CS 33 Spring 2014
Lab 4: OpenMP
DUE: Friday, June 6, 2014, 11:59:59 PM

Introduction

In this lab assignment, you will improve the performance of existing code using optimization techniques and parallelization using OpenMP.

The lab handout contains the following files:

`func.h`	**Header with typedefs and function prototypes.**
`func.c`	**Primary source file (edit and submit this).**
`filter.c`	**Filtering function file.**
`main.c`	**Source file containing main() and initialization code.**
`util.h util.c`	**Files for utility functions.**
`Makefile`	**Build script.**
`correct.txt`	**Correct output data (for default inputs).**
`seed.txt`	**Input file**
`submit`	**Script to submit a source file.**
`status`	**Script to check submission status.**
`results`	**Script to check submission results.**

You will edit and submit only **`func.c`.**

Grading

Your grade for this assignment will be proportional to the amount of speedup you achieve. For full credit, you must achieve a 4X speedup. Extra credit will be awarded for speedup beyond that amount. To get any credit, your code must produce the same output as the original code. In addition, for each memory leak in your code, your overall grade will be reduced by 1%. A memory leak is defined as a region of memory that was allocated but never freed.

Compiling and Running

To compile normally:	`make seq`
To compile with OpenMP enabled:	`make omp`
To compile using a different source file:	`make omp SRC=try2.c`
To compile with gprof enabled:	`make seq GPROF=1`
To compile with memory tracing enabled:	`make omp MTRACE=1`
To check that your output is correct:	`make check`
To check for memory leaks after a run:	`make checkmem`
To remove the executable and output files:	`make clean`

The generated executable is named `filter`. By default, it will generate output.txt file. It also prints the time taken to run filter().

If your output is not correct, a message will be printed saying your output differs from `correct.txt`. When you add the make flag `MTRACE=1`, all calls to malloc/free will be logged and saved to the file `mtrace.out`. When you run `make checkmem`, that file will be analyzed to verify that all allocated memory was eventually freed (note: does not work on Mac OS X).

Submission

To use the submission scripts, you must be logged in to lnxsrv02.

To submit a source file: `./submit func.c`

A unique cookie will be printed to allow you to identify your submissions.

To check the status of submissions in progress: `./status`

Your are limited to 1 submission in progress at a time.

To check the results of completed submissions: `./results`

You can also check the web scoreboard (see below).

It may take several minutes before your submissions run. Before submitting a file, run on your local machine or lnxsrv to check if your changes have any effect on the execution time.

Code Overview

The code performs filtering algorithm on a media file. The main function is `filter`, found in `filter.c`. The `filter` function calls 6 functions from `func0` to `func5`. Each function consists of for-loop, and your job is to try to optimize the given code and extract parallelism from each function using OpenMP.

Profiling

When optimizing a large program, it is useful to profile it to determine which portions take up the largest portion of the execution time. There is no point in optimizing code that only takes up a small fraction of the overall computations.

`gprof` is a simple profiling tool which works with `gcc` to give approximate statistics on how often each function is called. `gprof` works by interrupting your program at fixed time intervals and noting what function is currently executing. As shown earlier, to compile with `gprof` support, simply add `GPROF=1` to the make command. Then when you run `filter`, it will produce the file `gmon.out` containing the raw statistics. To view the statistics in a readable format, run `gprof` with the name of the executable: `gprof filter`.

You can also measure execution time of portions of the program using the `get_time()` and `elapsed_time()`. Check how they are used in `main()`.

Scoreboard

A full scoreboard is also available via the web, and is updated every 30 seconds:
`http://www.seas.ucla.edu/~hyunk/cs33s14/openmplab.html`