Josh Huff, huffj@oregonstate.edu
Professor Bailey
CS 475 – Spring 2017
April 20, 2017

Project 1: Commentary

1. Tell what machine you ran this on

My main workhorse, a home-built machine running Ubuntu 16.04 LTS (64-bit). The processor is Intel® Core™ i5-4690K CPU @ 3.50GHz × 4 and it has 15.6 GiB of usable RAM.

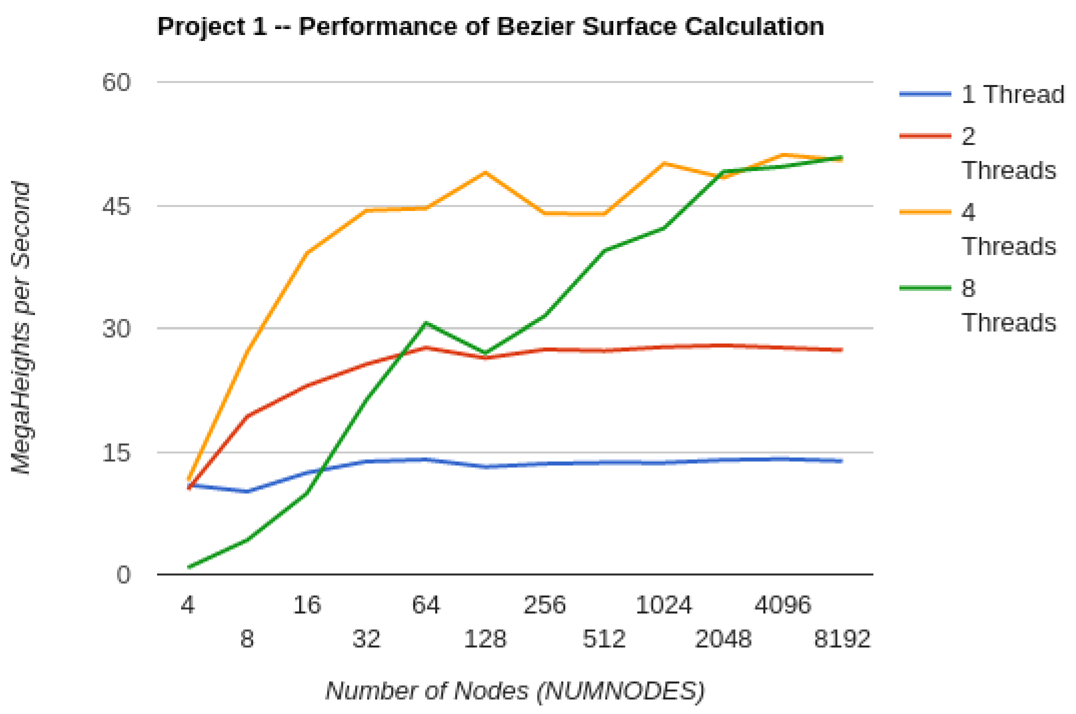
In contrast to my results with Project 0, the difference between peak and average performance was minimal, and the load averages were very low (less than 1) across the board. As such, there was no reason to ssh into flip2 to get the numbers used in this report.

2. What do you think the actual volume is?

The actual volume is 253.13 cubic units

There are considerable deviations from this number, particularly when the number of nodes is very small. Accuracy becomes reliable at 128 nodes.

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



Calculating X^2 Nodes with Y Threads (in MegaHeights Per Second)

Number of Nodes	Number of Threads			
	1	2	4	8
4	10.92	10.34	11.44	0.83
8	10.11	19.29	27.18	4.20
16	12.40	23.00	39.11	9.89
32	13.78	25.64	44.33	21.28
64	14.01	27.65	44.59	30.61
128	13.12	26.39	48.95	27.01
256	13.50	27.44	43.94	31.48
512	13.64	27.28	43.92	39.42
1024	13.62	27.74	50.06	42.17
2048	13.96	27.94	48.32	49.07
4096	14.08	27.65	51.12	49.68
8192	13.84	27.35	50.45	50.83

4. What patterns are you seeing in the speeds?

With one thread, the speeds are fairly uniform.
With two threads, except for the smaller node counts, the speed doubles effectively doubles.
With four threads, the top speed sees a roughly 66% increase from the two thread maximum.
With eight threads, there is no improvement over the four thread speeds – and execution noticeably lags at lower node counts.

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

Executing with one thread is our baseline; it is purely sequential, not parallel.

Executing with two threads splits the code into two paths and rejoins them, so barring the "overhead" necessary to make this division of labor possible, the speed is appropriately twice as fast for a significant number of nodes.

Executing with four threads does not double the speed, but merely increases it as much as possible.

Execution with eight threads takes a significant amount of nodes before it begins to perform comparably with previous runs. Even then, it does not exceed the performance of four threads. Around 50 MegaHeights per second, the program has reaches maximum parallelizability and will not see performance improvements no matter how many threads we assign to it.

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

The speedup (S) is 3.67 (using 8192 nodes as the benchmark)

(Avg Performance of 8 Threads) / (Avg Performance of 1 Thread)

S = 50.45 / 13.84 = 3.6726

Parallel Fraction (Fp) is 0.83

Fp = (8/7) * (1 - (1/S)) = 0.831679829

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

Max Speedup (Ms) is 5.88

1 / (1 – Fp) = 1 / (1-0.83) = 5.882352941