Xiaoran Li (NUID: 001023070)

# INFO 6205

# Program Structures & Algorithms

# Fall 2020

# Assignment No.5

- **Task**

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.

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.

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).

An appropriate combination of these.

There is a Main class and the ParSort class in the sort.par package of the INFO6205 repository. The Main class can be used as is but the ParSort class needs to be implemented where you see "TODO..."

Unless you have a good reason not to, you should just go along with the Java8-style future implementations provided for you in the class repository.Assignment Parallel Sort.pdf

You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

- **Output** (few outputs to prove relationship)

| cutoff | 1 thread | 2 thread | 4 thread | 8 thread | 16 thread | 32 thread | 64 thread |
|---|---|---|---|---|---|---|---|
| 25000 | 59.6 | 57.6 | 58.35 | 81.55 | 135 | 129.1 | 107.35 |
| 50000 | 44.25 | 43.05 | 42 | 67.8 | 64.2 | 69.4 | 34.85 |
| 75000 | 45.2 | 42.4 | 39.2 | 43 | 34 | 33.55 | 33.75 |
| 100000 | 42.5 | 43.55 | 37.7 | 40.15 | 33.7 | 31.1 | 40.35 |
| 125000 | 47.3 | 45.8 | 39.35 | 41.25 | 31.7 | 36.5 | 38.5 |
| 150000 | 49.35 | 32.5 | 27.65 | 27.65 | 28.4 | 33.65 | 40.05 |
| 175000 | 46.8 | 31.85 | 26.5 | 28.3 | 28.25 | 28 | 27.8 |
| 200000 | 47.9 | 32.9 | 26.7 | 27.7 | 28.25 | 27.25 | 29.5 |
| 225000 | 51.15 | 36.1 | 27 | 28.75 | 27.2 | 28.05 | 28.75 |
| 250000 | 51.4 | 32.15 | 27.65 | 31.1 | 28.45 | 26.4 | 28.05 |
| 275000 | 50.75 | 30.1 | 31.95 | 30.45 | 32.15 | 29.8 | 30 |
| 300000 | 49.8 | 30.65 | 29.25 | 29.85 | 31.7 | 29.25 | 29.35 |
| 325000 | 51.05 | 30.45 | 28.7 | 29.95 | 30.2 | 31.5 | 29.2 |
| 350000 | 50.75 | 29.75 | 28.75 | 31.25 | 31.1 | 37.25 | 32.9 |
| 375000 | 48.25 | 31.05 | 29.8 | 36.2 | 30.35 | 29.2 | 39.4 |
| 400000 | 48.8 | 31.35 | 28.9 | 30 | 29.75 | 28.95 | 33.95 |
| 425000 | 48.65 | 32.5 | 28.8 | 33.15 | 29.55 | 29.6 | 29.45 |
| 450000 | 51.7 | 30.55 | 28.9 | 31.4 | 29.45 | 29.55 | 30.4 |
| 475000 | 50.35 | 30.45 | 31.75 | 29.65 | 30.7 | 30.45 | 30.1 |
| 500000 | 50 | 37.4 | 30.05 | 29.8 | 32.2 | 29.8 | 29.6 |

| cutoff | 1 thread | 2 thread | 4 thread | 8 thread | 16 thread | 32 thread | 64 thread |
|---|---|---|---|---|---|---|---|
| 50000 | 96.7 | 95.7 | 93.65 | 125.4 | 170.25 | 160.05 | 131.55 |
| 100000 | 92.45 | 82.4 | 77.65 | 106.25 | 97.55 | 71.4 | 61.35 |
| 150000 | 94.7 | 83.25 | 77.9 | 86.65 | 62.8 | 56.6 | 54.7 |
| 200000 | 90.8 | 83 | 81.4 | 85.8 | 58.6 | 54.3 | 56.6 |
| 250000 | 91.8 | 86 | 77.8 | 83.45 | 59.05 | 57.65 | 57.95 |
| 300000 | 100.1 | 69.35 | 58.95 | 60.5 | 60.3 | 56.9 | 60.8 |
| 350000 | 96.3 | 65.8 | 59.15 | 60.4 | 62.3 | 61.55 | 57.65 |
| 400000 | 98.05 | 71.5 | 56.75 | 58.05 | 58 | 58.95 | 59.45 |
| 450000 | 95.3 | 68.55 | 55.15 | 61.2 | 56.85 | 59.5 | 58.9 |
| 500000 | 101.5 | 65.45 | 58 | 59.1 | 61.3 | 57.45 | 58.55 |
| 550000 | 103.25 | 65.35 | 60.5 | 61.8 | 67.6 | 64.3 | 60.7 |
| 600000 | 102.45 | 68 | 59.05 | 61.25 | 65.55 | 61 | 63.15 |
| 650000 | 102.85 | 63.35 | 61.35 | 66.1 | 62.2 | 60.3 | 63.65 |
| 700000 | 102.45 | 61.85 | 58.85 | 61.35 | 61.85 | 60.45 | 65.8 |
| 750000 | 103.35 | 61.45 | 64.6 | 62.25 | 66.5 | 64.9 | 63.25 |
| 800000 | 102.6 | 65.3 | 61.75 | 61.85 | 61.55 | 60.05 | 82.85 |
| 850000 | 101.6 | 63.25 | 59 | 66.4 | 61.8 | 59.95 | 78.85 |
| 900000 | 101.95 | 64.5 | 58.95 | 61.5 | 64.05 | 60.9 | 89.45 |
| 950000 | 105.15 | 86.05 | 63.15 | 61.75 | 67.6 | 75 | 77.8 |
| 1000000 | 101.25 | 62.5 | 59.3 | 61.65 | 63 | 62.5 | 80.55 |

| cutoff | 1 thread | 2 thread | 4 thread | 8 thread | 16 thread | 32 thread | 64 thread |
|---|---|---|---|---|---|---|---|
| 100000 | 203 | 200.1 | 171.4 | 204.6 | 307.8 | 245.85 | 277.65 |
| 200000 | 175.95 | 172.4 | 155.75 | 209.65 | 160.45 | 204.6 | 163.15 |
| 300000 | 184.9 | 176.05 | 156.6 | 143.9 | 111.75 | 115.1 | 131.75 |
| 400000 | 178.8 | 245 | 153.05 | 154.75 | 107.95 | 115.55 | 137.3 |
| 500000 | 207.45 | 214.55 | 154.95 | 149.95 | 108.7 | 118.55 | 135.15 |
| 600000 | 229.95 | 179.15 | 103.9 | 105.25 | 102.65 | 106.2 | 142.2 |
| 700000 | 185.45 | 151.2 | 105.25 | 106.6 | 106.75 | 112.15 | 125.95 |
| 800000 | 191.55 | 145.7 | 105.9 | 108.7 | 105.35 | 109.35 | 115.45 |
| 900000 | 189.35 | 148.5 | 105.2 | 106.85 | 105.95 | 109.4 | 110.35 |
| 1000000 | 190.1 | 137.15 | 102.4 | 107.55 | 106.2 | 109.45 | 111.65 |
| 1100000 | 193.3 | 129.5 | 115.8 | 121.3 | 117.9 | 121.75 | 124.6 |
| 1200000 | 193.6 | 128.45 | 115.85 | 123.65 | 117.35 | 123 | 124.9 |
| 1300000 | 192.9 | 140.6 | 118.1 | 121.15 | 118.05 | 121.65 | 124.5 |
| 1400000 | 193.25 | 145.7 | 115.9 | 121.1 | 117.3 | 119.25 | 125.25 |
| 1500000 | 193.6 | 134.1 | 116 | 121.3 | 118.2 | 121.7 | 124.4 |
| 1600000 | 192.4 | 136.9 | 116.7 | 122.8 | 117.25 | 118 | 124.35 |
| 1700000 | 193 | 135.6 | 115.5 | 120.85 | 116.9 | 118.25 | 124.8 |
| 1800000 | 195.75 | 130.8 | 117.3 | 121.55 | 116.95 | 116.5 | 124.15 |
| 1900000 | 195.4 | 134.25 | 115.5 | 121.15 | 117.35 | 119 | 125.15 |
| 2000000 | 196.15 | 132.6 | 115.7 | 123.75 | 118.15 | 120.7 | 124.45 |

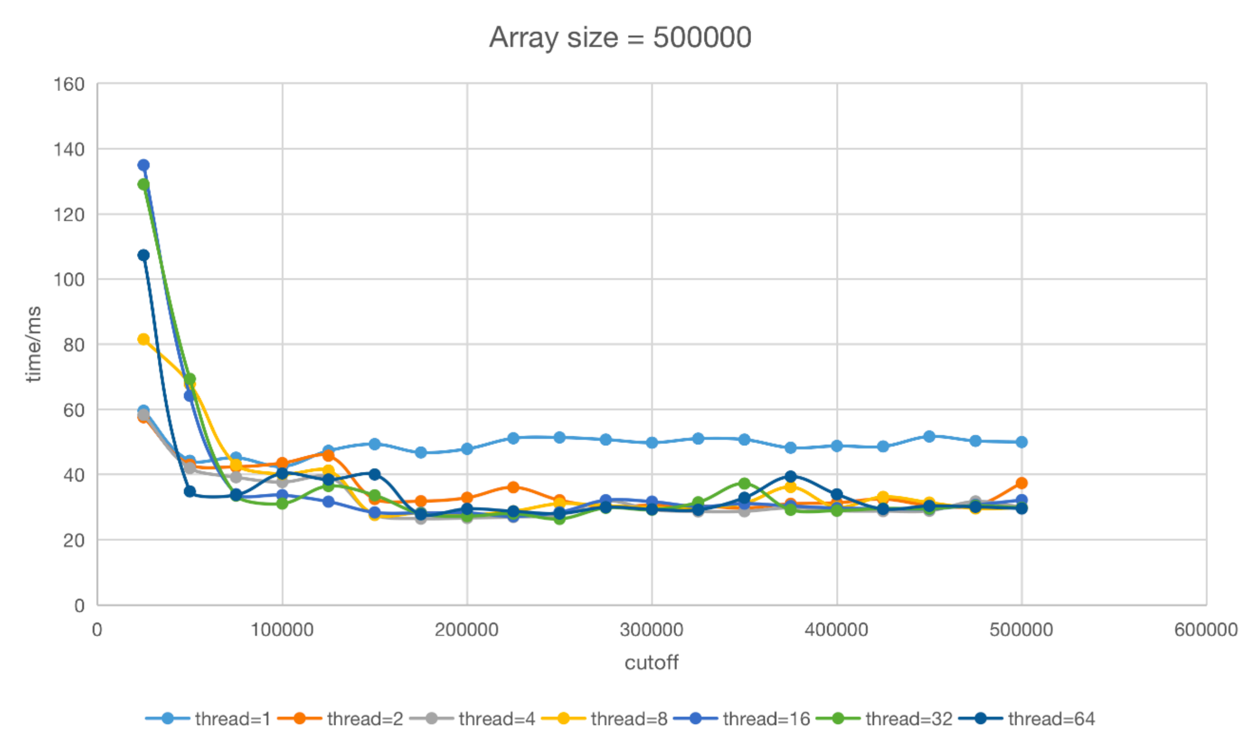
| cutoff | 1 thread | 2 thread | 4 thread | 8 thread | 16 thread | 32 thread | 64 thread |
|---|---|---|---|---|---|---|---|
| 200000 | 453.65 | 373.2 | 375.3 | 408.55 | 471.7 | 499.6 | 329.4 |
| 400000 | 327.9 | 333.65 | 317.9 | 373.45 | 368.85 | 307.25 | 242.5 |
| 600000 | 358.2 | 349.4 | 330.6 | 294.7 | 256 | 256.75 | 241.85 |
| 800000 | 364.5 | 352.15 | 333.3 | 294.05 | 241.45 | 238.3 | 247.2 |
| 1000000 | 399.6 | 344.75 | 327.95 | 310.85 | 258.7 | 241.5 | 241.1 |
| 1200000 | 406.8 | 270.15 | 244.1 | 239.75 | 265.2 | 261.05 | 254.25 |
| 1400000 | 388.85 | 394.05 | 242.05 | 239.2 | 253.55 | 258.4 | 251.75 |
| 1600000 | 391.6 | 310.5 | 234.8 | 240.25 | 265.25 | 253.45 | 267.25 |
| 1800000 | 409.6 | 280.45 | 235.35 | 241.35 | 243.65 | 254.1 | 247.4 |
| 2000000 | 408.55 | 284.5 | 245.75 | 239.75 | 268.15 | 267.5 | 255.95 |
| 2200000 | 402 | 314.3 | 248.4 | 253.15 | 253.15 | 273.1 | 264.35 |
| 2400000 | 403.8 | 329.8 | 252.25 | 252.25 | 261.85 | 275.45 | 272.85 |
| 2600000 | 401.85 | 324.85 | 253.05 | 249.8 | 251.9 | 270.25 | 271.65 |
| 2800000 | 424.65 | 329.05 | 249.7 | 251.6 | 249.3 | 271.15 | 265.75 |
| 3000000 | 416.9 | 286.6 | 253.35 | 253.7 | 251.55 | 271.1 | 265.5 |
| 3200000 | 433.15 | 321.3 | 252.55 | 252.85 | 259.9 | 268.1 | 264.7 |
| 3400000 | 401.4 | 299.15 | 251.65 | 257.35 | 267.05 | 270.6 | 273.45 |
| 3600000 | 435.05 | 297.7 | 254.9 | 267.85 | 245.95 | 269.25 | 266.85 |
| 3800000 | 401.95 | 266.5 | 247.75 | 258.75 | 252.1 | 275.4 | 270.65 |
| 4000000 | 403.15 | 266.35 | 256.85 | 267.35 | 248.9 | 271.1 | 264.65 |

