北京大学肖臻老师《区块链技术与应用》公开课笔记

> 以太坊挖矿难度调整，对应肖老师视频：click here 全系列笔记请见：click here **About Me:**点击进入我的Personal Page

前文中介绍了比特币难度调整是每隔2016个区块调整难度，从而达到维持出块时间10min的目标。而以太坊则与之不同，每个区块都有可能会进行难度调整。以太坊难度调整较为复杂，存在多个版本，网络上存在诸多不一致，这里遵循**以代码逻辑为准**的原则，从代码中查看以太坊难度调整算法。

以太坊难度调整

以太坊中区块难度调整算法如下图所示：

区块难度D(H)

当前区块编号(序号)

$$D(H) \equiv \begin{cases} D_0 & \text{if } H_i = 0 \\ \max(D_0, P(H)_{H_d} + x \times \varsigma_2) + \epsilon & \text{otherwise} \end{cases}$$

难度炸弹(后面会详细解释)

where:

(42) $D_0 \equiv 131072$

➢ 参数说明

- D(H)是本区块的难度，由基础部分$P(H)_{H_d}+x \times \varsigma_2$和难度炸弹部分$\epsilon$相加得到。
- $P(H)_{H_d}$为父区块的难度，每个区块的难度都是在父区块难度的基础上进行调整。
- $x \times \varsigma_2$用于自适应调节出块难度，维持稳定的出块速度。
- 基础部分有下界，为最小值$D_0 = 131072$。

自适应难度调整 $x \times \varsigma_2$

$$x \equiv \left\lfloor \frac{P(H)_{H_d}}{2048} \right\rfloor \tag{43}$$

$$\varsigma_2 \equiv \max\left(y - \left\lfloor \frac{H_s - P(H)_{H_s}}{9} \right\rfloor, -99\right) \tag{44}$$

➢ 参数说明

- x是调整的单位，ς_2为调整的系数。
- y和父区块的uncle数有关。如果父区块中包括了uncle ,则y为2，否则y为1。
  - 父块包含uncle时难度会大一个单位，因为包含uncle时新发行的货币量大，需要适当提高难度以保持货币发行量稳定。
- 难度降低的上界设置为−99，主要是应对被黑客攻击或其他目前想不到的黑天鹅事件。

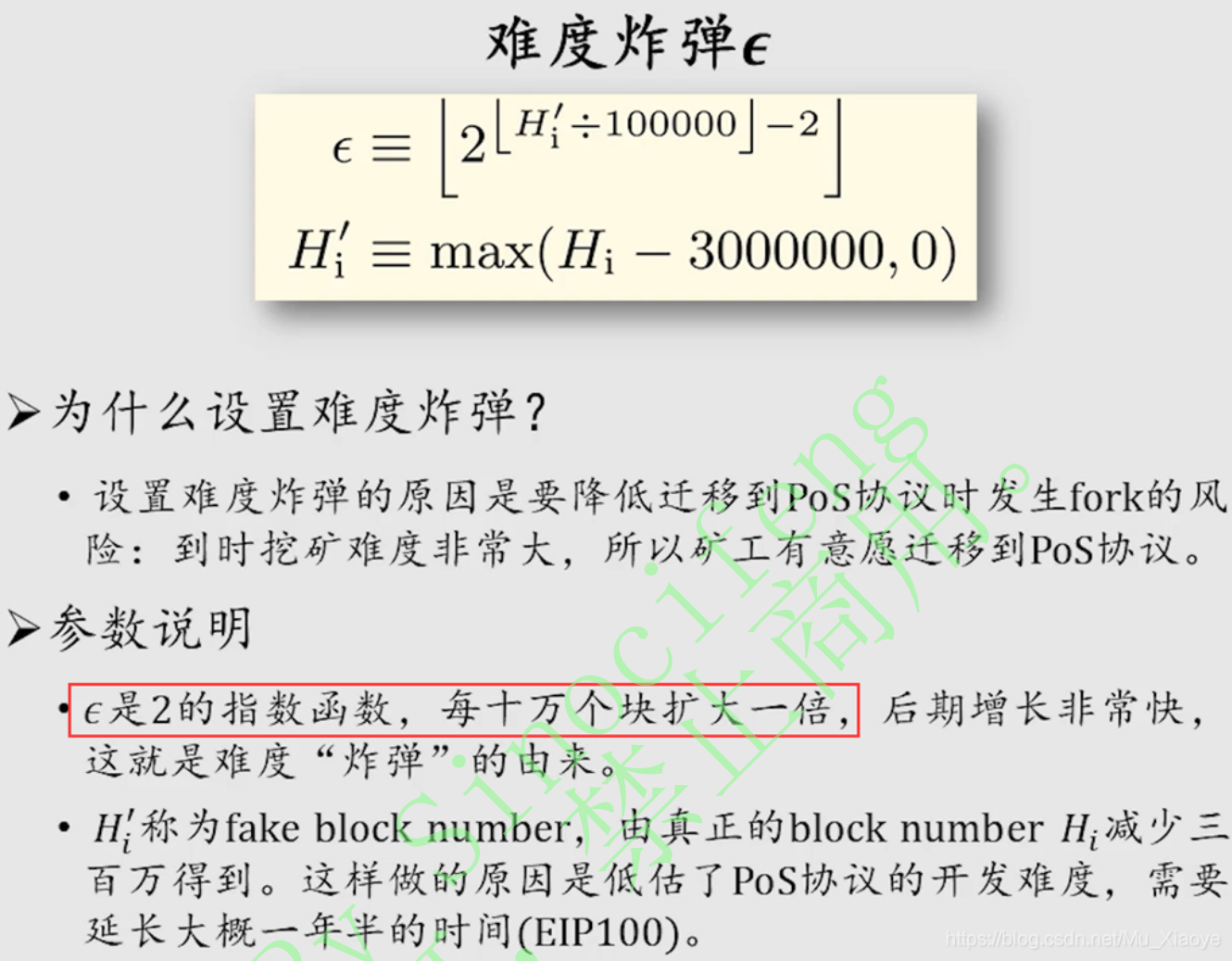
$$y - \left\lfloor \frac{H_s - P(H)_{H_s}}{9} \right\rfloor$$

- 参数说明
  - H_s是本区块的时间戳，$P(H)_{H_s}$是父区块的时间戳，均以秒为单位，并规定$H_s > P(H)_{H_s}$。
    - 该部分是稳定出块速度的最重要部分：出块时间过短则调大难度，出块时间过长则调小难度。
- 以父块不带uncle($y = 1$)示例
  - 出块时间在[1,8]之间，出块时间过短，难度调大一个单位。
  - 出块时间在[9,17]之间，出块时间可以接受，难度保持不变。
  - 出块时间在[18,26]之间，出块时间过长，难度调小一个单位。
  - ……

难度炸弹

为什么要设置难度炸弹? 根据以上以太坊难度调整算法可以看到，该算法可以很好地动态调整挖矿难度，从而保障系统整体出块时间维持在15s左右。但之前在挖矿算法的文章中有介绍到，以太坊在设计之初就计划要逐步从POW（工作量证明）转向POS（权益证明），而权益证明不需要挖矿。 从旁观者角度来看，挖矿消耗了大量电力、资金等，如果转入放弃挖矿，必然是一件好事。但从矿工的角度，花费了很大精力投入成本购买设备，突然被告知"不挖矿了"，这必然是一件很难接受的事情。而以太坊本身为一个分布式系统，其转入POS必须经过系统中大多数矿工认可才行，如果届时矿工联合起来转入POS，那么这一设计初衷就成了一江流水。 因此，以太坊在设计之初便

添加了难度炸弹，迫使矿工转入POS。那么 ***如何促使矿工自愿升级软件，而非坚持POS呢?***

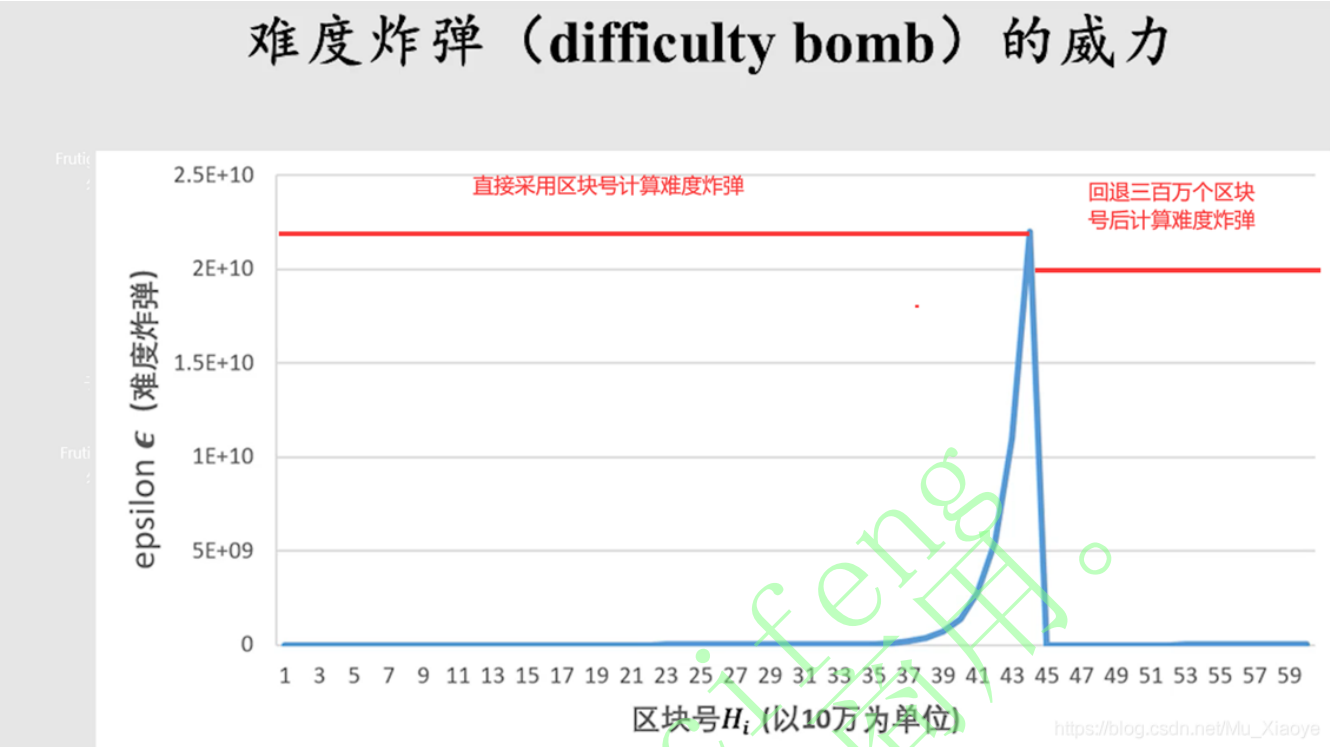


> 数学上，指数函数是一个很可怕的东西。我们谈论一个算法，无论其时间复杂度还是空间复杂度，只要达到了指数级别，这个算法必然难以应用于大规模计算上。指数函数在前期增长相对缓慢，但在后期呈现“指数爆炸”，而这往往是我们无法通过升级硬件所能解决的。

可以看到，在以太坊早期时，区块号较小，难度炸弹计算所得值较小，难度调整级别基本上通过难度调整中的自适应难度调整部分决定，而随着越来越多区块被挖出，难度炸弹的威力开始显露出来，这也就使得挖矿变得越来越难，从而迫使矿工愿意转入POS。

难度炸弹调整

上面提到，以太坊设想是通过埋设难度炸弹迫使矿工届时愿意转入权益证明，但现实中有一句话：“理想很丰满，现实很骨感”。在实际应用中，权益证明的方式仍然并不成熟，目前以太坊共识机制仍然是POW，依然需要矿工参与挖矿维护以太坊系统的稳定。也就是说，转入POS的时间节点被一再推迟，虽然挖矿变得越来越难，系统出块时间开始逐渐变长，但矿工仍然需要继续挖矿。 在上面难度炸弹的公式中，有人应该注意到了第二项中的fake block number，该数仅仅为对当前区块编号减去了三百万，也就是相当于将区块编号回退了三百万个。那么，在前三百万个区块的时候，这个fake block number就是负数吗？ 答案是否定的。实际上，在以太坊最初的设计中，并没有第二个公式。也就是说，最初就是简单地直接用区块编号除以100000。而在转入权益证明时间节点一再推迟后，以太坊系统采取了将区块编号回退三百万个区块的方法来降低挖矿难度，当然，为了保持公平，也将出块奖励从5个以太币减少到了3个以太币，这也是fake block number这一项出现的原因。 下图显示了难度调整对难度炸弹难度影响的结果：



