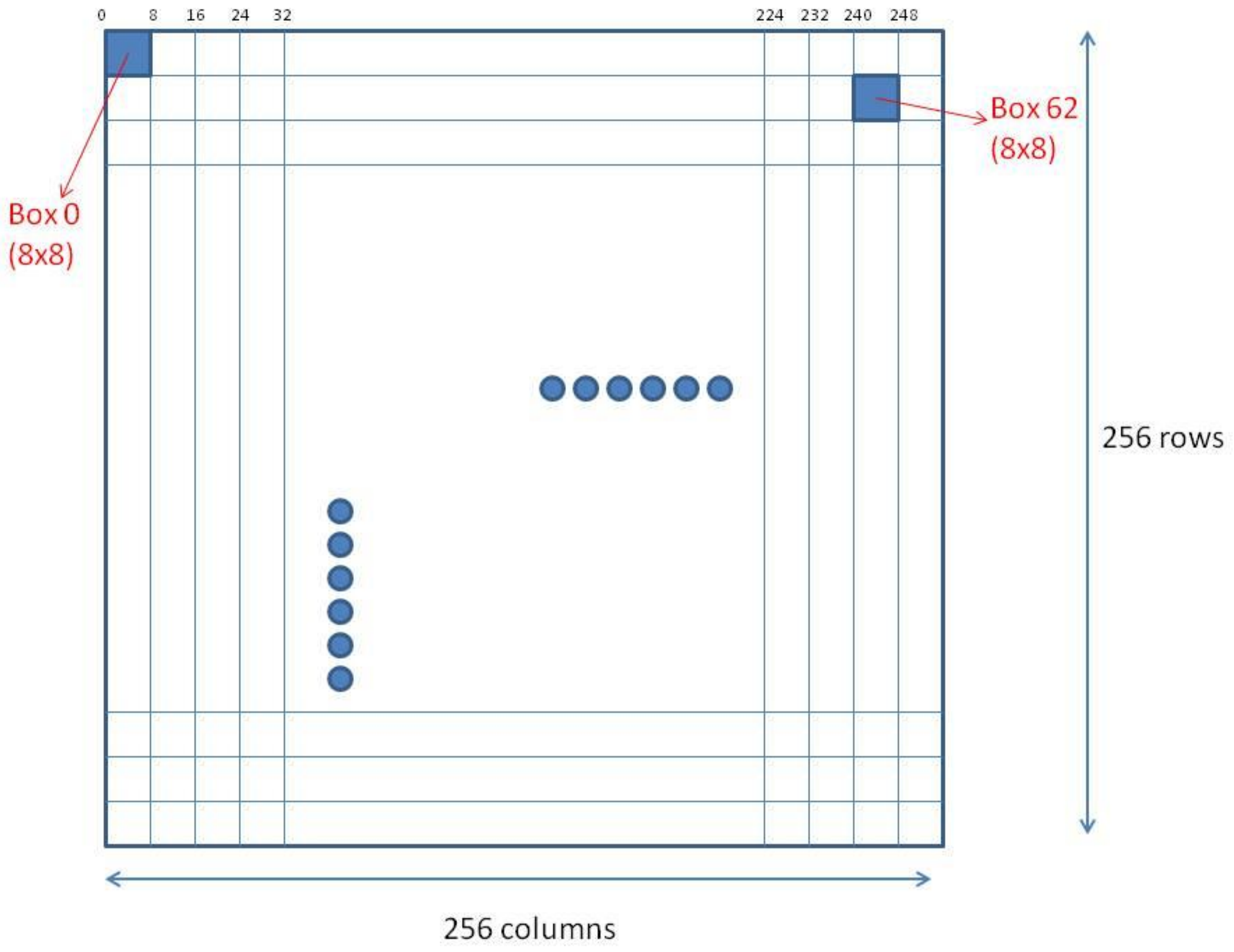
GPU Parallel Programming using C/C++

ECE206/406 CSC266/466

Final Project (35 pts) Due : Dec 12, 2014 11:59PM

- **No late submissions will be accepted**
- In this project, you will be developing a CUDA-based program to "unscramble" an image :
- Take the 256x256 Black & White dog image below as an example,
- This image has 256 pixels on every row, and has 256 rows total.
- If someone swapped the columns and rows of this image, you would get an unrecognizable image
- If the swaps were completely random, there would be no way to recover the original image
- However, if someone gave you a ***box-checksum*** of the image ***before the swap***
- You could try many different configurations of how the swap could have been done
- You could recover the original image, as long as the box checksums were unique.
- This will require a lot of tries though …Well, this is what you will do using the GPU
- A box is square, and may be an 8x8, 4x4, or 2x2 pixel unit.
- Your input :
  - a) a scrambled image scrambled.jpg,
  - b) scrambled.jpg_NxN,csv: box checksums (row, column) of this image before scrambling
- You are required to find :
  - The original image , original.jpg
- Some MATLAB code has been provided for you. You can use this MATLAB code to see how the scrambling and checksum-generating algorithms work, and even to start prototyping a solution.
- Note that you can run MATLAB on bluehive by loading the module ('module load matlab') then running 'matlab'. To get the full GUI, connect with X forwarding like we've been doing for nsight.

UNIVERSITY of ROCHESTER



- Look at the symbolic 256x256 image above:
    - Each box is composed of 8x8=64 pixels (64 byte values).
- Therefore, the very first box in the image is bounded by columns [0…7] and rows [0…7]
- This is shown as Box 0 above (8x8=64 pixels) …
- Since this is a B&W image, each pixel is an 8 bit gray shade value
- The checksum of a box is defined as the XOR of the 8 rows within that box. i.e.:
    - Concatenate the bytes of 1 row in a box, you get a 64 bit number (8 byte)
    - XOR all 8 of these numbers (i.e. all 8 rows), you get the BOX Checksum
