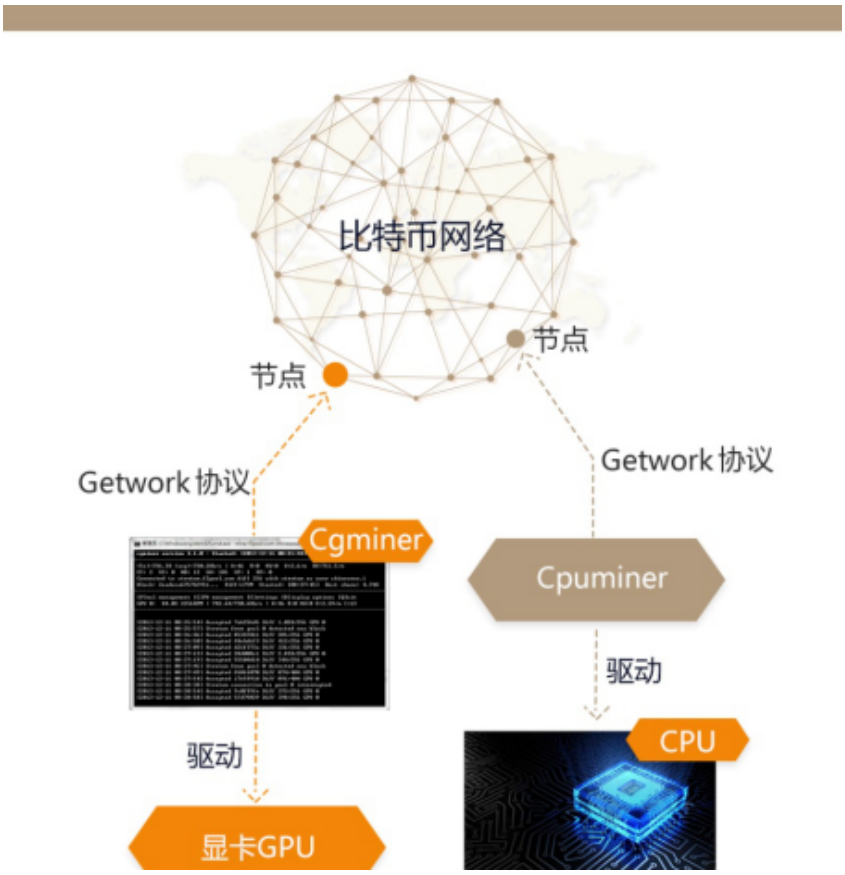
挖掘是算术运算的过程，从计算机和代码的角度来看，是反复执行哈希函数并检测执行结果的具体过程。和讨论算法一样，挖掘也是在采用POW共识机制的前提下讨论的。。很清楚，挖矿是从最开始的CPU挖矿到GPU挖矿，最后演变成现在的ASIC(专业矿机)挖矿时代。本文分析了逻辑设计和技术实现。挖掘的进化就是硬件和软件的进化。尤其是软硬件对接协议的改进过程，所以本文直接把与挖掘相关的几个核心协议作为小标题，一步步来讨论。在回顾这篇文章时，我发现这个词&quot矿工&quot是模糊的，这种情况在英国文学中也是类似的。在日常交流中，一般指拥有矿机的人。本文重点介绍区块链，采矿程序或机器统称为矿工。挖掘本节讨论挖掘的原理。首先，分析了比特币Blockheader的结构。我们说挖掘的本质是执行哈希函数的过程，哈希函数是单输入单输出函数，输入数据是块头。比特币块头有6个字段：int32_tnVersion//版本号，4字节uint256hashPrevBlock//前一个块的块头哈希值，32字节uint256hashMerkleRoot//此块中包含的所有事务结构的Merkle根。，32字节uint32_tnTime//Unix时间戳，4字节uint32_tnBits//记录这个块的难度，4字节uint32_tnNonce//随机数，如上4字节比特币每次挖矿都是连续对这80个字节进行SHA256运算(SHA256D)，运算结果固定为32个字节(二进制256位)。以上六个字段不同，nVersion，block版本号。，只会在升级时改变。HashPrevBlock，由前一个块确定。全网确定的NBits，每2016块重新调整一次，调整算法固定。所以以上三个字段可以理解为固定，对每个矿工都一样。。矿工可以自由调整的地方是剩下的三个字段，nNonce，它提供了nTime的2^32个可能值。其实这个字段提供的价值空间是非常有限的，因为合理的阻塞时间是有一个范围的，这个范围是根据之前的阻塞时间来确定的。如果过早或者比前一个块过早，就会被其他节点拒绝。值得一提的是，后一个区块的区块时间比前一个区块稍早，这是允许的。一般来说，矿工会直接使用机器的当前时间戳。hashMerkleRoot理论上有2256种可能性。该字段的更改来自于添加或删除块中包含的交易，或更改订单，或修改比特币基地交易的输入字段。根据哈希函数的特点，即使这三个字段中任何一位的变化，会导致Hash运行结果的巨大变化。在CPU挖矿时代，搜索空间主要由nNonce提供。矿机时代，nNonce提供的四个字节远远不够，搜索空间转向了hashMerkleRoot。

