**出块不均匀的问题**

**1、问题描述：**

**按程序的预计，约为1分钟出一个块，高度增加1，一天高度增加为1440左右，目前的现象是有时候10几秒就出一个，有时候是几分钟才出一个，一天出的块数约为1330左右,最少的只有900多个。**

**(1)、人气的区块浏览器看到的数据：**

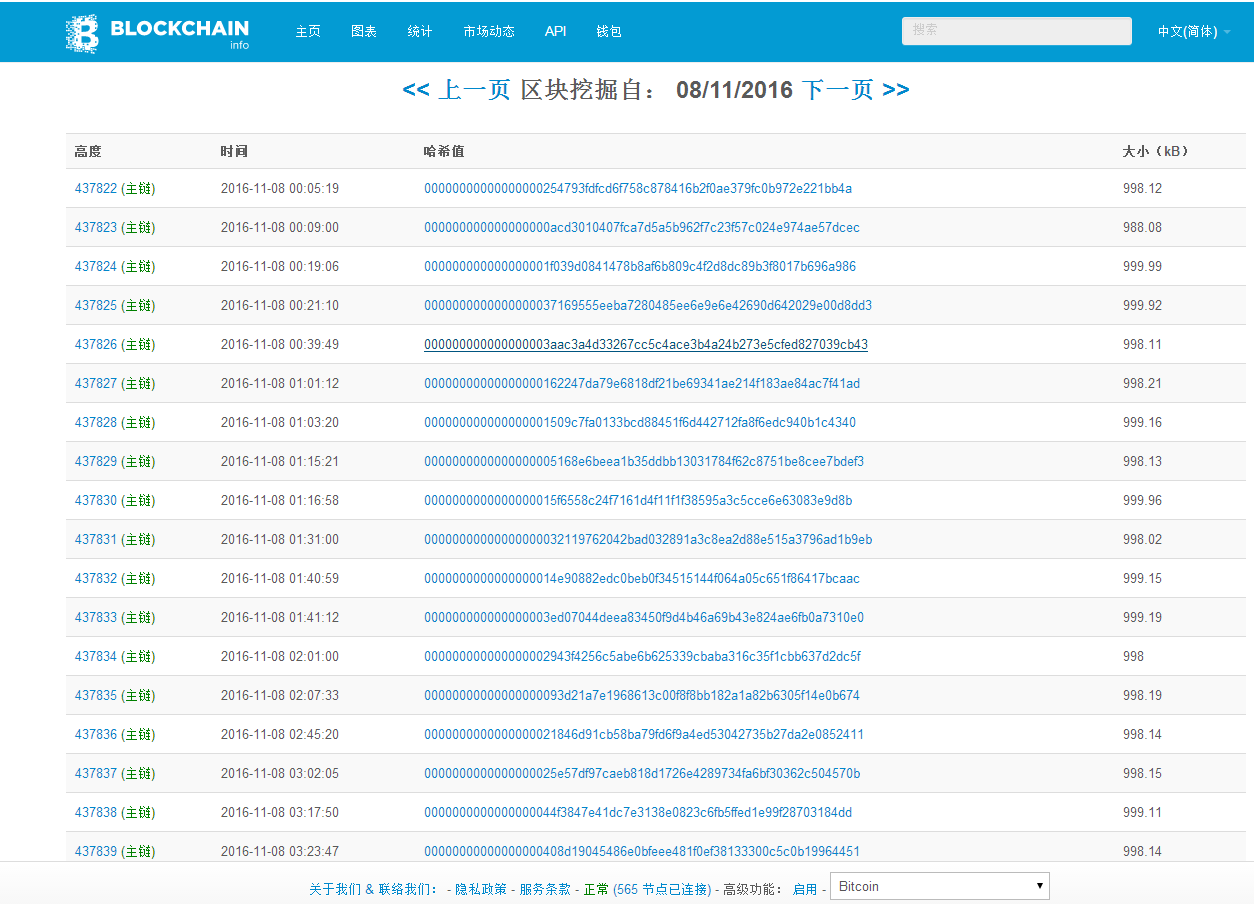
****

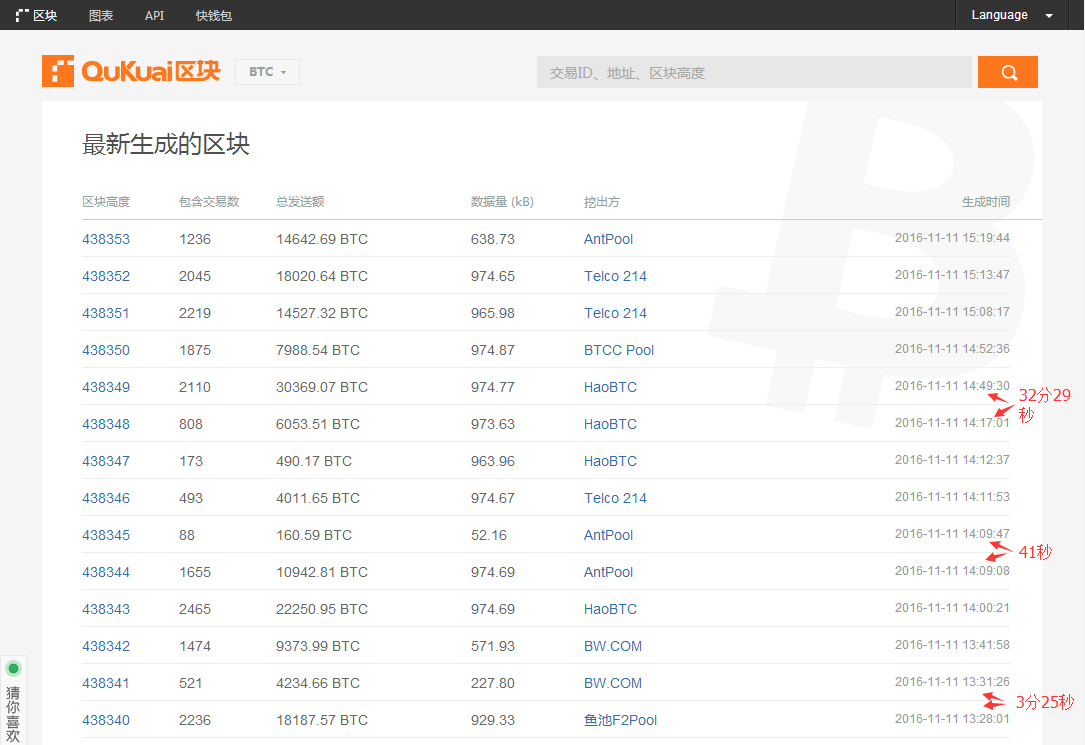
**(2)、比特币的区块：**

**比特币的目标是调到约为10分钟出一个区块，一天约出144个区块。**

**做一小部分统计：**

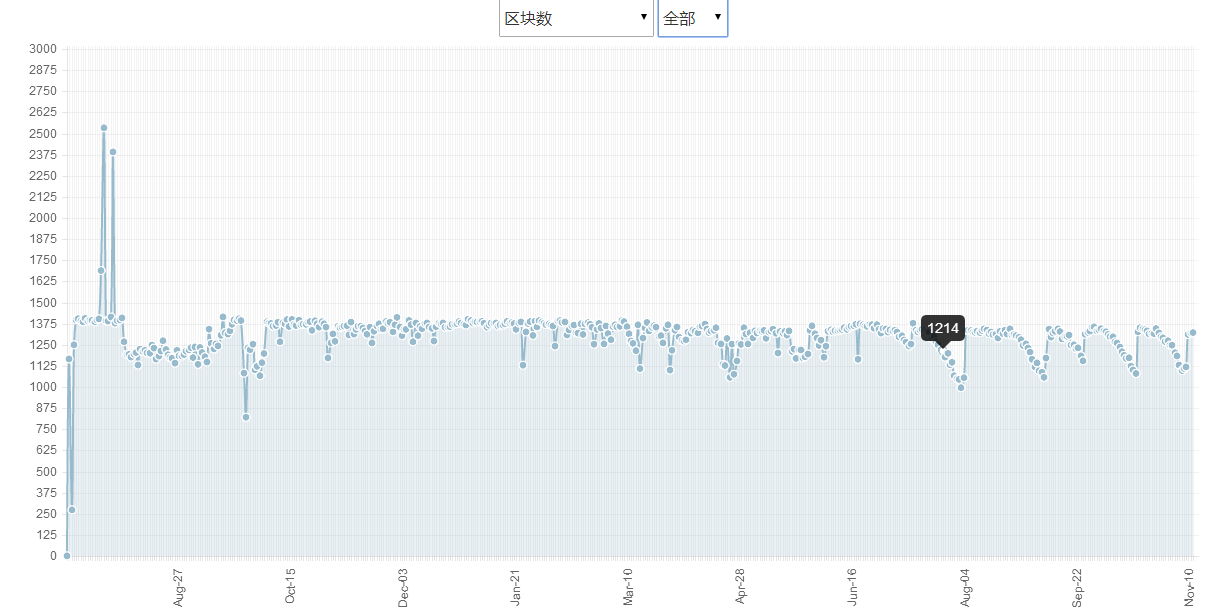
|  |  |  |
| --- | --- | --- |
| **时间** | **出块数** | **与144差值** |
| **2016-11-08** | **168** | **24** |
| **2016-11-09** | **146** | **2** |
| **2016-11-10** | **167** | **23** |
| **2016-11-11** | **162** | **18** |
| **2016-11-12** | **169** | **25** |
| **2016-11-13** | **141** | **-3** |

