Parallel & Concurrent Programming

Assignment Report

Comparing Different Parallel Implementations
for
Identifying Prime Numbers

Rohit Kapoor
(cs21mtech12011)

Program Design

The program involves three function calls namely SAM1, SAM2 and DAM. It records the time taken by each of the functions and stores the outputs (i.e. the prime no's evaluated) in the corresponding text files.

Input:

The input to the program is a text file named "inp-params.txt" which inputs the values of N and M which denoted the exponent of 10 below which the nos have to be checked and the no of threads used respectively.

Output:

The output of the program consists of the files "Primes-SAM1.txt", "Primes-SAM2.txt","Primes-DAM.txt" and "Times.txt", which output the prime numbers in the given range and the corresponding times taken to evaluate them.

Common function:

The function isPrime() is common to all the functions it evaluates whether the number provided as input is prime or not and it taken sqrt(n) time for each no.

Static Allocation 1

This method creates M threads having id 0 to M-1, and produces numbers in the range 0 to 10^n.
0 is excluded from the range of numbers checked and 1 is not a prime no (checked by the isPrime method).

The numbers are produced by the for loop in the order m+j,2*m+j... (where j is the id of the thread) and it covers all the numbers in the range.

Dynamic Allocation

In this method the numbers are given to the threads dynamically, a global object of the class Counter is created and the function. M threads are created and each thread is assigned an identity value from 0 to M-1.

GetandIncrement is called by each thread for getting the value of the no whose primality has to be tested.

The getandIncrement method is implemented in a thread safe manner so that is does not lead to the race condition.

Static Allocation 2

In this method the numbers are given to the threads in a static manner but only the odd numbers are considered . M threads are created and each is assigned an idenity value 0 to M-1.

If N>=1 then 2 is also included in the primes-file because it is the only even prime number.

The numbers are assigned in the order 2*m +2*j+1 where j denotes the id associated with the thread.

The numbers are then stored in the Primes-SAM2.txt file.

Results

Graph for time vs the exponent (n)

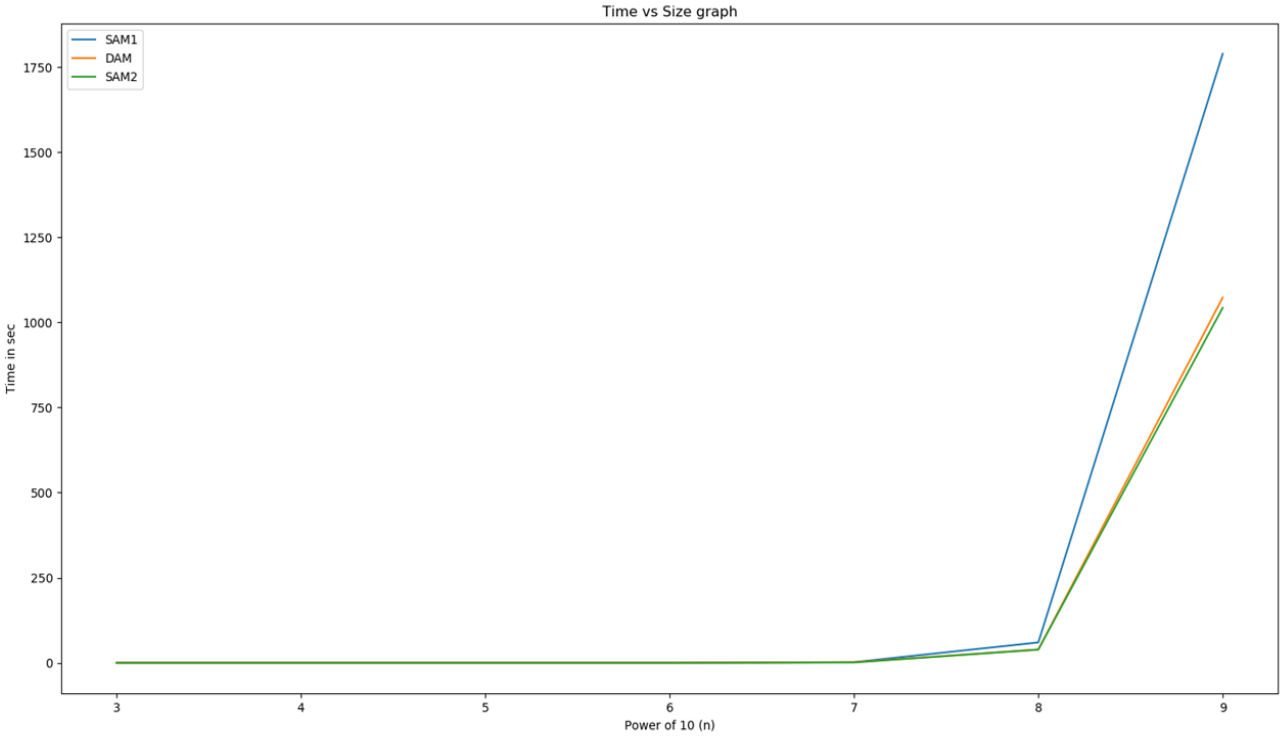

Fig 1: Plot between exponent and time taken keeping threads equal to 10

The graph was plotted between the power of 10 taken (denoting the no of primes) and the amount of time measured in seconds, the number of threads were fixed at 10. As the value of n increases so does the amount of time taken by all the three algorithms.