比特币挖矿的逻辑流程如下：1。打包事务，搜索要确认的事务内存池，选择块中包含的事务。矿工可以随意选择，甚至不选择(挖空块)，因为每个块都有容量限制(目前是1M)，所以矿工可以&#039；不要无限期地选择。。对于矿商来说，最合理的策略是先按照手续费对待确认的交易集进行排序，然后从高到低尽可能多的包含交易。2.在构造了比特币基地并确定了该块包含的交易集后，就可以统计出该块的总手续费并结合输出规则。矿工可以在这个区块计算自己的利润。3.为所有事务构造hashMerkleRoot和Merkle数。4.填写其他字段以获得完整的块标题。5.哈希操作：对块头执行SHA256D操作。。6.验证结果，如果符合难度，就全网播，挖下一

块；如果不符合难度，在哈希运算和验证之前，按照一定的策略改变上述字段中的一个。。合格块条件如下：SHA256D(Blockherder)F(nBits)，其中SHA256D(Blockherder)为挖掘结果，F(nBits)为难度对应的目标值，均为256位。全部作为大整数处理，直接比较大小来判断是否满足难度要求。为了节省区块链的存储空间，256位的目标值通过一定的变换无损压缩存储在32位的nBits字段中。具体的转换方法是拆分利用4个字节的nBits。，第一个字节代表向右移位的位数，用V1表示，后三个字节记录数值，用V3表示，所以有：另外，难度有最小限制，即有最大值，比特币的最小难度值为nBits=0x1d00ffff。对应的最大目标值为：0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000。例如，有256个硬币。给定数字1，2，3...256，每一次哈希运算就像扔一枚硬币，同时扔出256枚硬币。落地后，要求第一个数字N的硬币全部朝上。。SETGENERATESetgenerate协议的接口代表了CPU挖矿的时代。在他的论文中，中本聪描述了理想的数字民主概念&quot1CPU1投票&quot。在最初的版本中，挖掘功能是依附于客户端的，客户端上的挖掘非常简单。当然，需要同步数据后才能进行挖掘。现在很多算力不高的假币还是用客户端直接挖矿。开始挖掘有两种方式：1)在配置文件中设置gen=1，然后启动客户端，节点会自行开始挖掘。2)客户端启动后。，使用RPC接口setgenerate来控制挖掘。如果您使用经典的QT客户端，请单击&quot救命&quot菜单打开&quot调试窗口，在&quot控制台&quot，然后按enter，客户端将开始挖掘。后面的数字代表挖掘线程的数量。如果要关闭挖掘，在控制台中使用以下命令：setgeneratefalse，可以使用getmininginfo命令检查挖掘情况。节点挖掘的过程也很简单：结构块，初始化块头的各个字段，计算Hash，验证块，如果不合格，nNonce会自己增加，然后计算验证，以此类推。在CPU挖矿时代，nNonce提供的4字节搜索空间完全够用(4字节就是4G的可能性)，单核CPU运算SHA256D的计算能力一般在2M左右)，实际上，nNonce只遍历两个字节，然后返回重建块。GETWORKgetwork协议代表了GPU挖掘时代，需求主要来自于挖掘程序和节点客户端的分离。，区块链数据是从挖掘部分分离出来的。直接使用客户端节点进行挖掘需要同步完整的区块链，数据和程序紧密结合。也就是说，如果有多台计算机进行挖掘，每台计算机都需要单独同步一份区块链数据。这个其实没必要，对于矿工来说。至少需要一个完整的节点。同时，随着GPU挖掘时代的到来，也需要一种协议来与客户端节点进行交互。



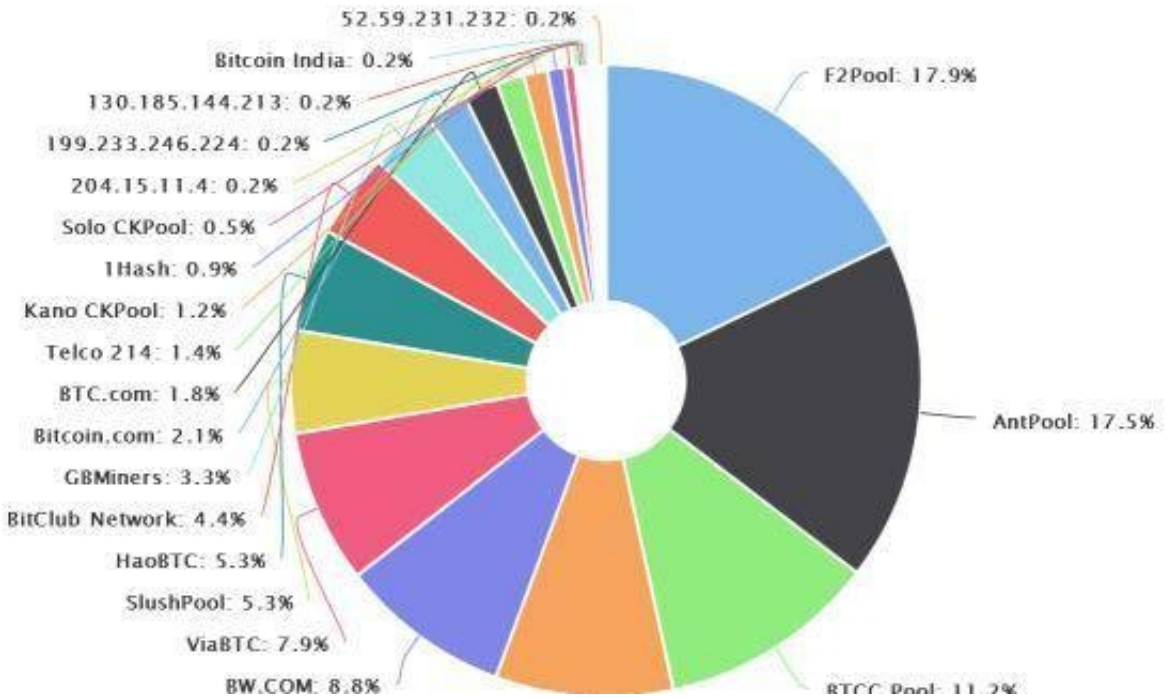
getwork的核心设计思想是：节点客户端构造块，然后将块头数据提交给外部挖掘程序，外部挖掘程序遍历nNonce进行挖掘，通过验证后回传到节点客户端，通过验证后广播到全网。如以前块头总共有80个字节。因为没有需要确认的区块链数据和事务池，所以总共72个字节的四个字段nVersion、hashPrevBlock、nBits和hashMerkleRoot必须由节点客户端提供。。挖掘程序主要以增量方式遍历nNonce，必要时可以对nTime字段进行微调。对于图形GPU来说，不用担心Nonce的4字节搜索空间不足，挖掘程序在从节点客户端获取一个数据后，也不要太长时间埋头工作。否则很有可能这块已经被别人挖了，继续挖下去也只会是徒劳。对于比特币来说，虽然它被设计为每10分钟一个块，但一个好的策略应该是在几秒钟内重新应用到新挖掘数据的节点上。对于显卡来说，运行SHA256D的计算能力一般在200M~1G到1G之间，nNonce提供4G搜索空间，也就是说再好的显卡也能支持4秒左右。如果调一次时间，可以再挖4秒，时间绰绰有余。节点提供RPC接口getwork该接口有一个可选参数。如果没有参数，则是申请挖掘数据。如果有一个参数，那就是提交挖掘块数据。调用不带参数的getwork,返回数据如下：{&quot中部州&quot:&quot9226a024e0b77f61d49FD5fdf828c6b5c4330c61ea2778c606A8e49d4ad8BD6&quot；&quot数据&quot:&quot0000002e9337BAC28ee28a949d2140f9FB0a0ab740acfd739d7BCF67ca31c2301db858ad2ca54d92c8C1cded71599字段共128字节(80区块头字节48补全字节)因为SHA256把输入数据分成固定长度的段，每段64字节，总输入长度必须是64字节的整数倍。如果输入长度一般不符合

要求，数据会按照一定的规则在元数据的末尾补全。事实上，对于采矿来说竣工数据是固定的，这里不需要提供。外部挖矿软件可以自行完成。甚至不需要提供nNonce字段，数据至少只需要提供前76个字节。nTime字段也是必需的。外部挖掘程序需要参考节点提供的块时间来调整nTime。目标字段是当前块的难度目标值，是小头字节顺序，需要翻转后才能使用。事实上，对于外部采矿计划，你可以用data和target两个字段正常挖掘，但是getwork协议充分考虑了各种情况，尽量帮助外部挖掘程序做力所能及的事情，额外提供了两个字段，data字段为这个思路返回完整的数据。。如上所述，SHA256对输入数据进行切片。矿工得到数据后，第一个切片(前64个字节)是固定的。中间状态是第一个切片的计算结果，节点帮助计算。因此在midstate字段的帮助下，外部挖掘程序甚至可以正常挖掘44字节的数据：midstate的第一个切片中剩余的12(76-64)字节数据中的32字节。。Hash1字段的比特币挖矿需要连续执行SHA256两次，第一次执行的结果是32字节，需要补充32字节的数据补足64字节作为SHA256第二次执行的输入。Hash1用同样的方法来完成数据。，hash1也是固定的。外部挖掘程序挖掘出符合条件的块后，再次调用getwork接口，将修改后的数据字段提交给节点客户端。节点客户端返回的数据也必须是128字节。。每次外部参数调用getwork，节点客户端都会构造一个新的块。在返回数据之前，新块应该完全存储在内存中，并将hashMerkleRoot用作唯一标识符。该节点使用一个映射来存储所有构造的块。当下一个区块已经被别人挖完，马上清空地图。getwork收到参数后，首先从参数中提取hashMerkleRoot，并在Map中找出之前保存的块。然后从参数中提取nnote和nTime，填入块的对应字段，块就可以验证了。如果难度符合要求，说明挖到了一个块，节点向全网广播。Getwork协议是最早的挖掘协议版本，实现了节点和挖掘的分离。，经典的GPU挖掘驱动cgminer和sgminer，还有cpuminer都是使用getwork协议进行挖掘。Getworkcgminer一直是非常经典的合作。当许多新算法被引入时，，都很快移植到了cgminer上。即使是现在，除了BTC和LTC，许多其他竞争对手仍在使用getwork协议进行挖掘。矿机出现后，挖矿速度大幅提升，目前比特币矿机的计算能力已经达到10T/s级别。。但getwork只为外部挖矿程序提供了32字节的4G搜索空间。如果继续使用getwork协议，矿机需要频繁调用RPC接口，显然是不可行的。现在BTC和LTC节点已经禁用了getwork协议。来更新更高效的getblocktemplate协议。getblocktemplategotblocktemplate协议诞生于2012年中期，当时矿石池已经出现。。矿池使用getblocktemplate协议与节点客户端交互，使用stratum协议与矿工交互，这是最典型的矿池构建模式。与getwork相比getblocktemplate协议最大的不同是getblocktemplate协议允许矿工自己构造块。这样，节点和挖掘就完全分离了。对于getwork来说，区块链是黑暗的。，getwork对区块链一无所知，他只知道修改4个字节的数据字段。对于getblocktemplate来说，整个区块链都是透明的，getblocktemplate拥有区块链所有与采矿相关的信息。，包括待确认的交易池。getblocktemplate可以自己选择要包含在块中的事务。Getblocktemplate在开发出来之后就不是静态的了，在后续版本的客户端中已经进行了升级和更改。，主要是增加一些字段，但核心概念和核心字段不变。目前比特币客户端返回的数据如下。考

虑到空间限制，在事务字段中只保留一个事务数据。事实上，根据目前的实际情况，交易池中有上万笔交易需要实时确认。目前区块基本满了(1M容量限制)，再加上附加信息。所以每次调用getblocktemplate基本都是返回1.5M左右的数据，而getwork是几百字节。，不可同日而语。

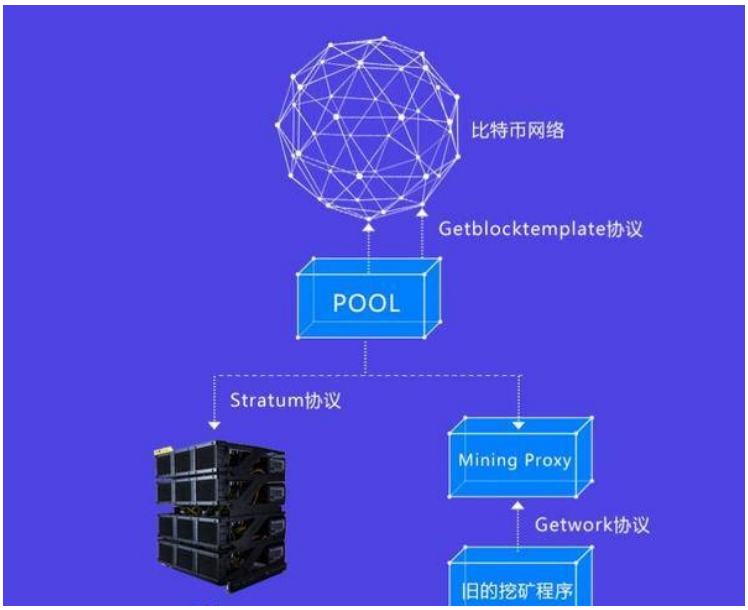
让&#039；s简单分析几个核心字段，即Version、PreviousblockHash和Bits，指的是块版本号和之前的块Hash，以及难度。矿工可以直接在块头的相应字段中填入数值。。Transactions，transactionset，不仅给出了每笔交易的十六进制数据，还给出hash，交易费等信息。Coinbaseaux，如果有任何你想写入区块链的信息，把它放在这个字段中，类似于中本聪&#039；s创世纪区块声明。。Coinbasevalue，挖掘下一个区块的最大利润值，包括发行新币和交易费用。如果挖掘器包含事务字段中的所有事务，则可以直接使用该值作为比特币基地输出。目标，格挡难度目标值。MinTime是指下一个块的最小时间戳，Curtime是指当前时间，作为矿工调整nTime字段的参考。高度，下一个街区难度目前协议规定这个值要写到coinbase的指定位置。矿工得到这些数据后，挖掘步骤如下：1。构建coinbase事务，涉及字段包括Coinbaseaux和Coinbasevalue。、交易、身高等。当然，最重要的是指定一个收入地址。2.构建一个hashMerkleRoot，并将coinbase放在transactions字段中包含的事务列表之前。，然后对相邻的事务进行SHA256D运算，最终可以构造出事务的Merkle树。因为coinbase有很多字节供矿工随意发挥，另外，交易列表可以随意更改或增删。因此，可以认为hashMerkleRoot值空间几乎是无限的。实际上，getblocktemplate协议设计的主要目标就是让矿工获得这个巨大的搜索空间。3.构建块头并使用版本。、Previousblockhash、Bits和Curtime分别填充块头的对应字段，nNonce字段可以默认设置为0。4.采矿，矿工可以随时进来。在hashMerkleRoot提供的搜索空间中设计自己的挖掘策略。5.交资料。当矿工挖掘一个区块时，他们会立即使用submitblock接口将该区块的完整数据提交给节点客户端，节点客户端将对其进行验证和广播。。需要注意的是，就像上面提到的GPU使用getwork进行挖掘一样，虽然getblocktemplate为矿工提供了巨大的搜索空间，但是矿工也不应该挖掘一个请求数据太久。相反，我们应该定期向节点请求最新的块和最新的事务信息，以提高挖掘收益。池中开采有两种方式，一种叫单人开采，一种叫矿池开采。如上所述，直接在节点客户端启动CPU挖掘。，以及依靠getworkcgminer驱动显卡直接连接节点客户端进行挖掘，都是单飞挖掘。单飞就像自己买彩票一样，不容易中奖。如果你获奖，所有的收益都将属于你。挖矿池就像一起

买彩票，大家一起买单。可以买一堆彩票，中奖后按照投资比例分配收益。理论上矿机可以借助getblocktemplate协议将节点客户端SOLO链接到矿上，但实际上没有哪个矿机会长期这么做。写这篇文章的时候，比特币的计算能力是1600PP。目前最先进的矿机计算能力在10T左右，所以单个矿机单挑挖块的概率不到16万分之一。矿工(人)会投入真金白银买矿机，交电费，不会做这么冒险的投资。显然，他们会投资矿池来降低风险。比较适合获得稳定的收入。因此，矿池的出现是不可避免的，是无法消除的，无论它是否破坏了制度的分权原则。矿池的核心工作是给矿工分配任务，统计工作量，分配收入。矿池把方块难度分成很多难度较小的任务，发给矿工计算。矿工在完成一项任务后，将工作量提交给矿池，称为提交份额。如果全网分块难度要求哈希运算结果的前70位全为0，那么矿池分配给矿工的任务可能只要求前30位为0(根据矿工调整&#039；计算能力)。矿工完成规定难度的任务后，上交分成，在30强为0的基础上，重新测试矿坑，看70强是否全部为0。矿池会根据每个矿工的计算能力分配不同难度的任务。矿池如何评判矿工&#039；计算能力分配适当的任务难度？调整思路和比特币区块难度一样。矿井需要矿工的帮助&#039；分享率，矿池希望分配给每个矿工的任务足够矿工计算一定时间，比如1秒。如果矿工一秒完成几个任务，说明矿池给的难度低，需要提高，反之亦然。这样经过一段时间的调整，矿池就可以给矿工分配合理的难度，计算出矿工的计算力。

