(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109359222 A
(43)申请公布日 2019.02.19

(21)申请号 201810884035.4

(22)申请日 2018.08.06

(71)申请人 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6层

(72)发明人 王志文 吴思进

(51)Int.Cl.
G06F 16/901(2019.01)
G06F 16/903(2019.01)
G06Q 40/04(2012.01)

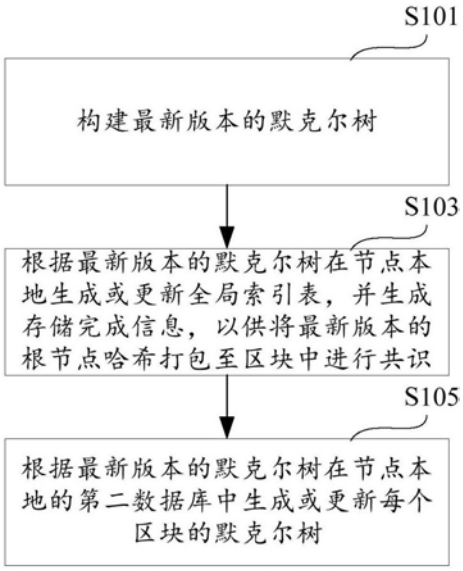
权利要求书2页 说明书6页 附图3页

(54)发明名称

数据存储方法及系统、设备和存储介质

(57)摘要

本发明提供一种数据存储方法及系统、设备和存储介质，该方法包括：在区块链网络的各节点本地存储对应于每个区块的默克尔树的全局索引表，以及每个区块的默克尔树。其中，版本的版本号为区块高度，全局索引表用于根据区块高度进行数据查询。本发明通过为默克尔树配置相对应的、以区块高度为版本号的全局索引表，为系统提供了可以直接通过区块高度查询全局索引表、通过一次读操作即可完成查询的数据查询途径，从而大幅提升了系统的数据读取性能；并进一步优化了系统在发生交易拥堵或节点崩溃时的健壮性。



CN 109359222 A

1.一种数据存储方法,其特征在于,包括:

在区块链网络的各节点本地存储对应于每个区块的默克尔树的全局索引表,以及所述每个区块的默克尔树;

其中,所述全局索引表以区块高度为版本号,用于根据所述区块高度进行数据查询。

2.根据权利要求1所述的方法,其特征在于,所述在区块链网络的各节点本地存储对应于每个区块的默克尔树的全局索引表,以及所述每个区块的默克尔树包括:

构建最新版本的默克尔树;

根据所述最新版本的默克尔树在节点本地的第一数据库中生成或更新全局索引表,并生成所述全局索引表的存储完成信息,以供所述节点中的共识单元将所述最新版本的默克尔树的根节点哈希打包至区块中进行共识;

根据所述最新版本的默克尔树在节点本地的第二数据库中生成或更新每个区块的默克尔树。

3.根据权利要求2所述的方法,其特征在于,所述根据所述最新版本的默克尔树在节点本地的第一数据库中生成或更新全局索引表,并生成所述全局索引表的存储完成信息,以供所述节点的共识单元将所述最新版本的默克尔树的根节点哈希打包至区块中进行共识之后还包括:

在所述节点崩溃后,根据所述全局索引表在所述第二数据库中恢复所述每个区块的默克尔树。

4.根据权利要求2所述的方法,其特征在于,所述构建最新版本的默克尔树包括:

通过所述全局索引表查询交易所需的业务数据;

根据所述业务数据和所述交易构建最新版本的默克尔树。

5.根据权利要求1-4任一项所述的方法,其特征在于,所述全局索引表在每个版本中只记录当前版本的默克尔树中新增或修改的叶子节点。

6.根据权利要求1-4任一项所述的方法,其特征在于,所述默克尔树为默克尔状态树,存储在键值(key-value)数据库中。

7.一种数据存储系统,配置在区块链网络的节点中,其特征在于,包括:

第一存储单元,配置用于在节点本地存储对应于每个区块的默克尔树的全局索引表;

第二存储单元,配置用于在节点本地存储所述每个区块的默克尔树;

其中,所述全局索引表以区块高度为版本号,用于根据所述区块高度进行数据查询。

8.根据权利要求7所述的系统,其特征在于,还包括:

构建单元,配置用于构建最新版本的默克尔树;

所述第一存储单元进一步配置用于根据所述最新版本的默克尔树在节点本地的第一数据库中生成或更新全局索引表,并生成所述全局索引表的存储完成信息,以供所述节点中的共识单元将所述最新版本的默克尔树的根节点哈希打包至区块中进行共识;

