Fiedorowicz Samuel
Nguyen Ngoc-Lam

Testeigenschaften:

Intel Core i7-3770K CPU 3.50GHz (Ivy Bridge), 1 CPU, 8 logical and 4 physical cores

Es wurde ein Array mit 10_000_000 Elementen erstellt welches zu sortieren war. Dieses Array war für alle Ausführungen gleich, damit kein Bias entsteht bzw. alle Methoden denselben Ausgang haben. Jede Methode wurde zwischen 15-100 mal durchlaufen und ein Mittelwert gebildet.

Sollte der Threshold Wert unterschritten werden wird die Sortierung Sequenziell anstatt Parallel durchgeführt.

Der negative Threshold Wert ist zum Vergleichen ob der Naive und Threshold Ansatz gleich sind.

Ergebnis:

```
// * Summary *

BenchmarkDotNet=v0.13.5, OS=Windows 11 (10.0.22000.1574/21H2/SunValley)
Intel Core i7-3770K CPU 3.50GHz (Ivy Bridge), 1 CPU, 8 logical and 4 physical cores
.NET SDK=7.0.201
  [Host]     : .NET 7.0.3 (7.0.323.6910), X64 RyuJIT AVX
  DefaultJob : .NET 7.0.3 (7.0.323.6910), X64 RyuJIT AVX


|                    Method |   Threshold |       Mean |    Error |    StdDev |         Gen0 |       Gen1 |      Gen2 |  Allocated |
|-------------------------- |------------ |-----------:|---------:|----------:|-------------:|-----------:|----------:|-----------:|
|      QuickSortSequentially | -2147483648 |   780.7 ms |  6.16 ms |   5.46 ms |            - |          - |         - |   38.15 MB |
|     QuickSortParallelNaive | -2147483648 | 1,799.5 ms | 29.83 ms |  27.90 ms |  627000.0000 |  1000.0000 |         - | 2530.43 MB |
| QuickSortParallelThreshold | -2147483648 | 1,988.4 ms | 37.87 ms |  37.19 ms |  627000.0000 |  1000.0000 |         - | 2530.33 MB |
|      MergeSortSequentially | -2147483648 | 1,440.3 ms | 24.99 ms |  23.38 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
|     MergeSortParallelNaive | -2147483648 | 3,105.6 ms | 62.10 ms |  48.48 ms | 1223000.0000 |  9000.0000 | 3000.0000 | 5178.75 MB |
| MergeSortParallelThreshold | -2147483648 | 3,286.7 ms | 63.24 ms |  56.06 ms | 1224000.0000 |  7000.0000 | 3000.0000 | 5178.75 MB |
|      QuickSortSequentially |      100000 |   780.7 ms |  5.04 ms |   4.21 ms |            - |          - |         - |   38.15 MB |
|     QuickSortParallelNaive |      100000 | 1,810.9 ms | 35.30 ms |  37.77 ms |  627000.0000 |          - |         - | 2530.39 MB |
| QuickSortParallelThreshold |      100000 |   195.9 ms |  3.53 ms |   6.19 ms |            - |          - |         - |   38.19 MB |
|      MergeSortSequentially |      100000 | 1,417.9 ms | 22.21 ms |  20.78 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
|     MergeSortParallelNaive |      100000 | 3,163.1 ms | 63.00 ms | 110.35 ms | 1224000.0000 |  7000.0000 | 3000.0000 | 5178.75 MB |
| MergeSortParallelThreshold |      100000 |   606.1 ms | 11.99 ms |  18.67 ms |  283000.0000 | 13000.0000 | 3000.0000 | 1440.38 MB |
|      QuickSortSequentially |     1250000 |   775.9 ms |  6.26 ms |   5.22 ms |            - |          - |         - |   38.15 MB |
|     QuickSortParallelNaive |     1250000 | 1,888.8 ms | 36.87 ms |  36.21 ms |  627000.0000 |          - |         - | 2530.37 MB |
| QuickSortParallelThreshold |     1250000 |   316.2 ms |  4.00 ms |   3.74 ms |            - |          - |         - |   38.15 MB |
|      MergeSortSequentially |     1250000 | 1,414.5 ms | 16.58 ms |  14.70 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
|     MergeSortParallelNaive |     1250000 | 3,166.8 ms | 62.82 ms | 101.44 ms | 1224000.0000 |  9000.0000 | 3000.0000 | 5178.75 MB |
| MergeSortParallelThreshold |     1250000 |   576.6 ms | 11.11 ms |  11.89 ms |  283000.0000 | 16000.0000 | 3000.0000 | 1440.35 MB |
|      QuickSortSequentially |     2500000 |   769.4 ms |  5.45 ms |   5.10 ms |            - |          - |         - |   38.15 MB |
|     QuickSortParallelNaive |     2500000 | 1,795.9 ms | 35.39 ms |  43.46 ms |  627000.0000 |          - |         - | 2530.79 MB |
| QuickSortParallelThreshold |     2500000 |   327.8 ms |  3.74 ms |   3.50 ms |            - |          - |         - |   38.15 MB |
|      MergeSortSequentially |     2500000 | 1,392.5 ms | 13.31 ms |  11.80 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
|     MergeSortParallelNaive |     2500000 | 3,091.0 ms | 61.38 ms | 107.51 ms | 1224000.0000 |  7000.0000 | 3000.0000 | 5178.76 MB |
| MergeSortParallelThreshold |     2500000 |   818.4 ms | 15.48 ms |  14.48 ms |  283000.0000 | 14000.0000 | 3000.0000 | 1440.34 MB |
|      QuickSortSequentially |     5000000 |   768.7 ms |  4.72 ms |   4.18 ms |            - |          - |         - |   38.15 MB |
|     QuickSortParallelNaive |     5000000 | 1,840.9 ms | 33.56 ms |  31.39 ms |  627000.0000 |  2000.0000 |         - | 2530.64 MB |
| QuickSortParallelThreshold |     5000000 |   663.6 ms |  7.62 ms |   7.12 ms |            - |          - |         - |   38.15 MB |
|      MergeSortSequentially |     5000000 | 1,411.8 ms | 21.91 ms |  20.49 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
|     MergeSortParallelNaive |     5000000 | 3,080.4 ms | 60.79 ms |  98.17 ms | 1224000.0000 |  9000.0000 | 3000.0000 | 5178.75 MB |
| MergeSortParallelThreshold |     5000000 | 1,386.1 ms | 14.50 ms |  12.85 ms |  278000.0000 |  4000.0000 | 4000.0000 | 1440.34 MB |
```

Schlussfolgerungen:

Es ist zu erkennen, dass in allen Naiven Implementationen die Software länger für das Sortieren braucht und gleichzeitig am meisten Speicher verwendet.

Die Threshold Variante ist mit Abstand die schnellste (außer der negative Threshold Wert) Methode bei den durchgeführten Tests.

Der Speicherplatzverbrauch beim Quicksort unterscheidet sich kaum zwischen den Sequenziellen und Threshold Ansatz. Beim naiven Ansatz ist ein deutlicher Speicherverbrauch zu sehen.

Auch beim Mergesort ist der Speicherverbrauch zwischen Sequenziellen und Thresholdansatz minimal, nur der naive Ansatz verbraucht deutlich mehr Speicherplatz.

Das Ergebnis ist insofern verwunderlich, dass selbst bei einem kleinen Threshold wie 100k trotzdem noch ein Performance gewinn rauszuholen ist.