PPAR: GPU lab assignment 2

Caroline Collange and Antonio Mucherino

March 2021

The goal of this lab assignment is to implement a 2-player version of Conway's Game of Life [1].

We consider a 2-dimensional domain composed of `domain_x` × `domain_y` cells. Each cell can be either red, blue, or empty. The domain has a torus shape: the right neighbor of the rightmost cell is the leftmost cell.

Each cell has 8 neighbors in the adjacent cells. At each time step, all cells evolve according to the following rules:

- a cell that has strictly less than 2 alive neighbors among the 8 adjacent cells dies (becomes empty),
- a cell that has strictly more than 3 alive neighbors dies too,
- a live cell that has 2 or 3 alive neighbors survives,
- an empty cell that has exactly 3 neighbors becomes occupied. Its color will be selected from the majority of its neighbors (i.e. if 2 or more neighbor cells are blue, the new cell is blue, otherwise it is red).

All cells are updated synchronously.

As in the previous lab, you can log in to parawell using:

```
ssh -i pparXX_id_rsa pparXX@parawell.irisa.fr
```

where `pparXX` is your login and `pparXX_id_rsa` is the complete path to your key. Once logged on your account on parawell, you will start from the template in: `gpu_lab2`.

To avoid race conditions, we follow a *ping-pong* approach: we maintain two copies of the domain. At each time step, we read from one copy and write to the other one, then exchange the pointers to the read domain and written domain.

For now, we represent each cell of the domain by an integer. 0 means empty cell, 1 means red, 2 means blue.

1. Program the simulation without optimizing memory accesses. We use the `read_cell` function to access neighbors.

1 Optimizing memory accesses

2. How many read memory accesses to global memory does each thread perform? How many read accesses per block of threads does this make?

We now want to reduce the number of global memory accesses. We will do so by sharing data between threads, within each block of threads.

3. Consider one block of threads. How many global memory locations are read by a least one thread of the block? Unlike in the prior question, locations that are accessed by multiple threads of the block are only counted once. *Hint: make a drawing*

4. Same question if blocks are 2-dimensional. Which block shape would minimize the number of unique locations read?

5. Modify the code to avoid the redundant reads to global memory by using shared memory.

References

[1] `http://en.wikipedia.org/wiki/Conway's_Game_of_Life`