CSC 6220: Parallel Computing I: Programming Project
Fall 2023

Student name: Kanchan Chopde

The standard Odd Even sorting algorithm sorts the list of numbers using an Odd-Even sorting algorithm. During each phase using p processes, the odd or even numbered processes perform compare and split operations, where they exchange data with neighbors. Now both the process and its neighbour have an array of numbers that they already have in addition to what they receive in exchange with their neighbors. Now the process combines the already existing list of numbers with received numbers, making a sorted list available at both processes. It then keeps the small part from the array to itself and discards the larger part. Its neighbor process does the same but keeps the larger part of the list of numbers and discards the smaller part.

A modified Odd-Even Sort algorithm is used to perform odd-even transposition iterations as long as the sub-arrays are changing. For this I have added the logic to check if our local elements were the same as that after compare split operation. The flag value is set to (1) if locally the array is same as previous phase and (0)locally the array has changed.

MPI reduce function is used which combines values from all processes either 0 or 1 performs Multiplication operation and distributes the result back to all processes. Next we Broadcast allTrue from rank 0 to all other ranks. We use Barrier to make sure all the processes are synchronized before Gathering the data. Finally we gather sorted subarrays to process 0 which is used write in result.txt file.

Below are the zoomed graphs for input sizes: $2^{16} = 65536, 2^{20} = 1048516, 2^{24} = 16777216$

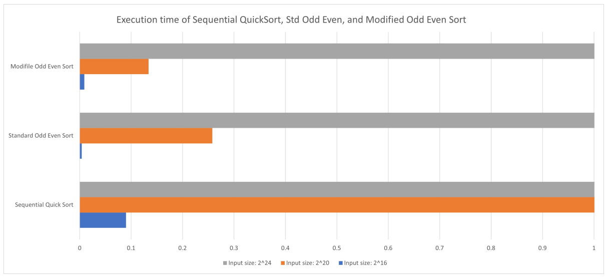

Figure 1: Execution time of Sequential QuickSort , Standard Odd Even Sort and Modified Odd Even Sort(Focus for $2^{16} elements$)

The blue colored bars in Figure 1 represent the execution time: Sequential Quick Sort: 0.09 seconds Standard Odd-Even Sort: 0.00380833 seconds Modified Odd-Even Sort: 0.009107 seconds It can be seen in this case clearly best performance is by our Modified algorithm and sequential Quick sorts performs worst.

The Orange colored bars in Figure 2 represent the execution time: Sequential Quick Sort: 20.19 seconds Standard Odd-Even Sort: 0.25785733 seconds Modified Odd-Even Sort: 0.1338643 seconds It can be seen in this case clearly best performance 0.133 seconds is by our Modified algorithm and sequential Quick sorts performs worst with 20.19 seconds execution time

The grey colored bars in Figure 3 represent the execution time: Sequential Quick Sort: 5130.90 seconds Standard Odd-Even Sort: 1.0939343 seconds Modified Odd-Even Sort: 2.316653 seconds It can be seen that we get the best performance by standard Odd-Even sort and our Modified algorithm just taking 1.09 and 2.31 sec respectively and sequential Quick sorts which takes about 5139.90 seconds. These are approximate values and the difference can be seen by

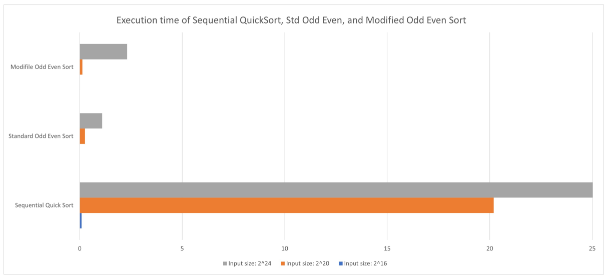

Figure 2: Execution time of Sequential QuickSort , Standard Odd Even Sort and Modified Odd Even Sort(Focus for $2^{20} elements$)

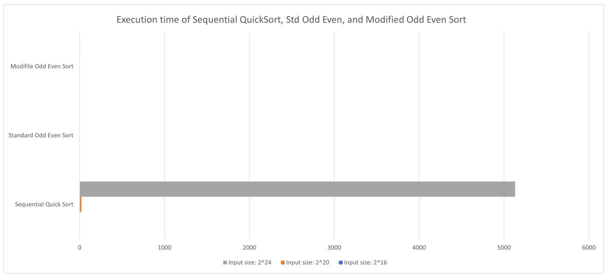

Figure 3: Execution time of Sequential QuickSort , Standard Odd Even Sort and Modified Odd Even Sort(Focus for $2^{24} elements$)

taking more larger inputs.

Below is the plot for Execution time for Speed up by Modified Odd-Even Sort vs Sequential QuickSort.

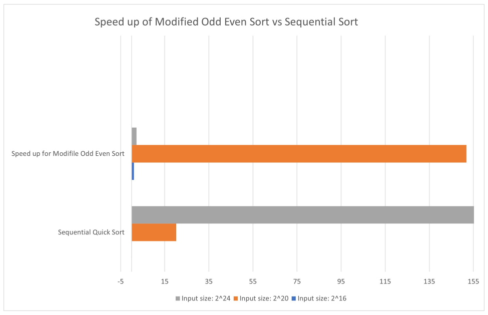

Figure 4: Speedup of Modified Odd-Even Sort vs Sequential QuickSort

Speed up = $T_{Sequential}/T_{Parallel_{M}odifiedOddEvenAlgo}$
For input size $2_{16} = 0.090/0.09107 = 0.988$ seconds ,Sequential Quick Sort: 0.09 seconds,
For input size $2_{20} = 20.19/0.133 = 151.80$ seconds, Sequential Quick Sort: 20.19 seconds,
For input size $2_{24} = 5130.90/2.316 = 2215$ seconds,Sequential Quick Sort: 5130.90 seconds
The chart shows that the modified odd-even sort algorithm is significantly faster than the sequential quick sort algorithm, especially for large input sizes. For example, for an input size of 2_{24} elements, the modified odd-even sort algorithm is about 155 times faster than the sequential quick sort algorithm. The modified odd-even sort algorithm is a parallel sorting algorithm, which means that it can be executed on multiple processors simultaneously. This is why it is significantly faster than the sequential quick sort algorithm, which is a serial sorting algorithm. The speed-up of the modified odd-even sort algorithm is not perfectly linear. This is because the parallel efficiency of the algorithm decreases with the problem size.