Lab C: “Parallelizing a Sudoku Solver”

Parallel Functional Programming Group 13

Anton Bengtson

Deepak Kumar Singh

Task: Running Benchmark in Parallel

We created a new function parallel_benchmarks() to spawn new processes in-order to solve different puzzles in parallel. We used Intel(R) Core(TM) i7-6820HQ CPU @ 2.70 GHz in order to run the program. Please refer the following result time stamps for sequential run and parallel run:

```
Serial benchmarks:

> sudoku:benchmarks().

{92494776,

 [{wildcat,0.78},

  {diabolical,78.78061},

  {vegard_hanssen,138.06263},

  {challenge,12.16023},

  {challenge1,628.5424499999999},

  {extreme,16.38044},

  {seventeen,50.24137}]}


Parallel Benchmarks:

======================

> sudoku:benchmarks().

{55771525,

 [{wildcat,3.28009},

  {diabolical,62.87171},

  {vegard_hanssen,110.14300999999999},

  {challenge,20.91057},

  {challenge1,557.71525},

  {extreme,25.590700000000002},

  {seventeen,52.42143}]}
```

From the above results, it seemed parallel run has improved the overall time by 38 seconds, well that was some improvement in performance. Another interesting noticeable point was, some of the puzzles for ex: extreme or wildcat, took bit more time in parallelized version than the sequential one.

Task: Parallelizing Solver, Subtask1:

Refining rows in parallel: In the provided Sudoku Solver, there's a function named refine_rows() which in-turn makes calls for refine_row() for every row of the matrix. We implemented a new function named parallel_refine_rows() which creates a reference list for all the rows and spawns processes to refine_row in parallel. Let's have a look at benchmarking results:

```
Serial benchmarks:
==================
> sudoku:benchmarks().
{89887000,
 [{wildcat,0.78},
  {diabolical,63.96},
  {vegard_hanssen,182.36},
  {challenge,14.35},
  {challenge1,566.29},
  {extreme,12.63},
  {seventeen,58.5}]}

Parallel benchmarks (Parallel refining of Rows):
================================================
> sudoku:benchmarks().
{88702000,
 [{wildcat,1.25},
  {diabolical,51.79},
  {vegard_hanssen,146.79},
  {challenge,15.29},
  {challenge1,584.23},
  {extreme,18.87},
  {seventeen,68.8}]}
```

As you see from the above time stamps, there is slight improvement (1.5 sec) in parallelized run when compared to sequential run total time. However, individual runtime for different puzzles has increased.

Task: Parallelizing Solver, Subtask2:

Guessing in parallel: Here also, we followed the similar approach as of in previous subtask of running refine_row in parallel for all the rows of a matrix. We implemented a new function named parallel_guesses which spawns a new process for every guess in guess list with a reference. Lets have a look of timestamp results for this parallel function:

```
Serial benchmarks:

==================

> sudoku:benchmarks().

{89887000,

 [{wildcat,0.78},

  {diabolical,63.96},

  {vegard_hanssen,182.36},

  {challenge,14.35},

  {challenge1,566.29},

  {extreme,12.63},

  {seventeen,58.5}]}


Parallel benchmarks (Parallel guessing):

================================================

> sudoku:benchmarks().

{171226119,

 [{wildcat,0.94},

  {diabolical,124.9498},

  {vegard_hanssen,305.60790999999995},

  {challenge,17.46988},

  {challenge1,1118.67443},

  {extreme,29.329819999999998},

  {seventeen,115.28932}]}
```

The parallel run results doesn’t look like an improvement.

Task: Parallelizing Solver using worker pool technique:

As suggested in the review comments in fire, we implemented a pool of workers which try solving refined matrix in parallel. When a worker is required to guess and busy, it will ask other worker to do the next task. If all workers are busy, it will keep the task to itself. After implementing worker pool function, we got the best result so far:

```
Parallel benchmarks (worker pool mechanism):
==============================================
25> sudoku:benchmarks().
{43055828,
 [{wildcat,1.09},
  {diabolical,46.33036},
  {vegard_hanssen,60.69047},
  {challenge,6.55004},
  {challenge1,166.4487},
  {extreme,37.59966},
  {seventeen,111.84900999999999}]}
26>
```