Submission Instructions

1. Submission Deadline: Feburary 7, 11:59pm. No extension will be allowed.
2. How to submit: To turn in your source code files, execute turnin command "turnin -c cs525 -p hw1 DIR" where **DIR** is the directory your code reside in; to check your submission, use "turnin -c cs525 -p hw1 -v". A guide to turnin command can be found here and here.
3. Machine to use: You can use MC servers to run your program. The information of MC servers can be found here. Before executing your program, you can use system system monitoring command such as "top" to check the utilization of your current machine.

1. **Problem:** Parallelizing Quicksort

   This assignment requires you to write a threaded quicksort. The program takes two inputs -- the number of integers in the array and the number of threads to use. It then goes through the following steps:

   i Initializes an array of specified size with random numbers. This is done in serial execution (no threads at this point)

   ii Starts the execution clock.

   iii Creates the specified number of threads that execute parallel quicksort.

   iv Waits for all threads to be done and stops the execution clock.

   v Reports the execution time.

   Notes: Algorithm Details

   1 The list is partitioned across the threads. All threads pick **a common pivot**, e.g., a pivot is selected by one of the processors and made known to all processors. Each thread marks its assigned entries as less than, greater than, or equal to pivots. The threads then perform global sum scan operations (prefix-sum) to identify locations of each assigned entry. The entries are then permuted in parallel. The list is then partitioned into two lists – one less than pivot and one greater than pivot.

   2 The threads are then assigned to one of the two lists – the less-than list or the greater-than list in proportion of the size of the lists.

   3 Both lists are then repartitioned and the process is repeated until a single thread is assigned to a sublist.

   4 At this point, the thread sorts the sublist locally (using qsort).

   5 More details on this process are available in the Sorting Chapter of your book or slides for Sorting on course homepage. Please feel free to refer forward.

Grading:

Your program will be run on 16 cores and graded on the following rubric:

Your program will be executed on a list of 100M entries.

- Basic execution: 25% (A single-thread version will give you 25%)
- Speedup of 4 - 8: 50%
- Speedup of 8 - 12: 75%
- Speedup of over 12: 100%