Zachary K. Lewis

Dr. Vivek Sarin

CSCE 435

7th Febuary 2022

HW1-435

1. (20 points) Plot execution time versus p to demonstrate how time varies with the number of threads. Use a logarithmic scale for the x-axis.
2. (20 points) Plot speedup versus p to demonstrate the change in speedup with p.
3. (10 points) Using the definition: efficiency = speedup/p, plot efficiency versus p to demonstrate how efficiency changes as the number of threads are increased.
4. (10 points) What value of p minimizes the parallel runtime?

The number of threads p that minimizes the parallel runtime is 256 threads, which maxes out the speedup value at ~35.61 speedup. After 256 threads, the speedup decreases and the time allotted increases.

1. (10 points) Repeat the experiments with n=109 To obtain the execution time for p=2k, for k = 0, 1, …, 13. In this case, what value of p minimizes the parallel runtime?

The value of p that minimizes the parallel runtime is 512 threads. This number of threads gives the maximum speedup value of ~45.84.

1. (10 points) Why does the runtime start to increase as p is increased beyond a certain value?

Even though the thread splitting can speed up calculations to a certain extent, the overhead required to manage these threads results in additional wasted time. As you increase the treads, the speed of the parallelizable portion of the code decreases while the overhead time increases. The overhead is not a problem until numerous threads are being used, which is why after 2256 and 2512 threads are created, the overhead’s time bloat outweighs the speedup from parallelization.

1. (10 points) Why is there a difference in the number of threads needed to obtain the minimum execution time for two values of n?

The reason why the number of threads needed to obtain the minimum execution time is different between the two values of n is again related to the speedup versus overhead. Comparing n = 108 versus n = 109, the initial time needed for the two n’s in sequence and the amount of time reduced from parallelization for small values of p is greater for the larger n than the smaller n. When comparing the efficiency of the two this can also be inferred, as the efficiency of the larger n is higher than the efficacy of the smaller n.

The explanation behind this is that the speedup from parallelization grows with larger values of n while the overhead stays relatively constant in comparison. This results in the overhead being more negligible and explains why it takes more threads to reach the minimized value.

1. (10 points) Plot error versus n to illustrate accuracy of the algorithm as a function of n. You may have to run experiments with different values of n; for example n could be chosen to be 10k, for k = 3, …, 9. Use p = 48.