比特币源码研读系列5 挖矿（2）

上一篇我们介绍了挖矿中最关键的target value值的来源及调整，处于便于理解的目的我们这里再次贴下挖矿过程：

**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。

target value是控制挖矿难度的因子，这个值越大，要找到比它小的值越容易，因此挖矿的难度越低。

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

前一篇我们已经介绍过block中会包含这些字段，从这篇开始我们来介绍下block中这些值是如何产生的？这个过程需要完成哪些工作？

1. **创建空block，然后装载一定数量的tx**
2. **计算PoW（挖矿），找到有效的blockId。**

下面我们逐一看看这个过程。

**CreateNewBlock**方法便是步骤1中的主要方法。下面我们看看它都做了哪些工作？

*std::unique\_ptr<CBlockTemplate> BlockAssembler::CreateNewBlock(const CScript& scriptPubKeyIn, bool fMineWitnessTx)*

*{*

*int64\_t nTimeStart = GetTimeMicros();*

***//初始化一个new block***

*resetBlock();*

*pblocktemplate.reset(new CBlockTemplate());*

*if(!pblocktemplate.get())*

*return nullptr;*

*pblock = &pblocktemplate->block; // pointer for convenience*

*// Add dummy coinbase tx as first transaction*

*pblock->vtx.emplace\_back();*

*pblocktemplate->vTxFees.push\_back(-1); // updated at end*

*pblocktemplate->vTxSigOpsCost.push\_back(-1); // updated at end*

***//以上是初始化block的一些信息***

*LOCK2(cs\_main, mempool.cs);*

*CBlockIndex\* pindexPrev = chainActive.Tip();*

*assert(pindexPrev != nullptr);*

*nHeight = pindexPrev->nHeight + 1;*

*pblock->nVersion = ComputeBlockVersion(pindexPrev, chainparams.GetConsensus());*

*// -regtest only: allow overriding block.nVersion with*

*// -blockversion=N to test forking scenarios*

*if (chainparams.MineBlocksOnDemand())*

*pblock->nVersion = gArgs.GetArg("-blockversion", pblock->nVersion);*

*pblock->nTime = GetAdjustedTime();****//更新block time***

*const int64\_t nMedianTimePast = pindexPrev->GetMedianTimePast();*

*nLockTimeCutoff = (STANDARD\_LOCKTIME\_VERIFY\_FLAGS & LOCKTIME\_MEDIAN\_TIME\_PAST)*

*? nMedianTimePast*

*: pblock->GetBlockTime();*

*// Decide whether to include witness transactions*

*// This is only needed in case the witness softfork activation is reverted*

*// (which would require a very deep reorganization) or when*

*// -promiscuousmempoolflags is used.*

*// TODO: replace this with a call to main to assess validity of a mempool*

*// transaction (which in most cases can be a no-op).*

*fIncludeWitness = IsWitnessEnabled(pindexPrev, chainparams.GetConsensus()) && fMineWitnessTx;****//是否启用SegWit更新。上上篇我们介绍过这个更新。***

*int nPackagesSelected = 0;*

*int nDescendantsUpdated = 0;*

*addPackageTxs(nPackagesSelected, nDescendantsUpdated);****//向block中添加一些tx。该方法是步骤1的主要方法，下面我们会具体来说。***

*int64\_t nTime1 = GetTimeMicros();*

*nLastBlockTx = nBlockTx;*

*nLastBlockWeight = nBlockWeight;*

***//block中的系统奖励是写在coinbase tx中的。Coinbase tx是一种没有inputs的tx***

*// Create coinbase transaction.*

*CMutableTransaction coinbaseTx;*

*coinbaseTx.vin.resize(1);*

*coinbaseTx.vin[0].prevout.SetNull();*

*coinbaseTx.vout.resize(1);*

*coinbaseTx.vout[0].scriptPubKey = scriptPubKeyIn;*

*coinbaseTx.vout[0].nValue = nFees + GetBlockSubsidy(nHeight, chainparams.GetConsensus());****//GetBlockSubsidy方法指的是block系统奖励。为什么比特币会每隔4年挖矿奖励减少一半，其中的原因就在GetBlockSubsidy方法中。***

*coinbaseTx.vin[0].scriptSig = CScript() << nHeight << OP\_0;*

*pblock->vtx[0] = MakeTransactionRef(std::move(coinbaseTx));*

*pblocktemplate->vchCoinbaseCommitment = GenerateCoinbaseCommitment(\*pblock, pindexPrev, chainparams.GetConsensus());*

*pblocktemplate->vTxFees[0] = -nFees;*

*LogPrintf("CreateNewBlock(): block weight: %u txs: %u fees: %ld sigops %d\n", GetBlockWeight(\*pblock), nBlockTx, nFees, nBlockSigOpsCost);*

*// Fill in header*

*pblock->hashPrevBlock = pindexPrev->GetBlockHash();****//指向前一个block***

*UpdateTime(pblock, chainparams.GetConsensus(), pindexPrev);*

*pblock->nBits = GetNextWorkRequired(pindexPrev, pblock, chainparams.GetConsensus());****//nBits便于挖矿的难度系数值，由GetNextWorkRequired方法产生，该方法我们在上一篇已经重点介绍过。***

*pblock->nNonce = 0;*

*pblocktemplate->vTxSigOpsCost[0] = WITNESS\_SCALE\_FACTOR \* GetLegacySigOpCount(\*pblock->vtx[0]);*

*CValidationState state;*

*if (!TestBlockValidity(state, chainparams, \*pblock, pindexPrev, false, false)) {*

*throw std::runtime\_error(strprintf("%s: TestBlockValidity failed: %s", \_\_func\_\_, FormatStateMessage(state)));*

*}*

*int64\_t nTime2 = GetTimeMicros();*

*LogPrint(BCLog::BENCH, "CreateNewBlock() packages: %.2fms (%d packages, %d updated descendants), validity: %.2fms (total %.2fms)\n", 0.001 \* (nTime1 - nTimeStart), nPackagesSelected, nDescendantsUpdated, 0.001 \* (nTime2 - nTime1), 0.001 \* (nTime2 - nTimeStart));*

*return std::move(pblocktemplate);*

*}*

下面我们再来**addPackageTxs**方法，该方法用于向一个block添加一些tx。

*void BlockAssembler::addPackageTxs(int &nPackagesSelected, int &nDescendantsUpdated)*

*{*

*// mapModifiedTx will store sorted packages after they are modified*

*// because some of their txs are already in the block*

*indexed\_modified\_transaction\_set mapModifiedTx;*

*// Keep track of entries that failed inclusion, to avoid duplicate work*

*CTxMemPool::setEntries failedTx;*

***//*** ***UpdatePackagesForAdded方法是把那些inBlock中的tx的子孙tx放到mapModifiedTx中。而mapModifiedTx则是那些本次可能会放入block中的tx。InBlock数组并不是所有在block的tx，下面我们会看到哪些tx会在inBlock中。需要注意的是，mapModifiedTx中的tx是按tx的分值来排序的。***

***//ancestor\_score是tx的分值，分值越高权重越大。nFee=money in vin – money in tx.vout，size为tx所占用的字节数。ancestor\_score计算规则为：***

***//a.nFee\*b.size > b.nFee\*a.size，则a的分值高，它意味着a的单位字节费用fee/size高于b，说明a的回报比b高，所以a的权重大。***

*// Start by adding all descendants of previously added txs to mapModifiedTx*

*// and modifying them for their already included ancestors*

*UpdatePackagesForAdded(inBlock, mapModifiedTx);*

*CTxMemPool::indexed\_transaction\_set::index<ancestor\_score>::type::iterator mi = mempool.mapTx.get<ancestor\_score>().begin();*

*CTxMemPool::txiter iter;*

*// Limit the number of attempts to add transactions to the block when it is*

*// close to full; this is just a simple heuristic to finish quickly if the*

*// mempool has a lot of entries.*

*const int64\_t MAX\_CONSECUTIVE\_FAILURES = 1000;*

*int64\_t nConsecutiveFailed = 0;*

*while (mi != mempool.mapTx.get<ancestor\_score>().end() || !mapModifiedTx.empty())*

*{*

***//遍历mempool中的tx，如果该tx已在block中，或在mapModifiedTx,或在failedTx中，则跳过该tx***

*// First try to find a new transaction in mapTx to evaluate.*

*if (mi != mempool.mapTx.get<ancestor\_score>().end() &&*

*SkipMapTxEntry(mempool.mapTx.project<0>(mi), mapModifiedTx, failedTx)) {*

*++mi;*

*continue;*

*}*

*// Now that mi is not stale, determine which transaction to evaluate:*

*// the next entry from mapTx, or the best from mapModifiedTx?*

*bool fUsingModified = false;****//*** ***fUsingModified为true表示使用mapModifiedTx中的tx，为false则使用mempool中的tx***

*modtxscoreiter modit = mapModifiedTx.get<ancestor\_score>().begin();*

*if (mi == mempool.mapTx.get<ancestor\_score>().end()) {*

*// We're out of entries in mapTx; use the entry from mapModifiedTx*

*iter = modit->iter;*

*fUsingModified = true;*

*} else {*

*// Try to compare the mapTx entry to the mapModifiedTx entry*

*iter = mempool.mapTx.project<0>(mi);*

*if (modit != mapModifiedTx.get<ancestor\_score>().end() &&*

*CompareModifiedEntry()(\*modit, CTxMemPoolModifiedEntry(iter))) {*

*// The best entry in mapModifiedTx has higher score*

*// than the one from mapTx.*

*// Switch which transaction (package) to consider*

*iter = modit->iter;*

*fUsingModified = true;*

*} else {*

*// Either no entry in mapModifiedTx, or it's worse than mapTx.*

*// Increment mi for the next loop iteration.*

*++mi;*

*}*

*}*

*// We skip mapTx entries that are inBlock, and mapModifiedTx shouldn't*

*// contain anything that is inBlock.*

*assert(!inBlock.count(iter));*

*uint64\_t packageSize = iter->GetSizeWithAncestors();*

*CAmount packageFees = iter->GetModFeesWithAncestors();*

*int64\_t packageSigOpsCost = iter->GetSigOpCostWithAncestors();*

*if (fUsingModified) {*

*packageSize = modit->nSizeWithAncestors;*

*packageFees = modit->nModFeesWithAncestors;*

*packageSigOpsCost = modit->nSigOpCostWithAncestors;*

*}*

*if (packageFees < blockMinFeeRate.GetFee(packageSize)) {*

*// Everything else we might consider has a lower fee rate*

*return;*

*}*

*if (!TestPackage(packageSize, packageSigOpsCost)) {****//*** ***TestPackage检查加入该tx后block大小是否超过最大值如1M，检查加入该tx后block的签名操作的消耗费用是否超过最大值。***

*if (fUsingModified) {*

*// Since we always look at the best entry in mapModifiedTx,*

*// we must erase failed entries so that we can consider the*

*// next best entry on the next loop iteration*

*mapModifiedTx.get<ancestor\_score>().erase(modit);*

*failedTx.insert(iter);*

*}*

*++nConsecutiveFailed;*

*if (nConsecutiveFailed > MAX\_CONSECUTIVE\_FAILURES && nBlockWeight >*

*nBlockMaxWeight - 4000) {*

*// Give up if we're close to full and haven't succeeded in a while*

*break;*

*}*

*continue;*

*}*

*CTxMemPool::setEntries ancestors;*

*uint64\_t nNoLimit = std::numeric\_limits<uint64\_t>::max();*

*std::string dummy;*

*mempool.CalculateMemPoolAncestors(\*iter, ancestors, nNoLimit, nNoLimit, nNoLimit, nNoLimit, dummy, false);****//*** ***CalculateMemPoolAncestors方法当最后一个入参是true时寻找该tx引用的在mempool的父tx。为false时则寻找该tx引用的在mempool的祖先tx。这里是寻找祖先tx(多代)，而不仅是父tx。***

*onlyUnconfirmed(ancestors);****//找到所有祖先tx后，再去掉已经在block中的tx***

*ancestors.insert(iter);*

*// Test if all tx's are Final*

*if (!TestPackageTransactions(ancestors)) {****//检查1. tx的locktime是否处于锁定状态，2. Segwit的更新功能是否启用。这是为了防止segwit尚未功能但是package里面已有segwit功能的tx。***

*if (fUsingModified) {*

*mapModifiedTx.get<ancestor\_score>().erase(modit);*

*failedTx.insert(iter);*

*}*

*continue;*

*}*

*// This transaction will make it in; reset the failed counter.*

*nConsecutiveFailed = 0;*

*// Package can be added. Sort the entries in a valid order.*

*std::vector<CTxMemPool::txiter> sortedEntries;*

*SortForBlock(ancestors, iter, sortedEntries);****//把block中的tx根据其祖先的数量来排序。比如说如果tx A引用了tx B，则A的祖先数量一定大有B。***

*for (size\_t i=0; i<sortedEntries.size(); ++i) {*

*AddToBlock(sortedEntries[i]);*

*// Erase from the modified set, if present*

*mapModifiedTx.erase(sortedEntries[i]);*

*}*

*++nPackagesSelected;*

*// Update transactions that depend on each of these*

*nDescendantsUpdated += UpdatePackagesForAdded(ancestors, mapModifiedTx);****//更新刚加入block的那些tx的状态。***

*}*

*}*

到此，我们便基本了解到了在挖矿时系统是如何挑选tx放入block了。好了，本篇到此为止，下篇我们来介绍下挖矿的PoW过程。