**Program Structures and Algorithms**

**Spring 2023(SEC –01)**

**Assignment-5**

Name:- Swarag Sanjay Gutte

NUID :- 002728422

Problem

Please see the presentation on *Assignment on Parallel Sorting* under the *Exams. etc.* module.

Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (*t*) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of *lg t* is reached).
3. An appropriate combination of these.

Evidence and conclusion:-

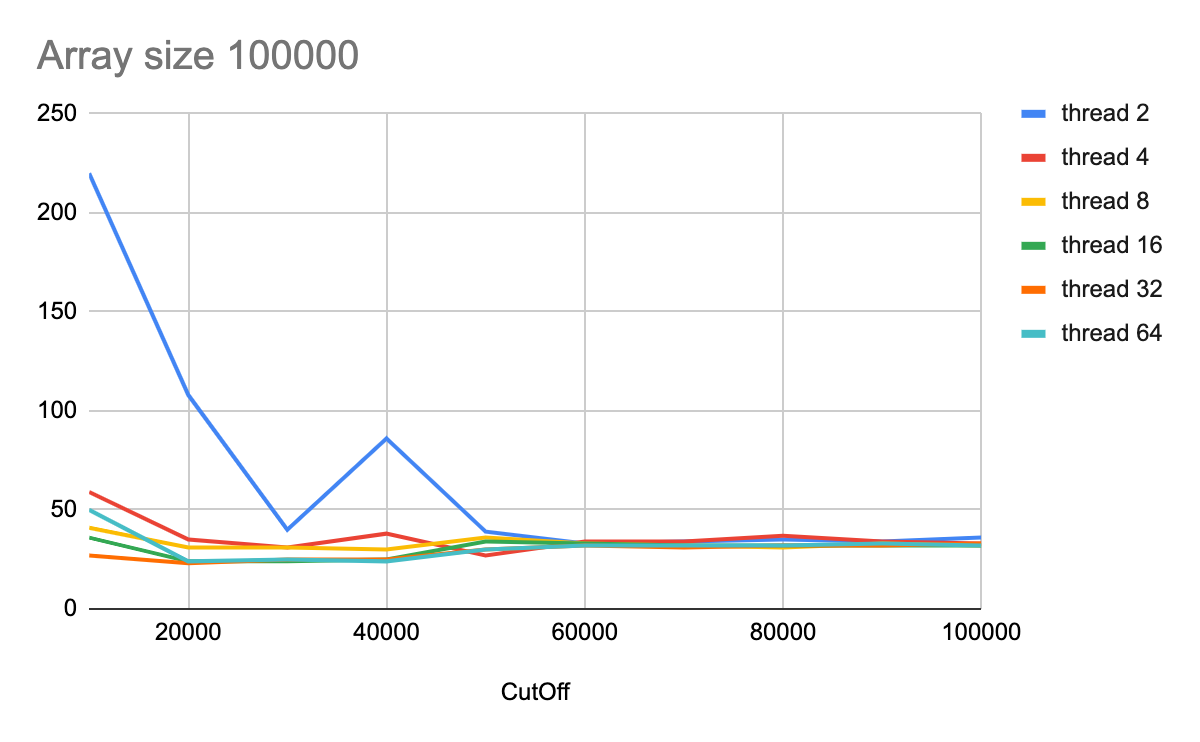
Conclusion

1. Based on the three graphs plotted using different values and their array sizes, it can be concluded that the cut-off ratio, can provide insights into their relationship.

2. When there are significant differences between the parallel sorting parameters and array size, the ratio tends towards a constant value over time. This constant behavior is observed in the plots for the largest array size. Therefore, it is recommended to choose a sufficiently large array size in comparison to the parallel sorting values to obtain an appropriate and comparable cut-off value.

