Assignment 4 : Image Filters using CUDA

CSCE-435 Fall 2024
Texas A&M University

November 15, 2024

1 Compile and execute project

- Upload the starter code to your scratch directory after logging into the grace portal.
- Navigate to the directory that the files were uploaded to after logging into the grace portal.
- Intialize the CMake build using the command:

```
$ . build.sh
```

- **Whenever** a change is made to the code, run make to re-build:

```
$ make
```

- Run the batch file by running the following command:

```
$ sbatch filter.grace_job
```

- Once the job is completed, you will be able to see the output file named according to the job id.

2 Assignment

You will be implementing the following methods for image filtering

- **Global** memory to read both the image and the filter
- **Global** memory to read the image and **Constant** memory to read the filter
- **Shared** memory to read the image and **Constant** memory to read the filter. Also use CUDA vector data type(`uchar4`) to copy four elements at a time per thread to shared memory.

Implementation

Implement the following functions present in the file `filter.cu`:

- `fliter_global()`: Kernel that filters the input image with both the image and filter read from **global** memory.
- `filter_constant()` : Kernel that filters the input image with the image read from **global** memory and the filter read from **constant** memory.
- `filter_shared()` : Kernel that filters the input image with the image read from **shared** memory and the filter read from **constant** memory. Also each thread transfers **4** elements to **shared** memory using the `uchar4` vector data type.

In addition, you will also need to fill in additional sections(such as initializing the image, allocating host memory, etc.) marked with a **TODO:** comment in `filter.cu`.

HINT: There is a CPU version of the filter already implemented in the starter code that can be used to check the output from the kernel.

IMPORTANT: The number of threads per block(`THREADS`) and the number of blocks(`BLOCKS`) need to be decided carefully to optimally distribute work among the threads. Consider the domain of the input(2D grid of numbers) when deciding on the dimension and number of threads per block and the dimension and the number of blocks. Calculation for the `BLOCKS` will typically depend upon the size of the image.

Caliper

Use **Caliper** to calculate the runtime of the **three** kernels `fliter_global()`, `filter_constant()` and `filter_shared()`.

NOTE: Caliper intercepts CUDA events to actually measure times so you will need CUDA events in your code.

Effective Bandwidth

Calculate the effective bandwidth of the three kernels(in GB/s). The following link might be helpful Performance Metrics.

Analysis

Analyze and plot the following

- **Image Size v/s Kernel Time**: Vary the image size as **1024 x 1024, 4096 x 4096, 8192 x 8192**. Plot the **three** kernels on the **same** graph. This is for both a **3 x 3 and** a **5 x 5** filter. So, in total this plot will have **six** lines.
- **Image Size v/s Effective Bandwidth**: Vary the image size as **1024 x 1024, 4096 x 4096, 8192 x 8192**. Plot the **three** kernels on the **same** graph. This is for both a **3 x 3 and** a **5 x 5** filter. So, in total this plot will have **six** lines.

NOTE: You are implementing a **3 x 3** and a **5 x 5** filter.

Observations

Write you observations on the variance of the time and effective bandwidth as the image size and the kernel changes.

Grading

Exercise	Points
Correctly implemented `fliter_global()`	20
Correctly implemented `filter_constant()`	20
Correctly implemented `filter_shared()`	20
Graphs	20
Observations	20

3 Submission

Submit a `.zip` file on Canvas containing

- A pdf with the graphs and your observations.
- The file `filter.cu` with your code changes.
- A `.zip` file containing your `.cali` files.