Krishna Sai Chemudupati

Lab 09

The cutoff values used are such that the threads used are at a maximum of 64. This is because Visual Studio sets this as maximum. If the cutoff value creates higher number of threads, only 64 chunks are sorted using the sequential insertion sort.

I could fix it by using insertion sort on the rest of the elements (considering it as one chunk) of the array after the 64th chunk. But that is just a hack for it to work on Visual studio.

This can also be avoided using clang++ or other compilers without such restrictions. The input array size is 1 million.

|  |  |
| --- | --- |
| Cutoff values | Runtime |
| 15,625 | 4.221 seconds |
| 15,626 | 4.265 seconds |
| 16,626 | 4.484 seconds |
| 17,626 | 4.721 seconds |
| 30,626 | 7.965 seconds |
| 50,626 | 13.042 seconds |
| 60,626 | 15.374 seconds |
| 80,000 | 20.267 seconds |

As the cutoff increases, the runtime increases because the parallelization decreases and sequential sort usage increases. The cutoff value 15,625, which uses all the 64 threads to parallelize the sorting results in lower runtimes. If more threads could be used, it could result in lower runtimes. But, as more chunks get sorted in parallel threads, more work is done by the merge function and therefore very low cutoffs can also result in higher runtimes.

With the available threads, I found that using maximum amount of parallelization, the cutoff of around 15, 625 would result in lowest runtime.