EE-193: Parallel Computing
Lab 1: Histogram Generation with C++ threads

In this homework, your goal is to write a program that processes input data and builds a histogram. You want to take advantage of multiple threads to run as fast as possible.

We noted in class that if the threads spend most of their time in a critical section, then the handoffs involved will be very slow. Thus, you want to somehow ensure that most of the work in the threads does not need to be in a critical section. How you do this is up to you.

The program provided to you accepts the size of the input data set as an argument. It first creates a randomly-initialized data set and computes the histogram with a known-correct single-threaded implementation (which is already coded). It then processes the same input data using a multi-threaded implementation (which you must write), and checks the answer.

Use an input-dataset size of 1 billion pieces of data. Collect multi-threaded data with 1, 2, 4, 8, 16, 32 and 64 cores.

Write a short report explaining how you shared the work between threads. Also discuss the speedup that your multithreaded version achieved over the serial version for various numbers of threads. We've not covered architecture in any detail, so just take your best guess. You can refer to the parallelized work vs. the serial work (i.e., Amdahl's Law), and to how much data you had to move around vs the chip's total memory bandwidth.

Logistics:

- The lab assignment is due ????? at midnight. You must work on this assignment on your own. Submit your project via **provide**. You should submit a copy of your report as a PDF, as well as your version of histogram.cxx.
- Edit the function compute_multithread() within the file histogram.cxx to complete the functionality of histogram generation. You may add multiple functions to the file as needed to achieve this functionality. Do not change the source code elsewhere.
- Compile with the line **c++ -pthread -std=c++11 -O2 histogram.cxx ee193_utils.cxx**. The **-O2** will make your program run much faster, and you should use that for your final results collection. However, for initial debug you may want to skip it, and add **-g** (which creates information for debugging).

Resources:

- Pages 66-70 of "An Introduction to Parallel Programming" cover the histogram problem.
- Use any Linux system for your development, and use the Dell24 cluster for your benchmarking. Sign up for Dell24 time using the spreadsheet on the class web page.