CS 629/729 GPU Programming
Lab 5

Spring 2024

Learning Outcomes

- Understanding how to design basic image processing applications
- Understanding the memory efficiency of array of structures and structures of arrays
- Understanding how to use the various optimization techniques to accelerate CUDA kernels

Task 1

In task 1 we are going to implement a kernel that convert a color image into a grayscale image. The code is provided in task01.cu. Complete the following changes:

1.1 Implement the "image_to_grayscale" kernel by having each thread calculate the conversion for each pixel:

$$grayscale\_output = r * 0.21 + g * 0.72 + b * 0.07$$

1.2 Analysis the memory access efficiency of the kernel for both load and store memory transactions.

1.3 The input image array of uchar3 contains RGB data in RGBRGBRGBRGR…. (array of structures order), which is not efficient for read. Adjust the necessary parts of the code (host and device) to organize data as RRRR…GGGG…BBBB… (structure of arrays order). You can modify the kernel to take three uchar arrays instead of one uchar3 array.

1.4 Report the changes in execution time.

Task 2

In task 2 we are going to implement a kernel that apply blur effect on a color image by calculating 2D convolutions. The code is provided in task02.cu. Complete the following changes:

2.1 Implement the “image_blur_A”, “image_blur_B”, and “image_blur_C” kerneld by applying convolution using three different filters:

Filter	Radius	Weights
A	1	1/(1*2+1)^2
B	2	1/(2*2+1)^2
C	3	1/(3*2+1)^2

2.2 For out of bound cases, wrap the index (x, y) as follows:

```
if (x < 0)
        x += IMAGE_DIM;
if (x >= IMAGE_DIM)
        x -= IMAGE_DIM;
if (y < 0)
        y += IMAGE_DIM;
if (y >= IMAGE_DIM)
        y -= IMAGE_DIM;
```

2.4 Since neighboring threads may load the same data, apply tiling with shared memory to reduce memory load costs. Let a block of thread collaboratively load a tile of input data with halo regions, then each thread works on its on pixel.

2.5 Since output of image will be in uchar3 type, directly writing in RGBRGBRGB order is not efficient. Apply corner turning technique improve the memory access efficiency for output data. Specifically, after each thread finishes working on its pixel, let a block of threads collaborative store data back to output array in a coalesced way.

2.6 Report execution times before and after you apply each optimization for all three kernels.