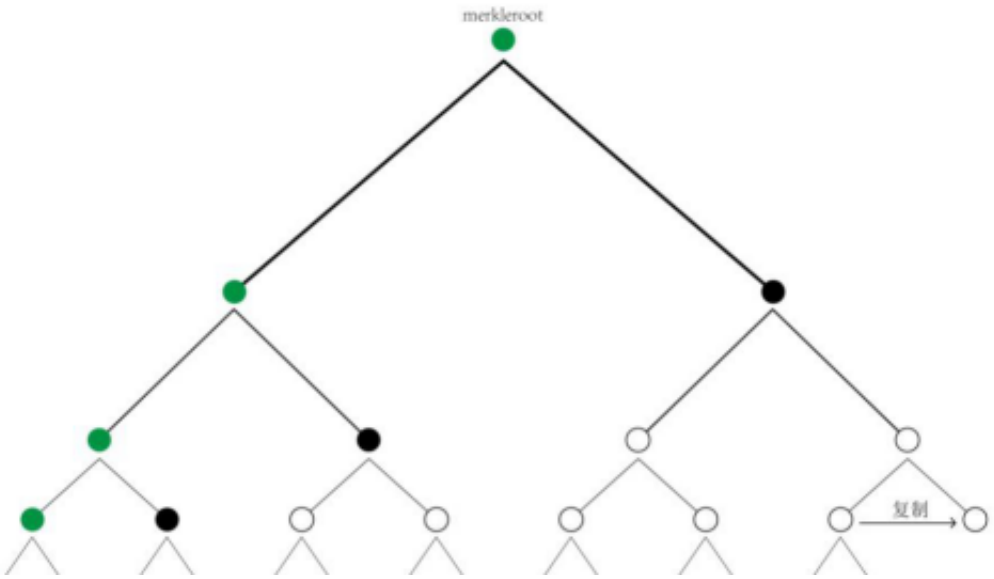


矿池一直是个矛盾体。毫无疑问，矿池是集中的。如上图所示，全网计算能力集中在几个矿池手中。虽然网络中成千上万的节点同时在线，但只有点击几次的矿池链接有投票权，其他节点只能行使监督权。。矿坑再次把矿工们置于&quot黑暗&quot，矿工们又一次变得对区块链一无所知。他们只知道完成矿池分配的任务。还有一集是关于矿池的。矿池刚出现的时候，反对意见特别强烈。很多人悲观地认为矿

池最终会导致计算能力的集中。，危害系统安全，甚至扼杀比特币。于是有人设计实现了一个P2P矿池，试图去中心化&quot抱着一群人去挖矿&quot，而且代码也是开源的。但由于效率远不及集中式矿池，未能吸引太多计算能力，所以所谓理想很丰满，现实很骨感。。推荐几个成熟的开源矿池项目，有兴趣的读者可以自己研究一下：uPHP-MPOS，早期非常经典的矿池，非常稳定，使用最多，尤其是山寨币矿池，后端使用StratumMing协议。,sourceaddresshttps://github.com/MPOS/php-mposunode-openpit-mining-portal,supportingmulti-currencymining.,thesourceaddressishttps://github.com/zone117x/node-open-mining-portaluelectricpool,whichsupportsmixedmining.在https://github.com/sigwo/powerpool.运行一个矿池需要考虑的问题有很多比如为了得到全网最及时的信息，矿池一般连接几个网络节点，最好分布在地球的几个大洲。。此外，提高块率、降低孤块率、降低空块率是矿池的核心技术问题。这张纸可以&#039；t逐一讨论，然后只详细讨论一个问题，即矿池和矿工的具体工作模式，——stratum协议。。stratum矿池通过getblocktemplate协议与网络节点交互获取区块链最新信息，通过STRATUM协议与矿工交互。此外，为了使以前用getwork协议进行挖掘的软件也能连接到矿池进行挖掘。一般矿池也支持getwork协议，通过分层挖掘代理机制实现。需要说明的是，矿池刚出现的时候，显卡挖矿还是主力，getwork用起来非常方便。另外，早期的FPGA矿机也有一部分是用getwork实现的。stratum以TCP方式与矿池通信，数据以JSON格式封装。



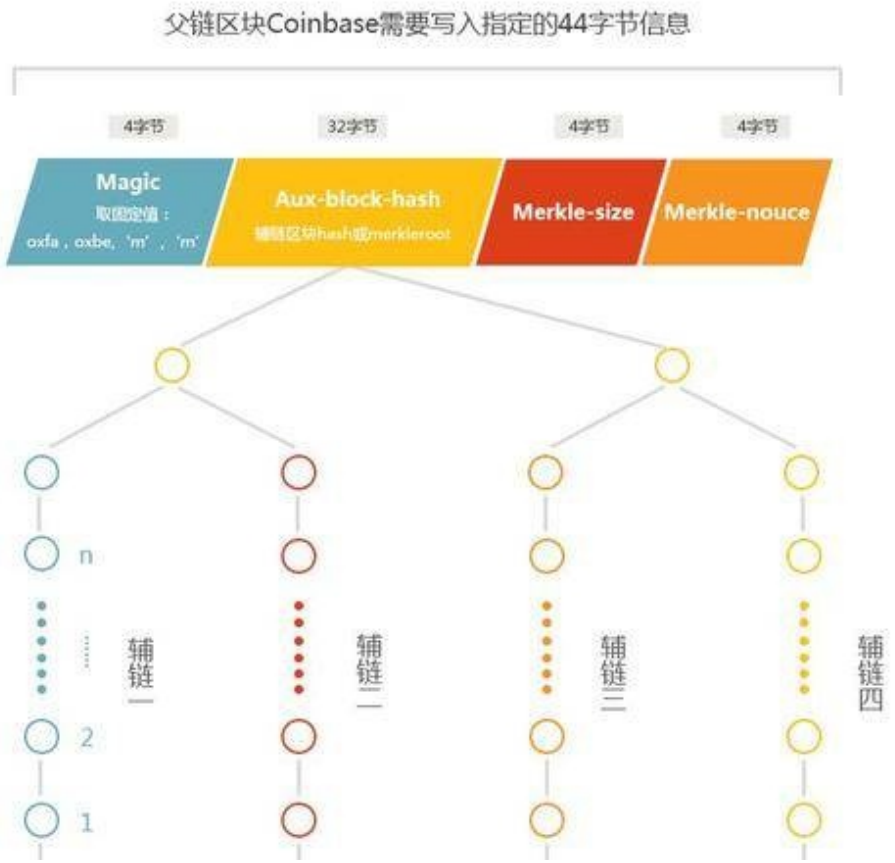
让&#039；让我们谈谈getblocktemplate:Miners&#039；驱动：在getblocktemplate协议中，矿工仍然通过HTTP主动调用RPC接口，向节点申请挖掘数据，也就是说，矿工可以&#039；不能及时获知网络最新区块的变化，导致计算能力丧失。数据加载：如上所述，正常调用getblocktemplate节点会反馈1.5M左右的数据，其中主要数据是事务列表。矿工和矿池需要经常交换数据，显然不可能每次分配工作都给矿工附加这么多信息。而且巨大的内存需求会极大的影响矿机的性能，增加成本。Stratum协议彻底解决了上述问题。Stratum协议采用主动分配任务的方式。也就是说，矿池可以随时给矿工分配新任务。对于矿工来说，如果接到矿池分配的新任务，要立即无条件转向新任务；矿工也可以主动用矿池申请新任务。现在的核心问题是如何让矿工获得更大的搜索空间。根据getwork协议，如果只有矿工可以更改Nonce和nTime字段，那么交互数据量很小，但是搜索空间肯定不够。如果想增加搜索空间，只能在hashMerkleroot上下功夫。如果允许矿工自己构造coinbase，搜索空间的问题就解决了，但代价是需要把块中包含的所有事务交给矿工，他们才能构造事务列表的Merkleroot，这对矿工来说压力更大。对矿池带宽的要求也更高。Stratum协议巧妙地解决了这个问题，成功的实现不仅可以为矿工增加足够的搜索空间，而且需要的交互数据量很小，这也是Stratum协议最具创新性的地方。



让&#039；让我们回顾一下块头的六个字段，它们都是80个字节。这一点非常重要。nVersion、nBits和hashPrevBlock三个字段是固定的，Nonce和nTime两个字段现在矿工可以改了。。增加搜索空间只能通过hashMerkleroot完成，它可以&#039；不要被绕过。Stratum协议允许矿工构造自己的coinbase事务，coinbase的scriptSig字段中有很多字节是矿工可以自由填充的。而coinbase的变化意味着hashMerkleroot的变化。不需要所有事务都从coinbase构造一个hashMerkleroot。如上图所示，如果块将包含13个事务，矿池将首先处理这13个事务。最后，只要把图中的四个黑点(哈希值)传递给挖掘者，把构造coinbase需要的信息传递给挖掘者，挖掘者就可以构造自己的hashMerkleroot(图中的绿点都是挖掘者自己计算出来的，当哈希成对合并时，规定下一个黑点代表的哈希值总是放在右边)。这样，如果块包含N个事务，则矿池可以被浓缩成log2(N)个哈希值并交付给矿工，这大大减少了矿池和矿工之间的数据交互量。。Stratum协议严格规定了矿工和矿池之间的接口数据结构和交互逻辑，具体如下：1.矿工订阅任务启动矿机。,使用mining.subscribe方法链接矿池{&quotid&quot:1，&quot方法&quot:&quotmining.subscribe&quot，&quot参数&quot:[]}n//申请链接{&quotid&quot:1，&quot结果&quot:[[['采矿。set_difference&#039,'B4b6693b72a50c7116db18d6497CAC52&#039；],['mining.notify&#039,'AE6812EB4CD7735a302a8a9DD95cf71f&#039；]，&quot080000002&quot，4，&quot错误&quot:null}n//返回数据返回数据很重要，矿工需要本地记录，在整个采矿过程中都要用到，其中：ub4b6693b72A50C7116db18d6497CAC52:为矿工指定初始难度。,UAE6812EB4CD7735A302A8A9DD95CF71F:subscriptionnumberIDu08000002:scientificnamevery1，用于构造coinbasetransactionu4:学名Extranonce2_size，即Extranonce2的长度，其中指定了四个字节的Extranonce1和Extranonce2，对于挖掘非常重要。增加的搜索空间在这里。现在，我们至少有8个字节的搜索空间，即4个字节的nNonce和4个字节的Extranonce2。2.矿池授权在矿池注册账号，添加矿工。矿池允许任何

数量的矿工被添加到每个帐户，并采取不同的名称来区分他们。矿工使用mining.authorize方法申请授权，只有矿池授权的矿工才能接收矿池分配的任务。。{&quot参数&quot:['slush.miner1&#039,'密码&#039；]，&quotid&quot:2，&quot方法&quot:&quot采矿。授权&quot}n{&quot错误&quot:null，&quotid&quot:2，&quot结果&quot:true}n3。矿池分配任务{&quot参数&quot:['bf&#039,'4d16b6f85af6e2198f44AE2a6de67f78487AE5611b77c6c0440b921e000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000以上每个字段信息都是必不可少，其中：ubf:任务号ID，每个任务都有唯一标识符u4d16b6f85af6e2198f44AE2a6de67f78487AE5611b7c6c0440b921e000000:前一个块的哈希值。，hashprevblocku01000000000000000000000000000000000000000000000000000000ffffffffffffffff20020862062f503253482f04b8864e5008:学名coinb1，序列数据第一部分u072f736c7573682f000000100f2052a010000001976a914d23fcdf86f756a7a9688ef9903327048ed988AC0000000:学名coinb2。,构造coinbase的第二部分序列数据，u[&#039；76cffd68BBA7ea661512b68EC6414438191b08aaeaec23608de26AC87820CBD02016&#039；'e5a796c0b88Fe695949a3e7b0b7b1948a327B2f28C5dbE8f36f0a18f96B2ffef2016&#039；]:学名梅克尔_布兰奇，事务列表的压缩表示，也就是上图中的黑点u0000002:block版本号，nVersionu1c2ac4af:block难度nBitsu504e86b9:当前时间戳，nTime有以上信息。，加上之前得到的Extranonce1和Extranonce2_size，就可以挖掘了。。4.挖掘1)用于构造coinbase事务的信息包括CoinB1、Extranonce1、Extranonce2_size和Coinb2。结构很简单：coinbase=coinB1extranonce1extranonce2coinB2为什么会这样？因为矿池已经为矿工们做了大量的工作，而且矿池已经建成比特币基地交易。序列化后分为coinb1和coinb2，包含指定的信息，比如coinb1包含块高，coinb2包含矿工&#039；收入地址和收入金额。但这些信息与矿工无关，矿工挖掘的地方只有4个字节的Extranonce2。另外，Extranonce1是矿池写入块的指定信息。一般来说，每个矿池都会写自己的信息。比如矿池的名称或者域名，我们根据这些信息统计全网每个矿池的计算能力比例。2)使用coinbase和merkle_branch构建Merkleroot。，如上所示构造hashMerkleroot字段。3)构建一个块头来填充剩余的五个字段。现在，如果搜索空间不够，可以随时搜索矿池进行开采。通过将Extranonce2_size增加几个字节，可以很容易地解决这个问题。5.矿工提交的工作量矿工找到符合难度的a股，就提交给矿池，提交的信息量很小。,都是必不可少的字段：{&quot参数&quot:['slush.miner1&#039,'bf&#039,'00000001''504e86ed&#039,'b2957c02&#039]，&quotid&quot:4，&quot方法&quot:&quot采矿。提交&quot}{&quot错误&quot:null，&quotid&quot:4，&quot结果&quot:true}slush.miner1:矿工名字矿池用于识别谁提交了工作量bf:任务编号ID，并且矿池在分配任务之前构建了比特币基地和其他信息。，用这个任务号唯一标识00000001:extranonce2504e86ed:ntime字段b2957c02:nNonce字段矿池得到以上五个字段后，首先根据任务ID，找出分配任务前存储的信息(主要是构造的coinbase事务和事务列表等。)，然

后重构块，再验证份额的难度，再检查满足难度要求的份额是否满足全网难度。。6.矿池为矿工调整难度。矿池记录每个矿工的难度，并根据份额率不断调整，以指定适当的难度。矿池可以通过mining.set_difference方法随时向矿工发送消息，改变难度。。{&quotid&quot:null，&quot方法&quot:&quot采矿。set_difference&quot，&quot参数&quot:[2]}如上，地层协议核心理念基本解析清楚在getblocktemplate协议和Stratum协议的配合下，矿池终于可以大声对矿工们说，让计算力来得更猛烈些吧。AUXPOW在矿业发展史上，也出现过一件令人浮想联翩的事情。，即合并挖掘。混合挖掘模式首先用于域名coin，它与比特币链相关联。矿工挖比特币的时候，可以同时挖域名币。后来Dogecoin也支持混采。，锚定在莱特币链上。混合挖掘通过辅助工作证明(AuxPOW)协议实现。虽然混合挖掘不是很流行，但是协议设计得很好。当我第一次看到协议时，我不能&#039；我不禁感叹这个社区的伟大。这个我能想到。以域名币的混合挖掘为例。比特币是母区块链，域名币是辅区块链。AuxPOW协议的实现不需要改变母链(比特币当然不会对域名货币做任何改变)，但是需要专门设计辅助链。比如dogecoin改支持混合挖掘，就做了硬分叉。



auxpow的实现得益于比特币比特币基地的输入领域。中本聪没有&#039；不知是有意还是无意，这里只规定了长度限制，留下了一个未定义的区域。这个领域后来对比特币的发展产生了深远的影响，很多升级和优化都集中在这个领域。，比如上面提到的Stratum协议。类似的有意无意的情况在中本聪还有很多，比如交易的nS

equence字段。因为没有明确的定义，&quot扩展性&quot因被黑客盯上而引发的问题，成了门头沟垮塌的替罪羊。。另一个例子是nNonce。如果一开始定义的字节更大，比如说32字节，挖掘的发展就不需要像上面讨论的那么多协议。AuxPOW协议的核心思想是不同的：对于经典的POW块，规定只有难度满足要求才能算一个合格的区块。AuxPOW协议并没有要求难度块，但是附加了两个条件：1。辅助链块的哈希值必须构建到父链块的比特币基地中。。2.母链块的难度必须满足二级链的难度要求。辅助链块的哈希值内置在父链的比特币基地中，实际上是利用父链来证明存在。这样就可以间接依靠母链的计算能力来维护二级链的安全。一般而言母链的计算能力大于二级链，所以满足母链难度要求的块也必须满足二级链的难度要求，反之亦然。这样，很多在母链中无法达到难度要求的方块，在二级链中就达到了难度要求。矿工g=广播到二级链网络，在二级链获得利益。，有何不可。AuxPOW协议对两个链都有一些数据结构规定。对于父链，需要在块的coinbase的scriptSig字段中插入44字节的数据，格式如下：

字段	大小(字节)	类型	作用
magic	4	char [4]	固定依次取值 0xfa，0xbe，'m'，'m' ，作为合并挖矿的标识，告诉辅链从这之后的 44 字节内容专用于合并挖矿
aux_block_hash	32	char [32]	当只挖一条辅链时，该值表示辅链区块 hash 值；同时挖多条辅链，该值为所有辅链的在挖区块 hash 构建的 Merklerroot
merkle_size	4	int32_t	同时挖辅链的数量，只有一条辅链时设置为 1

对于辅链，原来的块结构有了很大的改变，在nNonce字段和txn_count之间插入了5个字段。这个区块被命名为AuxPOW区块。

字段	大小 (字节)	类型	作用
version	4	uint32_t	
prev_block	32	char[32]	
merkle_root	32	char[32]	
timestamp	4	uint32_t	
bits	4	uint32_t	
nonce	4	uint32_t	
coinbase_txn	?	txn	父链区块的 Coinbase 交易，连接两条链的枢纽
block_hash	32	char[32]	父链区块（头）的 hash 值
coinbase_branch	?	Merkle branch	父链 Coinbase 所在 Merkleroot 的分支
blockchain_branch	?	Merkle branch	辅链区块 hash 所在 Merkleroot 的分支
parent_block	80	Block header	父链区块头

混合挖掘要求母链和辅链的算法一致。是否支持混合开采是矿池的决定，而矿工不&#039；我不知道他们是否在混合采矿。如果矿池支持混采，则需要对接辅助链的所有节点。次级链块的散列值被构建到父链的比特币基地中。，这意味着在构造父链比特币基地之前，挖掘者必须首先构造辅助链的AuxPOW块，并计算哈希值。如果只挖一个辅助链，情况就比较简单了；如果同时挖掘多个辅助链，首先为所有的辅助链挖掘块构造Merkleroot。。矿池可以将特定的44字节信息放入上面Stratum协议中提到的Coinb1中，交给矿工进行开采。为矿工返回的份额重建父链块和所有辅助链块，并检测难度。如果符合辅助链难度要求，整个AuxPOW块被广播到次级链。辅助链节点验证AuxPOW块逻辑的过程如下：1.取决于父链块头(parent_block)和块哈希值(block_hash)，这个字段实际上是不必要的。，因为节点可以自己计算)，并验证父链的块头是否满足二级链的难度要求。。2.依靠比特币基地事务(coinbase_txn)，它的分支(coinbase_branch)和父链块头(parent_block)。验证比特

币基地事务是否真的包含在父链块中。3.依靠二级链分支(区块链_分支)和比特币基地(aux_block_hash)放哈希值的地方。验证次级链块的散列是否内置于父链块的比特币基地事务中。经过以上三点验证，认为是合格的辅助手拉葫芦。中本聪最初设计比特币的时候，希望所有节点都用CPU挖矿。一般认为，只有这样才能充分保证区块链的去中心化特征，比特币在CPU时代也安全度过了萌芽阶段。Getwork和cgminer将挖矿带入GPU时代，国产显卡一度缺货，全网计算能力迅速提升到一个更高的层次。，CPU挖矿被淘汰。随着参与挖矿的人越来越多，整个网络的计算能力在上升，催生了分组挖矿(矿池)。然而，GPU时代的繁荣历史并没有&#039；getblocktemplate之前不会持续太久。地层和矿机已经进入ASIC时代。Getwork实现了数据和挖掘的分离，getblocktemplate为外部挖掘程序提供了最大的自由度。，完全解决了外部挖矿程序与节点交互的可扩展性问题，主要用于对接矿池和网络节点。Stratum不仅解决了搜索空间不足的问题同时，也解决了矿池与矿机之间数据交互量大的问题。getblocktemplate和stratum这两个协议使大型矿池、大型矿和大型矿机成为可能，矿业从此进入了一个全新的阶段。此后，采矿的演变主要集中在几个方向：矿池的设计优化和稳定运行，矿井的科学部署，以及提升矿机技术以提高计算能力和降低功耗。作者：周币科技副总裁、科技专家、区块链协会讲师。