PPAR: GPU lab 3 and 4

Sylvain Collange and Antonio Mucherino

March-April 2017

The objective 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` $\times$ `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,
- a cell that has strictly more than 3 alive neighbors dies too,
- a 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.

You can log in to paramax 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 paramax, you will start from the template in: `gpu_lab3`.

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 reads are performed for each cell? Are they coalesced reads?
3. Consider one thread block. What is the set of all locations that are read by a least one thread of the block?
4. Same question if blocks are 2-dimensional. Which block shape would minimize the number of unique locations read?
5. Use shared memory to remove duplicate reads.

2 Optimizing computations and datatypes

6. Where is there control flow divergence in your kernel? Can you reduce divergence?
7. Edit the CUDA compiler options to enable Verbose PTXAS output (`-Xptxas=-v` option to `nvcc`). The compiler will now print the number of registers/thread used in the compiler output window. How many registers do you get?

8. Instead of having one thread per cell, have each thread compute 16 cells. What is the runtime and register usage?

The values stored in cells can only be 0, 1 or 2. Using 32-bit ints to store them is a waste of memory capacity and bandwidth. However, our GPU cannot access words smaller than 32-bit in coalesced mode.
Each thread will still read 32-bit words from memory, but several cells will be packed in each word.

9. *Bonus.* Program the above data packing optimization and compare the run times. Now how many reads are performed for each cell?

References

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