(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 113220685 B
(45)授权公告日 2022.04.19

(21)申请号 202110510926.5
(22)申请日 2021.05.11
(65)同一申请的已公布的文献号
申请公布号 CN 113220685 A
(43)申请公布日 2021.08.06
(73)专利权人 支付宝(杭州)信息技术有限公司
地址 310000 浙江省杭州市西湖区西溪路556号8层B段801-11
(72)发明人 卓海振
(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 16/22(2019.01)
G06F 16/2455(2019.01)
G06F 16/27(2019.01)

(56)对比文件
US 2020195448 A1,2020.06.18
US 2020169402 A1,2020.05.28
US 2020379977 A1,2020.12.03
US 2020167345 A1,2020.05.28
CN 108846133 A,2018.11.20
CN 111737726 A,2020.10.02
CN 110334154 A,2019.10.15
CN 111488349 A,2020.08.04

审查员 蔡智勇

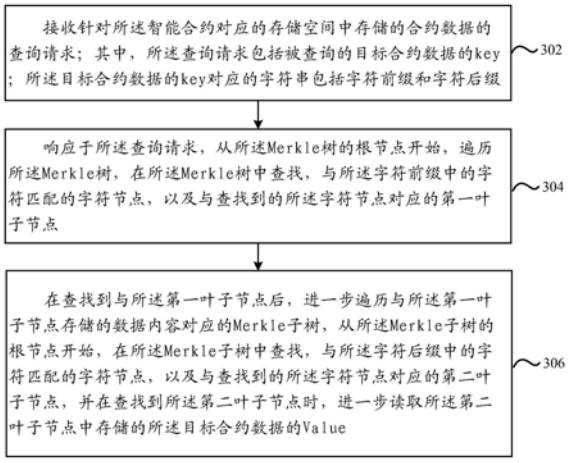
权利要求书3页 说明书23页 附图10页

(54)发明名称

智能合约存储内容的遍历方法及装置、电子设备

(57)摘要

一种智能合约存储内容的遍历方法，包括：接收针对智能合约中存储的合约数据的查询请求；查询请求包括目标合约数据的key；响应于查询请求，从与所述智能合约对应的Merk l e树的根节点开始，遍历Merk l e树，在Merk l e树中查找，与该key的字符前缀中的字符匹配的字符节点，以及与查找到的字符节点对应的第一叶子节点；在查找到第一叶子节点后，进一步遍历与第一叶子节点存储的数据内容对应的Merk l e子树，从Merk l e子树的根节点开始，在Merk l e子树中查找与该key的字符后缀中的字符匹配的字符节点，以及与该字符节点对应的第二叶子节点，并进一步读取第二叶子节点中存储的所述目标合约数据的Va l ue。



CN 113220685 B

1.一种智能合约存储内容的遍历方法,与所述智能合约对应的存储空间中存储的合约数据,以key-value键值对的形式被组织成Merkle树在数据库中存储;所述Merkle树的各叶子节点存储的数据内容,被进一步组织成与所述Merkle树对应的Merkle子树;其中,所述Merkle树以及所述Merkle子树均为融合了Trie字典树的树形结构的多层树状存储结构,包含至少一层字符节点和一层叶子节点;所述字符节点用于存储所述合约数据的key对应的字符串中的字符;所述叶子节点用于存储所述合约数据的Value;所述Merkle树和所述Merkle子树上的节点,通过在其上一层的节点中填充该节点的hash值,与其上一层的节点进行链接;

所述方法包括:

接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求;其中,所述查询请求包括被查询的目标合约数据的key;所述目标合约数据的key对应的字符串包括字符前缀和字符后缀;

响应于所述查询请求,从所述Merkle树的根节点开始,遍历所述Merkle树,在所述Merkle树中查找,与所述字符前缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第一叶子节点;

在查找到所述第一叶子节点后,进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树,从所述Merkle子树的根节点开始,在所述Merkle子树中查找,与所述字符后缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第二叶子节点,并在查找到所述第二叶子节点时,进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

2.根据权利要求1所述的方法,所述Merkle树中的字符节点,用于存储所述字符前缀中的字符;所述Merkle树的叶子节点,用于存储由所述字符后缀和所述合约数据的value构成的key-value键值对;

所述Merkle子树中的字符节点,用于存储所述字符后缀中的字符;所述Merkle子树的叶子节点,用于存储所述合约数据的value。

3.根据权利要求1所述的方法,当所述Merkle子树中的字符节点,存储了所述字符后缀中的部分字符时,所述Merkle子树的叶子节点,用于存储由所述字符后缀中的所述部分字符以外的剩余字符和所述合约数据的value构成的key-value键值对。

4.根据权利要求1所述的方法,所述Merkle树上的叶子节点的上一层字符节点中填充的该叶子节点的hash值,为与所述叶子节点存储的数据内容对应的Merkle子树的根节点的hash值。

5.根据权利要求1所述的方法,所述Merkle树;和/或,所述Merkle子树为MPT树。

6.根据权利要求1所述的方法,所述Merkle树;和/或,所述Merkle子树为FDMT树;

其中,所述FDMT树中的各字符节点,均包括分别代表不同字符的多个block;所述block进一步包括多个分别代表不同字符的槽位;所述槽位用于填充该字符节点链接的下一层节点的hash值。

7.根据权利要求6所述的方法,所述FDMT树中包含的字符节点的层数为固定值。

8.根据权利要求6所述的方法,所述FDMT树中的任一字符节点存储的字符,为所述字符节点中的block所代表的字符,与所述block中填充了hash值的槽位所代表的字符,进行拼

接生成的字符串。

9.根据权利要求6所述的方法,所述字符节点包括的block的数量,与所述block包括的槽位的数量相同。

10.根据权利要求9所述的方法,所述字符节点包括分别代表不同的16进制字符的16个block;所述block包括16个分别代表不同的16进制字符的槽位。

11.根据权利要求1所述的方法,所述Merkle树存储的合约数据的key,为所述合约数据的原始key进行hash计算得到的hash值。

12.根据权利要求11所述的方法,所述Merkle子树的叶子节点,还存储了所述合约数据的原始key和所述合约数据的value的对应关系;

读取所述第二叶子节点中存储的所述合约数据的Value,包括:

读取所述第二叶子节点中存储的所述目标合约数据的原始key和所述合约数据的Value,并将读取到的所述目标合约数据的原始key和所述目标合约数据的Value返回给与所述查询请求对应的查询方。

13.一种智能合约存储内容的遍历装置,与所述智能合约对应的存储空间中存储的合约数据,以key-value键值对的形式被组织成Merkle树在数据库中存储;所述Merkle树的各叶子节点存储的数据内容,被进一步组织成与所述Merkle树对应的Merkle子树;其中,所述Merkle树以及所述Merkle子树均为融合了Trie字典树的树形结构的多层树状存储结构,包含至少一层字符节点和一层叶子节点;所述字符节点用于存储所述合约数据的key对应的字符串中的字符;所述叶子节点用于存储所述合约数据的Value;所述Merkle树和所述Merkle子树上的节点,通过在其上一层的节点中填充该节点的hash值,与其上一层的节点进行链接;

所述装置包括:

接收模块,接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求;其中,所述查询请求包括被查询的目标合约数据的key;所述目标合约数据的key对应的字符串包括字符前缀和字符后缀;

查找模块,响应于所述查询请求,从所述Merkle树的根节点开始,遍历所述Merkle树,在所述Merkle树中查找,与所述字符前缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第一叶子节点;在查找到所述第一叶子节点后,进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树,从所述Merkle子树的根节点开始,在所述Merkle子树中查找,与所述字符后缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第二叶子节点,并在查找到所述第二叶子节点时,进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

14.根据权利要求13所述的装置,所述Merkle树中的字符节点,用于存储所述字符前缀中的字符;所述Merkle树的叶子节点,用于存储由所述字符后缀和所述合约数据的value构成的key-value键值对;

所述Merkle子树中的字符节点,用于存储所述字符后缀中的字符;所述Merkle子树的叶子节点,用于存储所述合约数据的value。

15.根据权利要求13所述的装置,当所述Merkle子树中的字符节点,存储了所述字符后缀中的部分字符时,所述Merkle子树的叶子节点,用于存储由所述字符后缀中的所述部分

字符以外的剩余字符和所述合约数据的value构成的key-value键值对。

16.根据权利要求13所述的装置,所述Merkle树上的叶子节点的上一层字符节点中填充的该叶子节点的hash值,为与所述叶子节点存储的数据内容对应的Merkle子树的根节点的hash值。

17.根据权利要求13所述的装置,所述Merkle树;和/或,所述Merkle子树为MPT树。

18.根据权利要求13所述的装置,所述Merkle树;和/或,所述Merkle子树为FDMT树;

其中,所述FDMT树中的各字符节点,均包括分别代表不同字符的多个block;所述block进一步包括多个分别代表不同字符的槽位;所述槽位用于填充该字符节点链接的下一层节点的hash值。

19.根据权利要求18所述的装置,所述FDMT树中包含的字符节点的层数为固定值。

20.根据权利要求18所述的装置,所述FDMT树中的任一字符节点存储的字符,为所述字符节点中的block所代表的字符,与所述block中填充了hash值的槽位所代表的字符,进行拼接生成的字符串。

21.根据权利要求18所述的装置,所述字符节点包括的block的数量,与所述block包括的槽位的数量相同。

22.根据权利要求21所述的装置,所述字符节点包括分别代表不同的16进制字符的16个block;所述block包括16个分别代表不同的16进制字符的槽位。

23.根据权利要求13所述的装置,所述Merkle树存储的合约数据的key,为所述合约数据的原始key进行hash计算得到的hash值;所述Merkle子树的叶子节点,还存储了所述合约数据的原始key和所述合约数据的value的对应关系;

读取所述第二叶子节点中存储的所述合约数据的Value,包括:

读取所述第二叶子节点中存储的所述目标合约数据的原始key和所述合约数据的Value,并将读取到的所述目标合约数据的原始key和所述目标合约数据的Value返回给与所述查询请求对应的查询方。

24.一种电子设备,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1-12中任一项所述的方法。

25.一种计算机可读存储介质,其上存储有计算机指令,该指令被处理器执行时实现如权利要求1-12中任一项所述方法的步骤。

智能合约存储内容的遍历方法及装置、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，尤其涉及一种智能合约存储内容的遍历方法及装置、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，是一种由若干台节点设备共同参与“记账”，共同存储和维护一份完整的分布式数据库的新兴技术。

[0003] 对于区块链的节点设备来说，需要存储和维护的区块链数据，通常包括区块数据、区块链中的区块链账户对应的账户状态数据；而区块数据又可以进一步包括区块头数据、区块中的区块交易数据、以及与区块中的区块交易数据对应的交易收据，等等。

[0004] 区块链的节点设备在存储以上示出的各种区块链数据时，通常可以将上述各种区块链数据以key-value键值对的形式，组织成Merkle树在数据库中存储。当需要查询节点设备存储的上述各种区块链数据时，可以通过将上述各种区块链数据的key作为查询索引，遍历上述Merkle树来高效的查询数据。

发明内容

[0005] 本说明书提出一种智能合约存储内容的遍历方法，与所述智能合约对应的存储空间中存储的合约数据，以key-value键值对的形式被组织成Merkle树在数据库中存储；所述Merkle树的各叶子节点存储的数据内容，被进一步组织成与所述Merkle树对应的Merkle子树；其中，所述Merkle树以及所述Merkle子树均为融合了Trie字典树的树形结构的多层树状存储结构，包含至少一层字符节点和一层叶子节点；所述字符节点用于存储所述合约数据的key对应的字符串中的字符；所述叶子节点用于存储所述合约数据的Value；所述Merkle树和所述Merkle子树上的节点，通过在其上一层的节点中填充该节点的hash值，与其上一层的节点进行链接；

[0006] 所述方法包括：

[0007] 接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求；其中，所述查询请求包括被查询的目标合约数据的key；所述目标合约数据的key对应的字符串包括字符前缀和字符后缀；

[0008] 响应于所述查询请求，从所述Merkle树的根节点开始，遍历所述Merkle树，在所述Merkle树中查找，与所述字符前缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第一叶子节点；

[0009] 在查找到所述第一叶子节点后，进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树，从所述Merkle子树的根节点开始，在所述Merkle子树中查找，与所述字符后缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第二叶子节点，并在查找到所述第二叶子节点时，进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

[0010] 本说明书还提出一种智能合约存储内容的遍历装置，与所述智能合约对应的存储空间中存储的合约数据，以key-value键值对的形式被组织成Merkle树在数据库中存储；所述Merkle树的各叶子节点存储的数据内容，被进一步组织成与所述Merkle树对应的Merkle子树；其中，所述Merkle树以及所述Merkle子树均为融合了Trie字典树的树形结构的多层树状存储结构，包含至少一层字符节点和一层叶子节点；所述字符节点用于存储所述合约数据的key对应的字符串中的字符；所述叶子节点用于存储所述合约数据的Value；所述Merkle树和所述Merkle子树上的节点，通过在其上一层的节点中填充该节点的hash值，与其上一层的节点进行链接；

[0011] 所述装置包括：

[0012] 接收模块，接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求；其中，所述查询请求包括被查询的目标合约数据的key；所述目标合约数据的key对应的字符串包括字符前缀和字符后缀；

[0013] 查找模块，响应于所述查询请求，从所述Merkle树的根节点开始，遍历所述Merkle树，在所述Merkle树中查找，与所述字符前缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第一叶子节点；在查找到所述第一叶子节点后，进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树，从所述Merkle子树的根节点开始，在所述Merkle子树中查找，与所述字符后缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第二叶子节点，并在查找到所述第二叶子节点时，进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

[0014] 以上技术方案中，由于将存储合约数据的Merkle树的叶子节点上存储的数据内容，也组织成了与该Merkle树对应的Merkle子树；因此，在遍历叶子节点中存储的数据内容进行合约数据的查询时，对叶子节点中存储的数据内容的遍历，将转换成对一颗Merkle子树的遍历，因此可以提升针对该叶子节点中存储的数据内容的遍历效率。

附图说明

[0015] 图1是一示例性实施例提供的一种将区块链数据组织成MPT状态树的示意图；

[0016] 图2是一示例性实施例提供的将合约账户对应的存储空间中存储的合约数据组织成MPT storage树的示意图；

[0017] 图3是一示例性实施例提供的一种智能合约存储内容的遍历方法的流程图；

[0018] 图4是一示例性实施例提供的一种将两层Merkle Tree的架构扩展为三层Merkle Tree的架构的示意图；

[0019] 图5是一示例性实施例提供的一种FDMT树的树形结构图；

[0020] 图6是一示例性实施例提供的一种字符节点的结构图；

[0021] 图7是一示例性实施例提供的一种bucket数据桶的结构图；

[0022] 图8是一示例性实施例提供的另一种bucket数据桶的结构图；

[0023] 图9是一示例性实施例提供的一种为Merkle树上的节点设置节点ID的示意图；

[0024] 图10是一示例性实施例提供的一种电子设备的结构示意图；

[0025] 图11是一示例性实施例提供的一种智能合约存储内容的遍历智能合约存储内容的遍历装置的框图。

具体实施方式

[0026] 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。

[0027] 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。

[0028] 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，还可以有上述多种类型的结合，比如私有链+联盟链、联盟链+公有链等。

[0029] 其中，去中心化程度最高的是公有链。加入公有链的参与者(也可称为区块链中的节点)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。而且，各节点可自由加入或者退出网络，并进行相关操作。

[0030] 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，其对节点具有严格限制且节点数量较少。这种类型的区块链更适合于特定机构内部使用。

[0031] 联盟链则是介于公有链以及私有链之间的区块链，可实现“部分去中心化”。联盟链中各个节点通常有与之相对应的实体机构或者组织；节点通过授权加入网络并组成利益相关联盟，共同维护区块链运行。

[0032] 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。

[0033] 对于链外产生的数据，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对该交易进行共识，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。

[0034] 在区块链领域，有一个重要的概念就是账户(Account)；在实际应用中，通常可以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，也称之为用户账户；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)。

[0035] 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。

[0036] 在一个例子中，账户的结构体通常包括Balance，Nonce，Code和Storage等字段。其中：

[0037] Balance字段，用于维护账户目前的账户余额；

[0038] Nonce字段,用于维护该账户的交易次数;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;

[0039] 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。

[0040] Storage字段,用于维护该账户的存储内容;对于合约账户而言,通常会分配一个独立的持久化的存储空间,用以存储该合约账户对应的存储空间中存储的合约数据;该独立的存储空间通常称之为该合约账户的账户存储。

[0041] 合约账户的存储内容通常会以key-value键值对的形式被构建成MPT(Merkle Patricia Trie)树的数据结构存储在上述独立的存储空间之中。MPT树是区块链领域用于存储和维护区块链数据的一种逻辑的树形结构,在这种树形结构中通常包括根节点、中间节点、叶子节点。

[0042] 其中,基于合约账户的存储内容构建成的MPT树,通常也称之为Storage树。而Storage字段通常仅维护该Storage树的根节点的hash值;因此,Storage字段通常也称之为Storage Root hash字段。其中,对于外部账户而言,以上示出的Code字段和Storage字段的字段值均为空值。

[0043] 对于大多数区块链模型,通常都会采用Merkle树;或者,基于Merkle树的数据结构的Merkle树变种等逻辑的树形结构,来存储和维护数据。例如,MPT树,就是一种用来存储和维护区块链数据的,融合了Trie字典树的树形结构的Merkle树变种。

[0044] 以下以使用MPT树,来存储区块链数据为例进行说明;

[0045] 在一个例子中,区块链中需要存储和维护的区块链数据,通常包括账户状态(state)数据、交易数据和收据数据;因此,在实际应用中,可以分别将上述账户状态数据、交易数据和收据数据以key-value键值对的形式,组织成MPT状态树(即world state)、MPT交易树和MPT收据树等三棵MPT树,分别进行存储和维护。

[0046] 其中,除了以上三棵MPT树以外,合约账户对应的存储空间中存储的合约数据,通常也会被构建成为一棵MTP Storage树(以下简称为Storage树)。该Storage树的根节点的hash值,会被添加到与该Storage树对应的合约账户的上述结构体中的Storage字段。

[0047] MPT状态树,是由区块链中所有账户(包括外部账户和合约账户)的账户状态(state)数据,以key-value键值对的形式组织成的MPT树;MPT交易树,是由区块链中的交易(transaction)数据,以key-value键值对的形式组织成的MPT树;MPT收据树,是区块中的交易在执行完毕后生成的与每笔交易对应的交易(receipt)收据,以key-value键值对的形式组织成的MPT树。

[0048] 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,最终都会被添加至对应区块的区块头中。

[0049] 其中,MPT交易树和MPT收据树均与区块相对应,即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。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,区块链中这些被执行交易的相关账户(可以是外部账户也可以是合约账户)的账户状态,通常也会随之发生变化。

[0050] 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字段的字段值),通常也会随之发生变化。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状态树,用于维护区块链中所有账户的最新状态。

[0051] 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,导致区块链中的部分账户的账户状态发生了变化,节点设备需要基于区块链中所有账户最新的账户状态数据,重新构建一棵MPT状态树。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。

[0052] 请参见图1,图1为本说明书示出的一种将区块链中的各个区块链账户的账户状态数据以key-value键值对的形式组织成MPT状态树的示意图。

[0053] MPT树,是一种较为传统的经过改良的Merkle树变种,其融合了Merkle树和Trie字典树(也称之为前缀树)两种树形结构的优点。

[0054] 在MPT树中通常包括三种节点,分别为叶子节点(leaf node),扩展节点(extension node)和分支节点(branch node)。其中,MPT树的根节点通常可以是扩展节点;MPT树的中间节点通常可以是分支节点或者其它的扩展节点。

[0055] 其中,扩展节点和分支节点可以统称为字符节点,用于存储账户状态数据的key(即账户地址)对应的字符串的字符前缀部分;其中,对于MPT树而言,上述字符前缀部分通常是指共享字符前缀;所述共享字符前缀,是指所有账户状态数据的key(即区块链账户地址)所具有的相同的一个或者多个字符组成的前缀。而上述叶子节点,用于存储区块链数据的key对应的字符串的字符后缀部分和Value(即具体的账户状态数据)。

