CompX341 Assignment 4

Annika Bell

ID: 1318741

Prime Sever and

Jmeter Stress Testing

Contents

[Prime Server 3](#_Toc11360970)

[Test Cases 3](#_Toc11360971)

[JMeter tests 4](#_Toc11360972)

[Tests and Process 4](#_Toc11360973)

[Discussion of Results 4](#_Toc11360974)

[Result Graphs 5](#_Toc11360975)

[Link to github repository: 8](#_Toc11360976)

Prime Server

Test Cases

Black Box selestions:

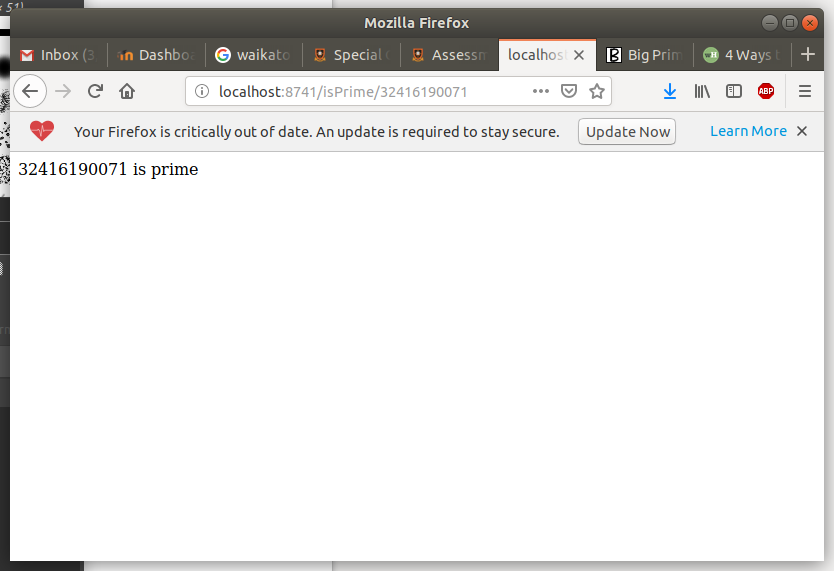
Cases 1, 0, and -1 were chosen because they are special cases in the definition of what it means to be a prime number. Case 2 was chosen as a marginal case, the lowest prim number. Several other primes were tested at random to ensure more than one or two cases had been tested. 32416190071 was tested as the largest prime I found on the website I retrieved it from. This was selected to test the server’s ability to handle large numbers correctly. Several random non-prime numbers were also tested to test that the server was returning non-prime results correctly. Several random negative numbers were tested, both those that are negations of primes numbers, and those that are negations of non-prime numbers. This was to make sure that no negative numbers were being registers as primes.

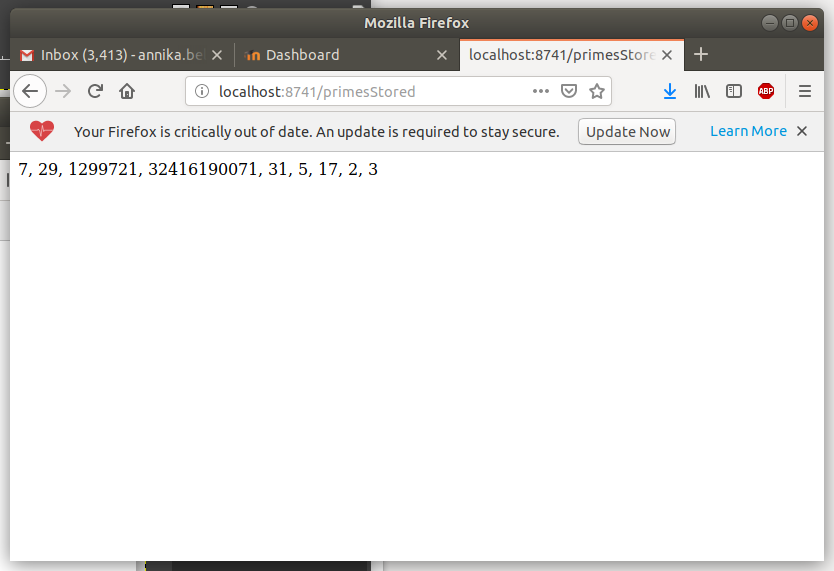
White Box Selections:

3 and 4 were choses as test cases as they are marginal using the method I used to determine if the number was a prime number, specifically marginal to the if number < 4 check.  
5, 7, 11, 13, 17, and 19 were chosen as test cases to check both the (num%6)-1 == 0 and the (num%6)+1 == 6 checks were functioning correctly

Results:

These test cases did highlight several errors in the code, giving me the opportunity to correct them.

Some screenshots from testing have been included.



JMeter tests

Tests and Process

The assignment called for us to design two testing scenarios.

Scenario 1 involved repeatedly checking if the number 2147483647 is a prime number.

Scenario 2 involved checking every number between 1 and 100 inclusive to see if it is a prime number, the repeatedly requesting the list of prime numbers stored.

I tested each scenario using all combinations of CPU settings of .1, .25, and .5 of a virtual CPU, with delays of 0ms, 250ms, and 500ms.

Discussion of Results

Overall trends:  
There were no failed results from any of the tests, so the server was not being pushed to breaking point.

In general, bigger timer delays led to quicker response times and lower throughput. My interpretation of this is that the throughput being lower when the timer delay was large possibly indicates the server has some time periods between requests where it has no queue and there are available idle server nodes, so its throughput falls due to a lack of sufficient requests to keep it busy. However, because it is not busy it able to handle some of, or in some cases possibly all the requests without queueing them, so the response times are lower.

In general, assigning the server more CPU resulted in quicker response times and higher throughput.  
Because the server had more CPU to utilize it was able to handle requests more quickly (and possibly to handle more requests at a time), and because requests spent less time being processed, they also moved through the queue more quickly and took even less time overall.

In many of the tests the response time would be initially very low, while the throughput was rising, and then would rise quickly where the throughput levelled off. This would seem to indicate the point at which the sever had reached its capacity to handle requests immediately so that further requests had to be queued, resulting in longer wait times. Two good examples of this are scenario 1, test 8, and scenario 2, test 8. Graphs of all results can be viewed in the next section.

Result Graphs

Scenario 1

|  |  |  |
| --- | --- | --- |
| Test 1 CPU = .1 Timer Delay = 0ms | Test 2 CPU = .1 Timer Delay = 250ms | Test 3 CPU = .1 Timer Delay = 500ms |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Test 4 CPU = .25 Timer Delay = 0ms | Test 5 CPU = .25 Timer Delay = 250ms | Test 6 CPU = .25 Timer Delay = 500ms |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Test 7 CPU = .5 Timer Delay = 0ms | Test 8 CPU = .5 Timer Delay = 250ms | Test 9 CPU = .5 Timer Delay = 500ms |
|  |  |  |

Scenario 2:

|  |  |  |
| --- | --- | --- |
| Test 1 CPU = .1 Timer Delay = 0ms | Test 2 CPU = .1 Timer Delay = 250ms | Test 3 CPU = .1 Timer Delay = 500ms |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Test 4 CPU = .25 Timer Delay = 0ms | Test 5 CPU = .25 Timer Delay = 250ms | Test 6 CPU = .25 Timer Delay = 500ms |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Test 7 CPU = .5 Timer Delay = 0ms | Test 8 CPU = .5 Timer Delay = 250ms | Test 9 CPU = .5 Timer Delay = 500ms |
|  |  |  |

Link to github repository:

<https://github.com/Anniebe11/CompX341-Assignment-4>