Primy: finding a large prime

Johan Montelius

October 2, 2016

Introduction

Your task will be to implement a distributed system that will find large primes. The system should have one server that is in control of the computation and a dynamic set of workers that are assigned numbers to test for primarity.

1 Finding a prime

This is a test that we should use in a coming exercise. The task is to test if a number is prime. A complete algorithm is quite expensive to execute for large numbers so we use an algorithm that will detect if number is a prime with very high accuracy. The algorithm is from Fermat and to compute it we need a fast implementation of modular exponentiation. Open up a new file `fermat.erl` and declare the module `fermat`.

```
mpow(N, 1, _) ->
    N;
mpow(N, K, M) ->
    mpow(K rem 2, N, K, M).

mpow(0, N, K, M) ->
    X = mpow(N, K div 2, M),
    (X * X) rem M;
mpow(_, N, K, M) ->
    X = mpow(N, K - 1, M),
    (X * N) rem M.
```

This algorithm will calculate $N^K \bmod M$ either by $N^{K/2} * N^{K/2} \bmod M$ if K is even, or by $N^{K-1} * N \bmod M$ if K is odd. Modular multiplication is nice since you can apply the modular operation on both terms and the result will be the same. Try it and see if it works.

Next we implement the test by Fermat. If a random number R, less than P, raised to $P - 1$ modulo P is equal to 1 e.g. if R^{P-1} is relative prime to P, then it is very likely that P is prime.

```
fermat(1) ->
    ok;
```

```
fermat(P) ->
    R = random:uniform(P-1),
    T = mpow(R,P-1,P),
    if
       T == 1 ->
         ok;
       true ->
         no
    end.
```

Note that it is only likely! We want to perform this test several times using different random numbers.

```
test(_, 0) ->
    ok;
test(P, N) ->
    case fermat(P) of
     ok ->
       test(P, N-1);
     no ->
       no
     end.
```

How many times do we have to perform the test, well it depends on how many false primes you want to find. Note that most numbers are not prime and fail the test in one try, if we stumble on a prime we might as well run the test a couple of time since it will not slow down the overall performance that much.

That's it. Compile, load and do some experiments. Why not build a prime generator, start with an integer and test if it's a prime. If it's not, then move on to the next integer. Notice that Erlang is implemented using a big-num packet and can handle integers of (almost) arbitrary size. This is (probably) a prime that I found after some searching:

```
    75654596987987976987
    68756756757657656987
    98789798798789796546
    54654564217541236547
    65421378512736521765
    73658765123765123786
    512378657852319179
```

Now lets' implement network of nodes that collaborate in the search process.

2 The server

We should implement a server that keeps track of the highest prime number found so-far and the number next in turn to examine. The server should accept request from workers and hand out numbers that should be examined. If the workers can determine that it is a prime number it will return it to the server.

We will not handle the situation were workers accept a number and then dies nor workers that are malicious and report prime numbers without proper checking.

3 Speed it up

There are two ways to speed up the process. One is to implement a more efficient prime test function on the worker, another is to do part of the prime checking on the server before delegating a number to a worker. How much should be done at the server? We don't want the server to spend it's time doing pre-checking of numbers if the workers are idle. At the same time replying on a request takes time so we should reduce the number of requests as much as possible.

What's the bottle neck in our system: the number of workers, the server or the fact that we're communicating over a shared WLAN? Does this change as numbers get larger?

4 Making it robust

How can we handle the situation with dying workers. Can we have redundancy in the system so dead workers are detected and the assigned number is given to another worker?

Can we handle malicious workers? Should we send all numbers to more than one worker? Is it more important to find all numbers or making sure that reported primes are actually prime?