**Multithreaded Merge Sort Analysis**

Kurt Aquino, Seaver Choi, Angelo Matias, Stanley Yu Galan

Proglan G02

**Processor Used:** Intel i7 4700MQ Overclocked

**Clock Speed:** 2.4ghz (3.6ghz max turbo boost)

**Cores:** 4 physical cores (8 virtual cores [hyper threaded])

Performance scales positively with every even number increase in thread count up until 8 threads which is the number of virtual cores available (4 physical, 8 virtual due to hyper threading). CPU usage maxes out at about 83% and does not increase beyond this point regardless of the number of threads probably due to the short maximum run time imposed by Java’s limited memory allocation, preventing the use of extremely large arrays as datasets as well as the memory bottleneck caused by the heavy data transfer requirements of merge sort that forces the CPU to idle at times.

Odd numbered thread counts, the extra odd core cannot be effectively used by our implementation of the merge sort algorithm, which works by splitting the values to be sorted into two halves and then recursively calling itself then merging the result, and thus does not contribute any performance advantages.

At higher thread counts above the number of available cores, performance drops. At very high thread counts (16384 threads), there is a dramatic degradation in performance with the algorithm taking 60x longer to complete. The time taken to create, manage and destroy the threads becomes significant relative to the actual time spent actually performing computations. CPU utilization also drops at extremely high thread counts indicating a memory bottleneck that leaves the CPU idle most of the time as portions of the dataset are divided among the various threads. Memory usage increases significantly with each additional functional thread but only increases slightly in performance with each additional non-value adding thread.

Overall merge sort performs best with a number of threads equal to the number of virtual cores.

**40,000,000 element array**

|  |  |  |  |
| --- | --- | --- | --- |
| Cores | Average Time (10 Runs) | Average CPU (10 Runs) | Average RAM (10 Runs) |
| 1 | 3.650278 Seconds | 16% | 1.68gb |
| 2 | 2.2015483 Seconds | 32% | 1.97gb |
| 3 | 2.261217 Seconds | 31% | 1.96gb |
| 4 | 1.7904396 Seconds | 51% | 2.17gb |
| 5 | 1.8107485 Seconds | 51% | 2.14gb |
| 6 | 1.7179224 Seconds | 51% | 2.16gb |
| 7 | 1.7741476 Seconds | 51% | 2.11gb |
| 8 | 1.6616504 Seconds | 79% | 2.21gb |
| 9 | 1.6746153 Seconds | 82% | 2.26gb |
| 10 | 1.6639631 Seconds | 80% | 2.23gb |
| 11 | 1.6930071 Seconds | 78% | 2.20gb |
| 12 | 1.68006 Seconds | 79% | 2.26gb |
| 16 | 1.8278773 Seconds | 82% | 2.19gb |
| 32 | 2.3751903 Seconds | 78% | 2.21gb |
| 64 | 2.138163 Seconds | 81% | 2.24gb |
| 128 | 2.207291 Seconds | 80% | 2.21gb |
| 256 | 2.5222995 Seconds | 79% | 2.23gb |
| 512 | 2.9422784 Seconds | 78% | 2.27gb |
| 1024 | 3.203072 Seconds | 77% | 2.28gb |
| 2048 | Crash (Out of Heap Space) | Crash (Out of Heap Space) | Crash (Out of Heap Space) |
| 4096 | Crash (Out of Heap Space) | Crash (Out of Heap Space) | Crash (Out of Heap Space) |
| 8192 | Crash (Out of Heap Space) | Crash (Out of Heap Space) | Crash (Out of Heap Space) |
| 16384 | Crash (Out of Heap Space) | Crash (Out of Heap Space) | Crash (Out of Heap Space) |

**10,000,000 element array**

|  |  |  |  |
| --- | --- | --- | --- |
| Cores | Average Time (10 Runs) | Average CPU (10 Runs) | Average RAM (10 Runs) |
| 1 | 0.8833845 Seconds | 13% | 1.17gb |
| 2 | 0.525851 Seconds | 26% | 1.34gb |
| 3 | 0.5293334 Seconds | 28% | 1.34gb |
| 4 | 0.43502203 Seconds | 51% | 1.47gb |
| 5 | 0.4253773 Seconds | 51% | 1.46gb |
| 6 | 0.42018014 Seconds | 51% | 1.50gb |
| 7 | 0.42017712 Seconds | 51% | 1.46gb |
| 8 | 0.39120698 Seconds | 78% | 1.55gb |
| 9 | 0.40565381 Seconds | 74% | 1.53gb |
| 10 | 0.39610952 Seconds | 75% | 1.58gb |
| 11 | 0.38246146 Seconds | 73% | 1.55gb |
| 12 | 0.40767226 Seconds | 72% | 1.59gb |
| 16 | 0.43313295 Seconds | 72% | 1.61gb |
| 32 | 0.41889405 Seconds | 78% | 1.71gb |
| 64 | 0.4267431 Seconds | 72% | 1.61gb |
| 128 | 0.43505478 Seconds | 73% | 1.69gb |
| 256 | 0.50482756 Seconds | 79% | 1.67gb |
| 512 | 0.51409763 Seconds | 78% | 1.73gb |
| 1024 | 0.58835685 Seconds | 72% | 1.70gb |
| 2048 | 0.82714844 Seconds | 63% | 1.83gb |
| 4096 | 1.6551281 Seconds | 45% | 1.88gb |
| 8192 | 5.4369125 Seconds | 27% | 2.07gb |
| 16384 | 23.340544 Seconds | 18% | 2.00gb |