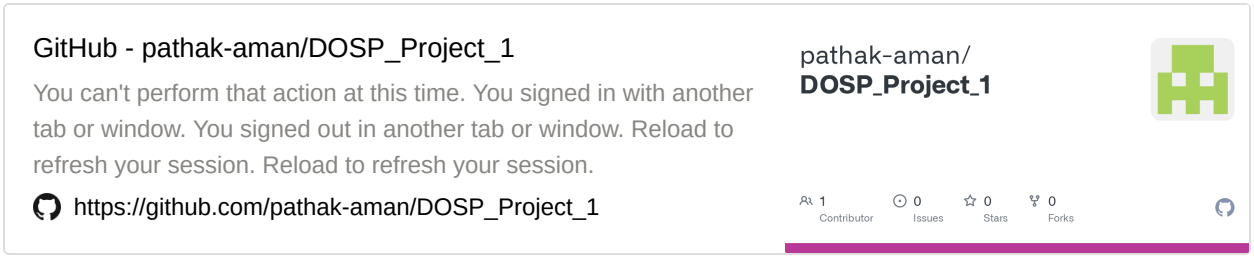
# Bitcoin Mining: ReadME



# GitHub Link



# Authors:


1. Dipali Patidar - UFID# 12866807
2. Aman Pathak - UFID# 88828029


# Our take on the project statement:

We have created Server-worker architecture using two separate machine. The Server take K, number of workers and subproblem size as input and spawn a thread to mine Bitcoin.  If second machine  is connected then Server tries to create worker threads, and then assigns bunch of strings to each thread and spawn them to mine Bitcoin. No of substrings processed by each workers is defined by size of subproblem. Moreover, whenever a worker finishes mining all the batch of strings, it sends the result back to the second machine, which displays that as one of the output. So, Server has 3 tasks - mining bitcoin on its own on  batch of string, spawning a worker threads and processing the bitcoins from the worker thread. The worker thread has 2 tasks - get a batch of strings from Server thread and mine bitcoin, and the send mined result back to the Server.

# Configuration of our machines:

This project has been tested on the following machines.

**Specifications**

| Machine | RAM | Number of cores | Processor | Operating System |
|---|---|---|---|---|
| Apple MacBook Pro | 16 GBs | 8 | M1 Chip | macOS Monterey |
| Alienware M15 | 32 GBs | 8 | Ryzen 9 5900HX | Windows 11 |

# How to run:

Download all the files and place them in a directory.

## Steps at Server:

1. Open a terminal shell
2. Type the following code:

```
# Use werl if on windows machine
erl -name server@<ip address here> -setcookie "xyz"
Eg: erl -name server@192.168.0.192 -setcookie "xyz"
```

3. In the erlang shell environment:

```
% To compile the file and generate the .beam file
1> c(final).

% To run the server
2> final:start_pong().

% Follow the on prompt screen to proceed
Enter K: 3
Enter Number of workers: 30
Enter Number of sub-problems a single worker handles: 10000

% The server starts mining bitcoins of K = 3
```

## Steps at Worker:

1. Open a terminal shell
2. Type the following code:

```
# Use werl if on windows machine
erl -name worker@<ip address here> -setcookie "xyz"
Eg: erl -name worker@192.168.0.192 -setcookie "xyz"
```

3. In the erlang shell environment:

```
% To compile the file and generate the .beam file
1> c(final).

% To up a worker
2> final:start_ping(<server ip address>).
```

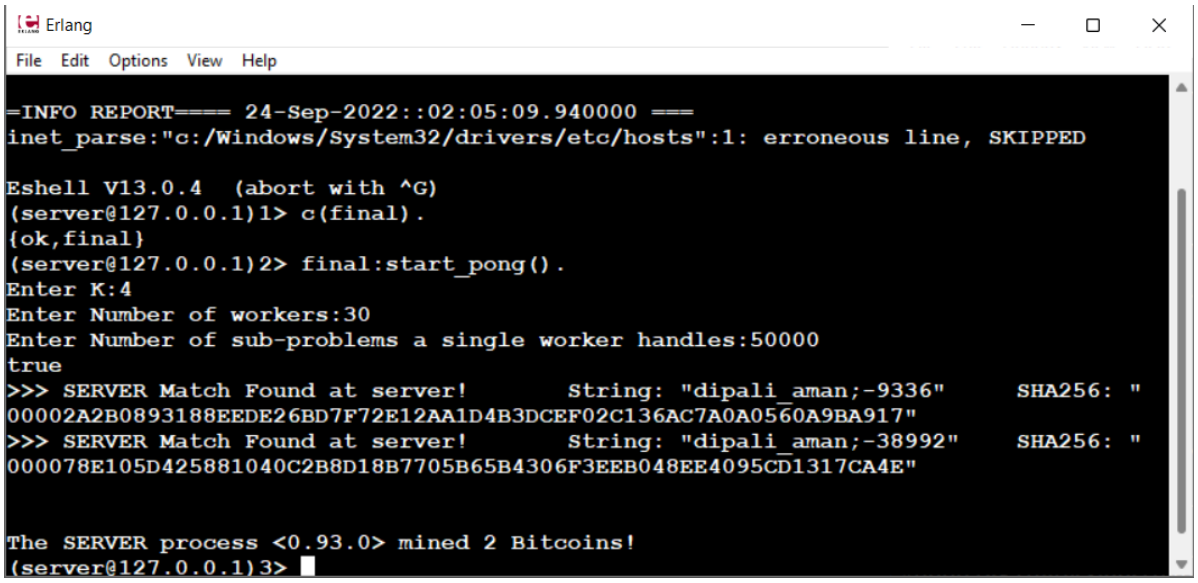
```
Eg:final:start_ping('server@192.168.0.192').

% The worker gets engaged in the mining
```

# Screenshots

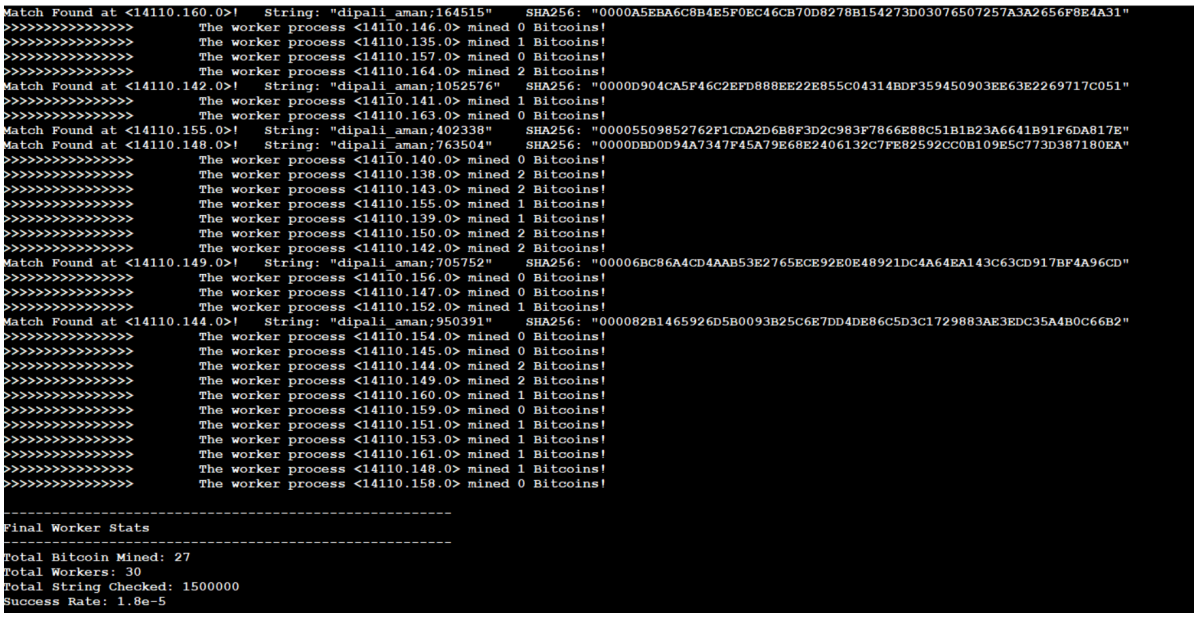
## At Server:

