Akash Bharadwaj Karthik (002787011)

**Program Structures & Algorithms**

**Spring 2023 (Sec -8)**

**Assignment No. 5**

**Task**

* (Part 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.*
* (Part 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)*
* (Part 3) *An appropriate combination of these.*

**Relationship Conclusion:**

After running the algorithm for different array sizes and cutoff values, and using varying thread counts for each array size, the following results are observed:

The cutoff values are always maintained below array size, and is varied from 0.1 to 1.0 in steps of 0.1; this helps accurately measure run time of parallelized sort and prevents system sort.

Maximum depth is given by:

**Evidence to support the conclusion:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Average Run Time (ms) for Array Size = 100000 | | | | | |
| Cutoff/Array Size | 2 Threads | 4 Threads | 8 Threads | 16 Threads | 32 Threads | 64 Threads |
| 0.1 | 19 | 16 | 11.4 | 7.6 | 6.4 | 8.6 |
| 0.2 | 7.3 | 11.6 | 8.2 | 6.5 | 5.9 | 7.2 |
| 0.3 | 12.2 | 9.4 | 8.8 | 7.3 | 6.9 | 7.9 |
| 0.4 | 9.4 | 9.3 | 8.1 | 7.4 | 7.2 | 8.8 |
| 0.5 | 7.1 | 9.5 | 8.4 | 7.3 | 7.3 | 8.7 |
| 0.6 | 7.3 | 11.3 | 9.2 | 8.3 | 7.9 | 11.2 |
| 0.7 | 8.1 | 11.2 | 9 | 8.3 | 8 | 10.8 |
| 0.8 | 9.7 | 11.6 | 8.9 | 8.4 | 9.1 | 10.6 |
| 0.9 | 9.6 | 10.9 | 8.8 | 7.9 | 8.6 | 9.8 |
| 1 | 15.3 | 11.1 | 8.3 | 7.9 | 9.1 | 9.3 |
| Optimal (cutoff/Array Size) Ratio: | 0.5 | 0.4 | 0.2 | 0.2 | 0.2 | 0.2 |
| Optimal Thread Count: | 32 |  |  |  |  |  |

There are a few outliers for optimal **(cutoff/Array Size) Ratio**, but the majority **optimal ratio** which gives the **lowest run-time** is found to be 0.2. The optimal thread count is found to be 32, but larger array sizes need to be considered as this could be an outlier.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Average Run Time (ms) for Array Size = 200000 | | | | | |
| Cutoff/Array Size | 2 Threads | 4 Threads | 8 Threads | 16 Threads | 32 Threads | 64 Threads |
| 0.1 | 21.3 | 18.6 | 15.9 | 15.3 | 16.5 | 13.2 |
| 0.2 | 18.8 | 16 | 11.4 | 11.7 | 16 | 12.7 |
| 0.3 | 23.6 | 14.8 | 14.5 | 14.7 | 16.5 | 14.3 |
| 0.4 | 24.3 | 14.2 | 14.3 | 15.2 | 16.2 | 15 |
| 0.5 | 26 | 14.6 | 14.2 | 15.3 | 16 | 15.1 |
| 0.6 | 21.4 | 17 | 16 | 18 | 18.7 | 17.6 |
| 0.7 | 19.7 | 16.4 | 15.7 | 17 | 18 | 17.4 |
| 0.8 | 18.8 | 16.2 | 15.8 | 16.7 | 17.5 | 17.4 |
| 0.9 | 17.2 | 15.7 | 15.6 | 16.4 | 16.6 | 16.3 |
| 1 | 17.1 | 15.6 | 15.5 | 19.1 | 16.4 | 16.4 |
| Optimal (cutoff/Array Size) Ratio : | 1 | 0.4 | 0.2 | 0.2 | 0.2 | 0.2 |
| Optimal Thread Count : | 8 |  |  |  |  |  |

