Contents

1 Overview

The codebase `game-of-life`, which can be found at `https://github.com/dgsaf/game-of-life`, consists of the original code provided by Dr Pascal Elahi, with the following additions:

- `src/02_gol_cpu_serial_fort.f90`: a serial GOL code which derives from `src/01_gol_cpu_serial_fort.f90`, but improves loop ordering to match the column-major format of Fortran.
- `src/02_gol_cpu_openmp_loop_fort.f90`: a parallel GOL code which derives from `src/02_gol_cpu_serial_fort.f90`, but implements OMP parallel do loops to yield performance benefits.
- `profiling/`: a directory which includes the profiling results of the `01_gol_cpu_serial_fort` and `02_gol_cpu_serial_fort` versions of the GOL code, as well as a brief summary and comparison of the profiling results. The results were collected for a GOL simulation on a 1000×1000 grid, for 100 steps with no visualisation enabled.
- `gol-job-submission.slurm`: a bash script which submits a `sbatch` job request for a GOL simulation with a given set of parameters which include:
  - `version_name`
  - `n_omp`
  - `grid_height`
  - `grid_width`
  - `num_steps`
  - `intial_conditions_type`
  - `visualisation_type`
  - `rule_type`
  - `neighbour_type`
  - `boundary_type`

  An output directory is created for the given set of parameters, with the logging output and statistics of the GOL simulation confined there. If the output directory already exists, the job isn't submitted to prevent repeating work needlessly.
- `gol-job-set-submission.sh`: a bash script which constructs different sets of parameters, and executes `gol-job-submission.slurm` for each parameter set. Three batches of jobs are submitted:
  - A verification batch, which submits a job for every version of the GOL simulation, on a 10×10 grid, for 10 steps, with ASCII visualisation. This is intended to allow for visual confirmation that each version produces uniform results. The logging output and statistics are compared to verify this.

  - A scaling batch, which submits a job for every version of the GOL simulation, on a range of grid sizes, $2^n \times 2^n$ for $n = 1, \ldots, 14$, for 100 steps, with no visualisation. This is intended to collect data for analysing the scaling behaviour of each version with increasing grid size, with the total elapsed time being compared.
  - An OMP batch, which submits a job for every parallel version of the GOL simulation, for a range of assigned threads, $n_{omp} = 1, \ldots, 16$, on a 10×10 grid, for 10 steps, with ASCII visualisation. This is intended to allow for visual confirmation that each parallel version produces uniform results, independent of the number of threads assigned to the program. The logging output and statistics are compared to verify this.

- `output/`: a directory which includes subdirectories (for each parameter set submitted), each of which include the logging output and statistics file; that is,
  `output/<unique_parameter_set>/log.txt` and
  `output/<unique_parameter_set>/stats.txt`.

- `report/`: a directory which includes this `.tex` file and other files suitable for submission of this assignment.

Additionally, some minor modifications have been made to the following files:

- `Makefile`: the make rule `make cpu_serial_fort` has been modified to include `src/02_gol_cpu_serial_fort.f90`.

- `src/common_fort.f90`: the length of the variable `arg` has been increased from 32 to 2000 to allow for larger filenames for the variable `statsfile`.

- `src/01_gol_cpu_serial_fort.f90`: a bug in the `game_of_life_stats()` subroutine has been

2 Code Synopsis

Here, we will provide a synopsis of the most significant changes and additions to the codebase. Small, but important, sections of code will be presented and discussed; the codebase in its entirety is presented in Appendix A.

2.1 Bash Scripts

2.2 Serial Code

```
126     integer, dimension(0:NUMSTATES-1) :: num_in_state
127     real*4, dimension(0:NUMSTATES-1) :: frac
```

```
150     do i = 0, NUMSTATES-1
151         write(10,fmt, advance="no") "Frac in state ", i, " = ", frac(i), " "
152     end do
```

2.2.1 Original Serial Code

2.2.2 Improved Serial Code

2.2.3 Profiling Comparison

2.3 Parallel Loop Code

3 Results

3.1 Uniform Behaviour Verification

3.2 Scaling Behaviour

3.3 Thread Independence Verification

A Appendix