Lab 7

CMPUT 398

Objective

The lab's objective is to implement an optimized kernel that performs reduction of a 1D list. The reduction should give the sum of the list. Your kernel should be able to handle input lists of arbitrary length. However, for simplicity, you can assume that the input list will be at most 2048 x 65535 elements so that it can be handled by only one kernel launch. The boundary condition can be handled by filling "identity value (0 for sum)" into the shared memory of the last block when the length is not a multiple of the thread block size. Further assume that the reduction sums of each section generated by individual blocks will be summed up by the CPU.

This lab will be submitted as one zipped file through eclass. Details for submission are at the end of the lab.

Instructions

Edit the code where the TODOs are specified and perform the following:

- Allocate device memory
- Copy host memory to device
- Initialize thread block and kernel grid dimensions
- Invoke CUDA kernel
- Copy results from device to host
- Deallocate device memory
- Use shared memory to reduce the number of global accesses, handle the boundary conditions in when loading input list elements into the shared memory
- Implement a CPU loop to perform final reduction based on the sums of sections generated by the thread blocks

Local Setup Instructions

Steps:

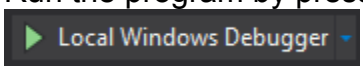
1. Download “Lab7.zip”.
2. Unzip the file.
3. Open the Visual Studios Solution in Visual Studios 2013.
4. Build the project. Note the project has two configurations.

a. Debug

b. Submission

But make sure you have the “Submission” configuration selected when you finally submit.

5. Run the program by pressing the following button:



Make sure the “Debug” configuration is selected. Running the program in Visual Studios will run one of the tests located in “Dataset/Test”.

Testing

To run all tests located in “Dataset/Test”, first build the project with the “Submission” configuration selected. Make sure you see the “Submission” folder and the folder contains the executable.

You are given a reference simple reduce .exe file that is the un-optimized version. The tests will be marked by comparing your run speed to this file. To run the tests, click on “Testing_Script.bat”. This will take a couple of seconds to run and the terminal should close when finished. The output is saved in “Marks.js”, but to view the calculated grade open “Grade.html” in a browser. If you make changes and rerun the tests, then make sure you reload “Grade.html”. You can double check with the timestamp at the top of the page.

If you get the correct answer on all test, you get 50 points. If your program is optimized according to the grade.html file, you get another 30 points. The optimization test is actually done on the last test (number 10) in your dataset folder but this information is not relevant for this lab because you do not need to submit a report on run time of your program.

Do not submit the reference simple reduction file or you will get 0 mark.

Report

You need to submit a report file contains all these questions based on your code:

1. How many floating operations are being performed in your reduction kernel? Don’t include assign as floating operation (=). So only the following: +, -, /, and *.
2. How many global memory reads are being performed by your kernel?
3. How many global memory writes are being performed by your kernel?
4. How many times does a single thread block synchronize to reduce its portion of the array to a single value?

The answer must be algebraic expressions of numInputs (length of the input) and blockSize where applicable

Each question worth 5 points

Submission

After you build your project with the “Submission” configuration selected (see above) you will see a new folder called “Submission” in the project directory. If you open the folder you will see your executable. Make sure you add your questions as a text or pdf file. Finally zip the folder “Submission” and upload the zipped folder to eclass.

If you don’t see this folder or are missing the executable, then one of two things happened. First your build could have failed and you should check for any errors in your build log in Visual Studios. Also you might have had the “Debug” configuration selected (see above). Make sure the “Submission” configuration is selected.

Mark Breakdown

It is important that you **do not** add or remove any print statements or wb functions. We use them for part of your grading so if you make changes the marking script could fail.

Part	Breakdown (%)
Reduce	50 %
Optimization	30 %
Report	20 %