Project #2 Numeric Integration with OpenMP Reduction

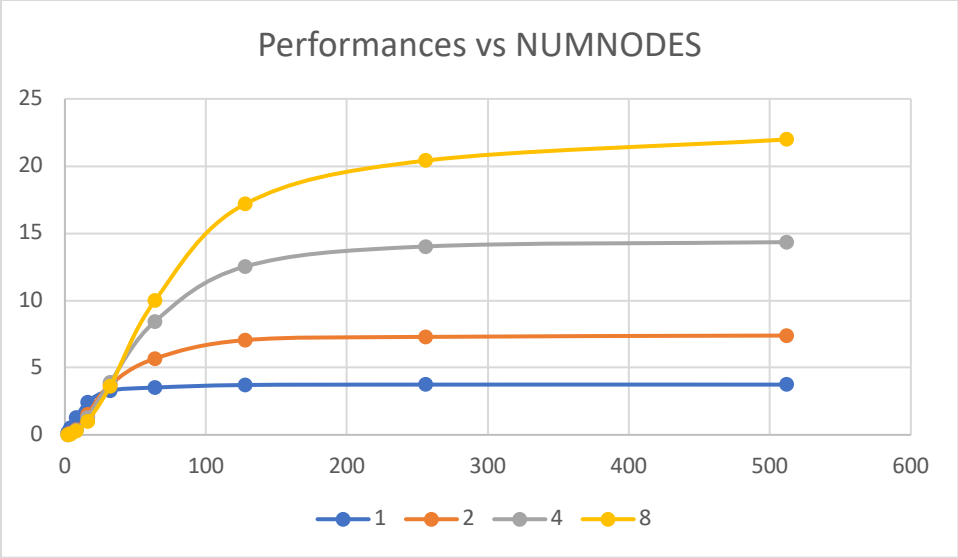
Qingxiao Yuan

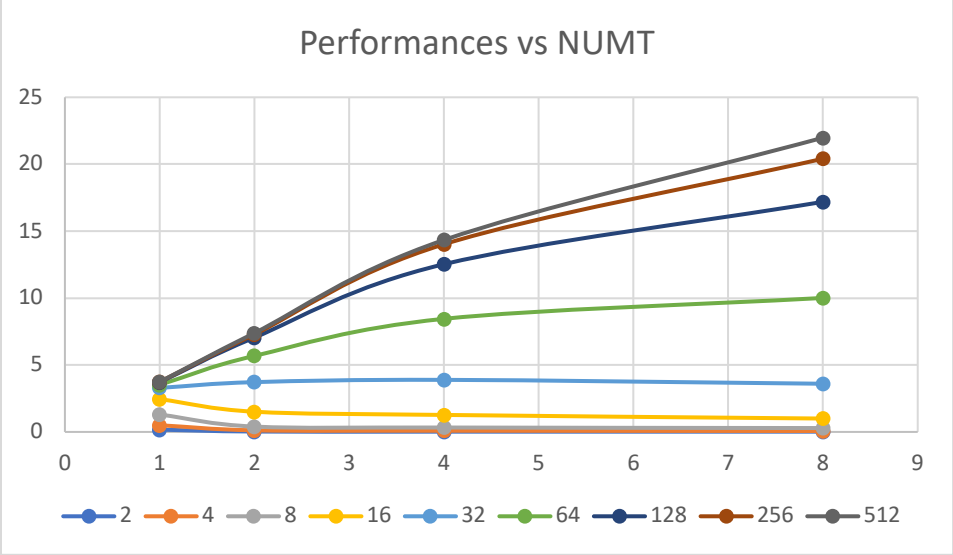
yuanqi@oregonstate.edu

1. Tell what machine you ran this on: flip1of Oregon State University.
2. What do you think the actual volume is? I think it's around 6.48.
3. Show the performances you achieved in tables and graphs as a function of NUMNODES and NUMT.

NUMT	2	Vol	4	Vol	8	Vol	16	Vol	32	Vol	64	Vol	128	Vol	256	Vol	512	Vol
1	0.17	0	0.5	3.53	1.3	5.54	2.45	6.19	3.29	6.39	3.52	6.45	3.71	6.47	3.74	6.48	3.74	6.48
2	0.03	0	0.12	3.53	0.4	5.54	1.51	6.19	3.72	6.39	5.68	6.45	7.05	6.47	7.29	6.48	7.39	6.48
4	0.02	0	0.09	3.53	0.33	5.54	1.27	6.19	3.89	6.39	8.45	6.45	12.53	6.47	14.02	6.48	14.34	6.48
8	0.02	0	0.07	3.53	0.29	5.54	1	6.19	3.6	6.39	10.01	6.45	17.19	6.47	20.41	6.48	21.98	6.48

	2	4	8	16	32	64	128	256	512
1	0.17	0.5	1.3	2.45	3.29	3.52	3.71	3.74	3.74
2	0.03	0.12	0.4	1.51	3.72	5.68	7.05	7.29	7.39
4	0.02	0.09	0.33	1.27	3.89	8.45	12.53	14.02	14.34
8	0.02	0.07	0.29	1	3.6	10.01	17.19	20.41	21.98





4. What patterns are you seeing in the speeds?

Condition: I set NUMT={1,2,4,8} and NUMNODES= {2, 4, 8, 16, 32, 64, 128, 256, 512} to collect data.

In the Performances vs NUMNODES graph, it looks like a Logarithmic function, which means that the performance could increase first then reach a certain limitation as the number of

subdivisions increase. Furthermore, we could see that the performance would increase as the number of thread increase. But we don't know whether there's an limitation about that.

In the performances vs NUMT graph, we could see more trends here. First, the performance keeps going as the threads grow. Second, unlikely the limitation of Performances vs NUMNODES graph, it doesn't show the limitation. Since all the curves have similar trends, we take the most obvious data- in 8 threads as an example. Here, we could see that the performance is still keeping growing.

Overall, we could see that, under my present condition, the performance could increase as the threads/subdivisions increase. However, it also has a difference. Though there's an limitation growth in subdivisions increase, the performance will keep growing as the number of threads grow.

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

First, the curve shows increase trend. The reason is that I use the parallel programming dividing the task into several parts by logarithmic operations and the performance in my code is the task finish time-parallel start time. Also, the more tasks are divided into parts, the faster the tasks are executed.

Second, the threads seem like to keep growing and subdivisions are not. I guess it's because the whole object has a certain volume and each thread could handle large enough values when the subdivisions are big enough. In other words, the subdivisions can't benefit from multithreading at that time. Therefore, it will hardly have improvement in the performance.

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

   The calculation based on the performance of 1 thread and 8 threads, under 512 subdivisions.

   $F_p = \frac{n}{(n-1)}\left(1-\frac{1}{speedup}\right) = \frac{8}{8-1}\left(1-\frac{3.74}{21.98}\right) = 94.84\%$

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

   Max Speedup= $\frac{1}{1-F_p}$ = 19x.