OpenMP lab

Introduction

The basic idea is to use OpenMP to make a program go faster.

Keep a log

As usual, keep a log in the file openmplab.txt of what you do in the lab so that you can reproduce the results later. This should not merely be a transcript of what you typed: it should be more like a true lab notebook, in which you briefly note down what you did and what happened. The log should help us verify that you've done the steps of this assignment, so that we can in principle reproduce them.

Files

Your 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

You should edit 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 3.5× 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 a file output.txt. It also outputs the time taken to run the filter function. If your output is not correct, a message will be output saying your output differs from correct.txt. When you add the make operand MTRACE=1, all calls to malloc and free are 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. This last step requires tools that are available on the SEASnet GNU/Linux servers.

Code Overview

The code performs filtering algorithm on a media file. The main function is filter, found in filter.c. It calls six functions func0 through func5. Each function consists of a 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, 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, e.g., gprof filter.

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

Submit

Submit the files func.c and openmplab.txt, along with any other files you think useful.

All text files should be ASCII files, with no carriage returns, and with no more than 200 columns per line. For example, the shell command

```
expand func.c openmplab.txt |
  awk '/\r/ || 200 < length'
```

should output nothing.