Brian Laccone
CS 475
4/25/3019
lacconeb@oregonstate.edu

Project 2

A. Tell what machine you ran this on

All project 2 executions were run on Flip 3.

B. What do you think the actual volume is?

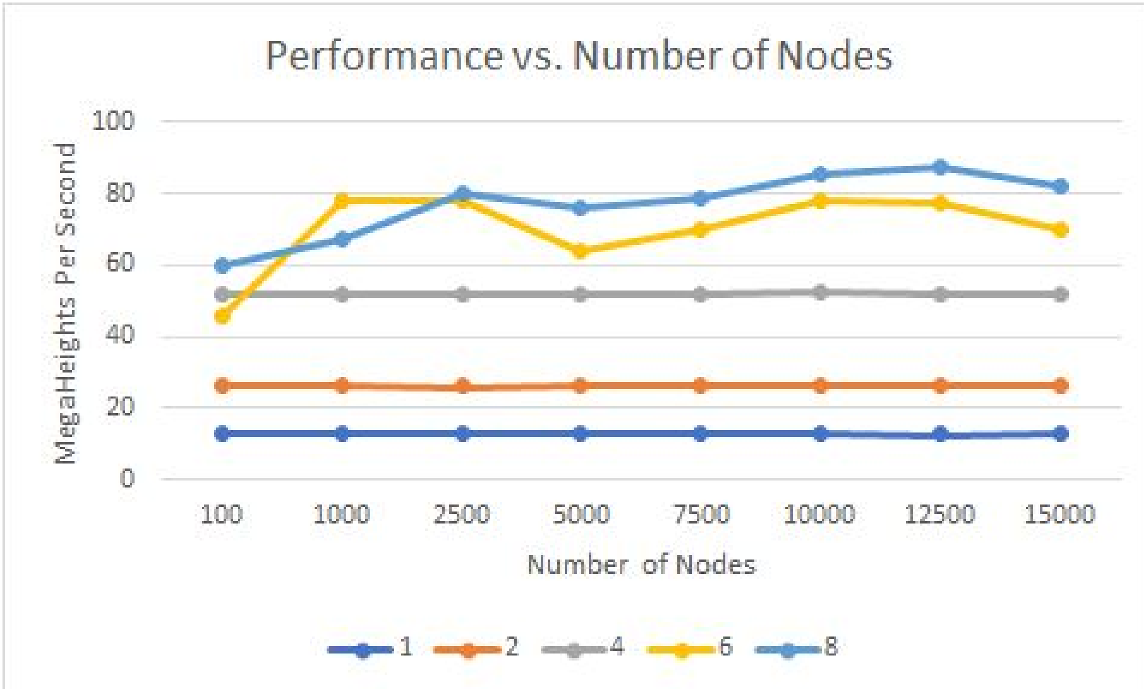
The volume that I got for every single thread and node combination was 28.687499. If I did my calculations correctly, then I believe that number is the correct volume or at least very close to the actual volume.

