Explicit_Lab

December 6, 2023

Massive parallel programming on GPUs and applications, by Lokman ABBAS TURKI

1 6 Explicit simulation scheme for Black and Scholes Partial Differential Equation

1.1 6.1 Objective

This is the first lab of a series of four labs dedicated to the simulation of Parabolic Partial Differential equations using discretization schemes. We start with the simplest situation where we use the explicit scheme with global memory. Even with this simple example, multiple optimizations can be considered to reduce the execution time. These optimizations will be implemented step by step starting with the naive implementation of PDE_diff_k1 much improved in PDE_diff_k2 which is also improved in PDE_diff_k3. In addition to kernels, students need only to comment or to uncomment three lines in the wrapper function + CPU2GPU end GPU2CPU memory transfer. Neither the main function nor the NP function should be modified.

As usual, do not forget to use CUDA documentation, especially:

1) the specifications of CUDA API functions within the CUDA_Runtime_API.
2) the examples of how to use the CUDA API functions in CUDA_C_Programming_Guide

1.2 6.2 Content

Compile PDE.cu using

```
[ ]: !nvcc PDE.cu -o PDE
```

Execute PDE using (on Microsoft Windows OS ./ is not needed)

```
[ ]: !./PDE
```

As long as you did not include any additional instruction in the file PDE.cu, the execution above is supposed to return incorrect values on the left column. The right column is supposed to contain the true results that we should approximate with the discretization scheme.

1.2.1 6.2.1 PDE_diff_k1 and memory copy

In a lecture video and slides, we showed the implementation of the explicit scheme on the host. PDE_diff_k1 will be the kernel that executes one step of the time induction on the device. The time loop has to stay on the host calling N times the kernel PDE_diff_k1.

a) Justify the allocation of 2*sizeof(MyTab) for the array on the device.

b) Write the necessary code for CPU2GPU and GPU2CPU memory copy.

c) Inspired by the host solution from slides, complete the syntax of the kernel PDE_diff_k1.

1.2.2 6.2.2 PDE_diff_k2

In this optimization step, we put the time for loop in the kernel PDE_diff_k2.

a) Complete the syntax of the kernel PDE_diff_k2.

b) Compare the execution time of the solution involving PDE_diff_k2 to the one involving PDE_diff_k1 using !nvprof ./PDE.

1.2.3 6.2.3 PDE_diff_k3

In this optimization step, we propose to remove the if statement in the kernel PDE_diff_k3 and impose the limit conditions differently.

a) Complete the syntax of the kernel PDE_diff_k3.

b) Compare the execution time of the solution involving PDE_diff_k3 to the one involving PDE_diff_k2 using !nvprof ./PDE.

[ ]:

```

```