所述第二存储单元进一步配置用于根据所述最新版本的默克尔树在节点本地的第二数据库中生成或更新每个区块的默克尔树。

9.根据权利要求8所述的系统,其特征在于,所述第二存储单元进一步配置用于在所述节点崩溃后,根据所述全局索引表在所述第二数据库中恢复所述每个区块的默克尔树。

10.根据权利要求8所述的系统,其特征在于,所述构建单元进一步配置用于通过所述

全局索引表查询交易所需的业务数据，以及，根据所述业务数据和所述交易构建最新版本的默克尔树。

11.根据权利要求7-10任一项所述的系统，其特征在于，所述第一存储单元进一步配置用于在所述全局索引表的每个版本中只记录当前版本的默克尔树中新增或修改的叶子节点。

12.根据权利要求7-10任一项所述的系统，其特征在于，所述默克尔树为默克尔状态树，存储在键值(key-value)数据库中。

13.一种设备，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1-6中任一项所述的方法。

14.一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1-6中任一项所述的方法。

数据存储方法及系统、设备和存储介质

技术领域

[0001] 本申请涉及区块链技术领域，具体涉及一种数据存储方法及系统、设备和存储介质。

背景技术

[0002] 当前区块链系统的数据读写的解决方案中，通常会采用默克尔树(Merkle Tree)的结构，例如比特币的系统中通过默克尔树进行spv验证、以太坊的系统中通过默克尔前缀树(Merkle Patricia Tree，简称MPT)进行数据的读写，等等。

[0003] 当前利用默克尔树的结构进行数据存储的方案的缺陷在于：

[0004] 所存储的数据限制了系统的读取性能，查询一笔交易的数据需要通过多次读操作来完成。例如，对于一颗20层的默克尔树，查询一个叶子节点的数据需要进行20次读操作来完成，导致数据查询的效率仅为普通数据库的查询效率的1/20，对于每秒能完成10万次读操作的系统，每秒仅能读取5000笔交易的数据。

[0005] 更进一步地，现有方案需要在节点本地数据库中写完默克尔树的数据后才能进行区块的共识，导致在交易数量剧增时容易发生交易拥堵。

[0006] 此外，现有方案在节点本地数据库中写默克尔树的数据时若发生崩溃，会导致无法生成区块。

发明内容

[0007] 鉴于现有技术中的上述缺陷或不足，期望提供一种通过优化所存储的数据结构提升系统读取性能的数据存储方法及系统、设备和存储介质，并进一步期望优化系统在发生交易拥堵或节点崩溃时的健壮性。

[0008] 第一方面，本发明提供一种数据存储方法，包括：

[0009] 在区块链网络的各节点本地存储对应于每个区块的默克尔树的全局索引表，以及每个区块的默克尔树。

[0010] 其中，该全局索引表以区块高度为版本号，用于根据区块高度进行数据查询。

[0011] 第二方面，本发明提供一种数据存储系统，配置在区块链网络的节点中，包括第一存储单元和第二存储单元。

[0012] 第一存储单元配置用于在节点本地存储对应于每个区块的默克尔树的全局索引表；

[0013] 第二存储单元配置用于在节点本地存储每个区块的默克尔树。

[0014] 其中，该全局索引表以区块高度为版本号，用于根据区块高度进行数据查询。

[0015] 第三方面，本发明还提供一种设备，包括一个或多个处理器和存储器，其中存储器包含可由该一个或多个处理器执行的指令以使得该一个或多个处理器执行根据本发明各实施例提供的数据存储方法。

[0016] 第四方面，本发明还提供一种存储有计算机程序的存储介质，该计算机程序使计

算机执行根据本发明各实施例提供的数据存储方法。

[0017] 本发明诸多实施例提供的数据存储方法及系统、设备和存储介质通过为默克尔树配置相对应的、以区块高度为版本号的全局索引表，为系统提供了可以直接通过区块高度查询全局索引表、通过一次读操作即可完成查询的数据查询途径，从而大幅提升了系统的数据读取性能；

[0018] 本发明一些实施例提供的数据存储方法及系统、设备和存储介质进一步通过在存储完全局索引表的数据后即打包区块共识，并异步存储对应的默克尔树的数据，从而可以大幅缓解交易数量剧增时的区块拥堵问题，同时还降低了存储数据时发生节点崩溃导致无法生成区块的风险；

[0019] 本发明一些实施例提供的数据存储方法及系统、设备和存储介质进一步通过在节点崩溃时通过全局索引表恢复每个区块的默克尔树，从而无需通过向其它节点同步数据进行恢复，缓解了节点间的通信压力；

[0020] 本发明一些实施例提供的数据存储方法及系统、设备和存储介质进一步通过查询全局索引表得到的数据构建最新版本的默克尔树，并将根节点哈希打包共识，实现了无需对全局索引表进行共识即可确保全局索引表的安全性；

[0021] 本发明一些实施例提供的数据存储方法及系统、设备和存储介质进一步通过在全局索引表中只记录默克尔树新增或修改的叶子节点，从而大幅降低了为多版本默克尔树配置全局索引表所带来的数据冗余。

附图说明

[0022] 通过阅读参照以下附图所作的对非限制性实施例所作的详细描述，本申请的其它特征、目的和优点将会变得更明显：

[0023] 图1为本发明一实施例中多版本默克尔树与全局索引表的对应关系示意图。

[0024] 图2为本发明一优选实施例提供的一种数据存储方法的流程图。

[0025] 图3为图2所示方法的一种优选实施方式的流程图。

[0026] 图4为本发明一实施例提供的一种数据存储系统的结构示意图。

[0027] 图5为图4所示系统的一种优选实施方式的结构示意图。

[0028] 图6为本发明一实施例提供的一种设备的结构示意图。

具体实施方式

[0029] 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。

[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] 在本实施例中，本发明提供一种数据存储方法，包括：

[0032] S10：在区块链网络的各节点本地存储对应于每个区块的默克尔树的全局索引表，以及每个区块的默克尔树。

[0033] 其中，该全局索引表以区块高度为版本号，用于根据区块高度进行数据查询。

[0034] 具体地，在本实施例中，上述默克尔树具体配置为默克尔状态树(Merkle State Tree)，高度配置为20层，存储在节点本地的键值(key-value)数据库中，全局索引表存储在节点本地的另一数据库中；在更多实施例中，可根据实际需求将默克尔树配置为本领域常用的其它不同类型的默克尔树、不同高度、存储在节点本地的本领域常用的其它不同类型的数据库中，还可以将默克尔树和全局索引表存储在同一数据库中，均可实现相同的技术效果。

[0035] 图1为本发明一实施例中多版本默克尔树与全局索引表的对应关系示意图。

[0036] 如图1所示，以高度为3层的默克尔二叉状态树为例，在A节点中，当区块高度H＝9的区块打包了一笔K_4账户进行支付(或收款)、余额由V_4变为V_4'的交易时：

[0037] 在步骤S10中，一方面在A节点本地的kv数据库中新增一个叶子节点H_4'，并新增相对应的索引节点H_{34}'和根节点H_{1234}'，其中，H_4'为(K_4,V_4')的哈希值；

[0038] 另一方面，在A节点本地的另一数据库中更新全局索引表，新增版本号为9的记录(K_4,V_4')。

[0039] 上述默克尔树和全局索引表的存储可以同步进行，也可以异步进行，以下将结合图2-4详细介绍异步进行存储的优选方案。

[0040] 在本实施例中，被替换的叶子节点和索引节点将保留在数据库中，以便于在区块链发生分叉等场景下进行数据回滚；在另一些实施例中，还可以根据实际配置的不同方案将被替换的叶子节点和索引节点配置为直接删除或经过若干个区块高度后进行清理，可实现相同的技术效果。

[0041] 显而易见地，在通过上述存储方法为每个区块的默克尔树配置相对应的全局索引表后，可以通过区块高度直接在全局索引表中查询所需的数据，而无需再根据默克尔树的根节点哈希对默克尔树进行逐层的读取和查询。

[0042] 如图1所示，在本实施例中，全局索引表配置为在每个版本中只记录当前版本的默克尔树中新增或修改的叶子节点，而不再重复记录未发生修改的叶子节点；上述查询在所查找的区块高度若未查找到目标数据，可以通过数据库的指针进行滑动查找；

[0043] 在另一实施例中，还可以将全局索引表配置为在每个版本中记录当前版本的默克尔树的所有叶子节点；在该实施例中，上述查询可以通过区块高度直接查找到目标数据。

[0044] 上述实施例通过为默克尔树配置相对应的、以区块高度为版本号的全局索引表，为系统提供了可以直接通过区块高度查询全局索引表、通过一次读操作即可完成查询的数据查询途径，从而大幅提升了系统的数据读取性能；并进一步通过在全局索引表中只记录默克尔树新增或修改的叶子节点，从而大幅降低了为多版本默克尔树配置全局索引表所带来的数据冗余。

[0045] 图2为本发明一优选实施例提供的一种数据存储方法的流程图。

[0046] 如图2所示，在一优选实施例中，步骤S10包括：

[0047] S101：构建最新版本的默克尔树；

[0048] S103：根据最新版本的默克尔树在节点本地的第一数据库中生成或更新全局索引表，并生成全局索引表的存储完成信息，以供节点中的共识单元将最新版本的默克尔树的根节点哈希打包至区块中进行共识；

[0049] S105：根据最新版本的默克尔树在节点本地的第二数据库中生成或更新每个区块

的默克尔树。

[0050] 优选地，步骤S101包括：通过全局索引表查询交易所需的业务数据；以及，根据业务数据和交易构建最新版本的默克尔树。

[0051] 具体地，在步骤S101中，当交易在默克尔树中新增叶子节点时，例如区块高度H＝22时，用户甲开设新账户K_{11}，无需进行余额V_{11}的查询，直接在内存中新增叶子节点和对应的索引节点；

[0052] 当交易在默克尔树中修改叶子节点时，例如区块高度H＝23时，用户乙的账户K_4与用户丙的账户K_6发生交易，则需要通过上述全局索引表查询账户K_4的余额V_4，以及账户K_6的余额V_6；再根据V_4和V_6以及交易的数额计算出交易后账户K_4的余额V_4'，以及账户K_6的余额V_6'，再在内存中新增对应的叶子节点和索引节点，完成最新版本的默克尔树的构建。

[0053] 在步骤S103中，根据步骤S101在内存中所构建的最新版本的默克尔树在节点本地的第一数据库中生成或更新全局索引表，并在第一数据库写完全局索引表后生成存储完成信息，通知节点中的共识单元可以进行区块的打包。共识单元在收到通知后将最新版本的默克尔树的根节点哈希打包至区块中，再将打包的区块广播进行共识。

[0054] 其中，若节点本地存储的全局索引表被篡改导致查询到的交易前的余额不正确，则计算出的交易后的余额、最新版本的默克尔树的根节点哈希均不正确，从而导致打包的区块无法通过共识；因此，该方法可以保障节点本地存储的全局索引表无需通过共识，但只有根据正确的全局索引表所生成的区块可以通过共识。

[0055] 在步骤S105中，当节点收到的需要打包共识的交易数量较少时，步骤S105的数据存储可以与步骤S103同步进行；而当交易数量较多，需要存储的默克尔树的数据量超过系统的存储性能瓶颈时，步骤S105的数据存储可以与步骤S103异步进行。

[0056] 例如系统每秒可以存储5000笔交易的默克尔树的数据，但在高峰期节点每秒收到20000笔交易，由于全局索引表的数据量较小，可以实时完成步骤S103的数据存储并打包区块进行共识，而在高峰期过后再异步进行步骤S105的数据存储。

[0057] 同时由于全局索引表的数据量较小、存储耗时较短，在全局索引表存储完成之前发生节点崩溃的概率较低，从而还可以大幅降低存储数据时发生节点崩溃导致无法生成区块的风险。

[0058] 上述实施例进一步通过在存储完全局索引表的数据后即打包区块共识，并异步存储对应的默克尔树的数据，从而可以大幅缓解交易数量剧增时的区块拥堵问题，同时还降低了存储数据时发生节点崩溃导致无法生成区块的风险；并进一步通过查询全局索引表得到的数据构建最新版本的默克尔树，并将根节点哈希打包共识，实现了无需对全局索引表进行共识即可确保全局索引表的安全性。

[0059] 图3为图2所示方法的一种优选实施方式的流程图。如图3所示，在一优选实施例中，步骤S103之后还包括：

[0060] S107：在节点崩溃后，根据全局索引表在第二数据库中恢复每个区块的默克尔树。

[0061] 具体地，当全局索引表的数据存储完成之后，在默克尔树的数据存储之前发生节点崩溃，或在默克尔树的数据存储进行过程中发生节点崩溃，均可通过全局索引表在第二数据库中恢复每个区块的默克尔树。

[0062] 上述实施例进一步通过在节点崩溃时通过全局索引表恢复每个区块的默克尔树，

从而无需通过向其它节点同步数据进行恢复，缓解了节点间的通信压力。

[0063] 图4为本发明一实施例提供的一种数据存储系统的结构示意图。图4所示的系统可对应执行上述包括步骤S10的方法。

[0064] 如图4所示，在本实施例中，本发明提供一种数据存储系统10，配置在区块链网络的节点20中，包括第一存储单元101和第二存储单元103。

[0065] 第一存储单元101配置用于在节点本地存储对应于每个区块的默克尔树的全局索引表；

[0066] 第二存储单元103配置用于在节点本地存储每个区块的默克尔树。

[0067] 其中，该全局索引表以区块高度为版本号，用于根据区块高度进行数据查询。

[0068] 图4所示系统的数据存储原理可参照上述包括步骤S10的方法，此处不再赘述。

[0069] 图5为图4所示系统的一种优选实施方式的结构示意图。图5所示的系统可对应执行图2-3所示的方法。

[0070] 如图5所示，在一优选实施例中，数据存储系统10还包括构建单元105。

[0071] 构建单元105配置用于构建最新版本的默克尔树；

[0072] 第一存储单元101进一步配置用于根据最新版本的默克尔树在节点20本地的第一数据库中生成或更新全局索引表，并生成全局索引表的存储完成信息，以供节点20中的共识单元将最新版本的默克尔树的根节点哈希打包至区块中进行共识；

[0073] 第二存储单元103进一步配置用于根据最新版本的默克尔树在节点20本地的第二数据库中生成或更新每个区块的默克尔树。

[0074] 优选地，第二存储单元103进一步配置用于在节点20崩溃后，根据全局索引表在第二数据库中恢复每个区块的默克尔树。

[0075] 优选地，构建单元105进一步配置用于通过全局索引表查询交易所需的业务数据，以及，根据业务数据和交易构建最新版本的默克尔树。

[0076] 优选地，第一存储单元101进一步配置用于在全局索引表的每个版本中只记录当前版本的默克尔树中新增或修改的叶子节点。

[0077] 图5所示系统的数据存储原理可参照图2-3所示的方法，此处不再赘述。

[0078] 图6为本发明一实施例提供的一种设备的结构示意图。

[0079] 如图6所示，作为另一方面，本申请还提供了一种设备600，包括一个或多个中央处理单元(CPU)601，其可以根据存储在只读存储器(ROM)602中的程序或者从存储部分608加载到随机访问存储器(RAM)603中的程序而执行各种适当的动作和处理。在RAM603中，还存储有设备600操作所需的各种程序和数据。CPU601、ROM602以及RAM603通过总线604彼此相连。输入/输出(I/O)接口605也连接至总线604。

[0080] 以下部件连接至I/O接口605：包括键盘、鼠标等的输入部分606；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607；包括硬盘等的存储部分608；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609。通信部分609经由诸如因特网的网络执行通信处理。驱动器610也根据需要连接至I/O接口605。可拆卸介质611，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610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608。

[0081] 特别地，根据本公开的实施例，上述任一实施例描述的数据存储方法可以被实现

为计算机软件程序。例如，本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数据存储方法的程序代码。在这样的实施例中，该计算机程序可以通过通信部分609从网络上被下载和安装，和/或从可拆卸介质611被安装。

[0082] 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数据存储方法。

[0083] 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。

[0084] 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。

[0085] 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。

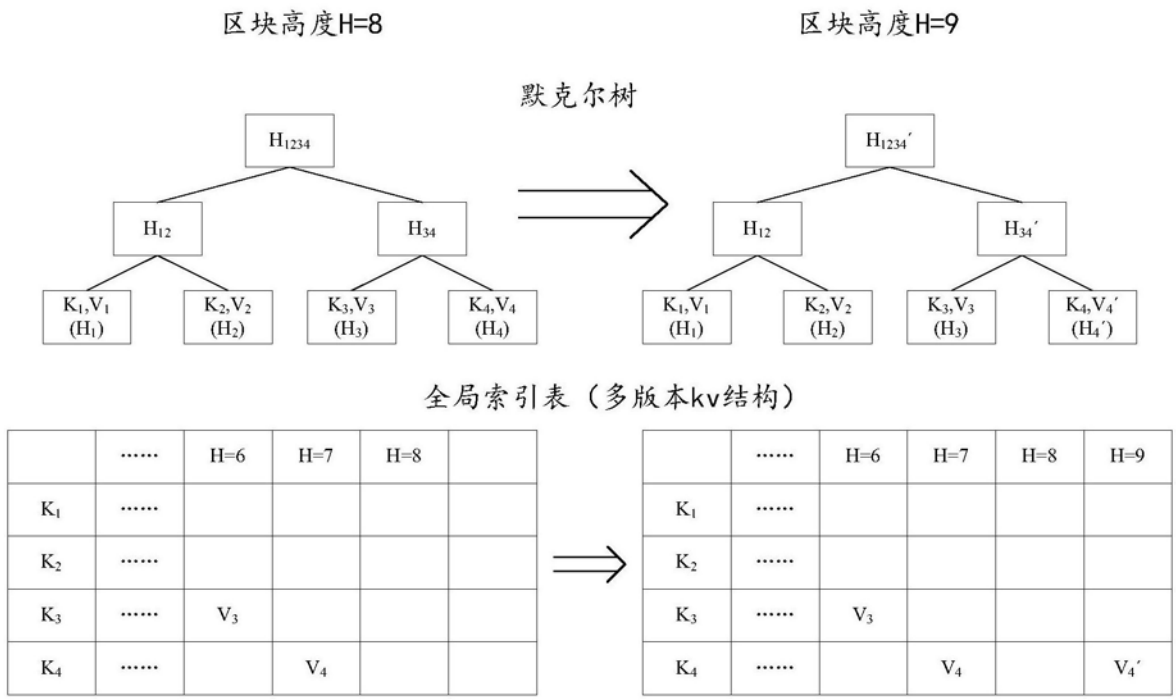

区块高度H=8
区块高度H=9
默克尔树
H_{1234}
H_{12}
H_{34}
K_1,V_1 (H_1)
K_2,V_2 (H_2)
K_3,V_3 (H_3)
K_4,V_4 (H_4)
H_{1234}'
H_{34}'
K_4,V_4' (H_4')
全局索引表（多版本kv结构）
H=6
H=7
H=8
H=9
K_1
K_2
K_3
K_4
V_3
V_4
V_4'

图1

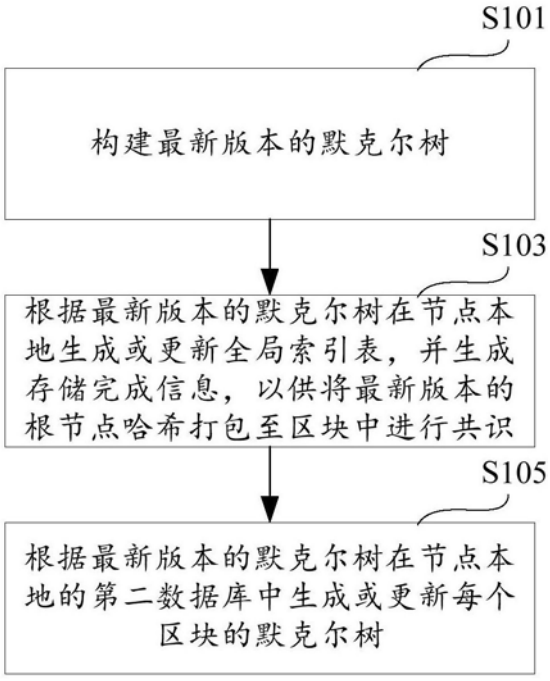

S101
构建最新版本的默克尔树
S103
根据最新版本的默克尔树在节点本地生成或更新全局索引表，并生成存储完成信息，以供将最新版本的根节点哈希打包至区块中进行共识
S105
根据最新版本的默克尔树在节点本地的第二数据库中生成或更新每个区块的默克尔树

图2

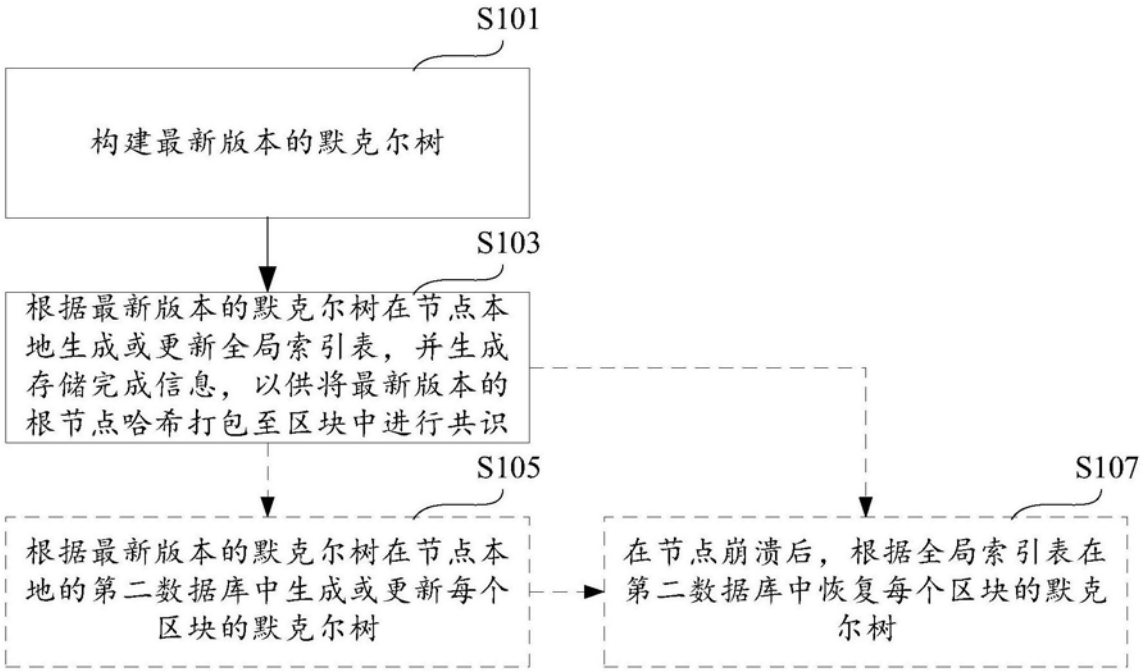

S101
构建最新版本的默克尔树
S103
根据最新版本的默克尔树在节点本地生成或更新全局索引表，并生成存储完成信息，以供将最新版本的根节点哈希打包至区块中进行共识
S105
根据最新版本的默克尔树在节点本地的第二数据库中生成或更新每个区块的默克尔树
S107
在节点崩溃后，根据全局索引表在第二数据库中恢复每个区块的默克尔树

