Charles Haithcock

1 March 2018

CSC 591-604, Hung-Wei Tseng

Homework 1

	Mode 0, Speedup		Mode 1, Speedup	
Run 1	16191837	0	16443246	0.9984
Run 2	16198219	0	16419580	0.9865
Run 3	16200626	0	16417500	0.9867

The threaded mode 1 has slightly worse times than the qsort() mode 0. The complexity of the code is inherently $O(nlog(n) + \frac{n}{2}log(\frac{n}{2}))$ which is thus dominated by the nlog(n). The implementation divides the problem in half and each of the two threads mergesorts the halves. Once complete, a final merge is done on the entire dataset, as, upon completion, the data consists of two sorted subarrays. The final merge provides the cost of nlog(n) while each of the threads takes on half the input and thus $\frac{n}{2}log(\frac{n}{2})$. As the work is divided into two processes for each processor to work on, the complexity of the threads is not $nlog(\frac{n}{2})$. The complexity is difficult to compare with the C qsort function as there is no clear description within the standard C library ISO or otherwise defining the complexity.

Considering my implementation of parallelized mergesort performs almost as well as qsort provided from the standard C library, improvements should be made. For example, given time, the temporary storage buffer for subarrays could simply be switched off with the real data input while sorting rather than memcpy() every time a subarray was sorted. Additional threads were attempted (up to 6 were attempted) but the performance was best at 2 threads. I had no board mates so no way to compare the run times.