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

**Submitted by:**

**Rohith Kalwa (52075407)**

**Venkata Sai Karthik Metlapalli (65764476)**

**Problem Definition**

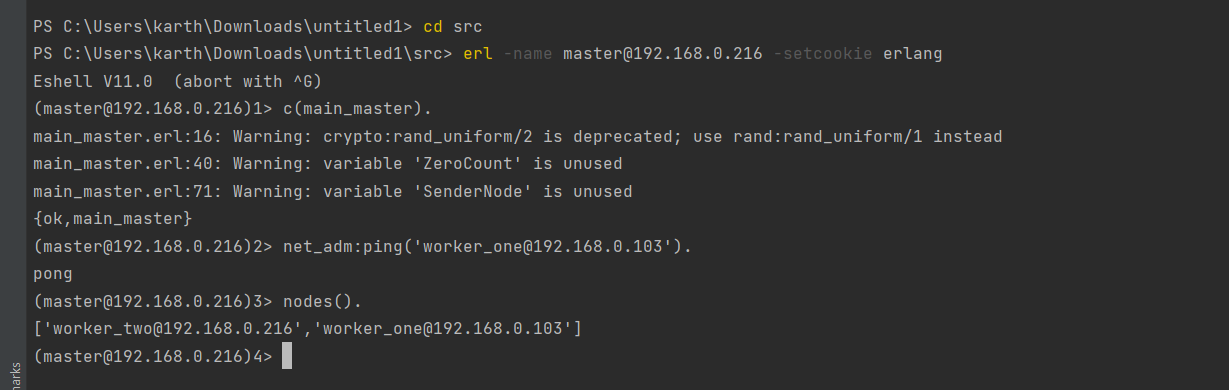
Bitcoins (see http://en.wikipedia.org/wiki/Bitcoin) are the most popular crypto-currency in common use. At their heart, bitcoins use the hardness of cryptographic hashing (for a reference see http://en.wikipedia.org/wiki/Cryptographic Hash Function)to ensure a limited “supply” of coins. In particular, 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**

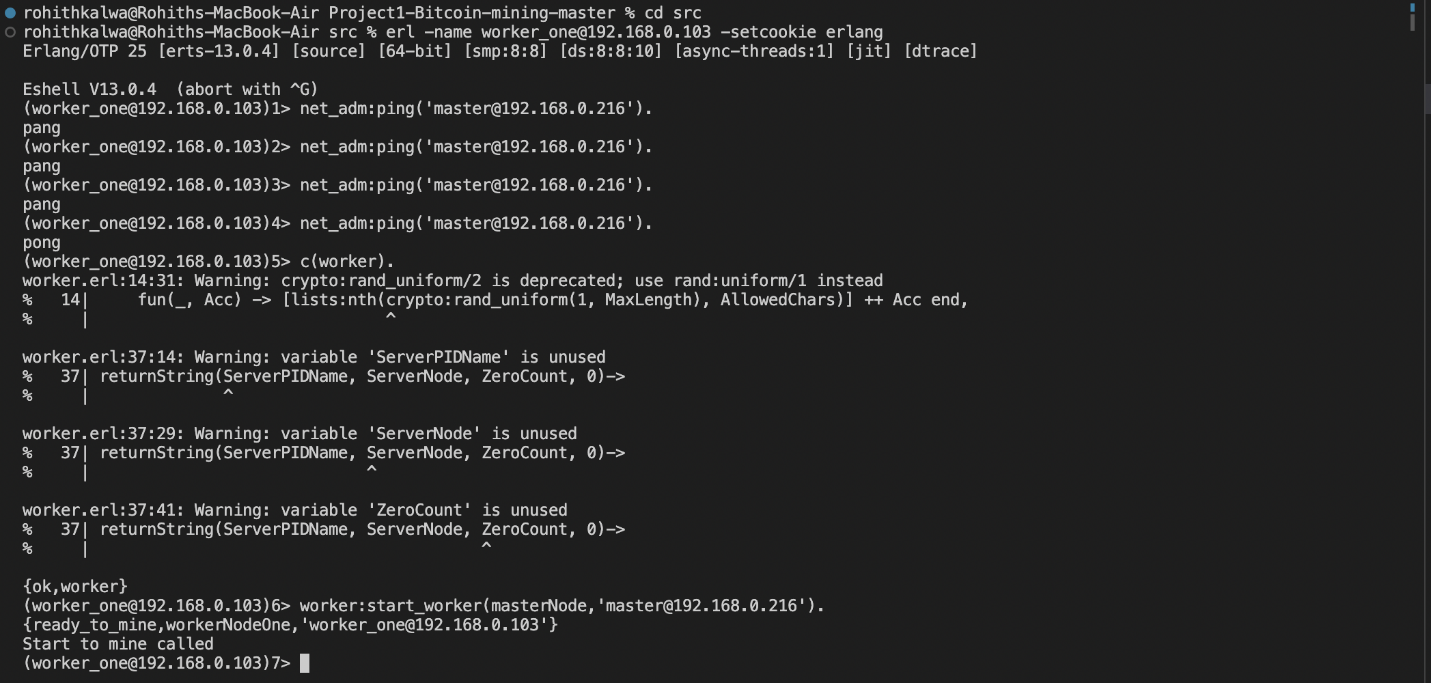
Initially the master and worker nodes are registered in the same network pool using shared secure cookie. After this the worker nodes, who already know the master node and master process name, ping the master node to establish a connection for message passing.

Once the connection is established, first the master process is started followed by the worker processes.

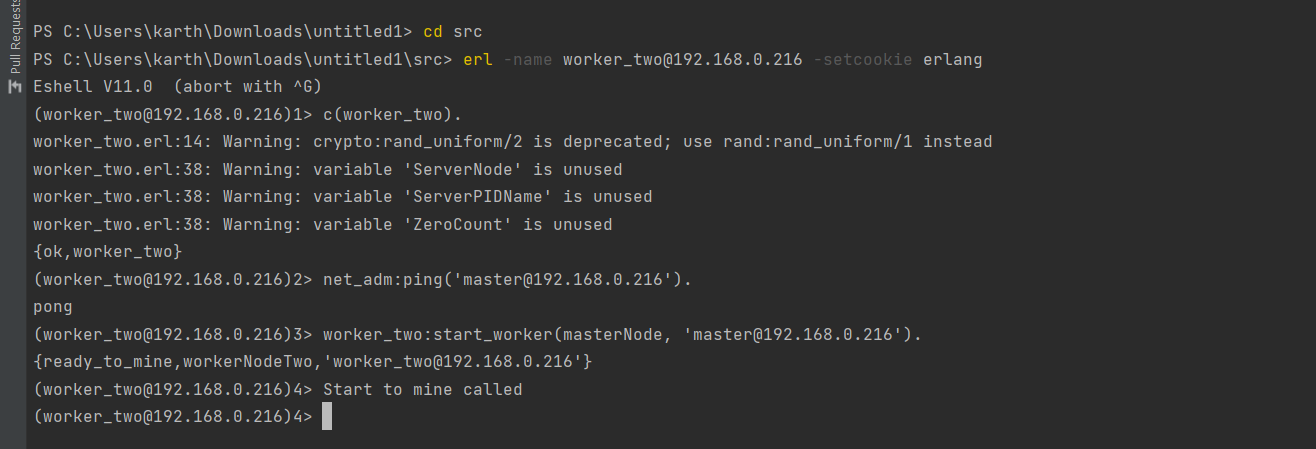
**Master:**



**Worker One( On remote node in a different machine):**



**Worker Two( On different node in same machine):**



Master takes the number of leading zeroes to be present in bitcoin hash as input and starts mining coins. Once a worker process is available, it sends an available\_to\_mine message to master. Then the master allocates the mining task to that worker by sending a start to mine message along with input\_number number of zeroes. And the master listens from the worker and prints output while parallelly mining 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 exactly “K” leading zeros for a successful match.
* As the number of actors grows, so does the number of possible combinations, resulting in the increased hash generation and a higher chance of finding the Bitcoins until a certain threshold.

