Performance enhancement using CUDA in a simulation of heat diffusion

Gonçalo Lourenço
nº55780
gm.lourenco@campus.fct.unl.pt

Joana Faria
nº55754
js.faria@campus.fct.unl.pt

I. Introduction

This assignment aims to optimize a base code, written in `C`, that computes a simulation for heat diffusion. For the optimization, we seek to take advantage of GPU programming using `CUDA`, and explore several alternatives to find the most efficient one.

To find the best performance we will explore different approaches, namely analyzing different configurations, exploring the impact of using shared memory, and the difference between using streams and not.

Knowing that the architecture of the system influences the performance obtained, the first step of our work is to understand the architecture of the system where our program will run. So we present the characteristics below:

```
Device: NVIDIA GeForce GTX 1050 Ti
  CUDA Driver Version: 11.4
  Runtime Version: 11.2
  CUDA Capability Major/Minor version number: 6.1
  6 Multiprocessors (SM)
  Max Grid sizes: 2 147 483 647, 65 535, 65 535
  Max Block sizes: 1024, 1024, 64
  Max Blocks per SM: 32
  Max threads per Block: 1024
  Warp size: 32
  Total global memory: 4 137 024 MB
  Shared Mem per SM: 98 304
  Max ShMem per Block: 49 152
```

II. Results

To accurately compare the different versions all the tests were performed in the university cluster and averaged the execution times over ten execution. The parameters chosen were:

```
const int nx = 200;// Width of the area
const int ny = 200;// Height of the area
const float a = 0.5;// Diffusion constant
const float h = 0.005;// h=dx=dy  grid spacing
const int numSteps = 100000;// Number of time
    steps to simulate (time=numSteps*dt)
const int outputEvery = 100000;// How frequently
     to write output image
```

A. *Sequential Verion*

We start by analyzing and profiling the sequential version to understand what improvements can be done, and what parts are the more problematic.

By a quick analysis of the code, we expect the cycle present in the `main` function to be the biggest problem. This assumption is supported by the reports of the profiling tools. The output of these tools can be seen in the files `results/sequential/gprof` and `results/sequential/perf`.

The average execution time obtained with this version is 153.979 seconds.