ECE454, Fall 2020
Homework5: Performance Optimization
Assigned: Nov 19th, Due: Dec 6th, 11:59PM

TAs: Mike Qin (mike@eecg.toronto.edu) and Jack Luo (jack.luo@mail.utoronto.ca)

1 Introduction

OptsRus will now put all of its optimization knowledge to work in its biggest challenge to date: optimizing an artificial life simulator. Given original C code from the client, OptsRus can use parallelize the code to exploit multicore machines, as well as perform other optimizations to improve performance.

2 Background

You will be optimizing Conway's "Game of Life" (GoL), a famous, simple algorithm for computing artificial life (cellular automota), that through very simple rules can generate very complex and interesting behavior. You can read more about the history and details of GoL here:

https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

You can also play with a GoL emulator here:

http://www.bitstorm.org/gameoflife/

3 Setup

Copy and extract the GoL source code from /cad2/ece454f/hw5/gol.tar.gz into a protected directory in your UG account. GoL should build by simply typing make. Input and output files are in a simple image format called .pbm. The executable initboard can create an arbitrary-sized input file by running:

```
initboard <num_rows> <num_cols> <generated_file_name>
```

The executable gol is run as follows:

```
gol <num_iterations> <infilename> <outfilename>
```

If you want to visualize an input or output file, you can convert it to a jpg for viewing in a browser with the command:

```
convert filename.pbm filename.jpg
```

Visualizing can potentially help you think of opportunities for optimization.

4 Optimizing

Your main task in this homework is to speedup GoL using everything you learned in the course. You can parallelize and use as many threads as you like. You are also free to do any optimization you wish on your program, including adjusting the Makefile and/or compilation flags.

4.1 Rules

- You must work individually
- Given some input file and some number of iterations, your program must produce the identical output file as the original unmodified GoL program.
- Your program must be be faster than the reference implementation
- The code must be your own, i.e., you cannot incorporate GoL-specific acceleration code or libraries written by others. However, you can study them and come up with a similar implementation. You must be able to explain exactly what is happening with the code when asked by the TA.
- Your program must be able to execute on both the tester machine and the ug lab machine.

4.2 Assumptions

- You can assume the dimensions of all input game boards to be N x N, where N is a power of two.
- You can assume a maximum world size of 10000x10000. However, your program should at least exit gracefully if a larger size is attempted.
- Your code does not have to handle cases for worlds sizes less than 32x32.

5 Measurement

For this homework your implementation should be measured from start to finish, using /usr/bin/time. In other words, you will include the entire computation in your measurement, including reading and writing files, initialization, and thread creation or other overheads associated with parallelization. We will measure your speedup only by running GoL for 10,000 iterations on the 1k.pbm file provided. Please test your speedup with the command below which provides the "wall clock" timing of your program in seconds. Afterwards, submit to the autotester to compare against others. If you find this execution is prohibitively long for quick optimization prototypes, feel free to experiment with smaller files, or fewer Life generations, but recall that your speedup is based on the configuration below.

```
/usr/bin/time -f "%e real" ./gol 10000 inputs/1k.pbm outputs/1k.pbm
```

6 Correctness

As stated above, for the same input, the output of your optimized/parallelized program must match the output of the original program. We will test the correctness of your code using several input files of varying sizes and initial configurations. We have provided one (read-only) output file 1k verify out.pbm, which you can use to verify your program in the "speedup" configuration above.

Verify correctness by running

```
diff outputs/1k.pbm outputs/1k_verify_out.pbm
```

Be sure to frequently test/debug your current program state on many different inputs. Consider generating a few new input files (and the original output file) beyond those we have provided.

7 Marking Scheme

The total available marks are divided into 2 portions (70% + 30%). The first portion is for non-competitive portion and modest improvement over reference implementation. You will receive full marks if your solution compiles and runs successfully and is at least as faster as the class average (excluding top two speedups) on the autotester during the first week. The minimum speedup to obtain non-competitive will be announced on Piazza after the average has been determined by the TA. A text file containing a short description of how your program and your optimization work, and what you have tried need to be included as well. The report is only for reference purpose to support you if we suspect you are cheating.

The second portion is competitive. This lab is designed to have significant room for performance optimizations. For this portion, we will be using an automated scoring system similar prior labs. Once you submit your work using the usual `submitece` command, your submission will be placed onto a queue for auto-grading. The competitive portion mark will be cored based on the following formula:

```
category mark = (yourspeedup-worstspeedup)/(topspeedup-worstspeedup)
```

Marks will only be assigned if the program run successfully!

8 Submission

Please submit your report in .pdf or text format, as well as a tarball of all source code and Makefiles needed to compile and run your GoL solution. Suppose your source code is in the directory src (you can name the directory whatever you wish). Enter src, remove the input/output images and directories, clean all object/binary files in the directory, and compress.

```
cd src
make clean
cd ..
tar -cvvf gol.tar src/
gzip gol.tar
submitece454f 5 gol.tar.gz report.txt
```

Note:

- You must modify your team information into your teaminfo.txt file
- You must not modify inputs/*.pbm and outputs/1k_verify_out.pbm files
- Your source code dir must be named src, and your submission must be named gol.tar.gz

9 Autotester Machine Specs

Server Specs:

- Output of "uname -srvmo": Linux 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018 x86_64 GNU/Linux
- GCC version: gcc (Ubuntu 7.3.0-16ubuntu3) 7.3.0
- RAM: more than you will ever need
- Output of lscpu:

```
$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              12
On-line CPU(s) list: 0-11
Thread(s) per core:  1
```

```
Core(s) per socket:  1
Socket(s):           12
NUMA node(s):        2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               45
Model name:          Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz
Stepping:            7
CPU MHz:             2200.000
BogoMIPS:            4400.00
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            15360K
NUMA node0 CPU(s):   0-5
NUMA node1 CPU(s):   6-11
```