Project 1

Brandon Lee

Machine Specifications

I ran this project on the OSU's flip server through my 2015 MacBook Pro.

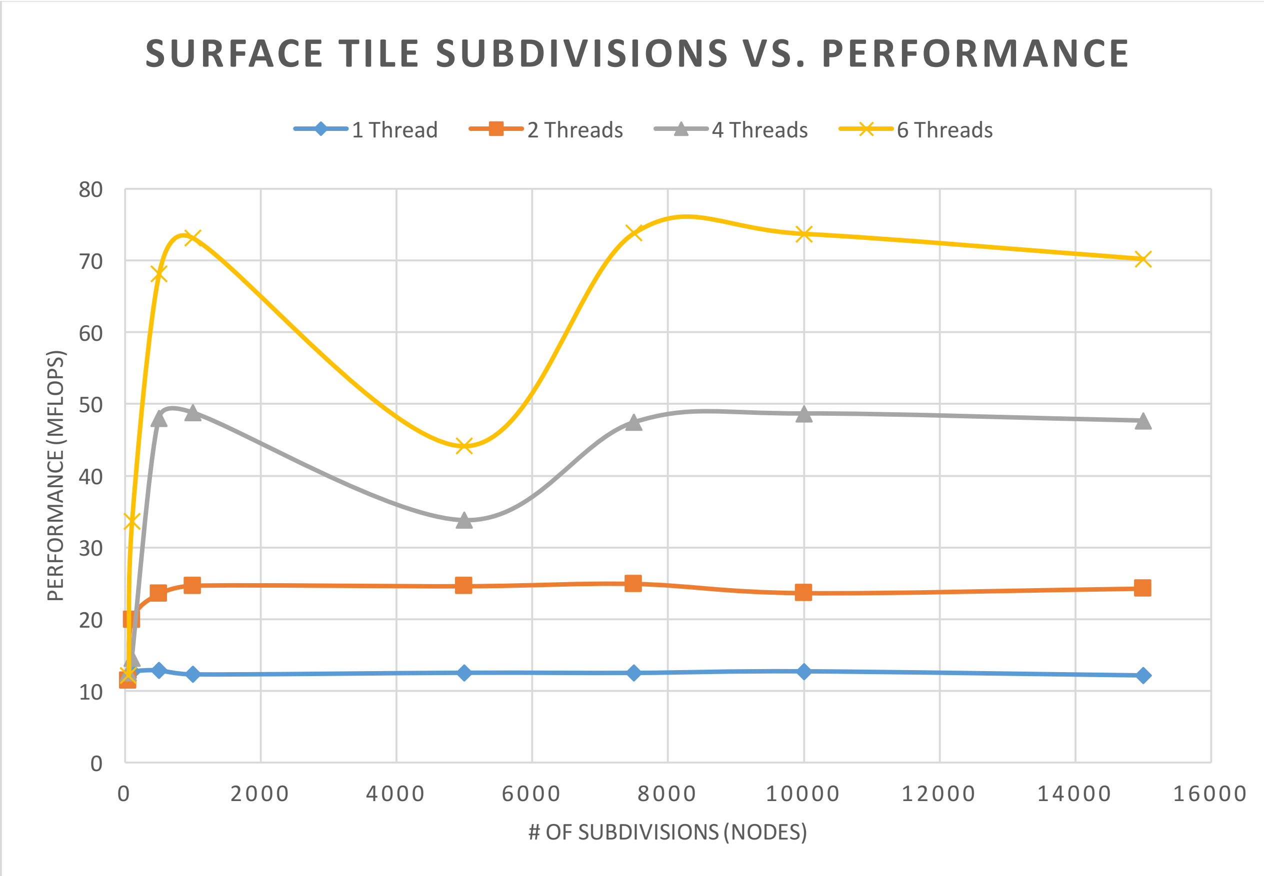
Actual Volume

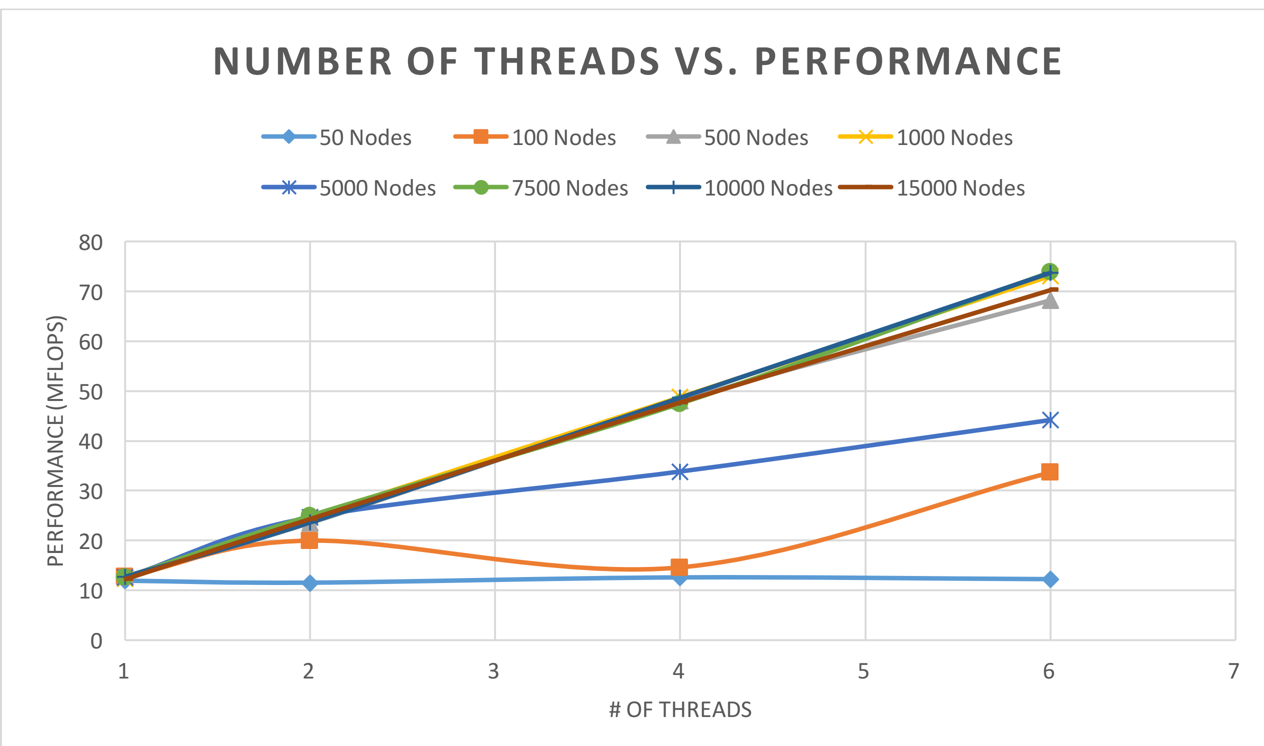
I believe that the actual volume between the two surfaces is around ~ 25.31. I believe that it is somewhere near this value as many iterations of my project with varying subdivisions and number of threads yielded values near this.

Show how the performances you achieved in tables and graphs as a function of NUMNODES and NUMT

	50 Nodes	100 Nodes	500 Nodes	1000 Nodes	5000 Nodes	7500 Nodes	10000 Nodes	15000 Nodes
1 Thread	11.87	12.67	12.88	12.3	12.53	12.51	12.73	12.16
2 Threads	11.48	19.95	23.54	24.66	24.59	24.91	23.63	24.26
4 Threads	12.53	14.56	48.01	48.77	33.79	47.43	48.64	47.65
6 Threads	12.18	33.67	68.17	73.08	44.12	73.8	73.69	70.2

***Values are represented in MFLOPs


SURFACE TILE SUBDIVISIONS VS. PERFORMANCE
1 Thread
2 Threads
4 Threads
6 Threads
PERFORMANCE (MFLOPS)
0
10
20
30
40
50
60
70
80
0
2000
4000
6000
8000
10000
12000
14000
16000
OF SUBDIVISIONS (NODES)


NUMBER OF THREADS VS. PERFORMANCE
50 Nodes
100 Nodes
500 Nodes
1000 Nodes
5000 Nodes
7500 Nodes
10000 Nodes
15000 Nodes
PERFORMANCE (MFLOPS)
0
10
20
30
40
50
60
70
80
1
2
3
4
5
6
7
OF THREADS

What patterns are you seeing in the speeds?

In both graphs, it is observed that as the number of threads increases, the overall performance increases as well. It is observed in the first graph that the performance appears to plateau as we continue to increase the number of subdivisions. In the second graph, we observe that the performance increase from the increasing number of threads is almost linear. Additionally, for my specific data set above - it is observed that there is a dip in performance near the 5000 subdivision mark in the first graph.

Why do you think it is behaving this way?

I think as the number of threads increases, the amount of delegated work is distributed to each thread, allowing for greater concurrency as the thread count increases. I think that the performance appears to plateau in the first graph due to the fact that there is a threshold of improvement that we can reach in this scope (Amdahl's Law).

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

In order to calculate the speedup, we will be comparing results by using the average performance of 1 thread versus using 6 threads.

We observe that:

$$Speedup = \frac{Avg.\, execution\ time\ of\ 1\ thread}{Avg.\, execution\ time\ of\ 6\ threads} = \frac{4146503.325\ microseconds}{1157558.27\ microseconds} = 3.58$$

Using inverse Amdahl's equation, we derive:

$$F = \frac{n}{n-1}\left(1 - \frac{1}{Speedup}\right) = \frac{6}{5}\left(1 - \frac{1}{3.58}\right) = 0.864$$

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

Using the parallel fraction obtained from the previous problem, we can calculate the maximum speedup through:

$$\max speedup = \frac{1}{1 - F_{parallel}} = \frac{1}{0.136} = 7.35$$