(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106406896 B
(45)授权公告日 2020.03.17

(21)申请号 201610857645.6

(22)申请日 2016.09.27

(65)同一申请的已公布的文献号
申请公布号 CN 106406896 A

(43)申请公布日 2017.02.15

(73)专利权人 北京天德科技有限公司
地址 100089 北京市海淀区知春路113号1708-048

(72)发明人 邓恩艳

(51)Int.Cl.
G06F 8/20(2018.01)
G06F 16/21(2019.01)
G06F 16/27(2019.01)
G06Q 40/04(2012.01)

(56)对比文件
CN 105912618 A,2016.08.31,
CN 104463001 A,2015.03.25,
CN 105592098 A,2016.05.18,
CN 105488675 A,2016.04.13,
US 2016027229 A1,2016.01.28,

审查员 李艳军

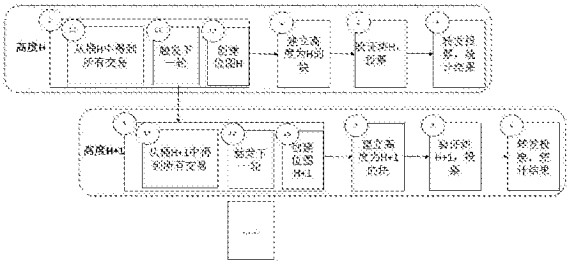
权利要求书2页 说明书5页 附图1页

(54)发明名称

一种并行PipeLine技术的区块链建块方法

(57)摘要

本发明提供了一种并行PipeLine的区块链建块方法,其特征在于包含以下步骤:(1)将数据分桶存储;(2)建立临时数据序号;(3)更改新一轮建块的触发点;(4)对剩余交易进行处理;以及(5)进行父哈希赋值运算。利用该建块方法,桶模型保证建块信息的完整性。



CN 106406896 B

1.一种并行PipeLine的区块链建块方法,其特征在于包含以下步骤:

(1)将交易数据分桶存储:从交易缓冲池取到交易数据,将交易数据按桶的序号分桶存储,桶的容量不小于块中交易数据的最大容量;

(2)分桶建立临时交易数据块序号,该序号对应于桶的高度,并建块;

(3)更改新一轮建块的触发点,这个触发点是指在交易数据分桶存储后,该桶拿到一个块序号时的事件点,触发下一轮建块;

(4)对缓冲池中还未处理的剩余交易数据进行建块处理;

(5)进行父哈希赋值运算:上一个区块建完之后,计算其哈希值,将其记入正在建块的块数据中。

2.根据权利要求1所述的一种并行PipeLine的区块链建块方法,其特征在于:所述步骤(1)具体为:对于数据存储进行了按桶序号分桶处理以实现多个块同时建立,并且从交易缓冲池开始,对每一个建块流程提供一个唯一的标识以表示该块的高度。

3.根据权利要求1所述的一种并行PipeLine的区块链建块方法,其特征在于:所述步骤(2)的建块运算包括:

(a)在要建一个新的高度height的块时,假设第height-1块已经开始,并从交易缓冲池中取走其交易集合;

(b)将新高度的块的序号定义为height,从缓冲池中取走该块所有的交易,放到一个块序号被标识为height的桶中;

(c)节点通信的所有信息,包括bitmap、block、投票和转发的投票都加上唯一的标识height,以保证在一个节点收到所有类型的消息都可以对应到唯一的一个桶序号对应的块,由此数据存储在多个桶中,使得系统中可以允许同时存在多个建块流程对应的数据信息。

4.根据权利要求1所述的一种并行PipeLine的区块链建块方法,其特征在于:所述步骤(3)包括:

(a)将原始的串行建块的系统中触发下一轮建块的操作关闭;

(b)在建块的步骤(1)将所有的交易拿到一个有交易块序号的桶中之后,触发下一轮建块。

5.根据权利要求1所述的一种并行PipeLine的区块链建块方法,其特征在于:所述步骤(4)在每一个建块过程结束之后,将没有放到区块中的交易存放到缓冲池中,留给接下来的建块流程使用。

