Kevin Harvell
harvellk@oregonstate.edu

Project 2 - Numeric Integration with OpenMP

1. **Tell what machine you ran this on:**
   I ran this program on flip with uptime of ~3.

2. **What do you think the actual volume is?**
   Based on the average volume of all of my trials, I believe the volume is approximately 28.74 units cubed.

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

Raw Data

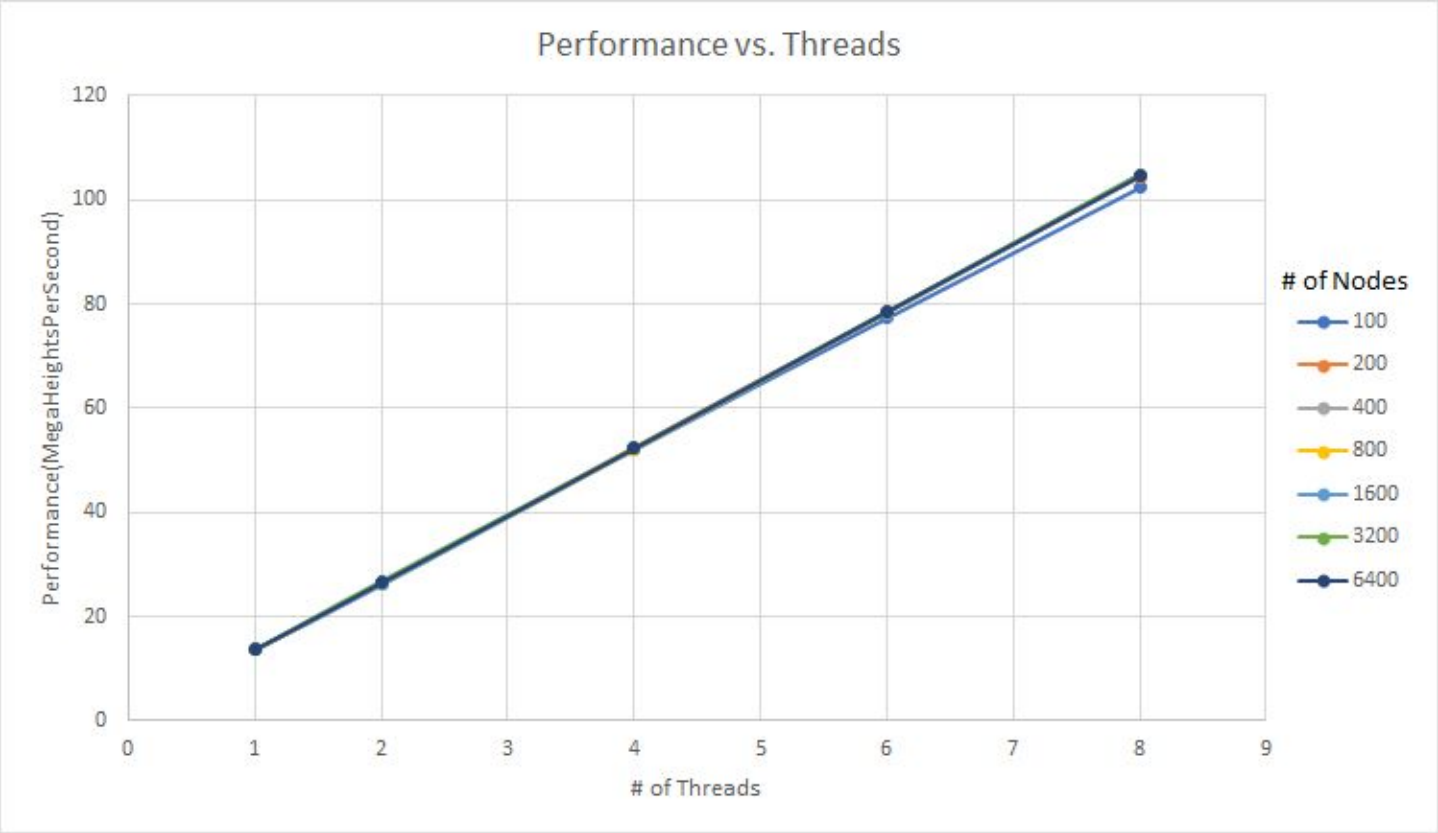
# of Threads	# of Nodes	Volume	Peak Performance (MegaHeightsPerSecond)
1	100	28.6876	13.7336
1	200	28.6876	13.5987
1	400	28.6873	13.6307
1	800	28.6869	13.5751
1	1600	28.6963	13.6092
1	3200	28.8015	13.6012
1	6400	29.9863	13.6244
2	100	28.6876	26.2139
2	200	28.6875	26.2629
2	400	28.6876	26.2504
2	800	28.6873	26.2333
2	1600	28.6848	26.1812
2	3200	28.5239	26.7958
2	6400	29.4244	26.4514
4	100	28.6876	52.0834

