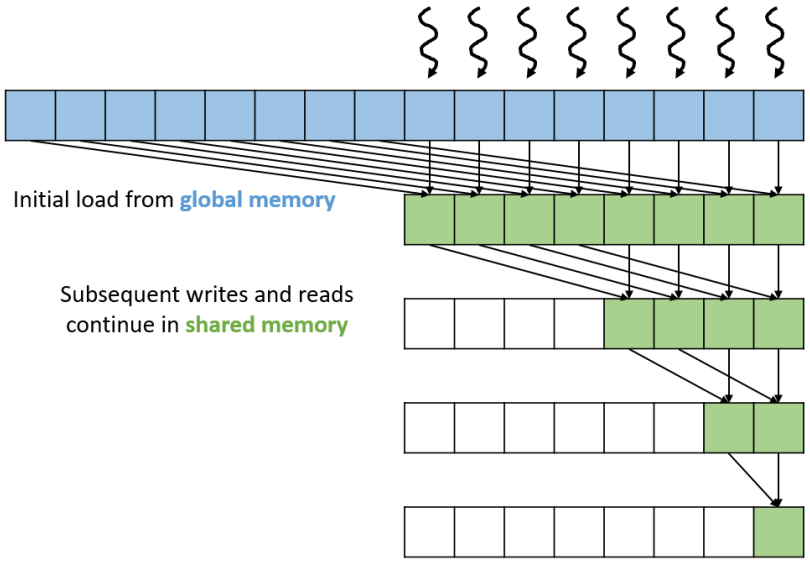
CMPS 224/396AA: GPU COMPUTING
ASSIGNMENT 6

In this assignment, you will implement a reduction kernel with the following optimizations applied: memory coalescing, eliminating control divergence, using shared memory, and using warp shuffle instructions. You do not need to apply thread coarsening.

Unlike the reduction pattern implemented in class, your implementation should follow the pattern illustrated below. No credit will be given if you use the reduction pattern implemented in class (since the code is already available in the slides).



Your kernel is expected to work for any set of input dimensions so make sure to handle boundary conditions correctly. You should test your implementation on small input sizes, since the default large input size may pass because of rounding even if boundary conditions are not handled correctly.

Instructions

1. Place the files provided with this assignment in a single directory. The files are:
   - `main.cu`: contains setup and sequential code
   - `kernel.cu`: where you will implement your code (you should only modify this file)
   - `common.h`: for shared declarations across `main.cu` and `kernel.cu`
   - `timer.h`: to assist with timing
   - `Makefile`: used for compilation

2. Edit `kernel.cu` where TODO is indicated to implement the reduction kernel.

3. Compile your code by running: `make`

4. Test your code by running: `./reduction`
    - If you are using the HPC cluster, do not forget to use the submission system. Do not run on the head node!
    - For testing on different input sizes, you can provide your own values for input size as follows: `./reduction <N>`

5. You are also provided with a file called `questions.txt` which contains questions about the assignment. Answer the questions in the file after implementing your kernel.

Submission

Submit your modified `kernel.cu` and `questions.txt` files via Moodle by the due date. Do not submit any other files or compressed folders.