图3

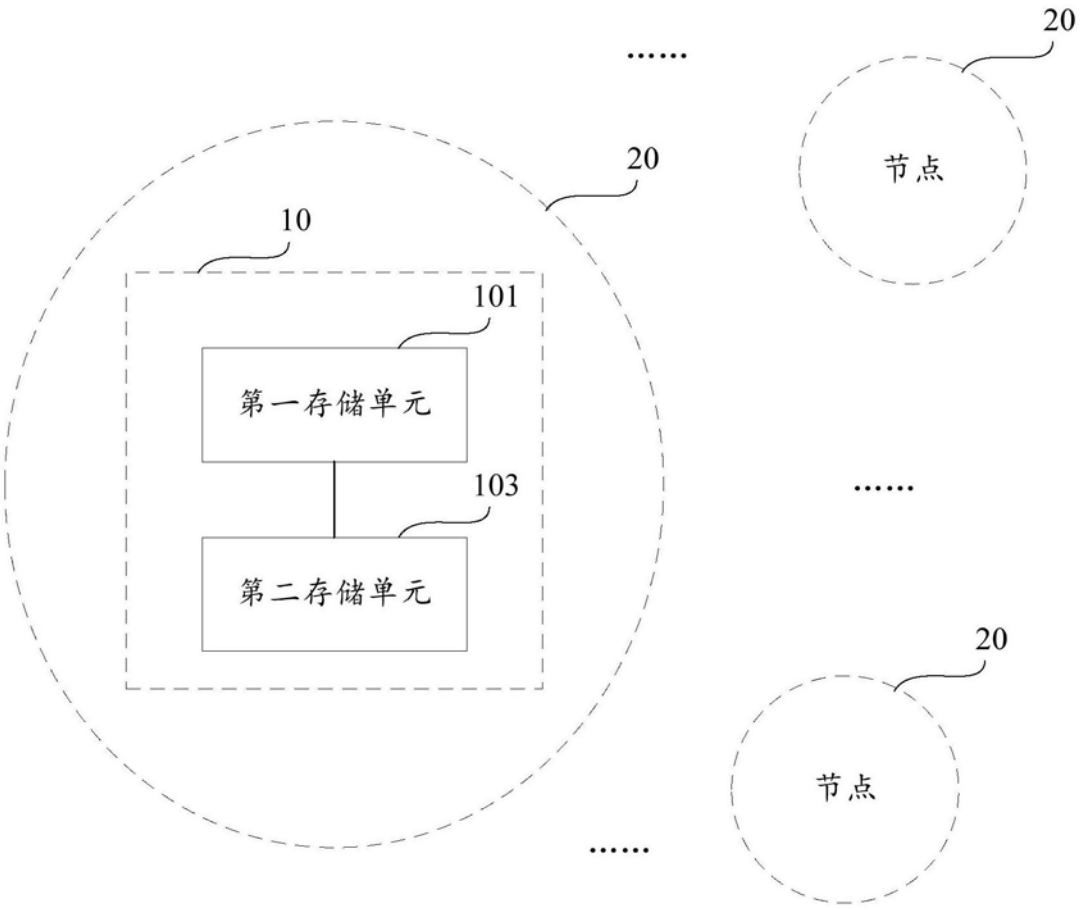

20
节点
20
10
101
第一存储单元
103
第二存储单元
20
节点

图4

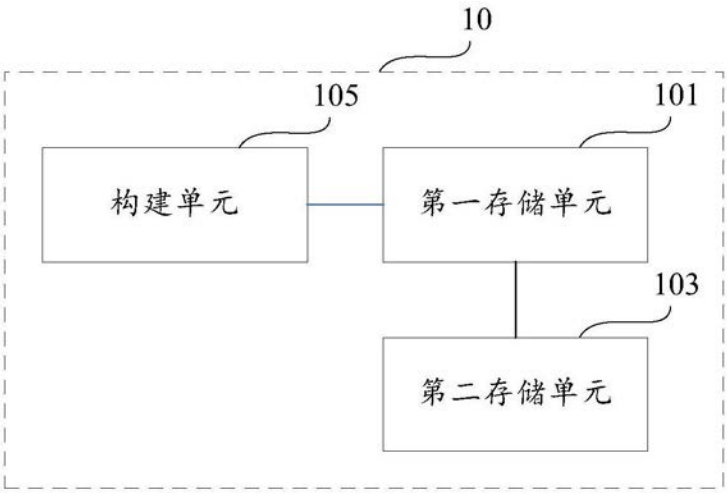

10
105
101
构建单元
第一存储单元
103
第二存储单元

图5

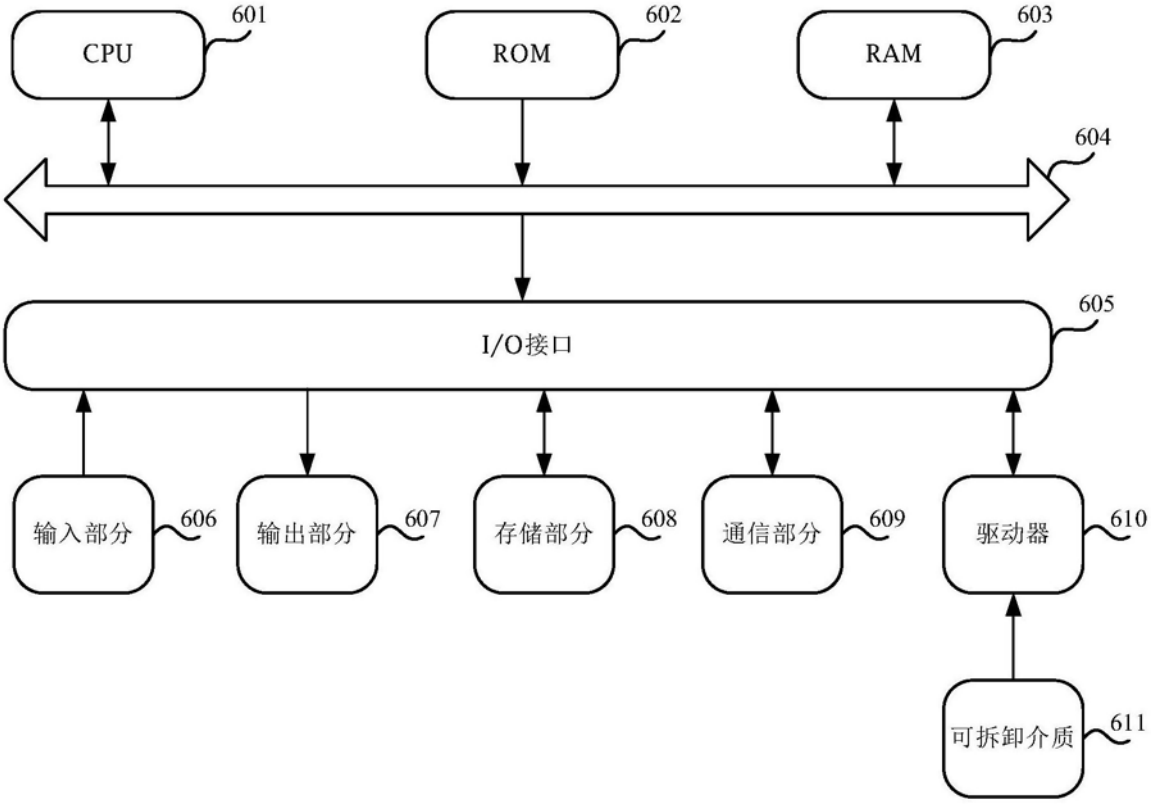

601
CPU
602
ROM
603
RAM
604
605
I/O接口
输入部分
606
输出部分
607
存储部分
608
通信部分
609
驱动器
610
可拆卸介质
611

图6