Programmierung paralleler Rechnerarchitekturen, Winter 2014/15

Exercise 6

Task 6.1 (Conways Game of Life - bigger, faster, better)**.** The *Game of Life* initially conceived by John Conway is a common example for a *cellular automaton*.
For this "game" an rectangular area is partitioned in square cells. Each cell has eight neighboring cells (north-east, north, north-west, west ...). Each cell is either alive or dead. The state of a cell in the next turn depends solely on the number of living neighboring cells of this turn, according to the following rules:

- A cell "dies" if it has less than 2 or more than three neighbors.
- A cell stays "alive" if it has exactly 2 neighbors.
- A cell becomes "alive" if it has three neighbors.

More information about the *Game of Life* is available on the internet, e.g. wikipedia `http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`.
Hint: This cellular automaton approach is used in certain areas for solving real problems. An example is the Lattice Boltzmann method (compare `http://en.wikipedia.org/wiki/Lattice_Boltzmann_methods` for more details.
In Moodle you will find a file called "Exercise_06.zip". It contains a simple (and slow) serial implementation of the Game of Life.

a) Analyze and understand the serial version of the provided Game of Life.

b) Parallelize the serial version of the Game of Life using MPI. Use whatever MPI functions that you consider appropriate.

c) Investigate the used data structures. Create Custom-MPI data types for the different Halo (Ghost-Cell) regions.
Hint: It may be beneficial, if there would be a separate type for the top and bottom, left and right and for the corner cases.

d) Add an additional OpenMP parallelization to your MPI implementation.

e) Perform several benchmarks for your hybrid version. Use different Process / Thread Counts. What can you observe?

Hint: You may perform whatever steps you deem necessary to achieve a well running parallel version of the Game of Life code. Note, that you should not modify the input-output format.
Hint: You may use a so called glider for testing:

Please note: The border of the game shall, according to the global variable 'BOUNDRY', consist of

a) permanently alive cells (BOUNDRY=BOUNDRY_ALIVE),

b) permanent dead cells (BOUNDRY=BOUNDRY_DEAD)

c) or a periodic copy of the other sides game-border (BOUNDRY=BOUNDRY_PERIODIC).

This shall be determined at compile-time.

The code expects a command line in the form of

```
gameOfLife N M InputFile.cgolf NumSteps NumOutput Output1 ... Output(n-1)
```

Here N and M denote the dimensions of the game field, InputFile.cgolf the compressed inputfile, NumSteps the number of steps and NumOutputSteps the number of snapshots to be taken. A final output is to be generated at the end of the game.
On MICE and PEACOCK there are additional initial configurations for your tests in the `/opt/lehre/Exe06` directory.