PPAR: GPU lab assignment 1

Caroline Collange and Antonio Mucherino

March 2021

Connection information

You will receive a key named `pparXX_id_rsa`. It corresponds to your account `pparXX` on `parawell.irisa.fr`, where *XX* is the unique number assigned to you. First make sure the access rights to the key are correctly set by running:

```
chmod 600 pparXX_id_rsa
```

where `pparXX_id_rsa` is the complete path to the key.
To log in, enter

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

likewise, replacing `pparXX_id_rsa` with your own key.

These instructions should work on Linux, MacOSX, or within a Unix environment under Windows. Under Windows, you may alternatively use the PuTTY SSH client.

Once logged on your account on parawell, you will start from the template in: `gpu_lab0`.

Returning the assignment

- This assignment is due **Friday, April 17**.
- For the code itself, just modify the files provided in `gpu_lab0` on your personal directory on `parawell`.
- You will write the answers to the questions on a text file (or PDF file) that you will create in the same directory. If you prefer to edit the document on your local machine, you can use `scp` (or `WinSCP` on Windows) to transfer the file to `parawell`

  ```
  scp -i pparXX_id_rsa  my_file.txt  pparXX@parawell.irisa.fr:gpu_lab0/
  ```

Subject

In this lab assignment, we want to compute the following series:

$$\frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} \ldots,$$

or equivalently:

$$\sum_{i=0}^{n-1} \frac{(-1)^i}{i+1}$$

for large n.

Parallelization

1. Program this computation in sequential C in function `log2_series`.
Hints: In C, computing $(-1)^i$ can be done by testing whether the value of `i % 2` is 0 or 1. Make sure the division is done in floating-point.

2. Compare the result obtained when summing the terms by increasing order of indices (from 1 to n) and by decreasing order (from n back to 1). How do you explain your observations?

We will now offload this computation to the GPU. We could implement it with a reduction using n threads, but it would require too much memory. Instead, each thread will compute multiple terms of the series.

3. We want to split the computation to parallelize it on m threads ($m < n$). Give two possible strategies to split the n elements equally between the m threads.

For now, we return one result per thread, and finish the computation on the CPU.

4. Implement memory allocation, copy, and de-allocation for the results.

5. Program the computation in `summation_kernel`. Perform the final computation on the CPU.

6. Compare the result with the CPU-only versions and explain possible differences. Implement the other solution from question 3 and compare its result.

7. Compare the run-times obtained for different number of threads per block and number of blocks. What is the best block size and grid size for your GPU?

Reduction

8. We want to reduce the amount of data transferred back. How to adapt the summation algorithm to return one value per block? Implement it.

9. Perform the whole computation on the GPU.