CS5300 - Parallel & Concurrent Programming
Spring 2022
Comparing Different Parallel Implementations for Identifying Prime Numbers
Submission Date: 16th January 2022, 9:00 pm

Goal: The goal of this assignment is to implement different parallel implementations for identifying prime numbers and compare their performance. Implement these algorithms in C++.

Details. The textbook has described a method to identify all the prime numbers less than some N which balances the work load among the various thread using a shared counter. Each thread gets a dynamically determined number of numbers to test. It is defined in figure 1.2 on page 4 of the book. It can be seen that method getAndIncrement() of counter object is crucial. It has to be implemented in a thread-safe manner. The implementation given on page 5, Fig 1.3 is an implementation which, as the book says, is clearly wrong. So, you have to come up with a correct way of implementing this method. Let us denote this technique as *dynamic allocation method* or *DAM*.
There is another logical but static way (discussed in the class) to identify all the prime numbers while having the load among the threads balanced. Given that you have to find the primes less than N and there are m threads. You can pre-assign the numbers to verify to each of the m threads as follows: Numbers $1, m+1, 2*m+1, ...$ will be assigned to thread 1; Numbers $2, m+2, 2*m+2, ...$ will be assigned to thread 2 and so on. Let us denote this technique as *static allocation method-one* or *SAM1*.

You have to implement both the algorithms DAM and SAM1 **in C++**. You can use any standard algorithm for testing whether a number is prime or not.

Input: The input to the program will be a file, named inp-params.txt, consisting of the parameters described above:n, m where $10^n = N$. Here N is the number below which you to find the number of primes and m is the number of threads that needs to be created.

Output: You program should output the following files:

1. "Primes-DAM.txt" and "Primes-SAM1.txt". Both these file contain all the primes less than N produced by DAM & SAM1 algorithms respectively. The prime numbers in the file are separated by space as follows: $\langle PrimeNumber1\ PrimeNumber2\ ...\rangle$. Please follow this format as the TAs will compare the output of your program with the file they already have. If it is not in this format you may not be given the marks.

2. "Times.txt" which contains the time taken to compute by these techniques. It will contain a single file consisting of the times taken as follows: $\langle Time1\ Time2\rangle$ which are the times taken by DAM & SAM1 algorithms (in whatever suitable unit) respectively.

Report: You have to submit a report for this assignment. The report should first explain the design of your program while explaining any complications that arose in the course of programming.

The report should contain a comparison of the performance of DAM & SAM1 algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph. Specifically, the report should contain the following graphs:

1. Time vs Size, N: In this graph, the y-axis will have the time taken by the prime number algorithms, DAM & SAM1. The x-axis will be the values of n varying from 3 to 8 in the increments of 1. Note that $N = 10^n$. Have m fixed to be 10 for all these experiments.

2. Time vs Number of threads, m: In this graph like the previous graph, the y-axis will have the time taken by the prime number algorithms, DAM & SAM1. The x-axis will be the values of m, number of threads varying from 5 to 35 in the increments of 5. Have to be N fixed at 10^8 for all these experiments.

It can be seen that both these graphs will have two curves corresponding to DAM & SAM1 algorithms. Also, please explain the reason for your results.

Deliverables: You have to submit the following:

- The source file containing the actual program to execute. Please name it as Src-<rollno>.cpp. Please follow this convention. Otherwise, your program will not be evaluated. If you are submitting more than one file name it as SAM1-<rollno>.cpp, DAM-<rollno>.cpp and SAM2-<rollno>.cpp.

- A readme.txt that explains how to execute the program.

- The report as explained above.

Zip all the three files and name it as ProgAssn1-<rollno>.zip. Then upload it on the google classroom page of this course. Submit it by above mentioned deadline. Please follow the naming convention. Otherwise, your assignment will not be evaluated by the TAs.

Evaluation: The break-up of evaluation of your program is as follows:

1. Program Design as explained in Report - 30%

2. The Graphs obtained and the corresponding analysis shown in the report - 30%

3. Program Execution - 30%

4. Code Documentation & Indentation - 10%.

Please make sure that you your report is well written since it accounts for 60% of the marks.

Extra Credit: One can clearly see that SAM1 is not efficient. A thread given an even number to test for prime can immediately return as opposed to a thread given an odd number. So, one can develop a new static allocation method, say *static allocation method 2* or *SAM2* in which all the threads are allocated only odd numbers to test for primality. So for extra-credit, please develop such an algorithm. Explain its working clearly in your report.

Then, evaluate the performance of SAM2 by comparing it against DAM and SAM1. If you are able to describe your algorithm clearly in the report and get correct results for implementation then you can obtain an extra-credit of 10%.

Please see the instructions given above before uploading your file. Your assignment will NOT be evaluated if there is any deviation from the instructions posted there.

All assignments for this course has the late submission policy of a penalty of 10% each day after the deadline for 8 days Submission after 8 days will not be considered.

Kindly remember that all submissions are subjected to plagiarism checks.