

# Distributed Systems

## Bitcoin Mining

READ ME file for Distributed Operating Systems - Project 1

Date: September 18th, 2017

Group Name: BitTorrent

Group members:

1. **Mohit Mewara**, UFID: 8413-2114, mohitmewara@ufl.edu
2. **Nakshatra Sharma**, UFID: 4935-2902, nakshatra2312@ufl.edu

### Project Execution Instructions:

#### **For Server:**

```
mix escript.build
```

```
./bitcoin_mining k
```

- Where k represents the number of 0's in the bitcoin

#### **For Worker/Client**

```
./ bitcoin_mining server_ip_address
```

- Where server\_ip\_address represents the IP address of server

### Possible Exception:

While executing the script, it might happen that you get an error of econrefused. In such scenario, you must run command `epmd -daemon` in Linux system.

For windows you should run the `epmd.exe` from `C:\Program Files\erl9.0\erts-9.0\bin`

### Implementation Details:

1. **Work Unit:** The nonce generation process for our Bitcoin Mining project should be such that
  - It should not be completely dependent on any randomness function
  - The work must be divided equally among all the miner workers
  - It should not have an upper bound and scalable to any number of workers, which can utilize it simultaneously without exhausting the range
  - It should assign unique work to all the workers

Keeping above points in mind we followed an incremental approach for nonce generation.

The strings are generated by encoding (with base 64) a string integer series starting at "1", which gets incremented by 1 in every iteration.

Number	Encoding	Nonce Generated
"1"	Base.encode64("1")	MQ==
"2"	Base.encode64("2")	Mg==
"3"	Base.encode64("3")	Mw==
.	.	.
.	.	.
.	.	.
"1000000"	Base.encode64("1000000")	MTAwMDAwMA

We have created an assign\_work module which assigns work to server as well as client workers as a range of work unit. Each work unit range consists of **1000000** string integers. Workers receive a range of integers and encode each number in the range to base 64 to generate string. Once a worker completes a unit of work, it asks for more work from the server and our module assigns the next available range to the worker.

The worker takes around 20 – 30 seconds to use the range (work unit) of 1000000 (million) numbers. If we reduce this range, then the workers flood the message queue of server too frequently and till the time server does not respond back to worker, the worker remain in idle state. With a range of million numbers, we got the average performance of 700% CPU utilization for the 8-core machine. We could easily keep track of the work being executed by different workers and the workers does not flood message queue too frequently of server.

Workers	Range
Server	1-1000000
Worker1	1000001-2000000
Worker2	2000001-3000000
.	.
.	.
Worker1 asks for rework	5000000 - 6000000
Worker2 asks for rework	6000000 - 7000000
.	.