- **Relationship conclusion**

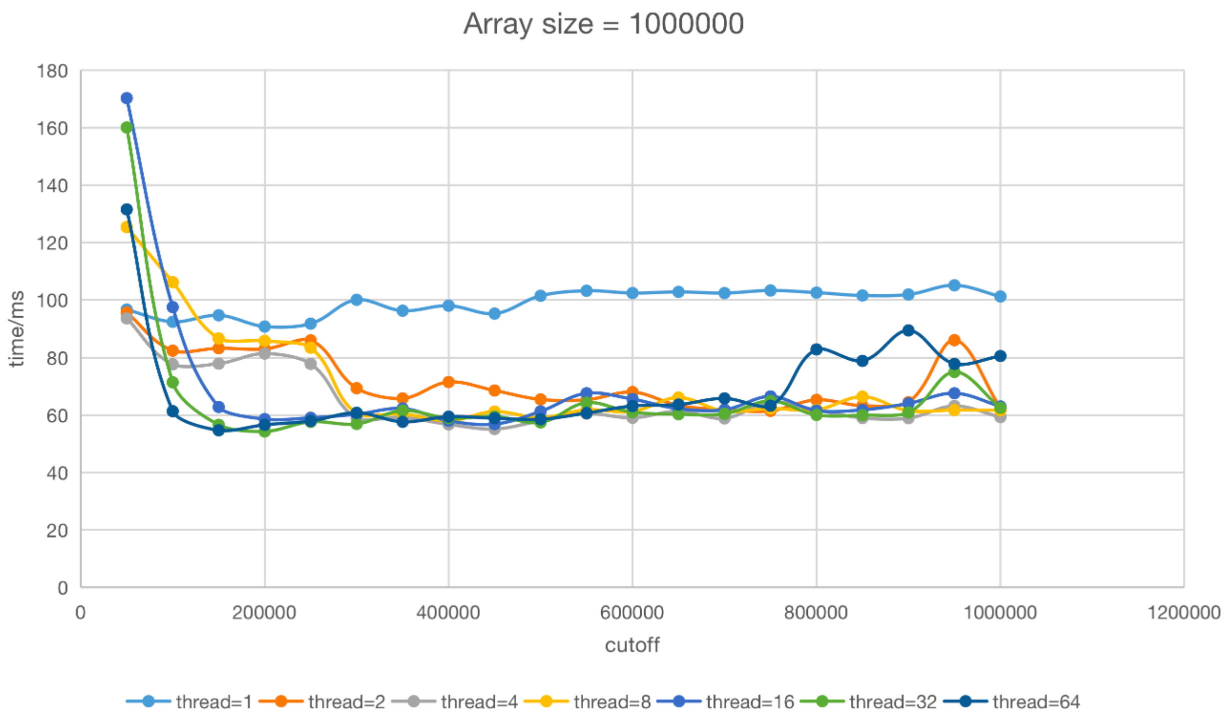
I changed both the number of thread and the number of cutoff in four different size of arrays. According to these outputs and graphs. I find these conclusions:

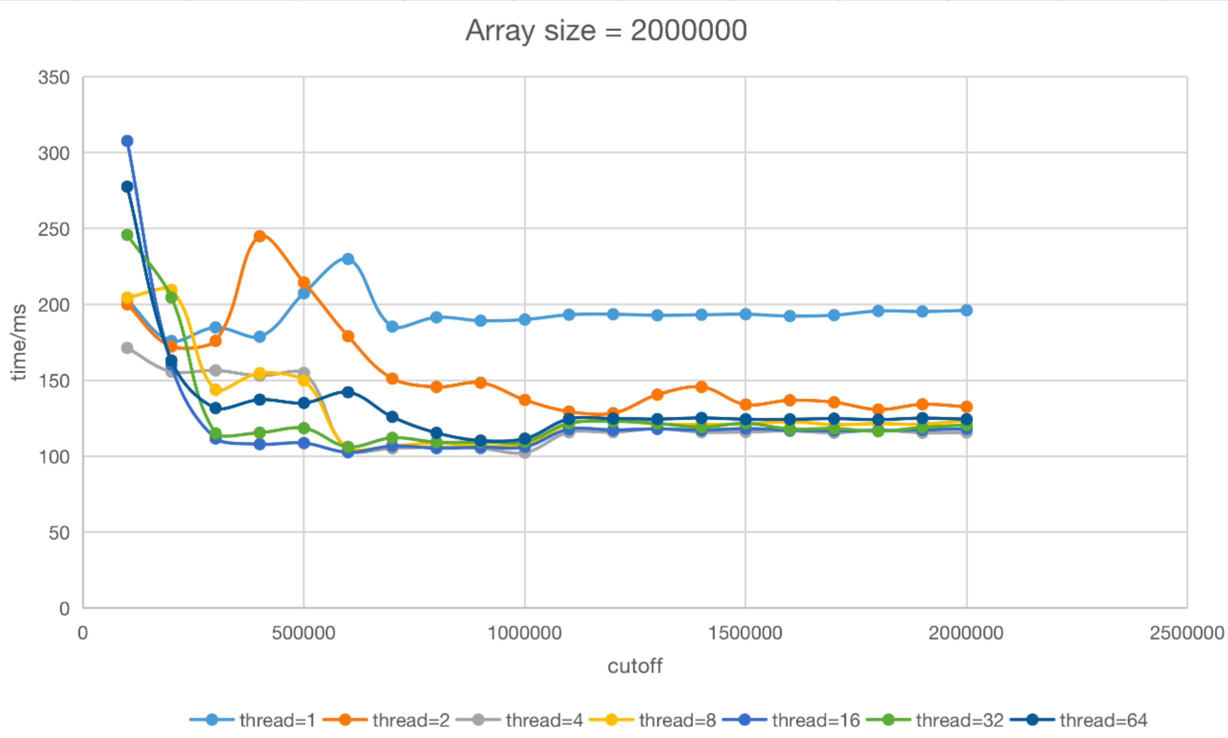
1. When the number of thread bigger than 4, there is nearly no benefit to keep increasing the number of thread. In the other words, 4 threads always have the better performance than 1 threads and 2 threads. But 8 threads, 16 threads and even more threads cost the same time withe 4 threads. Therefore, 4 threads is the best choice.
2. Look at the graph, it is obviously that when the value of cutoff is 30% of the size of arrays. The time decreases a lot. This situation is suitable for all four different size of arrays. So, the best value of cutoff should be 30% of the size of arrays.
3. According to 1 and 2, I think the best combination of thread and cutoff is that uses 4 threads and the value of cutoff is 30% of the size of arrays.

- **Evidence to support relationship** (screen shot and/or graph and/or spreadsheet)



Xiaoran Li (NUID: 001023070)



Xiaoran Li (NUID: 001023070)