COL730 Parallel Programming

Assignment-4

Divyans Yadav 2020EE30593

November 28, 2023

Algorithm ParallelSort using CUDA & OMP

1 Methodology

- My algorithm is overall same as in Assignment 1. But the sorting of the partitions is done on GPU.
- For sorting on GPU, I used Radixsort because of its sorting not based on comparison. Thus, I found this algorithm as best for the Cuda threads which perform instructions in lock step.
- I created Histogram on shared memory for storing the count of the elements position on the sorted output array.
- I achieved more parallelism with help of parallel prefix sum on histogram for counting the elements position.

2 Remarks

- As seen in the benchmark testing of my algorithm, it is quite slow compared to CPU, meaning I have been not able to utilize the GPU efficiently.
- I used at most 32 threads per block as going more than that was not giving correct results, I do not why that was happening.
- I could have tried achieving parallelism at block level also by sorting the small count of elements at thread level and then again sorting on block level.

3 Results

3.1 Time vs Array Size

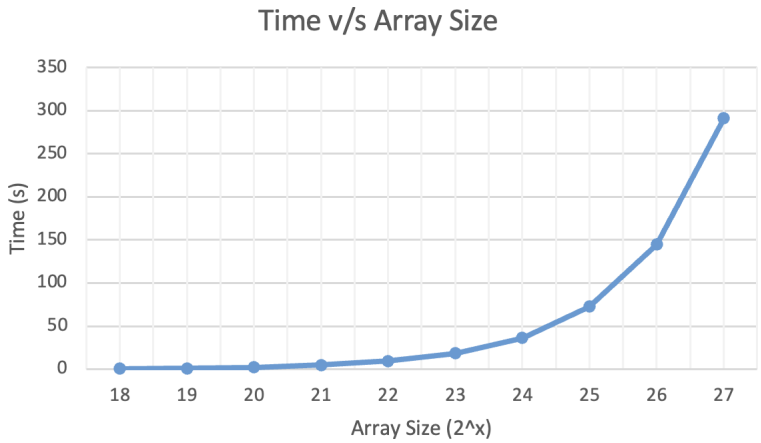

Figure 1: CPU Count fixed at 14 and GPU count at 1

The timing for array size below 2^{18} have less time taken by using conventional sequential sorting rather than our parallel. So, I did not included their data in the graph.