}{12.98}\right) = 0.98$

Averaged Fp = $\frac{0.99+0.99+0.98+0.98+0.98}{5} = 0.98$

From the above calculations, the parallel fraction is 0.98.

6. **Given that Parallel Fraction, what is the maximum speed-up you could *ever* get?**

Max Speedup from averaged Fp = $\frac{1}{1-Fp} = \frac{1}{1-0.98} = 50$

Max Speedup from peak Fp = $\frac{1}{1-Fp} = \frac{1}{1-0.99} = 100$

From the averaged Fp we just acquired, the maximum speedup is 50. If we look at our peak Fp obtained from the earlier speedup calculations we could obtain a maximum speedup of 100.