[0056] 扩展节点,用于存储账户地址的共享字符前缀中一个或者多个字符(即图1示出的shared nibble),和该扩展节点链接的下一层的节点的hash值(即图1示出的Next node)。

[0057] 分支节点,包含17个槽位,前16个槽位对应着key中的16个可能的十六进制字符,一个字符对应一个nibble(半字节),前16个槽位中的每一个槽位,分别表示一个账户地址的共享字符前缀中的一个字符,这些槽位用于填充该分支节点链接的下一层的节点的hash值。最后一个槽位为value槽位,一般为空值。

[0058] 叶子节点,用于存储账户地址的字符后缀(即图1示出的key-end),和账户状态数据的value(即以上描述的账户的结构体);其中,账户地址的字符后缀和账户地址的共享字符前缀共同组成了一个完整的账户地址;所述字符后缀,是指除了账户地址的共享字符前缀以外的最后一个或者多个字符组成的后缀。

[0059] 假设需要组织成MTP状态树的账户状态数据如下表1所示:

[0060]

账户地址（Key）								账户状态（Value）			
								balance	nonce	codehash	storage root
a	7	1	1	3	5	5		45.0ETH	n1		
a	7	7	d	3	3	7		1.00WEI	n2		
a	7	f	9	3	6	5		1.1ETH	n3		
a	7	7	d	3	9	7		0.12ETH	n4	c1	s1

[0061] 表1

[0062] 其中，需要说明的是，在表1中，前三行的账户地址对应的区块链账户为外部账户，Codehash和Storage root字段为空值。第4行的账户地址对应的区块链账户为合约账户，Codehash字段维护了该合约账户对应的合约代码的hash值；Storage root字段维护了该合约账户的存储内容构成的Storage树的根节点的hash值。

[0063] 最终按照表1中的账户状态数据组织成的MPT状态树，如图1所示；该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(其中一个扩展节点作为根节点)构成。

[0064] 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字段的不同字段值可以用于表示不同的节点类型。

[0065] 例如，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。prefix字段的取值为1，表示包含奇数个nibble(s)的扩展节点；prefix字段的取值为2，表示包含偶数个nibbles的叶子节点；prefix字段的取值为3，表示包含奇数个nibble(s)的叶子节点。

[0066] 而分支节点，由于其是并列单nibble的字符节点，因此分支节点不具有上述prefix字段。

[0067] 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)。

[0068] 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层节点的hash值。

[0069] 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(账户地址的字符后缀)。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码后，填充至叶子节点的Value字段。

[0070] 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；

[0071] 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，可以为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。

[0072] 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。

[0073] 由于MPT状态树上的node，以Key-value键值对的形式进行存储；其中，Key可以是node所包含的数据内容的hash值，Value可以是node所包含的数据内容；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。

[0074] 请参见图2，图2为本说明书示出的一种将合约账户对应的存储空间中存储的合约数据组织成MPT storage树的示意图。

[0075] 请继续参见表1，表1中示出的账户地址为“a77d397”的账户为合约账户，因此该合约账户对应的存储空间中存储的合约数据会被组织成一颗storage树；其中，该storage树的根节点的hash值S1，会被添加到图1示出的MTP状态树中与该合约账户对应的叶子节点中的storage root字段里。

[0076] 假设该合约账户的存储空间中存储的合约数据如下表2所示：

[0077]

Key							Value
3	3	5	b	2	e	4	张三的资产A余额(状态变量)：张三_A=20
7	c	2	5	9	8	8	李四的资产B余额(状态变量)：李四_B=50
f	a	9	9	3	6	5	王五的资产A余额(状态变量)：王五_A=35
f	a	6	b	e	3	3	staff（状态变量）：staff=new

[0078] 表2

[0079] 需要说明的是，合约账户的存储空间中存储的合约数据，通常可以是状态变量的形式；在进行存储时，可以将状态变量名以key-value键值对的形式组织成如图2所示的storage树进行存储。

[0080] 例如，在一个例子中，可以将合约账户的账户地址和状态变量在合约账户的账户存储中的存储位置的hash值作为key，将状态变量对应的变量取值作为value。

[0081] 其中，图2示出的storage树的基本结构与图1示出的MTP状态树相似，在本说明书中不再赘述。通过上述图1和图2的描述可知，基于MPT树的树形结构设计，分支节点可以存储所有账户地址的共享字符前缀中的其中一个字符；而扩展节点则可以存储所有账户地址的共享字符前缀中的一个或者多个字符。

[0082] 通过上述图1和图2的描述可知，在将区块链数据组织成Merkle树进行存储的区块链平台中，通常采用的是两层Merkle Tree的架构；其中，上层Merkle Tree，是由所有区块链账户对应的状态数据组织成的状态树，下层是上述状态树上的各个合约账户对应的账户存储空间中存储的合约数据组织成的storage树。

[0083] 在实际应用中，无论是上述MPT状态树上的叶子节点，还是上述storage树上的叶子节点，通常采用的都是bucket数据桶的结构，可以存储若干条key-value键值对；

[0084] 例如，对于上述MPT状态树上的叶子节点而言，可以存储若干条由账户地址的字符后缀作为key和对应的账户状态数据作为value，构成的key-value键值对。对于上述storage树上的叶子节点而言，可以存储若干条由状态变量名的hash值作为key，将状态变量名对应的变量取值作为value，构成的key-value键值对。

[0085] 其中，由于合约账户相对于外部账户，通常具有更大的存储空间；因此，在合约账户对应的账户存储空间中，通常会存储大量的用户数据，并且基于实际的业务需求，通常还可能需要对上述storage树上的叶子节点存储的若干条key-value键值对进行遍历；

[0086] 例如，以基于智能合约进行数据存证为例，用户可以通过调用智能合约的方式，将需要存证的用户数据提交给智能合约，在智能合约对应的账户存储空间中进行数据存证。当用户在需要查询智能合约对应的账户存储空间中存证的用户数据时，可以在上述storage树上查找数据所在的叶子节点，再遍历该叶子节点存储的若干条key-value键值对。

[0087] 在实际应用中，在将合约数据以key-value键值对的形式组织成上述storage树时，出于安全性的考量，合约数据对应的key-value键值对中的key，通常并不采用合约数据的原始key，而是采用合约数据的原始key的hash值作为key；

[0088] 例如，合约数据可以不再以<key,Value>的形式组织成上述storage树；而是以<Hash(key),Value>的形式组织成上述storage树。

[0089] 而对于上述状态树而言，在将各区块链的账户状态数据以key-value键值对的形式组织成上述状态树时，也是出于安全性的考量，账户状态数据对应的key-value键值对中的key，通常也并不采用账户状态数据的原始key，而是采用账户状态数据的原始key的hash值作为key，不再赘述。

[0090] 将上述storage树上存储的合约数据对应的key-value键值对中的key，替换为原始hey的hash值，虽然可以提升上述storage树上存储的数据的数据安全，确保合约数据的原始key不会泄露；但由于hash值本身具有无序和离散性；因此。上述storage树上的叶子节点上存储的所有的key-value键值对，也是无序且离散的，这势必会对上述storage树上的叶子节点存储的若干条key-value键值的遍历造成影响，进而对实际的业务需求也造成影响基于此，本说明书提出一种将存储合约数据的Merkle树的叶子节点上存储的数据内容，组织成与Merkle树对应的Merkle子树，以提高对上述叶子节点上存储的数据内容进行遍历的效率的技术方案。

[0091] 在实现时，可以将图1和图2示出的，两层Merkle Tree的架构，转变成三层Merkle Tree的架构。

[0092] 对于与智能合约对应的存储空间中存储的合约数据，仍然可以以key-value键值对的形式被组织成Merkle树在数据库中存储；其中，该Merkle树的各叶子节点存储的数据内容，可以被进一步组织成与该Merkle树对应的Merkle子树；

[0093] 上述Merkle树以及上述Merkle子树，仍然可以是类似于MPT树的，融合了Trie字典树的树形结构的多层树状存储结构，可以包含至少一层字符节点和一层叶子节点；上述字符节点用于存储所述合约数据的key对应的字符串中的至少部分字符；上述叶子节点用于

存储所述合约数据的Value；

[0094] 例如，在一个例子中，上述合约数据的key对应的字符串可以包括字符前缀和字符后缀；上述Merkle树的字符节点，可以用于存储上述合约数据的key的字符前缀中的字符；上述Merkle树的叶子节点，可以用于存储由上述合约数据的key的字符后缀和上述合约数据的value构成的key-value键值对；相应的，上述Merkle子树的字符节点，则可以用于存储上述字符后缀中的、字符；上述Merkle子树的叶子节点，可以用于存储上述合约数据的value。

[0095] 在针对上述智能合约对应的存储空间中存储的合约数据进行数据查询时，可以接收针对上述合约数据的查询请求；其中，该查询请求包括被查询的目标合约数据的key；上述合约数据的key对应的字符串可以包括字符前缀和字符后缀；

[0096] 进一步的，从上述Merkle树的根节点开始，遍历上述Merkle树，在上述Merkle树中查找，与上述字符前缀中的字符匹配的字符节点，以及与查找到的上述字符节点对应的第一叶子节点；

[0097] 在查找到与上述第一叶子节点后，可以进一步遍历与上述第一叶子节点存储的数据内容对应的Merkle子树，从上述Merkle子树的根节点开始，在上述Merkle子树中查找，与上述字符后缀中的字符匹配的字符节点，以及与查找到的上述字符节点对应的第二叶子节点，并在查找到上述第二叶子节点时，进一步读取上述第二叶子节点中存储的上述目标合约数据的Value。

[0098] 在以上技术方案中，由于将存储合约数据的Merkle树的叶子节点上存储的数据内容，也组织成了与该Merkle树对应的Merkle子树；因此，在遍历叶子节点中存储的数据内容进行合约数据的查询时，对叶子节点中存储的数据内容的遍历，将转换成对一颗Merkle子树的遍历，因此可以提升针对该叶子节点中存储的数据内容的遍历效率。

[0099] 请参见图3，图3是一示例性实施例提供的一种智能合约存储内容的遍历方法的流程图。所述方法应用于区块链节点设备；与所述智能合约对应的存储空间中存储的合约数据，以key-value键值对的形式被组织成Merkle树在数据库中存储；其中，所述Merkle树的各叶子节点存储的数据内容，被进一步组织成与所述Merkle树对应的Merkle子树；所述方法包括以下步骤：

[0100] 步骤302，接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求；其中，所述查询请求包括被查询的目标合约数据的key；所述合约数据的key对应的字符串包括字符前缀和字符后缀；

[0101] 步骤304，响应于所述查询请求，从所述Merkle树的根节点开始，遍历所述Merkle树，在所述Merkle树中查找，与所述字符前缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第一叶子节点；

[0102] 步骤306，在查找到所述第一叶子节点后，进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树，从所述Merkle子树的根节点开始，在所述Merkle子树中查找，与所述字符后缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第二叶子节点，并在查找到所述第二叶子节点时，进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

[0103] 在本说明书中，上述Merkle树，具体可以包括融合了Trie字典树的树形结构的任

意形式的Merkle树变种;例如,在实际应用中,上述Merkle树,具体可以是上述图1和图2描述的MPT树;或者,也可以是其它类型的融合了Trie字典树的树形结构的Merkle树变种。

[0104] 区块链中的节点设备,可以将智能合约对应的存储空间中存储的合约数据,以key-value键值对的形式组织成Merkle树(即上述Storage树)进行存储;

[0105] 例如,在初始状态下,区块链节点设备可以在内存中初始化出一棵空的Merkle树,在获取到需要存储的合约数据后,可以将该合约数据处理成key-value键值对,然后将该key-value键值对写入该空的Merkle树,最后再通过执行诸如commit命令,将内存中的该Merkle树上的节点持久化存储到数据库中。

[0106] 需要说明是,与以上描述的MPT树类似,上述Merkle树仍然是一棵逻辑树,上述Merkle树上的节点的存储结构也均为逻辑存储结构;

[0107] 其中,所谓逻辑的树形结构,是指在数据库的底层物理存储中,并不存在与树形结构完全对应的物理存储结构,而仅在数据库中存储上述树形结构上的各个节点的物理数据以及各个节点之间的链接关系数据,从而可以基于数据库中存储的各个节点的物理数据和链接关系数据,在逻辑层面上还原出上述树形结构。

[0108] 上述合约数据的key,具体可以是指合约数据在数据库中对应的查找键值;实际应用中,上述查找键值具体可以是一个与合约数据对应的字符串;例如,上述查找键值具体可以是合约地址对应的字符串;相应的,上述合约数据的Value,具体可以是上述合约数据的原始内容;例如,在一个例子中,上述key具体可以是状态变量名的hash值;上述Value具体可以是状态变量的变量取值。

[0109] 在本说明书中,上述合约数据的key,仍然可以包括字符前缀部分(Shared nibble)和字符后缀部分(key-end);其中,合约数据的key的字符前缀部分,可以是该合约数据的key与其它合约数据的key的共享字符前缀;例如,请参见上述表2中的合约数据“fa99365”和“fa6be33”就具有共享字符前缀“f”。相应的,上述合约数据的key的字符后缀部分,则可以是合约数据的key对应的字符串中,除了上述字符前缀部分以外的字符部分。

[0110] 当然,在实际应用中,合约数据的key的字符前缀部分,也可以不是共享字符前缀;例如,对于MPT树而言,上述key的字符前缀部分通常是共享字符前缀,而对于其它类型的融合了Trie字典树的树形结构的Merkle树变种,上述key的字符前缀部分也可以不是共享字符前缀。

[0111] 需要说明的是,上述合约数据的key的字符前缀部分的字符长度,也可以与上述合约数据的key对应的字符长度完全相同;在这种情况下,上述合约数据的key的字符后缀部分,可以为空值(即不包含后缀部分);此时,上述叶子节点可以只存储该合约数据对应的value。

[0112] 在本说明书中,上述Merkle树的前N层,可以是用于存储上述合约数据的key对应的字符串中的字符的字符节点;上述Merkle树的最后一层,可以是用于存储上述合约数据的value的叶子节点。

[0113] 例如,在一个例子中,上述Merkle树的前N层的字符节点,也可以存储上述合约数据的key对应的字符串中的至少部分字符;则上述Merkle树的最后一层的叶子节点,则可以存储上述至少部分字符以外的剩余字符和上述合约数据的value构成的key-value键值对;

[0114] 在实际应用中,上述叶子节点通常可以采用bucket数据桶的结构,可以存储若干

条用于存储上述合约数据的value的数据记录。

[0115] 例如,在一个例子中,仍以上述Merkle树的前N层的字符节点,存储上述合约数据的key对应的字符串中的至少部分字符为例,在这种情况下,上述数据记录包括由上述至少部分字符以外的剩余字符和上述合约数据的value构成的key-value键值对。

[0116] 请继续参见图1和图2,在实际应用中,在将区块链数据组织成Merkle树进行存储的区块链平台中,通常采用的是两层Merkle Tree的架构;上层Merkle Tree,是由所有区块链账户对应的状态数据组织成的状态树,下层Merkle Tree是上述状态树上的各个合约账户对应的账户存储空间中存储的合约数据组织成的storage树。

[0117] 在本说明书中,为了提升针对上述storage树上的叶子节点上存储的数据内容进行遍历的效率,可以将上述两层Merkle Tree的架构,进一步扩展为三层Merkle Tree的架构。

[0118] 请参见图4,图4为本说明书中示出的一种将上述两层Merkle Tree的架构,进一步扩展为三层Merkle Tree的架构的示意图;

[0119] 如图4所示,在如上述图1和图2示出的两层Merkle Tree的架构的基础上,可以进一步将上述storage树的各叶子节点存储的数据内容,分别组织成与上述storage树对应的storage子树(即上述Merkle子树);

[0120] 例如,在示出的一种实施方式中,由于上述合约数据的key通常包括字符前缀部分和字符后缀部分;因此,上述storage树中的字符节点存储的数据内容,可以是上述字符前缀中的字符;上述storage树的叶子节点存储的数据内容,则可以用于存储由上述字符后缀和上述合约数据的value构成的key-value键值对。

[0121] 区块链节点设备在将上述storage树的叶子节点存储的数据内容组织成storage子树时,可以在内存中初始化出一棵空的Merkle树作为上述storage子树,并将原本应当在该叶子节点中存储的上述key-value键值对,进一步写入上述空的storage子树,最后再通过执行诸如commit命令,将内存中的该storage子树持久化存储到数据库,进而可以将上述图1和图2示出的两层Merkle Tree的架构扩展为如图4所示的三层Merkle Tree的架构。

[0122] 请继续参见图4,图4示出的三层Merkle Tree的架构,上层Merkle Tree,仍然是由所有区块链账户对应的状态数据组织成的状态树,中间层是上述状态树上的各个合约账户对应的账户存储空间中存储的合约数据组织成的storage树;而最下层是上述storage树上的各个叶子节点存储的数据内容组织成的与上述storage树对应的storage子树。其中,上述storage子树的数量,通常与上述storage树包含的叶子节点的数量对应。

[0123] 其中,上述storage子树的存储结构与上述storage树相同,上述storage子树的前N层,仍然可以是用于存储上述合约数据的key对应的字符串中的字符的字符节点;上述storage子树的最后一层,可以是用于存储上述合约数据的value的叶子节点。

[0124] 在示出的一种实施方式中,由于上述合约数据的key通常包括字符前缀部分和字符后缀部分;因此,上述storage树中的字符节点,可以用于存储上述字符前缀中的字符;上述storage树的叶子节点,则可以用于存储由上述字符后缀和上述合约数据的value构成的key-value键值对。

[0125] 相应的,上述storage子树中的字符节点,则可以用于存储上述字符后缀中的字符;上述storage子树的叶子节点,则可以用于存储上述合约数据的value。

[0126] 其中,需要说明的是,在一个例子中,上述storage子树的字符节点,也可以用于存储上述字符后缀中的部分字符;在这种情况下,如果上述storage树中的字符节点,存储了上述字符后缀中的部分字符时,上述storage子树的叶子节点,则可以用于存储由上述字符后缀中的上述部分字符以外的剩余字符和上述合约数据的value构成的key-value键值对。

[0127] 在本说明书,上述Storage树的根节点的hash值,通常会被写入上述状态树中,与上述Storage树对应的合约账户的账户结构体中的Storage字段;相应的,在将上述Storage树中的叶子节点存储的数据内容组织成与上述Storage树对应的storage子树之后,也可以将该storage子树的根节点的hash值,填充到与该该叶子节点链接的上一层字符节点中。

[0128] 例如,在实现时,与上述Storage树上的叶子节点链接的上一层前缀节点,其填充的通常是该叶子节点的hash值;而在将上述Storage树中的叶子节点存储的数据内容组织成与上述Storage树对应的storage子树之后,可以将与该叶子节点链接的上一层字符节点中填充的该叶子节点的hash值,替换为该storage子树的根节点的hash值。

[0129] 其中,需要说明的是,与上述storage树类似,上述storage子树,具体也可以包括融合了Trie字典树的树形结构的任意形式的Merkle树变种;

[0130] 在示出的一种实施方式中,上述storage树和上述storage子树,均可以是上述图1和图2描述的MPT树;或者,至少上述storage树和上述storage子树中的其中之一可以是上述MPT树,而另一可以是其它类型的融合了Trie字典树的树形结构的Merkle树变种。

[0131] 也即,在实际应用中,上述storage树和上述storage子树可以采用相同类型的上述Merkle树变种,也可以采用不同类型的上述Merkle树变种。

[0132] 在示出的一种实施方式中,上述storage树和上述storage子树,也可以是FDMT(Fixed Depth Merkle Tree)树;或者,至少上述storage树和上述storage子树中的其中之一可以是上述FDMT树,而另一可以是其它类型的融合了Trie字典树的树形结构的Merkle树变种(比如MPT树)。其中,FDMT树,是一种为了避免前N层的字符节点频繁发生分裂,而提出的一种前N层的字符节点的层数固定的,融合了Trie字典树的树形结构的Merkle树变种。

[0133] 请参见图5,图5为本说明书示出的一种FDMT树的树形结构图。

[0134] 如图5所示,在本说明书示出的FDMT树的树形结构中,包含前N层(图5示出的为3层,仅为示意)的字符节点(即图3中示出的Tree node),和最后一层的叶子节点(即图3中示出的Leaf node)。

[0135] 其中,与以上描述的MPT树不同的是,上述FDMT树前N层的各个字符节点,将采用统一的数据结构,从而使得上述FDMT树前N层的各个字符节点,将不再发生节点分裂。

[0136] 如图5所示,上述FDMT树上的前N层的字符节点,均可以包括分别代表不同字符的多个block;而每一个block可以进一步包括多个分别代表不同字符的槽位;例如,图5示出的为每一个字符节点包括N个block;每个block进一步包括N个slot。

[0137] 其中,上述FDMT树上各层的节点之间,仍然可以采用在上一层的节点中填充下一层的节点的hash值(hash指针)的方式,来进行节点间的链接。也即,该FDMT树上的节点仍然通过其hash值链接至其上一层的节点;

[0138] 需要解释的是,图5中示出的上述FDMT树上各层的节点之间的链接关系仅为示意性的,并不是指对上述FDMT树上各层的节点之间的链接关系的一种特殊限定。

[0139] 相应的,上述槽位具体可以用于填充当前的字符节点所链接的下一层节点的hash

值。需要说明的是，字符节点的下一层节点，具体仍然可以是下一层的字符节点，也可以是下一层的叶子节点。

[0140] 请继续参见图5，图5示出的上述FDMT树上的各字符节点，均可以用于存储上述区块链数据的key的字符前缀中的至少部分字符。各字符节点实际存储的字符，具体可以是该字符节点中的block(即至少有一个槽位填充了hash值的非空block)所代表的字符，与该block中填充了hash值的槽位(即非空槽位)所代表的字符，进行拼接生成的字符串。

[0141] 其中，需要说明的是，在实际应用中，字符节点中的每一个block，可以仅代表一位字符；也即，基于图5示出的字符节点的具体存储格式，每一个字符节点实际存储的上述区块链数据的key的字符前缀中的部分字符，为长度是2位字符的字符串。

[0142] 例如，请参见图6，图6为本说明书示出的一种字符节点的结构图；如图6所示，该字符节点包含16个代表不同的16进制字符的block；每一个block进一步包括16个分别代表不同的16进制字符的slot(图6中只示出了block6包含的16个槽位)；假设该字符节点中的block6(代表了16进制字符6)为非空block，该block中的slot4(代表了16进制字符4)、slot6(代表了16进制字符6)和slot9(代表了16进制字符9)为填充了该字符节点链接的下一层节点的hash值的非空slot；则该字符节点存储的上述区块链数据的key的字符前缀中的部分字符，分别为16进制字符串“64”、“66”和“69”。

[0143] 其中，上述字符节点中所包含的block的数量，以及每一个block所包含的槽位的数量，在本说明书中不进行特别限定；在实际应用中，可以基于上述区块链数据的key对应的字符串所包含的字符元素的类型数，来确定上述字符节点所包含的block的数量；以及，block所包含的槽位的数量；

[0144] 例如，假设上述区块链数据对应的key为16进制字符串，此时上述区块链数据的key对应的字符串所包含的字符元素的类型数为16；则上述字符节点中所包含的block的数量，以及每一个block所包含的槽位的数量都可以为16。

[0145] 需要说明的是，在实际应用中，上述字符节点所包含的block的数量，和上述block所包含的槽位的数量，可以保持相同；

[0146] 例如，在一个例子中，以上述区块链数据的key对应的字符串为16进制字符串为例，在这种情况下，上述前N层的字符节点可以均包括16个分别代表不同的16进制字符的block；而每一个block可以进一步包括16个分别代表不同的16进制字符的槽位。

[0147] 通过这种方式，则上述FDMT树前N层每一层中的单个字符节点，可以具有16*16＝256个槽位，显然图5示出的FDMT树上的字符节点相对于图1示出的MPT树上用于存储字符前缀的节点而言，将具有更大的存储容量。

[0148] 当前，在实际应用中，上述字符节点所包含的block的数量，和上述block所包含的槽位的数量，具体也可以不相同；

[0149] 例如，在实际应用中，上述区块链数据的key对应的字符串，具体也可以是由两种不同进制字符构成的字符串；比如，上述区块链数据的key对应的字符串具体可以是由16进制的字符和10进制的字符混合构成的字符串，在这种情况下，上述前N层的字符节点可以均包括16个分别代表不同的16进制字符的block；而每一个block可以进一步包括10个分别代表不同的10进制字符的槽位；或者，上述前N层的字符节点可以均包括10个分别代表不同的10进制字符的block；而每一个block可以进一步包括16个分别代表不同的16进制字符的槽

位。

[0150] 在示出的一种实施方式中，上述FDMT树包含的字符节点的层数，具体可以是一个固定值；在实际应用中，上述N的取值具体可以是一个大于或者等于1的整数；也即，上述FDMT树具体可以是一棵包含至少一层字符节点，并且包含的字符节点的层数相对固定的Merkle树。

[0151] 例如，在一个例子中，以上述合约数据的key为合约账户地址为例，假设区块链系统支持的合约账户地址被设计为，前6位地址字符可以相同，那么在这种情况下，可以将区块链账户地址前6位的地址字符，作为区块链账户地址的字符前缀；而且，由于字符节点存储的合约账户地址的字符前缀中的字符的长度为2位字符；因此，上述Merkle树可以被设计成包含三层字符节点的树状结构。

[0152] 基于图5示出的FDMT树的树形结构设计，，一方面，由于上述FDMT树中的每一个字符节点，均包括分别代表不同字符的多个block；并且，各block又进一步包括多个分别代表不同字符的槽位；因此，通过这种设计，使得每一个字符节点将具有更大的数据存储容量和数据承载能力，从而在将上述FDMT树中的字符节点写入数据库进行存储时；或者，在访问数据库中存储的上述Merkle树中的字符节点时，可以使得字符节点的数据存储容量，与承载上述数据库的存储介质的单次IO读写的容量更加适配，从而可以充分利用承载上述数据库的存储介质自身的IO读写能力，提升数据读写效率；

[0153] 例如，以承载上述数据库的存储介质为单个物理扇区为4KB大小的磁盘为例，假设该磁盘单次IO读写的容量为4kb(一个扇区)，对于图1描述的MPT树上的1个Branch node而言，假设Branch node包含的16个字段均填充了32byte的hash值，则该Branch Node的数据存储容量是32字节*16＝512byte左右；显然针对Branch node的一次IO读取的容量最大只能是512byte左右，远小于该磁盘一次IO读取最大可以读取到4Kb的读取能力，这显然无法充分利用磁盘的IO读取能力，存在严重的性能浪费。

[0154] 但是，如果采用如图5所示出的FDMT树的树形结构设计，假设上述前N层的字符节点可以均包括16个分别代表不同的16进制字符的block；每一个block可以进一步包括16个分别代表不同的16进制字符的槽位，则上述FDMT树前N层每一层中的单个字符节点，可以具有16*16＝256个槽位；假设每一个槽位填充的是32byte的hash值，则在所有槽位满负荷的情况下，一个字符节点的最大存储容量则是256*32byte＝8192byte＝8kb，刚好是两个扇区的容量。显而易见的，图5示出的FDMT树的树形结构中的每一个字符节点的数据存储容量，与上述磁盘的单次IO读写的容量更加适配，可以充分利用该磁盘自身的IO读写能力，提升数据读写效率。

[0155] 而且，上述FDMT树上单个字符节点的数据存储容量和数据承载能力的提升，势必也会导致上述FDMT树整体的数据存储容量和数据承载能力的提升，使得上述FDMT树上可以存储更多的区块链数据；

[0156] 例如，假设图5示出的FDMT树的树形结构包含3层字符节点，每一层的字符节点可以均包括16个分别代表不同的16进制字符的block；每一个block可以进一步包括16个分别代表不同的16进制字符的槽位；那么，每一层中的单个字符节点，可以具有16*16＝256个槽位；则三层字符节点共计可以承载256*256*256＝16.77M个字符组合，共计可以链接16.77M个bucket数据块；假设用户定义每一个bucket数据块可以承载16条数据记录，则整棵

Merkle树最多可以承载16.77M*16条区块链数据;显而易见的,相较于图1示出的MPT树,图5示出的上述FDMT树可以存储更多的区块链数据,具有更大的数据承载能力。

[0157] 第二方面,由于FDMT树中的每一个字符节点均采用统一的数据结构;对于上述FDMT树中各层的字符节点来说,其实际存储的区块链数据的key的字符前缀的字符长度也将保持固定;因此,通过这种设计,可以避免由于各层字符节点其实际存储的字符长度不固定,而导致的节点的频繁分裂,从而可以确保FDMT树的树形结构中所包含的字符节点的层数,始终处于一个相对稳定的状态;

[0158] 第三方面,由于通过这种设计,每一个字符节点将具有更大的数据存储容量和数据承载能力;并且,上述FDMT树的树形结构中包含的字符节点的层数,将处于一个相对稳定的状态;因此,字符节点存储容量的提升和字符节点的层数相对稳定,在某种程度上可以确保上述FDMT树的字符节点将具有更少的层数;从而,在字符节点具有更大的数据存储容量和数据承载能力,以及上述FDMT树的字符节点具有更少的层数的基础之上,系统在进行冷启动,需要从承载上述数据库的存储介质中,将上述FDMT树的前N层的字符节点作为需要频繁读取的数据加载到内存中时,则可以显著降低将上述存储介质存储的上述前N层的字符节点读取到内存中时的IO读取次数,以及将FDMT树的前N层的字符节点加载到内存时的整体读取时长,进而从根本上缩短系统冷启动时的启动时延。

[0159] 例如,以图5示出的FDMT树为3层字符节点固定,且每一层的字符节点包括16个分别代表不同的16进制字符的block;每一个block进一步包括16个分别代表不同的16进制字符的槽位为例,在所有槽位满负荷的情况下,一个字符节点的最大存储容量则是256*32byte=8192byte=8kb;对于图1示出的MPT树而言,由于其需要进行频繁的节点分裂,MPT树的层数不固定;而且,由于其单个BranchNode节点的存储容量512byte,远小于图5示出的FDMT树上的字符节点;势必造成MPT树具有更大的层数(比如MPT树最大可以达到64层,远大于3层)。而系统在冷启动,将Merkle树前N层读取到内存时,通常是一层一层读取的;因此,基于图1示出的MPT树,显然需要更多的读取次数。