1. Server starts minting coins on its own.




Here server has mined 2 coins! 🪙

2. Server gets worker the task of mining coins on the worker machine.

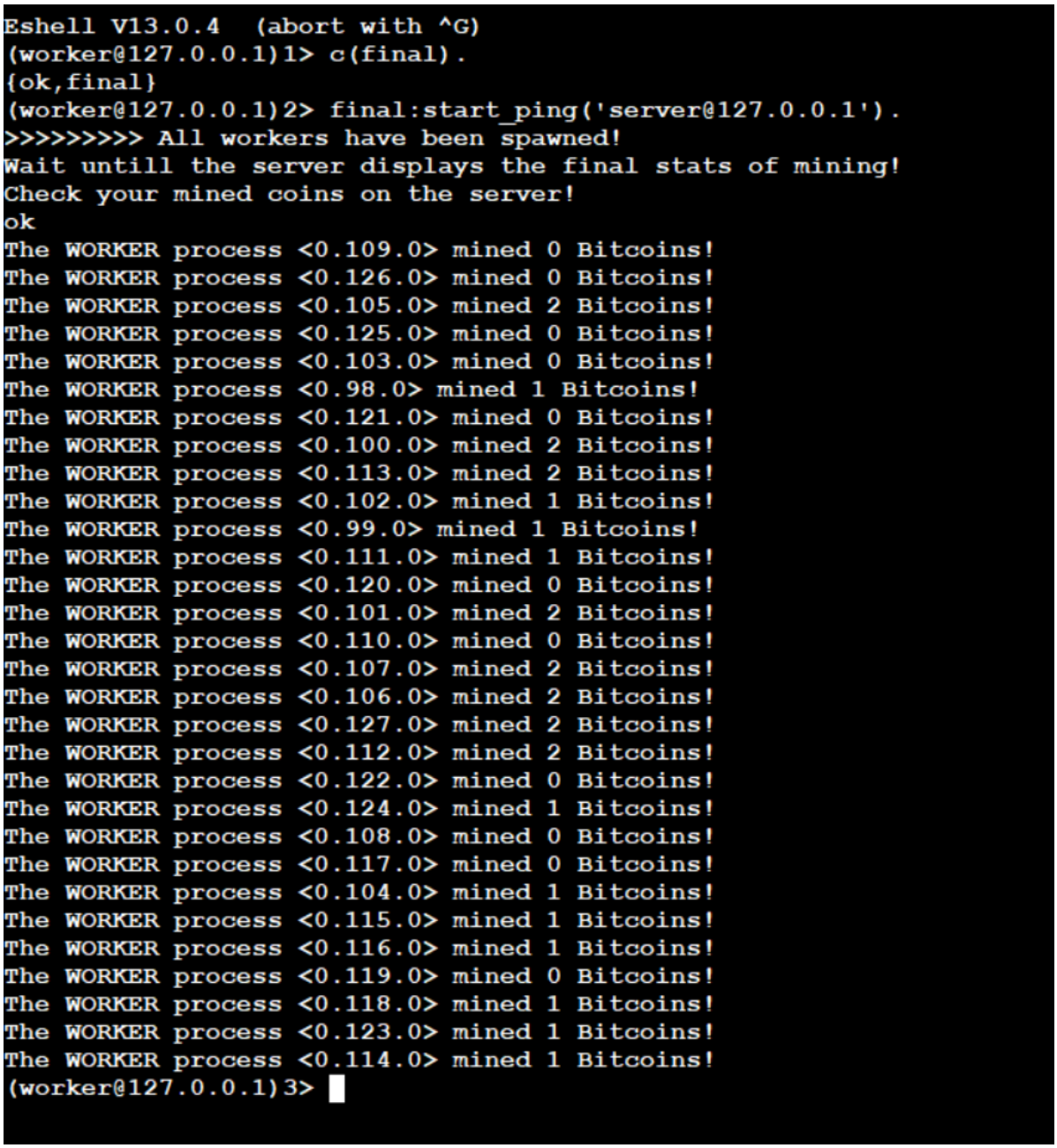



As soon as worker is up, server distributes the task to it.

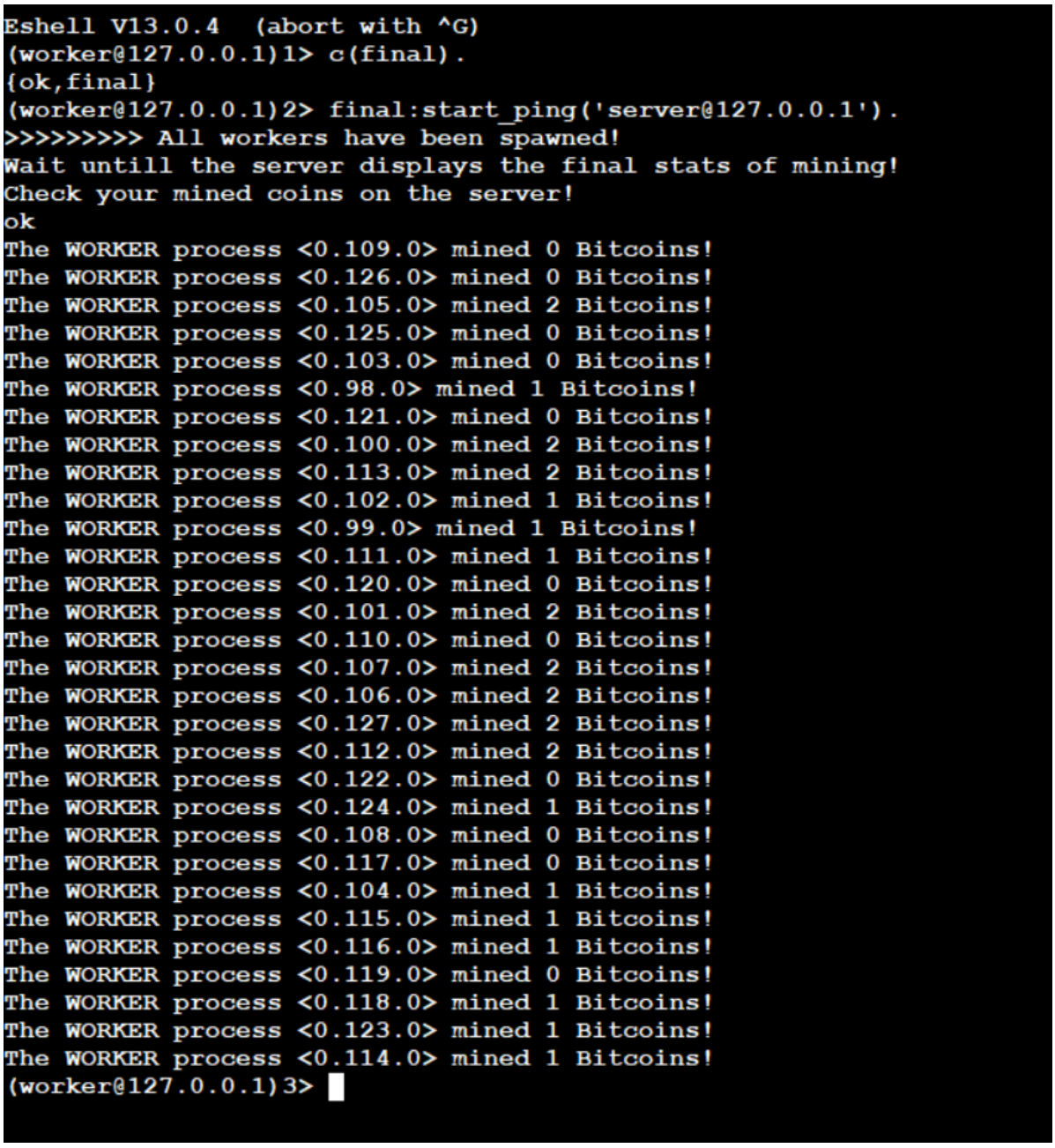
The final **worker** stats at the end, show that worker mined **27 bitcoins!** +

## At Worker:

1. On connecting with server, it distributes the mining and contributes!




The worker's terminal. It displays stats per worker process of how many bitcoin it mined!
Look many processes including <0.100.0> mined **2 bitcoins!**




The worker's terminal. It displays stats per worker process of how many bitcoin it mined!
Look many processes including <0.100.0> mined **2 bitcoins!**

# Part A) Subproblem Analysis:

> Size of the work unit that you determined results in the best performance for your implementation and an explanation of how

> you determined it. The size of the work unit refers to the number of sub-problems that a worker gets in a single request from the boss.

For this we ran multiple expeirments for different value of subproblems.
We chose K = 4 and Worker = 30 and plotted graph for number of subproblem per worker v/s CPU time to Real time ratio. We found optimal solution for subproblem = 50000.

**Table 1: Subproblem Analysis**

| Aa Sr No | # K | # Workers | # Number of Subproblem | # REAL Time (ms) | # CPU Time (ms) | ∑ Ratio of CPU/Real time |
|---|---|---|---|---|---|---|
| 1 | 4 | 30 | 100 | 69000 | 110000 | 1.594202898551 |
| 2 | 4 | 30 | 1000 | 109000 | 320000 | 2.935779816514 |
| 3 | 4 | 30 | 5000 | 20508000 | 62000000 | 3.023210454457 |
| 4 | 4 | 30 | 10000 | 20822000 | 80000000 | 3.842090097013 |
| 5 | 4 | 30 | 20000 | 29400000 | 125000000 | 4.251700680272 |
| 6 | 4 | 30 | 50000 | 61007000 | 295000000 | 4.835510679102 |
| 7 | 4 | 30 | 100000 | 70279000 | 314000000 | 4.467906487002 |
| 8 | 4 | 30 | 200000 | 73964000 | 270000000 | 3.650424530853 |
| 9 | 4 | 30 | 500000 | 79160000 | 290036000 | 3.663921172309 |
| 10 | 4 | 30 | 1000000 | 64866000 | 254391000 | 3.921792618629 |
| 11 | 4 | 30 | 2000000 | 118660000 | 488307000 | 4.115177818979 |

# Graph

To find the optimal size of work-unit, subproblems we tested with K =4,


Plot of Size of work unit vs Ratio of CPU and Real time.

# Part B) Result of K - 4

> The result of running your program for input 4

1. Server terminal:

   https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c528780b-564f-40e1-a0cc-58275870c771/server.pdf

2. Worker Terminal:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/eaf02afc-cd0c-48dd-b494-2538ef24a677/worker.pdf

# Part C) Ratio of CPU time to REAL TIME

> The running time for the above is reported by time for the above and report the time. The ratio of CPU time to REAL TIME tells you how many cores were effectively used in the computation.  If you are close to 1 you have almost no parallelism (points will be subtracted).

By plotting graph of  ***number of sub - workers*** vs ***CPU/Real time ratio*** for  K=4 and workers = 30, we observed optimal perform for subproblem = 50000.

💡 **Final Worker Stats**

Total Bitcoin Mined: 27
Total Workers: 30
Total String Checked: 1500000
Success Rate: 1.8e-5
Total CPU time spent : 295000000
Total Real time spent : 61007000

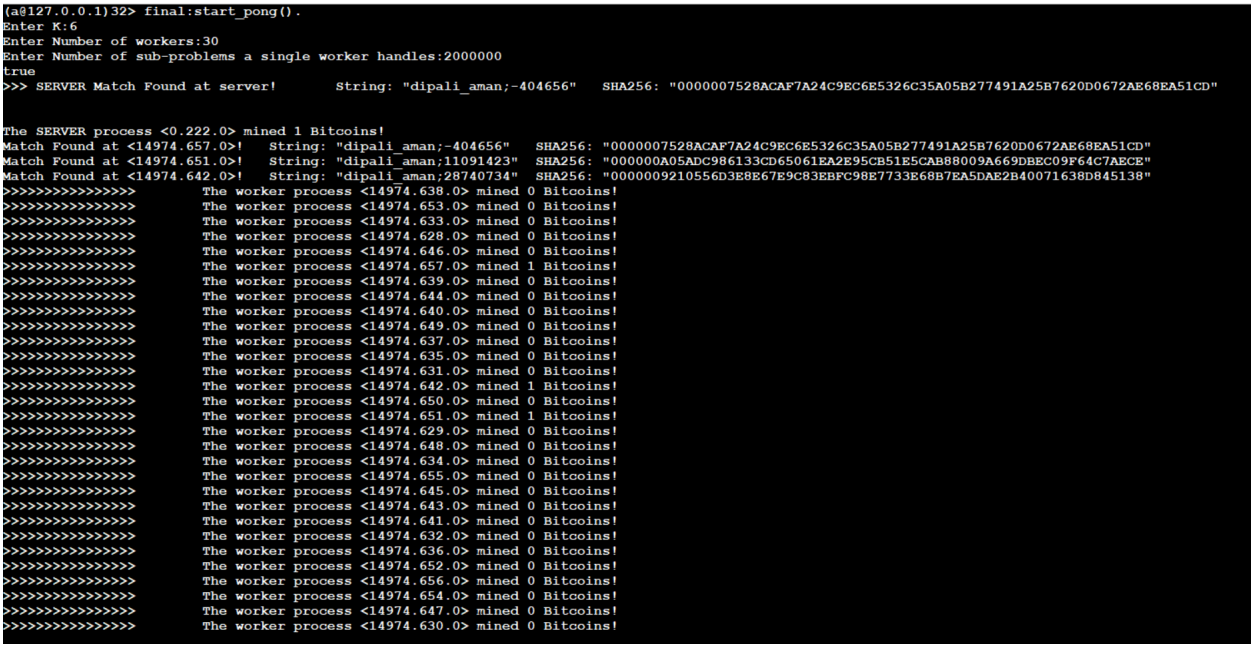
$CPU/REAL = \frac{295000000}{61007000} = 4.835$

# Part D) Coins with Most 0s

> The coin with the most 0s you managed to find.

We were able to mine 4 coins with 6 leading zeroes in the hashed string.
One by server and Three by worker.

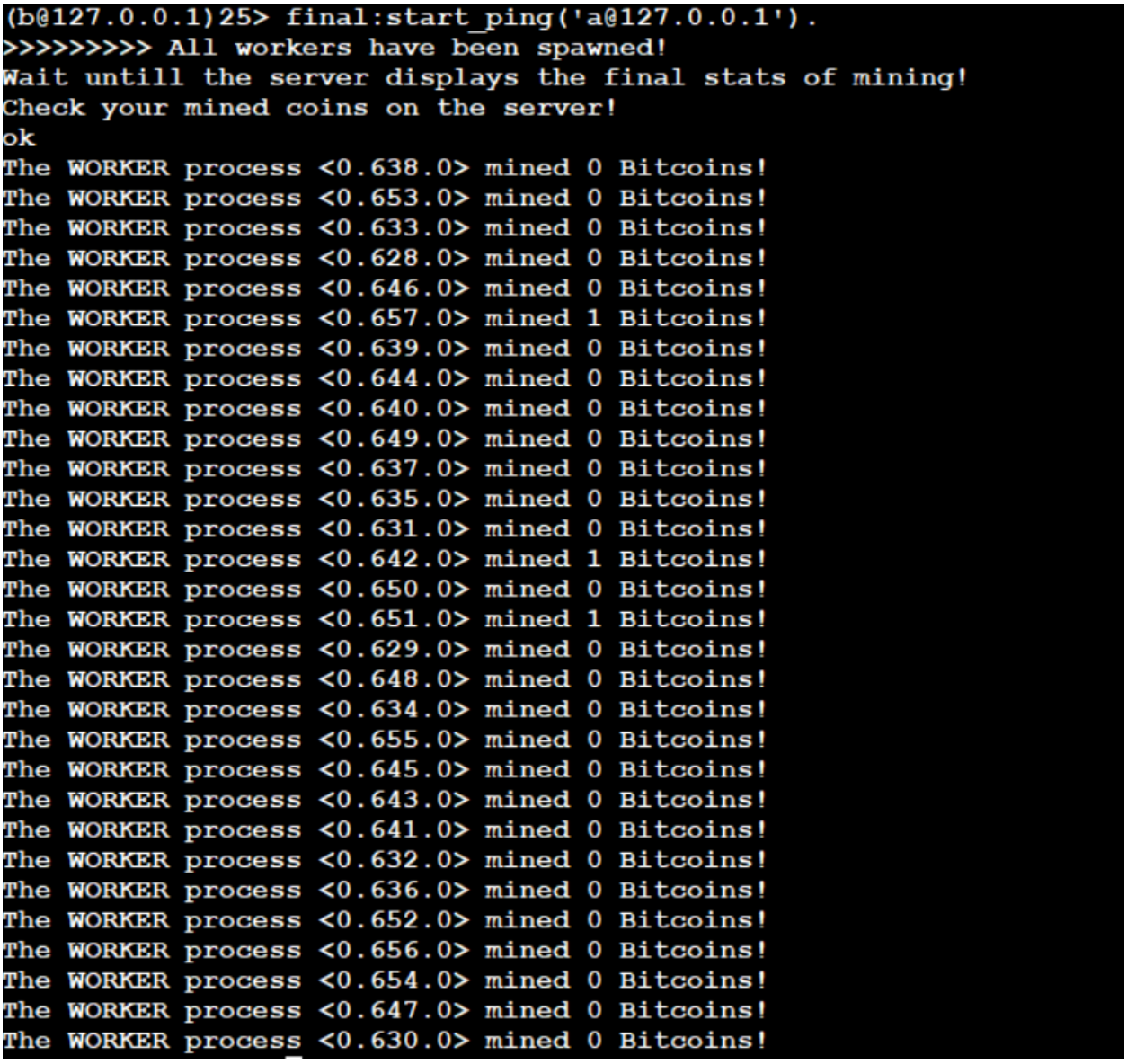
## Server Terminal Screenshot:




Server mined one bitcoin at K = 6! 🪙

## Worker Terminal Screenshot:

Workers found 3 bitcoins!

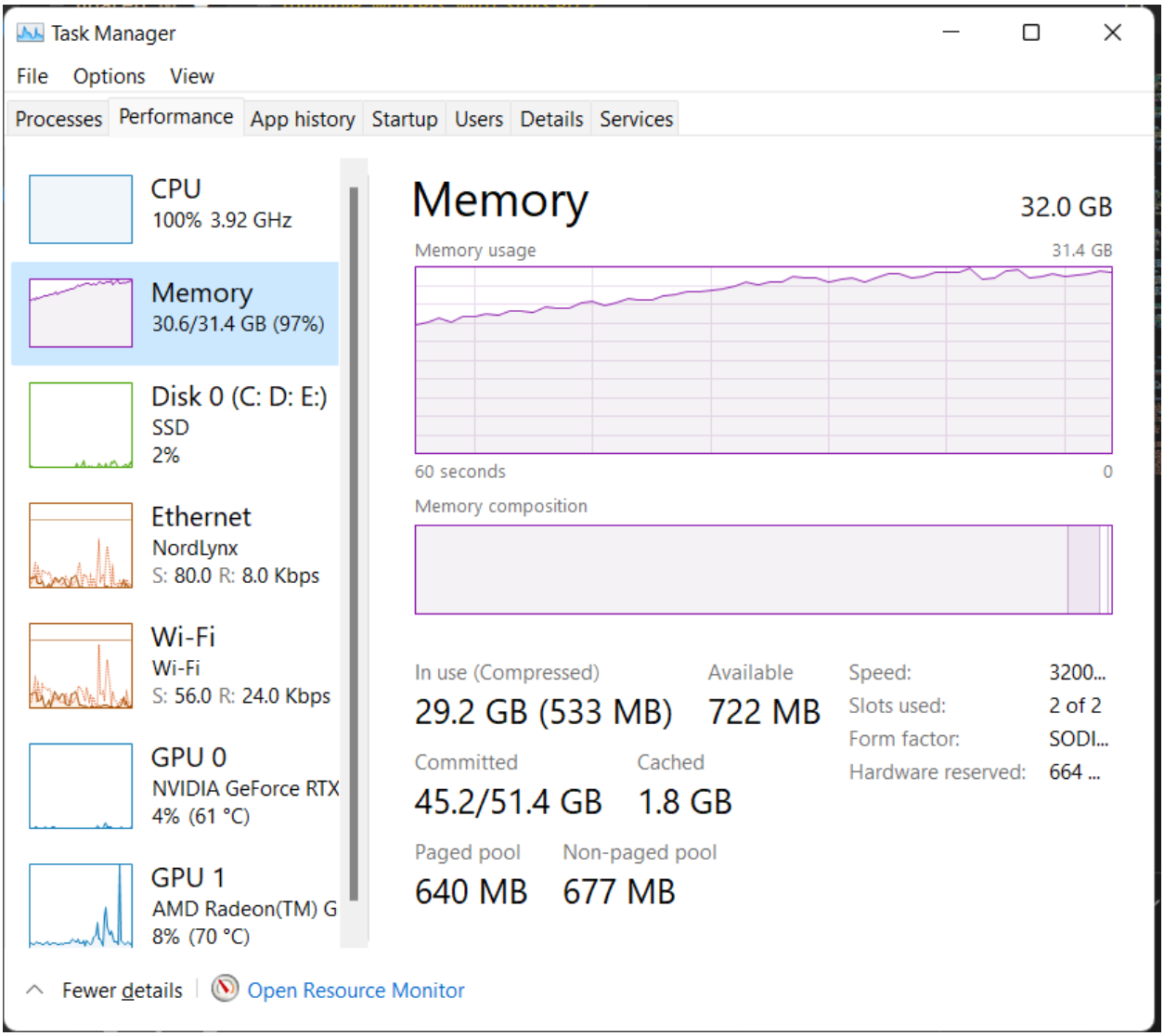



Worker node mined three bitcoin at K = 6. 🪙🪙🪙

## Performance stats when a large K = 7 or above:

Total Workers: 200

Total String Checked: 200000000




CPU is at 100% and Memory is at 97%!

# Part E) Largest Machine

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

We were able to run our code with 2500 workers.

**Final Worker Stats**

Total Bitcoin Mined: 1883
Total Workers: 2500
Total String Checked: 125000000
Success Rate: 1.5064e-5

## Screenshot at Server terminal: