**Distributed Operating System Principles (COP5615) : Project1**

**Submitted by:**

**Rohith Kalwa (52075407)**

**Karthik Metlapalli ()**

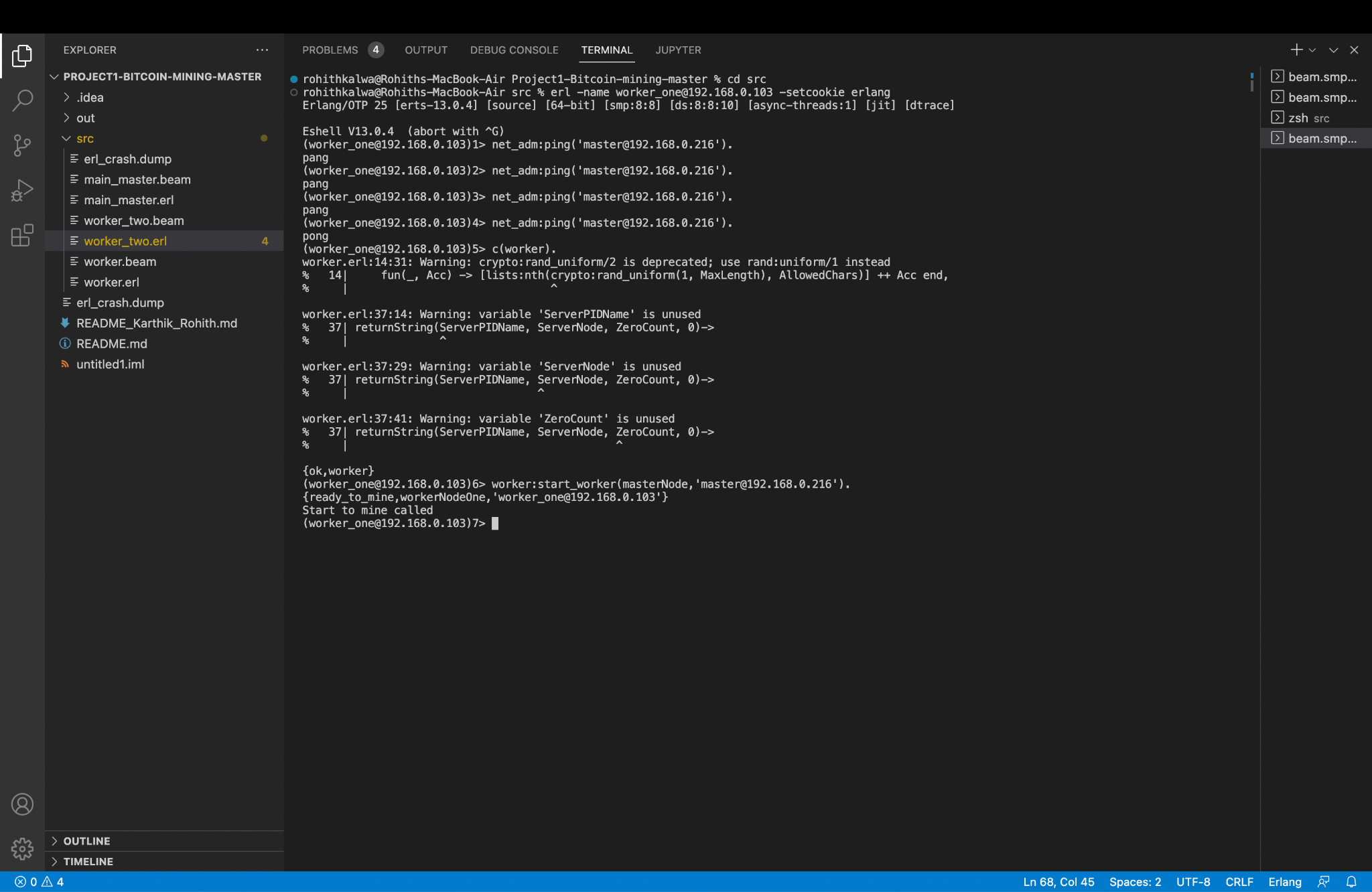
**Problem Definition**

Bitcoins (see http://en.wikipedia.org/wiki/Bitcoin) are the most popular crypto-currency in common use. At their heart, bitcoins use the hardness of cryptographic hashing (for a reference see http://en.wikipedia.org/wiki/Cryptographic Hash Function)to ensure a limited “supply” of coins. In particular, the key component in a bit-coin is an input that, when “hashed” produces an output smaller than a target value. In practice, the comparison values have leading 0’s, thus the bitcoin is required to have a given number of leading 0’s (to ensure 3 leading 0’s, you look for hashes smaller than 0x001000*...* or smaller or equal to *0x000ff...*.The hash you are required to use is SHA-256. You can check your version against this online hasher:http://www.xorbin.com/tools/sha256-hash-calculator. For example, when the text “COP5615 is a boring class” is hashed, the value fb4431b6a2df71b6cbad961e08fa06ee6fff47e3bc14e977f4b2ea57caee48a4 is obtained. For the coins, you find, check your answer with this calculator to ensure correctness. The goal of this first project is to use Erlang and the Actor Model to build a good solution to this problem that runs well on multi-core machines.

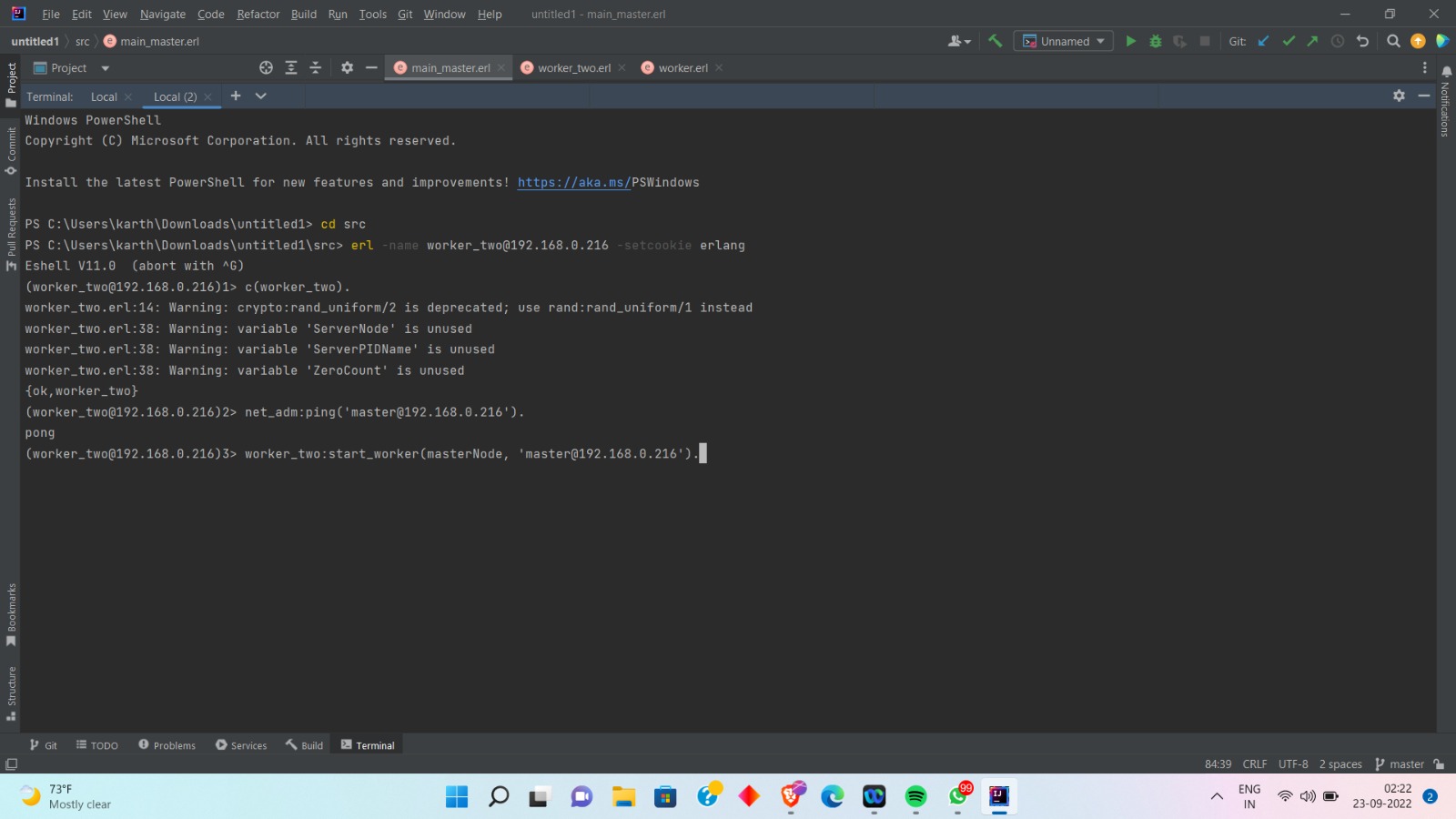
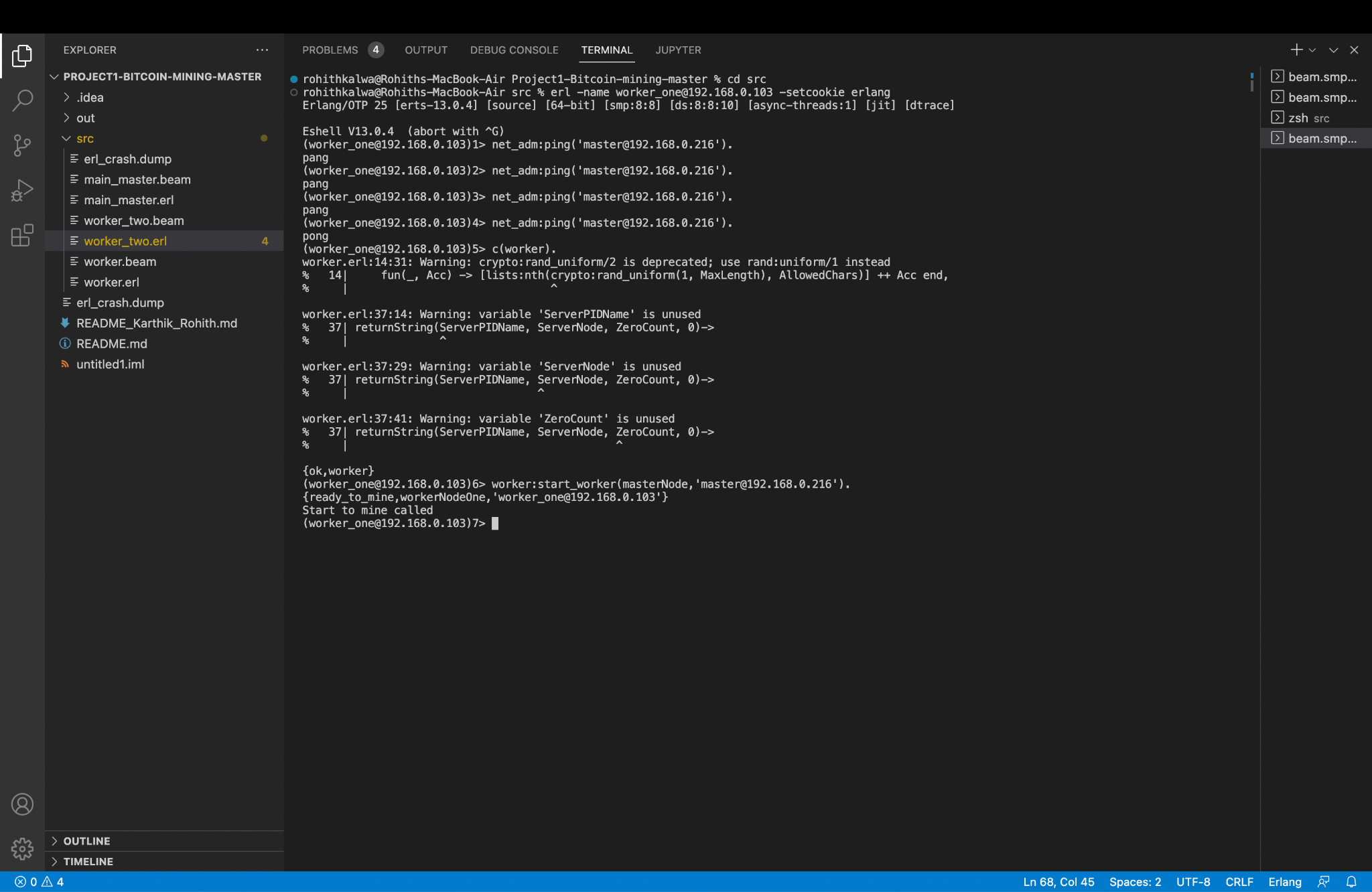
**Implementation**

Initially worker1 and worker2, who already knows the master node and process name, pings the master to establish the connection.

Once the connection is established first the master and the workers should be started.



Then the master sends the coins to the next available workers that are to be mined and also mines some coins by itself.



**Assumptions**

* All the nodes should be under the same network pool and should be aware of the unique cookie that is used for secure connection.
* The hash generated should have “K” leading zeros for a successful match.
* As the number of actors grow, so does the number of possible combinations, resulting in the increased hash generation and a higher chance of finding the Bitcoins.

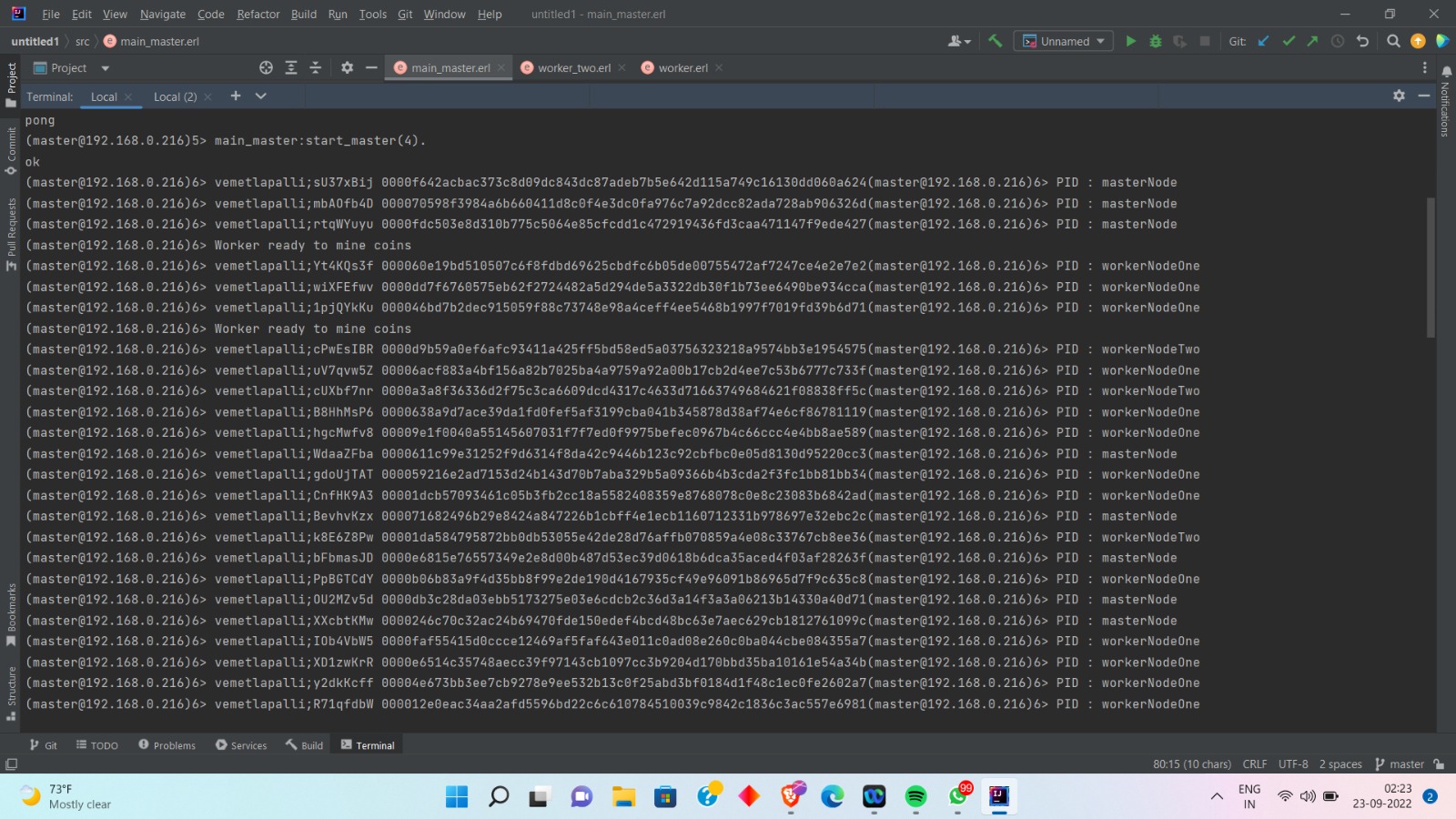
**Size of the work unit**

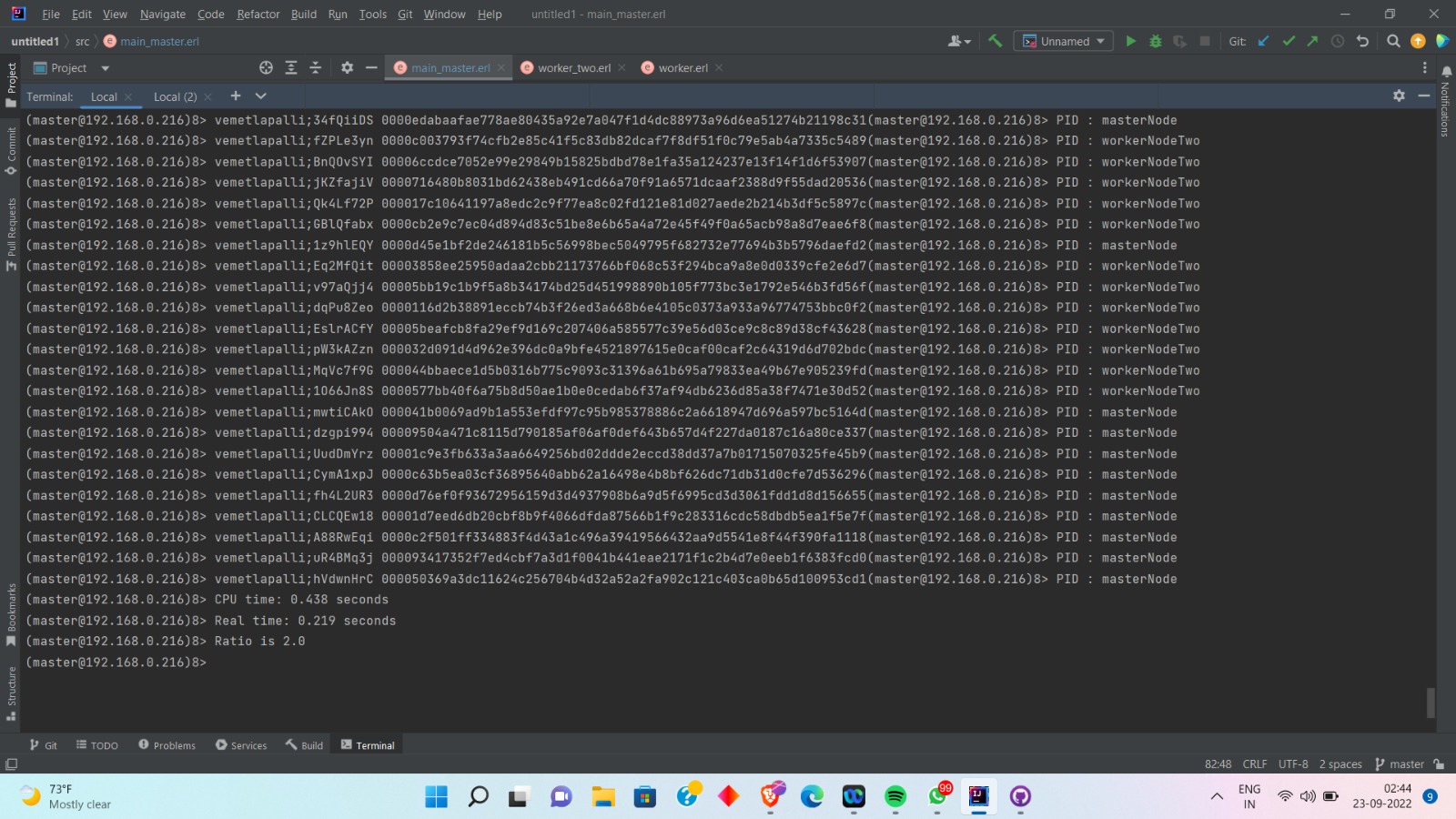
The workers receive a mining request with no upper limit on the number of coins to be mined in this problem. For bitcoin mining we launched 2\*logproc worker process/node, where logproc is the number of logical processors available on teh computer. We also supervise each worker’s process and restart it if it fails to guarantee that all worker processes remain operational.

The work unit is 3 in size. A single worker (i) Produces a random worker, (ii) Performs SHA-256 encoding, (iii) Checks the encoded string for number of reading zeroes.

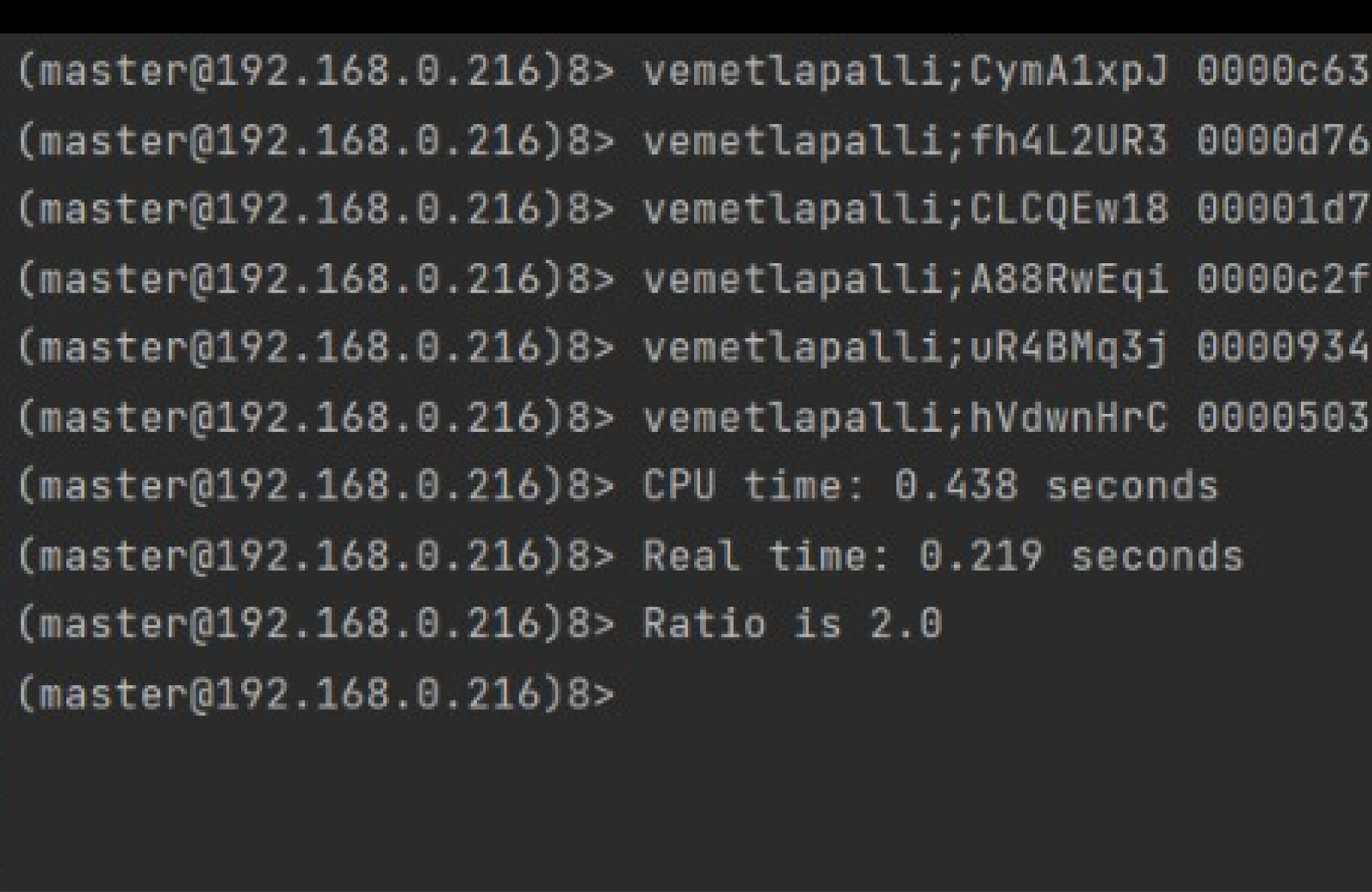
**The result of running your program for input 4**

The below are the coins mined that contain 4 leading zeros.





**The ratio of CPU time to REAL TIME**



CPU/Real-time ratio (i.e., CPU time/Real time) = 0.438/0.219=2.0 (>1, therefore Parallelism exists).

**The coin with the most 0s you managed to find**

The coins with the most 0s we found were 7.

**The largest number of working machines you were able to run your code with.**

Since we only have three machines, we made it work on them. However, we can tweek this code to make it work on numerous machines.

**Conclusion:**

The CPU\_time/Real\_time ratio in the client-server architecture is 2, which is greater than the one obtained while running on a single machine for the same amount of computation. As a result, the introduced multi-system design has improved performance.