CSCI-GA.3033-023 – Lab2

This lab is intended to be performed **individually**, great care will be taken in verifying that students are authors of their own submission. Coding exercises are identified by **C<number>**.

C1 – Convolution in CUDA

In this part of the lab we implement a convolution of an image using a set of filters. Consider the following:

- An input tensor *I* with dimensions: C*, H, W.* Each element of *I* is generated as follows:

$$I[c,x,y] = c \cdot (x+y)$$

- A set of convolution filters with dimensions: *K, C, FH*, *FW.* Each element of the filter *F* is generated as follows:

$$F[k,c,i,j] = (c+k) \cdot (i+j)$$

- Dimensions are: H=1024, W=1024, C=3, FW=3, FH=3, K=5
- All tensors have *double* elements (double precision)
- A tensor I_0 of sizes C, W+2P, H+2P where P=1. I_0 is obtained from the tensor I adding the padding rows and columns with the elements set to zero.
- The output tensor *O* with dimensions: *K, W, H.* Each pixel of the output tensor $O[k,x,y]$ is obtained as:

$$O[k,x,y] = \sum_{c=0}^{C-1} \sum_{j=0}^{FH-1} \sum_{i=0}^{FW-1} F[k,c,FW-1-i,FH-1-j] \cdot I_0[c,x+i,y+j]$$

Note that we need to use the transpose of the filter in order to compute a convolution rather than a cross-correlation. Implement a simple convolution algorithm in CUDA without tiling and without shared memory.

- Print the checksum as the total sum of the elements of O along all its dimensions
- The checksum is expected to be: `1060238529900.00`
- Report the time to execute the CUDA kernel with the convolution, without including the time to generate the data and to copy it into CUDA memory

C2 – Tiled Convolution with CUDA

Implement the convolution at exercise C1 using shared memory and tiling.

- Print the checksum as the sum of the elements of O along all its dimensions. The checksum is expected to be the same as C1

- Report the time to execute the CUDA kernel with the convolution, without including the time to generate the data and to copy it into CUDA memory

C3 –Convolution with cuDNN

Implement the convolution at exercise C1 using cuDNN with the CUDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM algorithm.

- Print the checksum as the sum of the elements of O along all its dimensions. The checksum is expected to be the same as C1
- Report the time to execute the CUDA kernel with the convolution, without including the time to generate the data and to copy it into CUDA memory

Grading

The grade will be a total of 50 points distributed as follows:

EXERCISE	DESCRIPTION	POINTS
C1	Convolution with CUDA	20
C2	Tiled Convolution with CUDA using shared memory	20
C3	Convolution with cuDNN	10

Other grading rules:

- Late submission is -3 points for every day, maximum 3 days.
- Non-compiling code: 0 points

Program output

A single binary should be used for the 3 exercises. The program output should look as follows:

C1_checksum,C1_execution_time

C2_checksum,C2_execution_time

C3_checksum,C3_execution_time

Where each string contains the checksum and the execution time of the exercises C1, C2 and C3 respectively. Execution times should be printed in milliseconds with 3 decimals (use a printf with *%4.3lf*).

Failing to respect this format is -3 points.

Running and Submission instructions

Submission through NYU Classes.

Submission format:

- Please submit a zip file with file name *your-netID*.zip with a folder named as your netID (example: am9031.zip containing am9031/)
- The folder should contain the following file: A file named lab2.cu with exercises C1, C2 and C3
- Before compiling on Prince make sure you load the cudnn module:
  module load cudnn/9.0v7.0.5
- Compile with the makefile: */scratch/am9031/CSCI-GA.3033-023/lab2/Makefile*
- Run experiments on Prince with the sbatch script:
  /scratch/am9031/CSCI-GA.3033-023/lab2/launch_gpu_develop.s

Failing to follow the right directory/file name specification is -1 point. Failing to have programs executed in sequence is -1 point.