(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110958303 B
(45)授权公告日 2022.06.24

(21)申请号 201911110047.2

(22)申请日 2019.11.14

(65)同一申请的已公布的文献号
申请公布号 CN 110958303 A

(43)申请公布日 2020.04.03

(73)专利权人 杭州复杂美科技有限公司
地址 310000 浙江省杭州市西湖区文三路90号东部软件园6号楼7层702室

(72)发明人 虞康 王志文 曹兢 李斌 吴思进

(51)Int.Cl.
H04L 67/1097(2022.01)
H04L 67/1023(2022.01)
H04L 45/7453(2022.01)
H04L 67/1095(2022.01)

(56)对比文件
CN 109272316 A,2019.01.25
CN 109977274 A,2019.07.05
CN 110264207 A,2019.09.20
CN 110442577 A,2019.11.12

审查员 王伟超

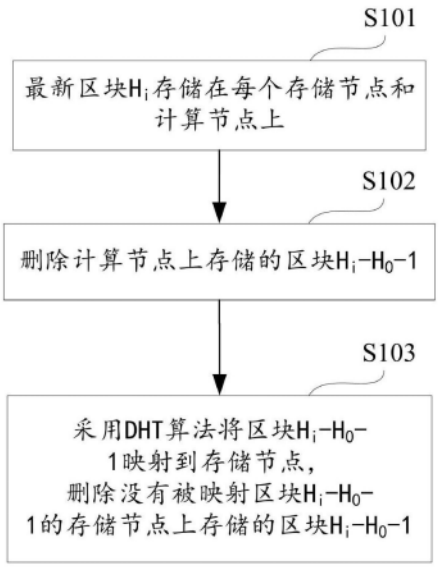
权利要求书1页 说明书6页 附图3页

(54)发明名称

一种区块存储和查询方法、设备及存储介质

(57)摘要

本发明公开了一种区块存储和查询方法、设备及存储介质，属于区块链技术领域。包括：最新区块H_i存储在每个存储节点和计算节点上；删除计算节点上存储的区块H_i-H_0-1；采用DHT算法将区块H_i-H_0-1映射到存储节点，删除没有被映射区块H_i-H_0-1的存储节点上存储的区块H_i-H_0-1；其中，区块H_i-H_0-2及其之前的区块均采用DHT算法映射，并存储到存储节点；每个存储节点和计算节点上均存储区块H_i-1至区块H_i-H_0-1；$H_i>H_0>1$，H_i和H_0均为整数，H_0大于区块回滚区间。可以降低整个网络的存储成本，实现账本的分布式存储，弹性存储和扩容。



CN 110958303 B

1.一种区块存储方法,其特征在于,包括:

最新区块H_i存储在每个存储节点和计算节点上;

删除计算节点上存储的区块H_i-H_0-1;

采用DHT算法将区块H_i-H_0-1映射到存储节点,删除没有被映射区块H_i-H_0-1的存储节点上存储的区块H_i-H_0-1;

其中,区块高度从1至H_i-H_0-2的区块均采用DHT算法映射,并存储到存储节点;每个存储节点和计算节点上均存储区块H_i-1至区块H_i-H_0-1;

H_i>H_0>1,H_i和H_0均为整数,H_0大于区块回滚区间。

2.根据权利要求1所述的方法,其特征在于,所述DHT算法为KAD算法。

3.根据权利要求2所述的方法,其特征在于,每个存储节点和计算节点上均保存一张KAD路由表。

4.根据权利要求2所述的方法,其特征在于,所述采用DHT算法将区块H_i-H_0-1映射到存储节点,进一步为:

采用哈希算法计算区块H_i-H_0-1哈希值和存储节点ID哈希值;找到与区块H_i-H_0-1哈希值最接近的N个存储节点ID哈希值对应的N个存储节点,建立区块H_i-H_0-1与存储节点之间的映射关系;N为区块冗余度。

5.根据权利要求2所述的方法,其特征在于,所述区块H_i-H_0-2及其之前的区块均采用DHT算法映射,并存储到存储节点,进一步为:

采用哈希算法计算区块i哈希值和存储节点ID哈希值;找到与区块i哈希值最接近的N个存储节点ID哈希值对应的N个存储节点,建立区块i与存储节点之间的映射关系;将区块i存储到对应的存储节点上;i=i+1,重复上述步骤,直到i=H_i-H_0-2为止;i为整数,取值范围为[1,H_i-H_0-2];N为区块冗余度。

6.根据权利要求2所述的方法,其特征在于,当存在所述存储节点离开时,采用KAD算法存储离开节点上存储的区块。

7.根据权利要求2所述的方法,其特征在于,当存在所述存储节点增加时,根据存储节点的种子节点更新新增存储节点的KAD路由表,采用KAD算法重新分配新增存储节点相邻的区块信息。