以太坊发展

个人感觉这一块肖老师讲解得较为粗略，对于以太坊的四个阶段都只是简单提及，对"拜占庭"这一区块链中非常经典的问题并未进行介绍。可能是肖老师希望这些内容大家自行了解或是课程安排时间匆忙的缘故。 等写完肖老师整个区块链技术课程系列之后，如果大家对这块感兴趣，我会考虑查找资料写一下这些内容。在这里先占个坑。　　拜占庭将军问题暂时占坑，链接点击无效

区块链发展阶段介绍暂时占坑，链接点击无效 以太坊发展存在四个阶段，我们目前处于第三个阶段中的拜占

庭阶段，难度炸弹回调就是在拜占庭阶段进行的。

具体代码实现

1. 难度计算公式 bigTime为当前区块时间戳，bigParentTime为当前区块的父区块时间戳。

```
320  // calcDifficultyByzantium is the difficulty adjustment algorithm. It returns
321  // the difficulty that a new block should have when created at time given the
322  // parent block's time and difficulty. The calculation uses the Byzantium rules.
323  func calcDifficultyByzantium(time uint64, parent *types.Header) *big.Int {
324      // https://github.com/ethereum/EIPs/issues/100.
325      // algorithm: 难度计算公式:
326      // diff = (parent_diff +
327      //         (parent_diff / 2048 *
328      //            max((2 if len(parent.uncles) else 1) - ((timestamp - parent.timestamp) // 9), -99))
329      //        ) + 2^(periodCount - 2)难度炸弹
330      bigTime := new(big.Int).SetUint64(time)
331      bigParentTime := new(big.Int).Set(parent.Time)
332      x := new(big.Int)
333      y := new(big.Int)
```

2. 基础部分计算

```
// (2 if len(parent_uncles) else 1) - (timestamp - parent_timestamp) // 9
x.Sub(bigTime, bigParentTime)   当前时间戳-父区块时间戳，得到出块时间
x.Div(x, big9)      除以9向下取整
if parent.UncleHash == types.EmptyUncleHash {   判断是否有叔父区块
    x.Sub(big1, x)
} else {
    x.Sub(big2, x)
}
// max((2 if len(parent_uncles) else 1) - (timestamp - parent_timestamp) // 9, -99)
if x.Cmp(bigMinus99) < 0 {  和-99(下调界限)相比
    x.Set(bigMinus99)
}
// parent_diff + (parent_diff / 2048 *
// max((2 if len(parent.uncles) else 1) - ((timestamp - parent.timestamp) // 9), -99))
y.Div(parent.Difficulty, params.DifficultyBoundDivisor)  难度调整粒度
x.Mul(y, x)                    DifficultyBoundDivisor = big.NewInt(2048)
x.Add(parent.Difficulty, x)
// minimum difficulty can ever be (before exponential factor)
if x.Cmp(params.MinimumDifficulty) < 0 {
    x.Set(params.MinimumDifficulty)
}                  MinimumDifficulty      = big.NewInt(131072)   D0,难度下限
```