**Size of the work unit**

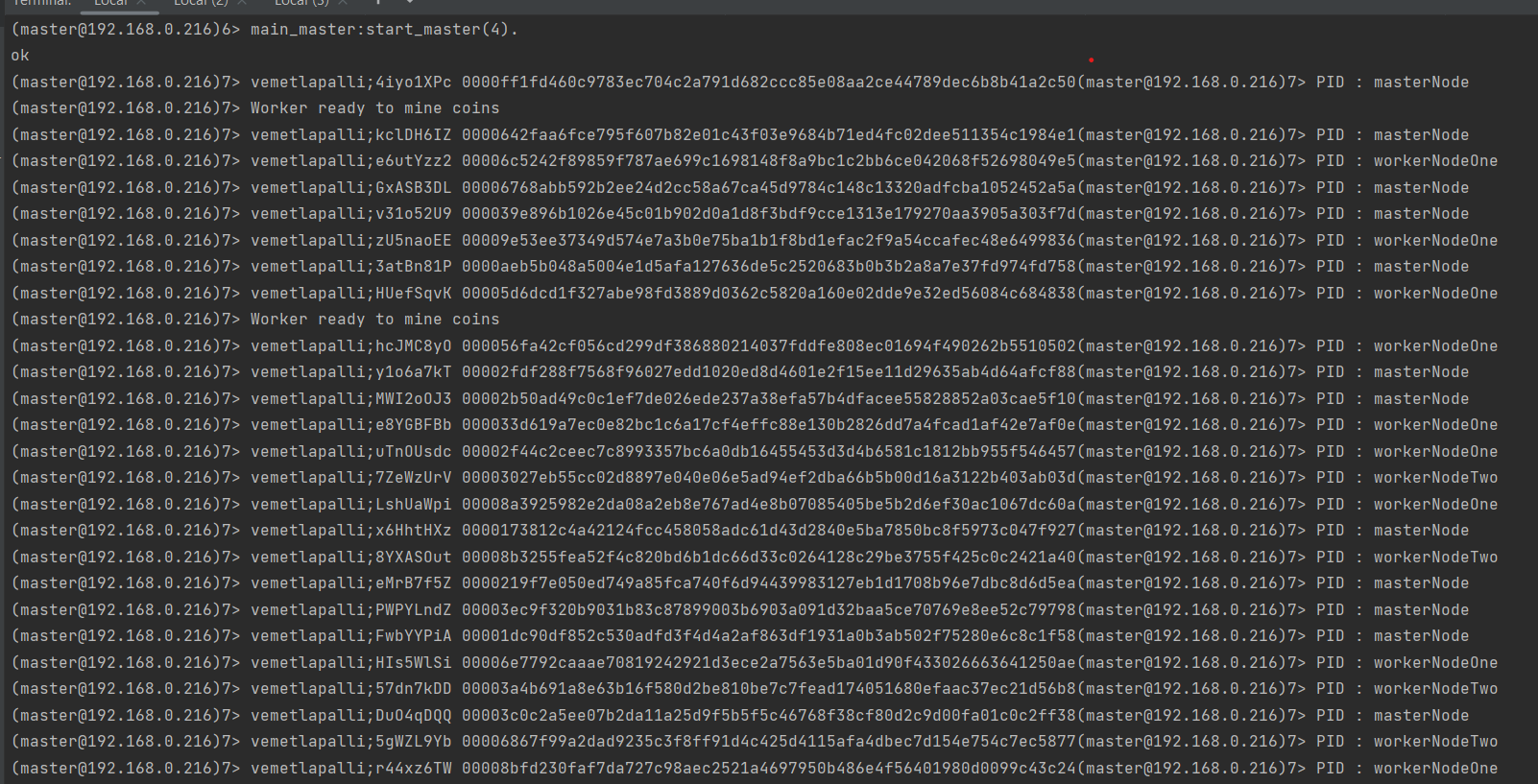
The workers receive a mining request with no upper limit on the number of coins to be mined in this problem. For bitcoin mining we launched 2\*number of logical processors per worker process/node using **erlang:system\_info(logical\_processors\_available)**

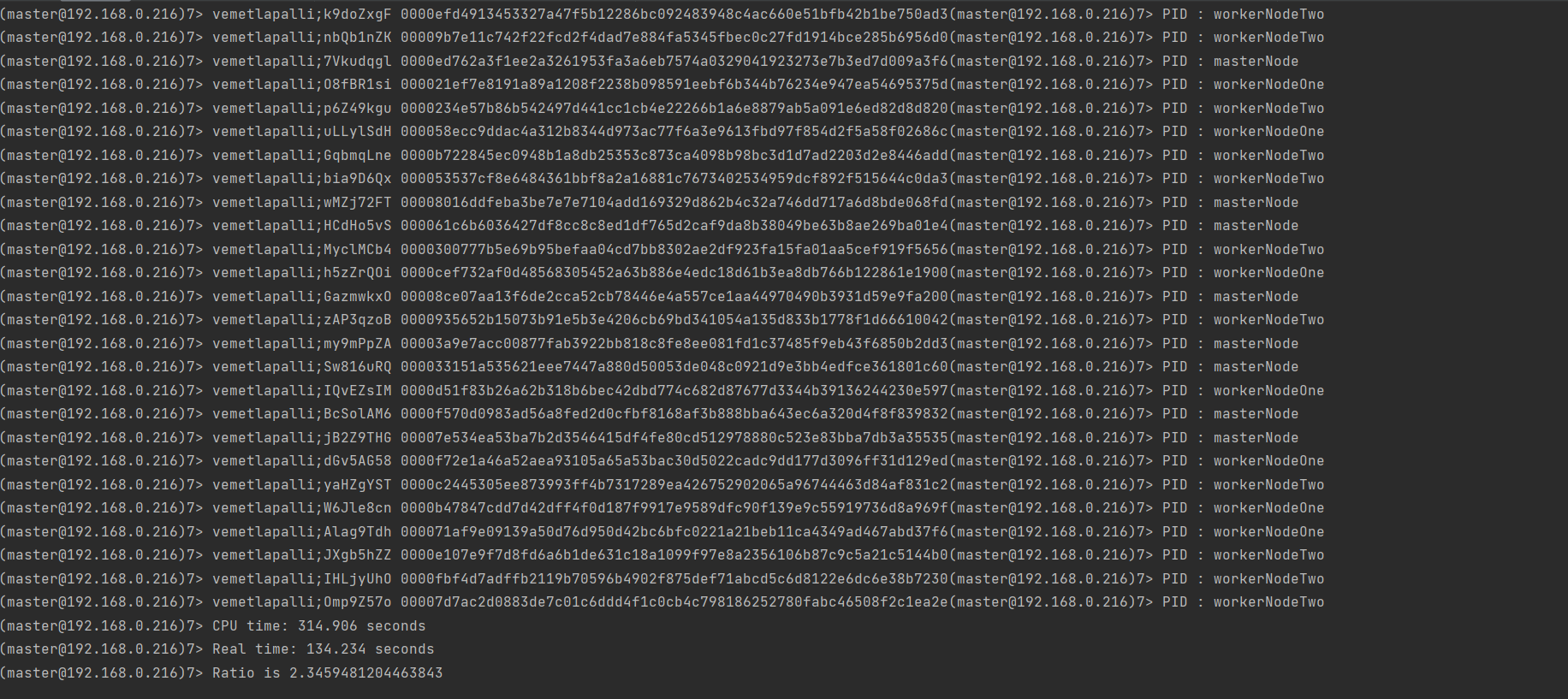
The work unit is 3 in size i.e, each spawned process mines 3 coins before exiting. A single worker (i) Produces/spawns random worker processes, (ii) Each of these performs SHA-256 encoding, (iii) Checks the encoded string for number of reading zeroes.

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

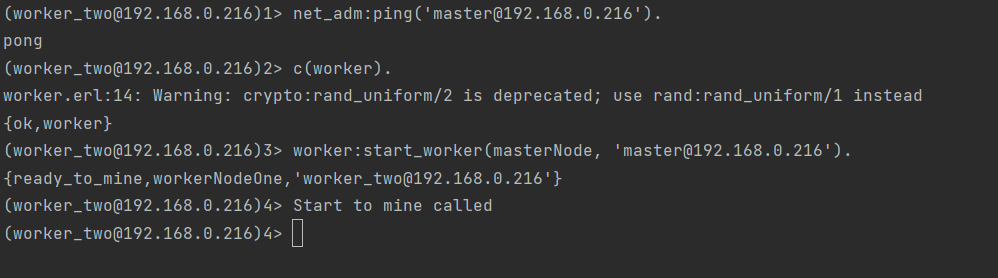
The below are the coins mined that contain 4 leading zeros.

**Master Process Output**

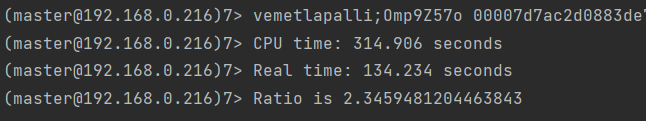


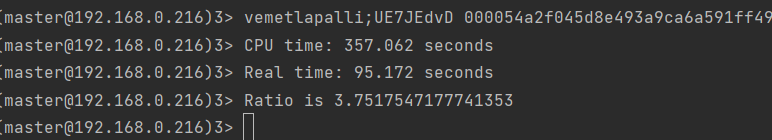


**Worker Process Output**



**The ratio of CPU Time to REAL Time for multiple runs**





CPU/Real-time ratio (i.e., CPU time/Real time) = 357.062/95.172=3.752 (>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. The master is capable of connecting with and accepting the output from a large number of workers.

**Conclusion:**

The CPU\_time/Real\_time ratio in the client-server architecture is greater than 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.