6.根据权利要求1所述的一种并行PipeLine的区块链建块方法,其特征在于:所述步骤(5)的哈希值,除了区块链第一个区块之外,每一个区块建成之后,如果上一个区块还没有建完,将其挂在内存中,待上一个区块建完之后,将其存入本块中,再将其放到区块链里。

7.根据前述任意一个权利要求所述的一种并行PipeLine的区块链建块方法,其特征在于:为了容忍f个节点的故障或者被攻击(f是任何大于零的自然数),系统需要有3f+1个节点;节点在出现故障或者被攻击成功的情况下,如果节点总数超过被控制节点的三倍,系统的容错算法可以保证其余正常节点正常运作。

8.根据权利要求7所述的一种并行PipeLine的区块链建块方法,其特征在于:若每次投票过程中如果只有少于1/3的节点出现故障或者被攻击控制,系统可以正常运作,异常节点

恢复正常之后，会有一个同步机制，向其余节点进行请求，得到完整的区块链，从而保证任何一个节点在恢复正常之后可以正常的参与到新一轮的建块投票中，且保持了分布式系统数据的一致性和每个节点数据的完整性。

一种并行PipeLine技术的区块链建块方法

技术领域

[0001] 本发明涉及一种区块链建块技术，特别是一种基于并行PipeLine的区块链建块技术。

背景技术

[0002] 目前的区块链系统是以串行的方式建块的，串行方式的含义是完成建块的各个步骤后开始下一个新块的创建。由于建块流程本身可以进行切分成一些子流程，所以对于目前的串行建块方法，在确定了建一个块的交易集合之后，交易缓冲区中收到的交易可以用于建下一个块。目前的区块链系统中每次建一个块，每个建块过程大概包括以下几个阶段：

[0003] (1)将系统中所有交易映射到位图；选出一个leader节点，将自己的位图发送给其它所有节点；

[0004] (2)所有的节点对收到的位图求交集，根据得到的交集，确定建块的交易集合。leader节点建块，并将块发送给其余节点；

[0005] (3)所有的节点对块进行验证，根据验证结果，发送投票信息；

[0006] (4)所有节点转发自己的投票信息。在得到所有的投票之后，确定最后的投票结果。触发下一轮建块。

[0007] 分析上面的建块过程可以发现，现有的建块过程所需时间较长，对区块链系统响应时间长，计算机资源应用效率低，造成了内存的极大浪费。进一步研究发现在串行建块第一步完成时，已经可以确定建块的交易集合了，在对于这之后收到的交易，可以将其确定为第二个块中的内容。为了实现系统中同时存在两个或者多个这样的建块流程，系统利用桶模型进行多个块的并行建块。

发明内容

[0008] 本发明的目的在于提供一种并行PipeLine的区块链建块方法，包含以下步骤：(1)将数据分桶存储；(2)建立临时数据索引；(3)更改新一轮建块的触发点；(4)对剩余交易进行处理；以及(5)进行父哈希赋值运算。

[0009] 优选的，步骤(1)具体为：对于数据存储进行了按序号分桶处理以实现多个块同时建立，并且从交易缓冲区开始，对每一个建块流程提供一个唯一的标识height，表示这个块的高度。

[0010] 优选的，步骤(2)的建块运算包括：(a)在要建一个新的高度height的块时，假设第height-1块已经开始，并从交易缓冲区中取走其交易集合；(b)将新的高度块序号定义为height，从缓冲区中取走该块所有的交易，放到一个标识为height的桶中；(c)节点通信的所有信息，包括bitmap、block、投票和转发的投票都加上唯一的标识height，以保证在一个节点收到所有类型的消息都可以对应到唯一的一个序号对应的块，由此数据存储在多个桶中，使得系统中可以允许同时存在多个建块流程对应的数据信息。

[0011] 优选的，步骤(3)包括：(a)将原始的串行建块的系统中触发下一轮建块的操作关

闭；(b)在建块的步骤(1)将所有的交易拿到一个有标识的桶中之后，触发下一轮建块。

[0012] 优选的，步骤(4)在每一个建块过程结束之后，将没有放到区块中的交易存放到缓冲区中，留给接下来的建块流程使用。

[0013] 优选的，步骤(5)的哈希值，除了区块链第一个区块之外，每一个区块建成之后，如果上一个区块还没有建完，将其挂在内存中，待上一个区块建完之后，将其存入本块中，再将其放到区块链里。

[0014] 优选的，为了容忍f个节点的故障或者被攻击，系统需要有3f+1个节点。节点在出现故障或者被攻击成功的情况下，如果节点总数超过被控制节点的三倍，系统的容错算法可以保证其余正常节点正常运作。

[0015] 优选的，若每次投票过程中如果只有少于1/3的节点出现故障或者被攻击控制，系统可以正常运作，异常节点恢复正常之后，会有一个同步机制，向其余节点进行请求，得到完整地区块链，从而保证任何一个节点在恢复正常之后可以正常的参与到新一轮的建块投票中，且保持了分布式系统数据的一致性和每个节点数据的完整性。

[0016] 为了实现系统中同时存在两个或者多个这样的建块流程，系统利用桶模型进行多个块的并行建块。具体来讲，每个建块流程会有较多的中间数据和一些消息传递，包括建块的交易数据、位图、尚未存入区块链的块和投票信息。这些临时数据存储在内存中，为系统中一个唯一的建块流程服务。为了允许系统同时存在多个建块流程，每个建块流程的中间数据会在内存中被放到一个带有标号的桶中，这个桶的标号是唯一的。在开始新一轮建块时，将所有从缓冲区中的交易拿出来，放到一个桶中。接下来所有涉及对交易的读写的请求都会从这个桶中的交易进行。显然，建块会用到交易集合是桶中的交易集合的子集。而对于建块过程中没有用到的交易，从桶中将其重新放到接收交易的缓冲区中，在接下来的建块过程中处理。

[0017] 不同的交易集合放在不同的桶中，每个建块流程根据自己的标号从桶中读取数据，用这样的方式将数据分桶处理，桶之间的数据不会有交集和干扰。建块数据分开之后，为了实现建块流程的并行，系统还要开启多线程，不同的建块流程有不同的线程处理。对于单机单核节点，桶模型可以增加建块速度，因为在一个建块流程中有很多的等待时间，在选出主节点之后，其余所有节点此时同步等待主节点将块发来，而在主节点发块之后，等待其余所有节点发来的投票信息，处于空闲状态。除此之外，在系统进行投票的过程中，同步的等待其余节点的票。因此，一个建块流程中计算机有大量的空闲时间。如果利用桶模型同时建多个块，计算机的使用效率会更高，交易的响应速度也更快。

[0018] 本发明所提供了的一种并行PipeLine技术的建块方法，可以达到对计算机的高效利用，实现区块链系统的快速响应。利用并行PipeLine技术，每个节点的计算机资源都被更加充分的利用。计算资源和内存资源的增加可以直接提高建块的效率，这提高了系统的可扩展性。当系统的压力更大时，通过增加节点CPU以及内存容量就可以进一步提高系统的响应速度。桶模型的提出保证建块信息的完整性，包括交易bitmap、块、投票信息；块与块信息之间的隔离性；还有重要一点是保证块的顺序，使其不受网络延迟等的影响。另外，并行PipeLine技术的使用对现有的建块方式性能进行了改善，对于上层业务是透明的，因此具有很好的可移植性，可以在不同的建块方案中运用。

[0019] 根据下文结合附图对本发明具体实施例的详细描述，本领域技术人员将会更加明

了本发明的上述以及其他目的、优点和特征。

附图说明

[0020] 后文将参照附图以示例性而非限制性的方式详细描述本发明的一些具体实施例。附图中相同的附图标记标示了相同或类似的部件或部分。本领域技术人员应该理解，这些附图未必是按比例绘制的。本发明的目标及特征考虑到如下结合附图的描述将更加明显，附图中：

[0021] 图1是根据本发明实施例的基于并行PipeLine的并行建块过程示意图；

[0022] 图2是根据本发明实施例的桶模型示意图；

[0023] 图3是根据本发明实施例的建块结束后的处理示意图。

具体实施方式

[0024] 在进行具体实施方式的说明之前，为了更为清楚的表达所论述的内容，首先说明本发明所涉及具体实施方式的机理，即为了实现基于并行PipeLine的并发建块，需要对原有的建块方案进行两个方面的改变：

[0025] (1)多线程技术异步编程：并行的建块策略只有结合多线程技术才能真正的提高系统利用率；

[0026] (2)基于桶模型的中间数据管理：建块过程中的所有数据都要实现隔离，保证不会出现数据的混乱。建块过程中的数据可以分为两个方面，网络传输的数据和缓存数据。

[0027] 网络传输的数据都是利用Json格式进行传输的，为了标识每个传输的数据是哪一个桶中的数据，在Json对象中增加了一个字段height，用于表示每个传输数据是给哪一个块使用的。

[0028] 系统的缓存系统采用redis，所有的缓存数据主要存储在redis中。redis是以key/value形式存储的，key的类型是字符串。为了实现数据的分桶，在原有的key的基础上增加height号，例如对于height为1的桶，只要对原有的key的前部增加“1_”，即可标识是哪一个桶。每个交易都会根据不同的height号，在Redis中分到不同的桶中。

[0029] 此外，系统至少要有4个节点，每4个节点可以容忍1个节点出现故障或者被攻击。为了容忍f个节点的故障或者被攻击，系统需要有3f+1个节点。节点在出现故障或者被攻击成功的情况下，如果节点总数超过被控制节点的三倍，系统的容错算法可以保证其余正常节点正常运作。根据拜占庭算法，即M.Pease,L.Lamport,S.Shostak.The Byzantine generals problem[J].ACM Trans.Programming Languagesand Systems,1982,4(3):382～401中的相关内容可知，为了容忍f个单机发生拜占庭故障，冗余系统至少需要存在3f+1个单机，也就是说系统至少要有4个节点，4个节点可以容忍一个节点出现故障或者被攻击。为了容忍f个节点的故障或者被攻击，系统需要有3f+1个节点。在进行一轮建块的过程中，如果出现了建块失败，即最终大家的肯定投票数量不足总结点数的2/3，则认为本轮建块失败，开始新一轮建块，此时区块链的高度不会增加。

[0030] 每次投票过程中如果只有少于1/3的节点出现故障或者被攻击控制，系统可以正常运作。异常节点恢复正常之后，会有一个同步机制。向其余节点进行请求，得到完整地区块链。这样的方式保证了任何一个节点在恢复正常之后可以正常的参与到新一轮的建块投

票中，保持了分布式系统数据的一致性和每个节点数据的完整性。

[0031] 实施例

[0032] 假设区块链系统中有4个节点(即M＝4)，分别为节点A、节点B、节点C、节点D，当采用本发明的方法进行建块时，系统有如下几步：

[0033] 步骤一：每个节点首先将交易从缓冲区中拿出来，放到桶中，标号为h。将h中的交易映射得到一个bitarray，记为为H_bitarray，如图2所示。重新启动一个线程，执行步骤一。进入步骤二，流程示意图如图1所示。

[0034] 步骤二：

[0035] 节点A:将自己收到的交易映射到bitarray上，得到一个bitarray发给节点B、C、D；

[0036] 节点B:将自己收到的交易映射到bitarray上，得到一个bitarray发给节点A、C、D；

[0037] 节点C:将自己收到的交易映射到bitarray上，得到一个bitarray发给节点A、B、D；

[0038] 节点D:将自己收到的交易映射到bitarray上，得到一个bitarray发给节点A、B、C；

[0039] 在第一次的通信之后，所有节点根据得到的bitarray求2/3交集运算，运算结果记为ba，也就是说每一位如果有2/3以上的bitarray都是1，则运算结果的bitarray在该位为1，否则为0运算结果记为BA。

[0040] 在系统中运行RoundRobin算法，随机得到一个leader。具体的做法是根据当前块的高度H和轮次R做hash运算，hash运算结果对M取模，根据取模结果确定第几个节点来建块，从而得到leader节点。不失一般性，假设节点A被选为leader，此时节点A根据BA和自己收到的交易，得到一个交易集合BS，BS满足其中的每一个交易映射到BA上所对应的位都为1。

[0041] 利用这个交易集合构建一个块AB，开始第二轮通信：

[0042] 节点A:将块AB发给节点B、C、D；

[0043] 节点B、C、D在收到块AB之后，利用自己的BA，遍历块AB中的交易。如果块中的某一个交易映射到BA中的一位对应位置为0，则认为投票信息为0+hash(AB)，否则为1+hash(AB)。

[0044] 节点A的投票信息为1+hash(AB)，对投票信息利用自己的私钥进行加密，得到数字签名，投票信息结构如图3所示。

[0045] 接下来开始第三次通信，也就是第一轮投票：

[0046] 节点A:将投票信息和数字签名发给节点B、C、D；

[0047] 节点B:将投票信息和数字签名发给节点A、C、D；

[0048] 节点C:将投票信息和数字签名发给节点A、B、D；

[0049] 节点D:将投票信息和数字签名发给节点A、B、C。

[0050] 每个节点会收到3个投票，根据数字签名验证收到的投票信息的真伪性。抛弃所有的非法投票信息后，得到一个投票集合，对这个投票集合求hash散列值之后，利用自己的私钥对其加密得到数字签名。

[0051] 接下来开始第四次通信，即第二轮投票：

[0052] 节点A:发送投票列表和数字签名给节点B、C、D；

[0053] 节点B:发送投票列表和数字签名给节点A、C、D；

[0054] 节点C:发送投票列表和数字签名给节点A、B、D；

[0055] 节点D:发送投票列表和数字签名给节点A、B、C。

[0056] 每个节点可以得到节点的投票信息,利用数字签名进行合法性认证,认为不合法的投票信息都是投否定票。对所有的投票信息进行统计汇总。不失一般性,以节点A对投票结果的统计为例展示每个节点的统计方式,A节点根据B在第三次通信发给自己的投票和节点C、D在第四次通信发给自己的他们所收到的B的次一轮投票,得到了B投给A、C、D三个节点的投票信息,假设B的投票结果为(A:1,C:1,D:1),由于肯定票的个数大于2/3,认定B的投票结果为1,否则认为B的投票为0。对于节点C、D,利用同样的方式即可得到其最终的投票结果。

[0057] 根据节点B、C、D以及自己的投票,如果投肯定票的数量超过3个(节点总数的2/3),则认为这个块合法,将其存入链中。否则抛弃。具体的执行示意图如图3所示。

[0058] 以上仅对M=4的情况进行了说明,当M=5或6时,其进行两轮通信的原理和方法与M=4的情况相同。

[0059] 虽然本发明已经参考特定的说明性实施例进行了描述,但是不会受到这些实施例的限定而仅仅受到附加权利要求的限定。本领域技术人员应当理解可以在不偏离本发明的保护范围和精神的情况下对本发明的实施例能够进行改动和修改。

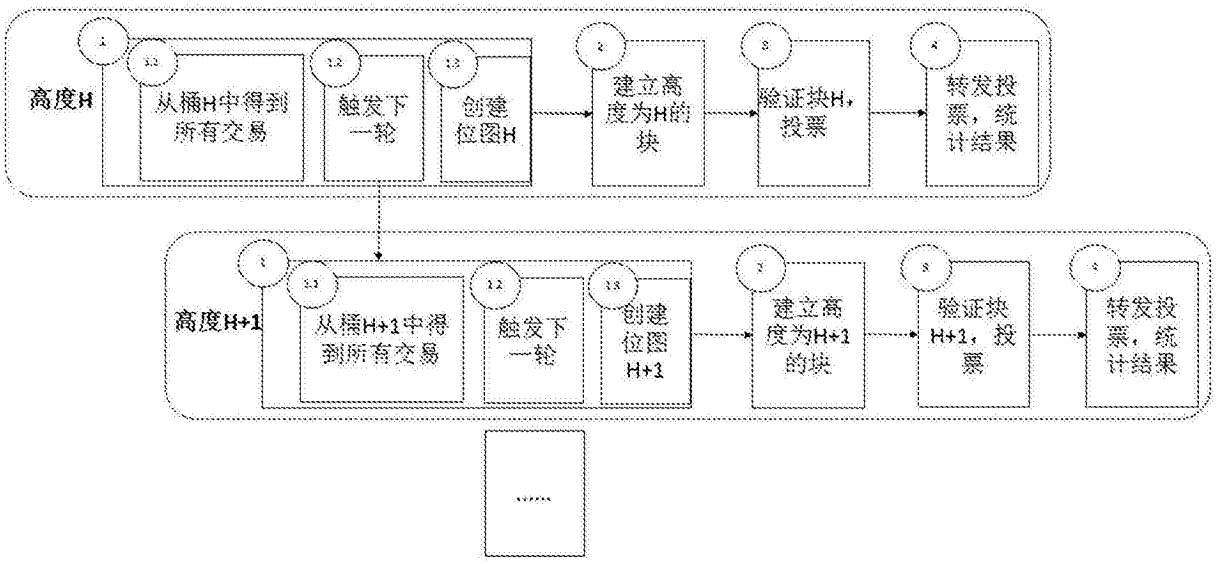

高度H
从桶H中得到所有交易
触发下一轮
创建位图H
建立高度为H的块
验证块H，投票
转发投票，统计结果
高度H+1
从桶H+1中得到所有交易
触发下一轮
创建位图H+1
建立高度为H+1的块
验证块H+1，投票
转发投票，统计结果
……

图1

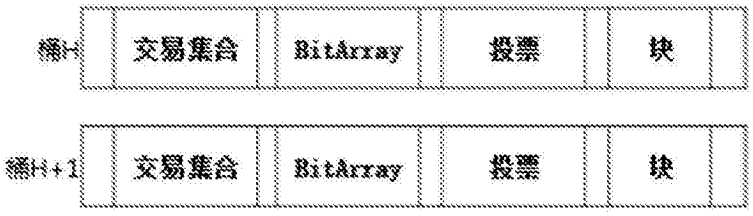

桶H
交易集合
BitArray
投票
块
桶H+1
交易集合
BitArray
投票
块

图2

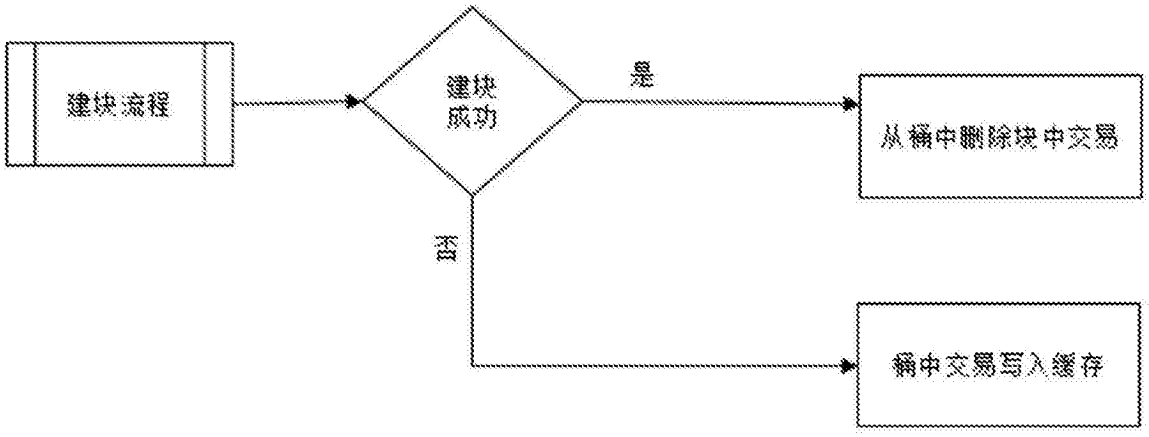

建块流程
建块成功
是
否
从桶中删除块中交易
桶中交易写入缓存

图3