(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 110334154 B
(45)授权公告日 2020.07.21

(21)申请号 201910579371.2

(22)申请日 2019.06.28

(65)同一申请的已公布的文献号
申请公布号 CN 110334154 A

(43)申请公布日 2019.10.15

(73)专利权人 阿里巴巴集团控股有限公司
地址 英属开曼群岛大开曼资本大厦一座四层847号邮箱

(72)发明人 陆钟豪 卓海振 俞本权

(74)专利代理机构 北京博思佳知识产权代理有限公司 11415
代理人 周嗣勇

(51)Int.Cl.
G06F 16/27(2019.01)
G06F 16/28(2019.01)
G06F 16/22(2019.01)

(56)对比文件
CN 109684333 A,2019.04.26,
CN 108197226 A,2018.06.22,
US 2017264684 A1,2017.09.14,
CN 109359222 A,2019.02.19,
CN 109919756 A,2019.06.21,
CN 109726229 A,2019.05.07,
CN 109903049 A,2019.06.18,
CN 109165224 A,2019.01.08,
US 7647329 B1,2010.01.12,
CN 109144414 A,2019.01.04,
Jingqiang Liu等.A Data Storage Method Based on Blockchain for Decentralization DNS.《2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)》.2018,第189-196页.
Sanat Ghoshal等.Exploiting Block-Chain Data Structure for Auditorless Auditing on Cloud Data.《International Conference on Information Systems Security》.2016,第359-371页.
邵奇峰等.区块链技术:架构及进展.《计算机学报》.2018,第41卷(第5期),第969-988页.

审查员 丁雪龙

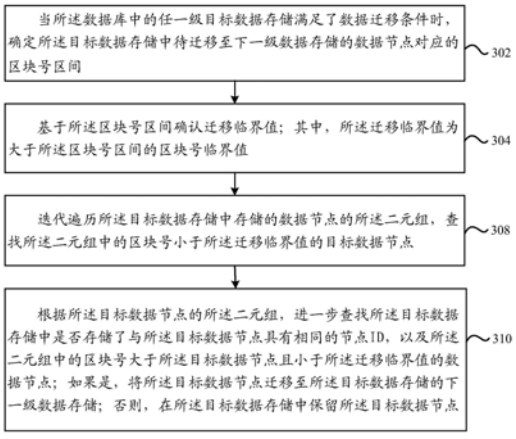
权利要求书3页 说明书18页 附图4页

(54)发明名称

基于区块链的分级存储方法及装置、电子设备

(57)摘要

一种基于区块链的分级存储方法,区块链的账户状态数据组织成的Merkle状态树上的数据节点,以Key-Value键值对的形式存储在数据库中;key为节点ID和区块号组成的二元组;数据库包括多级数据存储;区块号指示数据节点发生数据更新时所在的区块;包括:当任一级目标数据存储满足了数据迁移条件时,确定目标数据存储中待迁移的区块号区间;基于区块号区间确定迁移临界值;迭代遍历目标数据存储中存储的二元组,查找二元组中的区块号小于迁移临界值的目标数据节点;查找目标数据存储中是否存储了与目标数据节点具有相同的节点ID,以及二元组中的区块号大于目标数据节点且小于迁移临界值的数据节点;如果是,将目标数据节点迁移至下一级数据存储。



CN 110334154 B

1.一种基于区块链的分级存储方法,所述区块链的账户状态数据组织成的Merkle状态树上的数据节点,以Key-Value键值对的形式存储在数据库中;其中,所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组;所述数据库包括多级数据存储;所述区块号指示所述数据节点发生数据更新时所在的区块;所述方法包括:

当所述数据库中的任一级目标数据存储满足了数据迁移条件时,确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间;

基于所述区块号区间确定迁移临界值;其中,所述迁移临界值为大于所述区块号区间的区块号临界值;

迭代遍历所述目标数据存储中存储的数据节点的所述二元组,查找所述二元组中的区块号小于所述迁移临界值的目标数据节点;

根据所述目标数据节点的所述二元组,进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID,以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点;如果是,将所述目标数据节点迁移至所述目标数据存储的下一级数据存储;否则,在所述目标数据存储中保留所述目标数据节点。

2.根据权利要求1所述的方法,所述基于所述区块号区间确定迁移临界值,包括:

如果所述区块号区间的右区间为开区间,将所述区块号区间的右端点值,确定为所述迁移临界值;

如果所述区块号区间的右区间为闭区间,将所述区块号区间的右端点值与所述区块链的区块号递增步长的和确定为所述迁移临界值。

3.根据权利要求1所述的方法,所述查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID,以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点,包括:

查找所述目标数据存储中存储的与所述目标数据节点具有相同的节点ID,并且所述二元组中的区块号为小于所述迁移临界值的区块号中的最大区块号的数据节点;

进一步确定查找到的该数据节点的所述二元组中的区块号,是否大于所述目标数据节点的所述二元组中的区块号。

4.根据权利要求1所述的方法,所述方法还包括:

确定最新区块的Merkle状态树上发生数据更新的数据节点;为所述最新区块的Merkle状态树上发生数据更新的数据节点构建Key-Value键值对,并将所述Key-Value键值对存储至所述数据库中的最高级数据存储;

其中,所述Key-Value键值对的key为所述最新区块的区块号和所述数据节点的节点ID组成的二元组;所述Key-Value键值对的Value为所述数据节点包含的数据内容。

5.根据权利要求1或4所述的方法,所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种;所述数据节点的节点ID,为所述Merkle树的根节点到该数据节点的路径对应的字符前缀。

6.根据权利要求5所述的方法,所述Merkle状态树为Merkle Patricia Tree状态树。

7.根据权利要求1所述的方法,所述数据库为LevelDB数据库;或者基于LevelDB架构的数据库。

8.根据权利要求7所述的方法,所述数据库为基于LevelDB架构的Rocksdb数据库。

9.根据权利要求1所述的方法,所述多级数据存储对应的存储介质的读写性能,存在性能差异;其中,级数高的数据存储对应的存储介质的读写性能,高于级数低的数据存储对应的存储介质的读写性能。

10.一种基于区块链的分级存储装置,所述区块链的账户状态数据组织成的Merkle状态树上的数据节点,以Key-Value键值对的形式存储在数据库中;其中,所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组;所述数据库包括多级数据存储;所述区块号指示所述数据节点发生数据更新时所在的区块;所述装置包括:

确定模块,当所述数据库中的任一级目标数据存储满足了数据迁移条件时,确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间;基于所述区块号区间确定迁移临界值;其中,所述迁移临界值为大于所述区块号区间的区块号临界值;

查找模块,迭代遍历所述目标数据存储中存储的数据节点的所述二元组,查找所述二元组中的区块号小于所述迁移临界值的目标数据节点;

迁移模块,根据所述目标数据节点的所述二元组,进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID,以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点;如果是,将所述目标数据节点迁移至所述目标数据存储的下一级数据存储;否则,在所述目标数据存储中保留所述目标数据节点。

11.根据权利要求10所述的装置,所述确定模块:

如果所述区块号区间的右区间为开区间,将所述区块号区间的右端点值,确定为所述迁移临界值;

如果所述区块号区间的右区间为闭区间,将所述区块号区间的右端点值与所述区块链的区块号递增步长的和确定为所述迁移临界值。

12.根据权利要求10所述的装置,所述迁移模块进一步:

查找所述目标数据存储中存储的与所述目标数据节点具有相同的节点ID,并且所述二元组中的区块号为小于所述迁移临界值的区块号中的最大区块号的数据节点;进一步确定查找到的该数据节点的所述二元组中的区块号,是否大于所述目标数据节点的所述二元组中的区块号。

13.根据权利要求10所述的装置,所述装置还包括:

存储模块,确定最新区块的Merkle状态树上发生数据更新的数据节点;为所述最新区块的Merkle状态树上发生数据更新的数据节点构建Key-Value键值对,并将所述Key-Value键值对存储至所述数据库中的最高级数据存储;

其中,所述Key-Value键值对的key为所述最新区块的区块号和所述数据节点的节点ID组成的二元组;所述Key-Value键值对的Value为所述数据节点包含的数据内容。

14.根据权利要求10或13所述的装置,所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种;所述数据节点的节点ID,为所述Merkle树的根节点到该数据节点的路径对应的字符前缀。

15.根据权利要求14所述的装置,所述Merkle状态树为Merkle Patricia Tree状态树。

16.根据权利要求10所述的装置,所述数据库为LevelDB数据库;或者基于LevelDB架构

的数据库。

17.根据权利要求16所述的装置,所述数据库为基于LevelDB架构的Rocksdb数据库。

18.根据权利要求10所述的装置,所述多级数据存储对应的存储介质的读写性能,存在性能差异;其中,级数高的数据存储对应的存储介质的读写性能,高于级数低的数据存储对应的存储介质的读写性能。

19.一种电子设备,包括:

处理器;

用于存储处理器可执行指令的存储器;

其中,所述处理器通过运行所述可执行指令以实现如权利要求1-9中任一项所述的方法。

20.一种计算机可读存储介质,其上存储有计算机指令,其特征在于,该指令被处理器执行时实现如权利要求1-9中任一项所述方法的步骤。

基于区块链的分级存储方法及装置、电子设备

技术领域

[0001] 本说明书一个或多个实施例涉及区块链技术领域，尤其涉及一种基于区块链的分级存储方法及装置、电子设备。

背景技术

[0002] 区块链技术，也被称之为分布式账本技术，是一种由若干台计算设备共同参与“记账”，共同维护一份完整的分布式数据库的新兴技术。由于区块链技术具有去中心化、公开透明、每台计算设备可以参与数据库记录、并且各计算设备之间可以快速的进行数据同步的特性，使得区块链技术已在众多的领域中广泛的进行应用。

发明内容

[0003] 本说明书提出一种基于区块链的分级存储方法，所述区块链的账户状态数据组织成的Merkle状态树上的数据节点，以Key-Value键值对的形式存储在数据库中；其中，所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组；所述数据库包括多级数据存储；所述区块号指示所述数据节点发生数据更新时所在的区块；所述方法包括：

