Glob_Reg_Lab

December 6, 2023

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

1 5 Normal cumulative distribution function, using global and local memory allocation, and registers

1.1 5.1 Objective

The main purpose of this lab is to introduce the student to the use of different memory spaces available on the device. The example presented here is an approximation of the Normal cumulative distribution function that involves the value of some parameters. We consider storing these parameters in global memory, local memory, and registers. Keep in mind that this is only an exercise in which we show how to use each type of memory space. Indeed, from the beginning of this lab, we already know that the best option for this function is to use registers!

The presented example will be continued, in another session, with the use of other memory spaces on the device.

As usual, I urge students to open 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 5.2 Content

Compile NP.cu using

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

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

[ ]:
```
!./NP
```

As long as you did not include any additional instruction in the file NP.cu, the execution above is not supposed to return any value. At least, no compilation error is detected by the compiler!

In the following questions you will need to include your own code in the file NP.cu, then compile it and execute it. In the following sections, we assume that the device is a pre-Hopper GPU (otherwise add -arch=compute_80 -code=sm_Yy, when Yy>=90).

1.2.1 5.2.1 Fewer values in global memory using synchronization among threads

For kernel NP_GLOwSync_k, we want to keep using global memory Glob as in NP_GLO_k but with fewer values that are shared by threads within the same block.

a) How should you set the values of Glob within the kernel NP_GLOwSync_k?

b) Complete the syntax of the kernel NP_GLOwSync_k.

c) Explain why the execution time decreased when compared to NP_GLO_k.

1.2.2 5.2.2 Using local memory

For kernel NP_LOC_k, we replace the use of the global memory Glob with local memory array allocation associated with each thread.

a) Is the local memory a real memory space on the device?

b) Complete the syntax of the kernel NP_LOC_k.

c) Compare the execution time of NP_LOC_k to NP_GLOwSync_k and to NP_GLO_k using !nvprof ./NP.

1.2.3 5.2.3 Using registers

For kernel NP_REG_k, we replace the use of the local memory array specific to each thread with registers also associated with each thread.

a) Could we define arrays within registers?

b) Complete the syntax of the kernel NP_REG_k.

c) Compare the execution time of NP_REG_k to NP_LOC_k and to NP_GLOwSync_k using !nvprof ./NP.

[ ]: