ssh

95
Most significant comments: adequate optimizations applied. Theoretical analysis included, work/span correct, but T_p discussion incorrect. Detailed comments below.

1 Problem size is never specified. “Effective” and “Ineffective” should really be “Efficient” and “Inefficient”.

2 (cache) Fixed row/column problem. No cache optimizations attempted for memory efficient algorithms.

3 (openmp) Used parallel for directives in all 3 algorithms. Parallelized convert-path-to-used.

4 (parallel results) 5x/19x/10x speedups on unspecified problem size using 32 threads. Nice strong scaling experiment, but because all algorithms are normalized to 1 using 1 thread, it’s not clear their timings relative to each other. One way to fix this is to plot ideal speedup lines for each algorithm (maybe dotted) and use absolute times rather than relative ones. Another way to improve speedup plot is to use log scale on y-axis to match log scale on x-axis. Then ideal speedup curve will be straight line, and you can see difference for lower numbers of threads as well.

5 (conclusions) Analysis doesn’t really make sense though I think your work and spans are accurate. On PRAM, T_p = T_1/p+T_infty = nW/p + n log W. I think your hypothesis on the better speedups for memory efficient code is probably right; smaller memory footprint means less contention for bandwidth to memory across threads.