

**NAME:** Rayed Muhammad Saeed

**ROLL NO:** 20i-1822

**DEGREE:**  BSCS

**SECTION:** F

**SUBJECT:** Parallel and Distributed Computing

**SUBMITTED TO:** Sir Aleem

**SUBJECT CODE:** CS-3006

**ASSIGNMENT NO:** 2

**DATE:** 4th March 2023

**Report For Assignment 2**

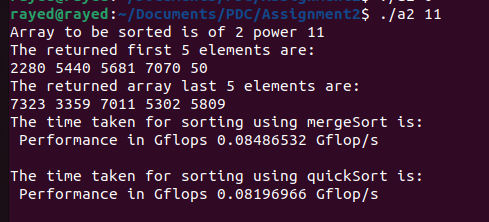
**Task 1:**

In task 1 we had to make 4 arrays of different sizes (2^11, 2^13, 2^15 and 2^17) respectively. Then we had to store the arrays in a file. According to given command line input we had to read the file of the array given in input on the command line, and then apply two different sorts on that array, the merge sort and the quick sort. After which we were to analyze the two.

**Performance analysis:**

First, we did the analysis for array size **2^11**.

The analysis between the merge and quick sort was done by calculating the performance of both the sorts in Gflops/s. The results are below:



The above screenshot shows the performance in Gflops/s for both the sorting algorithms MergeSort and QuickSort.

Analyzing the two performances we can see that for a smaller size array the merge sort performs better than the quick sort. But they are relatively close to each other in performance.

After which we did the analysis for array size **2^13**.

The analysis between the merge and quick sort was done by calculating the performance of both the sorts in Gflops/s. The results are below:

Text

Description automatically generated

The above screenshot shows the performance in Gflops/s for both the sorting algorithms MergeSort and QuickSort.

Analyzing the two performances we can see that for a array size 2^13 the merge sort performs way better than the quick sort.

Next, we did the analysis for array size **2^15**.

The analysis between the merge and quick sort was done by calculating the performance of both the sorts in Gflops/s. The results are below:

Text

Description automatically generated

The above screenshot shows the performance in Gflops/s for both the sorting algorithms MergeSort and QuickSort.

Here there is a change in the performances, as we move towards a larger array size we start to see that quick sort algorithm performs better than the merge sort algorithm.

Next, we did the analysis for array size **2^17**.

The analysis between the merge and quick sort was done by calculating the performance of both the sorts in Gflops/s. The results are below:

Text

Description automatically generated

The above screenshot shows the performance in Gflops/s for both the sorting algorithms MergeSort and QuickSort.

The above analysis shows how for the array size of 2^17 the quick sort has started to perform much better than the merge sort algorithm. As the size is increased the performance difference between the two sorts will increase and quick sort will perform much better.

Hence, from observing all the 4 array sizes and their respective performance in Gflops/s we made a graph to help us understand the different sorting algorithm performances.

In the above graph the y axis shows the Gflops/s (performance). The x-axis shows the array sizes. The blue color bars are for merge sort and the red colored bars are for quick sort algorithm.

Inferring from the graph above we can see that as the array size increases the performance of merge sort starts to decrease but the performance of quick sort increases. As it is that quick sort is better for larger array sizes, which can be proven from the graph above.

**Task 2:**

this was done using 4 processes with different leaf sizes and different array sizes using single and multi machine.