4	**200**	**28.6875**	**52.4885**
4	**400**	**28.6875**	**52.2013**
4	**800**	**28.6874**	**52.1037**
4	**1600**	**28.6867**	**52.2353**
4	**3200**	**28.7323**	**52.2816**
4	**6400**	**28.6606**	**52.2763**
6	**100**	**28.6876**	**77.3199**
6	**200**	**28.6875**	**78.537**
6	**400**	**28.6875**	**78.4357**
6	**800**	**28.6875**	**78.2584**
6	**1600**	**28.6875**	**78.3585**
6	**3200**	**28.6836**	**78.4106**
6	**6400**	**28.6984**	**78.5448**
8	**100**	**28.6876**	**102.322**
8	**200**	**28.6875**	**104.503**
8	**400**	**28.6875**	**71.5567**
8	**800**	**28.6875**	**104.799**
8	**1600**	**28.6875**	**104.8**
8	**3200**	**28.6826**	**104.704**
8	**6400**	**28.6433**	**104.571**

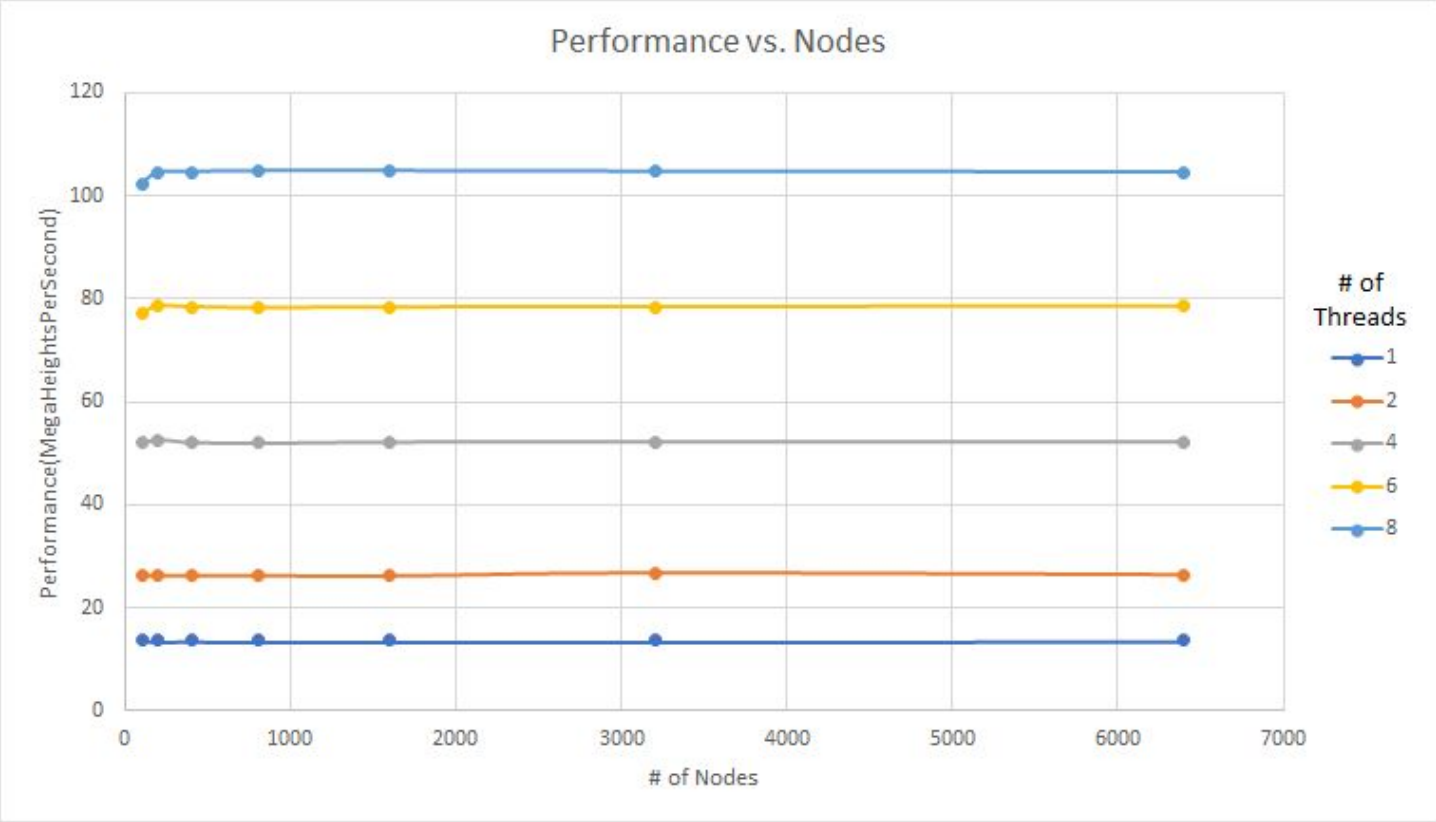
Kevin Harvell
harvellk@oregonstate.edu
Project 2- Numeric Integration with OpenMP

