Yiqing Huang (001525629)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 5**

* **Task (List down the tasks performed in the Assignment)**

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.

* **Step 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.

* **Step 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).

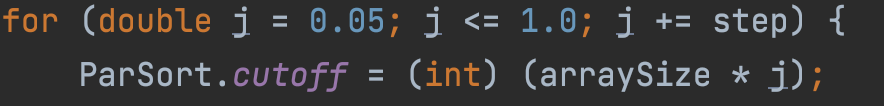
* **Step 3:**

An appropriate combination of these.

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.

* **Relationship Conclusion:**
  + The value of cutoff is not a fixed value for a better performance, it depends on the size of the array. However, the ratio () has a range for the best performance. It is between 0. to 0..
  + When using multi-threads to sort the partition, the performance is significantly better than using linear programming. Theoretically, the larger the size of the array is, the more threads the program use, the better the performance is. From my observation, using 4 threads can have a better performance than using 2 threads when sorting an array with a size range from 1 million to 16 million. *(Due to the limitation of my dual-core CPU, the maximum number of threads that can be occupied at the same time is only 4 threads)*
  + The appropriate combination from my experiment is for the ratio () and for the number of threads.
* **Evidence to support the conclusion:**

1. **I designed the experiment as the following:**
   1. For a more consolidated data when observing the result, I replaced cutoff with ratio ();



* 1. I picked up 1M, 2M, 4M, 8M, and 16M as the size of the array;
  2. Due to the limitation of my dual-core CPU, I only use 1, 2, 4, and 8 threads to sort for the partition. The purpose of choosing 8 threads is only to make sure there is little difference between 4 and 8 threads, since the maximum number of threads that can be occupied at the same time is 4 threads.

**Text

Description automatically generated**

1. **Output (Snapshot of Code output in the terminal)**

**A picture containing text

Description automatically generated**

**Calendar

Description automatically generated with medium confidence**

1. **Raw Data**
   1. **1-thread**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cutoff/array\_size | 1M | 2M | 4M | 8M | 16M |
| 0.050 | 159.3 | 103.5 | 250.2 | 425.7 | 907.1 |
| 0.075 | 72.7 | 104.0 | 212.9 | 449.1 | 897.4 |
| 0.100 | 50.9 | 104.5 | 208.8 | 424.5 | 892.0 |
| 0.125 | 50.7 | 120.8 | 207.9 | 426.6 | 893.9 |
| 0.150 | 57.7 | 121.4 | 252.1 | 507.5 | 981.4 |
| 0.175 | 56.3 | 123.6 | 252.7 | 485.6 | 1019.7 |
| 0.200 | 56.3 | 125.2 | 243.9 | 483.8 | 1003.7 |
| 0.225 | 56.6 | 129.8 | 251.4 | 496.7 | 994.1 |
| 0.250 | 56.3 | 158.5 | 239.6 | 485.3 | 1002.5 |
| 0.275 | 71.4 | 166.3 | 283.2 | 580.3 | 1357.9 |
| 0.300 | 72.2 | 154.1 | 303.4 | 621.0 | 1391.0 |
| 0.325 | 74.8 | 139.5 | 303.3 | 616.2 | 1268.7 |
| 0.350 | 73.9 | 140.1 | 300.8 | 650.1 | 1251.9 |
| 0.375 | 72.9 | 157.2 | 299.0 | 609.9 | 1271.9 |
| 0.400 | 72.3 | 144.7 | 309.2 | 600.3 | 1503.8 |
| 0.425 | 70.7 | 150.0 | 309.9 | 621.9 | 1332.8 |
| 0.450 | 69.9 | 151.4 | 316.2 | 581.9 | 1286.3 |
| 0.475 | 72.6 | 147.1 | 333.9 | 622.7 | 1398.7 |
| 0.500 | 70.8 | 141.0 | 310.7 | 638.2 | 1437.5 |
| 0.525 | 72.1 | 162.4 | 312.1 | 646.8 | 1346.9 |
| 0.550 | 72.5 | 151.4 | 312.5 | 638.3 | 1332.1 |
| 0.575 | 73.6 | 151.5 | 314.9 | 647.8 | 1342.0 |
| 0.600 | 72.2 | 150.8 | 326.1 | 654.6 | 1328.0 |
| 0.625 | 83.6 | 151.7 | 309.3 | 642.5 | 1334.8 |
| 0.650 | 72.6 | 150.8 | 310.2 | 655.4 | 1338.1 |
| 0.675 | 73.0 | 150.8 | 310.4 | 648.5 | 1337.3 |
| 0.700 | 72.0 | 150.5 | 332.6 | 655.0 | 1492.2 |
| 0.725 | 72.5 | 150.6 | 310.6 | 643.6 | 1387.0 |
| 0.750 | 72.2 | 151.3 | 319.0 | 640.8 | 1395.4 |
| 0.775 | 72.6 | 160.5 | 321.5 | 647.8 | 1333.3 |
| 0.800 | 75.0 | 152.5 | 306.2 | 641.9 | 1336.3 |
| 0.825 | 80.8 | 150.6 | 346.8 | 664.9 | 1417.2 |
| 0.850 | 74.3 | 151.0 | 309.9 | 650.8 | 1335.8 |
| 0.875 | 75.5 | 153.8 | 306.7 | 659.0 | 1319.1 |
| 0.900 | 72.1 | 153.6 | 307.2 | 647.2 | 1331.9 |
| 0.925 | 72.1 | 150.6 | 304.8 | 683.0 | 1328.2 |
| 0.950 | 72.7 | 151.0 | 315.6 | 669.5 | 1339.5 |
| 0.975 | 72.5 | 149.7 | 305.6 | 748.5 | 1322.5 |