****

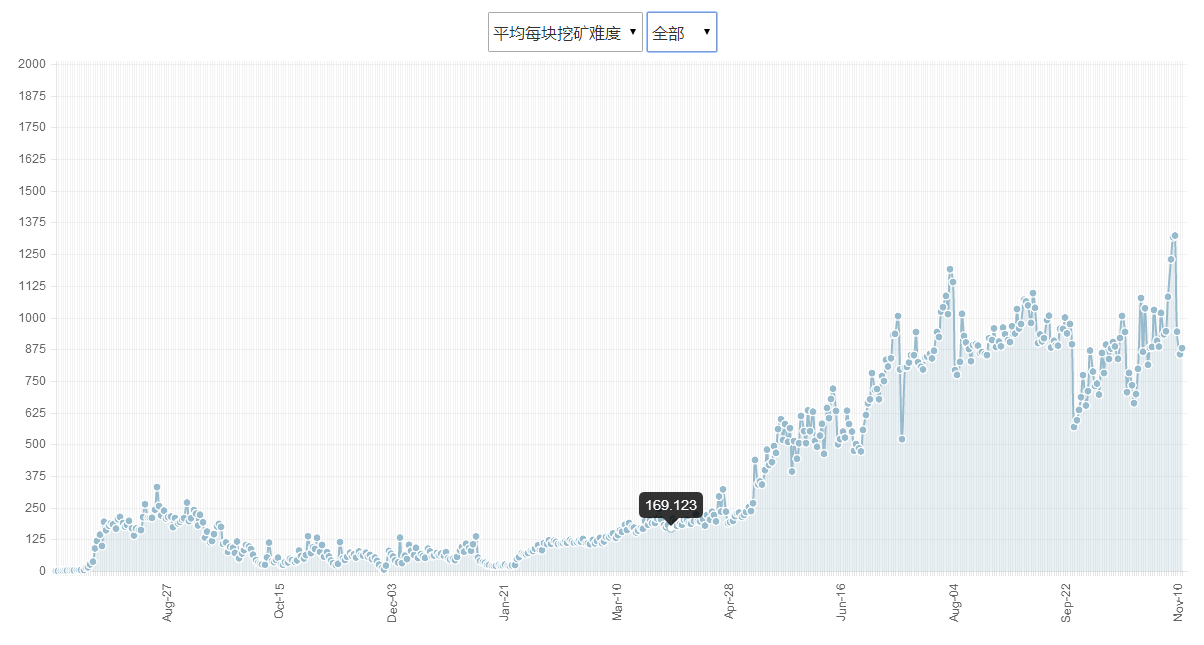
****

**(3)、智能坊的一些统计图表：**

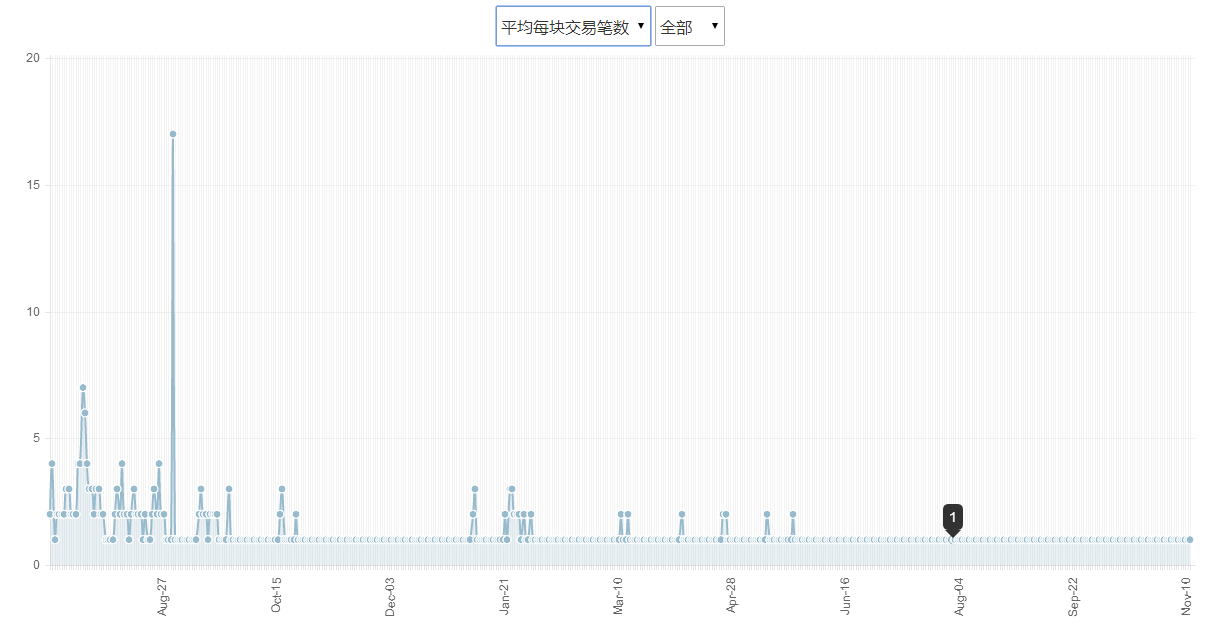
**图3-1：区块数：**

****

**图3-2：平均挖矿难度：**

****

**图3-3：每块的交易笔数：**

****

**2、影响BLOCK增加的因素：**

**从以上的图表可以看出，比特币的出块时间也只是大体上的均匀，有时差别比例也大。智能坊的统计图表上可以看到，大部分的点是在1375左右浮动。从图3-3可以看出，后面几乎没有其它的交易，都只有挖矿打包的交易。图3-1显示，后段的出块成了锯齿状。在只有挖矿的情况下，出现出块不均的情况。因为智能坊的POS方式，币龄对出块难度也有影响.**

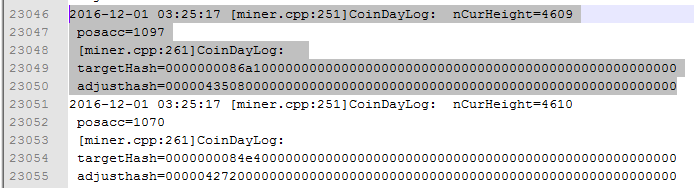
****

出块时间

出块难度

币龄

**打印出其中一条信息：**

****

**高度4609时，币龄1097的矿工打包了这个区块。**

**在没有用币龄调整过的时候，**

**目标HASH：**

**targetHash=0000000086a10000000000000000000000000000000000000000000000000000**

**经过币龄调整后的HASH：**

**adjusthash=0000043508000000000000000000000000000000000000000000000000000000**

**可见，经过加入币龄的运算后，目标HASH值变大了，难度就变小了很多，出块时间相对要快。**

**3、一些分析**

**因为出块是根据找到的HASH值小于目标HASH值来算的，所以，理论上来说，HASH值越大，难度越小，越容易找到符合条件的HASH值。但是因为这个HASH值是随机的，存在不确定性，有可能存在的情况是找到难度小的块的时间比找到难度大的块的时间要长。因为没有人能直接控制计算的结果，只能通过难度进行间接的粗调。所以，如果这个要保证准确的一天出1440块的话，应该是做不到的。最好的情况是只能接近这个目标值。通过难度自适应的调整一下。**

**调整的策略也是一个方面，就是取一段区块数，统计它们出块的时间间隔，看是否能保证在1分钟左右出一个。然后根据这个偏差值，或正或负的调整出块的难度。**

**4、一些末经验证的想法：**

**（1）增加调整难度的取样区间，如直接按日期进行计算，看看当天0点到当前时间出了多少块，然后按这个时间区间内，目标是出了多少块，用这个偏差值作为因子进行下一个难度运算处理。**

**（2）用滑差时间进行取样，在达到规定的区间块个数后，每增加一个块时，运算一次偏差值，作为因子进行下一个难度运算处理。**

**5、下面是关于难度的一些介绍：**

区块哈希值的计算结果是一个随机数，没有人能直接控制计算的结果。  
  
打个比方，重复N次（N趋近于无穷大）掷一个六面的骰子，每掷一次就可以掷出一个6（包含6）以下的数字，但是如果要想掷出3以下的数字，取得每个结果时平均雪要掷2次才行；也就是说，掷出“3以下（含3）”比掷出“6以下（含6）”要难一倍，需要多做一倍的“工作量”。  
  
如果定义“6以下”这个规则的难度系数（困难度）为“1”，那么，“3以下”这个规则的难度系数为“2”，意味者要想符合规则要求，需要2倍的工作量；“1以下“的难度系数则为6，意味着该规则需要6倍的工作量。  
  
比特币协议中，规定一个256位的整数0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF为难度“1”，在当时的全网算力下大约需要10分钟左右的哈希计算工作量才可以满足这一规则（小于等于这个数）。  
  
(目标值（TARGET） = 0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 时，难度值为1）  
  
如果想确保10分钟这个工作时间恒定，那么当算力提高n倍时，需要把难度值也调高n倍。目前（2015年10月）Bitcoin网络的难度值为608亿，可以推算出现在的全网算力比2009年初增大了608亿倍。  
  
难度值的计算实际上就是简单的除法。只不过涉及到的整数太大（远远超过内置的整数位数），没法直接用普通数学库的除法来运算，需要借助高精度的算法库（比如GMP）来完成，这就比较麻烦。所以bitcoin wiki中给出了一种借助泰勒级数变体实现的快速对数算法计算难度值的示例。（笔者吐槽：这TMD更麻烦）  
  
区块存储时，block\_header中有一个字段是与难度值相对应的，即bits。用一个32位的整数来压缩存储当期256位的目标值（TARGET）。压缩规则很简单，bits字段的最高位字节为指数（little endian中的第4个字节），存储目标值有效的字节数，如果目标值的最高位为1（大于0x80)，需要在前面补上一个0x00。（比如难度值1的目标值有效位为0x00FFFF......，补0后共0x1D个字节)。另外3个字节为尾数（mentissa)，截取目标值最高3个字节来存储。  
  
这样，难度值1对应的目标值写成bits格式，就是0x1D00FFFF。  
以此类推，目标值为 0x00000000000404CB000000000000000000000000000000000000000000000000时，bits的格式为0x1B0404CB。  
  
计算难度值时其实可以采用简易算法：尾数部分相除，再乘以指数部分带来的差异即可。（2 ^ [指数部分相减的结果 \* 8])，计算速度比“快速对数法”要快很多（但是如果超出double型所容许的位数会有溢出）。  
  
现有的算法中，难度值每2016个区块调整一次，但新的难度值不需要与难度“1”进行比较运算，而是根据前2015个块的出块时间来计算。（问：为什么是2015？答：因为最初程序员的代码写错了，纠正这个错误会导致分叉，所以只能将就一下了）。  
difficulty = [prev\_target] \* 【前2015个区块生成所用的时间】 /  1209600 （按标准每10分钟出一个块，2016个块所需要的秒数）  
  
因为算法确定，所以分布计算时也能保证所有节点计算出的难度值都一致，不会出现分歧。区块链同步时，节点或客户端会优先选取累积难度最大的链作为主链，仅有高度（height)的“伪造慢链”很难被认可，除法伪造出的难度值也很高。但是只要伪造了一个块的难度，后续所有的区块必须用相应的难度重新计算一遍，这往往需要攻击者拥有的算力与当前全网的真实算力相当才有可能完成。  
  
比特币发展的早期，“伪造慢链”攻击应该是很难防的，但是一旦算力达到一定规模，在PoW下这种攻击需要极大的成本和海量的计算时间（不光是金钱，还需要时间），因而一般只需通过优化算法根据累积难度来标识出主链即可。这是PoW下用高额成本的代价换来的成果——算力保护。  
【new\_target】 = 【prev\_target】 \* 【前2015个区块生成所用的时间】 /  1209600 （按标准每10分钟出一个块，2016个块所需要的秒数）