CS 33 Spring 2016
Lab 4: Optimization and OpenMP
DUE: **Monday**, **May 30**, 2016, 11:59:59 PM

Introduction

In this lab assignment, you will improve the performance of existing code using optimization techniques and parallelization using OpenMP. Lab4 handout is located in **lnxsrv02** at

```
/space/CS33/14S/submission_sp14/lab4_handout
```

You must log in lnxsrv02 server to get the lab4 handout. Check the tutorial in page 5~6.

The lab4 handout contains the following files:

`func.h`	Header with function prototypes.
`func.c`	**Primary source file (modify the code and submit it).**
`main.o`	Object file containing main() and initialization code.
`main_omp.o`	Object file for OPENMP
`Makefile`	Build script.
`compare`	Directory for compare files.
`dump`	Directory for output files
`submit`	Script to submit a source file.
`status`	Script to check submission status.
`results`	Script to check submission results.
`clear`	Script to clear submission in progress.

`func.c` file consists of five functions, and each function consists of some for-loops. Your job is to try to optimize the given code and extract parallelism from the code using OpenMP.

Grading

- To get any credit,
  - Your code should produce the same result as the original code.
  - Your code should provide **4X** speedup **at least**.
- For full credit, you should achieve a more than **16X** speedup.
- Extra credit can be awarded for speedup beyond that amount.

Compiling and Running

To compile sequential executable:	`make seq`
To compile with OpenMP enabled:	`make omp`
To compile using a different source file:	`make omp SRC=func_rev.c`
To compile with gprof enabled:	`make seq GPROF=1`
To check that your output is correct:	**`make check`**
To remove the executable and output files:	`make clean`

- The generated executable is named `lab4_seq or lab4_omp`.
- The OpenMP version (`lab4_omp)` is set to use 8 threads.
- The grading server uses 20 threads in the measurement.
- You **cannot** run the executable in **lnxsrv02** server.
- You should use other seasnet servers to run the executables (lnxsrv01, 03, 04, 05, 06, 07, 08). Or you can use your own Ubuntu machine.
- Run "`make check`" to make sure your code is correct. If you code is correct, then it shows nothing. If you code is incorrect, then it shows which function output is different (wrong).

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 `the executable`, 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. Ex) `gprof lab4_seq`

Submission

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

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

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

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

You are limited to **1 submission in progress** at a time.

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

It shows your last three submission results (**speedup**). For example, a score of 4.0 means your submission improves the performance by 4.0X.

From all your submissions, the "Best" score is considered as your score.

You can also check your scores in the web scoreboard.

To clear the processing submission: `./clear`

It clears your submission in progress. A cleared submission generates RUN_FAILURE error.

If your submission is delivered to the auto-grading server and your submission is correct, then it shows a number as the speedup. Otherwise, it shows the following ERRORs:

COMPILATION_FAILURE It fails to compile your submission.

- ➔ See if your code passes "**make omp**"

INCORRECT_OUTPUT Your submission generates the incorrect output.

- ➔ See if the result with "**./lab4_omp; make check**"

TIMEOUT	It takes too much time to run your submission.
SERVER_FULL	The submission queue in the server is full.
RUN_FAILURE	Unexpected Runtime Error or submission is cleared
UNDEFINED_FAILURE	Unexpected Error

Scoreboard

http://www.seas.ucla.edu/~hyunk/cs33/lab4.html
http://www.seas.ucla.edu/~hyunk/cs33/lab4_ranking.html

Note:

- **Measurement Variable**: there can be a variation in the performance measurement. The grading system is designed to minimize the variation, but send me email if you experience a large variation (e.g., more than 5%).
- **Server Maintenance**: submission might be disabled for the server maintenance. The maintenance schedule is during **the class (12 to 2pm)** on **5/18, 5/23**, and **5/25**. However, there could be unscheduled maintenance.
- It may take from **10s seconds** to about **30 minutes** to process your submission depending on the workload (queuing submissions) in the auto-grading server.
- Initially, **60-sec time gap** is set between your submissions. If the number of submission surges, then the time gap can be increased. The **status** and **clear** script reset the time gap.
- If the clear script does not work and you status keeps saying your submission **"being processed : 1"** more than an hour, then send me **your cookie number** via email.
  - Email address: roykim78@cs.ucla.edu
  - Subject: Lab4 submission stuck
  - "Your cookie number"
- "`make check`" command does not check all the output values. It does the checksum and sampled output comparison to reduce the size of comparison and dump files. It is possible (still unlikely) that you pass `make check`, but the submission result is "INCORRECT_OUTPUT". Send an email if you experience this issue.
- If there is any other trouble in submission, then send an email.