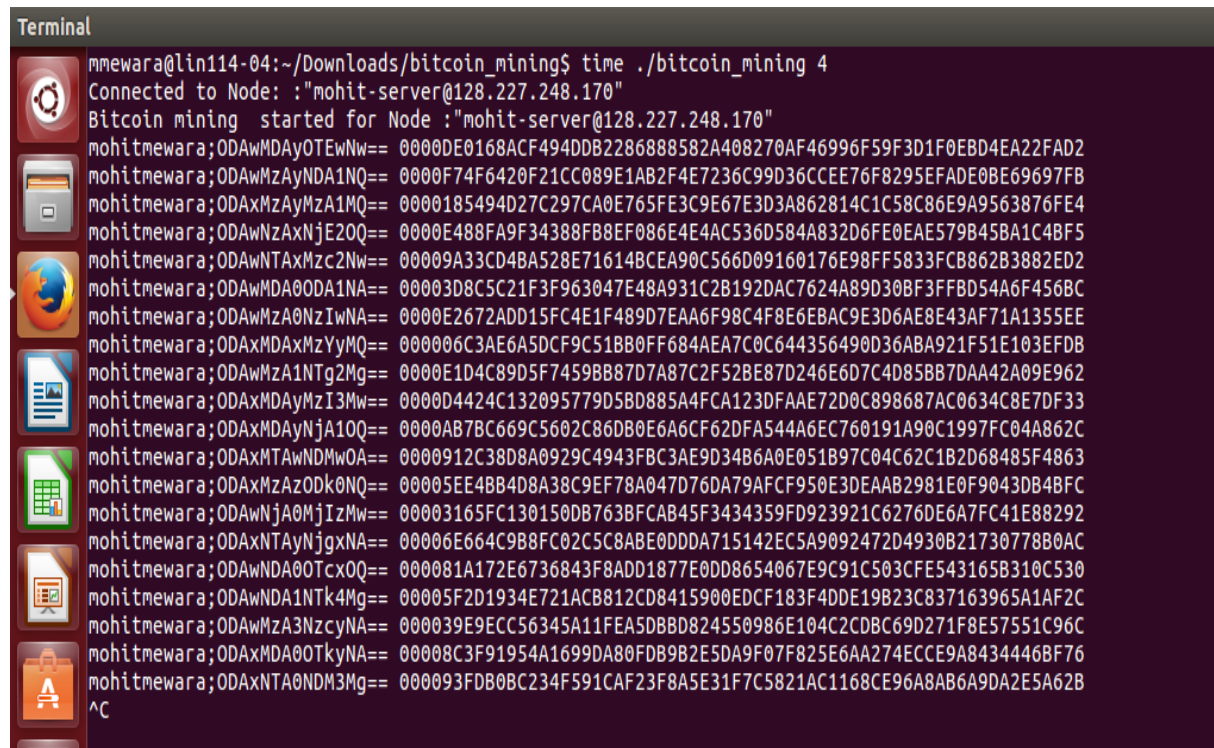
**The benefits of the above approach are:**

- Since multiple strings can never generate same encoded string, our nonce will remain **unique for every worker** and hence there will never be any duplicate work.

- We are dividing the range equally, hence the task is being **equally divided among workers**
- It makes our model **scalable**, as there is no upper bound on range of numbers and any number of workers can connect the server and ask for work.

2. The **result** of running the program for 4 O's bitcoin:

Please find below the screenshot:



```

Terminal
mmewara@lin114-04:~/Downloads/bitcoin_mining$ time ./bitcoin_mining 4
Connected to Node: "mohit-server@128.227.248.170"
Bitcoin mining started for Node: "mohit-server@128.227.248.170"
mohitmewara;ODAwMDAyOTEwNw== 0000DE0168ACF494DDB2286888582A408270AF46996F59F3D1F0EBD4EA22FAD2
mohitmewara;ODAwMzAyNDA1NQ== 0000F74F6420F21CC089E1AB2F4E7236C99D36CCEE76F8295EFADE0BE69697FB
mohitmewara;ODAwMzAyMzA1MQ== 0000185494D27C297CA0E765FE3C9E67E3D3A862814C1C58C86E9A9563876FE4
mohitmewara;ODAwNzAxNjE2OQ== 0000E488FA9F34388FB8EF086E4E4AC536D584A832D6FE0EAE579B45BA1C4BF5
mohitmewara;ODAwNTAxMzc2Nw== 00009A33CD4BA528E71614BCEA90C566D09160176E98FF5833FCB862B3882ED2
mohitmewara;ODAwMDA0ODA1NA== 00003D8C5C21F3F963047E48A931C2B192DAC7624A89D30BF3FFBD54A6F456BC
mohitmewara;ODAwMzA0NzIwNA== 0000E2672ADD15FC4E1F489D7EAA6F98C4F8E6EBAC9E3D6AE8E43AF71A1355EE
mohitmewara;ODAwMDAxMzYyMQ== 000006C3AE6A5DCF9C51BB0FF684AEA7C0C644356490D36ABA921F51E103EFD8
mohitmewara;ODAwMzA1NTg2Mg== 0000E1D4C89D5F7459BB87D7A87C2F52BE87D246E6D7C4D85BB7DAA42A09E962
mohitmewara;ODAwMDAyMzI3Mw== 0000D4424C132095779D5BD885A4FCA123DFAAE72D0C898687AC0634C8E7DF33
mohitmewara;ODAwMDAyNjA1OQ== 0000AB7BC669C5602C86DB0E6A6CF62DFA544A6EC760191A90C1997FC04A862C
mohitmewara;ODAwMTAwNDMwOA== 0000912C38D8A0929C4943FBC3AE9D34B6A0E051B97C04C62C1B2D68485F4863
mohitmewara;ODAwMzAzODk0NQ== 00005EE48B4D8A38C9EF78A047D76DA79AFCF950E3DEAAB2981E0F9043DB48FC
mohitmewara;ODAwNjA0MjIzMw== 00003165FC130150DB763BFCAB45F3434359FD923921C6276DE6A7FC41E88292
mohitmewara;ODAwNTAyNjgxNA== 00006E664C9B8FC02C5C8ABE0DDA715142EC5A9092472D4930B21730778B0AC
mohitmewara;ODAwNDA0OTcxOQ== 000081A172E6736843F8ADD1877E0DD8654067E9C91C503CFE543165B310C530
mohitmewara;ODAwNDA1NTk4Mg== 00005F2D1934E721ACB812CD8415900EDCF183F4DDE19B23C837163965A1AF2C
mohitmewara;ODAwMzA3NzcyNA== 000039E9ECC56345A11FEA5DBBD824550986E104C2CDBC69D271F8E57551C96C
mohitmewara;ODAwMDA0OTkyNA== 00008C3F91954A1699DA80FDB9B2E5DA9F07F825E6AA274ECCE9A8434446BF76
mohitmewara;ODAwNTA0NDM3Mg== 000093FDB0BC234F591CAF23F8A5E31F7C5821AC1168CE96A8AB6A9DA2E5A62B
^C

