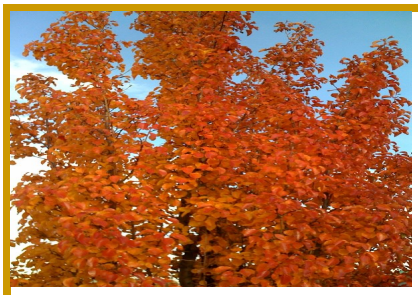

(../../index.php)

CS 475: Parallel Programming

Computer Science Department

(http://www.cs.colostate.edu)

Fall 2016

Assignment 2 - Sequential and OpenMP Prime Sieve

Colorado State University

(http://welcome.colostate.edu/)

Objectives

The objective of this homework is to step wise parallelize a Prime Sieve Program, and record the performance of your programs on a Capital machine (http://www.cs.colostate.edu/~info/machines) (Albany to Trenton in the list), and to experimentally determine the gains you get with 1, 2, 3, 4, 5, 6, 7, and 8 threads.

You will submit sieve 1, an optimized sequential code and sieve 2 a parallelization of sieve 1 with their performance analysis, sieve 3, a sequential, blocked code and sieve 4 a parallelization of sieve 3 with their analysis.

Download and untar PA2.tar (PA2.tar). You will need to update your makefile for each parallel program you are writing. Do not set the number of threads in the program, set it at the Linux OS level (using the export or setenv command). Otherwise we cannot run your program for different numbers of threads. See the Mandelbrot lab (https://www.cs.colostate.edu/~cs475/f16/more_assignments/L1/instructions.html). Write your main report in pdf format.

- [**Sieve 1**] Your first task is to improve the sequential program provided, creating a program sieve1.c where you apply the sequential optimizations discussed in the lecture:
  - marking using a stride rather than testing every element for divisibility,
  - starting the marking off at the square of the current prime,
  - marking off multiples of only those primes that are no more than sqrt(n), and
  - marking and storing only odds.

  Analyze the execution time of sieve1.c and compare it to the performance of sieve.c. Report performance for n = 100,000, n = 200,000, and 300,000. Analyse the order of magnitude improvement of your code.
- [**Sieve 2**] Your next program sieve2.c should be a parallelization of sieve1. You should experiment with parallelizing one or more of the loops, and report how the speedup varies as a function of the number of threads (1, 2, 3, 4, 5, 6, 7, 8) on a Capital machine, for n=500,000,000, n=1,000,000,000, and

n=1,500,000,000. If you do not get speedup, explain the reasons, as best as you can. Think about PA1. For which codes did you not get good speedup? Why was that there? Is there a relationship?

- [**Sieve 3**] Now return to your sequential program Sieve 1, and block the loop to improve locality as discussed in class. Make the blockSize as a program parameter that can be passed as a command line argument. For this program, you should again analyze the performance gains achieved by blocking for the values of *n* given above. Play with blockSize and report which gives you good performance. What performance do you get when blockSize is 1,000,000 ?
- [**Sieve 4**] is a parallelization of Sieve3. Report the speedup as a function of the number of threads. Ask yourself: which loop can be parallelized?

How to Submit:

- Make a README file that contains the best blocksize from your exploration for the 3 values of n(0.5, 1 and 1.5 billion). Please make the format of the README as shown in the sample (README) so as to pass through the automatic grading.
- Submit sieve1.c, sieve2.c, sieve3.c and sieve4.c, README file and your makefile in a tar file: PA2.1.tar(PA2.2.tar) either through the website or using the provided checkin command.
- Submit your report (PA2.R.pdf) through Checkin.

Preliminary Testing:

- Your executables must be named sieve1, sieve2, sieve3, sieve4.
- Do not change the format and the order of printing statements. Example:

```
$./sieve1 100
There are 25 primes less than or equal to 100
First three primes:2 3 5
Last three primes:97 89 83
elapsed time = 0.000046 (sec)
$./sieve2 100
There are 25 primes less than or equal to 100
First three primes:2 3 5
Last three primes:97 89 83
elapsed time = 0.000046 (sec)
$./sieve3 100 10
There are 25 primes less than or equal to 100
First three primes:2 3 5
Last three primes:97 89 83
elapsed time = 0.000046 (sec)
$./sieve4 100 10
There are 25 primes less than or equal to 100
First three primes:2 3 5
Last three primes:97 89 83
elapsed time = 0.000046 (sec)
```

- The order and format should be the same even when your code is running on multiple threads.
- Note: The execution time in the example is a sample one. It is not the actual execution time after making the changes. The third parameter (10) in running sieve3 and sieve4 is a sample blockSize value.

How to tar files:

- Enter the directory above the directory that contains the files you want to submit.
- Use the command:

```
tar cvf PA2.1.tar [dir(PA2.1/PA2.2)]
```

You can see the Grading Rubric (PA2_grading_rubric.pdf) here. Good luck and have fun!