Measuring Environment

Your submission will be compiled with OpenMP and will be run in Xeon E5-2650 v3 Processor with 20 threads. The table below show the CPU specification.

Xeon E5-2650 v3	
# of Cores	10
# of Threads	20
Processor Base Frequency	2.3 GHz
Max Turbo Frequency	3 GHz
L1 Cache	10 x 32 KB 8-way set assoc.
L2 Cache	10 x 256 KB 8-way set assoc.
L3 Cache	25 MB shared cache

Tutorial

# -> comments

1. Getting lab4 handout in lnxsrv02 server

```
bash-4.3$ ssh your_id@lnxsrv02.seas.ucla.edu
bash-4.3$ cp -r /space/CS33/14S/submission_sp14/lab4_handout .
bash-4.3$ cd lab4_handout
bash-4.3$ make
gcc -o lab4_seq -O3 -lm main.o func.c

#Cannot run the executable in lnxsrv02
bash-4.3$ ./lab4_seq
Do not run lab4 in lnxsrv02 server!

bash-4.3$ make omp
gcc -o lab4_omp -O3 -lm -DOPENMP  -fopenmp main_omp.o func.c

#Cannot run the executable in lnxsrv02
bash-4.3$ ./lab4_omp
Do not run lab4 in lnxsrv02 server!
```

2. Compile or Run the executable in lnxsrv0X server (0X is 01~08 except 02)

```
bash-4.3$ ssh your_id@lnxsrv01.seas.ucla.edu
bash-4.3$ cd lab4_handout
bash-4.3$ make
gcc -o lab4_seq -O3 -lm main.o func.c

#run the executable and check its correctness
#the output shows each function's execution time, total execution time,
and a sum of all function's execution time (Main Secs).
bash-4.3$ ./lab4_seq ;make check
Initializing
Starting Compute
Fun1...
Fun1 =    3.4343
Fun2...
Fun2 =    5.4588
Fun3...
Fun3 =    3.4812
Fun4...
Fun4 =    5.1686
Fun5...
Fun5 =    6.7885
Dumping the results...
Total =    24.663, Main Secs =    24.331

bash-4.3$ make omp
gcc -o lab4_omp -O3 -lm -DOPENMP  -fopenmp main_omp.o func.c
```

```
#omp does not improve performance here because func.c is not parallelized
bash-4.3$ ./lab4_omp ;make check
#Threads= 8
Initializing
Starting Compute
Fun1...
Fun1 =    3.4334
Fun2...
Fun2 =    5.4507
Fun3...
Fun3 =    3.4810
Fun4...
Fun4 =    5.1698
Fun5...
Fun5 =    6.7697
Dumping the results...
Total =    24.643, Main Secs =    24.305
```

3. Submit your file in lnxsrv02 server

```
bash-4.3$ ssh your_id@lnxsrv02.seas.ucla.edu
bash-4.3$ cd lab4_handout

#submit your func.c, then it returns your cookie
bash-4.3$ ./submit func.c
Waiting for confirmation of submission ...
Your cookie: 10269
Submitting your file ...

#your submission is processing
bash-4.3$ ./status
Waiting for response ...
Number of jobs submitted by you currently being processed : 1

#few minutes later, measurement is finished
bash-4.3$ ./status
Waiting for response ...
Number of jobs submitted by you currently being processed : 0


#results script returns your speedup
bash-4.3$ ./results 10269
COOKIE    SCORER            OLDEST           OLDER             CURRENT
10269        .99            X                 .99              .99
```

Tip: It is convenient if you **use two ssh terminals**: one for submission (lnxsrv02) and one for run the executable (lnxsrv0X).