Parallel Sudoku

Group 24

Benchmarks

Sequential solving

Full execution: 60630185 wildcat: 0.39699 diabolical: 48.43309 vegard_hanssen: 106.55684 challenge: 7.47393 challenge1: 397.17806 extreme: 9.65853 seventeen: 36.60404

Parallel solving of each puzzle

Full execution: 38781504 wildcat: 0.57841 diabolical: 71.76407 vegard_hanssen: 134.24873000000002 challenge: 11.28142 challenge1: 387.77229 extreme: 32.45287 seventeen: 65.74575

Parallel solving of first guesses in parallel

Full execution: 42858542 wildcat: 0.29794 diabolical: 18.948610000000002 vegard_hanssen: 44.39331 challenge: 3.78094 challenge1: 313.23831 extreme: 16.76906 seventeen: 31.13776

Parallelising tactic

We use a worker pool and try to distribute the work out on it as effitiently as possible.

We first implemented a version parallelising the seperate puzzlesolves. This would of course only ever give you as much of a speedup as the slowest puzzle. However they still gave an ok speedup.

Next, we tried to implement a parallel version of the refinement, but even though this did solve it (so there were no huge bugs) it was so slow when running 100 executions like the others that we never even finished the benchmark. We think this was because it splits up the work too much, making communicating and moving data the bottleneck. We should probably have made it so that you tell a each worker to refine a whole matrix instead of just a row.

After that we did a version parallelising solving each separate guess. As soon as any worker returned a solution of a guess we stopped trying to search for further solutions and just used that one. This speed up solving all puzzles considerably, but the full execution actually end up slower than solving each puzzle in parallel. We think we should have let further guesses in each parallel guess also be run in parallel to make it run faster.

Execution

1. Compile “pool.erl”.
2. Compile “par_bench.erl” or “par_guess.erl”.
3. Run the “benchmarks()” function that exists inside both of them.