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 *FirstInitialYourLastName*Hw4.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.

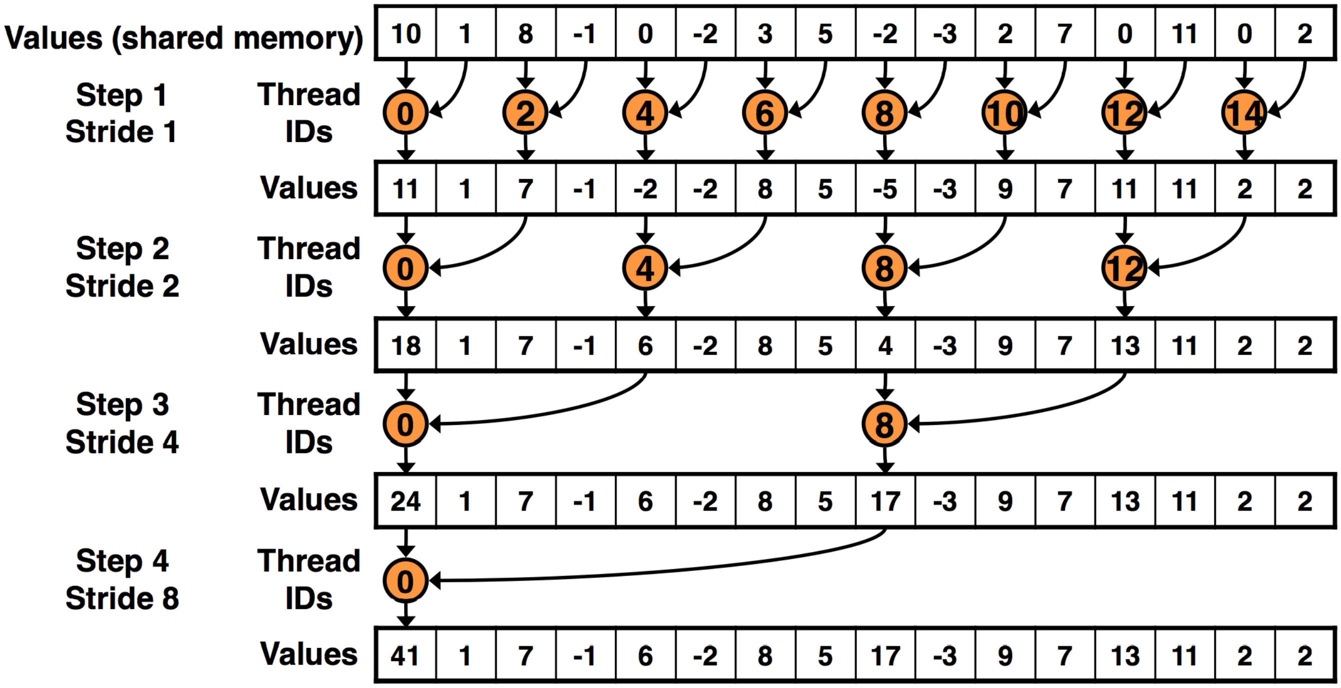
For archive purpose, please also submit your single zip file on EWU Canvas by following CSCD439-01 Course   Assignments   Hw4   Submit Assignment to upload your single zip file.

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_reduce4_add.tar.gz on canvas under Files   DemoCode. Read and understand the provided CUDA C program. We have already gone through this code in class. The demo code is also included in the start up package.

2, The provided kernel in the demo code, __global__ void reduce2(float *in, float *out, int n), is not optimized. The basic idea about kernel reduce2() is described in the lecture notes regarding reduction algorithm. The following diagram (figure 1) illustrates reduce2() kernel.



10 1 8 -1 0 -2 3 5 -2 -3 2 7 0 11 0 2

11 1 7 -1 -2 -2 8 5 -5 -3 9 7 11 11 2 2

18 1 7 -1 6 -2 8 5 4 -3 9 7 13 11 2 2

24 1 7 -1 6 -2 8 5 17 -3 9 7 13 11 2 2

41 1 7 -1 6 -2 8 5 17 -3 9 7 13 11 2 2

Figure 1

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 does, with each parameter maintaining the same meaning. But reduce3() kernel uses the following threads-data mapping and data access patterns, as described below in figure 2.