It can be observed that amount of time taken by different algorithms will be in the order SAM1>DAM>SAM2.

Now SAM2 has lesser number of values to check so despite static allocation it is perfroming better than DAM. It can be expected that the dynamic allocation of numbers would take lesser time compared to static allocation because if one thread is busy while checking the primality of the number then the other thread can proceed if it is free.

Graph for Time taken vs number of Threads

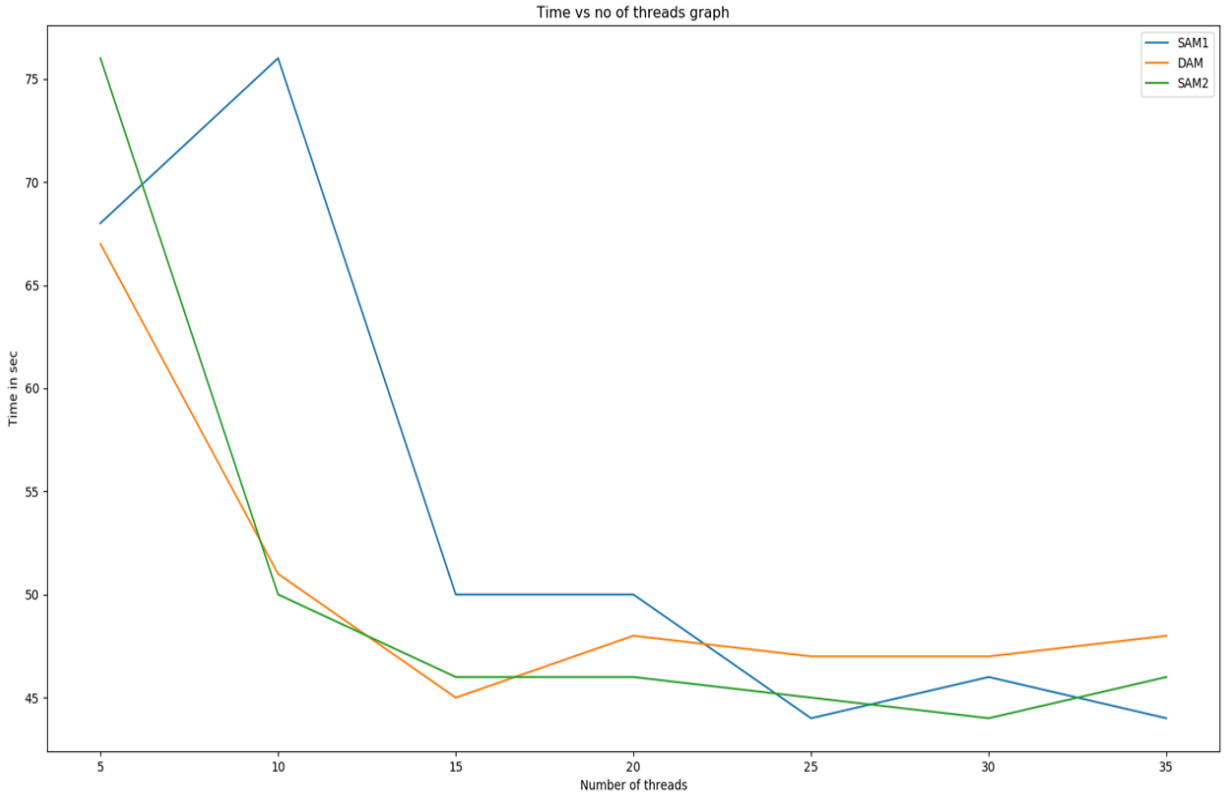

Fig 2. Plot between no of threads and time taken in sec keeping exponent at 8

The graph was plotted between the number of threads and the corresponding time taken by the algorithms while fixing the exponent to 8. Now it can be observed that as the number of threads increases the amount of time taken by the algorithm decreases rapidly and the amount of decrease becomes alomst constant over time.

DAM algorithm is taking lesser time compared to other algorithms initially but as the number of threads increses beyound 20 it starts taking more time compared to the other algorithms this is because as the threads increases then amount of time taken to synchronize the threads also increases.