[0160] 而且,由于MPT树上的单个Branch Node节点的存储容量512byte,仅为承载上述数据库的单个物理扇区为4KB的八分之一,读取效率很低;因此,即便按照图1的MPT树和图5示出的FDMT树存储同样的数据,系统在冷启动针对MPT树的IO读取次数,也可能是针对图5示出的FDMT树的IO读取次数的至少8倍。

[0161] 显而易见的,系统在冷启动时,针对图5示出的上述FDMT树的IO读取次数,将远小于针对图1示出的MPT树的IO读取次数;因此,图5示出的上述FDMT树的树形结构设计对系统冷启动将更加友好。

[0162] 在本说明书中,如图5所示的FDMT树的树形结构中的最后一层的叶子节点(即图5中示出的Leaf node),具体用于存储上述区块链数据的key的字符后缀;以及上述区块链数据的value两部分内容。

[0163] 其中,需要说明的是,由于上述叶子节点实际存储的数据,相对于上述字符节点来说,通常具有更大的数据容量;比如,上述叶子节点实际存储的上述区块链数据的value,通常为上述区块链数据的原始内容,上述区块链数据的原始内容相对于上述区块链数据的字符前缀而言,其所占用的存储空间也更大;因此,在本说明书中,为了确保上述叶子节点能够具有更大的数据容量,上述叶子节点具体可以采用大数据块的形式来存储数据。

[0164] 其中，上述数据块的具体形式以及存储结构，在本说明书中不进行特别限定；

[0165] 在示出的一种实施方式中，上述叶子节点仍然可以采用bucket数据桶的存储形式；其中，上述bucket具体可以是用于存储数据的容器或者存储空间。

[0166] 请参见图7，图7为本说明书示出的一种bucket数据桶的结构图；

[0167] 如图7所示，在上述bucket数据桶(即图7中示出的bucker node)中，可以包括若干条数据记录；其中，需要说明的是，上述bucket数据桶中包含的上述若干条数据记录，在逻辑上不再是一个整体，而是在逻辑上分离的若干条不同的数据记录。每一条数据记录都分别对应一条区块链数据，用于存储上述区块链数据的key中的字符后缀(key-end)和上述区块链数据的value；也即，一条数据记录，是指一条包含上述区块链数据的key中的字符后缀和上述区块链数据的value的存储记录。

[0168] 在示出的一种实施方式中，上述bucket数据桶中包含的上述数据记录，具体可以是用于存储上述区块链数据的key中的字符后缀和上述区块链数据的value的key-value键值对；

[0169] 其中，请参见图7，该key-value键值对的value，可以是上述区块链数据的key中的字符后缀(即图7中示出的key-end)和上述区块链数据的value两部分对应的数据内容；而该key-value键值对的key，可以是上述区块链数据的key中的字符后缀和上述区块链数据的value两部分对应的数据内容的hash值。

[0170] 通过这种方式，由于FDMT树中的叶子节点中所包含的若干条数据记录，在逻辑上不再是一个整体，而是在逻辑上分离的若干条key-value键值对；因此，上述Merkle树中的叶子节点所包含的数据，将不再作为数据库中的一个整体的存取单位进行整体存取，上述叶子节点所包含的每一条key-value键值对将作为数据库中的一个独立的存取单位，从而使得针对上述数据库的数据存取将更加灵活；

[0171] 例如，以上述区块链数据为区块链中的区块链账户对应的最新账户状态数据，上述区块链数据的key为区块链账户地址为例，在这种情况下，上述bucket数据桶中包含的上述数据记录对应的key-value键值对的key，可以是区块链账户地址的字符后缀和对应的账户状态数据两部分数据内容的hash值；该key-value键值对的value，可以是区块链账户地址的字符后缀和对应的账户状态数据两部分的数据内容。

[0172] 假设需要读取上述bucket数据桶中包含的某一个特定的账户地址的字符后缀和对应的账户状态数据，由于上述bucket数据桶中的各条key-value键值对都是一个独立的存取单位；因此，只需要基于该特定的账户地址的字符后缀和对应的账户状态数据这两部分数据内容的hash值，在数据库进行内容寻址即可，而不再需要从数据库中将该叶子节点中所包含的所有数据内容，整体读取到内存中，再在内存中进一步查找需要读取的账户地址的字符后缀和对应的账户状态数据；

[0173] 相应的，如果需要向上述bucket数据桶写入一个新的账户地址的字符后缀和对应的账户状态数据，或者对上述bucket数据桶中包含的一个特定的账户地址对应的账户状态数据进行更新，则可以直接根据新的账户地址的字符后缀和对应的账户状态数据构建key-value键值对，并将该key-value键值对写入上述bucket数据桶即可；或者，基于该特定的账户地址的字符后缀和对应的账户状态数据这两部分数据内容的hash值，在数据库进行内容寻址，查找到对应的key-value键值对，然后写入该特定的账户地址对应的更新后的账户状

态数据，对该key-value键值对原有的value进行更新即可。

[0174] 在示出的一种实施方式中，上述bucket数据桶中包含的若干条key-value键值对，除了可以是在逻辑上分离的数据以外，对于每一条key-value键值对的key和value而言，也可以是在逻辑上分离的数据；

[0175] 也即，在实际应用中，还可以进一步对上述bucket数据桶所包含的各条key-value键值对中的key和value，进行进一步的逻辑分离。

[0176] 其中，对上述bucket数据桶所包含的各条key-value键值对中的key和value，进行进一步的逻辑分离的具体方式，在本说明书中不仅特别限定；

[0177] 例如，在示出的一种实施方式中，可以将上述bucket数据桶的存储结构划分为在逻辑上隔离的数据头部分和数据体部分，在上述数据头部分集中存储上述bucket数据桶包含的各条key-value键值对中的key，在上述数据体部分集中存储上述bucket数据桶包含的各条key-value键值对中的value。

[0178] 请参见图8，图8为本说明书示出的另一种bucket数据桶的结构图；

[0179] 在图8示出的bucket数据桶的结构中，具体可以包括数据头部分(即图8示出的Header部分)和数据体部分(即图8示出的Body部分)；上述数据头部分和上述数据体部分为在逻辑上完全分离的两部分。

[0180] 其中，上述数据头部分，用于集中存储该bucket数据桶所包含的若干条key-value键值对中的key；也即，将该bucket数据桶所包含的若干条key-value键值对中的key统一存储在该bucket数据桶中的数据头部分。

[0181] 相应的，上述数据体部分，用于集中存储该bucket数据桶所包含的若干条key-value键值对中的value；也即，将该bucket数据桶所包含的若干条key-value键值对中的value统一存储在该bucket数据桶中的数据体部分。

[0182] 在实际应用中，上述数据头部分和上述数据体部分，可以分别包含若干条在逻辑上分离的数据记录；上述数据头部分包含的数据记录，用于存储该bucket数据桶所包含的若干条key-value键值对中的key；相应的，上述数据体部分包含的数据记录，用于存储该bucket数据桶所包含的若干条key-value键值对中的value。其中，所述数据头部分包含的若干数据记录与所述数据体部分包含的若干条数据记录可以是一一对应的关系；

[0183] 例如，如图8所示，数据头部分包含的各条数据记录，可以按照顺序与数据体部分包含的各条数据记录一一对应；比如，数据头部分包含的第N条数据记录，与数据体部分包含的第N条数据记录对应。

[0184] 在示出的一种实施方式中，当上述FDMT树上的叶子节点采用了如图8示出的bucket数据桶的结构时，则可以将上述FDMT树中的叶子节点中的数据头部分所包含的数据内容的hash值，作为该叶子节点的hash值；此时，上述FDMT树上的叶子节点，可以通过自身的数据头部分所包含的数据内容的hash值，链接至其上层的字符节点。

[0185] 在这种情况下，在计算上述bucket数据桶的hash值时，可以不再需要针对该bucket数据桶包含的所有数据内容进行hash计算，而是只针对该bucket数据桶的数据头部分包含的数据内容进行hash值计算；由于上述bucket数据桶的数据头部分包含的数据内容，通常要远小于上述bucket数据桶的数据体部分包含的数据内容；因此，通过这种方式，可以显著的降低hash计算的计算量，进而减少hash计算所需要的时长，提升hash计算的计

算效率;

[0186] 例如,请继续参见图8,假设上述bucket数据桶的数据头部分和数据体部分,分别包含16条数据记录;对于数据头部分的一条数据记录,其本身实际上是一个hash值,通常大小为32bytes;而对于数据体部分的一条数据记录,其对应的是区块链数据的value,通常大小为1KB左右;因此,上述数据头部分包含的数据内容的总大小为32byte*16=512bytes=0.5KB;而上述数据体部分包含的数据内容的总大小为16*1KB=16KB;显而易见的,上述数据头部分包含的数据内容的总大小,远小于上述数据体部分包含的数据内容的总大小。

[0187] 请继续参见图8,假设图8示出的bucket node1中的value2更新为value2′;则只需要将该bucket node1的Header部分的第二条数据记录所包含的“key-end2+value2”更新为“key-end2+value2′”,然后重新计算“key-end2+value2′”的hash值;其中,该hash值可以表示成hash(key-end2+value2′);再将该bucket node1的Body部分的第二条数据记录所包含的key2更新为hash(key-end2+value2′);而bucket node1中除了Header部分和Header部分的第二条数据记录以外,其它条的数据记录则完全不需要进行更新。

[0188] 当针对bucket node1的以上更新完成后,此时可以进一步件将bucket node1的Header部分包含的各个key1-keyN拼接起来,再重新计算一个hash值,然后将计算得到的hash值,填充到该bucket node1的上一层字符节点相应的槽位中(比如图8示出的block1的第2个槽位)。由于上述bucket node1的Header部分包含的数据内容,要远小于上述bucket node1的Header部分包含的数据内容;因此,通过这种方式,可以显著的降低hash计算的计算量,进而减少hash计算所需要的时长,提升hash计算的计算效率。

