**Ram Gopal Varma Alluri**

**002196886**

**INFO 6205 Program Structures and Algorithms**

**Assignment 5**

Task:

This task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. 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 cut-off schemes.

Performance based on different cut-offs and arrays sizes (time: millisecond)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cutt-off | 20^4 | 40^5 | 80^5 | 20^6 | 40^6 |  |
| 10000 | 105 | 490 | 934 | 1922 | 2894 |  |
| 11000 | 36 | 316 | 473 | 1016 | 1975 |  |
| 12000 | 39 | 245 | 473 | 1014 | 2005 |  |
| 13000 | 34 | 224 | 419 | 986 | 1880 |  |
| 14000 | 10 | 231 | 414 | 962 | 1879 |  |
| 15000 | 9 | 227 | 456 | 946 | 2053 |  |
| 16000 | 26 | 234 | 460 | 808 | 2047 |  |
| 32000 | 23 | 287 | 533 | 890 | 2030 |  |
| 64000 | 21 | 192 | 463 | 725 | 2208 |  |
| 128000 | 12 | 207 | 345 | 871 | 2361 |  |
|  |  |  |  |  |  |  |

Performance based on different threads

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cut-off | 1 thread | 2 thread | 4 theads | 8 threads | 16 thread | 32 thread | 64 threads |
| 510000 | 2141ms | 1764ms | 1174ms | 788ms | 778ms | 769ms | 811ms |
| 520000 | 1779ms | 1869ms | 1180ms | 807ms | 798ms | 797ms | 764ms |
| 520000 | 1880ms | 1843ms | 1160ms | 805ms | 769ms | 758ms | 759ms |
| 540000 | 1913ms | 1763ms | 1174ms | 817ms | 769ms | 775ms | 764ms |
| 550000 | 1882ms | 1785ms | 1186ms | 821ms | 770ms | 779ms | 786ms |
| 560000 | 1785ms | 1889ms | 1271ms | 801ms | 840ms | 781ms | 777ms |
| 570000 | 1770ms | 1737ms | 1184ms | 816ms | 770ms | 760ms | 776ms |
| 580000 | 1831ms | 1726ms | 1187ms | 810ms | 778ms | 779ms | 781ms |
| 590000 | 1942ms | 1750ms | 1180ms | 825ms | 769ms | 777ms | 776ms |
| 600000 | 1947ms | 1905ms | 1195ms | 831ms | 779ms | 783ms | 768ms |
| 610000 | 1910ms | 1723ms | 1190ms | 794ms | 775ms | 762ms | 767ms |

Relationship conclusion :

* Parallel sorting is more efficient at lower cut-off values than system sorting.
* Sorting a small array does not depend on the size of the cut-off however, when we increase the size of the random array, the larger the cut-off size, the lower the efficiency will be, which means that sorting larger data will require more time. In this case, we would split it into smaller arrays and then sort it by system sort, which is much more efficient.
* As I increased the threads from 2 to 4, the runtime significantly decreased. This indicates that having more threads brings better performance. However, as I increased threads from 8 to 16, the runtime remained the same as 4 threads. Probably because there will be a certain cost when switching execution between multiple threads.

**Evidence to support relationship**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated