Project Report On

Cluster Mining Optimization using Dynamic Bin Nonces

Submitted by

Hrishikesh Thakkar (15IT120)
Pratyush Prakash (15IT130)
Akshay U Prabhu (15IT203)

VI SEM B.Tech (IT)

Under the guidance of

Ananthanarayana V. S.
Dept of IT, NITK Surathkal

in partial fulfillment for the award of the degree

of

Bachelor of Technology
In

Information Technology
At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

May 2018

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

<u>End Semester Evaluation (March 2018)</u>

Course code : IT 399

Course Title: Minor Project

Title of the Project: Cluster Mining Optimization using Dynamic Bin Nonces

Details of Project Group

Name of the Student	*Register No.*	*Signature with Date*
1. Hrishikesh Thakkar	15IT120	
2. Pratyush Prakash	15IT130	
3. Akshay U Prabhu	15IT203	

Name of Project Guide: Ananthanarayana V. S.

Signature of the Project Guide:

Place: IT Department, NITK

Date: 2nd May, 2018

<u>CERTIFICATE</u>

This is to certify that the project entitled **"<u>Cluster Mining Optimization using Dynamic Bin Nonces</u>"** has been presented for Mid-Semester Evaluation by Pratyush Prakash, Hrishikesh Thakkar and Akshay U Prabhu students of third year, B.Tech (IT), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, on May_2018, during the even semester of the academic year 2017- 2018, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology at NITK, Surathkal.

Place: NITK, Surathkal

Date: 03/05/2018 (Signature of the Examiner)

Abstract

Bitcoin mining and transactions are a thriving area of computing and investments. Due to sharp increase of number of miners, groups are made consisting of many miners. This is referred to as group mining, where a batch of miners work together in solving the cryptographic puzzle given to them. The distribution of nonce values is done by keeping static bins of a particular size and the server sending it to the miner when he requests a new bin. Though this method is currently operational, it has a few drawbacks related mainly to network latency. In our project we have proposed a method to allocate the bins dynamically thereby minimizing the network latency.

Table of Contents

List of Figures

List of Tables

Introduction

Our project deals with the concept of blockchain and its major use case ‘Bitcoin’. Blockchain works as a distributed ledger which uses the majority of the populus to determine the validity of transactions across the network. It involves public key encryption which uses digital signatures to secure the data being transferred and keep it confidential.

Once the concept of a distributed ledger was conceived, the next step was to implement transactions which can be seen by all members of the network. The use of this technology resulted in the advent of the first decentralized cryptocurrency. Bitcoin had taken the world aback, it instantly began to be used more than what was originally thought to be. In 2017 1 BTC was equivalent to $15,000 USD.

The requirement of Bitcoin is a concept referred to as mining. Mining involves solving a difficult cryptographic puzzle. For these puzzles expensive GPUs and ASICs are required, to mitigate this groups of people work together to solve the puzzle. This grouping is referred to as ‘clusters’ and the action of collaborative mining is commonly referred to as ‘cluster mining’ which is what our project is centered around.

1.1. Motivation

Blockchain and Cryptocurrencies have been a growing field of interest of the past decade, however it is not feasible for everyone due to expensive hardware requirements. To ensure that the hardware is used to its maximum , an algorithm which could solve that problem can potentially save plenty of resources.

Literature Review

2.1. Outcome of Literature Review

From the literature review conducted it was determined that the allocation of the bin sizes used in the distribution of nonces amongst cluster-miners can be optimized to a dynamic manner rather than the current static allocation. This is to ensure that miners with a better hash rate are unhindered by network latency and their performance is maximized.

2.2. Problem Statement

Given a Bitcoin network with cluster miners the objective is to mine the blocks efficiently by allocating dynamic bins for each user according to their mining rates and previous performance.

2.3. Research Objectives

- To create an algorithm for creation of dynamic allocation of the bin size.
- To reduce the overall down time of mining.

2.4 Requirements analysis

- Functional Requirements
  - The latency should be reduced to minimize network congestion.
  - Increase the overall time of hashing by adjusting the nonce.
  - Minimize downtime
- Non Functional Requirements
  - Security
  - Performance
  - Accuracy
- Use Case Scenarios
  - The sole use case of this application is optimizing cluster mining

Methodology and Framework

3.1. System Architecture

Solo mining is just a single miner who exhausts his resources individually to solve the cryptographic puzzle. This technique is almost non existent as cluster miners have completely taken over the mining sphere. As shown in Fig 1

Mining Pools are synchronous group of miners, over the official pool mining protocols.. The have configured mining equipment which allow them to connect to a mining server using a pool account. Mining Hardware is used by the pool whilst the mining procedure goes on. Thus, the miners are distributed their rewards as per hashes calculated. As shown in Fig(2)

Fig 1: Solo mining

Fig 2: Pool mining

The main server adjust the difficulty in favor of the smaller pool miners so they may win a share of the earnings so as to keep encouraging them to mine and contribute to the pool. This reduced difficulty ensures that spoof hashes are not being generated and that each miner is given an opportunity to show his work done. The server then needs to verify the hashes are of the reduced difficulty and distribute the reward on this criteria. Also by continuously iterating through the reduced difficulty hashes the actual difficulty hash is found as well.

- Client - A node connecting to a mining pool to participate in shared mining
- Server - The coordination server which is responsible for distributing the reward.

The candidate block is constructed by the pool server using the UTXOs and adding the coinbase transaction, merkle root and the previous block hash. This forms the header which is sent to the pool miners and the successful hashes are relayed back to the server for verification

It is important to note that the client need not be connected to a full bitcoin node, but only to the coordination server.

The client nodes implement a double SHA256 [5] on the block header with the nonce to ensure that the hash is within the difficulty set by the pool.

The algorithm used to calculate the bitcoins in the coinbase transaction or the reward bitcoins is as follows:

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

The Mining algorithm which is being used in the project:

```
P := The hash of the previously mined block
B := A block of transactions
H := A hash function
D := Difficulty Level

0 Retrieve P
1 Construct/Modify B
2 IF H(P, B, Some Random Number) > D END
3 GOTO 1
```

The Pool coordination server or pool master will communicate with the clients via HTTP. The server will use a modified GetBlockTemplate standard (Table 1).

Table 1: The Block Header Template

Size	**Field**	**Description**
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Target	The Proof-of-Work algorithm target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

Table 2: The Transaction Template

The server will provide the following endpoints for each client to request information,

- GET /getblock - Client makes a request at this url to receive a block template on which they have to work on.
- POST /newclient - New clients make a POST request on this url to receive an id, and register themselves with the server.

- POST /getbin - Client nodes make a request on this url to receive their next bin of nonces. The server should calculate the bin size on the basis of the past performance of the node.
- POST /reduce_difficulty_hash - Client nodes send their calculated reduced difficulty hashes to the server via this url. These hashes are verified by the user and used for payout calculation.
- GET /status - Clients make a request to this url to know the status of the current block being mined.
- POST /my_status - Clients make a request to this url to know their performance over the session.

The workflow of a client would be:

- Join the mining pool by registering with the pool server.
- Request a block template.
- Request a bin of nonces.
- Perform hashes of the nonces in the bin with the block template. Send any reduced difficulty hashes found back to the server.
- Request a new bin of nonces when finished with the nonces in the current bin.
- Periodically check the pool status to ensure that hashing is being performed with the latest block template.

Work Done

4.1. Experimental Framework

The implementation is carried out in Golang using the 'BTCD' Framework. The client side required functionality to find hashes of the blockheader in the nonce range given to it. The nonces are of two types, the main nonce in the blockheader and the extra-nonce in the coinbase transaction.

The Btcd framework allows for the creation of the modified coinbase transaction with the extra-nonce.

For the purposes of demonstration, a pool mining server was created, and several client nodes were requesting bins of nonces from the server.

4.2. CPU Mining

Although CPU Mining is not an efficient way to mine cryptocurrencies, we have chosen to keep our implementation restricted to CPU mining since GPU mining requires extra paradigms and expensive hardware.

It must be noted however, that comparisons between standard pool mining and dynamic bin mining can be made by taking only CPU mining in consideration.

4.3 Results and Discussion

We have devised a methodology to pre-emptively find the clients which are going to need a new bin, before they exhaust their nonce ranges. We have implemented a CPU miner and data parser for the client node. As per the figure (fig. 3) we see that the Dynamic binning reduces the number of requests sent for the same nonce range.

Figure 3: Requests for each Binning Technique

4.4. Individual Contribution of project members

1. Hrishikesh Thakkar - Data processing on client side[1]
2. Akshay Prabhu - Bin allocation strategies for nonce distribution, extra-nonce embedding in coinbase transaction[2]
3. Pratyush Prakash - Orphan block detection, false mining and cpu miner implementation

Conclusion and Future Work

As we observed, there were fewer requests made by clients per nonce range in dynamic binning when compared to static binning. There are a few aspects in which we can improve on further.

1. Reward Calculation
2. Use of the extra nonce in dynamic allocation
3. Connecting to the Blockchain network

5.1. Proposed Work Plan of the project

We plan to implement the JSON-RPC server which will be full bitcoin node and capable of handling bin request from a large concurrent base of users.

Testing will be done on the Bitcoin Testnet 1 network, wich resets it's difficulty every few blocks, this ensures CPU mining can be used to test our pool mining cluster.

References

1. Satoshi Nakamoto, 'Bitcoin Whitepaper ,2009' https://bitcoin.org/bitcoin.pdf
2. Bitcoin Community ,2018 https://en.bitcoin.it/wiki/Getblocktemplate
3. Pallets Team, 2010 'Server Client Flask' http://flask.pocoo.org/docs/0.12/tutorial/
4. 2017 BLOCKCHAIN LUXEMBOURG , ''https://blockchain.info
5. Bitcoin Community ,'SHA-256 explained,2016' https://en.bitcoin.it/wiki/SHA-256