[0189] 其中,需要说明的是,上述bucket数据桶中所包含的数据记录的条数,在本说明书中不进行特别限定;在示出的一种实现方式中,上述bucket数据桶中包含的数据记录的条数,具体可以由用户进行自定义配置。

[0190] 例如,以上述区块链数据为区块链中的区块链账户对应的最新账户状态数据,上述区块链数据的key为区块链账户地址为例,在这种情况下,上述bucket数据桶中的每一条数据记录,分别与一个区块链账户的账户状态相对应;上述bucket数据桶中的数据记录的条数,实际上代表着该bucket数据桶可以容纳区块链账户的账户承载能力;因此,用户通过对上述bucket数据桶能够容纳的数据记录的条数进行自定义,可以实现对上述bucket数据桶的账户承载能力进行灵活的定制;比如,在一个例子中,上述bucket数据桶中包含的数据记录的条数可以由用户配置为16条或者64条,从而使得单个bucket数据桶可以承载16或者64个区块链账户的状态数据。

[0191] 在本说明书中,上述节点设备用于存储上述Merkle树的数据库的具体类型,在本说明书中不进行特别限定,本领域技术人员可以基于实际的需求进行灵活的选择;

[0192] 在一种实现方式中,上述数据库,具体可以是Key-Value型数据库;例如,在一个例子中,上述数据库可以为采用多层存储结构的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。

[0193] 在这种情况下,在将上述Merkle树在数据库中进行存储时,具体可以将上述Merkle树上的节点,以Key-Value键值对的形式进一步存储至上述数据库中。

[0194] 例如,上述数据库通常是存储在上述节点设备上搭载的持久化存储介质(比如存储磁盘)中的;上述存储介质为与上述数据库对应的物理存储;在将上述Merkle树在数据库

中进行存储时，具体可以通过执行commit命令，将上述Merkle树上的节点，以Key-Value键值对的形式，从节点设备的内存中，进一步写入承载上述数据库的存储介质。

[0195] 需要说明的是，上述Merkle树上的节点，在以Key-Value键值对的形式在上述数据库中进行存储时，上述Key-Value键值对的key，具体可以是述Merkle状态树上的节点的节点ID；

[0196] 其中，上述节点ID具体可以包括能够唯一标识上述Merkle树上的节点的标识信息；

[0197] 例如，在一种实现方式中，上述节点ID具体可以是上述Merkle树上的节点包含的数据内容的hash值；在这种情况下，在需要查询上述Merkle树上的节点时，可以基于节点所包含的数据内容的hash值作为key来进行内容寻址。

[0198] 在另一种实现方式中，上述节点ID具体也可以包括上述Merkle树上的节点在上述Merkle树中的路径信息；也即，将上述Merkle树上的节点在上述Merkle树中的路径信息作为该节点的节点ID；其中，上述路径信息具体可以包括，能够描述节点与其它节点之间的链接关系，以及节点在上述Merkle树上的位置的任意形式的信息；

[0199] 例如，请参见图9，图9为本说明书示出的一种为Merkle树上的节点设置节点ID的示意图；对于如图7所示包含三层字符节点的Merkle树，假设作为根节点的tree node的节点ID用0x00表示，那么图7示出的bucket node的节点ID，可以表示成0x00123456。

[0200] 其中，0x00为根节点的节点ID；123456是指上述bucket node在Merkle树上从根节点到该bucket node的路径信息；12表示第一层tree node的第一个block的第2个槽位；34表示第二层tree node的第3个block的第4个槽位；56表示第三层tree node的第5个block的第6个槽位。

[0201] 基于该节点ID，可以明确该bucket node与其它node之间的链接关系，以及该bucket node在上述Merkle树上的具体位置；比如，基于该节点ID，可以明确该bucket node链接在第三层tree node的第5个block的第6个槽位；而第三层的tree node又链接在第二层tree node的第3个block的第4个槽位；第二层的tree node又进一步链接在第一层作为根节点的tree node的第1个block的第2个槽位。

[0202] 通过这种方式，可以在使用节点ID来检索Merkle状态树上存储的节点时，可以精确定位到该节点在Merkle树上所在的具体槽位。

[0203] 在另一种实现方式中，上述节点ID具体也可以包括上述Merkle树上的节点在上述Merkle树中的路径信息，和该节点所包含的数据内容的hash值；也即，将上述Merkle树上的节点在上述Merkle树中的路径信息，和该节点所包含的数据内容的hash值作为该节点的节点ID。

[0204] 例如，在实现时，可以将节点在上述Merkle树中的路径信息，和该节点所包含的数据内容的hash值拼接生成的字符串，作为该节点的节点ID；当然，在实际应用中，在进行拼接的过程中，也可以进一步引入上述路径信息和上述hash值以外的其它类型的信息来生成节点ID；在本说明书中不再一一列举。

[0205] 通过这种方式，除了可以基于节点所包含的数据内容的hash值作为key来进行内容寻址以外，还可以在使用节点ID来检索Merkle状态树上存储的节点时，可以精确定位到该节点在Merkle树上所在的具体槽位。

[0206] 在本说明书中，数据查询方在需要查询智能合约对应的存储空间中存储的合约数据时，可以生成一个包含被查询的目标合约数据的key的查询请求，其中，如前所述，该目标合约数据的key可由包括字符前缀部分和字符后缀部分；然后，可以将该查询请求发送给节点设备。而节点设备在接收到该查询请求，可以响应该查询请求，从与该智能合约对应的上述storage树的根节点开始，遍历该storage树，在该storage树中查找，与上述目标合约数据的key的字符前缀中的字符匹配的字符节点，以及与查找到的该字符节点对应的第一叶子节点；

[0207] 进一步的，在查找到上述第一叶子节点后，由于该第一叶子节点存储的数据内容已经被组织成了与上述storage树对应的一颗storage子树；此时，可以进一步遍历与该第一叶子节点存储的数据内容对应的storage子树，从该storage子树的根节点开始，继续在该storage子树中查找，与上述目标合约数据的key的字符后缀中的字符匹配的字符节点，以及与查找到的该字符节点对应的第二叶子节点。

[0208] 当查找到上述第二叶子节点之后，此时节点设备可以进一步读取该第二叶子节点中存储的上述目标合约数据的Value，并将查找到的上述目标合约数据的Value返回给上述数据查询方即可。

[0209] 在本说明书中，出于安全性的考量，仍然可以采用该合约数据的原始key的hash值，作为上述合约数据对应的key-value键值对中的key；但在实际应用中，作为数据查询方而言，其具体的业务流程中，可能会需要了解上述合约数据的原始key。

[0210] 在这种情况下，在上述storage子树的叶子节点中，除了可以存储上述合约数据的value以外，还可以存储上述合约数据的原始key；也即，在上述storage子树的叶子节点中，存储上述合约数据的value和上述合约数据的原始key的对应关系。当节点设备通过遍历上述storage树和上述storage子树，查找到上述第二叶子节点时，读取该第二叶子节点中存储的上述目标合约数据的原始key和上述合约数据的Value，并将读取到的上述目标合约数据的原始key和上述目标合约数据的Value一并返回给上述数据查询方。

[0211] 在以上技术方案中，由于将存储合约数据的storage树的叶子节点上存储的数据内容，也组织成了与该storage树对应的storage树子树；因此，在遍历叶子节点中存储的数据内容进行合约数据的查询时，对叶子节点中存储的数据内容的遍历，将转换成对一颗storage树子树的遍历，因此可以提升针对该叶子节点中存储的数据内容的遍历效率。

[0212] 与上述方法实施例相对应，本说明书还提供了一种智能合约存储内容的遍历装置的实施例。

[0213] 本说明书的智能合约存储内容的遍历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。

[0214] 从硬件层面而言，如图10所示，为本说明书的智能合约存储内容的遍历智能合约存储内容的遍历装置所在电子设备的一种硬件结构图，除了图10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。

[0215] 图11是本说明书一示例性实施例示出的一种智能合约存储内容的遍历装置的框

图。

[0216] 请参考图11,所述智能合约存储内容的遍历装置110可以应用在前述图10所示的电子设备中,其中与所述智能合约对应的存储空间中存储的合约数据,以key-value键值对的形式被组织成Merkle树在数据库中存储;所述Merkle树的各叶子节点存储的数据内容,被进一步组织成与所述Merkle树对应的Merkle子树;其中,所述Merkle树以及所述Merkle子树均为融合了Trie字典树的树形结构的多层树状存储结构,包含至少一层字符节点和一层叶子节点;所述字符节点用于存储所述合约数据的key对应的字符串中的字符;所述叶子节点用于存储所述合约数据的Value;所述Merkle树和所述Merkle子树上的节点,通过在其上一层的节点中填充该节点的hash值,与其上一层的节点进行链接;所述装置110包括:

[0217] 接收模块1101,接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求;其中,所述查询请求包括被查询的目标合约数据的key;所述目标合约数据的key对应的字符串包括字符前缀和字符后缀;

[0218] 查找模块1102,响应于所述查询请求,从所述Merkle树的根节点开始,遍历所述Merkle树,在所述Merkle树中查找,与所述字符前缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第一叶子节点;在查找到所述第一叶子节点后,进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树,从所述Merkle子树的根节点开始,在所述Merkle子树中查找,与所述字符后缀中的字符匹配的字符节点,以及与查找到的所述字符节点对应的第二叶子节点,并在查找到所述第二叶子节点时,进一步读取所述第二叶子节点中存储的所述目标合约数据的Value。

[0219] 在本实施例中,所述Merkle树中的字符节点,用于存储所述字符前缀中的字符;所述Merkle树的叶子节点,用于存储由所述字符后缀和所述合约数据的value构成的key-value键值对;

[0220] 所述Merkle子树中的字符节点,用于存储所述字符后缀中的字符;所述Merkle子树的叶子节点,用于存储所述合约数据的value。

[0221] 在本实施例中,当所述Merkle子树中的字符节点,存储了所述字符后缀中的部分字符时,所述Merkle子树的叶子节点,用于存储由所述字符后缀中的所述部分字符以外的剩余字符和所述合约数据的value构成的key-value键值对。

[0222] 在本实施例中,所述Merkle树上的叶子节点的上一层字符节点中填充的该叶子节点的hash值,为与所述叶子节点存储的数据内容对应的Merkle子树的根节点的hash值。

