Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 14:00-15:50 @ UTS 308; Office Hr: M 13:00-14:00 & by appt.

Programming Assignment 2

(Due Wednesday, 5/6/15)

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 a small extra part is for CS515 students only (indicated below). This assignment carries a total of 100 points.

Setup

Download the file `assign2.zip` to your CS Linux account and unzip it. You'll see two program files, `prime.c` and `qsort.c`. These are the sequential prime-finding and quicksort programs. (`qsort.c` is the same program you used in Assignment 1.) Copy these two programs to `prime_omp.c` and `qsort_omp.c`. Work on the programs for the insertion of OpenMP directives. Keep the two original programs unchanged for timing use.

Program Modification

For each program, study to see where parallelism may be introduced. The next step is to decide what OpenMP directives to use. Two quick 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).

Program Compilation and Execution

While you may develop your programs somewhere else, you need to compile and run them on the CS Linux Lab machines. For collecting timing results, run your programs on the 30-core server `babbage.cs.pdx.edu`. To compile an OpenMP program on Linux, use the `"-fopenmp"` flag.

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`.)

Timing Measurements

Use the simple Linux timing routine `time` to conduct a rudimentary timing study, 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 real time.

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.

Compare timing results with that from the sequential programs. Do you see speedup or slowdown? If you don't see any speedup, you need to check and change your directives, until you can get better performance.

[For 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.

Requirements

You should tune your programs to achieve some speedup against the sequential programs for large array sizes. Report timing results in a nice, readable format (*e.g.* tables or charts). Summarize your experience in a short report. 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. 40% of points are to be assigned to the summary.

What to Turn In

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