CS6023: GPU Programming

Assignment 2

Deadline: 23:59 March 10, 2024

1 Problem Statement

The convolution operation is a fundamental tool in signal processing and image analysis, enabling the extraction of key features from the data. It applies a small matrix called a kernel or filter to an input matrix, often an image.

Convolution serves the purpose of extracting specific features or patterns from the input data. By sliding the filter over the input matrix, element-wise multiplication is performed at each position, and the results are summed. This process generates a new matrix, often termed the feature map, which highlights the presence of particular features in the original data.

Your task is to implement the convolution operation on a 2D matrix using a 2D filter by considering the aspects of **memory coalescing** and **shared memory**.

2 Input and Output Format

2.1 Input Format

Each input is stored in a file.
The first line contains three integers m ,n,and k respectively. m is the number of rows of the input matrix,n is the number of columns of the input matrix and k is the number of rows and columns of the filter.
Each of the next m lines contains n integers, representing m rows of the input matrix.
Each of the next k lines contains k integers, representing k rows of the filter.k will always be odd integer.

2.2 Output Format

The final output from the device should be stored in the array h_ans. This array will be copied to the host and written to a file. This is present in the starter code.

2.3 Constraints

- $1 <= m, n <= 1024$
- $1 <= k <= 50$
- k is odd
- Test cases will ensure the filter fits into shared memory every time .
- Each element of the matrix and filter will be between 0 and 100.

2.4 Sample Input

$$Matrix = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

$$Filter = \begin{bmatrix} 5 & 16 & 1 \\ 0 & 4 & 2 \\ 19 & 15 & 8 \end{bmatrix}$$

2.5 Sample Output

$$Output = \begin{bmatrix} 131 & 255 & 303 & 269 \\ 265 & 487 & 557 & 500 \\ 142 & 190 & 218 & 211 \end{bmatrix}$$

At each position, the center of the filter aligns with the current element of the input matrix. We then calculate the product of the corresponding elements of the matrix and the filter. However, if any part of the filter extends beyond the boundaries of the input matrix, we consider the product as zero for those elements that don't have corresponding matrix elements.
This process continues across the entire matrix, resulting in a new matrix of same dimension $m * n$ where each element represents the sum of products obtained by sliding the filter over the input matrix.

When filter is over the first element 1 of the matrix
5*0+16*0+1*0+0*0+4*1+2*2+19*0+15*5+8*6=131
When filter is over element of 2nd row and 3rd colunm 7 of the matrix
5*2+16*3+1*4+0*6+4*7+2*8+19*10+15*11+8*12=557

It is compulsory to optimize for coalesced accesses. Also, make use of shared memory.Exploit shared memory as much as possible to gain performance benefits

3 General Guidelines

- Do not change any other existing pieces of code that are given in the starter code. Please add necessary memory allocation and *memcpy* operations wherever necessary.
- Please use *cudaFree(void* devPtr)* wherever necessary. This will ensure memory doesn't run out, even for large test cases.
- We will time the kernel calls to ensure that your implementation is parallel.
- This assignment is an individual effort, and it is expected that all work submitted is your own. Plagiarism, the act of using someone else's work, is a serious academic offence and will not be tolerated. Any instances of plagiarism will result in severe consequences, including receiving a grade of zero for the assignment or even facing academic disciplinary actions.

3.1 Testing

- You can use the tester program given along with the starter code to test your program.
- Run *python tester.py* to run the test script. It will show the number of test cases passed.
- You should place your *.cu* file in the *code* folder, given along with tester program.
- In case you are not a Linux user, you may have to change certain function calls in the tester program.

4 Submission Guidelines

- Fill in your code in the *starter.cu* file. Rename it as your *ROLLNO.cu*. For example, if your roll number is *CS22M056*, your file should be named as *CS22M056.cu*.
- Zip this file as *ROLLNO.zip*. Submit on Moodle.

5 Learning Suggestions (Ungraded)

- Implement the same question serially and run it on the CPU. Compare the time taken to execute the serial and CUDA versions of your program on small and large inputs.
- Try to use a minimum number of kernel calls.

- Try to use syncthreads() and syncwarps() as minimum as possible to gain performance benefits
- Exploit shared memory as much as possible to gain performance benefits.