```

3. We ran our program 'bitcoin\_mining' on an **Intel i7 machine with 8 cores** in UF Computer Science Lab (Dungeon Lab) and recorded the below timings:

```

Terminal
mmewara@lin114-04:~/Downloads/bitcoin_mining$ mix escript.build
mmewara@lin114-04:~/Downloads/bitcoin_mining$ time ./bitcoin_mining 5
Connected to Node: "mohit-server@128.227.248.170"
Bitcoin mining started for Node: "mohit-server@128.227.248.170"
mohitmewara;ODA1MjIwNw== 00000CE374CA9AC79386591D63DBB16B48A8C1427BF4830D928BF0C4350C269C
mohitmewara;MTExMTQ2NQ== 000005A31939F8528D91A1A308583B45D19B1E0FEC7085DE1AA0853CFB20AF8
mohitmewara;MTQwODkwOTA= 00000815E1BB3ED96E75F0388BDE110BF96CBB848CB6EA39D73D876AD4CA88A6
mohitmewara;MTQwOTgzMDE= 0000012CAC14843F0F0A4F16FD36039C05AD62A92ACDD23564B4C7FE96009A1F
mohitmewara;MTQxMzk5NTQ= 00000E4ED18B6664D04C039E59F845B69108952E1344A4E7815476DD2E310B8C
mohitmewara;NjE0MTM2Nw== 000009C87542D39A4BC7045471D70FF7E135C4F21A8B98D95AE3EA9D1D7B981A
mohitmewara;MTExNTIxNTk= 00000147B5E2047AEC03E0FC1F5998FC0A18D297FAF29FCB54FCC2757DE63ACD
mohitmewara;MTIXOTMwOTM= 00000518A2DA8A1619DA2F0E6922ECB6661FC9E31AF8DE0C8006D7085E78F0A1
mohitmewara;Mjk5ODAw 00000B56E41977E508689EF1697CDEBB98F786729922819CDAE28544E7A22643
mohitmewara;Mzg5ODU1 0000094E2BF0E51E7D56C40858D6529AF9A78CE135917EF2044BE61F183DC623
mohitmewara;OTQxMjYxNQ== 00000FC28A889464A4962B2F903C0AD0BB9D8F9C5A08B5E536E5694D3FC5DF51
mohitmewara;MTQyODI3OQ== 00000361AD4FAF5ECD5ACBB44FEF3F13B242668ACE2F0D8BD8F26AF5833E6C2A
mohitmewara;MTU0Mjc5NDM= 0000030488C8956F6F254BE548F876A00A689978604D81367F1E04DBBDA7580
mohitmewara;NjQzNDU5MA== 00000BFE85343A281A1BB8B9661D9595AF41D97206724378B9FA50A31184FEAA
mohitmewara;MzQ4MjU2Mw== 000007C93FCBFBFE7EA0E75A5A270978B10663E63C18F79940B05B6F7545B3AA
^C
real    1m56.977s
user    13m30.256s
sys     1m22.160s
mmewara@lin114-04:~/Downloads/bitcoin_mining$