[0004] 当所述数据库中的任一级目标数据存储满足了数据迁移条件时，确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间；

[0005] 基于所述区块号区间确定迁移临界值；其中，所述迁移临界值为大于所述区块号区间的区块号临界值；

[0006] 迭代遍历所述目标数据存储中存储的数据节点的所述二元组，查找所述二元组中的区块号小于所述迁移临界值的目标数据节点；

[0007] 根据所述目标数据节点的所述二元组，进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID，以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点；如果是，将所述目标数据节点迁移至所述目标数据存储的下一级数据存储；否则，在所述目标数据存储中保留所述目标数据节点。

[0008] 可选的，所述基于所述区块号区间确定迁移临界值，包括：

[0009] 如果所述区块号区间的右区间为开区间，将所述区块号区间的右端点值，确定为所述迁移临界值；

[0010] 如果所述区块号区间的右区间为闭区间，将所述区块号区间的右端点值与所述区块链的区块号递增步长的和确定为所述迁移临界值。

[0011] 可选的，所述查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID，以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点，包括：

[0012] 查找所述目标数据存储中存储的与所述目标数据节点具有相同的节点ID，并且所述二元组中的区块号为小于所述迁移临界值的区块号中的最大区块号的数据节点；

[0013] 进一步确定查找到的该数据节点的所述二元组中的区块号,是否大于所述目标数据节点的所述二元组中的区块号。

[0014] 可选的,所述方法还包括:

[0015] 确定最新区块的Merkle状态树上发生数据更新的数据节点;为所述最新区块的Merkle状态树上发生数据更新的数据节点构建Key-Value键值对,并将所述Key-Value键值对存储至所述数据库中的最高级数据存储;

[0016] 其中,所述Key-Value键值对的key为所述最新区块的区块号和所述数据节点的节点ID组成的二元组;所述Key-Value键值对的Value为所述数据节点包含的数据内容。

[0017] 可选的,所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种;所述数据节点的节点ID,为所述Merkle树的根节点到该数据节点的路径对应的字符前缀。

[0018] 可选的,所述Merkle状态树为Merkle Patricia Tree状态树。

[0019] 可选的,所述数据库为LevelDB数据库;或者基于LevelDB架构的数据库;

[0020] 可选的,所述数据库为基于LevelDB架构的Rocksdb数据库。

[0021] 可选的,所述多级数据存储对应的存储介质的读写性能,存在性能差异;其中,级数高的数据存储对应的存储介质的读写性能,高于级数低的数据存储对应的存储介质的读写性能。

[0022] 本说明书还提出一种基于区块链的分级存储装置,所述区块链的账户状态数据组织成的Merkle状态树上的数据节点,以Key-Value键值对的形式存储在数据库中;其中,所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组;所述数据库包括多级数据存储;所述区块号指示所述数据节点发生数据更新时所在的区块;所述装置包括:

[0023] 确定模块,当所述数据库中的任一级目标数据存储满足了数据迁移条件时,确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间;基于所述区块号区间确定迁移临界值;其中,所述迁移临界值为大于所述区块号区间的区块号临界值;

[0024] 查找模块,迭代遍历所述目标数据存储中存储的数据节点的所述二元组,查找所述二元组中的区块号小于所述迁移临界值的目标数据节点;

[0025] 迁移模块,根据所述目标数据节点的所述二元组,进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID,以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点;如果是,将所述目标数据节点迁移至所述目标数据存储的下一级数据存储;否则,在所述目标数据存储中保留所述目标数据节点。

[0026] 可选的,所述确定模块:

[0027] 如果所述区块号区间的右区间为开区间,将所述区块号区间的右端点值,确定为所述迁移临界值;

[0028] 如果所述区块号区间的右区间为闭区间,将所述区块号区间的右端点值与所述区块链的区块号递增步长的和确定为所述迁移临界值。

[0029] 可选的,所述迁移模块进一步:

[0030] 查找所述目标数据存储中存储的与所述目标数据节点具有相同的节点ID,并且所述二元组中的区块号为小于所述迁移临界值的区块号中的最大区块号的数据节点;进一步

确定查找到的该数据节点的所述二元组中的区块号，是否大于所述目标数据节点的所述二元组中的区块号。

[0031] 可选的，所述装置还包括：

[0032] 存储模块，确定最新区块的Merkle状态树上发生数据更新的数据节点；为所述最新区块的Merkle状态树上发生数据更新的数据节点构建Key-Value键值对，并将所述Key-Value键值对存储至所述数据库中的最高级数据存储；

[0033] 其中，所述Key-Value键值对的key为所述最新区块的区块号和所述数据节点的节点ID组成的二元组；所述Key-Value键值对的Value为所述数据节点包含的数据内容。

[0034] 可选的，所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种；所述数据节点的节点ID，为所述Merkle树的根节点到该数据节点的路径对应的字符前缀。

[0035] 可选的，所述Merkle状态树为Merkle Patricia Tree状态树。

[0036] 可选的，所述数据库为LevelDB数据库；或者基于LevelDB架构的数据库；

[0037] 可选的，所述数据库为基于LevelDB架构的Rocksdb数据库。

[0038] 可选的，所述多级数据存储对应的存储介质的读写性能，存在性能差异；其中，级数高的数据存储对应的存储介质的读写性能，高于级数低的数据存储对应的存储介质的读写性能。

[0039] 在以上技术方案中，可以实现对数据库中存储的Merkle状态树上的数据节点进行剪枝，将记录了历史状态数据的数据节点从Merkle状态树上移除，迁移至下一级数据存储，并在本级数据存储中继续存储和保留记录了最新状态数据的数据节点，进而完成针对数据库中存储的Merkle状态树的分级存储。

附图说明

[0040] 图1是一示例性实施例提供的一种将区块链的账户状态数据组织成MPT状态树的示意图；

[0041] 图2是一示例性实施例提供的一种MPT状态树上的node复用的示意图；

[0042] 图3是一示例性实施例提供的一种基于区块链的分级存储方法的流程图；

[0043] 图4是一示例性实施例提供的一种电子设备的结构示意图；

[0044] 图5是一示例性实施例提供的一种基于区块链的分级存储装置的框图。

具体实施方式

[0045] 这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施例中所描述的实施方式并不代表与本说明书一个或多个实施例相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本说明书一个或多个实施例的一些方面相一致的装置和方法的例子。

[0046] 需要说明的是：在其他实施例中并不一定按照本说明书示出和描述的顺序来执行相应方法的步骤。在一些其他实施例中，其方法所包括的步骤可以比本说明书所描述的更多或更少。此外，本说明书中所描述的单个步骤，在其他实施例中可能被分解为多个步骤进行描述；而本说明书中所描述的多个步骤，在其他实施例中也可能被合并为单个步骤进行

描述。

[0047] 区块链一般被划分为三种类型:公有链(Public Blockchain),私有链(Private Blockchain)和联盟链(Consortium Blockchain)。此外,还有多种类型的结合,比如私有链+联盟链、联盟链+公有链等不同组合形式。其中去中心化程度最高的是公有链。公有链以比特币、以太坊为代表,加入公有链的参与者可以读取链上的数据记录、参与交易以及竞争新区块的记账权等。

[0048] 而且,各参与者(即节点)可自由加入以及退出网络,并进行相关操作。私有链则相反,该网络的写入权限由某个组织或者机构控制,数据读取权限受组织规定。简单来说,私有链可以为一个弱中心化系统,参与节点具有严格限制且少。这种类型的区块链更适合于特定机构内部使用。

[0049] 基于区块链的基本特性,区块链通常是由若干个区块构成。在这些区块中分别记录有与该区块的创建时刻对应的时间戳,所有的区块严格按照区块中记录的时间戳,构成一条在时间上有序的数据链条。

[0050] 对于物理世界产生的真实数据,可以将其构建成区块链所支持的标准的交易(transaction)格式,然后发布至区块链,由区块链中的节点设备进行共识,并在达成共识后,由区块链中作为记账节点的节点设备,将这笔交易打包进区块,在区块链中进行持久化存证。

[0051] 在区块链领域,有一个重要的概念就是账户(Account);以以太坊为例,以太坊通常将账户划分为外部账户和合约账户两类;外部账户就是由用户直接控制的账户;而合约账户则是由用户通过外部账户创建的,包含合约代码的账户(即智能合约)。

[0052] 当然,对于一些基于以太坊的架构而衍生出的区块链项目(比如蚂蚁区块链),还可以对区块链支持的账户类型,进行进一步的扩展,在本说明书中不进行特别限定。

[0053] 对于区块链中的账户而言,通常会通过一个结构体,来维护账户的账户状态。当区块中的交易被执行后,区块链中与该交易相关的账户的状态通常也会发生变化。

[0054] 以以太坊为例,账户的结构体通常包括Balance,Nonce,Code和storage等字段。其中:

[0055] Balance字段,用于维护账户目前的账户余额;

[0056] Nonce字段,用于该账户的交易次数;它是用于保障每笔交易能且只能被处理一次的计数器,有效避免重放攻击。

[0057] code字段,用于维护该账户的合约代码;在实际应用中,code字段中通常仅维护合约代码的hash值;因而,code字段通常也称之为codehash字段。对于外部账户而言,该字段为空值。

[0058] storage字段,用于维护该账户的存储(默认为空)。在实际应用中,storage字段仅维护基于账户的存储内容构建的MPT(Merkle Patricia Trie)树的根节点;因此,storage字段通常也称之为storageRoot字段。

[0059] 其中,对于外部账户而言,以上示出的code字段和storage字段为空值。

[0060] 而大多数区块链项目,通常都会使用Merkle树;或者,基于Merkle树的数据结构,来存储和维护数据。以以太坊为例,以太坊使用了MPT树(一种Merkle树变种),作为数据组织形式,用来组织和管理账户状态、交易信息等重要数据。