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

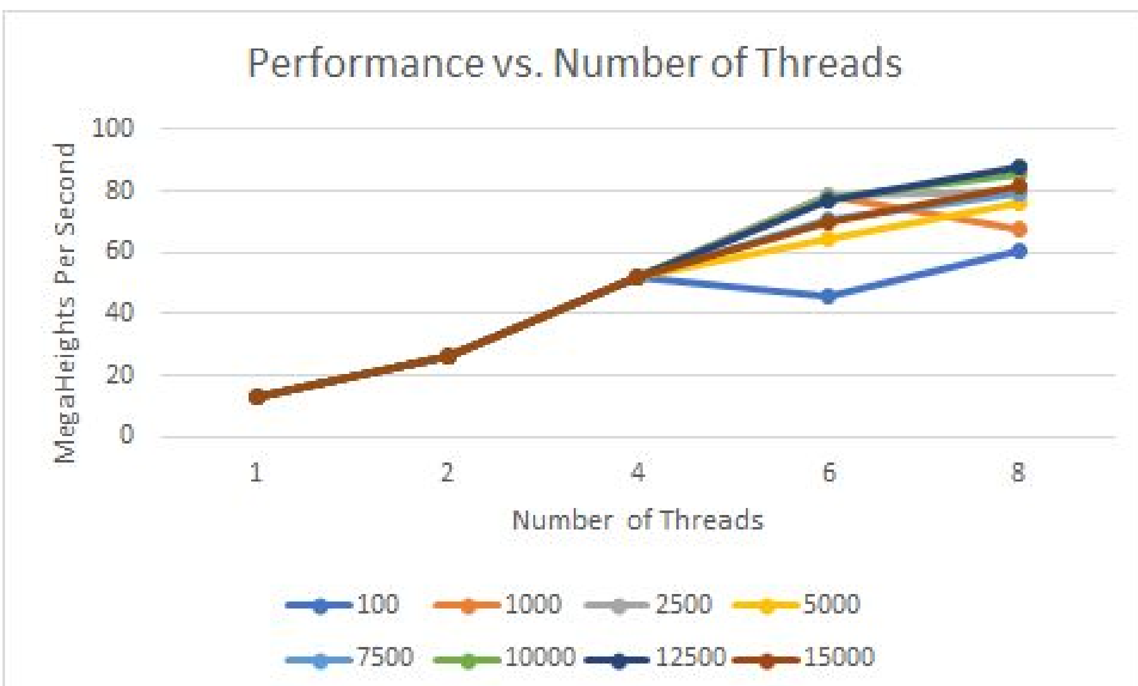
Table

	A	B	C	D	E	F	G	H	I
1		100	1000	2500	5000	7500	10000	12500	15000
2	1	13.1	13.078	13.068	13.067	13.069	13.061	13.055	13.062
3	2	26.091	26.122	26.012	26.061	26.115	26.013	26.127	26.118
4	4	51.77	52.164	51.945	51.987	52.003	52.257	52.103	52.11
5	6	45.732	78.292	78.174	64.254	70.299	77.836	77.102	69.945
6	8	60.151	67.193	79.906	76.289	79.063	85.186	87.693	81.892

Graph A: Performance vs. Number of Nodes



Graph B: Performance vs. Number of Threads



D. What patterns are you seeing in the speeds?

The performance speed patterns from this project were very similar to how I would have expected the results to be. As shown in the charts, performance increases the more threads are used. Threads 1,2 and 4 were the most consistent threads with the speeds being very similar no matter how many nodes were used. However, threads 6 and 8 were not as consistent as the previous 3 threads. The first 4 nodes used were very inconsistent for threads 6 and 8 but the last 4 nodes used showed more steady results. Another interesting result, not shown in the graphs, was that the peak performance was significantly higher than the average performance when using 8 threads. It took multiple tries to get the peak performance to be in the same ballpark as the average performance and some of the time I went with a high peak performance that was 10 above the average performance for 8 threads because after numerous tries, the gap did not decrease. If not for the time it took to run each node, I would have done higher nodes because it looked like 6 and 8 threads were on the decline and it would have been interesting to see how far that decline would go.

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

The fact that the speeds almost always increase the more threads you use, except when using small node values, shows that adding more threads is working as intended. If you have more threads that can work on multiple problems at the same time, the faster each program should run. Inconsistencies when using small node values is also to be somewhat expected. There simply isn’t enough values to take advantage of more threads. As far as why threads 6 and 8 are not as consistent as 1,2 and 4 threads, I do not know for sure. I can only speculate as to why those threads in particular are inconsistent in both project 1 and project 2. Amdahl’s law could possibly explain why the speeds seem a little bit weird as the threads increase. It could be that the overhead is more noticeable when the threads increase past 4. I would like to try that same experiment on another computer besides the OSU servers to see if the same things happen again.

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

2 Threads

A. 100 nodes

   S = 26.091/13.1 = 1.991679

   Fp = (2/(2-1)) * (1-1/1.991679) = 0.995822

B. 1000 nodes

   S = 26.122/13.078 = 1.9974

   Fp = (2/(2-1)) * (1-1/1.9974) = 0.998698

C. 2500 nodes

   S = 26.012/13.068 = 1.990511

   Fp = (2/(2-1)) * (1-1/1.990511) = 0.995233

D. 5000 nodes

   S = 26.061/13.067 = 1.994413

   Fp = (2/(2-1)) * (1-1/1.994413) = 0.997199

E. 7500 nodes

   S = 26.115/13.069 = 1.99824

   Fp = (2/(2-1)) * (1-1/1.99824) = 0.999119

F. 10000 nodes

S = 26.0.13/13.061 = 1.991655

Fp = (2/(2-1)) * (1-1/1.991655) = 0.99581

G. 12500 nodes

S = 26.127/13.055 = 2.001302

Fp = (2/(2-1)) * (1-1/2.001302) = 1.000651

H. 15000 nodes

S = 26.118/13.062 = 1.999541

Fp = (2/(2-1)) * (1-1/1.999541) = 0.99977

I. Average Fp

Average Fp = 0.997788

4 Threads

A. 100 nodes

S = 51.77/13.1 = 3.951908

Fp = (4/(4-1)) * (1-1/3.951908) = 0.995944

B. 1000 nodes

S = 52.164/13.078 = 3.988683

Fp = (4/(4-1)) * (1-1/3.988683) = 0.999054

C. 2500 nodes

S = 51.945/13.068 = 3.974977

Fp = (4/(4-1)) * (1-1/3.974977) = 0.997902

D. 5000 nodes

   S = 51.987/13.067 = 3.978495

   Fp = (4/(4-1)) * (1-1/3.978495) = 0.998198

E. 7500 nodes

   S = 52.003/13.069 = 3.979111

   Fp = (4/(4-1)) * (1-1/3.979111) = 0.99825

F. 10000 nodes

   S = 52.257/13.061 = 4.000995

   Fp = (4/(4-1)) * (1-1/4.000995) = 1.000083

G. 12500 nodes

   S = 52.103/13.055 = 3.991038

   Fp = (4/(4-1)) * (1-1/3.991038) = 0.999251

H. 15000 nodes

   S = 52.11/13.062 = 3.989435

   Fp = (4/(4-1)) * (1-1/3.989435) = 0.999117

I. Average Fp

   Average Fp = 0.998475

6 Threads

A. 100 nodes

S = 45.732/13.1 = 3.490992

Fp = (6/(6-1)) * (1-1/3.490992) = 0.856258

B. 1000 nodes

S = 78.292/13.078 = 5.986542

Fp = (6/(6-1)) * (1-1/5.986542) = 0.99955

C. 2500 nodes

S = 78.174/13.068 = 5.982094

Fp = (6/(6-1)) * (1-1/5.982094) = 0.999401

D. 5000 nodes

S = 64.254/13.067 = 4.917273

Fp = (6/(6-1)) * (1-1/4.917273) = 0.955962

E. 7500 nodes

S = 70.299/13.069 = 5.379.65

Fp = (6/(6-1)) * (1-1/5.379.65) = 0.976913

F. 10000 nodes

S = 77.836/13.061 = 5.959421

Fp = (6/(6-1)) * (1-1/5.959421) = 0.998638

G. 12500 nodes

S = 77.102/13.055 = 5.905936

Fp = (6/(6-1)) * (1-1/5.905936) = 0.996815

H. 15000 nodes

S = 69.945/13.062 = 5.354846

Fp = (6/(6-1)) * (1-1/5.354846) = 0.975904

I. Average Fp

Average Fp = 0.96993

8 Threads

A. 100 nodes

S = 60.151/13.1 = 4.591679

Fp = (8/(8-1)) * (1-1/4.591679) = 0.89396

B. 1000 nodes

S = 67.193/13.078 = 5.137865

Fp = (8/(8-1)) * (1-1/5.137865) = 0.920419

C. 2500 nodes

S = 79.906/13.068 = 6.114631

Fp = (8/(8-1)) * (1-1/6.114631) = 0.955952

D. 5000 nodes

S = 76.289/13.067 = 5.838295

Fp = (8/(8-1)) * (1-1/5.838295) = 0.947105

E. 7500 nodes

S = 79.063/13.069 = 6.049659

Fp = (8/(8-1)) * (1-1/6.049659) = 0.953945

F. 10000 nodes

S = 85.186/13.061 = 6.522165

Fp = (8/(8-1)) * (1-1/6.522165) = 0.96763

G. 12500 nodes

S = 87.693/13.055 = 6.717196

Fp = (8/(8-1)) * (1-1/6.717196) = 0.972718

H. 15000 nodes

S = 81.892/13.062 = 6.269484

Fp = (8/(8-1)) * (1-1/6.269484) = 0.960568

I. Average Fp

Average Fp = 0.946537

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

***Maximum Speed-up is calculated using the average Fp for each thread**

2 Threads

maxSpeedup = 1/(1-0.997788) = 100

4 Threads

*This is the best maximum speed up that I could achieve based on average Fp.

maxSpeedup = 1/(1-0.998475) = 100

6 Threads

maxSpeedup = 1/(1-0.96993) = 33.25599

8 Threads

maxSpeedup = 1/(1-0.946537) = 18.70458