Parallel and Concurrent Programming

Assignment 1

Irfan Ali

CS16BTECH11019

Problem Statement :

Implement different parallel implementations for identifying prime numbers upto a given number and comparing their performances. I have implemented 3 different parallel implementations.

1. **DAM :** Dynamic Allocation Method. This method creates the threads and dynamically assigns the numbers to be checked by each thread.

2. **SAM1 :** Static Allocation Method 1. This method creates the threads and statically assigns the numbers to be checked by each thread based on its thread id.

3. **SAM2 :** Static Allocation Method 2. This method is an optimized version of SAM1. In this method we do not assign any even numbers to any thread as we know that all even numbers except 2 are not prime.

Implementation :

1. **DAM :** In this method we create a counter which is shared by all the threads. The counter is initialized to 1. Each thread accesses the counter value, checks if it is prime or not and then increments the counter value. This is continued until the value of counter reaches the limiting value. To make this action thread safe, I have used *mutex* locks. Every time a thread wants to access the counter value, it has to acquire a common lock shared by threads. Once it acquires the lock, it stores the counter value into a temporary variable, increments the counter value

and then releases the lock. This makes the program thread safe. The counter has been implemented as a C++ Class. This class has two private variables : *value, lock.* The class has a public function *getAndIncrement( )* which in a thread safe manner returns the counter value and increments it.

2. **SAM1 :** In this method, we assign numbers to each thread based on its thread id. The thread ids range from 1 to m (m is the number of threads). Let thread id of a thread be *i*. That thread has to check the numbers *i, i+m, i+2m ….* This ensures that all the numbers are being checked for being prime or not.

3. **SAM2 :** This method is very similar to SAM1. In SAM1, we have some threads which check for only even numbers which makes the thread useless. To prevent this we assign the number to each thread based on their thread id, but now in a little different manner to leave out the even numbers. Let a thread have an id *i*. That thread has to check for numbers *2i - 1, 2i - 1 + 2m, 2i - 1 + 4m ….* This ensures that all the numbers are being checked except for the even numbers.

All the threads add the prime numbers they checked to a shared vector. This is also done in a thread safe manner by using locks. Later the vector is sorted and it is written the output file.

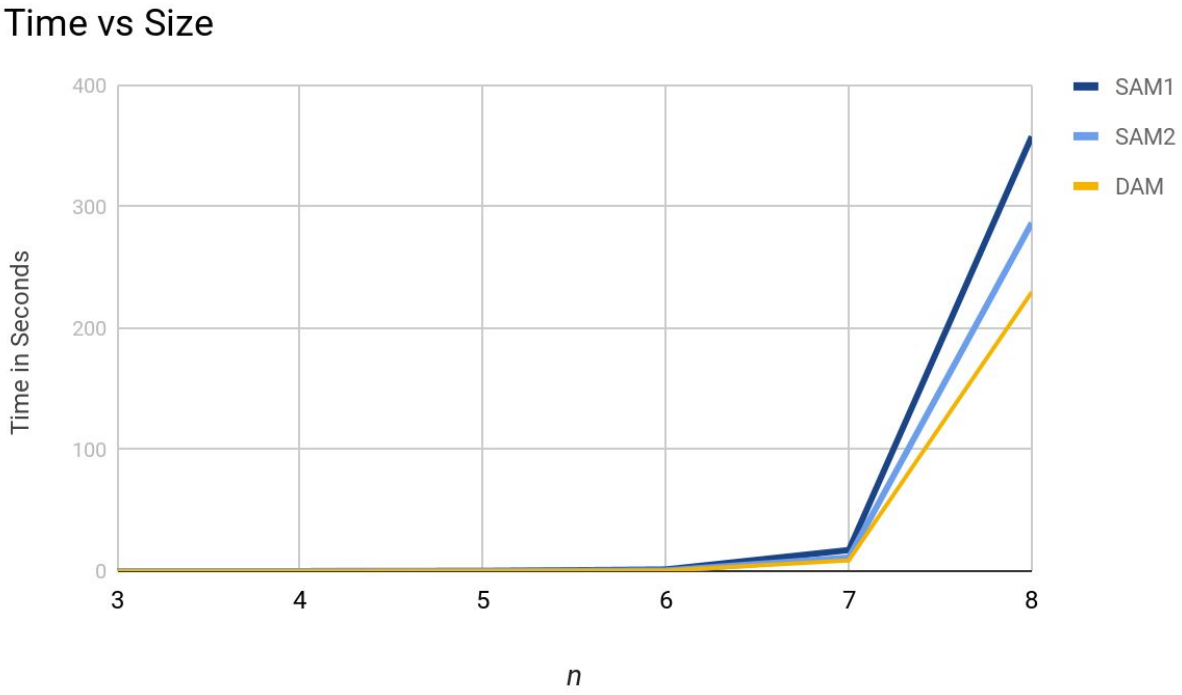
Problems while Implementation :

Main problems were faced in multithreading like passing arguments to the thread function using pthread library. Also maintaining concurrency without issues was also a little troubling. Later formulation of SAM2 method was a little hard.

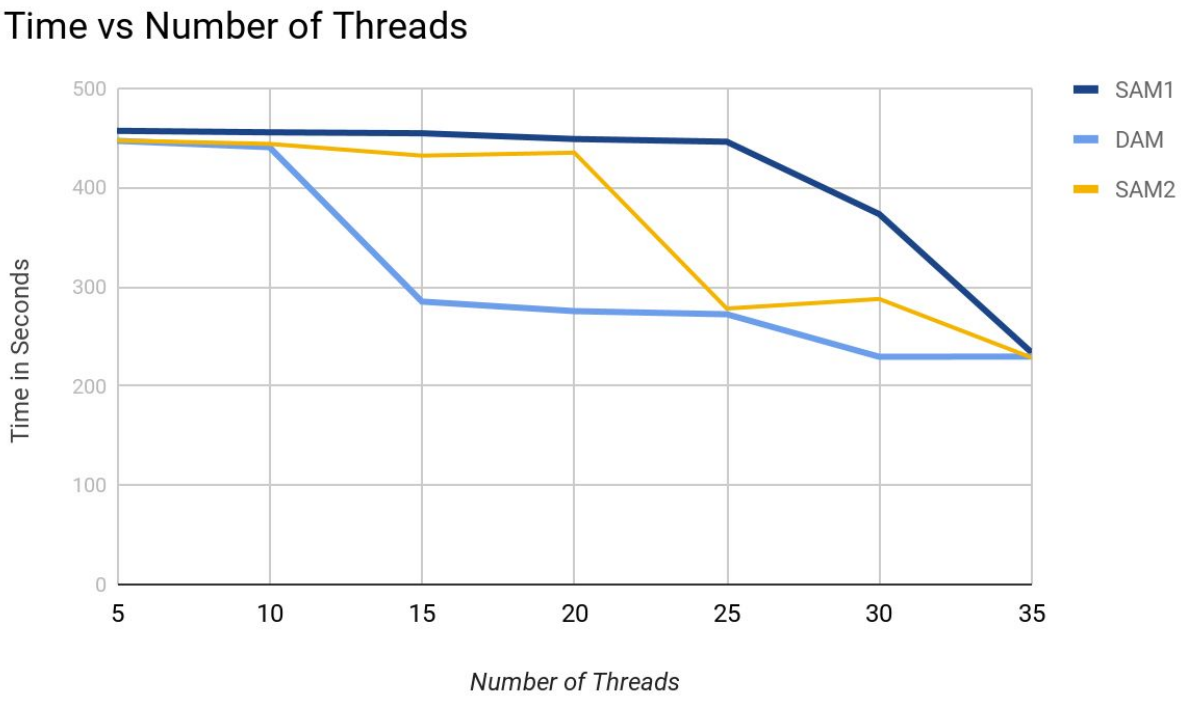
Apart from all these, no major issues were faced.

Performance Comparision :

1 . Time vs Size, N:



2. Time vs Number of Threads, m :



CONCLUSION

As we can see, DAM performs the best as it checks which thread is free and allocates it a number to check. In this way the load is balanced. In the other two methods, load is also balanced but it is not very efficient.

System Details :

Operating System : Ubuntu 16.04

Processor : Intel i7 8700