Java Redefined

Friday, 22 December 2017

Java Blockchain Mining and Consensuses

Mining and Consensuses in Blockchain

In the previous blog we got to know what a Blockchain is, and how you can we create a Java based Blockchain which can not be tampered (Integrity).

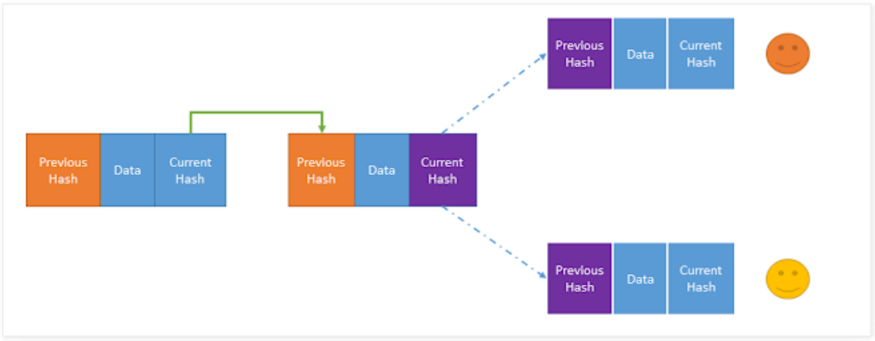
Blockchain has couple of more design principles other than Integrity : Decentralized, Consensuses, Public Ledger etc. Before we jump to Mining it is important to understand these concepts first.

Public Ledger

Assume that different participants are adding transactions and blocks to a Blockchain. That means there has to be a central Blockchain that is accessible to all, since all participants needs to add block to the existing Blockchain, This means there has to be a master participants governing and storing the Blockchain. But this is against the principle of Blockchain (Decentralized). There is no master participant in the network. To solve the above issue, each participant keeps a copy of the Blockchain with itself. Once a block is added, it is distributed across the network and all participants update their Blockchain. This concept is called Public Ledger. A ledger (list of transactions) which is public.

But this causes another problem in the network.

Lets assume that 2 participants are trying to add a block, simultaneously to a Blockchain. Only one can be allowed to do so and the other has to be discarded since the previous hash in that block will be invalid.



As seen above only one participant can attach its Block, the other participant has to re-calculate the hash since the blockchain is changed now.

But which participant should be allowed to attach the block?
In case of private businesses where participants are trusted, the participant having the highest share can be given the opportunity to add the block.
But this does not work in a peer-to-peer non trusted network like Bitcoin,

Proof of work

Bitcoin introduced a different technique to solve the conflict of interest, called the ***Proof of work***
The participant who can solve a complex mathematical problem first and submit the proof of work wins. Most of the time the problem is so complex that participants team up to solve it.

Now what is the complex calculation ? It is in a certain way the Current Hash of the block needs to be created. In the previous blog we saw how to generate a CurrentHash, it is a combination of Data + Timestamp and Previous Hash. The output can be any aplha-numeric key. The proof of work asks the participants to create hash preceding by certain number of "0"s.

For e.g:

A Hash of "Hello" lets say is *x435845dlasdhtow575927uhjsdf83650927164dwerksh,* the proof of work requires the participant to create a hash with preceding two "0"s. The participant will than try to get to a hash which has two "0"s at the beginning of the hash. For this he may try to append the Hello string with a integer :

"Hello1" - *x43584sds5dlasdhtow575927uhjsdf83650927164dwerksh*
"Hello2" - *xdfsdf5dlasdhtow5759454527uhjsdf8365097e16sdfsdfsdfs*
.
.
.

Featured post

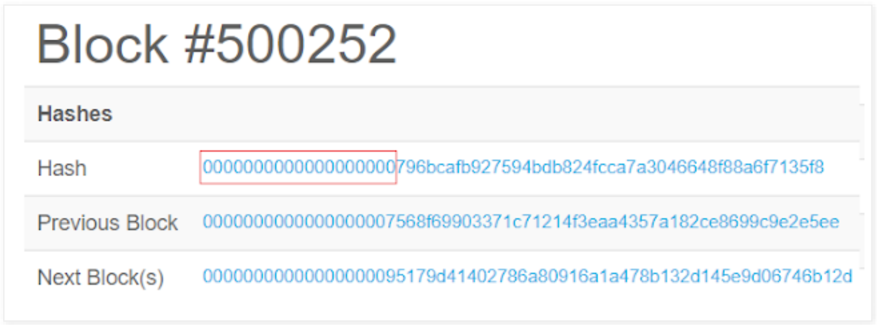
JAVA and OOPS with answeres

Since now a days the
job has increased, co
process. This screen

"Hello4521" - 00*lskdfs8oiklj45j34590sdflkjh2348sadlas834lksdf89*

and after some hit-and-try after 4521 hashes he may get his hash which has two "0"s at the beginning.

The number of "0"s required is also called difficulty in Bitcoin terms. The current difficulty as of today's date is eighteen "0"s. This can be viewed at blockchain.info



Block #500252

Hashes	
Hash	000000000000000000796bcafb927594bdb824fcca7a3046648f88a6f7135f8
Previous Block	0000000000000000007568f69903371c71214f3eaa4357a182ce8699c9e2e5ee
Next Block(s)	000000000000000000095179d41402786a80916a1a478b132d145e9d06746b12d

Mining

This process of calculating hash is also called Mining in Crypto-currency domain

Lets modify our Java Blockchain to accommodate this logic

We will represent the integer that is appended to data to generate the hash as **nonce**.

The changed Block class looks like this now:

```
import java.util.Date;

public class Block {

 public String currentHash;
 public String previousHash;
 private String data; //our data will be a simple message.
 private long timeStamp; //as number of milliseconds since 1/1/1970.
 private int nonce;

 //Block Constructor.
 public Block(String data,String previousHash ) {
  this.data = data;
  this.previousHash = previousHash;
  this.timeStamp = new Date().getTime();
  this.currentHash = calculateHash();
 }

 public String calculateHash() {
  String calculatedhash = HashUtil.applySha256(
    previousHash +
    Long.toString(timeStamp) +
    Integer.toString(nonce) +
    data
    );
  return calculatedhash;
 }
}
```

We have added a integer called **nonce** and it is used in calculating the Currenthash as well. We will also add a method called mineBlock that calculates the required hash based upon the difficulty in the Block class.

```
 //Increases nonce value until hash target is reached.
 public void mineBlock(int difficulty) {
  String target = new String(new char[difficulty]).replace('\0', '0');
  while(!currentHash.substring( 0, difficulty).equals(target)) {
   nonce ++;
   currentHash = calculateHash();
  }
  System.out.println("Block Mined!!! : " + currentHash);
 }
```

Create replica databa couple

Share the post

G+ 0 Like

Blog Archive

- ► 2018 (1)
- ▼ 2017 (2)
  - ▼ December (2)
    - Java Blockchai Mining and Consensuse
    - Java Blockchai tutorial
- ► 2016 (6)
- ► 2015 (9)
- ► 2014 (22)
- ► 2013 (19)

Popular Posts

Java Collections :

Java Custom Arra

Symmetric Key Er

Data Structure : D

Data Structure : B

Create Custom Ar Artifactory

Java Collection In

Custom Java Has

Google+ Badge

Here until the desired hash is reached we increment the nonce and use it in the calculation of hash.

After every block is created we will call this method to calculate the hash based upon the difficulty set.

We need to call mineBlock from addBlock method of out test class, before adding the block to the blockchain

Video Bar

```
import java.util.ArrayList;
import java.util.List;

public class Test {

 static List blockChain = new ArrayList();

 public static int difficulty = 5;

 public static void main(String[] args) {

  addBlock(new Block("First Block", "0"));
  addBlock(new Block("Second Block", blockChain.get(blockChain.size() - 1).currentHash));
  addBlock(new Block("Third Block", blockChain.get(blockChain.size() - 1).currentHash));

  for (Block b : blockChain) {
   System.out.println(b);
  }

  addBlock(new Block("Thief Block",blockChain.get(blockChain.size() - 2).currentHash));

 }

 private static boolean isBlockChainValid(List blockChain) {
  if (blockChain.size() > 1) {
   for (int i = 1; i <= blockChain.size()-1; i++) {
    Block currentBlock = blockChain.get(i-1);
    Block nextBlock = blockChain.get(i);
    if (!(nextBlock.previousHash.equals(currentBlock.currentHash))) {
     return false;
    }
   }
  }
  return true;
 }

 public static void addBlock(Block b) {
 b.mineBlock(difficulty);
 blockChain.add(b);
 if(!isBlockChainValid(blockChain)) {
    System.out.println("Block :"+b.currentHash + " - " +  b.previousHash +" is not valid, removing it");
  blockChain.remove(b);
 }
 }

}
```

Here a static variable is used to set the difficulty as 5. And in the addBlock method mineBlock is called before adding the block to the blockchain.

Also we are removing the block if it is not a valid one.

Now lets run the program:

```
Block Mined!!! : 0000057cc51099f519519a03cd24465c19908cda253d38e665302e239fadac8d
true
Block Mined!!! : 000000aeb99ac981b0edfbe75aac8a9cae4e794c9f8ea0f15008813ed0af86ff
true
Block Mined!!! : 00000f38e87c5b3ecaa818cd64eb7144f001a98881b0848e7acd980f5f3e6a35
true
{
  currentHash=0000057cc51099f519519a03cd24465c19908cda253d38e665302e239fadac8d
  previousHash=0
  data=First Block
  timeStamp=1513939177198
  nonce=665708}
{
  currentHash=000000aeb99ac981b0edfbe75aac8a9cae4e794c9f8ea0f15008813ed0af86ff
  previousHash=0000057cc51099f519519a03cd24465c19908cda253d38e665302e239fadac8d
  data=Second Block
  timeStamp=1513939178452
  nonce=909296}
{
  currentHash=00000f38e87c5b3ecaa818cd64eb7144f001a98881b0848e7acd980f5f3e6a35
  previousHash=000000aeb99ac981b0edfbe75aac8a9cae4e794c9f8ea0f15008813ed0af86ff
  data=Third Block
  timeStamp=1513939180407
  nonce=328890}
Block Mined!!! : 00000039c71fa906e7c5f4647347dc5673e0491ce12220b2a69f1a73783b1bdd
false
```

Here it can be seen that all the hashes in the blocks have five preceding "0"s. Also from the nonce it can be determined how many iterations of hashes were done to get the correct hash result.
Mostly a desktop computer can mine with difficulty level up-to 7. Above this difficulty you need a powerful processor, ideally a graphics card. Hence most miners use graphics card and team up with other miners to calculate the hash, since the difficulty level has reached 18.

A typical mining machine looks like this. It has around 8-10 graphics cards.

Typically the data in a Bitcoin Blockchain is made up of transactions that happens between different participants.

A Transaction is made up of the **Sender Address**, **Receiver Address** and the **Amount** transferred. This is how a Transaction and Block class would look.

```
class  Transaction {
 String senderAddress;
 String recevierAddress;
 Integer amountTransfered;
 long timestamp;
}

class Block {

 String prevoiusHash;
 String currentHash;
 long timestamp;
 Transaction[] transactions;
}
```

The complete code for Bitcoin Mining with transactions is provided here :

https://github.com/hunaidee007/Java-Bitcoin-Minning

at

Reactions: funny (0) interesting (0) cool (0)

Labels: Bitcoin, Blockchain, Mining

1 comment:

preethi Shetty 20 March 2018 at 16:29

This is what I am looking for since days ago finally got the useful stuff. Thanks for sharing this post do keep sharing on latest updates...
Benefits of Bockchain Technology | Blockchain Training

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

LinkWithin

PrettyPrint