Intro to Parallel Programming

Amine Gaizi
gaizia@oregonstate.edu
Student ID: 934-044-900

Project n°2 – Numeric Integration with OpenMP Reduction

What machine was this run on ?

This project was ran on the university's flip servers. The load average at the moment of the execution was: 1,01, 1,63, 1,62 which is reasonably low.

What is the actual volume ?

According to my program, the actual volume is 6.481468 unit^3. Since we don't have a specific unit given to us (inches, cm, etc..), the volume is here reported using a generic unit to the power of 3.

Performance Table:

Threads\Nodes	4	8	16	32	64	128	256	512	1024
1	2,355	2,1937	2,0768	2,0047	2,1495	3,7817	3,762	3,7426	3,7418
2	3,2179	3,8288	4,0168	4,0039	3,9488	7,4659	7,519	7,479	7,4265
4	4,8396	6,2967	7,3673	7,6857	7,7183	14,2287	14,6304	14,4933	14,5499
6	5,444	12,08	18,2543	20,3138	11,5369	11,5142	21,0942	21,0019	21,0003
8	4,6282	12,3671	21,9724	26,5326	15,2996	15,411	28,0505	27,9575	27,7508

The performance is measured in Mega Heights Computed Per Second. For this particular table of performance, there were 256 tries to measure the max performance.

Performance Graphs:

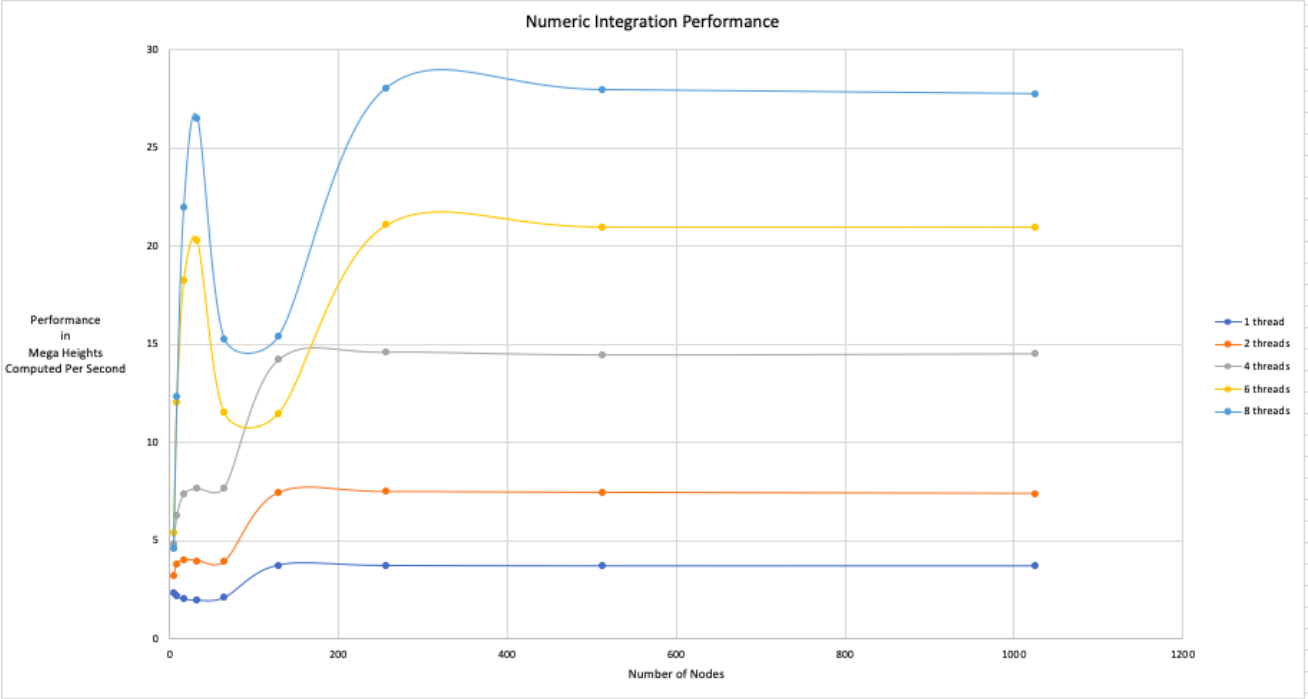

Figure 1: Graph of the Performance depending on the number of nodes

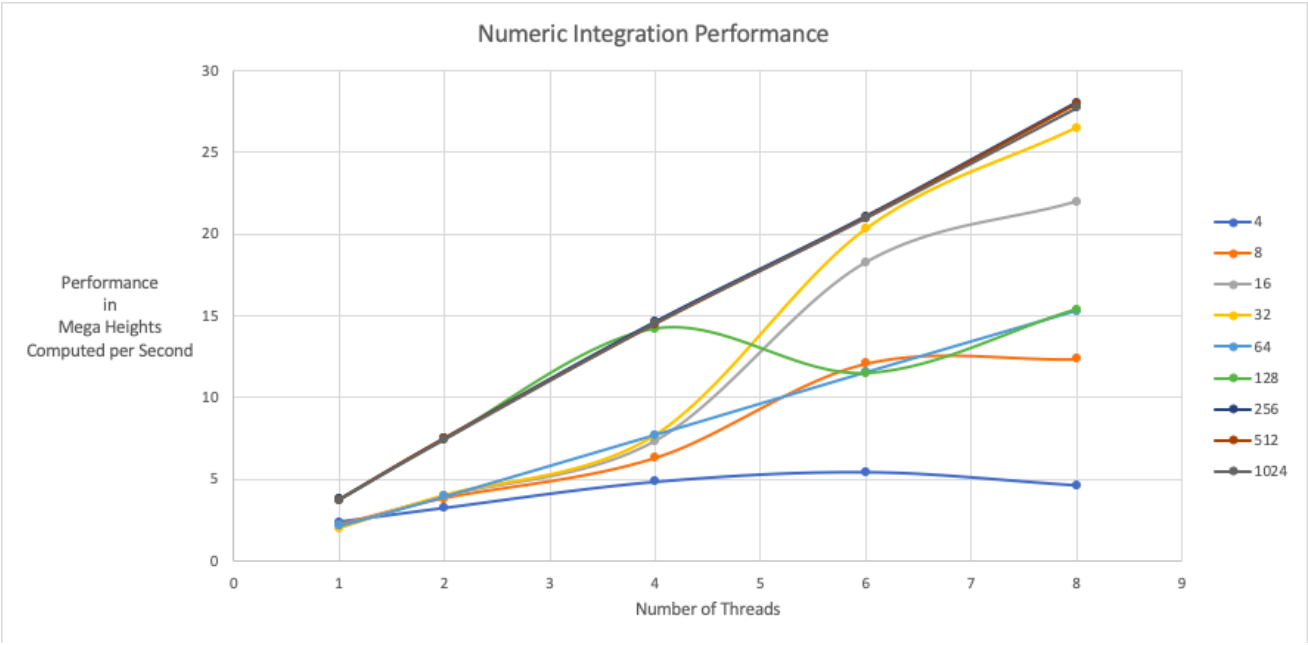

Figure 2: Graph of the Performance depending on the number of threads

Observations and interpretation:

In terms of speed, there is a clear pattern in figure 1. At first, the speed increases for 4 to 32 nodes. As explained in class, the bigger the dataset gets, the better the performance of our program is because of the overhead of computing resources needed when doing parallel programming. The bigger the dataset gets, the better the more value we get out of the computing resources we need and therefore we get a better performance.

However, between 32 and 256 nodes, there is a major decrease in performance for every case tested (1, 2, 4, 6 and 8 threads). The drop in performance is major when using 6 and 8 threads, when using a lower number of threads, the performance seems to be stabilized rather than increasing. This might be due to a specificity of the numeric integration which causes the performance to drop (or stop increasing) if the dataset is not big enough or small enough.

Then, the performance increases again stays stable after reaching 512 Nodes. The maximum performance seems to be achieved between 200 and 300 nodes for 6 and 8 threads, and between 100 and 200 nodes for 1, 2 and 4 threads.

What is the Parallel Fraction for this application ?

Since the best performance is achieved using 256 nodes, let's calculate the speedup using the performance for 8 threads and 1 threads at 256 nodes.

The results from the simulations give us the 1-thread-to-8 Speedup result:

$$S = \left(\frac{Performance\ 8\ threads}{Performance\ 1\ thread} *\right) = \frac{28.0505}{3.762} = 7.456$$ *for 256 tries.

Therefore, following the formula seen in class:

$$F_{parallel} = \frac{n}{n-1} * \left(1 - \frac{1}{Speedup}\right) = \frac{8}{8-1} * \left(1 - \frac{1}{7,456}\right) = 0,98957$$

Given this parallel fraction value, and using Amdahl's law, we can find the maximum speed-up we could ever get:

$$Speedup_8 = \frac{1}{\frac{Fp}{n} + Fsequential} = \frac{1}{\frac{Fp}{n} + (1 - Fp)} = \frac{1}{\frac{0,98957}{8} - (1 - 0,98957)} = 7,456$$

The value of the best speed up reachable is very close to the value of the actual speedup we are reaching. This means that we are reaching the best parallelism we could get in these circumstances.