teamy-mcteamface

90
Most significant comments: parallel results can be more detailed than just showing a lack of scaling after 10 threads; analysis could be stronger by computing actual work and span for this problem. Detailed comments below.

1 (inputs) Chose capacity 30,000 and numitems 5,000.

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

3 (openmp) Used parallel for directives in all 3 algorithms. Did not parallelize backtrack_implicit.

4 (parallel results) Strong scaling experiment with chosen input values. Scaling stops around 10 threads. Would be good to see speedup numbers or at least report absolute times in a table so they could be computed. General analysis of work/p+span discussed, but I would argue that that estimation does not explain performance numbers (span should be negligible for these numbers). Would be nice to compute the work and span for this problem to discuss this.

5 (conclusions) I don't agree with the statement "the recursive call used to find the 'midpoint' of the table is not parallelizable". The recursive calls work on separate parts to the table and can actually use the same vectors without write conflicts. They can be parallelized with task and taskwait pragmas, but the difficulty is in the nested parallelism and calls to the findk function. What other optimizations might you try?