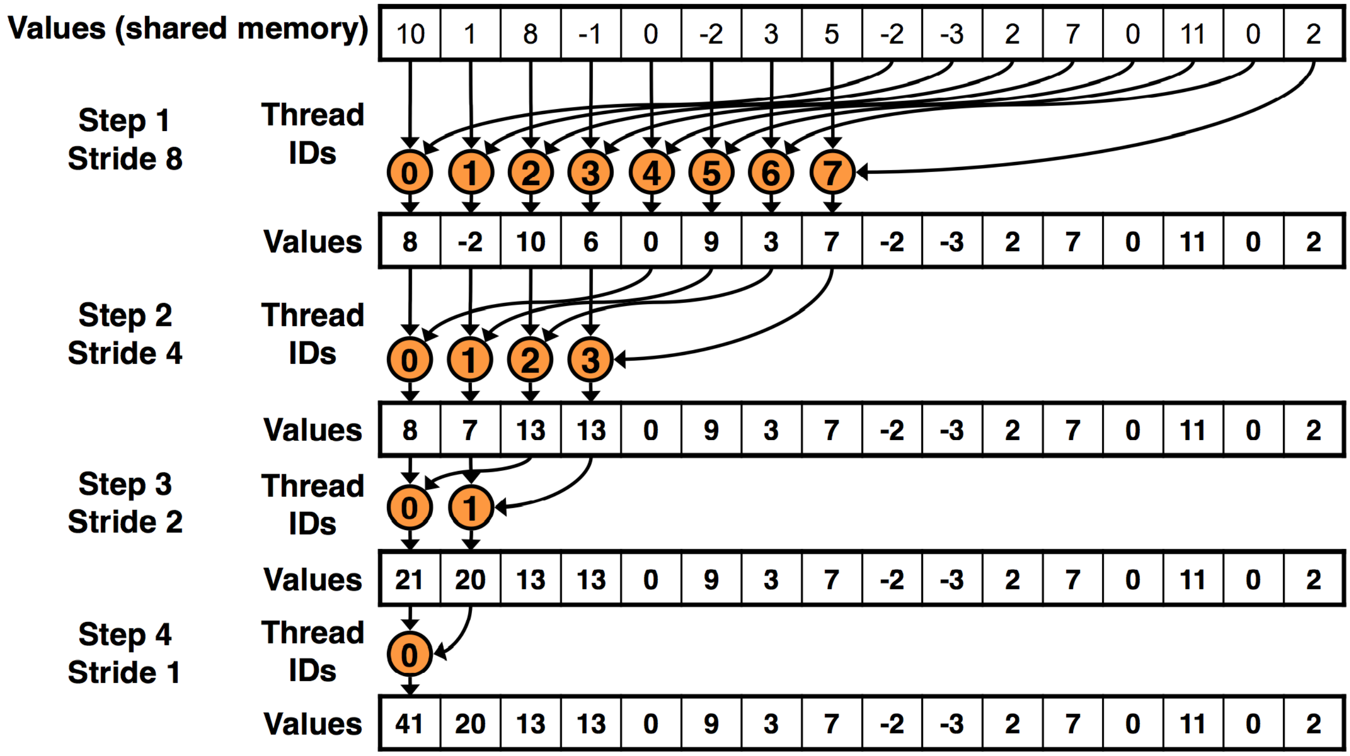


Figure 2

In the design of figure 2, **the size of thread block (in a shape of 1 * N) is required** to be half of the tile size, (i.e. also half of the thread block size used in the kernel reduce2()). In comparison, reduce3() will use less number of threads in each block, but can achieve the same goal as reduce2(). In this way, we improve the performance of reduce3(), compared with reduce2(). **You have to change the execution configuration parameters in the main()** in order to set up the new block size variable for reduce3(). Another major difference between reduce2() and reduce3() kernel pertains to 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.

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, condition divergence or idle threads.)