

CSCD 439/539 GPU Computing HW4

Modify Reduction

No Late Submissions are accepted. **Rules:** Your code must CUDA C Language. If your program shows a compilation or run-time error, you get a zero for this assignment.

Submission: Wrap up all your **source files** into a single zip file. Name your zip file as *FirstInitialYourLastNameHw4.zip*. For example, if your legal name is Will Smith, you should name your zip file as wsmithHw4.zip. A simple makefile has been provided in the zip file.

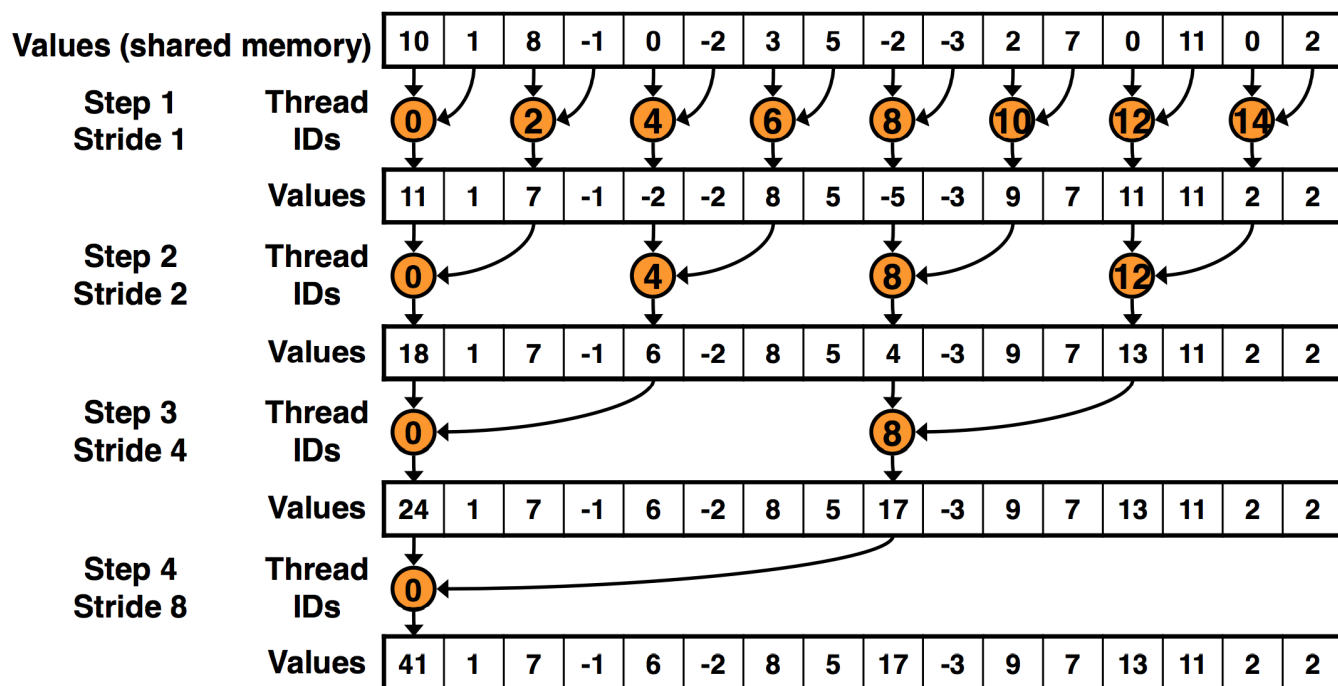
For archive purpose, please also submit your single zip file on EWU Canvas by following CSCD439-01 Course → Assignments → Hw4 → Submit Assignment to upload your single zip file.

Problem Description:

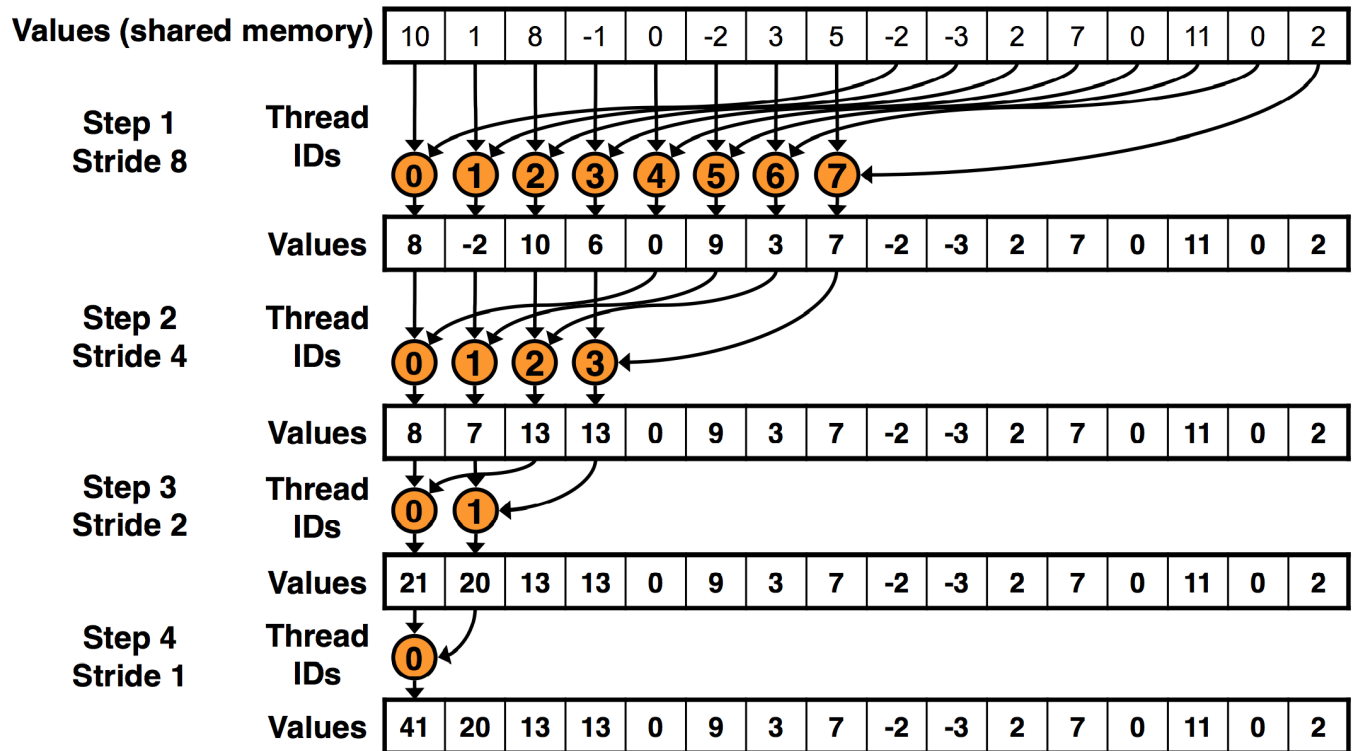
Based on the lecture about reduction algorithm on CUDA device, you are required to implement the following features and answer the questions.

1, Download the demo code d10_reduce.zip on canvas under Files → DemoCode. Read and understand the provide CUDA C program. We have already gone through this code in class.

2, The provided kernel in the demo code, `__global__ void reduce2(float *in, float *out, int n)`, is not optimized. The basic idea on which kernel `reduce2()` is designed is described in lecture notes regarding reduction algorithm. The following diagram illustrates `reduce2()` kernel.



3, You have to write another kernel `reduce3()`, that also performs reduction on CUDA device. Kernel `reduce3()` accepts the same set of parameters list as `reduce2()` kernel has, with each parameter maintaining the same meaning. But `reduce3()` kernel uses the following threads-data mapping and data access patterns.



In this design, the size of thread block (in shape of 1 * N) could be half of that in kernel `reduce2()`, while they are able to perform the same task. The major difference between `reduce2()` and `reduce3()` kernel lies in the way regarding which pair of data elements are processed by a thread.

3, Change the `main()` function to invoke the kernel **`reduce3()`** you implemented, and measure the kernel execution time. Then you have to compare the kernel `reduce3()` with the `reduce2()` by filling out the following table. **If the number of blocks you created exceeds the maximum allowed limits, please use 2D grid to create more threads. We discussed this idea in class.** In these cases, you have to modify `main()`.

Input size	1048576	16777216	67108864	134217728
Block Dimensions	1 X 1024	1 X 1024	1 X 1024	1 X 1024
T1:time cost for reduce2 (ms)				
T2:time cost for reduce3 (ms)				
Speedup = T1 / T2				

4, Do you identify any advantage of `reduce3()` over `reduce2()` kernel? Or vice versus? And why? (hint: in terms of performance, such as bank conflicts or condition divergence)

5, What are disadvantage(s) of `reduce2()`? And any disadvantages for `reduce3()`? And Why? (hint: in terms of performance, such as bank conflicts or condition divergence)