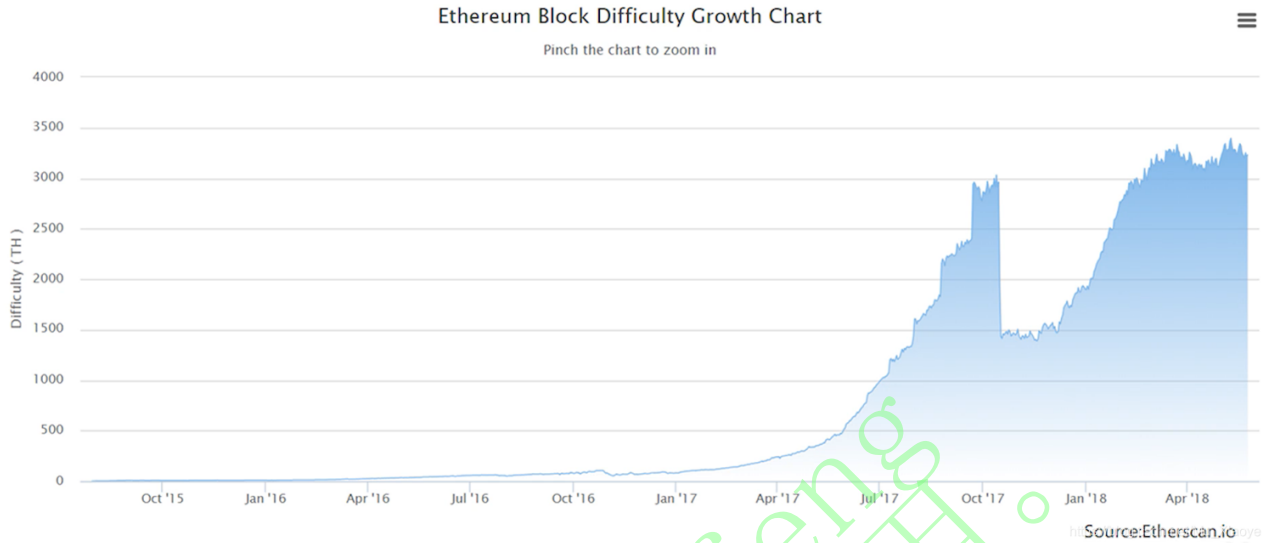
3. 难度炸弹计算

```
// calculate a fake block number for the ice-age delay:
//   https://github.com/ethereum/EIPs/pull/669
//   fake_block_number = min(0, block.number - 3_000_000
fakeBlockNumber := new(big.Int)  假的区块号
if parent.Number.Cmp(big2999999) >= 0 {  大于2999999则减去2999999
    fakeBlockNumber = fakeBlockNumber.Sub(parent.Number, big2999999)
}
// for the exponential factor
periodCount := fakeBlockNumber   expDiffPeriod = big.NewInt(100000)
periodCount.Div(periodCount, expDiffPeriod)  假区块号除以10万向下取整
// the exponential factor, commonly referred to as "the bomb"
// diff = diff + 2^(periodCount - 2)
if periodCount.Cmp(big1) > 0 {
    y.Sub(periodCount, big2)
    y.Exp(big2, y, nil)
    x.Add(x, y)  与基础部分难度相加
}
```

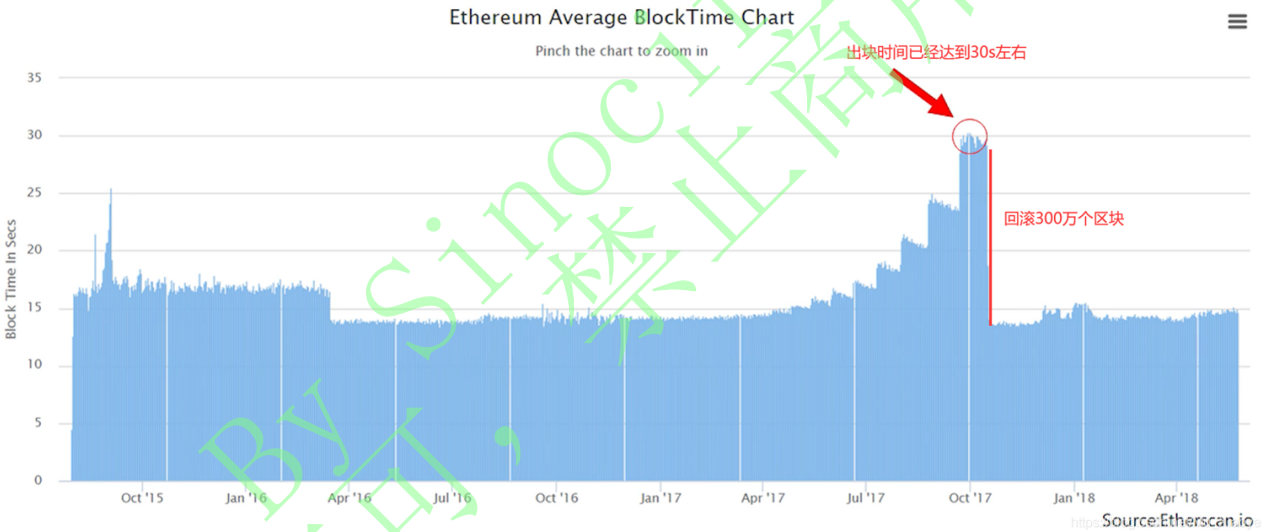
为什么不是减去3000000，而是2999999？ 因为这里判断的父区块号，而公式中是根据当前区块来算的。

以太坊实际统计数据(2018年，取自肖老师授课PPT)

1. 以太坊挖矿难度变化曲线 断崖式下跌是由于下调难度炸弹300万个区块。



2. 以太坊出块时间变化曲线图



3. 两个真实区块信息 difficulty为当前区块难度，total difficulty为当前区块链上所有区块难度相加。 可见，最长合法链也就等同于最难合法链（难度最大合法链）。

Height:	< Prev 5695161 Next >	Source:Etherscan.io
TimeStamp:	19 mins ago (May-29-2018 04:45:25 AM +UTC)	
Transactions:	89 transactions and 3 contract internal transactions in this block	
Hash:	0x76df197457effdbb736480393c70a016fe3bbdbfef619d16640cb665d748dcef	
Parent Hash:	0xbd3ecbcf5527bb6de899912cb86eadc762c86832c713bef3910bcec7184e0f7a	
Sha3Uncles:	0xde903bc6ba5e5ca6155d936f882a92a653f3b0a60a346f0f474fc56e61340ea9	
Mined By:	0xea674fdde714fd979de3edf0f56aa9716b898ec8 (**Ethermine**) in 20 secs	
Difficulty:	3,184,956,261,907,541	
Total Difficulty:	4,459,340,439,119,129,119,115	
Size:	18032 bytes	
Gas Used:	7,967,412 (99.74%)	
Gas Limit:	7,988,337	
Nonce:	0xd280930018199336	
Block Reward:	3.260603241218831558 Ether (3 + 0.166853241218831558 + 0.09375)	
Uncles Reward:	2.25 Ether (1 Uncle at Position 0)	
Extra Data:	ethermine-aws-us1-1 (Hex:0x65746865726d696e652d6177732d7573312d31)	

Height:	< Prev 5695150 Next >	Source:Etherscan.io
TimeStamp:	23 mins ago (May-29-2018 04:41:51 AM +UTC)	
Transactions:	160 transactions and 9 contract internal transactions in this block	
Hash:	0x92174a45e568b53e7aa0bdee81c73e7de9d214827546d11532f0c023889f4ee6	
Parent Hash:	0x3335cadce8ad38423511f0223fd7b9e5e2d1f46f0dca4d7d2464c00750a33fd1e	
Sha3Uncles:	0xabfb2427e51f6879e15b13c1aa9d327ec748fe99637628596fdc9f1b3ed52e4c	
Mined By:	0xea674fdde714fd979de3edf0f56aa9716b898ec8 (**Ethermine**) in 14 secs	
Difficulty:	3,189,637,521,586,694	
Total Difficulty:	4,459,305,357,839,994,234,039	
Size:	27993 bytes	
Gas Used:	7,994,188 (99.93%)	
Gas Limit:	8,000,029	
Nonce:	0xe1a977700b02217f	
Block Reward:	3.31510552614492296 Ether (3 + 0.12760552614492296 + 0.1875)	
Uncles Reward:	4.875 Ether (2 Uncles at Position 0, Position 1)	
Extra Data:	ethermine-eu8 (Hex:0x65746865726d696e652d657538)	

引入1个uncle的reward是$3 \times \frac{1}{32} = 0.09375$

引入2个uncle的reward是$2 \times 3 \times \frac{1}{32} = 0.1875$