ECE 459: Programming for Performance
Assignment 2

Jon Eyolfson

February 5, 2012 (Due: February 27, 2012)

Important Notes:

- **Make sure your working directory is called "assignment-02"**
- **Make sure you run your program on** `ece459-1.uwaterloo.ca`
- **Use the command "**`OMP_NUM_THREADS=4;export OMP_NUM_THREADS`**" to set 4 threads**
- **The guidelines were set by running on** `ece459-1` **with 4 threads**
- **Run "**`make report`**" and "**`make tar`**" and submit your** `tar.gz` **file**
- **Make sure the image from parts 1 and 2 are the same as the sequential version**

Automatic Parallelization (40 marks)

Benchmark your sequential program with the number of iterations (`-i`) such that the execution time is approixmately 10 seconds. On `ece459-1` I used `-i 9001` (it's over 9000!) to get the intended runtime.

Your first task is to modify your program so you can take advantage of automatic parallelization **(do not make any major changes)**. Make all your changes to `mandelbrot_auto.c`. I put SolarStudio 12.3 on `ece459-1` and use that compiler with the relevant flags in the `Makefile`. Your output when completed should look something like the following (the line numbers don't have to match, but the first outermost loop should be parallelized):

```
Compiling Part 1 Automatic Parallelization
/opt/oracle/solarisstudio12.3/bin/cc -O3 -xautopar -xloopinfo src/mandelbrot_auto.c
    -o bin/mandelbrot_auto
"src/mandelbrot_auto.c", line 13: not parallelized, loop has multiple exits
"src/mandelbrot_auto.c", line 40: PARALLELIZED
"src/mandelbrot_auto.c", line 41: not parallelized, not profitable
"src/mandelbrot_auto.c", line 50: not parallelized, loop has multiple exits
...
```

Clearly and concisely explain your changes. Explain why the code would not parallelize initially and why your changes are correct and preserve the behaviour of the sequential version. Run your benchmark again, using the same `-i` and calculate your speedup. Explain why the speedup is so poor (keep it mind in the next part the speedup will be greater than 3.5).

- **Minimum expected speedup:** 2
- **Initial solution speedup:** 2.3

Manual Parallelization with OpenMP (30 marks)

Now, it's time to show the compiler who's boss. Copy your `mandelbrot_auto.c` file over to `mandelbrot_omp.c`. Now, using **only** OpenMP pragma's increase the performance of the program over automatic parallelization. Benchmark your manual parallelization with OpenMP and calculate the speedup over the sequential version. Using your explanation from part 1 (as to why the performance was so bad) explain why your manual parallelization so drastically improved performance. Be clear and concise.

- **Minimum expected speedup:** 3.5
- **Initial solution speedup:** 3.7

Using OpenMP Tasks (30 marks)

We saw briefly how OpenMP tasks allow us to easily express some parallelisms. In this part, we apply OpenMP tasks to the naive Fibonacci number function. Benchmark the sequential version with a number that executes in approximately 10 seconds (47 on `ece459-1`, sadly not 42). Now, try to run the provided version with OpenMP tasks (let it run for at least 20 seconds to prove a point, let it run to completion to drive the point home), it should be much slower than the sequential version. Clearly and concisely explain why the performance is so dreadful.

Next, we'll modify the code so there's actually a speedup. Without modifying the naive calculation (it has to remain recursive with no caching, etc) make it so you get some improvement. A Google search reveals that a cutoff that decides whether or not to create tasks should work. The only modifications in this case should be a variable and an if statement. Benchmark your modified program and calculate the speedup. Clearly and concisely explain why your changes improved performance.

- **Minimum expected speedup:** 1
- **Initial solution speedup:** 1.33 (1.75 on my laptop)

Setup

Same as Assignment 1, the provided material is avaiable here: `http://ece459.eyolfson.com/media/assignments/provided-assignment-02.tar.gz`. Please use the provided `Makefile` and do not modify it. Grading will be done running `make`, running your programs, looking at the source code and reading the report.