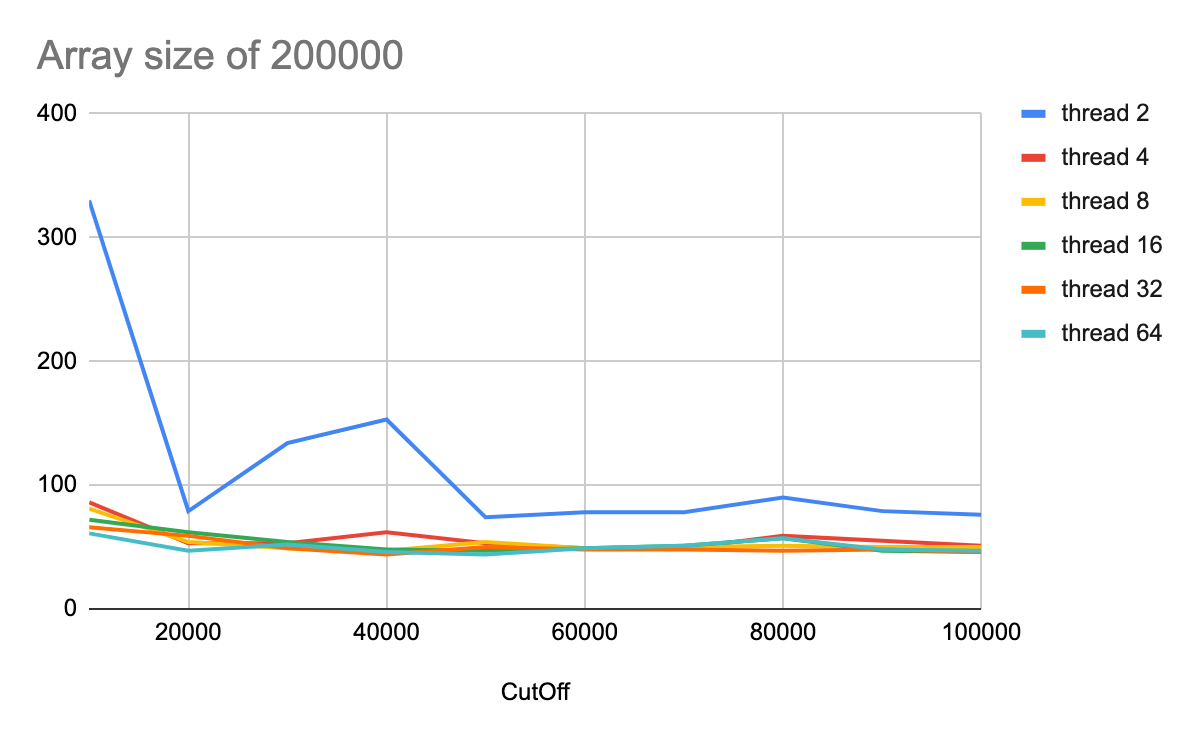
When Array size is :- 100000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CutOff | thread 2 | thread 4 | thread 8 | thread 16 | thread 32 | thread 64 |
| 10000 | 220 | 59 | 41 | 36 | 27 | 50 |
| 20000 | 108 | 35 | 31 | 24 | 23 | 24 |
| 30000 | 40 | 31 | 31 | 24 | 25 | 25 |
| 40000 | 86 | 38 | 30 | 25 | 25 | 24 |
| 50000 | 39 | 27 | 36 | 34 | 30 | 30 |
| 60000 | 33 | 34 | 33 | 33 | 32 | 32 |
| 70000 | 34 | 34 | 32 | 32 | 31 | 32 |
| 80000 | 35 | 37 | 31 | 32 | 32 | 32 |
| 90000 | 34 | 34 | 33 | 32 | 32 | 33 |
| 100000 | 36 | 33 | 32 | 32 | 33 | 32 |