* 1. **2-thread**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cutoff/array\_size | 1M | 2M | 4M | 8M | 16M |
| 0.050 | 53.0 | 98.5 | 207.2 | 456.3 | 830.5 |
| 0.075 | 49.6 | 96.7 | 191.0 | 393.6 | 813.9 |
| 0.100 | 49.0 | 102.8 | 185.4 | 399.8 | 816.9 |
| 0.125 | 51.2 | 93.7 | 185.8 | 400.1 | 786.8 |
| 0.150 | 57.7 | 97.2 | 202.3 | 401.0 | 839.0 |
| 0.175 | 50.4 | 95.5 | 211.9 | 400.1 | 834.1 |
| 0.200 | 48.2 | 95.0 | 193.8 | 400.6 | 830.9 |
| 0.225 | 49.7 | 95.0 | 195.4 | 408.8 | 833.0 |
| 0.250 | 61.6 | 101.5 | 188.9 | 401.0 | 828.6 |
| 0.275 | 49.8 | 99.0 | 201.2 | 416.9 | 865.8 |
| 0.300 | 49.1 | 99.8 | 198.0 | 416.2 | 853.2 |
| 0.325 | 49.9 | 104.4 | 197.3 | 419.8 | 866.8 |
| 0.350 | 49.7 | 105.8 | 208.6 | 411.8 | 851.3 |
| 0.375 | 50.4 | 100.2 | 196.4 | 433.4 | 861.4 |
| 0.400 | 49.4 | 98.3 | 200.4 | 431.6 | 854.8 |
| 0.425 | 50.0 | 98.3 | 200.5 | 418.2 | 863.8 |
| 0.450 | 48.0 | 98.1 | 198.1 | 417.4 | 874.6 |
| 0.475 | 51.5 | 110.1 | 198.1 | 405.3 | 910.2 |
| 0.500 | 50.4 | 101.6 | 198.6 | 426.8 | 864.0 |
| 0.525 | 46.6 | 90.8 | 187.9 | 408.6 | 793.3 |
| 0.550 | 46.3 | 91.4 | 199.3 | 393.1 | 819.3 |
| 0.575 | 47.3 | 92.0 | 192.2 | 408.9 | 805.8 |
| 0.600 | 48.2 | 92.2 | 191.6 | 410.3 | 804.7 |
| 0.625 | 52.4 | 92.7 | 186.3 | 390.4 | 807.3 |
| 0.650 | 52.7 | 91.1 | 188.7 | 390.9 | 810.0 |
| 0.675 | 49.6 | 90.9 | 186.1 | 401.6 | 801.2 |
| 0.700 | 50.0 | 91.9 | 186.0 | 390.2 | 812.8 |
| 0.725 | 49.3 | 91.1 | 187.3 | 390.7 | 818.6 |
| 0.750 | 50.3 | 90.8 | 202.5 | 389.2 | 797.3 |
| 0.775 | 47.1 | 93.5 | 186.5 | 412.6 | 812.1 |
| 0.800 | 45.1 | 90.8 | 187.5 | 390.7 | 796.2 |
| 0.825 | 45.7 | 91.1 | 187.9 | 390.3 | 804.2 |
| 0.850 | 47.9 | 91.1 | 188.4 | 389.7 | 806.2 |
| 0.875 | 45.3 | 103.3 | 186.3 | 405.7 | 897.3 |
| 0.900 | 45.0 | 92.3 | 186.5 | 390.3 | 799.9 |
| 0.925 | 45.3 | 91.3 | 187.6 | 403.8 | 806.7 |
| 0.950 | 45.2 | 91.9 | 202.4 | 396.0 | 796.4 |
| 0.975 | 48.8 | 91.2 | 186.7 | 397.7 | 809.0 |

