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.

Ergebnis:

```
BenchmarkDotNet=v0.13.5, OS=Windows 11 (10.0.22000.1696/21H2/SunValley)
Intel Core i7-3770K CPU 3.50GHz (Ivy Bridge), 1 CPU, 8 logical and 4 physical cores
.NET SDK=7.0.202
  [Host]     : .NET 7.0.4 (7.0.423.11508), X64 RyuJIT AVX
  DefaultJob : .NET 7.0.4 (7.0.423.11508), X64 RyuJIT AVX


|                    Method | Threshold |       Mean |     Error |    StdDev |     Median |         Gen0 |       Gen1 |      Gen2 |   Allocated |
|------------------------- |---------- |-----------:|----------:|----------:|-----------:|-------------:|-----------:|----------:|------------:|
|      QuickSortSequentially |      1000 |   776.9 ms |   3.78 ms |   3.35 ms |   775.9 ms |            - |          - |         - |    38.15 MB |
|     QuickSortParallelNaive |      1000 | 1,608.1 ms | 232.71 ms | 682.50 ms | 1,590.0 ms |  647000.0000 |          - |         - |  2607.03 MB |
| QuickSortParallelThreshold |      1000 |   136.0 ms |   7.36 ms |  21.72 ms |   136.6 ms |    1250.0000 |   500.0000 |  250.0000 |    42.38 MB |
|      MergeSortSequentially |      1000 | 1,394.6 ms |  21.19 ms |  17.69 ms | 1,386.3 ms |  278000.0000 |  4000.0000 | 4000.0000 |  1440.34 MB |
|     MergeSortParallelNaive |      1000 | 2,687.0 ms | 277.04 ms | 812.52 ms | 2,595.5 ms | 2451500.0000 |  4500.0000 | 1500.0000 | 10113.76 MB |
| MergeSortParallelThreshold |      1000 |   329.9 ms |  26.13 ms |  77.04 ms |   345.9 ms |  371666.6667 | 11333.3333 | 2000.0000 |  1863.39 MB |
|      QuickSortSequentially |     10000 |   766.9 ms |   2.65 ms |   2.22 ms |   766.5 ms |            - |          - |         - |    38.15 MB |
|     QuickSortParallelNaive |     10000 | 1,518.8 ms | 176.82 ms | 518.59 ms | 1,535.2 ms | 1049500.0000 |          - |         - |  4205.25 MB |
| QuickSortParallelThreshold |     10000 |   118.3 ms |   6.83 ms |  20.15 ms |   117.2 ms |            - |          - |         - |    38.98 MB |
|      MergeSortSequentially |     10000 | 1,364.6 ms |   5.46 ms |   4.84 ms | 1,364.5 ms |  278000.0000 |  4000.0000 | 4000.0000 |  1440.34 MB |
|     MergeSortParallelNaive |     10000 | 2,408.9 ms | 181.76 ms | 533.07 ms | 2,400.7 ms | 1433500.0000 |   500.0000 |         - |     5931 MB |
| MergeSortParallelThreshold |     10000 |   348.8 ms |  28.63 ms |  84.43 ms |   362.2 ms |  317000.0000 |  6000.0000 |  666.6667 |  1607.22 MB |
|      QuickSortSequentially |    100000 |   780.3 ms |   2.49 ms |   2.33 ms |   779.9 ms |            - |          - |         - |    38.15 MB |
|     QuickSortParallelNaive |    100000 | 1,589.8 ms | 109.23 ms | 322.06 ms | 1,626.1 ms |  618000.0000 |   333.3333 |         - |  2493.29 MB |
| QuickSortParallelThreshold |    100000 |   127.5 ms |   5.44 ms |  16.03 ms |   128.9 ms |     250.0000 |   250.0000 |  250.0000 |     38.2 MB |
|      MergeSortSequentially |    100000 | 1,390.1 ms |   8.44 ms |   7.05 ms | 1,388.6 ms |  278000.0000 |  4000.0000 | 4000.0000 |  1440.34 MB |
|     MergeSortParallelNaive |    100000 | 2,482.6 ms | 185.74 ms | 547.67 ms | 2,467.2 ms | 1602500.0000 |   500.0000 |         - |  6650.55 MB |
| MergeSortParallelThreshold |    100000 |   354.6 ms |  23.06 ms |  67.99 ms |   364.9 ms |  244666.6667 |  7000.0000 |  666.6667 |  1257.43 MB |
|      QuickSortSequentially |   1250000 |   771.8 ms |   1.21 ms |   1.01 ms |   771.9 ms |            - |          - |         - |    38.15 MB |
|     QuickSortParallelNaive |   1250000 | 1,587.2 ms | 160.84 ms | 474.25 ms | 1,579.0 ms |  882000.0000 |          - |         - |  3540.29 MB |
| QuickSortParallelThreshold |   1250000 |   116.6 ms |   6.12 ms |  17.95 ms |   111.5 ms |            - |          - |         - |    38.15 MB |
|      MergeSortSequentially |   1250000 | 1,378.2 ms |   4.35 ms |   3.63 ms | 1,377.2 ms |  278000.0000 |  4000.0000 | 4000.0000 |  1440.34 MB |
|     MergeSortParallelNaive |   1250000 | 2,477.6 ms | 207.34 ms | 611.36 ms | 2,456.0 ms | 1844500.0000 |   500.0000 |         - |  7622.15 MB |
| MergeSortParallelThreshold |   1250000 |   384.7 ms |  37.12 ms | 109.44 ms |   374.0 ms |  161000.0000 |  7500.0000 |         - |   872.21 MB |
|      QuickSortSequentially |   2500000 |   760.5 ms |   1.79 ms |   1.58 ms |   760.1 ms |            - |          - |         - |    38.15 MB |
|     QuickSortParallelNaive |   2500000 | 1,545.3 ms |  95.19 ms | 279.18 ms | 1,550.5 ms | 1118666.6667 |          - |         - |  4479.61 MB |
| QuickSortParallelThreshold |   2500000 |   123.7 ms |   2.47 ms |   6.77 ms |   122.6 ms |            - |          - |         - |    38.15 MB |
|      MergeSortSequentially |   2500000 | 1,362.5 ms |  13.73 ms |  11.46 ms | 1,358.3 ms |  278000.0000 |  4000.0000 | 4000.0000 |  1440.34 MB |
|     MergeSortParallelNaive |   2500000 | 2,419.5 ms | 222.15 ms | 655.01 ms | 2,426.6 ms | 1757000.0000 |          - |         - |  7267.95 MB |
| MergeSortParallelThreshold |   2500000 |   785.4 ms |  12.50 ms |  11.69 ms |   783.0 ms |  283000.0000 |  9000.0000 | 3000.0000 |  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 Methode bei den durchgeführten Tests.

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

Fiedorowicz Samuel
Nguyen Ngoc-Lam

Beim Quicksort gibt es kaum Performance Verbesserungen mit einem kleinerem Threshold.
Beim Mergesort gibt es leichte Performance Verbesserungen bei kleinerem Threshold. Die beste Performance gab es beim Threshold von 1.25M welches ungefähr 1/8 der Sortiergröße und der Anzahl der Threads entspricht.