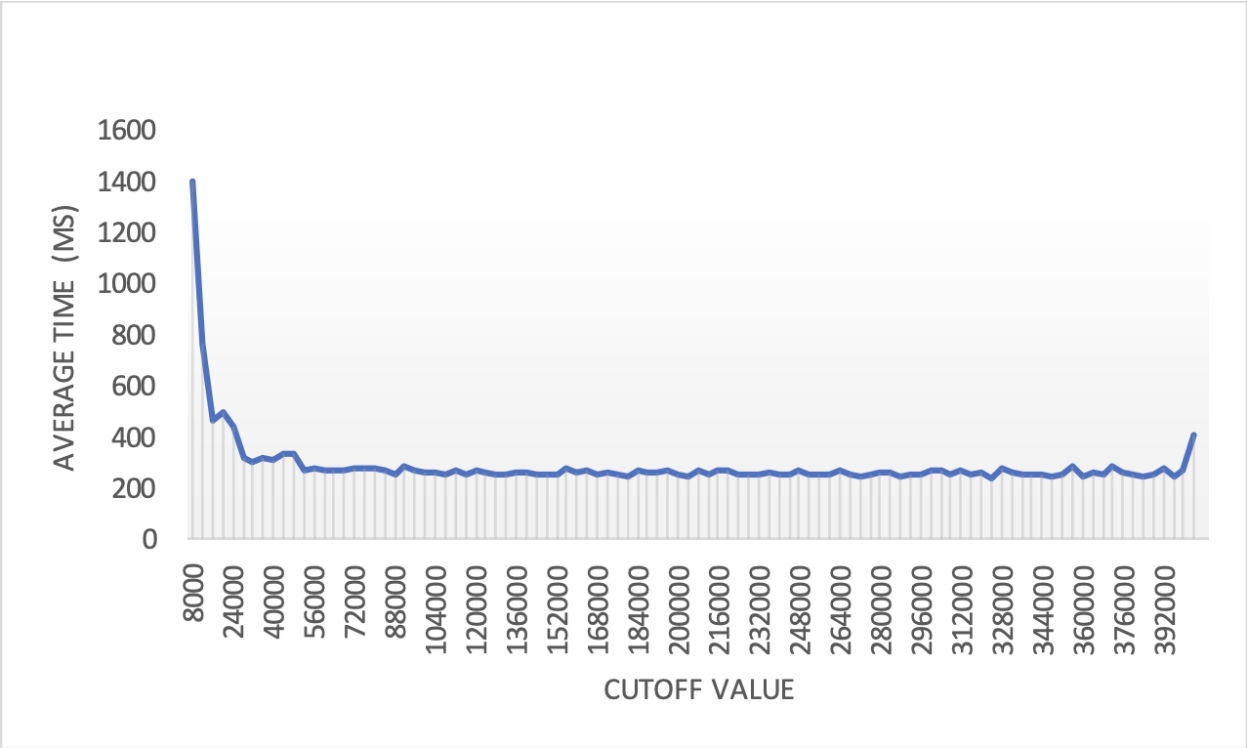
INFO 6205
Program Structures & Algorithms
Spring 2022
Assignment Number : 4
Sravya Ganda (002103774)

Task: The task of the assignment is to implement a parallel sorting algorithm so that each partition of the array is sorted in parallel. Expectation is to find the relationship between the thread count and cutoff value. The system sort should be performed for the optimum performance. The report is divided into three parts based on the size of the array to be sorted.
The array sizes that are taken into consideration are 4,00,000, 2,000,000 and 4,000,000. In each part, the best cutoff value without restricting the thread count is identified first. Then, the value of cutoff is fixed in order to find out the ideal number of threads that are to be assigned. Since the number of threads assigned should be controlled, I used 2 and 4 thread. In order to restrict the thread count (2 or 4) the sorting algorithm would use system's sorting algorithm rather than the merge sort with threads assigned.

Output

The experiment is performed for array with 3 different sizes:

1. Array size : 4,00,000

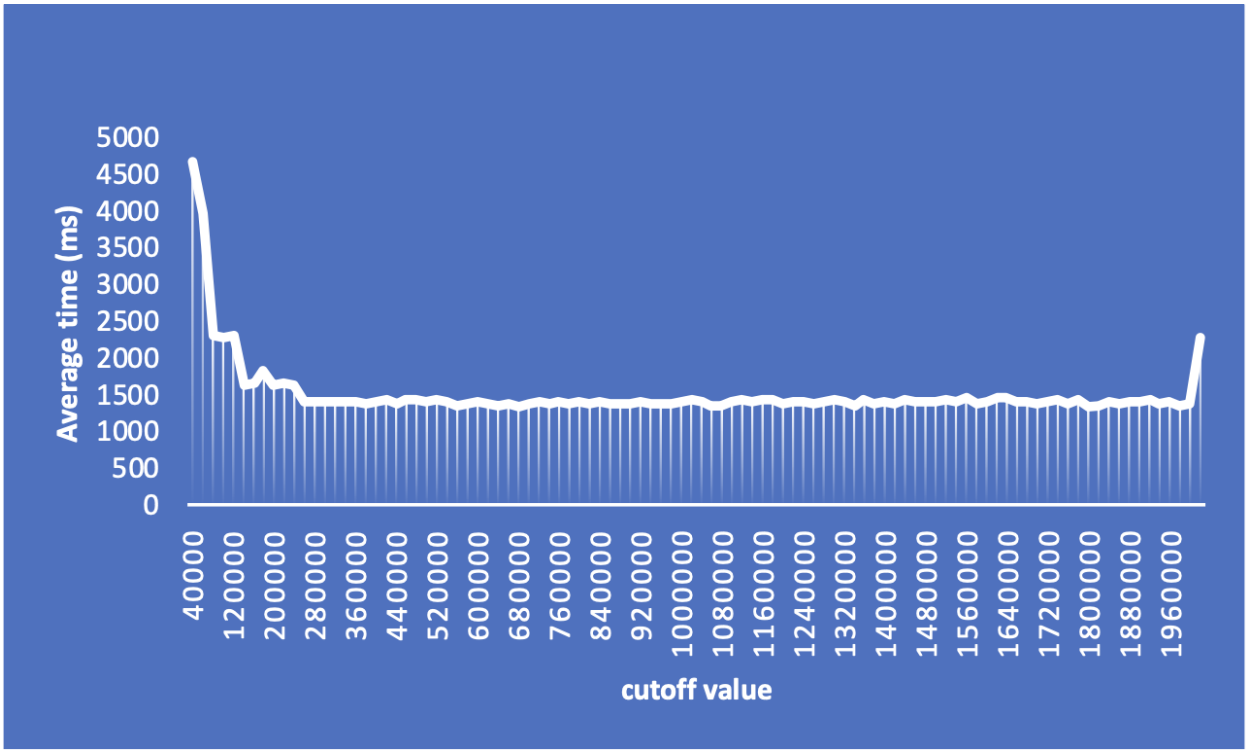


The cutoff value for the array of size 4,00,000 items was found to be 324000 where the average time observed to sort the array is minimum at 148ms. The cutoff value was fixed at 324000 and the solution was run with varied number of threads and observations listed below were made.

Array Size : 4,00,000		Cutoff: 324000	
Thread	2	4	
ms	148	154	

2. Array size : 2,000,000

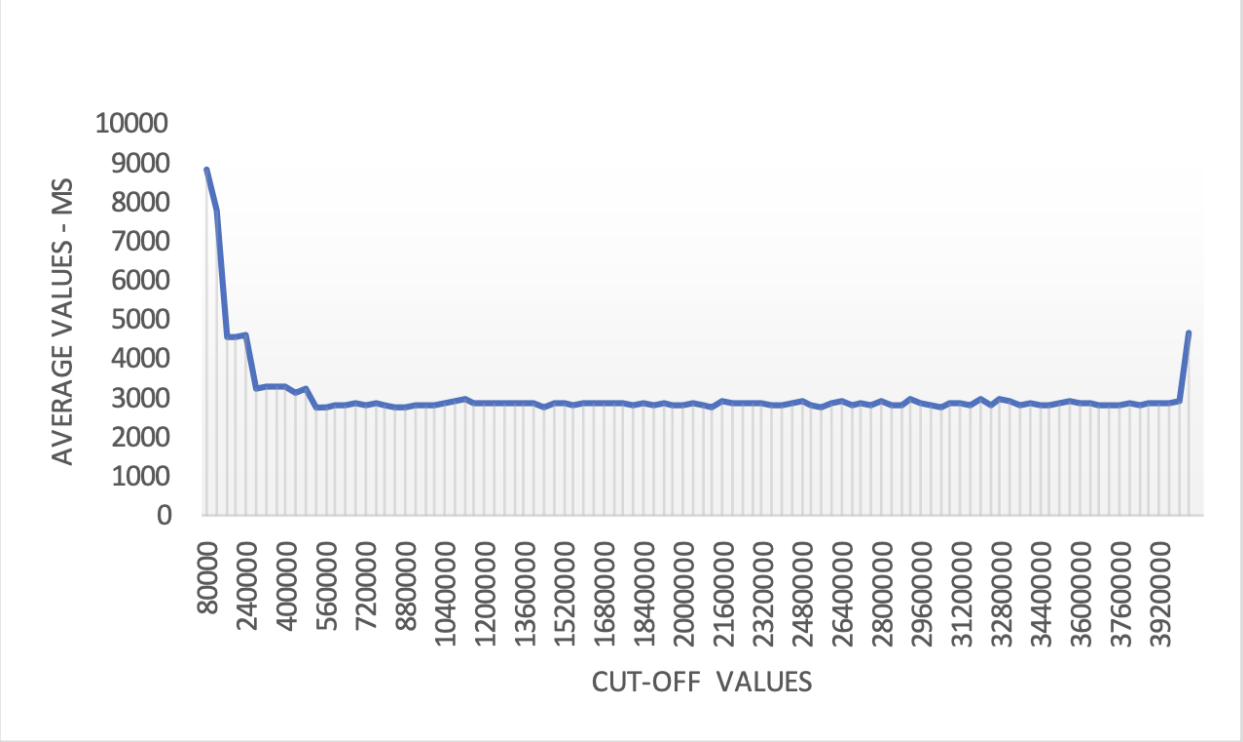
The cutoff value for the array of size 2,000,000 items was found to be 680000 where the average time observed to sort the array is minimum at 600ms. The cutoff value was fixed at 680000 and solution is run with varied number of threads and observations listed below were made.



ArraySize : 2,000,000		Cutoff : 680000	
Thread	2	4	
ms	600	605	

1. Array Size : 4,000,000

The cutoff value for the array of size 4,000,000 items was found to be 520000 where the average time observed to sort the array is minimum at 1280ms. The cutoff value was fixed at 520000 and solution is run with varied number of threads and observations listed below were made.



ArraySize : 2,000,000		Cutoff : 680000	
Thread	2	4	
ms	1280	1285	

Relationship conclusion:

From the above observations, we can see that, for each array set, as the size of the array increases, the threads assigned increases, the time consumed decreases.

When the method CompleteableFuture is called, the thread allocation increases which decreases the time consumed to perform the sort. However, the lesser the size of the array, the system sort gives better results than the merge sort that is being run asynchronously. Thus, it would be more efficient to run the default sorting algorithm than using the merge sort asynchronously when the size of the array is less.

To conclude, parallel sort is more efficient and faster when the depth of recursion is log of number of threads available for sorting.

Attaching the screenshot of the output: