Satya Sri Sowmya Chinimilli (NUID - 002102974)

**Program Structures and Algorithms**

Assignment 5

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.

Output:

Part1:

1. Keeping the thread count 8, prepared a report by taking different array sizes with cutoff values to find the best value for the cutoff.
2. As per the observations from the graph, 20% of the array size would be the best cutoff to select

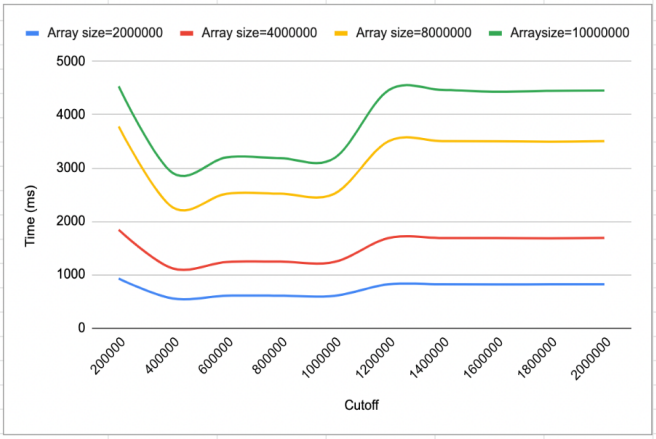
*Calculating best cut off with different array sizes*

| **Arraysize = 2000000, Threadcount = 8** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 925 |
| 400000 | 552 |
| 600000 | 604 |
| 800000 | 604 |
| 1000000 | 601 |
| 1200000 | 819 |
| 1400000 | 817 |
| 1600000 | 816 |
| 1800000 | 817 |
| 2000000 | 817 |

| **Arraysize = 4000000, Threadcount = 8** | |
|---|---|
| Cutoff | Time (ms) |
| 400000 | 1842 |
| 800000 | 1117 |
| 1200000 | 1237 |
| 1600000 | 1243 |
| 2000000 | 1238 |
| 2400000 | 1686 |
| 2800000 | 1687 |
| 3200000 | 1687 |
| 3600000 | 1681 |
| 4000000 | 1690 |

| **Arraysize = 8000000, Threadcount = 8** | |
|---|---|
| Cutoff | Time (ms) |
| 800000 | 3781 |
| 1600000 | 2263 |
| 2400000 | 2518 |
| 3200000 | 2519 |
| 4000000 | 2519 |
| 4800000 | 3504 |
| 5600000 | 3505 |
| 6400000 | 3503 |
| 7200000 | 3494 |
| 8000000 | 3505 |

| **Arraysize = 10000000, Threadcount = 8** | |
|---|---|
| Cutoff | Time (ms) |
| 1000000 | 4534 |
| 2000000 | 2909 |
| 3000000 | 3202 |
| 4000000 | 3185 |
| 5000000 | 3189 |
| 6000000 | 4460 |
| 7000000 | 4466 |
| 8000000 | 4433 |
| 9000000 | 4450 |
| 10000000 | 4454 |



Part 2:

1. Keeping the array size constant i.e, 2000000, observed the cutoff and time values for different thread counts.
2. As per the question taken the thread counts as powers of 2.
3. Based on the observations from the graph, thread size 16 is the best option to choose as it gives the optimum value.

*Calculating best number of threads*

| **Arraysize = 2000000, Threadcount = 1** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 1034 |
| 400000 | 971 |
| 600000 | 1245 |
| 800000 | 1241 |
| 1000000 | 1263 |
| 1200000 | 1303 |
| 1400000 | 1302 |
| 1600000 | 1297 |
| 1800000 | 1302 |
| 2000000 | 1300 |

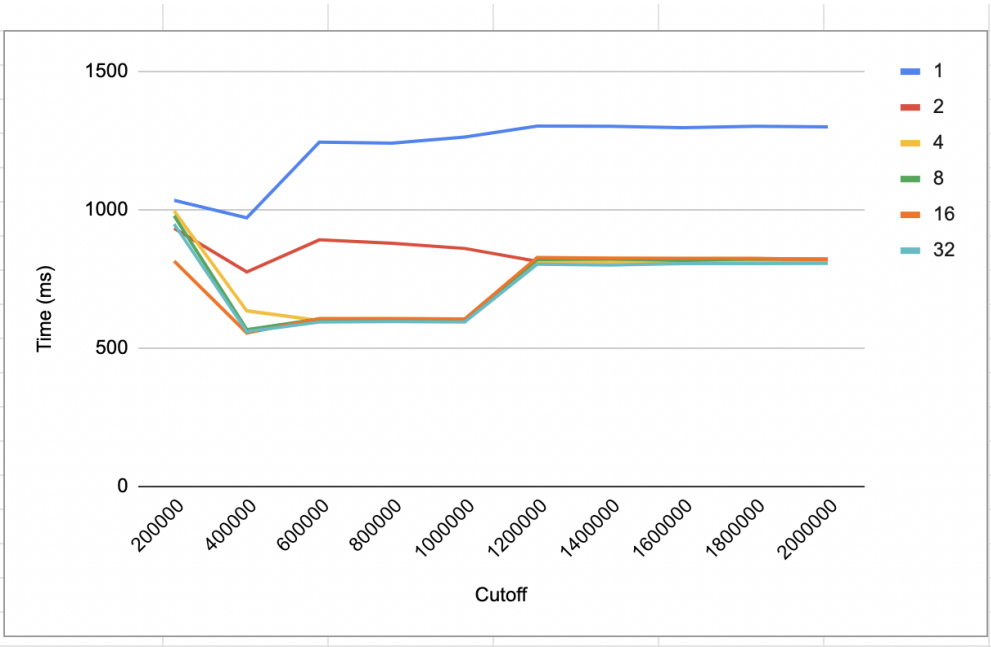
| **Arraysize = 2000000, Threadcount = 2** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 933 |
| 400000 | 775 |
| 600000 | 891 |
| 800000 | 879 |
| 1000000 | 860 |
| 1200000 | 813 |
| 1400000 | 819 |
| 1600000 | 822 |
| 1800000 | 817 |
| 2000000 | 817 |

| **Arraysize = 2000000, Threadcount = 4** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 995 |
| 400000 | 634 |
| 600000 | 599 |
| 800000 | 599 |
| 1000000 | 597 |
| 1200000 | 813 |
| 1400000 | 812 |
| 1600000 | 813 |
| 1800000 | 810 |
| 2000000 | 809 |

| **Arraysize = 2000000, Threadcount = 8** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 978 |
| 400000 | 566 |
| 600000 | 605 |
| 800000 | 604 |
| 1000000 | 603 |
| 1200000 | 820 |
| 1400000 | 822 |
| 1600000 | 817 |
| 1800000 | 822 |
| 2000000 | 822 |

| **Arraysize = 2000000, Threadcount = 16** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 814 |
| 400000 | 554 |
| 600000 | 606 |
| 800000 | 606 |
| 1000000 | 605 |
| 1200000 | 827 |
| 1400000 | 825 |
| 1600000 | 824 |
| 1800000 | 824 |
| 2000000 | 820 |

| **Arraysize = 2000000, Threadcount = 32** | |
|---|---|
| Cutoff | Time (ms) |
| 200000 | 948 |
| 400000 | 559 |
| 600000 | 594 |
| 800000 | 595 |
| 1000000 | 594 |
| 1200000 | 803 |
| 1400000 | 800 |
| 1600000 | 805 |
| 1800000 | 805 |
| 2000000 | 806 |

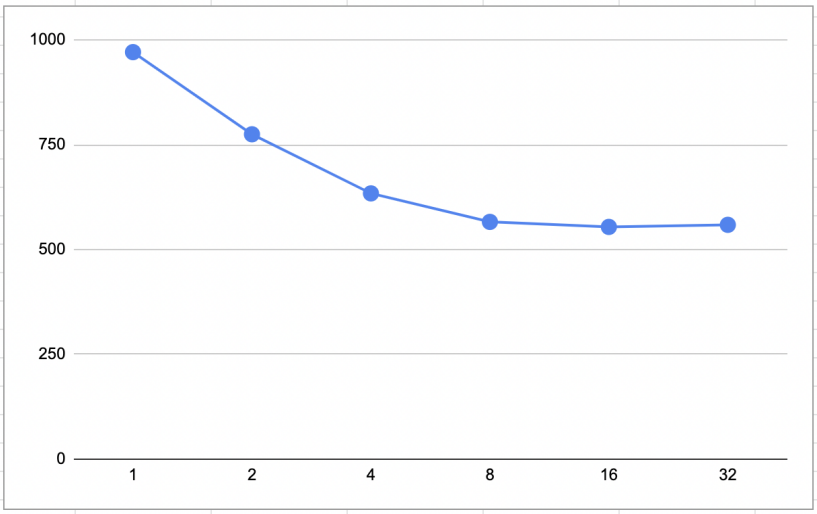


In this graph we can see that different threads have small differences after thread 4. Thus, the conclusion is that when thread increases, running time will decrease, after 50% cutoff, time will not change too much.

Part 3:

Combination of two: So I've taken 20% array size as in part1 and the thread size of 16 as in part 2, which is our best choice.

| Number of threads | Time (ms) |
|---|---|
| 1 | 971 |
| 2 | 775 |
| 4 | 634 |
| 8 | 566 |
| 16 | 554 |
| 32 | 559 |




The above graph shows the time it consumed on sorting where array size is 2000000 and cutoff is 400000 which is 20% of the array size. As per the observation, it can be noted that the lowest time occurs at thread count 16.