* 1. **4-thread**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cutoff/array\_size | 1M | 2M | 4M | 8M | 16M |
| 0.050 | 49.1 | 102.9 | 205.0 | 405.6 | 809.3 |
| 0.075 | 56.2 | 98.8 | 178.4 | 373.2 | 784.2 |
| 0.100 | 45.3 | 96.1 | 179.7 | 382.5 | 785.3 |
| 0.125 | 44.5 | 87.8 | 181.7 | 393.0 | 760.6 |
| 0.150 | 41.8 | 86.3 | 176.5 | 358.2 | 748.2 |
| 0.175 | 40.9 | 85.6 | 178.4 | 360.2 | 736.4 |
| 0.200 | 42.5 | 86.8 | 190.9 | 364.7 | 741.9 |
| 0.225 | 41.9 | 87.2 | 173.5 | 369.2 | 733.2 |
| 0.250 | 42.1 | 85.2 | 173.8 | 366.3 | 740.4 |
| 0.275 | 38.5 | 83.3 | 168.9 | 342.7 | 699.1 |
| 0.300 | 39.8 | 84.3 | 164.9 | 343.6 | 698.4 |
| 0.325 | 39.1 | 86.0 | 164.8 | 355.6 | 724.1 |
| 0.350 | 39.0 | 94.0 | 165.6 | 343.7 | 699.2 |
| 0.375 | 39.5 | 86.0 | 165.0 | 343.2 | 781.6 |
| 0.400 | 38.5 | 81.0 | 165.3 | 344.3 | 711.0 |
| 0.425 | 39.1 | 80.4 | 185.7 | 354.6 | 745.4 |
| 0.450 | 41.3 | 80.7 | 169.4 | 345.9 | 709.7 |
| 0.475 | 38.7 | 80.4 | 166.4 | 349.5 | 716.0 |
| 0.500 | 39.2 | 81.2 | 165.4 | 346.4 | 704.6 |
| 0.525 | 43.8 | 93.2 | 207.5 | 403.9 | 856.7 |
| 0.550 | 43.8 | 91.8 | 286.5 | 392.1 | 843.5 |
| 0.575 | 44.2 | 90.8 | 201.7 | 389.8 | 815.5 |
| 0.600 | 50.3 | 92.7 | 222.8 | 409.5 | 802.7 |
| 0.625 | 44.2 | 92.6 | 193.1 | 437.5 | 801.0 |
| 0.650 | 44.4 | 179.5 | 192.1 | 399.2 | 814.3 |
| 0.675 | 43.9 | 92.0 | 209.2 | 425.5 | 799.3 |
| 0.700 | 43.6 | 91.2 | 190.2 | 392.1 | 814.6 |
| 0.725 | 47.5 | 104.6 | 192.1 | 404.7 | 796.4 |
| 0.750 | 45.0 | 93.3 | 191.6 | 393.4 | 882.6 |
| 0.775 | 43.6 | 92.4 | 194.6 | 399.3 | 823.2 |
| 0.800 | 44.7 | 91.0 | 206.7 | 396.0 | 831.9 |
| 0.825 | 45.0 | 91.0 | 193.2 | 405.0 | 797.1 |
| 0.850 | 47.5 | 90.1 | 193.4 | 393.8 | 813.1 |
| 0.875 | 46.9 | 91.4 | 193.5 | 394.2 | 815.9 |
| 0.900 | 49.3 | 92.3 | 190.7 | 403.0 | 812.7 |
| 0.925 | 56.2 | 90.8 | 191.5 | 400.3 | 813.9 |
| 0.950 | 47.8 | 90.7 | 189.9 | 394.2 | 804.4 |
| 0.975 | 44.4 | 90.2 | 205.0 | 402.5 | 814.3 |

