Shawn Berg
bergsha@oregonstate.edu
April 23rd, 2019
CS 475

Project 2
Numeric Integration with OpenMP
Writeup

Machine:
This project was run on a Google Cloud Platform Compute Engine instance. The details of this machine are below:

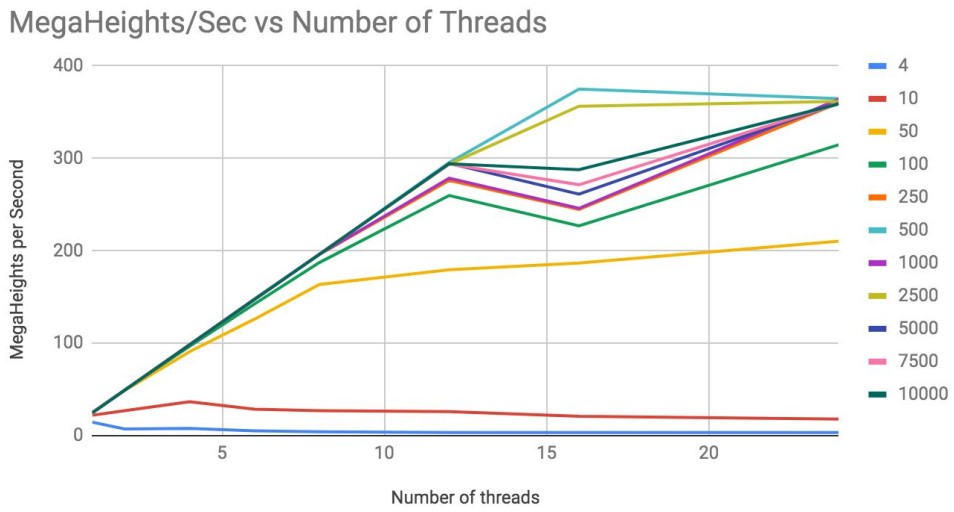
```
bergsm@more-threads:~/CS475/Project2$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) CPU @ 2.00GHz
Stepping:              3
CPU MHz:               2000.172
BogoMIPS:              4000.34
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              56320K
NUMA node0 CPU(s):     0-23
```

Table (Performance as a function of Threads vs Number of Nodes)

	4	**10**	**50**	**100**	**250**	**500**	**1000**	**2500**	**5000**	**7500**	**10000**
1	14.59	22.04	24.86	24.88	24.69	24.67	24.64	24.63	24.56	24.61	24.59
2	7.19	26.96	48.78	49.42	49.36	49.18	49.18	49.09	49.1	49.07	49.07
4	7.71	36.58	90.8	96.55	98.58	98.2	98.28	98.21	98.27	98.2	98.13
6	5.27	28.66	126.04	142.42	147.86	147.61	147.37	147.4	147.44	147.15	147.34
8	4.08	26.89	163.53	187.29	196.09	196.36	196.18	196.26	196.12	196.55	196.16
12	3.43	25.94	179.4	259.82	276.17	295.45	278.64	293.74	294.51	293.81	294.28
16	3.31	20.94	186.73	227.1	244.71	375.03	245.97	356.44	261.4	271.53	287.86
24	3.31	17.9	210.31	314.77	359.98	364.69	364.09	361.71	359.78	359.19	358.86

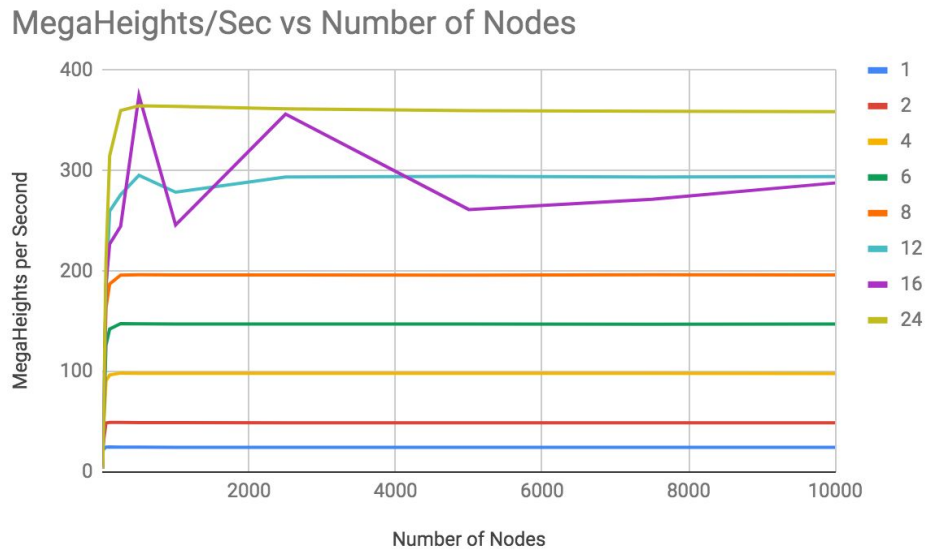
Units = MegaHeights calculated / sec

Graph 1 (MegaHeights per Second vs Number of Threads)



Each line represents number of trials

Graph 2 (MegaHeights per Second vs Number of Trials)



Each line represents number of threads.

Calculated Volume from run with most nodes: 28.6875

Speedup = P_{16} / P_1 = 375.03/24.67 = 15.202

Parallel Fraction = (16/15) * (1 - (1/15.202)) = 0.9965

Max Speedup = 1 / (1 - 0.9965) = 285.71

Commentary

Now that I have an environment and output script up and running, this project was pretty smooth sailing. All that was needed was to design the algorithm for calculating the volume. After I had this, I also needed to adjust the number of divisions to work with the $O(n^2)$ algorithm.

Analyzing the results from the Numeric Integration, you see a clear normalization of the volume as the number of Nodes increases. This would be expected behavior, as the volume is calculated by calculating the volume for each individual node. As the number of nodes increases, you're going to see a more accurate number for volume.

From a performance perspective, you generally see an increase in performance with an increase in threads, especially as the number of nodes increases to a point where utilization of the threads becomes relevant. You can see this more clearly in the MegaHeights/sec vs number of threads chart. For 1 and 10 threads, you really don't see a performance increase when utilizing more threads. This is most likely due to the overhead for using OpenMP, overshadowing the performance benefits of multithreading. There's also an interesting anomaly that occurs when using 16 threads. For all but the 500 and 2500 Node tests, the performance decreases and then increase when using 24 threads. For 500 and 2500 nodes, the opposite is true. That is, performance increases using 16 threads, and then decreases slightly using 24 threads. It's hard to say why this occurs, but if I had to make a guess, I would suspect it has something to do with caching behavior.