Assignment 4. 2D Convolution

Minjae Gwon, Department of Computer Science and Engineering.

Problem 1

> How many floating operations are being performed by your convolution kernel?

- `((Mask_width * Mask_width) * imageChannels) * (imageWidth * imageHeight)`
  - `Mask_width * Mask_width` operations are required for calculating one output entry.
  - Each thread runs `(Mask_width * Mask_width) * imageChannels` operations to obtain entries regarding to the number of channels.
  - `imageWidth * imageHeight` threads are needed to get convoluted image.

Problem 2

> How many global memory reads are being performed by your convolution kernel?

- `imageChannels * (((blockDim.x * blockDim.y) * (gridDim.x * gridDim.y) - ((imageWidth + Mask_radius) * (imageHeight + Mask_radius) - imageWidth * imageHeight)) + (Mask_width^2 * imageWidth * imageHeight))`
  - Most of threads read `1` entry of a channel of image.
  - `(blockDim.x * blockDim.y) * (gridDim.x * gridDim.y)` threads exist.
  - `(imageWidth + Mask_radius) * (imageHeight + Mask_radius) - imageWidth * imageHeight` entries are halo cells.
  - `imageWidth * imageHeight` threads run `Mask_width^2` read operations for reading mask.
  - Above procedures are applied `imageChannels` times.

Problem 3

> How many global memory writes are being performed by your convolution kernel?

- `imageChannels * imageWidth * imageHeight`
- Same as the number of input image's entries.

Problem 4

> How much time is spent as an overhead cost for using the GPU for computation? Consider all code executed within you host function with the exception of the kernel itself, as overhead.

> How does the overhead scale with the size of the input?

- The overhead should scale at same rate as the input grows, because copying data between host and GPU could be overhead for using the GPU.

Problem 5

> What do you think happens as you increase the mask size (say to 1024) while you set the block dimensions to 16x16? What do you end up spending most of your time doing? Does that put other constraints on the way you'd write your algorithm?

- Increased mask size, compared to the block dimensions, not only makes required size of shared memory too large but also generate relatively many halo cells. These can lead to inefficiency of tiling algorithms.

Problem 6

> Your version of `template.cu`.

- Please refer to the attached `template.cu`.

Problem 7

> The result as a table/graph of kernel execution times for different input data, with the system information where you performed your evaluation. Run your implementation with the input generated by the provided dataset generator. For time measurement, use `gpuTKTime_start` and `gpuTKTime_stop` functions (You can find details in `libgputk/README.md`).

- Please refer to the attached `evaluation.pdf`. 'Doing the computation on the GPU' column expresses the execution times of the kernel. Note that it is generated by `scripts/manage.py`.