8.一种区块查询方法,其特征在于,根据权利要求1所述的一种区块存储方法,包括:

采用哈希算法计算存储节点ID哈希值和计算节点ID哈希值;

将所有的存储节点和计算节点哈希后分桶建立路由表;

找到与区块Hn哈希值最近的计算节点ID哈希值对应的计算节点,向所述计算节点请求获取区块Hn,若最近的计算节点上存在区块Hn,则返回区块Hn;若最近的计算节点上不存在区块Hn,则通过KAD路由表迭代查找到最近的保存区块Hn的存储节点,请求获取区块Hn。

9.一种计算机设备,其特征在于,所述设备包括:一个或多个处理器;存储器,用于存储一个或多个程序,当所述一个或多个程序被所述一个或多个处理器执行时,使得所述一个或多个处理器执行如权利要求1-8中任一项所述的方法。

10.一种存储有计算机程序的存储介质,其特征在于,该程序被处理器执行时实现如权利要求1-8中任一项所述的方法。

一种区块存储和查询方法、设备及存储介质

技术领域

[0001] 本发明涉及区块链技术领域，尤其涉及一种区块存储和查询方法、设备及存储介质。

背景技术

[0002] 现有的区块链系统中，随着区块的不断增加，账本存储空间势必越来越大，然而存储空间总是有限的，这必将会引起空间不足的问题；另外，存储和计算捆绑在一起的方法，不利于资源的充分利用。

发明内容

[0003] 1.发明要解决的技术问题

[0004] 为了克服上述技术问题，本发明提供了一种区块存储和查询方法、设备及存储介质。可以降低整个网络的存储成本，实现账本的分布式存储，弹性存储和扩容。

[0005] 2.技术方案

[0006] 为解决上述问题，本发明提供的技术方案为：

[0007] 一种区块存储方法，包括：最新区块H_i存储在每个存储节点和计算节点上；删除计算节点上存储的区块H_i-H_0-1；采用DHT算法将区块H_i-H_0-1映射到存储节点，删除没有被映射区块H_i-H_0-1的存储节点上存储的区块H_i-H_0-1；其中，区块H_i-H_0-2及其之前的区块均采用DHT算法映射，并存储到存储节点；每个存储节点和计算节点上均存储区块H_i-1至区块H_i-H_0-1；$H_i>H_0>1$，H_i和H_0均为整数，H_0大于区块回滚区间。

[0008] 可选地，所述DHT算法为KAD算法。

[0009] 可选地，每个存储节点和计算节点上均保存一张KAD路由表。

[0010] 可选地，所述采用DHT算法将区块H_i-H_0-1映射到存储节点，进一步为：采用哈希算法计算区块H_i-H_0-1哈希值和存储节点ID哈希值；找到与区块H_i-H_0-1哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块H_i-H_0-1与存储节点之间的映射关系；N为区块冗余度。

[0011] 可选地，所述区块H_i-H_0-2及其之前的区块均采用DHT算法映射，并存储到存储节点，进一步为：采用哈希算法计算区块i哈希值和存储节点ID哈希值；找到与区块i哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块i与存储节点之间的映射关系；将区块i存储到对应的存储节点上；$i=i+1$，重复上述步骤，直到$i=H_i-H_0-2$为止；i为整数，取值范围为$[1,H_i-H_0-2]$；N为区块冗余度。

[0012] 可选地，当存在所述存储节点离开时，采用KAD算法存储离开节点上存储的区块。

[0013] 可选地，当存在所述存储节点增加时，根据存储节点的种子节点更新新增存储节点的KAD路由表，采用KAD算法重新分配新增存储节点相邻的区块信息。

[0014] 一种区块查询方法，根据以上所述的一种区块存储方法，包括：采用哈希算法计算存储节点ID哈希值和计算节点ID哈希值；分桶建立路由表；向最近的计算节点请求获取区

块;将所有的存储节点和计算节点哈希后分桶建立路由表;找到与区块Hn哈希值最近的计算节点ID哈希值对应的计算节点,向所述计算节点请求获取区块Hn,若最近的计算节点上存在区块Hn,则返回区块Hn;若最近的计算节点上不存在区块Hn,则通过KAD路由表迭代查找到最近的保存区块Hn的存储节点,请求获取区块Hn。

[0015] 一种设备,所述设备包括:一个或多个处理器;存储器,用于存储一个或多个程序,当所述一个或多个程序被所述一个或多个处理器执行时,使得所述一个或多个处理器执行如以上所述的方法。

[0016] 一种存储有计算机程序的存储介质,该程序被处理器执行时实现如以上任一项所述的方法。

[0017] 3.有益效果

[0018] 采用本发明提供的技术方案,与现有技术相比,具有如下有益效果:

[0019] 单一节点不再需要存储全账本数据,解决了账本无限膨胀的技术问题;实现了存储和计算节点的分离,最大化了资源的利用,降低了整个网络的存储成本,真正实现了账本的分布式存储,弹性存储和扩容变成可能。

附图说明

[0020] 图1为本发明提供的一种区块存储方法的流程图。

[0021] 图2为图1所示方法的区块存储情况示意图。

[0022] 图3为图1进一步改进方法的流程图之一。

[0023] 图4为图1进一步改进方法的流程图之二。

[0024] 图5为本发明提供的一种区块查询方法的流程图。

[0025] 图6为本发明的一种设备结构示意图。

[0026] 图7为实施例3中的区块存储情况示意图。

具体实施方式

[0027] 为进一步了解本发明的内容,结合附图及实施例对本发明作详细描述。

[0028] 下面结合附图和实施例对本申请作进一步的详细说明。可以理解的是,此处所描述的具体实施例仅仅用于解释相关发明,而非对该发明的限定。另外还需要说明的是,为了便于描述,附图中仅示出了与发明相关的部分。

[0029] 本发明中所述的第一、第二等词语,是为了描述本发明的技术方案方便而设置,并没有特定的限定作用,均为泛指,对本发明的技术方案不构成限定作用。

[0030] 需要说明的是,在不冲突的情况下,本申请中的实施例及实施例中的特征可以相互组合。下面将参考附图并结合实施例来详细说明本申请。

[0031] 实施例1

[0032] 一种区块存储方法,如图1所示,包括:

[0033] S101、最新区块H_i存储在每个存储节点和计算节点上;

[0034] S102、删除计算节点上存储的区块H_i-H_0-1;

[0035] S103、采用DHT算法将区块H_i-H_0-1映射到存储节点,删除没有被映射区块H_i-H_0-1的存储节点上存储的区块H_i-H_0-1;

[0036] 其中，区块H_i-H_0-2及其之前的区块均采用DHT算法映射，并存储到存储节点；每个存储节点和计算节点上均存储区块H_i-1至区块H_i-H_0-1；

[0037] H_i＞H_0＞1，H_i和H_0均为整数，H_0大于区块回滚区间，通过对区块回滚区间的限定，将区块数据分成两类：不会更改的历史数据，以及可能回滚的非历史数据，历史数据采用分布式存储方式存储在本申请所述的存储节点上，提高区块数据的冗余度，可以在确保区块数据安全的前提下，节省数据存储空间；非历史数据供高频率的读写和计算使用。

[0038] 单一节点不再需要存储全账本数据，解决了账本无限膨胀的技术问题；实现了存储和计算节点的分离，最大化了资源的利用，降低了整个网络的存储成本，真正实现了账本的分布式存储，弹性存储和扩容变成可能。

[0039] 采用所述区块存储方法后，区块存储情况示意图，如图2所示，区块1至区块H_i-H_0-1采用DHT算法映射到存储节点，并存储到对应存储节点上；区块H_i-H_0至区块H_i存储在每个存储节点和计算节点上。存储节点和计算节点均可具有区块共识的功能。

[0040] DHT算法比较典型的方案主要包括：CAN、CHORD、Tapestry、Pastry、Kademlia(简称KAD)和Viceroy等，而Kademlia协议则是其中应用最为广泛、原理和实现最为实用、简洁的一种。本实施例中所述DHT算法为KAD算法，比如eMule、BitTorrent等P2P文件交换系统。

[0041] 每个存储节点和计算节点上均保存一张KAD路由表，所述KAD路由表上存储了节点上存储的区块的关键词Key，以及区块哈希值Hash。

[0042] 步骤S103中，所述采用DHT算法将区块H_i-H_0-1映射到存储节点，如图3所示，进一步为：

[0043] S201、采用哈希算法计算区块H_i-H_0-1哈希值和存储节点ID哈希值；

[0044] S202、找到与区块H_i-H_0-1哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块H_i-H_0-1与存储节点之间的映射关系；N为区块冗余度。通过设置冗余度，保存区块的冗余数据作为故障容错冗余数据。

[0045] 哈希算法包括MD4、MD5、SHA-1、SHA-224、SHA-256、SHA-384和SHA-512等，在具体应用中，可选择适合的哈希算法。

[0046] 所述区块H_i-H_0-2及其之前的区块均采用DHT算法映射，并存储到存储节点，如图4所示，进一步为：

[0047] S301、采用哈希算法计算区块i哈希值和存储节点ID哈希值；

[0048] S302、找到与区块i哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块i与存储节点之间的映射关系；

[0049] S303、将区块i存储到对应的存储节点上；i＝i+1，重复上述步骤，直到i＝H_i-H_0-2为止；i为整数，取值范围为[1,H_i-H_0-2]；N为区块冗余度。

[0050] 当存在所述存储节点离开时，采用KAD算法存储将离开存储节点上存储的区块。存储节点M即将离开，其上存储包含区块Hm在内的若干区块，按照KAD算法存储，找到存储节点ID哈希值离区块Hm哈希值最近的尚未存储区块Hm的存储节点，将区块Hm存储在该存储节点上，同时更新该存储节点的KAD路由表，将区块Hm的关键词以及区块Hm哈希值存储在KAD路由表内。存储节点M上的其他区块高度小等于H_i-H_0-1的区块存储过程类似。而取值范围为：[H_i-H_0,H_i]的区块，因每个存储节点和计算节点均存储有，则不必再进行存储。

[0051] 当存在所述存储节点增加时，根据存储节点的种子节点更新新增存储节点的KAD

路由表，采用KAD算法重新分配新增存储节点相邻的区块信息。新增存储节点N，新增存储节点N除存储取值范围为：[H_i-H_0,H_i]的区块外，还存储区块哈希值离新增存储节点N的ID哈希值较近的区块，也就是存储临近存储节点的区块数据；而新增存储节点N的存储数据多少则根据相邻存储节点的存储情况来确定，例如新增存储节点N为超级计算机，另发现左右临近存储节点中，某存储节点存储区块信息过多，则可以适当从该存储节点拉取部分区块信息，以分担较多的存储内容。

[0052] 实施例2

[0053] 一种区块查询方法，根据实施例1所述的一种区块存储方法，如图5所示，包括：

[0054] S401、采用哈希算法计算存储节点ID哈希值和计算节点ID哈希值；分桶建立路由表；

[0055] S402、找到与区块Hn哈希值最近的计算节点ID哈希值对应的计算节点；

[0056] S403、向所述计算节点请求获取区块Hn，若最近的计算节点上存在区块Hn，则返回区块Hn；若最近的计算节点上不存在区块Hn，则通过KAD路由表迭代查找到最近的保存区块Hn的存储节点，请求获取区块Hn。

[0057] 实施例3

[0058] 如图7所示，S1-S4代表存储节点，C1-C4代表计算节点，(Key,Hash)代表需要存储的区块Hash；图7中存储到离自己最近的节点上，是指区块存储到存储节点ID哈希值与区块哈希值最接近的N(冗余度为2，即N＝2)个存储节点上。

[0059] 存储节点、计算节点和区块Hash整个映射到一个DHT网络上；每个存储节点或计算节点保存一张KAD路由表，并保存部分区块信息。假设当前状态下区块高度为10，存储结构说明如下，在该实施例中，为便于描述，将区块回滚区间H_0设为2，在实际的区块链系统H_0的可根据实际需要而定。

[0060] 已经稳定(固定)的历史账本区块假设高度为(1～8)，按KAD算法区块(hash,block)映射到4个存储节点上，并同时存储非历史账本数据，即最新区块高度(9-10)，各存储节点存储的区块高度情况为：

[0061] S1节点：2,3,5,7,9,10

[0062] S2节点：1,3,5,6,9,10

[0063] S3节点：1,4,6,8,9,10

[0064] S4节点：2,4,7,8,9,10

[0065] 最新区块高度假设为(9-10)，按现有KAD算法存储到每一个节点上，计算节点C1,C2,C3,C4存储的区块高度情况为：9,10。

[0066] 现有技术中各节点均为全账本存储，即每个节点上均存储区块1-10的数据，那么存储区块1-10占用的容量为:10x8＝80；采用本实施例的技术方案后，存储区块1-10占用的容量为:6x4+2x4＝32；存储缩减为原来的32/80＝40％。综上所述，四个存储节点之间也保存了冗余数据作为故障容错冗余数据。

[0067] 新增区块11按照实施例所述方法进行存储，将区块11存储在存储节点S1、S2、S3和S4，以及计算节点C1,C2,C3,C4上，所有计算节点删除区块9，区块9映射在存储节点上，没有被映射到区块9的存储节点删除区块9。计算节点和存储节点上的区块存储情况如下：

[0068] S1节点：2,3,5,7,10,11；

[0069] S2节点:1,3,5,6,9,10,11;

[0070] S3节点:1,4,6,8,9,10,11;

[0071] S4节点:2,4,7,8,10,11;

[0072] 计算节点C1,C2,C3,C4存储的区块高度情况为:10,11。同时更新所有节点的KAD路由表。

[0073] 区块查找采用实施例2所述查询方法,假设Client目前需要请求高度为3的区块,步骤如下:

[0074] 1)S1,S2,S3,S4,C1,C2,C3,C4节点ID哈希运算后,分桶建立路由表;

[0075] 2)Client首先向计算节点C3请求key=hash3(或height=3)的区块,计算节点C3没有区块3,通过路由表迭代查找到存储节点ID哈希值离区块3哈希值hash3最近的存储节点;

[0076] 3)区块3哈希值hash3离存储节点S1和存储节点S2最近,定位到区块3的资源在存储节点S2或存储节点S1上;

[0077] 4)返回存储节点S2或存储节点S1的节点信息,Client重新向存储节点S1(或存储节点S2,存储节点S1不在线的情况下)发起获取区块3哈希值hash3的块信息,完成数据请求。

[0078] 当存在存储节点离开的情况时,更新所有相关节点KAD路由表,同时更新临近存储节点存储信息,比如图7中的存储节点S4(存储了高度为4、8、2、7,10,11的区块)不在线后,存储节点S1和存储节点S3需要同步存储节点S4的数据,存储节点S1从存储节点S3拉取高度为4,8的区块并存储,存储节点S1从存储节点S3拉取高度为2,7的区块并存储。

[0079] 当存在存储节点增加的情况时,通过种子节点(是指已经存储了若干区块,并且在区块链网络采用本实施例技术方案创建对应存储系统之初已经存在的存储节点)更新KAD路由表,同时重新分配存储相邻存储节点的区块信息,存储节点S5加入后,原来存储在存储节点S4上的区块7哈希值Hash7和存储在存储节点S1上的区块2哈希值Hash2块不再存储,转而放到存储节点S5上。

[0080] 本实施例充分利用DHT和KAD算法,结合区块链的特点,实现真正意义上的区块链分布式存储,真真意义上解决账本无限膨胀的技术问题;并实现了存储节点和计算节点的分离,最大化了资源的利用;经过对区块存储结构的改进,降低了整个网络的存储成本;实现了账本的分布式存储,弹性存储(每个节点上存储的区块数据根据节点自身容量而定)和扩容变成可能。

[0081] 实施例4

[0082] 一种设备,所述设备包括:一个或多个处理器;存储器,用于存储一个或多个程序,当所述一个或多个程序被所述一个或多个处理器执行时,使得所述一个或多个处理器执行如以上所述的方法。

[0083] 一种存储有计算机程序的存储介质,该程序被处理器执行时实现如以上实施例1所述的方法。

[0084] 图6为本发明一实施例提供的一种设备的结构示意图。

[0085] 如图6所示,作为另一方面,本申请还提供了一种设备500,包括一个或多个中央处理单元(CPU)501,其可以根据存储在只读存储器(ROM)502中的程序或者从存储部分508加

载到随机访问存储器(RAM)503中的程序而执行各种适当的动作和处理。在RAM503中,还存储有设备500操作所需的各种程序和数据。CPU501、ROM502以及RAM503通过总线504彼此相连。输入/输出(I/O)接口505也连接至总线504。

[0086] 以下部件连接至I/O接口505:包括键盘、鼠标等的输入部分506;包括诸如阴极射线管(CRT)、液晶显示器(LCD)等以及扬声器等的输出部分507;包括硬盘等的存储部分508;以及包括诸如LAN卡、调制解调器等的网络接口卡的通信部分509。通信部分509经由诸如因特网的网络执行通信处理。驱动器510也根据需要连接至I/O接口505。可拆卸介质511,诸如磁盘、光盘、磁光盘、半导体存储器等等,根据需要安装在驱动器510上,以便于从其上读出的计算机程序根据需要被安装入存储部分508。

[0087] 特别地,根据本申请公开的实施例,上述任一实施例描述的方法可以被实现为计算机软件程序。例如,本申请公开的实施例包括一种计算机程序产品,其包括有形地包含在机器可读介质上的计算机程序,所述计算机程序包含用于执行上述任一实施例描述的方法的程序代码。在这样的实施例中,该计算机程序可以通过通信部分509从网络上被下载和安装,和/或从可拆卸介质511被安装。

[0088] 作为又一方面,本申请还提供了一种计算机可读存储介质,该计算机可读存储介质可以是上述实施例的装置中所包含的计算机可读存储介质;也可以是单独存在,未装配入设备中的计算机可读存储介质。计算机可读存储介质存储有一个或者一个以上程序,该程序被一个或者一个以上的处理器用来执行描述于本申请的方法。

[0089] 附图中的流程图和框图,图示了按照本发明各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分,该模块、程序段、或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意,在有些作为替换的实现中,方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如,两个接连地表示的方框实际上可以基本并行地执行,它们有时也可以按相反的顺序执行,这根据所涉及的功能而定。也要注意的是,框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合,可以通过执行规定的功能或操作的专用的基于硬件的系统来实现,或者可以通过专用硬件与计算机指令的组合来实现。

[0090] 描述于本申请实施例中所涉及到的单元或模块可以通过软件的方式实现,也可以通过硬件的方式来实现。所描述的单元或模块也可以设置在处理器中,例如,各所述单元可以是设置在计算机或移动智能设备中的软件程序,也可以是单独配置的硬件装置。其中,这些单元或模块的名称在某种情况下并不构成对该单元或模块本身的限定。

[0091] 以上描述仅为本申请的较佳实施例以及对所运用技术原理的说明。本领域技术人员应当理解,本申请中所涉及的发明范围,并不限于上述技术特征的特定组合而成的技术方案,同时也应涵盖在不脱离本申请构思的情况下,由上述技术特征或其等同特征进行任意组合而形成的其它技术方案。例如上述特征与本申请中公开的(但不限于)具有类似功能的技术特征进行互相替换而形成的技术方案。

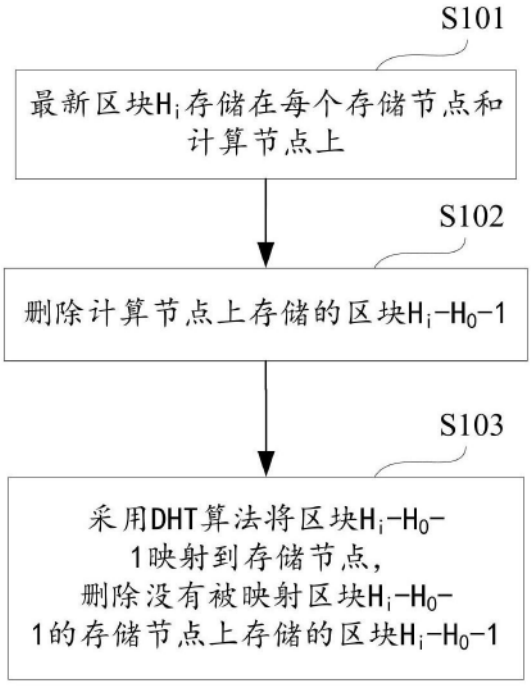

S101
最新区块H_i存储在每个存储节点和计算节点上
S102
删除计算节点上存储的区块H_i-H_0-1
S103
采用DHT算法将区块H_i-H_0-1映射到存储节点，删除没有被映射区块H_i-H_0-1的存储节点上存储的区块H_i-H_0-1

图1

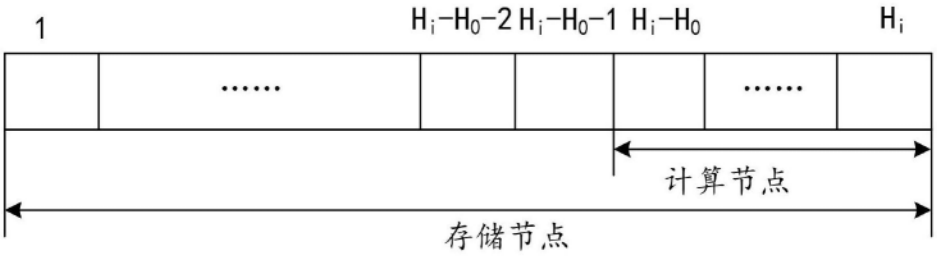

1
H_i-H_0-2 H_i-H_0-1 H_i-H_0
H_i
.....
.....
计算节点
存储节点

图2

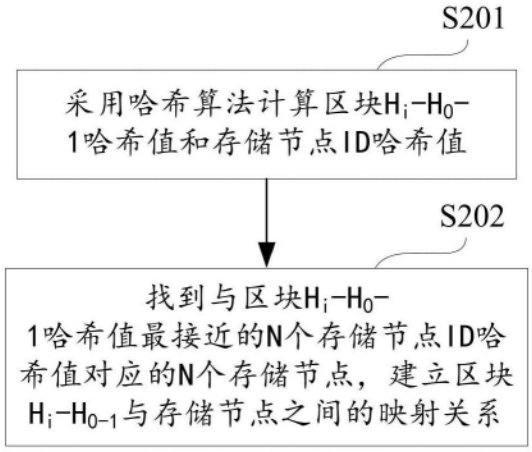

S201
采用哈希算法计算区块H_i-H_0-1哈希值和存储节点ID哈希值
S202
找到与区块H_i-H_0-1哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块H_i-H_{0-1}与存储节点之间的映射关系

图3

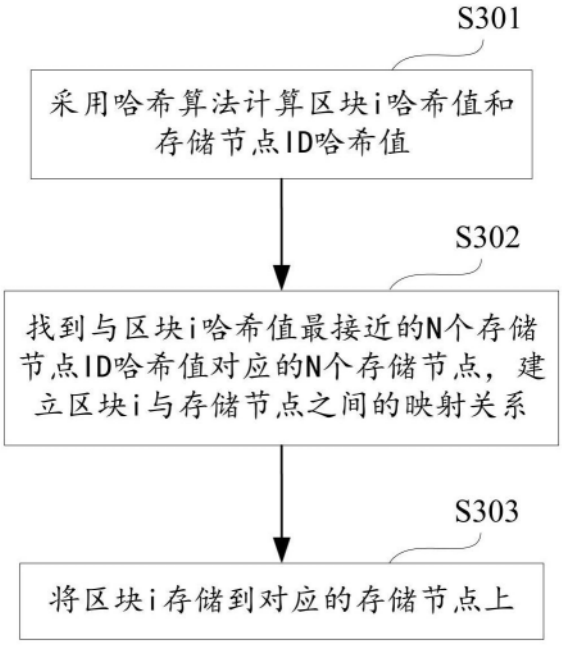

S301
采用哈希算法计算区块i哈希值和存储节点ID哈希值
S302
找到与区块i哈希值最接近的N个存储节点ID哈希值对应的N个存储节点，建立区块i与存储节点之间的映射关系
S303
将区块i存储到对应的存储节点上

图4

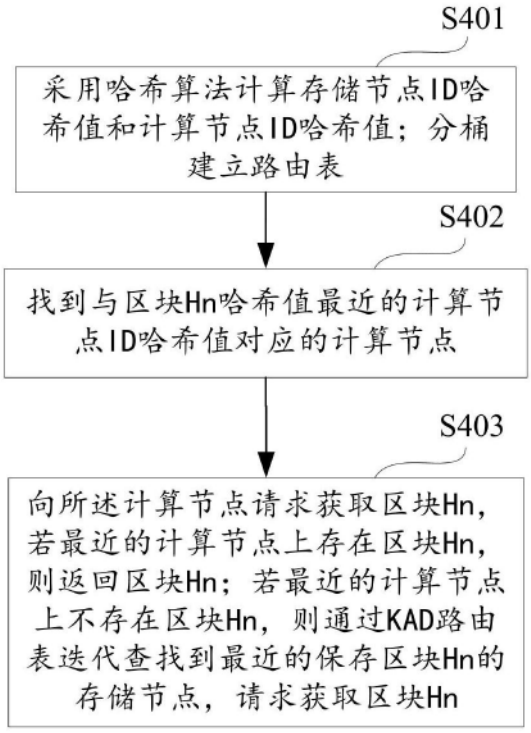

S401
采用哈希算法计算存储节点ID哈希值和计算节点ID哈希值；分桶建立路由表
S402
找到与区块Hn哈希值最近的计算节点ID哈希值对应的计算节点
S403
向所述计算节点请求获取区块Hn，若最近的计算节点上存在区块Hn，则返回区块Hn；若最近的计算节点上不存在区块Hn，则通过KAD路由表迭代查找到最近的保存区块Hn的存储节点，请求获取区块Hn

图5

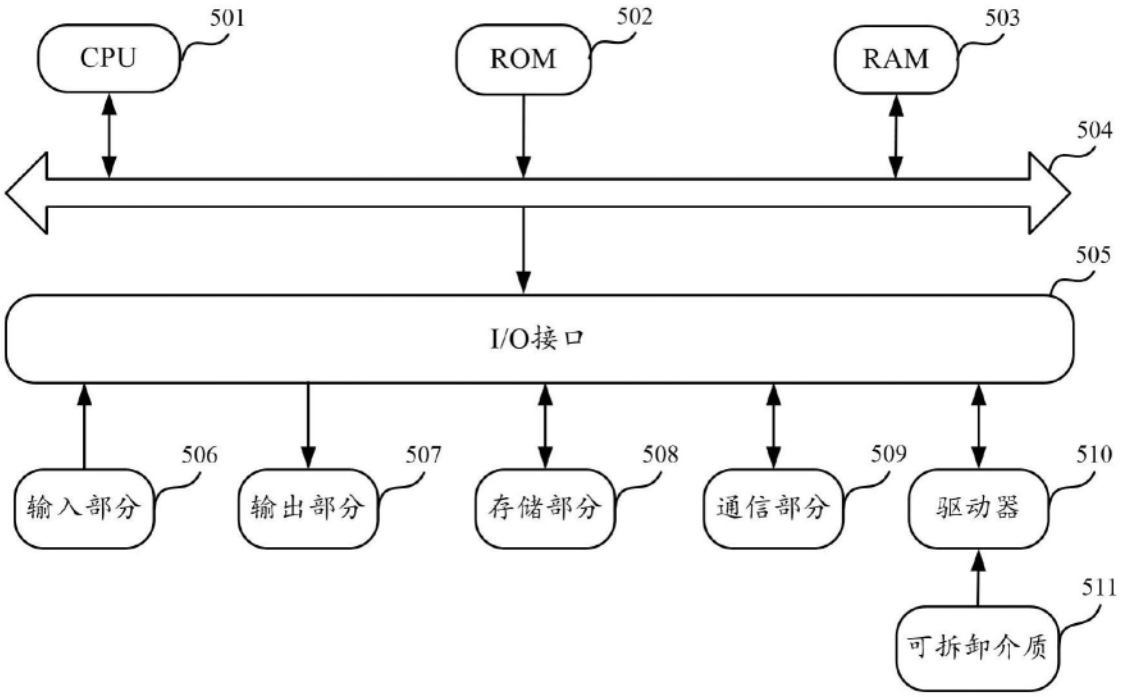

CPU
501
ROM
502
RAM
503
504
505
I/O接口
输入部分
506
输出部分
507
存储部分
508
通信部分
509
驱动器
510
可拆卸介质
511

图6

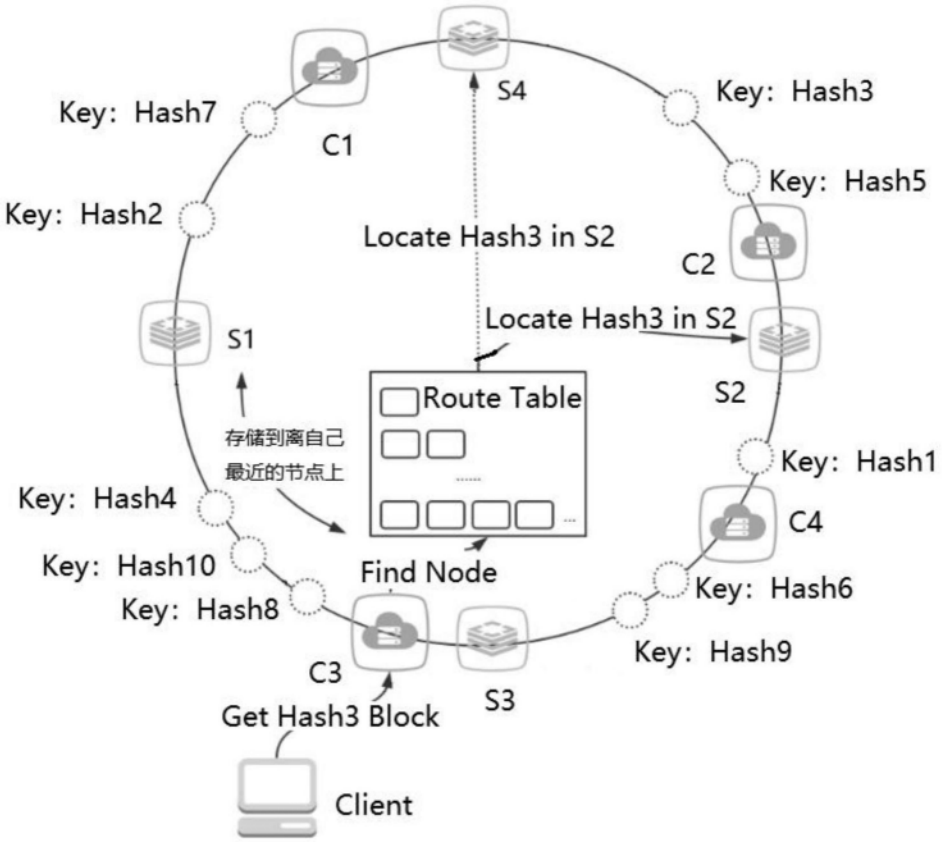

S4
Key: Hash7
C1
Key: Hash3
Key: Hash5
Key: Hash2
Locate Hash3 in S2
C2
Locate Hash3 in S2
S1
S2
Route Table
存储到离自己
最近的节点上
Key: Hash1
Key: Hash4
C4
Key: Hash10
Find Node
Key: Hash6
Key: Hash8
Key: Hash9
C3
S3
Get Hash3 Block
Client

图7