CMDA 3634 Fall 2017 Homework 04

Kevin Jiang

November 8, 2017

You must complete the following task by 11:59pm on 11/07/17.

Your write up for this homework should be presented in a LaTeX formatted PDF document. You may copy the LaTeX used to prepare this report as follows

1. Click on this link
2. Click on Menu/Copy Project.
3. Modify the HW04.tex document to respond to the following questions.
4. Remember: click the Recompile button to rebuild the document when you have made edits.
5. Remember: Change the author
6. Instructions for assignment referred to in **Q1** and **Q2** available on Canvas.

Each student must individually upload the following files to the CMDA 3634 Canvas page at https://canvas.vt.edu

1. `firstnameLastnameHW04.tex` LaTeX file.
2. Any figure files to be included by `firstnameLastnameHW04.tex` file.
3. `firstnameLastnameHW04.pdf` PDF file.
4. `ompMandelbrot.c` and `opi.c` text file with student code.
5. `network.c` text file with student code (if you choose to attempt the extra credit).

Other notes:

1. If you are attempting the extra credit problem, a `network.c` serial code is available on the course GitHub repository. Note this program only takes one input, the data file, and prints the PageRank of all nodes.

You must complete this assignment on your own.

60 points will be awarded for a successful completion.
Extra credit will be awarded as appropriate.

Q1 *(30 points) Computing Pi in Parallel with OpenMP.*

30 points will be awarded for attending class on October 18th and completing the "computing pi in parallel using OpenMP" assignment. Follow the instructions in the assignment PDF closely (available on Canvas). Remember to push the code to a L15 sub directory in a repository that Dr. Warburton and William Winter have access to. (NOTE: If you have already shared a private repository with tcew and wmwinter, you do not need to share a new directory. A L15 sub-directory in the already shared course repository will suffice.)

Copy and paste the contents of the batch script file used to submit a job to newriver into your LaTeX report. Submit code to canvas (as well as pushing code to your GitHub repository).

Script code:

```
#! /bin/bash
#
#PBS -l walltime=00:05:00
#PBS -l nodes=1:ppn=12
#PBS -W group_list=hokiespeed
#PBS -q normal_q
#PBS -j oe
cd $PBS_O_WORKDIR

module purge
module load gcc
gcc -o opi opi.c -lm -fopenmp
```

Q2 *(30 points) Generating Mandelbrot Sets in Parallel with OpenMP.*

30 points will be awarded for attending class on October 23rd and submitting "adding threading to a Mandelbrot set generator" code to GitHub. Follow the instructions in the assignment PDF closely. Remember to push the code to a L16 sub directory in a repository that Dr. Warburton and William Winter have access to. (NOTE: If you have already shared a private repository with tcew and wmwinter, you do not need to share a new directory. A L16 sub-directory in the already shared course repository will suffice.)

Copy and paste the contents of the batch script used to run on newRiver. Present scaling study results in a table in your LaTeXreport. Plot results in a graph and include the graph in your LaTeX report. (Review homework 1 tex code for help including graphics in LaTeX). If you choose to attempt the extra credit portion, present the strong scaling study results in the same manner (table and figure in LaTeX report). Submit code to canvas (as well as pushing code to a GitHub repository).

Script code:

```
#! /bin/bash
#
#PBS -l walltime=00:05:00
#PBS -l nodes=1:ppn=12
#PBS -W group_list=hokiespeed
#PBS -q normal_q
#PBS -j oe
cd $PBS_O_WORKDIR

module purge
module load gcc
```

```
gcc -O3 -o mandelbrot mandelbrot.c png_util.c -I. -lpng -lm -fopenmp

for n in {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}
do
./mandelbrot 4096 4096 $n
done
```

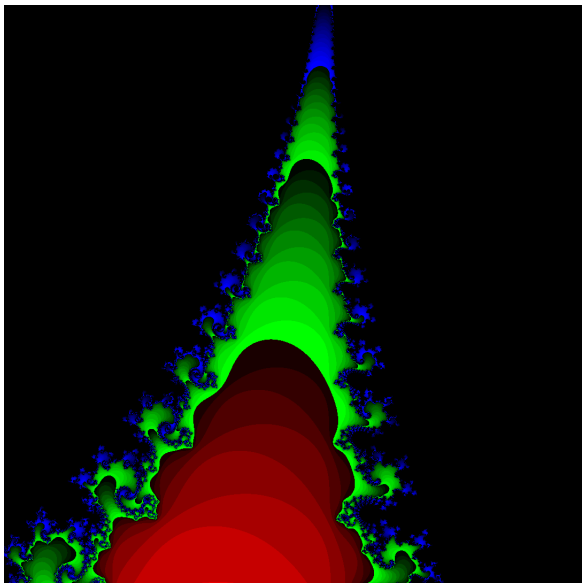

Figure 1: Mandelbrot graph

Q3: Extra Credit *(60 points) Making the PageRank Algorithm Parallel with OpenMP.*

A `network.c` serial code is available on the course GitHub repository. You are free to modify the `network.c` code to add the desired functionality to complete this extra credit task.

There are two main areas where the PageRank algorithm can be parallelized. These two areas are in the method performing the PageRank update and in the method computing `diff`.

1: *(10 points) Setup.*

Add an include statement to include the OpenMP library header file. In the main method, set the number of threads OpenMP uses to be 4.

2: *(25 points) Parallelizing the Update Method.*

Add a parallel for loop pragma to the outer for loop in the `updatePageRank` method. Be mindful of the variables: makes variables private as necessary.

The `updatePageRank` method may look strange to you. Some additional work is done in the `networkReader` method to set up an efficient update method that will not cause conflicts when parallelized with openMP. (Extra extra credit will be provided to students who read through the code, summarize what it is doing, and explaining what type of issue this update strategy avoids and why it sucessfully avoids such issue(s).)

3: *(25 points) Parallelizing the Compute Diff Method.*

Add a parallel for loop pragma to the `computeDiff` method. You will need to add a reduction clause to ensure the method computes `diff` correctly.