[0061] 以太坊针对区块链中需要存储和维护的数据，设计了三颗MPT树，分别是MPT状态树、MPT交易树和MPT收据树。

[0062] MPT状态树，是区块链中所有账户的账户状态数据(state)，组织成的MPT树；MPT交易树是区块中的交易数据(transaction)，组织成的MPT树；MPT收据树，是区块中的交易执行完毕后生成的与每笔交易对应的交易收据(receipt)，组织成的MPT树。以上示出的MPT状态树、MPT交易树和MPT收据树的根节点的hash值，都会被添加至区块头中。

[0063] 其中，MPT交易树和MPT收据树，与区块相对应，每一个区块都有自己的MPT交易树和MPT收据树。而MPT状态树是一个全局的MPT树，并不与某一个特定的区块相对应，而是涵盖了区块链中所有账户的账户状态数据。

[0064] 对于组织成的MPT交易树、MPT收据树和MPT状态树，最终都会在采用多级数据存储结构的Key-Value型数据库(比如，LevelDB)中进行存储。

[0065] 而采用多级存储结构的上述数据库，通常可以被划分为n级数据存储；例如，各级数据存储可以依次设为L0，L1，L2，L3....L(n-1)；对于上述数据库中的各级数据存储而言，等级编号越小通常级别越高；例如，L0存储的是最新的若干区块的数据，L1存储的是次新的若干区块数据，依次类推。

[0066] 其中，各级数据存储对应的存储介质的读写性能，通常也可以存在性能差异；级别高(即等级编号较小的)的数据存储对应的存储介质的读写性能，可以高于级别低的数据存储对应的存储介质的读写性能。

[0067] 例如，在实际应用中，级别高的数据存储，可以使用读写性能较高的存储介质；而级别低的数据存储，可以使用单位成本低，且容量较大的存储介质。

[0068] 在实际应用中，随着区块高度的增长，在数据库中存储的数据，会包含很多历史数据；而且，区块号越小的区块中的数据越久远，越不重要。因此，为了降低整体的存储成本，通常需要对不同区块高度的数据进行“区别对待”；

[0069] 例如，可以将区块号较小的区块中的数据，存储至成本较低的存储介质上；而将区块号较大的区块中的数据，存储在成本较高的存储介质上。

[0070] 在针对数据库中存储的MPT交易树、MPT收据树和MPT状态树等数据进行分级存储时，由于MPT交易树和MPT收据树，与各个区块相对应，实际上是“区块间无关”的数据；因此，对于MPT交易树和MPT收据树，很容易进行分级存储；例如，直接按照MPT交易树和MPT收据树上的node所属的区块号进行数据迁移即可完成分级存储。

[0071] 基于此，本说明书将不再具体阐述MPT交易树和MPT收据树的分级存储，而重点阐述MPT状态树的分级存储。

[0072] 请参见图1，图1为本说明书示出的一种将区块链的账户状态数据组织成MPT状态树的示意图。

[0073] MPT树，是一种经过改良的，融合了Merkle树和Trie字典树(也称之为前缀树)两种树形结构的优点的Merkle树变种。

[0074] 在MPT树中通常包括三种数据节点，分别为叶子节点(leaf node)，扩展节点(extension node)和分支节点(branch node)。

[0075] 叶子节点，表示为[key，value]的一个键值对，其中key是种特殊十六进制编码。

[0076] 扩展节点，也是[key，value]的一个键值对，但是这里的value是其他节点的hash

值(hash指针)。也就是说通过hash指针链接到其他节点。

[0077] 分支节点,因为MPT树中的key被编码成一种特殊的16进制的表示,再加上最后的value,所以分支节点是一个长度为17的list,前16个元素对应着key中的16个可能的十六进制字符(一个字符对应一个半字节nibble)。如果有一个[key,value]对在这个分支节点终止,最后一个元素代表一个value值,即分支节点既可以是搜索路径的终止也可以是路径的中间节点。

[0078] 假设需要组织成MTP状态树的账户状态数据如下表1所示:

[0079]

账户地址(Key)							账户状态(Value)
a	7	1	1	3	5	5	state1
a	7	7	d	3	3	7	state2
a	7	f	9	3	6	5	state3
a	7	7	d	3	9	7	state4

[0080] 表1

[0081] 在表1中,账户地址是由若干16进制的字符构成的字符串。账户状态state,是由上述Balance,Nonce,Code和storage等字段构成的结构体。

[0082] 最终按照表1中的账户状态数据组织成的MPT状态树,参见图1所示;如图1所示,按照表1中的账户状态数据组织成的MPT状态树,是由4个叶子节点,2个分支节点,和2个扩展节点构成。

[0083] 在图1中,prefix字段为扩展节点和叶子节点共同具有的前缀字段。该prefix字段的取值,在实际应用中可以用于表示节点类型。

[0084] prefix字段的取值为0,表示包含偶数个nibbles的扩展节点;如前所述,nibble表示半字节,由4位二进制组成,一个nibble可以对应一个组成账户地址的字符。

[0085] prefix字段的取值为1,表示包含奇数个nibble(s)的扩展节点;

[0086] prefix字段的取值为2,表示包含偶数个nibbles的叶子节点;

[0087] prefix字段的取值为3,表示包含奇数个nibble(s)的叶子节点。

[0088] 而分支节点,由于其是并列单nibble的前缀节点,因此分支节点不具有上述prefix字段。

[0089] 扩展节点中的Shared nibble字段,对应该扩展节点所包含的键值对的key值,表示账户地址之间的共同字符前缀;比如,上表中的所有账户地址均具有共同的字符前缀a7。Next Node字段中填充下一个节点的hash值(hash指针)。

[0090] 分支节点中的16进制字符0～f字段,对应该分支节点所包含的键值对的key值;如果该分支节点为账户地址在MPT树上的搜索路径上的中间节点,则该分支节点的Value字段可以为空值。0～f字段中用于填充下一个节点的hash值。

[0091] 叶子节点中的Key-end,对应该叶子节点所包含的键值对的key值,表示账户地址的最后几个字符。从根节点搜索到叶子节点的搜索路径上的各个节点的key值,构成了一个完整的账户地址。该叶子节点的Value字段填充账户地址对应的账户状态数据;例如,可以对上述Balance,Nonce,Code和storage等字段构成的结构体进行编号后,填充至叶子节点的Value字段。

[0092] 进一步的,如图1所示的MPT状态树上的node,最终也是以Key-Value键值对的形式

存储在数据库中；

[0093] 其中，当MPT状态树上的node在数据库中进行存储时，MPT状态树上的node的键值对中的key，为node所包含的数据内容的hash值；MPT状态树上的node的键值对中的Value，为node所包含的数据内容。

[0094] 也即，在将MPT状态树上的node存储至数据库时，可以计算该node所包含的数据内容的hash值(即对node整体进行hash计算)，并将计算出的hash值作为key，将该node所包含的数据内容作为value，生成Key-Value键值对；然后，将生成的Key-Value键值对存储至数据库中。

[0095] 由于MPT状态树上的node，是以node所包含的数据内容的hash值为Key，node所包含的数据内容为value进行存储；因此，在需要查询MPT状态树上的node时，通常可以基于node所包含的数据内容的hash值作为key来进行内容寻址。而采用“内容寻址”，对于一些“内容重复”的node，则通常可以进行“复用”，以节约数据存储的存储空间。

[0096] 如图2所示，图2为本说明书示出的一种MPT状态树上的node复用的示意图。

[0097] 在实际应用中，区块链每产生一个最新区块，则在该最新区块中的交易被执行之后，区块链中与这些被执行的交易相关账户的账户状态，通常也会随之发生变化；

[0098] 例如，当区块中的一笔“转账交易”执行完毕后，与该“转账交易”相关的转出方账户和转入方账户的余额(即这些账户的Balance字段的取值)，通常也会随之发生变化。

[0099] 而节点设备在区块链产生的最新区块中的交易执行完毕后，由于当前区块链中的账户状态发生了变化，因此节点设备需要根据区块链中所有账户当前的账户状态数据，来构建MPT树，用于维护区块链中所有账户的最新状态。

[0100] 也即，每当区块链中产生一个最新区块，并且该最新区块中的交易执行完毕后，导致区块链中的账户状态发生变化，节点设备都需要基于区块链中所有账户最新的账户状态数据，重新构建一颗MPT树。

[0101] 换句话说，区块链中每一个区块，都有一个与之对应的MPT状态树；该MPT状态树，维护了在该区块中的交易在执行完毕后，区块链中所有账户最新的账户状态。

[0102] 而需要说明的是，一个最新区块中的交易执行完毕后，可能仅仅会导致部分账户的账户状态发生变化；因此，在更新MPT状态树时，并不需要基于区块链中所有的账户当前的状态数据，重新构建一颗完整的MPT状态树，而只需要在该最新区块之前的区块对应的MPT状态树的基础上，对部分账户状态发生变化的账户对应的node进行更新即可。而对于MPT状态树上与账户状态未发生变化的账户对应的node而言，由于这些node为发生数据更新，可以直接复用该最新区块之前的区块对应的MPT状态树上相应的node即可。

[0103] 如图2所示，假设表1中的账户状态数据，为Block N中的交易执行完毕后，区块链上所有账户的最新账户状态；基于表1中的账户状态数据组织成的MPT状态树，仍如图1所示。

[0104] 假设当Block N+1中的交易执行完毕后，导致上述表1中的账户地址为“a7f9365”的账户状态，由“state3”更新为“state5”；此时，在Block N+1更新MPT状态树时，并不需要基于Block N+1中的交易执行完毕后，区块链中所有的账户当前的状态数据，重新构建一颗MPT状态树。

[0105] 请参见图2，在这种情况下，可以仅将Block N对应的MPT树上(即图1示出的MPT状

态树)，“key-end”为“9365”的叶子节点中的Value，由“state3”更新为“state5”，并继续更新从root节点到该叶子节点的路径上的所有节点的hash指针；也即，当MPT状态树上的叶子节点发生更新，由于该叶子节点整体的hash值发生更新，那么从根节点到该叶子节点的路径上的所有的节点的hash指针也会随之发生更新。例如，请继续参见图2，除了需要更新“key-end”为“9365”的叶子节点中的Value值以外，还需要更新该叶子节点的上一个分支节点(Branch Node)的f字段中填充的，指向该叶子节点的哈希指针；进一步的，还可以继续向根节点追溯，继续更新该分支节点的上一个根节点(Root Extension Node)的“Next Node”字段中填充的，指向该分支节点的hash指针。

[0106] 而除了以上发生更新的节点以外，其它未发生更新的节点，都可以直接复用Block N的MPT状态树上对应的节点即可；

[0107] 其中，由于Block N对应的MPT树，最终需要作为历史数据进行保留；因此，在Block N+1更新MPT状态树时，对于这些发生更新的node，并不是对Block N对应的MPT状态树上原来的node的基础上，直接进行修改更新，而是在Block N+1对应的MPT树上重新创建这些发生更新的node。

[0108] 也即，对于与Block N+1对应的MPT状态树上，实际上只需要重新创建少量发生更新的node，对于其它未发生更新的node，可以通过直接复用Block N对应的MPT状态树上对应的节点。

[0109] 例如，如图2所示，对于Block N+1对应的MPT状态树上，实际上只需要重新创建少量发生更新的node；比如，图2中仅需要重新创建一个作为根节点的扩展节点、一个分支节点和一个叶子节点；对于未发生更新的node，可以通过在该MPT状态树上这些重新创建的node中，添加指向Block N对应的MPT状态树上的相应node的hash指针来完成node的复用。而Block N对应的MPT状态树上那些更新前的node，将作为历史账户状态数据进行保存；比如，图2示出的“key-end”为“9365”，且Value为“state3”的叶子节点，将作为历史数据进行保留。在以上例子中，以Block N+1的MPT状态树上的少量node发生内容更新，可以“复用”上一个区块Block N的大多数node为例进行了说明。而在实际应用中，Block N+1的MPT状态树上也可能会较上一个区块Block N新增node。

[0110] 在这种情况下，该新增的node虽然无法直接从上一个区块Block N的MPT树中进行复用，但有可能从更早之前的区块的MPT状态树上进行“复用”；

[0111] 例如，Block N+1的MPT状态树上新增的node，虽然在Block N的MPT状态树上出现过，但出现在更早的Block的MPT状态树上；比如，出现在Block N-1的MPT状态树上；因此，Block N+1的MPT状态树上新增的node，可以直接复用Block N-1的MPT状态树上对应的node即可。

[0112] 可见，MPT状态树的节点复用，一共有两种“复用”的情形：

[0113] 一种情形是，一个区块的MPT状态树上只有少量的节点发生内容更新，则可以“复用”上一个区块的大多数节点；

[0114] 另一种情形是，一个区块的MPT状态树较上一个区块的MPT状态树新增了节点，则可以“复用”更早之前的区块的MPT状态树上对应的节点。

[0115] 然而，通过节点复用，虽然可以节约数据库的存储空间，但由于各个区块的MPT状态树上的节点之间，可能存在复杂的复用关系，每个区块的MPT状态树上的node，都可能被

下一个区块,或者是下一个区块之后的若干连续的区块进行复用;因此,这种复杂的节点复用关系,势必会对MPT状态树的分级存储造成困难。

[0116] 例如,在需要将一些节点作为历史数据,从本级数据存储向下一级数据存储迁移时,由于这些节点可能会被下一个区块;甚至是该下一个区块之后的若干个区块复用;而这些节点将会被哪些节点复用,又是无法准确预知的;因此,这就会导致无法对数据库中存储的MPT状态树上的节点进行精确剪枝;其中,所谓剪枝,是指清除各区块的MPT状态树上的节点之间的复用关系,将记录了历史状态数据的节点从MPT状态树上移除,并保留记录了最新状态数据的node。在这种情况下,显然无法满足分级存储的需求。

[0117] 基于此,本说明书提出一种针对由区块链的账户状态数据组成的Merkle状态树进行分级存储的方法。

[0118] 在实现时,仍然可以将区块链的账户状态数据组织成Merkle状态树,然后将Merkle状态树上的数据节点,以Key-Value键值对的形式在采用多级数据存储结构的数据库中进行存储;例如,仍然可以采用MPT树的数据结构,将区块链的账户状态数据组织成MPT状态树;而对于在数据库中以Key-Value键值对的形式存储的Merkle状态树上的node而言,可以在这些node对应的Key-Value键值对的key中,为这些node分别标记一个区块号,该区块号具体用于指示该node发生数据更新时所在区块的区块号;

[0119] 例如,在一种实施方式中,区块链产生的最新区块中的交易执行完毕后,可以基于该最新区块中的交易的执行结果,来确定该最新区块的Merkle状态树上发生数据更新的node;其中,发生数据更新的node,通常包含value值发生更新的node和新增的node。当确定该最新区块的Merkle状态树上发生数据更新的node之后,可以在这些node对应的Key-Value键值对的key中,为这些发生数据更新的node标记该最新区块的区块号,表示该node在该区块号对应的区块发生了数据更新。

[0120] 其中,Merkle状态树上的数据节点以Key-Value键值对的形式在数据库中进行存储时,可以不再采用上述数据节点所包含内容的hash值作为上述Key-Value键值对的key,而是采用上述数据节点的节点ID和能够指示该数据节点发生数据更新时所在区块的区块号组成的二元组作为key。

[0121] 进一步的,当数据库中的任一级目标数据存储满足了数据迁移条件时;比如,当该目标数据存储的存储容量达到阈值;首先,可以确定该目标数据存储中需要迁移至下一级数据存储的node对应的区块号区间,并基于该区块号区间确定迁移临界值;其中,该迁移临界值为大于该区块号区间的区块号临界值;

[0122] 例如,在一种实施方式中,如果该区块号区间的右区间为开区间,则将该区块号区间的右端点值,确定为上述迁移临界值;比如,假设该区块号区间为[a,b)时,则将b确定为上述迁移临界值;如果该区块号区间的右区间为闭区间,将该区块号区间的右端点值与区块链的区块号递增步长的和,确定为上述迁移临界值;比如,假设该区块号区间为[a,b],区块链的区块号递增步长为1(即区块链中的区块号以1为递增步长致密递增)时,则将b+1确定为上述迁移临界值。

[0123] 当确定了上述迁移临界值后,迭代遍历上述目标数据存储中存储的数据节点的上述二元组,查找上述二元组中的区块号小于上述迁移临界值的目标数据节点;

[0124] 当查找到上述目标数据节点之后,根据该目标数据节点的上述二元组,继续在上

述目标数据存储中进行寻址，进一步查找上述目标数据存储中是否存储了，与上述目标数据节点具有相同的节点ID；以及，上述二元组中的区块号大于上述目标数据节点，且小于所述迁移临界值的数据节点；

[0125] 如果是，可以将该目标数据节点迁移至上述目标数据存储的下一级数据存储；例如，将查找到的该目标数据节点写入到该目标数据存储的下一级数据存储，在写入成功后，再将该目标数据节点从目标数据存储中清除；否则，可以在上述目标数据存储中继续保留该目标数据节点。

[0126] 通过以上技术方案，可以实现对数据库中存储的Merkle状态树上的node进行精确剪枝，将记录了历史状态数据的node从Merkle状态树上移除，迁移至下一级数据存储，并在本级数据存储中继续存储和保留记录了最新状态数据的node，进而完成针对数据库中存储的Merkle状态树的分级存储。

[0127] 请参见图3，图3是一示例性实施例提供的一种基于区块链的分级存储方法的流程图。所述方法应用于区块链节点设备；其中，所述区块链的账户状态数据组织成的Merkle状态树上的数据节点，以Key-Value键值对的形式存储在数据库中；其中，所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组；所述数据库包括多级数据存储；所述区块号指示所述数据节点发生数据更新时所在的区块；所述方法包括以下步骤：

[0128] 步骤302，当所述数据库中的任一级目标数据存储满足了数据迁移条件时，确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间；

[0129] 步骤304，基于所述区块号区间确定迁移临界值；其中，所述迁移临界值为大于所述区块号区间的区块号临界值；

[0130] 步骤306，迭代遍历所述目标数据存储中存储的数据节点的所述二元组，查找所述二元组中的区块号小于所述迁移临界值的目标数据节点；

[0131] 步骤308，根据所述目标数据节点的所述二元组，进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID，以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点；如果是，将所述目标数据节点迁移至所述目标数据存储的下一级数据存储；否则，在所述目标数据存储中保留所述目标数据节点。

[0132] 上述数据库，具体可以是采用多级数据存储结构的Key-Value型数据库(比中进行存储；例如，在示出的一种实施方式中，上述数据库可以为LevelDB数据库；或者，基于LevelDB架构的数据库；比如，Rocksdb数据库就是一种典型的基于LevelDB数据库架构的数据库。

[0133] 区块链中的账户状态数据，可以被组织成Merkle状态树的数据结构，在上述数据库中进行存储；例如，上述Merkle状态树具体可以是MPT树，可以采用MPT树的数据结构，将区块链的账户状态数据组织成MPT状态树。

[0134] 以下以采用MPT树的数据结构，将区块链中的账户状态数据组织成MPT状态树为例，对本说明书的技术方案进行详细描述；

[0135] 其中，需要强调的是，以采用MPT树的数据结构来组织区块链中的账户状态数据，仅为示例性的。

[0136] 在实际应用中，对于基于以太坊架构而衍生出的区块链项目，除了可以采用诸如

MPT树等改良版的Merkle树以外，也可以采用其他形式的类似于MPT树的融合了Trie字典树的树形结构的Merkle树变种，在本说明书中不再进行一一列举。

[0137] 在本说明书中，接入区块链的用户客户端，可以将数据打包成区块链所支持的标准的交易格式，然后发布至区块链；而区块链中的节点设备，可以基于搭载的共识算法与其它节点设备一起，对用户客户端发布至区块链的这些交易进行共识，以此来为区块链产生最新区块；

[0138] 其中，区块链中支持的共识算法，通常分为节点设备需要争夺每一轮的记账周期的记账权的共识算法，和预先为每一轮记账周期选举记账节点(不需要争夺记账权)的共识算法。

[0139] 例如，前者以工作量证明(Proof of Work,POW)、股权证明(Proof of Stake,POS)、委任权益证明(Delegated Proof of Stake,DPOS)等共识算法为代表；后者以实用拜占庭容错(Practical Byzantine Fault Tolerance,PBFT)等共识算法为代表。

[0140] 对于采用工作量证明(Proof of Work,POW)以及股权证明(Proof of Stake,POS)、委任权益证明(Delegated Proof of Stake,DPOS)等共识算法的区块链网络中，争夺记账权的节点设备，都可以在接收到交易后执行该笔交易。争夺记账权的节点设备中可能其中一个在本轮争夺记账权的过程中胜出，成为记账节点。记账节点可以将收到的交易与其它交易一起打包并生成最新区块，并将生成的最新区块发送至其它节点设备进行共识。

[0141] 对于采用实用拜占庭容错(Practical Byzantine Fault Tolerance,PBFT)等共识算法的区块链网络中，具有记账权的节点设备在本轮记账前已经商定好。因此，节点设备在接收到交易后，如果自身不是本轮的记账节点，则可以将该交易发送至记账节点。

[0142] 对于本轮的记账节点，在将该交易与其它交易一起打包并生成最新区块的过程中或者之前，可以执行该交易。记账节点在将该交易与其它交易一起打包生成新区块后，可以将生成的最新区块或者该最新区块的区块头发送至其它节点设备进行共识。

[0143] 如上所述，无论区块链采用以上示出的哪种共识算法，本轮的记账节点都可以将接收到的交易打包并生成最新区块，并将生成的最新区块或者该最新区块的区块头发送至其它节点设备进行共识验证。如果其它节点设备接收到最新区块或者该最新区块的区块头后，经验证没有问题，可以将该最新区块追加到原有的区块链末尾，从而完成区块链的记账过程。

[0144] 在本说明书中，区块链中的节点设备在执行了经过共识产生的最新区块中打包的交易之后，区块链中与这些被执行的交易相关的账户状态，通常也会随之发生变化；因此，节点设备在最新区块中打包的交易执行完毕后，可以根据区块链中所有账户最新的账户状态数据，组织成MPT状态树的数据结构。

[0145] 其中，根据区块链中所有账户最新的账户状态数据，组织成MPT状态树时，仍然可以采用如图2示出的复用该最新区块之前的区块对应的MPT树上的节点的方式，在本说明书中不再赘述。

[0146] 当节点设备根据区块链中所有账户最新的账户状态数据，组织成MPT状态树后，可以将该MPT状态树上的数据节点，以Key-Value键值对的形式在采用多级数据存储结构的Key-Value型数据库中进行存储。

[0147] 例如，在实际应用中，对于最新的若干区块对应的MPT状态树上的数据节点，可以

默认存放在上述数据库中的级别最高的L0级数据存储中。对于次新的若干区块对应的MPT状态树上的数据节点，可以存放在上述数据库中的级别次高的L1级数据存储中；以此类推。其中，每一级数据存储所存储的MPT状态树对应的区块数，在本说明书中不进行特别限定；比如，可以规定级别最高的L0级数据存储中存储最新的N个区块的MPT状态树，级别次高的L1级数据存储次新的N个区块的MPT状态树；以此类推。

[0148] 在本说明书，对于构建的MPT状态树上的数据节点，可以被标记区块号；其中，为MPT状态树上的数据节点所标记的区块号，具体用于指示该数据节点发生数据更新时所在的区块；

[0149] 例如，如图2所示，以Block N+1对应的MPT状态树上发生数据更新的node为例，为这些node标记的区块号，即为N+1，以表示这些node在Block N+1中的交易执行完毕后，发生了数据更新。

[0150] 在实现时，节点设备可以在本地启动一个“区块状态更新线程”，来维护和更新MPT状态树上的node的状态。当区块链产生的最新区块中的交易执行完毕后，该“区块状态更新线程”可以确定该最新区块对应的MPT状态树上发生数据更新的node；其中，发生数据更新的node，通常包含value值发生更新的node和新增的node。

[0151] 例如，在实现时，可以直接根据该最新区块中的交易的执行结果，来确定该最新区块对应的MPT状态树上发生数据更新的node；或者，也可以通过判断MPT状态树上的node是否为复用的node，来确定该最新区块对应的MPT状态树上发生数据更新的node；比如，如果是复用的node，则表明该node是未发生数据更新的node；

[0152] 当确定该最新区块对应的MPT状态树上发生数据更新的node之后，该“区块状态更新线程”，可以为这些发生数据更新的node标记该最新区块的区块号，表示该node在该区块号对应的区块发生了数据更新。

[0153] 通过这种方式，每当区块链产生一个最新区块，“区块状态更新线程”都可以及时的对该最新区块对应的MPT状态树上发生数据更新的node，标记该最新区块的区块号，使得通过遍历上述数据库中存储的MPT状态树上的各个node，就能够通过为各个node标记的区块号，获知到该node发生数据更新时所在的区块；由于MPT状态树上的各个node(尤其是MPT树上的叶子节点)最近一次发生数据更新后的value值，通常可以指示该node的最新状态；因此，通过查看为各个node标记的区块号，则可以获知各个node的最新状态，是在哪个区块产生的；从而这种为node标记区块号的机制，可以对上述数据库的各级数据存储的数据迁移提供依据。

[0154] 其中，在本说明书中，“区块状态更新线程”，具体可以在MPT状态树上的node对应的Key-Value键值对的key中，为MPT状态树上的node标记上述区块号。

[0155] 当确定该最新区块对应的MPT状态树上发生数据更新的node之后，“区块状态更新线程”，可以为这些发生数据更新的node构建Key-Value键值对，并在构建的Key-Value键值对的key中，为上述node标记区块号；然后将构建的Key-Value键值在上述数据库中进行存储。

[0156] 其中，MPT状态树上的数据节点以Key-Value键值对的形式在数据库中进行存储时，可以不再采用上述数据节点所包含内容的hash值作为上述Key-Value键值对的key，而是采用上述数据节点的节点ID和能够指示该数据节点发生数据更新时所在区块的区块号

组成的二元组作为上述Key-Value键值对的key。上述Key-Value键值对的Value仍然可以为数据节点所包含的数据内容。例如，上述二元组可以记为[nid,h]，nid代表NodeID；代表为该节点标记的区块号。

[0157] 也即，在本说明书中，将不再使用数据节点所包含的数据内容的hash值，作为数据节点的key；而是采用数据节点的节点ID和能够指示该数据节点发生数据更新时所在区块的区块号组成的二元组作为数据节点的key。

[0158] 通过这种方式，上述数据库所采用的寻址方式也随之会发生变化；对于上述数据库而言，将不再采用node所包含的数据内容的hash值作为key来进行内容寻址；取而代之的是，将采用节点ID和数据节点发生数据更新时所在区块的区块号来进行寻址。

[0159] 其中，需要说明的是，上述数据节点的节点ID，具体是指数据节点在MPT状态树中的唯一标识；

[0160] 例如，在实际应用中，对应诸如MPT树这种融合了Trie字典树的树形结构的Merkle树变种，通常采用从为MPT树的根节点到该数据节点的路径对应的字符前缀；比如，如图1和图2示出的任意一个叶子节点而言，从根节点到该叶子节点的路径对应的字符前缀通常为叶子节点对应的账户地址；也即，MPT树上的叶子节点的节点ID，即为该叶子节点对应的账户地址。

[0161] 当然，在实际应用中，除了采用MPT树的根节点到数据节点的路径对应的字符前缀作为该数据节点的节点ID以外，上述数据节点的节点ID，具体也可以是在构建MPT树时，为各数据节点设置的能够唯一区分各个数据节点的节点编号，在本说明书中不进行特别限定。

[0162] 在本说明书中，节点设备还可以在本地启动一个“迁移线程”，用于将上述数据库中各级数据存储中存储的MPT状态树上的node数据，向下级数据存储进行迁移。

[0163] 上述“迁移线程”，具体可以执行一个定时任务，可以周期性的确定上述数据库中的各级数据存储是否满足预设的数据迁移条件；

[0164] 其中，上述数据库中的各级数据存储的数据迁移条件，可以基于实际的数据迁移需求来设置，在本说明书中不进行特别限定；

[0165] 例如，在实际应用中，上述数据库的各级数据存储的数据迁移条件，具体可以是各级数据存储的存储容量达到阈值；或者，也可以是各级数据存储所存储的数据所对应的区块数达到阈值。

[0166] 上述“迁移线程”，在确定上述数据库中任一级目标数据存储，满足了数据迁移条件，则该“迁移线程”可以针对该目标数据存储进行数据迁移处理，将该目标数据存储中存储的部分区块的MPT状态树，作为历史数据向下一级数据存储进行迁移。

[0167] 在实现时，当目标数据存储满足了数据迁移条件时，上述“迁移线程”可以确定该目标数据存储中，需要迁移至下一级数据存储的node对应的区块号区间；

[0168] 其中，该目标数据存储中需要迁移至下一级数据存储的node对应的区块号区间，可以基于上一次数据迁移发生后的下一区块的区块号，和该目标数据存储一次能够迁移走的最大区块数来确定；比如，假设上一次数据迁移后的下一个区块是Block N，该目标数据存储一次能够迁移走的区块数为30，那么上述区块号区间具体可以是[N,N+29]；或者，也可以是[N,N+30)。

[0169] 当上述“迁移线程”确定出该目标数据存储中，需要迁移至下一级数据存储的node对应的区块号区间之后，可以基于该区块号区间，来确定本次数据迁移的迁移临界值；其中，该迁移临界值具体可以是一个大于该区块号区间的区块号临界值；

[0170] 需要说明的是，在实际应用中，上述区块号区间的右区间具体可以是开区间，也可以是闭区间；而在基于上述区块号区间来确定上述迁移临界值时，上述区块号区间的右区间为开区间或者闭区间时，确定出的迁移临界值也可以存在一定的差异。

[0171] 在示出的一种实施方式中，如果上述区块号区间的右区间为开区间，在基于上述区块号区间来确定上述迁移临界值时，可以将上述区块号区间的右端点值，确定为上述迁移临界值；

[0172] 例如，假设上述区块号区间为[a,b)，那么基于该区块号区间确定出的迁移临界值为该区块号区间的右端点值b。

[0173] 在示出的另一种实施方式中，如果上述区块号区间的右区间为闭区间，在基于上述区块号区间来确定上述迁移临界值时，可以将上述区块号区间的右端点值与区块链的区块号递增步长的和，确定为上述迁移临界值；

[0174] 例如，假设上述区块号区间为[a,b]，区块链的区块号递增步长为1，那么基于该区块号区间确定出的迁移临界值为该区块号区间的右端点值b+1。

[0175] 其中，上述区块链的区块号递增步长，通常可以是1；也即，区块链的区块号按照1为步长进行致密递增；比如，区块号按照1、2、3、4…的顺序进行递增。

[0176] 当然，在实际应用中，区块链的区块号递增步长具体也可以是大于1的整数；比如，区块号也可以按照1、3、5、7…的顺序进行递增，在本说明书中不进行特别限定。

[0177] 进一步的，当上述“迁移线程”，基于上述区块号区间，确定出本次数据迁移的迁移临界值之后，可以根据该迁移临界值，将该目标数据存储中存储的与上述区块号区间对应的区块的MPT状态树，作为历史数据向下一级数据存储进行迁移。

[0178] 具体地，上述“迁移线程”，可以迭代遍历上述目标数据存储中存储的各个node的key；也即，迭代遍历上述目标数据存储中存储的各个node的节点ID和被标记的区块号所组成的上述二元组，来查找上述目标数据存储中存储的所有的node中，上述二元组中的区块号小于上述迁移临界值的目标node。

[0179] 当查找到一个上述二元组中的区块号小于上述迁移临界值的目标node之后，上述“迁移线程”可以根据该目标node的上述二元组，继续在上述目标数据存储中进行寻址，进一步查找上述目标数据存储中是否存储了，与上述目标node具有相同的节点ID(也即具有相同的字符前缀)；以及，上述二元组中的区块号大于上述目标node且小于上述迁移临界值的node；

[0180] 如果存在，表明该目标node所代表的账户状态，并不是最新的账户状态；此时该目标node，即可以作为迁移至该目标数据存储的下一级数据存储的历史数据，上述“迁移线程”可以将查找到的该目标node，迁移至该目标数据存储的下一级数据存储；

[0181] 例如，在实现时，上述“迁移线程”可以复制该目标node，并将复制的该目标node存储至下一级数据存储，然后在将复制的该目标node成功存储至下一级数据存储之后，再将该目标node从上述目标数据存储中清除。

[0182] 反之，如果不存在，表明该目标node所代表的账户状态，为最新的账户状态；在这

种情况下，该目标node将会继续在该目标数据存储中进行保留。

[0183] 例如，举例而言，假设上述区块号区间为[a,b)，那么基于该区块号区间确定出的迁移临界值为该区块号区间的右端点值b；在这种情况下，假设上述“迁移线程”在目标数据存储中查找到一个nodeA，该nodeA的上述二元组为(nid,i)；其中i<b；那么，上述“迁移线程”可以根据二元组(nid,i)进行寻址，进一步查找上述目标数据存储中是否存储了二元组为(nid,j)的nodeB；其中，b>j>i；如果存在nodeB，可以将nodeA迁移至上述目标数据存储的下一级数据存储；反之，在该目标数据存储中继续保留该nodeA。

[0184] 而上述“迁移线程”可以通过迭代遍历上述目标数据存储中存储的数据节点的上述二元组，并迭代执行上述迁移过程，最终可以仅在上述目标数据存储中保留，上述二元组中的区块号小于上述迁移临界值的最大区块号所对应的node；而除了该node以外的其它node，都将作为历史数据迁移至下一级数据存储。

[0185] 由于上述目标数据存储中存储的，上述二元组中的区块号小于上述迁移临界值的最大区块号所对应的node，通常代表了该node在最近一次更新后的最新状态，因此通过这种方式，相当于仅在上述目标数据存储中保留了各个node的最新状态。

[0186] 其中，需要说明的是，由于在本说明书中，MPT状态树上的node的Key-Value键值对的key，不再是node所包含的数据内容的hash值；因此，上述“迁移线程”在将上述目标node的上述二元组继续作为查询索引，在上述目标数据存储中进行寻址时，将不再采用内容寻址的方式。

[0187] 在示出的一种实施方式中，上述数据库可以提供“查询小于某值的最大值key”的功能；所谓“查询小于某值的最大值key”功能，具体是指将一个由[nid,h]组成的二元组key作为查询索引进行寻址，查找出数据库中存储的由[nid,h_{max}]组成的二元组key的过程；其中，h_{max}为数据库中存储的所有node的二元组中，小于h的区块号之中的区块号最大值。也即，在确定h_{max}时，可以先查找出数据库中存储的所有区块号小于h的二元组；然后再取查找出的这些二元组中的区块号最大值作为上述h_{max}。

[0188] 在这种情况下，上述“迁移线程”在根据上述目标node的上述二元组继续在上述目标数据存储中进行寻址时，可以根据上述目标node的节点ID(记为nid)和上述迁移临界值(记为h)拼接出二元组[nid,h]，然后将拼接出的该二元组[nid,h]作为查询索引，进一步查找上述目标数据存储中是否存储了二元组[nid,h_{max}]。

[0189] 当查找到二元组[nid,h_{max}]之后，可以进一步确定h_{max}是否大于上述目标node的上述二元组中的区块号；如果是，此时可以确定上述目标数据存储中存储了与上述目标node具有相同的节点ID，以及上述二元组中的区块号大于上述目标node且小于上述迁移临界值的node；反之，说明上述目标node的上述二元组中的区块号即为小于上述迁移临界值的最大区块号，此时可以确定上述目标数据存储中并没有存储与上述目标node具有相同的节点ID，以及上述二元组中的区块号大于上述目标node且小于上述迁移临界值的node。

[0190] 在以上技术方案中，由于上述目标数据存储中存储的，上述二元组中的区块号小于迁移临界值的node中，只有上述二元组中的最大区块号对应的node能继续保留在该目标数据存储，而其它的node都将作为历史数据迁移至下一级数据存储；

[0191] 因此，按照上述方式完成数据迁移后，相当于对数据库中存储的MPT状态树上的node进行了一次剪枝，将记录了历史状态数据的node从MPT状态树上移除，迁移至下一级数

据存储，并在本级数据存储中继续存储和保留记录了最新状态数据的node，进而完成针对数据库中存储的MPT状态树的分级存储。

[0192] 其中，需要说明的是，上述“区块状态更新线程”和上述“迁移线程”对应的处理动作，可以并发执行；也即，上述“区块状态更新线程”和上述“迁移线程”可以对MPT状态树上的同一个node进行并发竞争处理。因此，在实际应用中，为了避免上述“区块状态更新线程”和上述“迁移线程”对同一个node进行并发竞争处理，可能造成的处理错误，可以通过技术手段，来保证在任一时刻，只能有一个线程来访问MPT状态树上的同一个node。

[0193] 其中，在保证在任一时刻，只能有一个线程来访问MPT状态树上的同一个node所采用的具体技术手段，在本说明书中不进行特别限定；例如，在实现时，可以采用互斥锁技术、单线程技术来实现，在本说明书中不再进行详述。

[0194] 与上述方法实施例相对应，本申请还提供了装置的实施例。

[0195] 与上述方法实施例相对应，本说明书还提供了一种基于区块链的分级存储装置的实施例。

[0196] 本说明书的基于区块链的分级存储装置的实施例可以应用在电子设备上。装置实施例可以通过软件实现，也可以通过硬件或者软硬件结合的方式实现。以软件实现为例，作为一个逻辑意义上的装置，是通过其所在电子设备的处理器将非易失性存储器中对应的计算机程序指令读取到内存中运行形成的。

[0197] 从硬件层面而言，如图4所示，为本说明书的基于区块链的分级存储装置所在电子设备的一种硬件结构图，除了图4所示的处理器、内存、网络接口、以及非易失性存储器之外，实施例中装置所在的电子设备通常根据该电子设备的实际功能，还可以包括其他硬件，对此不再赘述。

[0198] 图5是本说明书一示例性实施例示出的一种基于区块链的分级存储装置的框图。

[0199] 请参考图5，所述基于区块链的分级存储装置50可以应用在前述图4所示的电子设备中，所述区块链的账户状态数据组织成的Merkle状态树上的数据节点，以Key-Value键值对的形式存储在数据库中；其中，所述Key-Value键值对的key为所述数据节点的节点ID和为所述数据节点标记的区块号组成的二元组；所述数据库包括多级数据存储；所述区块号指示所述数据节点发生数据更新时所在的区块；所述装置50包括：

[0200] 确定模块501，当所述数据库中的任一级目标数据存储满足了数据迁移条件时，确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间；基于所述区块号区间确定迁移临界值；其中，所述迁移临界值为大于所述区块号区间的区块号临界值；

[0201] 查找模块502，迭代遍历所述目标数据存储中存储的数据节点的所述二元组，查找所述二元组中的区块号小于所述迁移临界值的目标数据节点；

[0202] 迁移模块503，根据所述目标数据节点的所述二元组，进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID，以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点；如果是，将所述目标数据节点迁移至所述目标数据存储的下一级数据存储；否则，在所述目标数据存储中保留所述目标数据节点。

[0203] 在本说明书中，所述确定模块501：

[0204] 如果所述区块号区间的右区间为开区间，将所述区块号区间的右端点值，确定为所述迁移临界值；

[0205] 如果所述区块号区间的右区间为闭区间，将所述区块号区间的右端点值与所述区块链的区块号递增步长的和确定为所述迁移临界值。

[0206] 在本说明书中，所述迁移模块503进一步：

[0207] 查找所述目标数据存储中存储的与所述目标数据节点具有相同的节点ID，并且所述二元组中的区块号为小于所述迁移临界值的区块号中的最大区块号的数据节点；进一步确定查找到的该数据节点的所述二元组中的区块号，是否大于所述目标数据节点的所述二元组中的区块号。

[0208] 在本说明书中，所述装置50还包括：

[0209] 存储模块504(图5中未示出)，确定最新区块的Merkle状态树上发生数据更新的数据节点；为所述最新区块的Merkle状态树上发生数据更新的数据节点构建Key-Value键值对，并将所述Key-Value键值对存储至所述数据库中的最高级数据存储；

[0210] 其中，所述Key-Value键值对的key为所述最新区块的区块号和所述数据节点的节点ID组成的二元组；所述Key-Value键值对的Value为所述数据节点包含的数据内容。

[0211] 在本说明书中，所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种；所述数据节点的节点ID，为所述Merkle树的根节点到该数据节点的路径对应的字符前缀。

[0212] 在本说明书中，所述Merkle状态树为Merkle Patricia Tree状态树。

[0213] 在本说明书中，所述数据库为LevelDB数据库；或者基于LevelDB架构的数据库；

[0214] 在本说明书中，所述数据库为基于LevelDB架构的Rocksdb数据库。

[0215] 在本说明书中，所述多级数据存储对应的存储介质的读写性能，存在性能差异；其中，级数高的数据存储对应的存储介质的读写性能，高于级数低的数据存储对应的存储介质的读写性能。

[0216] 上述实施例阐明的系统、装置、模块或单元，具体可以由计算机芯片或实体实现，或者由具有某种功能的产品来实现。一种典型的实现设备为计算机，计算机的具体形式可以是个人计算机、膝上型计算机、蜂窝电话、相机电话、智能电话、个人数字助理、媒体播放器、导航设备、电子邮件收发设备、游戏控制台、平板计算机、可穿戴设备或者这些设备中的任意几种设备的组合。

[0217] 在一个典型的配置中，计算机包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

[0218] 内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器(RAM)和/或非易失性内存等形式，如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

[0219] 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带、磁盘存储、量子存储器、基于石墨烯的

存储介质或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体(transitory media)，如调制的数据信号和载波。

[0220] 还需要说明的是，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[0221] 上述对本说明书特定实施例进行了描述。其它实施例在所附权利要求书的范围内。在一些情况下，在权利要求书中记载的动作或步骤可以按照不同于实施例中的顺序来执行并且仍然可以实现期望的结果。另外，在附图中描绘的过程不一定要求示出的特定顺序或者连续顺序才能实现期望的结果。在某些实施方式中，多任务处理和并行处理也是可以的或者可能是有利的。

[0222] 在本说明书一个或多个实施例使用的术语是仅仅出于描述特定实施例的目的，而非旨在限制本说明书一个或多个实施例。在本说明书一个或多个实施例和所附权利要求书中所使用的单数形式的“一种”、“所述”和“该”也旨在包括多数形式，除非上下文清楚地表示其他含义。还应当理解，本文中使用的术语“和/或”是指并包含一个或多个相关联的列出项目的任何或所有可能组合。

[0223] 应当理解，尽管在本说明书一个或多个实施例可能采用术语第一、第二、第三等来描述各种信息，但这些信息不应限于这些术语。这些术语仅用来将同一类型的信息彼此区分开。例如，在不脱离本说明书一个或多个实施例范围的情况下，第一信息也可以被称为第二信息，类似地，第二信息也可以被称为第一信息。取决于语境，如在此所使用的词语“如果”可以被解释成为“在……时”或“当……时”或“响应于确定”。

[0224] 以上所述仅为本说明书一个或多个实施例的较佳实施例而已，并不用以限制本说明书一个或多个实施例，凡在本说明书一个或多个实施例的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本说明书一个或多个实施例保护的范围之内。

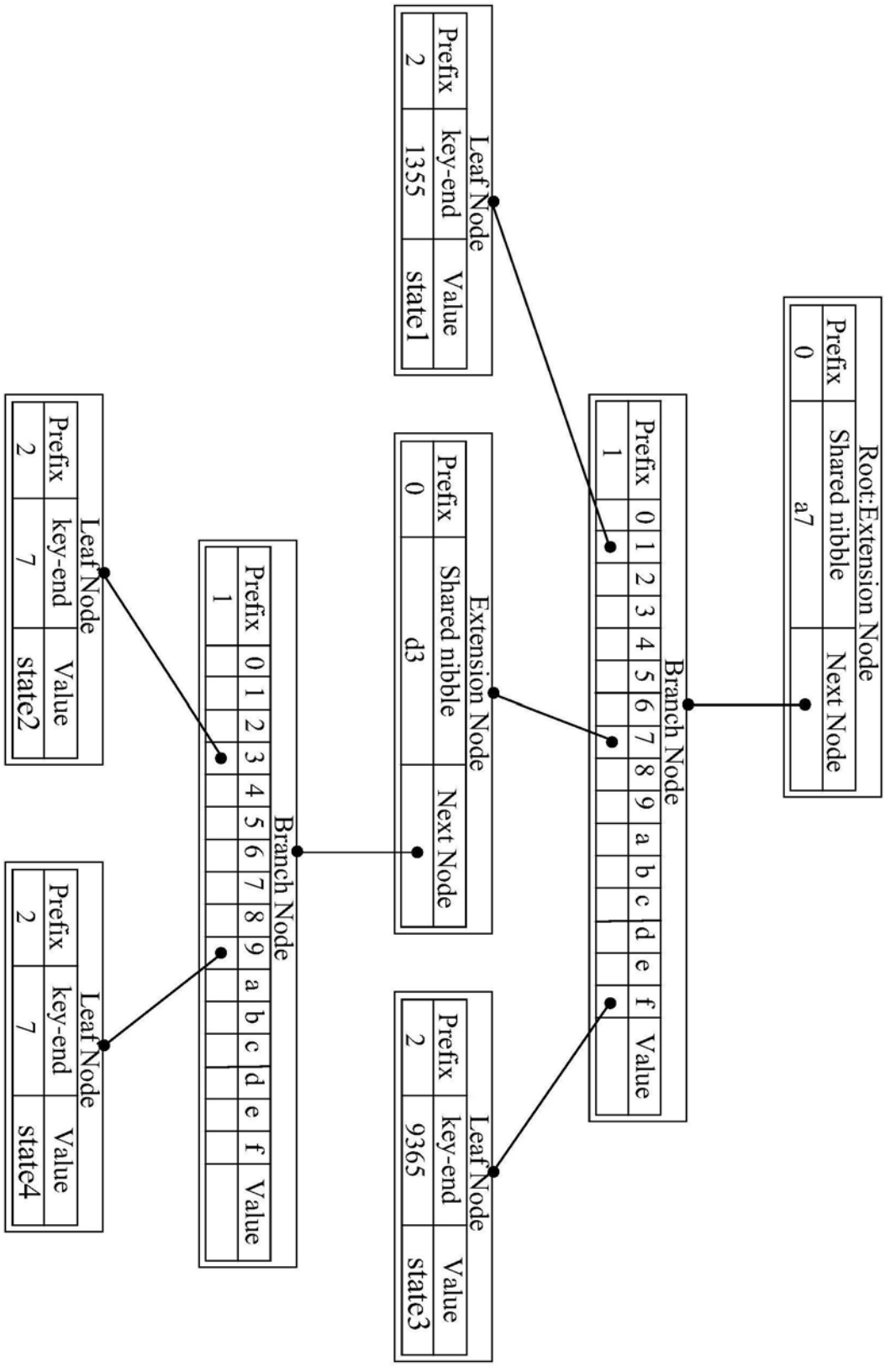

Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f
Value
1
Leaf Node
Prefix
key-end
Value
2
1355
state1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
9365
state3
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f
Value
1
Leaf Node
Prefix
key-end
Value
2
7
state2
Leaf Node
Prefix
key-end
Value
2
7
state4

