An array of elements sorted using the “quicksort” method.

Using a multi threaded approach.

TTest:   
#nThreads #workload #timeS #speedup

1 8192 1.435 1.0

2 8192 0.721 1.9902912621359226

4 8192 0.473 3.0338266384778017

8 8192 0.353 4.065155807365439

16 8192 0.375 3.8266666666666667

32 8192 0.354 4.053672316384181  
  
Cores 4 Logical processors 8.

Single threaded quicksort:

Array of 1000000 ir took 0.177 seconds

Array of 5000000 time it took 1.53 seconds  
Array of 10000000 time it took 6.5 seconds

Multithreaded quicksort (2 threads):

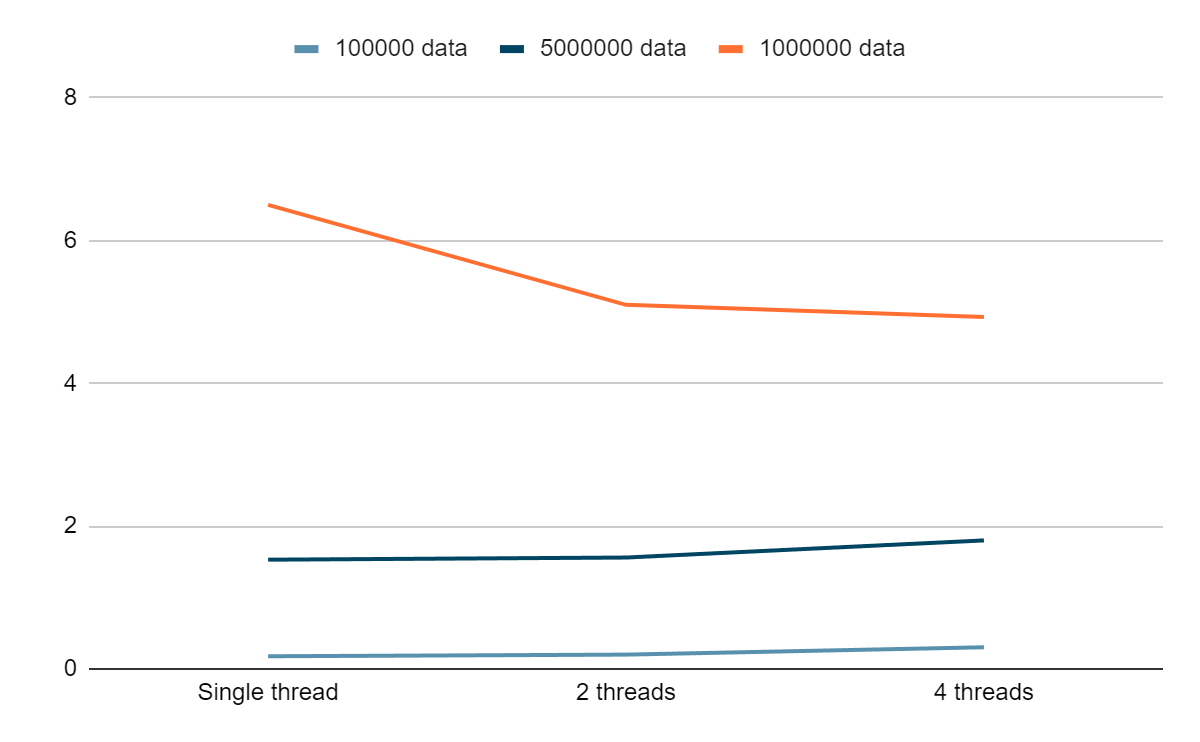
Array of 1000000 ir took 0.20 seconds

Array of 5000000 time it took 1.56 seconds  
Array of 10000000 time it took 5.1 seconds

Multithreaded quicksort (4 threads):

Array of 1000000 ir took 0.303 seconds

Array of 5000000 time it took 1.8 seconds  
Array of 10000000 time it took 4.93 seconds

  
  
How it works:   
  
First we check if we are not on the lowest depth allowed by the parameters depth 2 for 4 threads and depth 3 for 8.

Then we partition the array into two separate pieces. This is done by picking a random integer in the array and then comparing everything in the array to it and putting the pivot in between. Then we split the array and repeat for both pieces. Till we have only partitions where there are 0 or 1 element. And then we recombine them into a nice sorted array.  
  
Problems:

It is quite unreliable in its speed as it very much depends on where you pick the pivot. Its average time complexity is O(n\*log(n)) however the worst case is still O(n^2) where we pick pivots at the edges like the highest or the lowest. This is especially apparent in the multithreaded quicksort as if we pick the middle it works sometimes twice as fast instead of barely faster if we pick a value closer to the edge.