Formatted Data (# of Threads as rows, # of Nodes as columns, Mega Heights per Second in cells)

	100	200	400	800	1600	3200	6400
1	13.7336	13.5987	13.6307	13.5751	13.6092	13.6012	13.6244
2	26.2139	26.2629	26.2504	26.2333	26.1812	26.7958	26.4514
4	52.0834	52.4885	52.2013	52.1037	52.2353	52.2816	52.2763
6	77.3199	78.537	78.4357	78.2584	78.3585	78.4106	78.5448
8	102.322	104.503	104.651	104.799	104.8	104.704	104.571



Kevin Harvell
harvellk@oregonstate.edu
Project 2- Numeric Integration with OpenMP



4. **What patterns are you seeing in the speeds?**
   Speed seems to be increasing very close to linearly as cores increase. For example, performance for 2 threads is roughly twice as fast as performance for 1 thread. Also, performance seems to level out quickly independent of the number of the number of nodes. For example, for 8 threads, speed is fairly close to 104 MegaHeightsPerSecond as the number of nodes increases. It would be interesting to test greater node size, but the time it took beyond 10,000 nodes seemed counterproductive.

5. **Why do you think it is behaving this way?**
   Because we're using the reduction method in order to divide and conquer the numerous height/volume operations, we are getting a very efficient means of dividing the labor. Speedup is nearly directly impacted by the number of threads thrown at the problem. This indicates that a very high percentage of this program is being performed in parallel. Good thing we have the reduction method.

6. **What is the Parallel Fraction for this application, using the Inverse Amdahl equation?**
   The speedup using 1 core, 6,400 nodes and 4 cores, 6,400 nodes is:
   S = 52.2763 / 13.6244 = 3.837

   Parallel Fraction(Fp) = (4 / 3) * (1 - (1 / S))

   = (4 / 3) * (1 - 1 / 3.837)) = .986

Kevin Harvell
harvellk@oregonstate.edu

The following chart uses the above formulas for each of the threads I tested with 6,400 nodes:

Number of Threads	Speed Up	Parallel Fraction
1	-	-
2	1.941473	0.969854
4	3.836962	0.985836
6	5.76501	0.991848
8	7.675274	0.993956

7. **Given that Parallel Fraction, what is the maximum speed-up you could ever get?**
   Using 8 cores for calculations:
   max speedup = 1 / (1 - Fp) = 1 / (1 - .993956) = 165.45

   The following chart uses the above formulas for each of the threads I tested with 6,400 nodes and includes other maximum speedup numbers:

Number of Threads	Speed Up	Parallel Fraction	Max Speedup
1	-	-	-
2	1.941473	0.969854	33.17206
4	3.836962	0.985836	70.6023
6	5.76501	0.991848	122.6649
8	7.675274	0.993956	165.453

Kevin Harvell
harvellk@oregonstate.edu
Project 2- Numeric Integration with OpenMP

Summary:
I thought this was another cool example of Parallel programming. I am beginning to see why the reduction method is the preferred means of accumulating into a variable when using parallel programming. I am very impressed by the speedup numbers I have been getting. It is clear to me that parallel programming can make a significant difference in the performance in many different applications.