    - With this checksum alone, it may not be possible to unscramble the image (e.g. you cannot detect a row swap within the box). So we will provide you with *two* checksums for each Box: one checksum computed by XORing the rows as we just described (i.e. a 'row' checksum), and one computed the same way on the *transpose* of the Box (i.e. a 'column' checksum).
- One possible "recovery method" is straightforward (yet very compute-intensive):
    - Calculate every possible swap configuration for each row and column
    - If you find a configuration for which a checksum matches, you determined a row or column
    - Continue until every column/row is determined. Recovery is done.
- You are encouraged to try better recovery methods.

- We have provided scrambled images of a few different sizes (32x32, 96x96, 256x256), with checksums for all possible box sizes (8x8, 4x4, 2x2). Your code should take one image and one box size, and find the original image. (i.e., even though you have all 3 sets of box checksums available, you should be able to descramble the image using only one of them.)
- Finally, note that the checksums are not guaranteed to be unique; collisions may occur which prevent you from fully/correctly descrambling the image. But it should be much better than when you started.

➢ **PART A (3 pts) :** Write the CPU version of this program
➢ **PART B (12 pts):** Write the GPU version of this program
➢
➢ **REPORT:**
➢ Explain your algorithm
➢ Provide a very detailed report of how you chose the block size, number of threads, and the kernel. If your kernel runs unusually slow, you will lose points.
➢ Use the CUDA Occupancy Calculator and/or Visual Profiler to optimize your program.
➢ Comment on shared memory bank conflicts, thread divergence, and any other 'hindrances' to your performance… and how you overcame them.

Grading:

4 points : Clarity and technical detail (and correctness) of the report.
4 points : Program design, efficiency of the parameter choices (# blocks, etc ...)
4 points : Runtime comparison to other students. Unusually slow programs will lose 4 full points.
Efficiency is the key in this assignment. Use every trick I showed you to make your program faster.

+5 points : THE FASTEST IMPLEMENTATION IN THE CLASS WILL GET 5 BONUS POINTS.