Assignment5: Parallel Sort

Tianyu Wei 001371254

1. Initially, the arraysize is set as 120000 with cutoff divided by 2 in each loop. Then the arraysize was multiplied by 2 to 240000, 480000, 960000 to do the test. There we can generate a diagram shows down below.

Cutoff=32%\*Arraysize

According to the diagram above, each group is intended to decrease at first, reach the lowest point then increase a bit and have a minimum cutoff value around 0.3125 of the arraysize and it doesn’t have more point lower that it. Based on that, we are able to say that 32% of arraysize is the target point, since cutoff has the most efficacy around it.

1. In order to choose thread proper amount of thread, I use the way to increase the number of threads to sort the array parallelly stick with power of 2 from 2 to 2048. Compare the time it consumed on sorting where array size is 960000 and cutoff is 300000. In the diagram below, we can easily find that time doesn’t change significantly after 32 threads. It comes up with the conclusion that time fluctuated significantly if we use threads less than 32. After threads of 32, it tends to keep stable. Then 32 is the most efficacy amounts of threads in this case.
2. We set the arraysize as 960000 to make another set of tests to verify the conclusion. If we take a look at amounts of threads, cutoff, and the minimum time it takes for each group. They all show that when we choose to use 32 threads and 32% of arraysize at the same time as cutoff has the most efficacy. The column time in the diagram down below represents the minimum time it takes when using specific amounts of threads. Column ‘Cutoff’ represents the cutoff when it has the minimum time consumed.

As a result, the combination of 32 threads and 32% of arraysize as cutoff performs the most efficacy in our case.