Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 2:00-3:50pm @ FAB 40-07; Office Hr: MW 1-2pm & by appt.

Assignment 2: Programming with OpenMP
(Due Wednesday, 5/4/16)

This assignment is to practice multi-threaded programming using OpenMP (version 3.0). You'll convert two sequential programs into parallel programs by inserting OpenMP directives. You'll run these programs and collect timing results.

For this assignment, most work are for both CS415 and CS515 students. One small extra part is for CS515 students only (indicated below). This assignment carries a total of 10 points.

Download the file `assign2.zip` to your CS Linux account and unzip it. You'll see two program files, `prime.c` and `qsort.c`, for prime-finding and quicksort, respectively. Make a copy of these two programs as the OpenMP version; call them `prime-omp.c` and `qsort-omp.c`.

Directive Insertion

Study each program to see where parallelism may be introduced. Then decide what OpenMP directives to use. *Hints:* (1) you always need a `parallel` directive; and (2) you always need to specify the number of threads to use — either through the environment variable `OMP_NUM_THREADS` or through the `num_threads` clause.

Compilation and Execution

Once you have compiled your modified programs, you need to test them to make sure they produce correct results and that they use concurrent threads. (*Hint:* Insert debugging/printing code to verify results and to echo a message from each thread showing its `tid`.)

While you may develop your programs somewhere else, you need to compile and run them on the CS Linux Lab machines. To compile an OpenMP program on Linux, use the `"-fopenmp"` flag.

Timing Measurements

You are required to collect timing results from your two programs and the two provided sequential programs. For this part, run your programs on the 30-core server `babbage.cs.pdx.edu`.

Use the simple Linux timing utility `time` to collect timing data of a program execution, *e.g.*

```
babbage> time qsort 1000000
0.308u 0.004s 0:00.31 96.7%     0+0k 0+0io 0pf+0w
```

The first two numbers are accumulative user and system CPU times (across all cores); the third number is the real (elapsed) time; and the fourth is the percentage of the CPU that this job got. (Ignore the last group of numbers.) For our timing study, you should use the elapsed time data.

For each program, collect timing results for different thread-number and array-size settings. For thread numbers, cover the following cases: 1, 2, 4, 8, 16, 32, 64, and 128. For array sizes, cover at least four large sizes. (For `prime`, include at least sizes 10^7 and 10^8; and for `qsort`, include at least sizes 10^6 and 10^7.)

Hints: If you've inserted debugging code, comment them out before making timing runs. (Do not comment out the print statements from the original program.) Also, for each timing configuration, run the program multiple times, and use the best result for your timing report.

Requirements:

- Record your timing results in a nice, readable format, *e.g.* tables or charts.
- Compare timing results against the two sequential programs. You should expect to see some speedup for large array sizes. If you don't see any speedup at any configurations, you need to tune your programs. The tuning could include, (1) adjusting a directive's scope or clause; (2) using directives; and (3) adding or removing directives.

[CS515 Students] Comparison with Pthread Version

Compile and run the Pthread version of the quicksort program you wrote for Assignment 1 on `babbage`. Collect timing data for the same thread-number and array-size settings as you did for your OpenMP version. Compare the results of the two versions.

Program Submission

Summarize your experience in a short report. Include your timing results there. Speculate reasons for the performance of your programs. Include any lessons you learned and any conclusions you want to draw. The length of the summary should be around 2-3 pages.

Make a `zip` file containing your two OpenMP programs and the summary report. Use the Dropbox on the D2L site to submit your assignment file.