CS 475/575 -- Spring Quarter 2021

Project #2

Numeric Integration with OpenMP Reduction

Name: Chun-Yu, Chen

E-mail: chench6@oregonstate.edu

1) Tell what machine you ran this on:

- OSU Engineering server: flip3

2) What do you think the actual volume is?

- **Volume: 0.435723**
- The number of threads: 8
- The number of nodes: 3000
- MaxPerformance = 29.78 MegaHeights/Second

3) Show the performances you achieved in tables and two graphs showing:

- **Table**

Nodes / Threads	10	100	1000	1500	2000	2500	3000
1	1.81	4.36	4.36	4.46	4.46	4.47	4.44
2	0.87	7.88	8.86	8.85	8.7	8.83	8.58
4	0.69	11.29	15.15	16.51	16.56	16	15.99
6	0.56	16.79	23.45	22.38	20.39	19.36	23.88
8	0.46	17.73	25.8	29.84	29.03	29.81	29.78

Unit: MegaHeightsPerSeconds

- **Graphs**

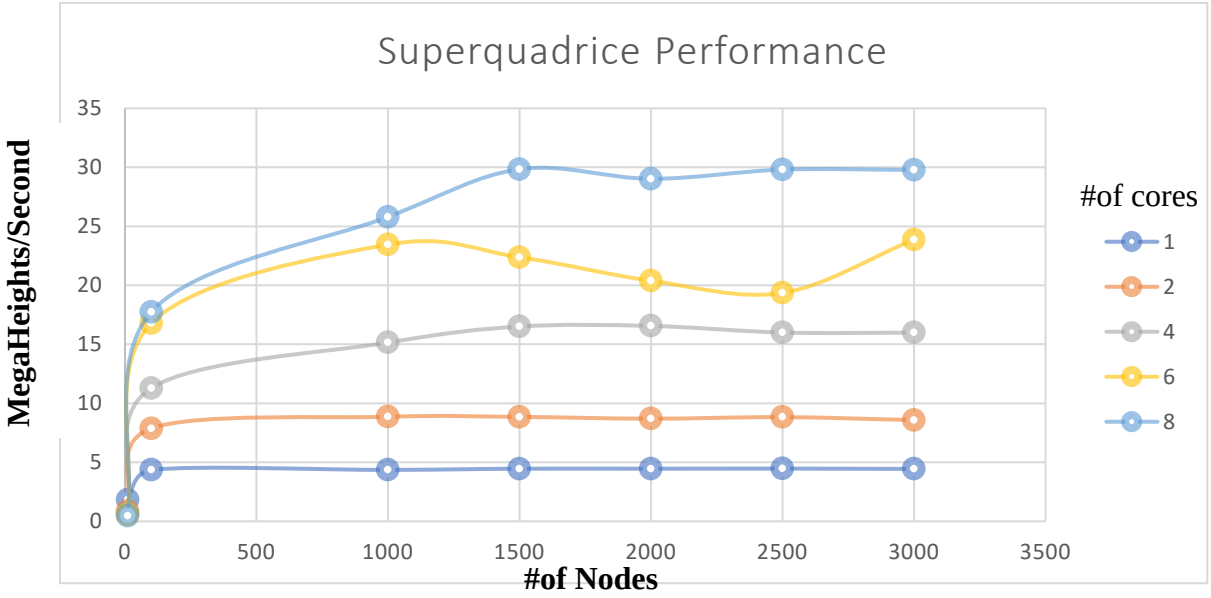

Figure 1:Graph of performance vs. number of nodes

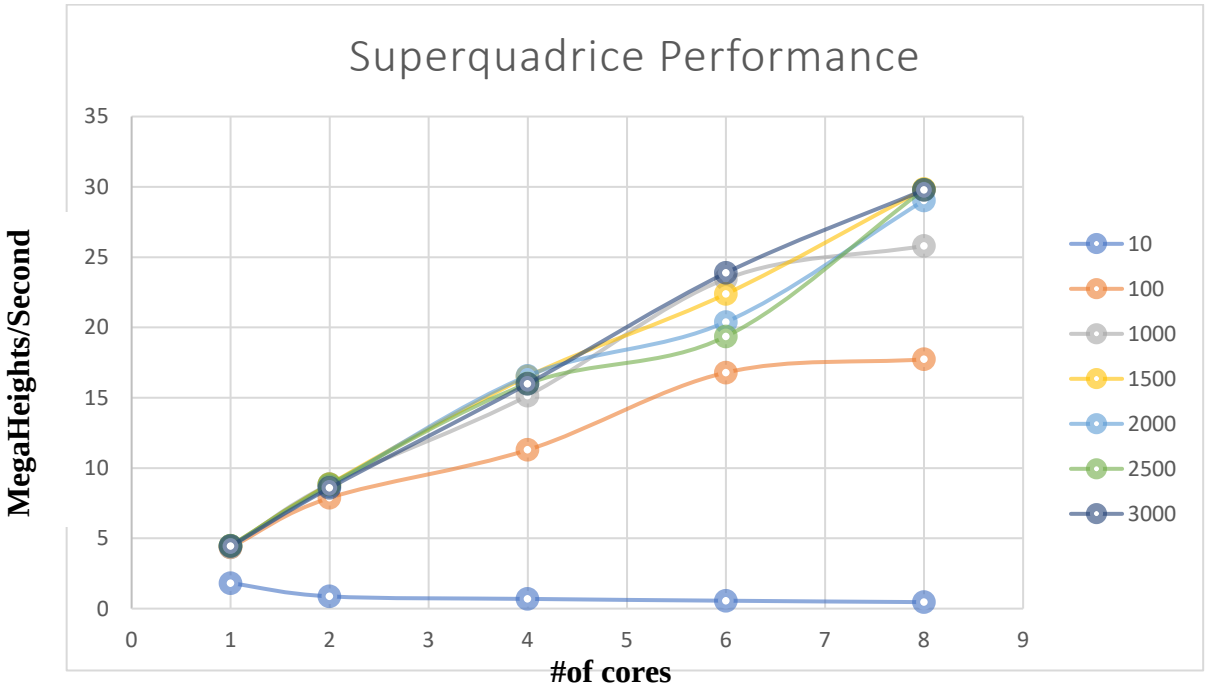

Figure 2:Graph of performance vs. number of threads

4) What patterns are you seeing in the speeds?

Why do you think it is behaving this way?

Based on figure 1, I discovered that the performance is kept improving if the number of threads goes up from 1 to 8. On the other hand, the performances of different threads numbers are almost the same when the number of nodes is over 1000 depending on the result of figure 2. Both patterns are linear and going to be stable. I think the reason is that the calculating number goes on to a certain level, they will keep the threads busy with a certain percentage. Thus, the performance will be stable relatively.

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

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

At 3000 nodes, 1-thread-to- 2-thread, S = 1.93, **Fp = (2/1)(1-(1/S)) = 0.96**
1-thread-to- 4-thread, S = 3.6, **Fp = (4/3)(1-(1/S)) = 0.96**
1-thread-to- 6-thread, S = 5.37, **Fp = (6/5)(1-(1/S)) = 0.97**
1-thread-to- 8-thread, S = 6.7, **Fp = (8/7)(1-(1/S)) = 0.97**

The maximum speed-up is 6.7.