图1

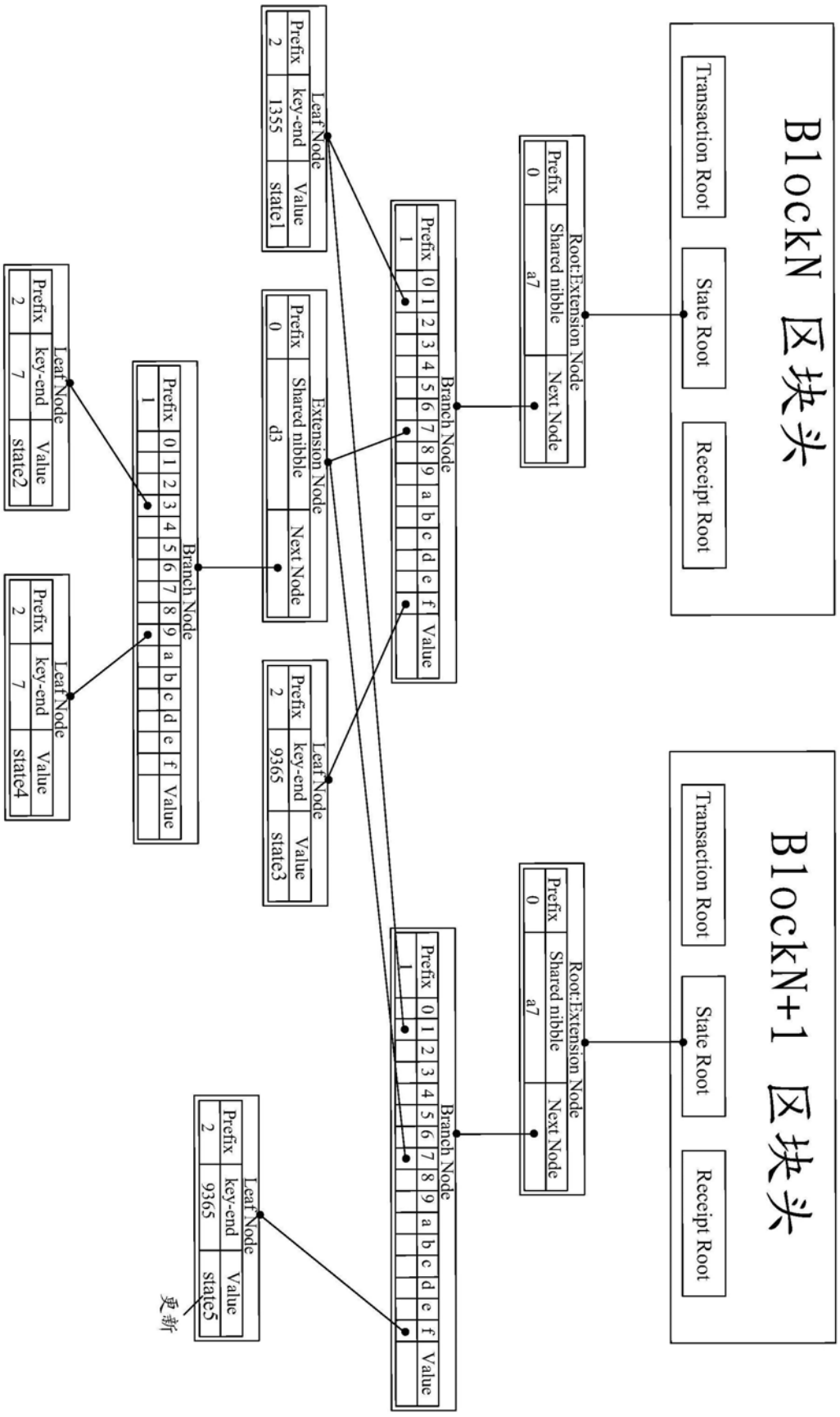

BlockN 区块头
Transaction Root
State Root
Receipt Root
Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Leaf Node
Prefix
key-end
Value
2
1355
state1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
9365
state3
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Leaf Node
Prefix
key-end
Value
2
7
state2
Leaf Node
Prefix
key-end
Value
2
7
state4
BlockN+1 区块头
Transaction Root
State Root
Receipt Root
Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Leaf Node
Prefix
key-end
Value
2
9365
state5
更新

图2

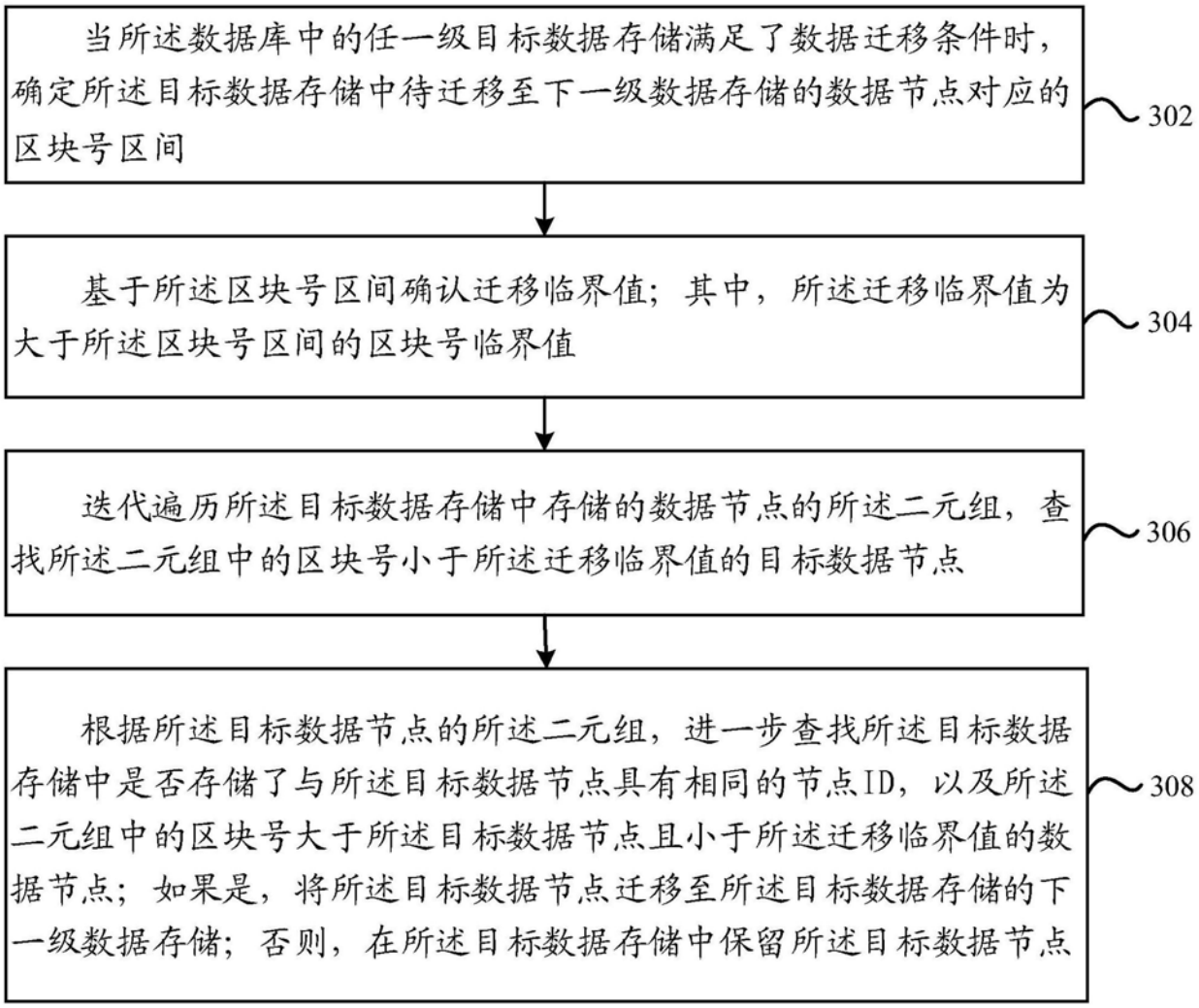

当所述数据库中的任一级目标数据存储满足了数据迁移条件时，确定所述目标数据存储中待迁移至下一级数据存储的数据节点对应的区块号区间
302
基于所述区块号区间确认迁移临界值；其中，所述迁移临界值为大于所述区块号区间的区块号临界值
304
迭代遍历所述目标数据存储中存储的数据节点的所述二元组，查找所述二元组中的区块号小于所述迁移临界值的目标数据节点
306
根据所述目标数据节点的所述二元组，进一步查找所述目标数据存储中是否存储了与所述目标数据节点具有相同的节点ID，以及所述二元组中的区块号大于所述目标数据节点且小于所述迁移临界值的数据节点；如果是，将所述目标数据节点迁移至所述目标数据存储的下一级数据存储；否则，在所述目标数据存储中保留所述目标数据节点
308

图3

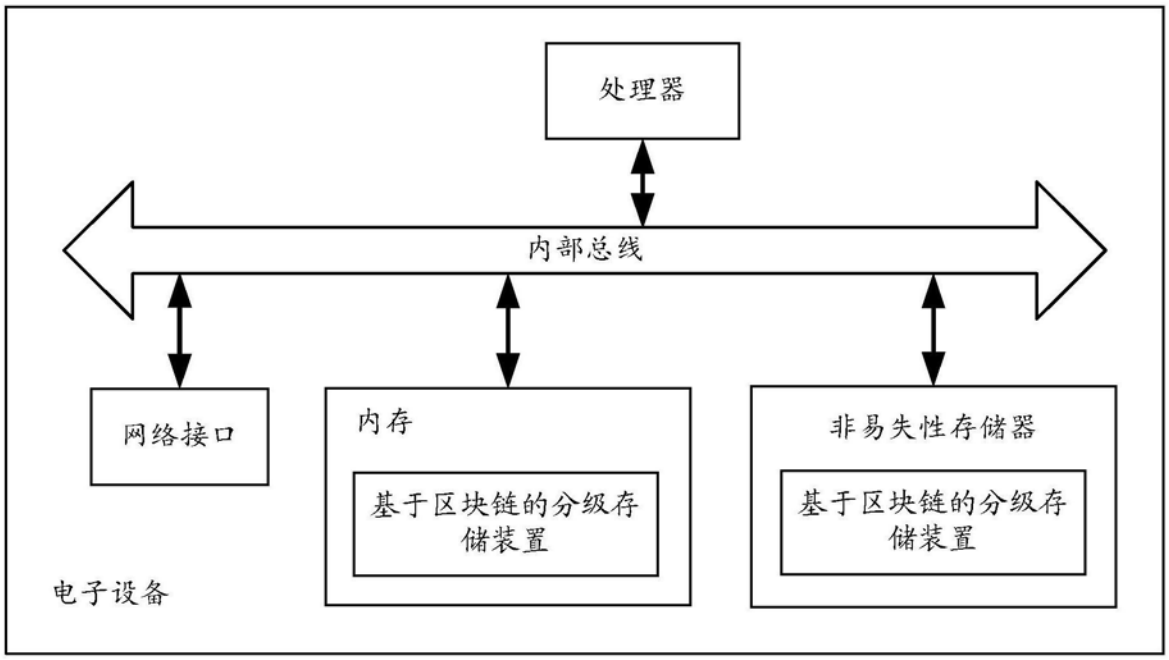

处理器
内部总线
网络接口
内存
基于区块链的分级存储装置
非易失性存储器
基于区块链的分级存储装置
电子设备

图4


基于区块链的分级存储装置50
确定模块501
查找模块502
迁移模块503

图5