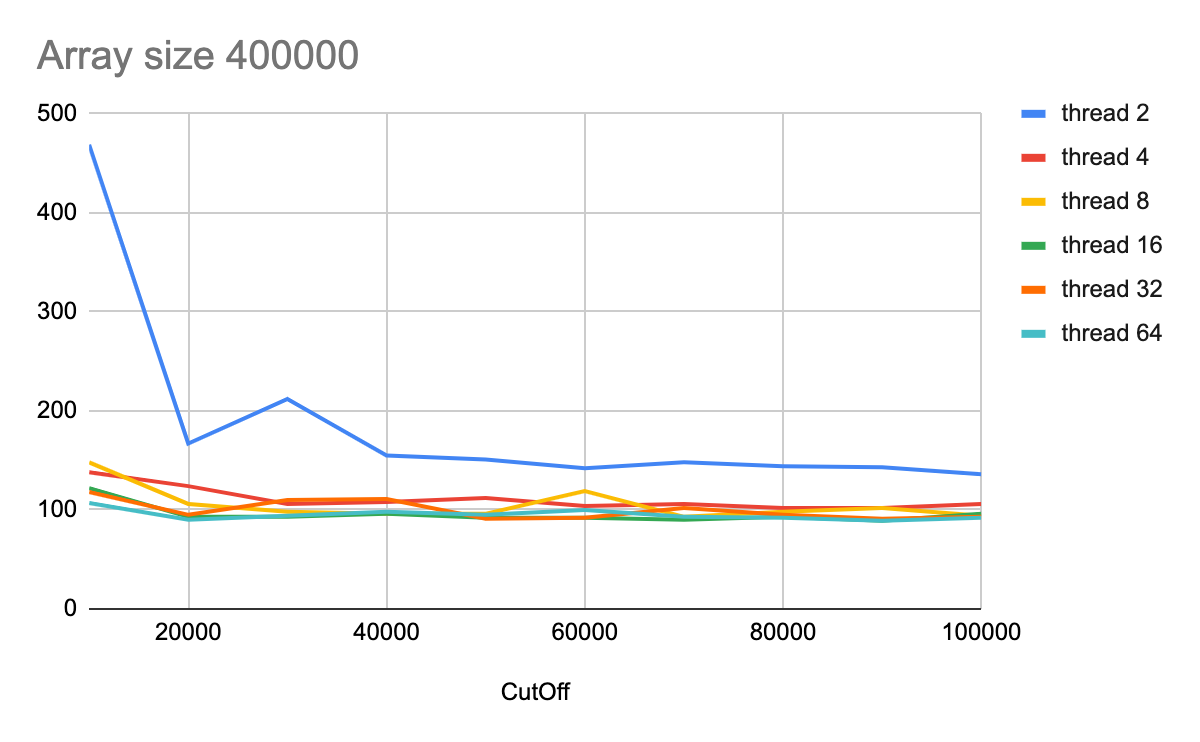
Array Size of 200000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CutOff | thread 2 | thread 4 | thread 8 | thread 16 | thread 32 | thread 64 |
| 10000 | 330 | 86 | 81 | 72 | 66 | 61 |
| 20000 | 79 | 53 | 54 | 62 | 59 | 47 |
| 30000 | 134 | 53 | 49 | 54 | 49 | 52 |
| 40000 | 153 | 62 | 47 | 48 | 44 | 46 |
| 50000 | 74 | 53 | 54 | 47 | 50 | 44 |
| 60000 | 78 | 49 | 49 | 49 | 48 | 49 |
| 70000 | 78 | 49 | 50 | 51 | 48 | 51 |
| 80000 | 90 | 59 | 51 | 57 | 47 | 57 |
| 90000 | 79 | 55 | 50 | 47 | 48 | 48 |
| 100000 | 76 | 51 | 50 | 46 | 46 | 47 |



Array size 400000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CutOff | thread 2 | thread 4 | thread 8 | thread 16 | thread 32 | thread 64 |
| 10000 | 469 | 138 | 148 | 122 | 118 | 107 |
| 20000 | 167 | 124 | 106 | 93 | 95 | 90 |
| 30000 | 212 | 106 | 98 | 93 | 110 | 94 |
| 40000 | 155 | 108 | 96 | 96 | 111 | 98 |
| 50000 | 151 | 112 | 96 | 92 | 91 | 95 |
| 60000 | 142 | 104 | 119 | 92 | 92 | 100 |
| 70000 | 148 | 106 | 93 | 90 | 102 | 93 |
| 80000 | 144 | 102 | 98 | 93 | 95 | 92 |
| 90000 | 143 | 102 | 102 | 89 | 91 | 89 |
| 100000 | 136 | 106 | 94 | 96 | 93 | 92 |



Output :-

