比特币源码研读系列5 挖矿过程PoW

前面我们陆续介绍到了block中的挖矿难度系数target value和用于SPV的hashMerkleRoot值的产生。本篇我们来介绍下挖矿的主体过程，看完本篇你可能就明白了为什么大家都说挖矿是一个费时费电费能源的过程了。

**PoW介绍**

PoW是proof of work的简写，中文称之为工作量证明，它的主要过程如下伪代码所示：

**Do repeatedly** output = f (transactions, previous Block ID， an additional random guess)，**until** output < target value，

**Then** Block ID = the output.

当得到有效的Block ID也就意味着生成了一个有效的Block，代表着成功挖到一个矿。

我们再来看下block的结构

|  |
| --- |
| **class CBlock** |
| int32\_t nVersion; |
| uint256 hashPrevBlock; |
| uint256 hashMerkleRoot; |
| uint32\_t nTime; |
| uint32\_t nBits; |
| uint32\_t nNonce;  vector<CTransactionRef> vtx |

到目前为止，nBits就是上面说到的挖矿难度系数target value，vtx是加入到block的那些tx，hashMerkleRoot是tx的merkle tree的树根，这几个我们就已经介绍过了，我们来看下nNonce，也就是an additional random guess的产生过程。

**generateBlocks**就是生产多个block的方法，下面我们来看下它的代码。

*UniValue generateBlocks(std::shared\_ptr<CReserveScript> coinbaseScript, int nGenerate, uint64\_t nMaxTries, bool keepScript)*

*{*

*static const int nInnerLoopCount = 0x10000;*

*int nHeightEnd = 0;*

*int nHeight = 0;*

*{ // Don't keep cs\_main locked*

*LOCK(cs\_main);*

*nHeight = chainActive.Height();*

*nHeightEnd = nHeight+nGenerate;*

*}*

*unsigned int nExtraNonce = 0;*

*UniValue blockHashes(UniValue::VARR);*

*while (nHeight < nHeightEnd)****//挨个生成有效的block***

*{*

***//* *CreateNewBlock方法产生一个新的block，它要完成如下工作：***

1. ***产生一个空的block***
2. ***向block添加一组tx，包括coinbase tx。添加tx的过程我们前面已经描述过***
3. ***产生nBits。***

***因此我们看到到此产生的新block基本只差PoW验证通过就会是有效的block了。***

*std::unique\_ptr<CBlockTemplate> pblocktemplate(BlockAssembler(Params()).CreateNewBlock(coinbaseScript->reserveScript));*

*if (!pblocktemplate.get())*

*throw JSONRPCError(RPC\_INTERNAL\_ERROR, "Couldn't create new block");*

*CBlock \*pblock = &pblocktemplate->block;*

*{*

*LOCK(cs\_main);*

*IncrementExtraNonce(pblock, chainActive.Tip(), nExtraNonce);****//计算merkle tree root。***

*}*

***//下面这个while循环就是PoW过程…nNonce从0开始不停加1，然后CheckProofOfWork验证blockID是否有效，直到CheckProofOfWork返回true，表示挖矿成功。nNonce最大尝试次数为nMaxTries，nMaxTries默认为100万。因此如果再尝试了100万次后还没发现有效的blockID则放弃本次挖矿。***

*while (nMaxTries > 0 && pblock->nNonce < nInnerLoopCount && !CheckProofOfWork(pblock->GetHash(), pblock->nBits, Params().GetConsensus())) {*

*++pblock->nNonce;*

*--nMaxTries;*

*}*

*if (nMaxTries == 0) {*

*break;*

*}*

*if (pblock->nNonce == nInnerLoopCount) {*

*continue;*

*}*

*std::shared\_ptr<const CBlock> shared\_pblock = std::make\_shared<const CBlock>(\*pblock);*

*if (!ProcessNewBlock(Params(), shared\_pblock, true, nullptr))****//挖矿成功后ProcessNewBlock方法再次检查下新产生的block的一些数据是否正确和有效，同时向网络中其他用户广播该block。***

*throw JSONRPCError(RPC\_INTERNAL\_ERROR, "ProcessNewBlock, block not accepted");*

*++nHeight;*

*blockHashes.push\_back(pblock->GetHash().GetHex());*

*//mark script as important because it was used at least for one coinbase output if the script came from the wallet*

*if (keepScript)*

*{*

*coinbaseScript->KeepScript();*

*}*

*}*

*return blockHashes;*

*}*

到此，我们便简单介绍完比特币中挖矿的全部过程了。比特币源代码的研读之旅我们也就暂告一段落了。