In his exalted name

Amirkabir University
of Technology

Parallel Processing

Fall 2015

Assignment No. 3

Total Points: 100

Due Date: December 25, 2015

Problem Statement

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.[1] Prime numbers play a pivotal role in current encryption algorithms and given the rise of cloud computing, the need for larger primes has never been so high. This increase in available computation power can be used to either try to break the encryption or to strength it by finding larger prime numbers.[2] Based on the importance of prime numbers you are supposed to develop a *distributed prime sieve* program using C/C++ and *MPI.* The problem is conceptually simple, but computationally intensive, and important in cryptography/security.[3]

The program is required to identify all prime numbers less than some (large) limit value. For example if the input to your program was 10 the output must be 4 because we have four prime numbers less than ten (2, 3, 5 and 7). To find all prime numbers up to some limit value N, this algorithm works as follows:

1. Create the sieve -- an array indexed from 2 to N. (This can be an array of Booleans or integers or anything else that supports random access and provides entries that are easy to "mark".)
2. Set k to 2.
3. Loop:
    1- In the sieve, mark all entries whose indices are multiples of k between k^2 and N, inclusive.
    2- Find the smallest index greater than k that is unmarked and set k to this value.

   Until $k^2 > N$.
4. The indices of the unmarked sieve entries are prime numbers.

Note that you need to implement this sequential algorithm so that you compare its runtime with the distributed version.

Next you are supposed to develop a parallel version of this algorithm that distributes "the sieve" across different processes, to see what kind of speedup and efficiency it offers. A document that describes the way to parallelize the sieve of Eratosthenes can be found in the course page or can be downloaded for here.

To setup an environment for developing MPI programs in Microsoft Windows please refer to following webpage:
How to compile and run a simple MS-MPI program.

[1] https://en.wikipedia.org/wiki/Prime_number
[2] https://github.com/carlosmccosta/Distributed-Prime-Sieve
[3] http://cs.calvin.edu/curriculum/cs/374/homework/MPI/06/

Variety of approaches has been proposed to solve this problem, beside the given document that explains an approach to parallelize the sieve, for more information you can Google the following keywords: Sieve of Eratosthenes, Segmented Sieve of Eratosthenes, Sieve of Atkins (might be useful), Fast Marking, Wheel Factorization. Of course, using any of these techniques as well as any other endeavor to reach a better implementation will be considered in your grade.

Deliverables

The executable file should get a command-line parameter as its input argument: The limit value.

E.g. the command below means that the limit value is 10,000,000 and the number of processes is 4.

```
mpiexec –n 4 prime.exe 10000000
```

(The output is 664,579; we have 664,579 prime numbers less than 10^7)

Report files should have a brief description of parallelization methods you used, the results and your analysis of them. The result section has time and speed-up tables for very large limit values.

Please do not forget to mention the specification of the platform you used to run your program.

Submission

Upload your source code(s), executable file(s) along with your report PDF in an archive file to our course webpage, named in the following format:

[Parallel Programming] [HW3] <First Name> <Last Name> - <Student ID>

Ex. [Parallel Programming] [HW3] Ahmad Siavashi – 94131100

Bonus

Executing your program on remote hosts will be appreciated and will have a significant impact on your overall grade but the experiment must be directly presented to the T.A.

The deadline is next Friday (December 25, 2015), 11:55 PM. There is a delay penalty of -5% per day.

Good Luck ☺