CS6023: GPU Programming

Assignment 3 (13 marks)

Due `September 24, 2017 by 23:55` on Moodle

Problem Specification

For this assignment, you are required to implement three `CUDA` kernels on a 2D array:

- histogram
- stencil
- updateBC

A detailed description about each of these is mentioned below.

1. histogram

```
__global__ void histogram(int *d_input, int* d_bin, int M, int N, int BIN_COUNT){
      // Implement your kernel here
}
```

`d_input` is the input 2D array having M rows and N columns.
`d_bin` is an array to hold the count of the number of elements in each bin. The kernel should populate `d_bin`.
`BIN_COUNT` is the number of bins, i.e. the size of array `d_bin`.

The histogram kernel should count the number of elements in the original array that fall in each bin.

Use the following function to determine the bin in which an *element* of `d_input` falls

$$element \ \% \ BIN\_COUNT$$

Assume `d_bin` to be initialized to zero.

`Kernel launch:` Please launch the histogram kernel with a 1D thread mapping.

int blocks = x1;
int threads = x2;
$histogram <<< blocks, threads >>> (...);$

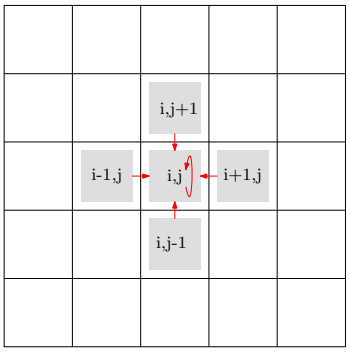

Figure 1: 5-point stencil

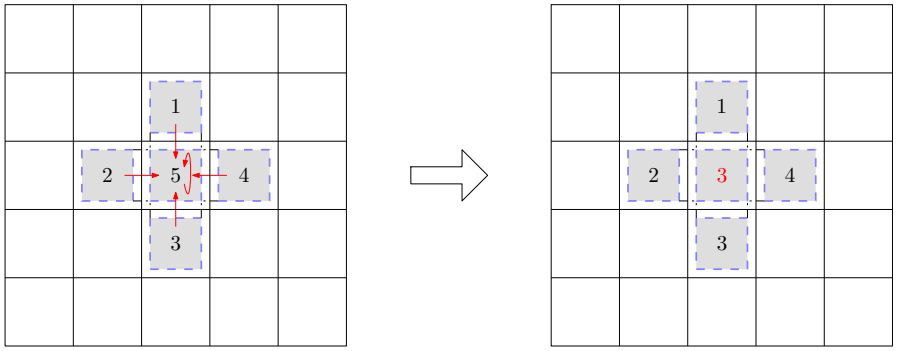

Figure 2: Example of updation

2. stencil

```
__global__ void stencil(int* d_input, int M, int N){
      // Implement your kernel here
}
```

`d_input` is the input 2D array having M rows and N columns.

The stencil kernel should perform a 5-point stencil computation on a 2D matrix (see Figure 1).

You are required to perform the following stencil computation (see Figure 2)

$$A(i,j) = 0.2 * [A(i,j) + A(i+1,j) + A(i-1,j) + A(i,j+1) + A(i,j-1)]$$

This stencil can only be applied on the inside and not on the boundary.

Please note that you need to perform the updation of the input array in-place, i.e. without using a second array.

The kernel should work for rectangular input matrix and rectangular grids.

`Kernel launch:` Please launch the stencil kernel with a 2D thread mapping.

dim3 blocks(x1,y1);
dim3 threads(x2,y2);
stencil $<<<$ *blocks*, *threads* $>>>$ (...);

1	1	1	1
1			1
1			1
1	1	1	1

Figure 3: Output of updateBC

3. updateBC

```
__global__ void updateBC(int* d_input, int M, int N){
      // Implement your kernel here
}
```

`d_input` is the input 2D array having M rows and N columns.

The updateBC kernel should update the boundary values of the input array, `d_input`.

For simplicity, the kernel should assign the value '1' to all boundary nodes of `d_input` (see Figure 3).

`Kernel launch:` Please launch the kernel with a 2D thread mapping.

dim3 blocks(x1,y1);
dim3 threads(x2,y2);
$updateBC <<< blocks, threads >>> (...);$

`Note:` Please launch the kernels, from `main()`, in the following order:

1. $histogram <<< blocks, threads >>> (...);$
2. $updateBC <<< blocks, threads >>> (...);$
3. $stencil <<< blocks, threads >>> (...);$

You are provided a tarball containing:

1. **kernels.cu**: Contains only the kernels' signatures. Your task is to implement the kernels here.

   Should your program require any extra memory (apart from kernel parameters) in any of the kernels, you may declare global-scope device variables outside the kernels, in `kernels.cu`.

2. **example.txt**: Contains a sample input matrix and the expected output after running the kernels.

Submission Instructions

When ready to submit,

1. Rename the file kernels.cu to ROLL_NUMBER.cu.
   For example, if your roll number is `CS14D406`, your file should be called `CS14D406.cu`
2. Upload ROLL_NUMBER.cu on moodle: `https://courses.iitm.ac.in/course/view.php?id=837`
3. Download your file, and make sure it was the one you intended to submit.