ziira

95
Most significant comments: good job trying out cache optimization on memory-efficient code; analysis of parallel results could be much stronger. Detailed comments below.

1 (inputs) Found different values for each algorithm to be 10 seconds; generally choosing capacity >> numitems.

2 (cache) Fixed row/column problem. Reported up to 2x speedup for memory inefficient alg. Tried switching to row ordering for memory efficient algs, saw 3% improvement.

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

4 (parallel results) Strong scaling experiment with 1M capacity and 300 items. Reported speedups of 8x/30x/24x on 44 threads. Nice speedup plot with absolute times given in table. Did not discuss results for other input values (would expect to see smaller speedup for many more items and smaller capacity).

5 (conclusions) Not much analysis or conclusions. What else might you try? Why do you think the parallel speedup was less for the memory inefficient code?

Performance on 5000/5000 input:
Your sequential memory-efficient solution+value implementation was the fastest in the class (thought not by much).
Your parallel memory-efficient solution+value implementation was the fastest in the class (thought not by much).
*I think this is because of your cache optimization. It apparently slowed down your other memory-efficient implementations, also not by much.