CSE 603 **Programming Assignment 3**
Due Date: Nov 11, 2014
Weight: 11% of your grade

Problem: Simple Prime Number Generator and Simple Parallel Bucket Sorting using GPU: study of scalability.

Description: In this assignment you will optimize parallel programs using nVidia GPUs and you have the option (for additional credit) of using multiple GPUs in a distributed-memory environment using MPI. In particular, you will have to parallelize a simple prime number generator and a simple bucket-sorting algorithm on GPUs (both of which you have implemented using OpenMP and MPI).

The description of the prime number generator and bucket sorting algorithms were given in the previous two assignments (1 and 2).

- Your task is to parallelize these algorithms using the GPU cluster in CCR
- You will use one GPU for each of these algorithms
- The performance you achieve will be a factor in grading, so use shared memory in GPUs
- For the sorting algorithm, use the programs that were provided for assignment 2 to generate random numbers with a normal distribution.
- You will need to do a comparison of the performance of this GPU implementation with that of OpenMP and MPI.

What you need to do:

- Your report should include the following:
  - Your rationale for parallelizing the program
  - Speedup plots for varying problem size and cores (if you are using more than one GPU)
  - Complete source codes in different directories – sequential, GPU, MPI+ GPU (if you are implementing this version) including makefiles for each embedded within the directories. Include scripts you may have used to run programs with different parameters and also scripts that you may have used to collate results and create graphs (highly recommended to do this to make it easier to finish your work in time).
  - Script (with complete description) to run your programs (including samples)
  - Discussion of results to include the following (graphs)
    - Impact on the performance of the algorithm based on
      - Increase in the size of the problem
      - Increase in the number of GPUs (if you are doing this)
      - Impact of load balance on performance
  - How does this relate to your theoretical evaluation of the scalability of this algorithm?

Extra Credit (2%): Incorporate a combination of MPI and GPU and compare the performance in terms of increase/decrease of speedup and scalability.

Specific Submission Guidelines:

1. Please submit your assignment as username.zip
2. Root directory should include Prime and Sort two directories. Each directory should include Sequential, GPU, and GPU_MPI (for extra point only) sub directories. Folder name should be as indicated above.
3. Each sub directory must contain a Makefile and source file.
4. Name of the executable file (after running make command), should be as same as source file (e.g. for sort.cu file, the executable -o file should be named as sort)
5. For each program, when <-t problem size> arguments are passed, it should print out the results and execution time. For example, when running ./sort -t 100, it should print out the sorted result for problem size 100 on standard output along with execution time.
6. Submit your report as username.pdf.
7. In your report, please include execution time for the following cases. (For sorting problem, please set maximum number as equal to 10 * problem size)

Program	Number of cores and device	Problem size
Sequential	1 node 1 task-per-node 0 GPU	100,000
Sequential	1 node 1 task-per-node 0 GPU	1,000,000
GPU	1 node 1 task-per-node 1 GPU	100,000
GPU	1 node 1 task-per-node 1 GPU	1,000,000
GPU_MPI	1 node 2 task-per-node 2 GPU	100,000
GPU_MPI	2 node 4 task-per-node 2 GPU	100,000
GPU_MPI	2 node 4 task-per-node 2 GPU	1,000,000
MPI	2 node 4 task-per-node 0 GPU	1,000,000
OpenMP	1 node 8 task-per-node 0 GPU	1,000,000

You should also include other experiment cases in your report.

Note: username in this context is your UB username, and the other file names should be followed exactly as stated.