There are a few outliers for optimal **(cutoff/Array Size) Ratio**, but the majority **optimal ratio** which gives the **lowest run-time** is found to be 0.2. The **optimal thread count** for which performance time is least is found to be 8,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Average Run Time (ms) for Array Size = 300000 | | | | | |
| Cutoff/Array Size | 2 Threads | 4 Threads | 8 Threads | 16 Threads | 32 Threads | 64 Threads |
| 0.1 | 28.6 | 28.9 | 24.5 | 22.9 | 17.6 | 18.6 |
| 0.2 | 24.4 | 23.8 | 17.8 | 17.8 | 17.8 | 18.1 |
| 0.3 | 26.7 | 22 | 20.7 | 21.9 | 21.5 | 22.8 |
| 0.4 | 25.7 | 21.1 | 22.9 | 21.3 | 22.4 | 20.7 |
| 0.5 | 27.1 | 21.6 | 21.4 | 21.8 | 22.1 | 22.6 |
| 0.6 | 24.1 | 27.2 | 25.6 | 24.6 | 25 | 25.7 |
| 0.7 | 24.9 | 25 | 24.1 | 23.7 | 24 | 23.9 |
| 0.8 | 24 | 24.8 | 23.8 | 23.9 | 24 | 24.3 |
| 0.9 | 25.4 | 24.2 | 24.3 | 24.7 | 24.1 | 24.2 |
| 1 | 25.8 | 24.1 | 24.2 | 23 | 23.8 | 24 |
| Optimal (cutoff/Array Size) Ratio : | 0.8 | 0.4 | 0.2 | 0.2 | 0.1 | 0.2 |
| Optimal Thread Count : | 8 |  |  |  |  |  |

There are a few outliers for optimal **(cutoff/Array Size) Ratio**, but the majority **optimal ratio** which gives the **lowest run-time** is found to be 0.2. The **optimal thread count** for which performance time is least is found to be 8,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Average Run Time (ms) for Array Size = 400000 | | | | | |
| Cutoff/Array Size | 2 Threads | 4 Threads | 8 Threads | 16 Threads | 32 Threads | 64 Threads |
| 0.1 | 33.7 | 33.8 | 34.4 | 27 | 24 | 24.2 |
| 0.2 | 41.5 | 30.5 | 23 | 23.5 | 24 | 23.7 |
| 0.3 | 42.7 | 26.1 | 30.5 | 28.4 | 29.4 | 30.1 |
| 0.4 | 37.8 | 29 | 28.2 | 28.7 | 31.2 | 32.1 |
| 0.5 | 37.6 | 27.3 | 29.2 | 29.3 | 31 | 30 |
| 0.6 | 32.6 | 33 | 32.7 | 34 | 34.1 | 36.6 |
| 0.7 | 32.1 | 32.8 | 34 | 32.4 | 32.3 | 35.4 |
| 0.8 | 33.2 | 32.4 | 33.2 | 32.2 | 32.6 | 34.1 |
| 0.9 | 33.2 | 32.7 | 31.5 | 32.6 | 31.6 | 32.9 |
| 1 | 32.3 | 30.8 | 33.2 | 32.1 | 30.9 | 32.5 |
| Optimal (cutoff/Array Size) Ratio : | 0.7 | 0.3 | 0.2 | 0.2 | 0.2 | 0.2 |
| Optimal Thread Count : | 8 |  |  |  |  |  |

There are a few outliers for optimal **(cutoff/Array Size) Ratio**, but the majority **optimal ratio** which gives the **lowest run-time** is found to be 0.2. The **optimal thread count** for which performance time is least is found to be 8,

**Graphs – (Cutoff/Size) vs Runtime for majority optimal Thread Count of 8:**

**Console Output (raw cutoff values and 10-run timings):**

**Array Size: 100,000:**

A picture containing text, plaque

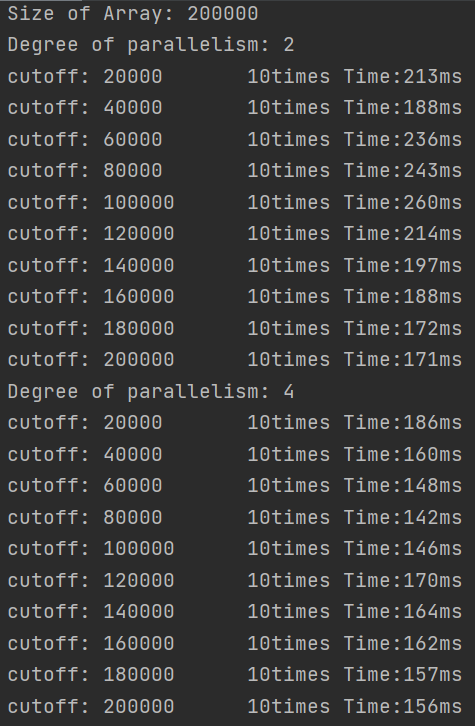
Description automatically generatedA picture containing text, plaque

Description automatically generated

A picture containing text, plaque

Description automatically generated

**Array Size: 200,000:**



A picture containing text, plaque

Description automatically generatedA picture containing text, plaque

Description automatically generated

**Array Size: 300,000:**

A picture containing text, plaque

Description automatically generatedA picture containing text, plaque

Description automatically generated

A picture containing text, plaque

Description automatically generated

**Array Size: 400,000:**

A screenshot of a computer

Description automatically generated with medium confidenceA picture containing text, plaque

Description automatically generated

A picture containing text, plaque

Description automatically generated