CS475

Project #0

Hsiang Lo

Project 0: Simple OpenMP Experiment

Objective: Become familiar with OpenMP and making sure my system is operable with OpenMP as well as seeing how OpenMP works. In the given example, we will be testing the differences between 1 threads versus 4 threads.

Note:

- This way of timing measures **wall-clock time**, which is what we want to know in parallel time, not CPU time.
- Use **uptime** to make sure CPU Load average is low.
- S (SpeedUp) = (Execution time with one thread) / (Execution time with four thread)
- I completed this assignment using ARRAYSIZE of 100 and NUMTRIES of 24
- It was noted that SpeedUp can either be T1/T4 or P4/P1 where T is time of execution and p is performance.

Data

Thread	Execution Effort/time(s)	Thread	Execution Effort/Time(s)	SpeedUp
1	154.08	4	54.47	2.828713053
1	153.86	4	55.83	2.755866022
1	160.02	4	52.55	3.045099905
1	154.58	4	54.94	2.813614853
1	160.26	4	53.62	2.988810145
1	142.45	4	53.82	2.646785582
1	151.74	4	55.37	2.74047318
1	158.22	4	55.84	2.833452722
1	159.99	4	52.77	3.031836271
1	151.74	4	53.83	2.818874234
Average	154.694		54.304	
Peak Values	160.02		55.84	2.865687679

Questions

1. Tell what machine you ran this on?

   I ran project 0 on my macbook pro that was built and purchased in January 2018.

2. What performance results did you get?

   The peak performance result I received for T1 is 160.26, and the peak performance I received for T4 is 55.84. These peak performance were on part with the average computation as well. When compared to average, the peak for T1 only differs by 6 points from average and for T4, the peak performance only differs by 1.4 points from average.

3. What was your 4-thread-to-one-thread speedup?

   Using the equation of SpeedUp = (Execution time with one thread) / (Execution time with four thread), I took a sample of 10 peak performances for both of NUMT equals to 1 and NUMT equals to 4, I find that when using the peak performance to compute my speedup, I received 2.86.

4. If the 4-thread-to-one-thread speedup is less than 4.0, why do you think it is this way?

   While ideally, a perfect 4 is optimal. However, due to server traffic on FLIP, I am receiving a much lower value. At 2.6, I checked with uptime, it wasn't significant

especially in the 1-minute frame, but there were still 70 something active users. I believe this could be the main cause of the differentiation.

5. What was your Parallel Fraction, Fp?

   Given that the equation is float Fp = (4./3.)(1. - (1./S)) and my SpeedUp is 2.6, Fp will be (1.333) * (0.616), which is then equal to 0.821128, rounded to two decimal place, it will be 0.82*

Conclusion From this project, I learned that splitting a process into more thread indicates that the processing power will be divided and splits between the threads. When splitting a process from one thread to four thread, we would expect to see equal division of power, but instead we see that it was below that of equal division.