Operating systems   
Lab 1 - Report  
Threads

Adriana Bostandzhieva

Nikolay Stroitsov

Table of Contents

[Threads 2](#_Toc448676265)

[Description 2](#_Toc448676266)

[Solution and Results 2](#_Toc448676267)

[Analysis 2](#_Toc448676268)

# Threads

## Description

The aim of this laboratory exercise is to construct a Java program to find prime numbers using the Sieve of Eratosthenes algorithm. The program accepts two parameters from the console namely the range of numbers to be checked and the maximum threads that could run simultaneously.

## Solution and Results

The program gets the requested inputs from the console and initiates the process of finding prime numbers.

Due to the fact that the modern hardware is quite fast number of concurrent threads could hardly reach the limit. For example, the program evaluated the numbers from 2 to 35000 for 2.719s with maximum 9 threads running simultaneously and a limit of 10 threads. Thus, for illustrative purposes a delay of 1ms was introduced during the lab. The data collected after the delay was introduced is presented in Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Range | M = 5 | M = 10 | M = 20 |
| N = 100 | 3.524 | 1.844 | 0.704 |
| N = 500 | 48.441 | 20.479 | 12.16 |
| N = 1,000 | 146.533 | 90.743 | 39.976 |

Table 1. Average times per simulation with 1ms delay

The consecutive data gathered without the delay is shown in Table 2. Large range of numbers to be evaluated was chosen to illustrate the benefits of having higher maximum thread count.

|  |  |  |  |
| --- | --- | --- | --- |
| Range | M = 5 | M = 10 | M = 20 |
| N = 100 | 0.010s1 | 0.010s1 | 0.010s[[1]](#footnote-1) |
| N = 500 | 0.049s1 | 0.047s1 | 0.050s1 |
| N = 1,000 | 0.087s1 | 0.083s1 | 0.083s1 |
| N = 10,000 | 0.715s | 0.754s | 0.739s1 |
| N = 100,000 | 8.332s | 9.174s | 9.408s1 |
| N = 500,000 | 139.106s | 132.907s | 127.131s1 |
| N = 1,000,000 | 537.574s | 487.342s | 467.454s1 |

Table 2. Average times per simulation

## Analysis

It is easily visible from Figure 1 that the more threads are running the faster the computation time. However, due to the fact that this data contains 1ms delay for each thread, it represents an extreme case where the threads require many operations to finish.

Figure 1. Time required for evaluation of prime numbers with delay

Figure 2 shows the plotted data from Table 2 i.e the time required for the program to evaluate the prime numbers plotted against the number of allowed threads without a delay. The most of the cases the graph is a straight line. Advantage of having 20 threads running simultaneously is present only in the case of a very large range of numbers for evaluation, namely when N is 500,000 and 1 million.

Figure 2. Time required for evaluation of prime numbers

1. Maximum number of threads was not reached. [↑](#footnote-ref-1)