```

**Real :** 1m56.977sec = 60+56 → 116 seconds

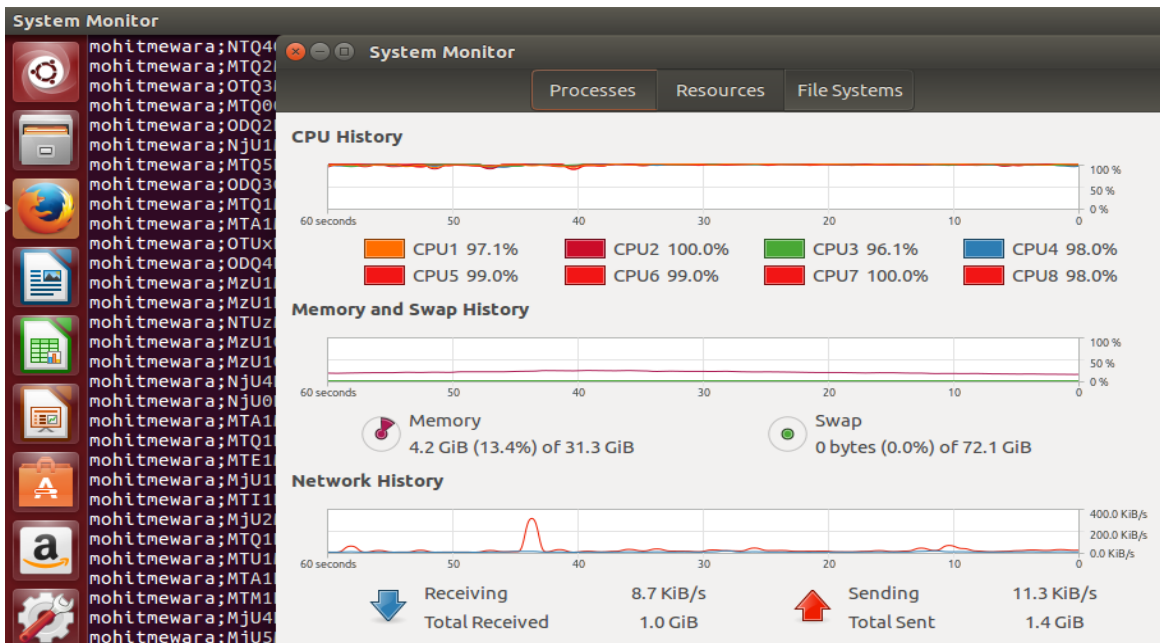
**User :** 13m30.256sec = 60+56 → 810 seconds

**Sys :** 1m22.160sec = 60+56 → 82 seconds

Ratio of CPU time to Real time = User/Real = 810/116 = **6.98 ~ 7**

Hence our project is giving a performance of **700%**.

The below is the screenshot of CPU utilization of all the cores. Almost all the cores are working on 100% CPU utilization.



4. We found the below coin with most number of zeroes:

storm.cise.ufl.edu - PuTTY

```
storm:10% mix escript.build
warning: mtime (modified time) for "lib/mining/listener.ex" was set to the future, resetting to now
Compiling 1 file (.ex)
Generated escript bitcoin_mining with MIX_ENV=dev
storm:11% ./bitcoin_mining 7
Connected to Node: : "mohit-server@128.227.205.236"
Bitcoin mining started for Node : "mohit-server@128.227.205.236"
mohitmewara;MTMzMDk4MTUyOA== 000000052A1EA17E076695CBA51763D85205DA99816B32EC98500861E079A0BC

^C
storm:12% █
```

mohitmewara;MTMzMDk4MTUyOA==  
000000052A1EA17E076695CBA51763D85205DA99816B32EC98500861E079A0BC  
Number of zeroes in our Bitcoin: **7**

5. The largest number of working machines we tested our code, was with 4 machines where 1 machine acted as a server and other three workers acted as clients.