Analysis of Parallel Merge Sort Algorithm

Ben Lerchin — 24 April 2013

A wall clock study of my test program confirms that when run sequentially (limited to one thread) it falls within the expected range of performance for quick sort. Rough analysis of behavior on a single thread and a range of problem sizes demonstrates a growth rate less than n^2, but above nlog(n). See Figure 1.

Subsequent tests taking advantage of the parallelized algorithm show sizeable performance gains between 0 and 8 threads (See Figure 2). Averaging the results across problem sizes between 2000 and 200,000, a parallel sort operation running on two threads took 27.5% as much time to complete as a single process sort. Split between four threads, the sort took 11.1% the time of a sequential sort; divided amongst 8 threads the sort was completed in 4.7% the time. Beyond this threshold (the number of processors on the test machine) gains are small and eventually become losses as overhead from spawning multiple processes overshoots any extra efficiency the scheduler can provide. This is visible in the timing results where d=6 (64 threads). At this point, a sort operation took on average 10% longer than the sort operation where d=5 (32 threads).

My calculations of alpha yield unreasonable values , some of which are greater than one, which should be theoretically impossible (on average, α=1.16). This suggests either the data are flawed, my calculations are in error, or that somehow my implementation breaks Ahmdahl's law (the first two seem more likely). My calculated results do seem to satisfy Ahmdahl's equation, even if the values are unreasonable, so I suspect the data are at fault here. Further experimentation would be required to figure out where the issue lies, or if I have indeed created an algorithm that is 116% parallelizable.

Calculated percent of alogorithm that is parallelizable:

threads:

1	2	3	avg.
1.43%	1.081%	.995%	1.16%

Figure 1: Time to Complete Sequential Sort Operation

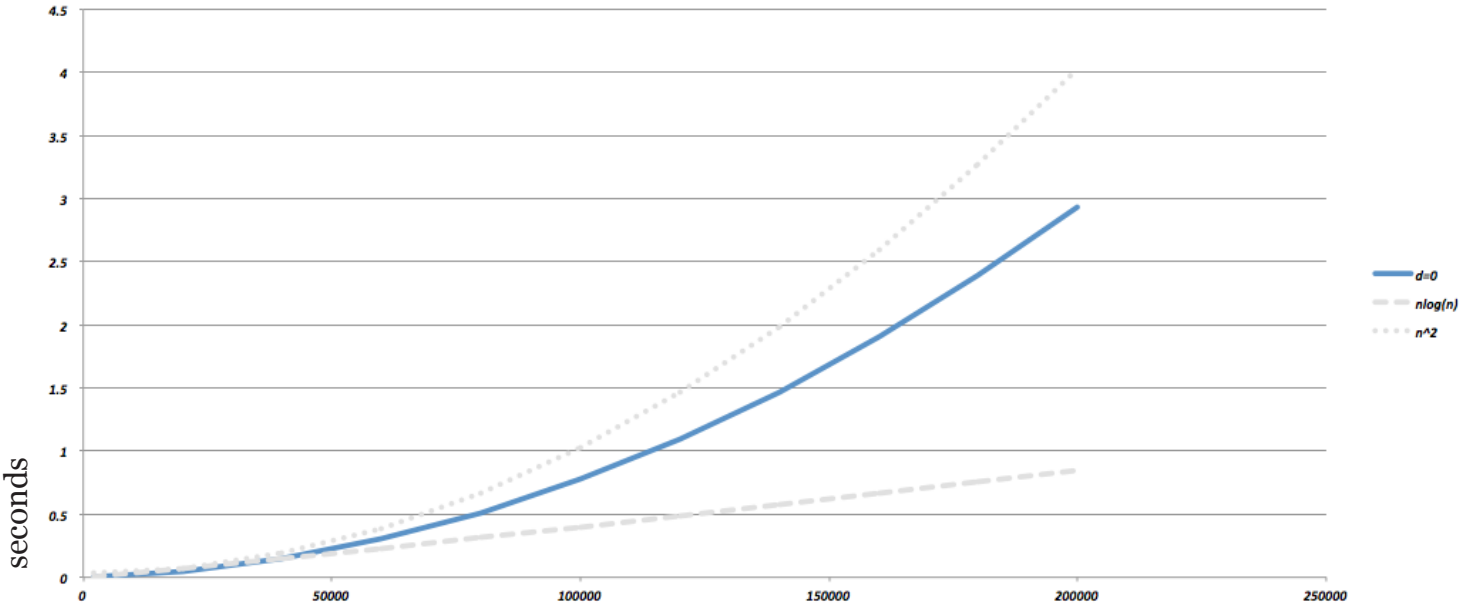

Figure 2: Time to Complete Parallelized Sort Operation