* 1. **8-thread**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cutoff/array\_size | 1M | 2M | 4M | 8M | 16M |
| 0.050 | 51.1 | 107.8 | 210.4 | 441.2 | 856.4 |
| 0.075 | 46.3 | 92.6 | 183.4 | 368.2 | 781.5 |
| 0.100 | 44.1 | 88.3 | 180.8 | 378.9 | 745.0 |
| 0.125 | 43.4 | 89.0 | 184.5 | 398.6 | 794.6 |
| 0.150 | 41.9 | 86.6 | 179.5 | 375.8 | 787.1 |
| 0.175 | 43.1 | 86.7 | 183.6 | 353.2 | 811.6 |
| 0.200 | 41.6 | 101.5 | 179.8 | 362.0 | 1022.9 |
| 0.225 | 42.3 | 87.1 | 194.5 | 355.8 | 802.3 |
| 0.250 | 41.6 | 87.3 | 175.6 | 452.6 | 733.2 |
| 0.275 | 39.9 | 84.6 | 167.1 | 422.8 | 701.4 |
| 0.300 | 47.3 | 86.2 | 168.8 | 433.6 | 713.3 |
| 0.325 | 41.1 | 87.5 | 171.2 | 347.3 | 703.8 |
| 0.350 | 57.0 | 82.1 | 169.0 | 357.4 | 699.4 |
| 0.375 | 46.8 | 83.3 | 170.6 | 349.0 | 756.6 |
| 0.400 | 43.9 | 81.2 | 169.8 | 343.4 | 704.0 |
| 0.425 | 41.4 | 81.7 | 168.5 | 344.4 | 714.9 |
| 0.450 | 40.3 | 79.8 | 186.3 | 367.0 | 702.5 |
| 0.475 | 39.2 | 81.4 | 169.7 | 347.4 | 717.3 |
| 0.500 | 40.5 | 80.8 | 168.5 | 346.8 | 778.6 |
| 0.525 | 44.5 | 90.5 | 192.5 | 391.6 | 918.6 |
| 0.550 | 43.6 | 90.8 | 192.9 | 416.0 | 930.5 |
| 0.575 | 46.6 | 90.9 | 189.2 | 393.5 | 814.2 |
| 0.600 | 50.0 | 93.6 | 190.0 | 391.4 | 1163.8 |
| 0.625 | 50.0 | 105.3 | 191.8 | 391.0 | 1395.0 |
| 0.650 | 47.6 | 94.2 | 201.2 | 402.8 | 816.7 |
| 0.675 | 45.8 | 89.7 | 193.1 | 396.6 | 815.1 |
| 0.700 | 43.9 | 91.0 | 193.8 | 396.3 | 804.3 |
| 0.725 | 44.1 | 91.3 | 192.6 | 399.2 | 812.5 |
| 0.750 | 45.2 | 90.6 | 190.8 | 398.4 | 830.8 |
| 0.775 | 45.1 | 91.0 | 190.2 | 390.6 | 812.0 |
| 0.800 | 44.3 | 92.2 | 191.7 | 394.0 | 824.1 |
| 0.825 | 65.4 | 91.3 | 189.6 | 400.2 | 811.6 |
| 0.850 | 43.0 | 90.9 | 203.5 | 392.2 | 900.1 |
| 0.875 | 44.4 | 91.6 | 191.9 | 392.9 | 803.2 |
| 0.900 | 44.6 | 91.0 | 193.3 | 392.6 | 826.3 |
| 0.925 | 43.3 | 90.6 | 191.8 | 410.7 | 801.1 |
| 0.950 | 43.5 | 90.4 | 190.1 | 390.6 | 818.2 |
| 0.975 | 44.8 | 91.0 | 191.2 | 391.0 | 812.7 |

1. **Thread Fixed Graphics**

**Chart, line chart

Description automatically generatedChart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart, histogram

Description automatically generated**

1. **Array Size Fixed Graphics**

**Chart, histogram

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart, histogram

Description automatically generated**

**Chart, histogram

Description automatically generated**

1. **Observation**
   1. Form the *Thread Fixed Graphics*, it is obvious that in general, when the ratio () is in range of to , the executing time is shorter.
   2. Form the *Array Size Fixed Graphics*,
      1. The executing time is significantly shorter when using multi-threads than using only one thread;
      2. Also, it proves my speculation that there is little difference between 4-thread and 8-thread, since the maximum number of threads that can be occupied at the same time is 4 threads;
      3. For the size of arrays that I chose (1M, 2M, 4M, 8M, and 16M), 4 threads can have the best performance.