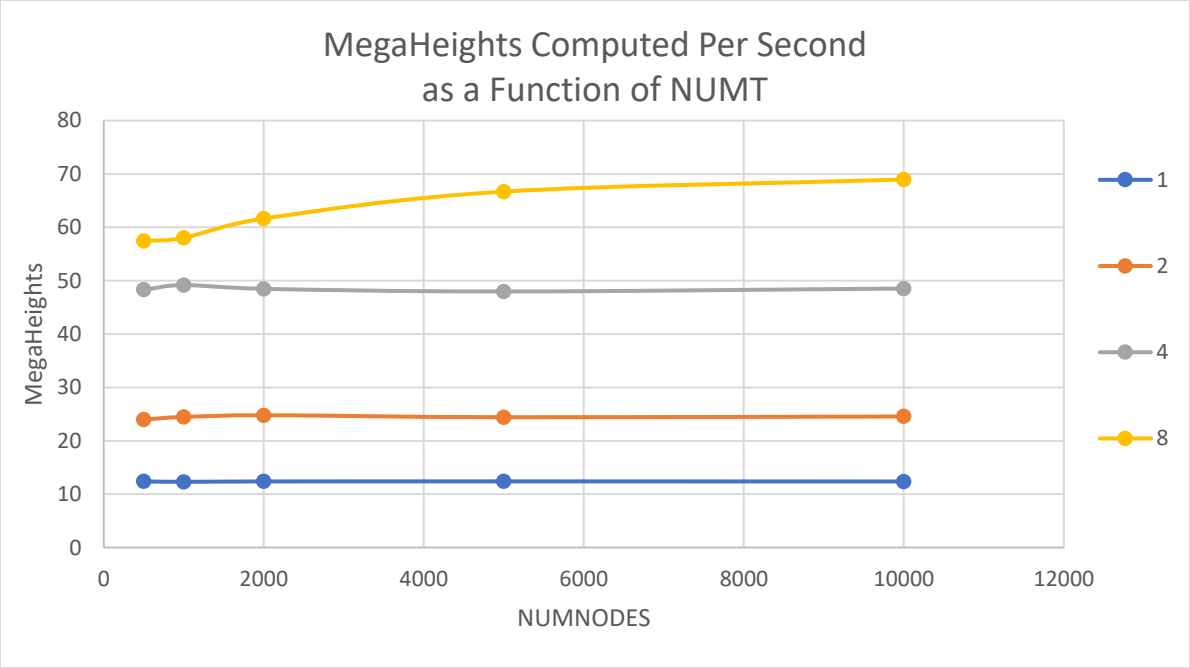
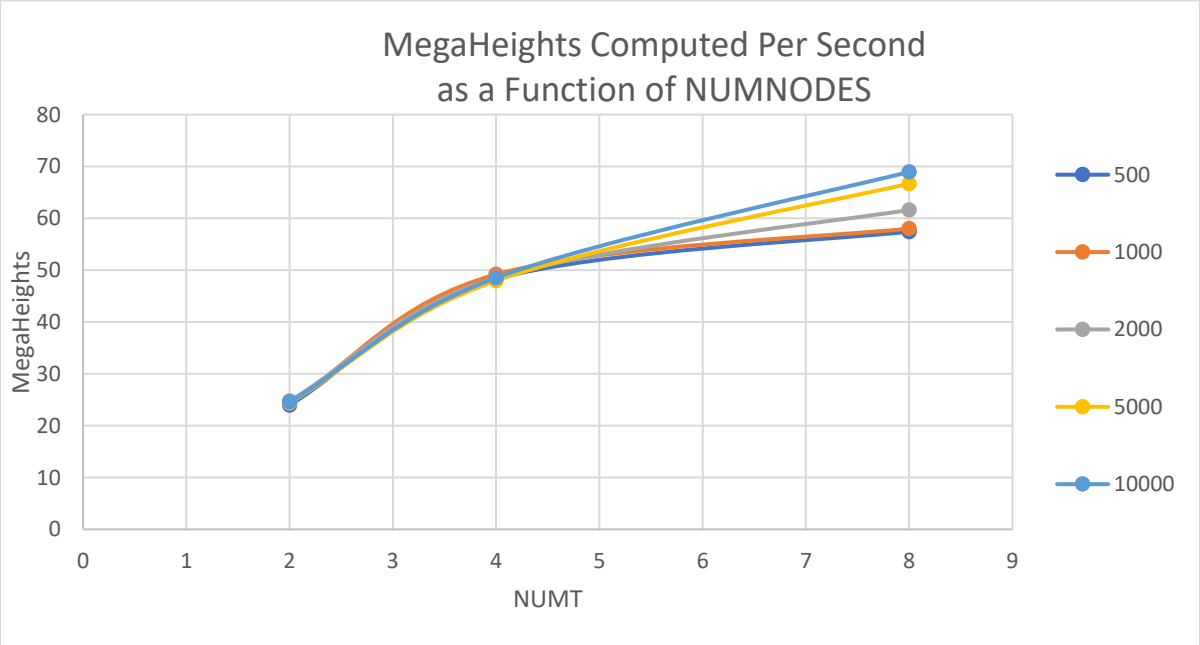
1. This was run on Flip1. More specifically:
flip1.engr.oregonstate.edu
Dual Intel(R) Xeon(R) CPU X5650 (6 cores each with Hyperthreading, 24 total threads)
96GB RAM

2. Based on the raw data that I obtained from running this code with several different nodes and number of threads, I believe the volume is somewhere around 25.3.

3. A number of tables and graphs that I used with the data that I obtained:





MegaHeights Computed Per Second					
NUMT	500	1000	2000	5000	10000
1	12.393674	12.31309	12.387524	12.3993	12.376266
2	23.975981	24.46917	24.786263	24.41905	24.557247
4	48.297634	49.18565	48.474072	47.97163	48.525753
8	57.430748	58.01518	61.625256	66.68888	68.95079

Raw Data			
NUMNODES	NUMT	Volume	Time
500	1	25.31249	0.020172
1000	1	25.31473	0.081214
2000	1	25.29679	0.322906
5000	1	25.20125	2.016242
10000	1	16	8.079982
500	2	25.31263	0.010427
1000	2	25.31303	0.040868
2000	2	25.29978	0.16138
5000	2	24.886	1.023791
10000	2	21.47362	4.072118
500	4	23.31252	0.005176
1000	4	25.31234	0.020331
2000	4	25.31784	0.082518
5000	4	25.53867	0.521141
10000	4	25.79083	2.060761
500	6	25.31252	0.005747
1000	6	25.31252	0.022426
2000	6	25.31036	0.069243
5000	6	25.2171	0.475438
10000	6	25.63435	2.055748
500	8	25.31249	0.004353
1000	8	25.31257	0.017237
2000	8	25.3124	0.064908
5000	8	25.2883	0.374875
10000	8	25.32872	1.45031

Used for storing Speedup and Fp calculations for 1-4 and and 1-2 NUMT

Speedup 1-4	Speedup 1-2	Fp 1-4	maxspeedup 1-4
0.9210	1	0.1144	1.1292
0.9999	0.9999	0.0013	1.0013
1.0008	1.0001	0.0153	1.0155
1.0134	0.9875	0.176	1.2136
1.3402	0.3421	0.3384	1.5114

4. Observing the graphs, I can see a clear pattern that more threads are increasing the speed of the calculations of the volume. This is much more pronounced after switching the X and Y axis and observing the graph as a function of NUMT.

5. This seems like a reasonable and expected pattern as increasing the number of threads doing the "work" of calculations by dividing the work would logically increase the speed of the calculations.

6. The Parallel Fraction for this application, using the Inverse Amdahl equation is Fp = n/(n-1) * T_1-T_n / T_1 or n/(n-1) * (1- (1/Speedup)). I used the latter equation after first calculating the Speedup.

7. If you used a million cores, you could calculate the Speedup to be 1,000,000. Then you could find the Fp with the equation above which would be 0.000001.

The max Speedup then calculated with 1/1-Fp would be 1.000001 which would be the max Speedup possible with a million cores.