The University of Birmingham
School of Computer Science

Assignment 01

This assignment will be marked out of 10 and contributes

- 10% of the total module marks for 06-26945 Distributed and Parallel Computing
- 10% of the total module marks for 06-26944 Distributed and Parallel Computing (Extended)

Write a CUDA program, using your implementation of the work efficient parallel Blelloch algorithm for scan, to calculate the windowed average of a large vector of integers. The integers should be in the range 0 to 9 inclusive (use `rand()%10` to generate the integers). Your final version should calculate an integer scan of the input array, and then calculate the floating point windowed average of the input array by means of subtracting the scan total before the window from the scan total at the end of the window and (floating point) dividing by the size of the window for each element of the output vector.

Marks will be assigned as follows:

06-26945 Distributed and Parallel Computing:

- 3 marks for getting full scan for small arrays working (second level scan required: test vector should be of len 1,000,000)
- 3 marks for getting full scan for large arrays working (second and third level scans required, test vector should be of length 10,000,000)
- 2 marks for getting the parallel calculation of the windowed average from the scan output working
- 2 discretionary marks for clean coding style, good implementations, good performance improvements, interesting aspects of your implementation etc.

06-26944 Distributed and Parallel Computing (Extended):

- 2 marks for getting full scan for small arrays working (second level scan required: test vector should be of len 1,000,000)
- 2 marks for getting full scan for large arrays working (second and third level scans required, test vector should be of length 10,000,000)
- 2 marks for supporting data segment sizes in the scans that are twice the size of the thread block size so that all threads in a thread block compute sums in the first iteration of the reduce
- 2 marks for getting the parallel calculation of the windowed average from the scan output working
- 2 discretionary marks for clean coding style, good implementations, good performance improvements, interesting aspects of your implementation etc.

Your submission should include:

- Your implementation as a single "`.cu`" program code file
- A comment at the top of the file that reports:
  - Your name and student id number
  - Which of the assignment goals you achieved:
    - scan for small arrays
    - scan for large arrays
    - data segment size of twice block size (For extended students only)
    - windowed average calculation from scanned output vector
  - What your performance speedup is for your target goal on one of the GeForce GTX 960 gpu based machines in the LG04 lab
  - Any implementation details or performance improvement strategies that you successfully implemented and which improve upon a base level implementation of the target goals.

Notes:

- All final performance measures should be taken on the GeForce GTX 960 gpu based machines in the LG04 lab (in Row A and most of row B). Both the sequential and the parallel timings reported should be on these machines.
- Note that the sequential versions that you compare against must correspond to the parallel version, for example, if you only get the single level block scan working, then your sequential version should also do a block scan so that the elements of the results for the sequential version can be compared to those of the parallel version.
- The scan should be an **integer** scan that generates an **integer** result vector. This means you can easily test the scanned result without having to worry about round-off errors. The windowed average results, however, should be calculated as floating point: there will be no significant problems with round-off error there.
- Calculating the windowed average from the scan results is independent of getting scan working: you can earn the marks for the windowed average calculation by showing it works on a vector of increasing integers, even if you must create the input vector manually instead of by using scan.
- Getting your implementation working for data segments that are the same size as the thread block size is likely to be easier that getting it working for data segments that are twice the size of the thread block — it might be advisable to get the easier implementation working first.
- Details of the windowed average can be found in the handout for the previous lab exercise.
- Optimise and fine tune your program to get the best speedup possible. Note that this includes re-structuring your code as well as trying different configuration parameters.
- Upload your final code with the comment reporting your results via Canvas

The deadline for this assignment is

18:00 Friday 11/11/2016