[0223] 在本实施例中,所述Merkle树;和/或,所述Merkle子树为MPT树。

[0224] 在本实施例中,所述Merkle树;和/或,所述Merkle子树为FDMT树;

[0225] 其中,所述FDMT树中的各字符节点,均包括分别代表不同字符的多个block;所述block进一步包括多个分别代表不同字符的槽位;所述槽位用于填充该字符节点链接的下一层节点的hash值。

[0226] 在本实施例中,所述FDMT树中包含的字符节点的层数为固定值。

[0227] 在本实施例中,所述FDMT树中的任一字符节点存储的字符,为所述字符节点中的block所代表的字符,与所述block中填充了hash值的槽位所代表的字符,进行拼接生成的字符串。

[0228] 在本实施例中,所述字符节点包括的block的数量,与所述block包括的槽位的数

量相同。

[0229] 在本实施例中，所述字符节点包括分别代表不同的16进制字符的16个block；所述block包括16个分别代表不同的16进制字符的槽位。

[0230] 在本实施例中，所述Merkle树存储的合约数据的key，为所述合约数据的原始key进行hash计算得到的hash值；所述Merkle子树的叶子节点，还存储了所述合约数据的原始key和所述合约数据的value的对应关系；

[0231] 所述查找模块1102：

[0232] 读取所述第二叶子节点中存储的所述目标合约数据的原始key和所述合约数据的Value，并将读取到的所述目标合约数据的原始key和所述目标合约数据的Value返回给与所述查询请求对应的查询方。

[0233] 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。

[0234] 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。

[0235] 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。

[0236] 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。

[0237] 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。

[0238] 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。

[0239] 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。

[0240] 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”。

[0241] 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。

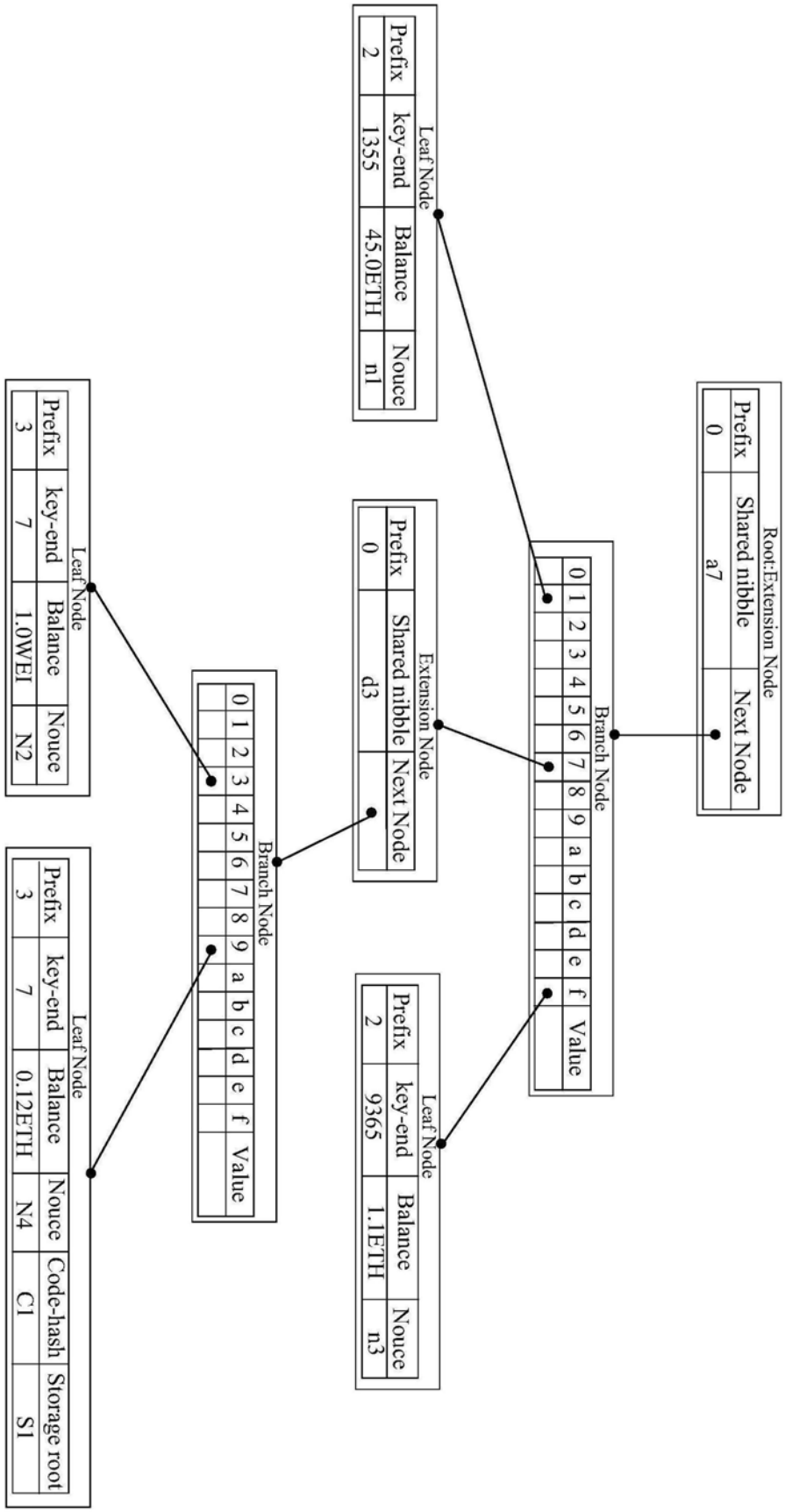

Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
Leaf Node
Prefix
key-end
Balance
Nouce
2
1355
45.0ETH
n1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Balance
Nouce
2
9365
1.1ETH
n3
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
Leaf Node
Prefix
key-end
Balance
Nouce
3
7
1.0WEI
N2
Leaf Node
Prefix
key-end
Balance
Nouce
Code-hash
Storage root
3
7
0.12ETH
N4
C1
S1

图1

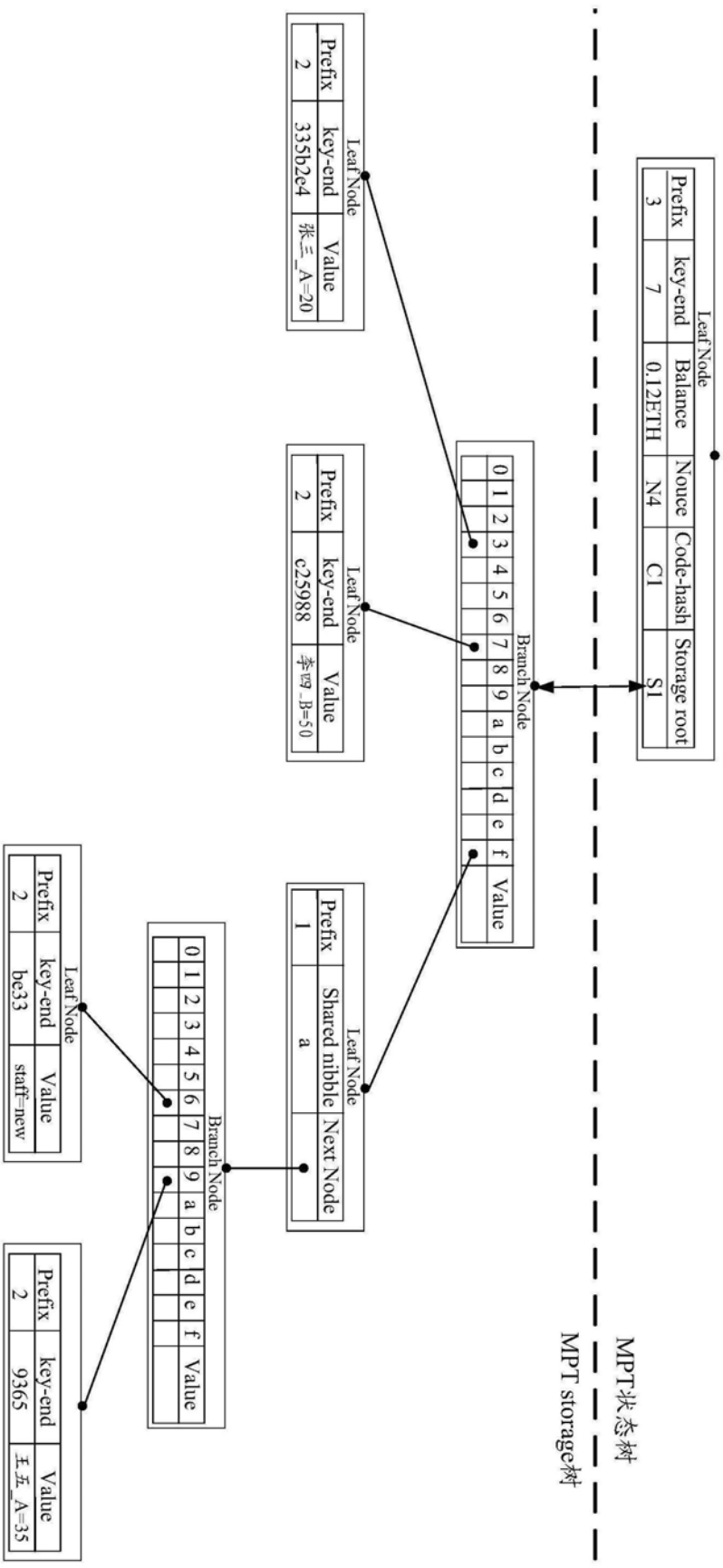

MPT状态树
MPT storage树
Leaf Node
Prefix
key-end
Balance
Nouce
Code-hash
Storage root
3
7
0.12ETH
N4
C1
S1
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f
Value
Leaf Node
Prefix
key-end
Value
2
335b2e4
张三_A=20
Leaf Node
Prefix
key-end
Value
2
c25988
李四_B=50
Leaf Node
Prefix
Shared nibble
Next Node
1
a
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f
Value
Leaf Node
Prefix
key-end
Value
2
be33
staff-new
Leaf Node
Prefix
key-end
Value
2
9365
王五_A=35

图2

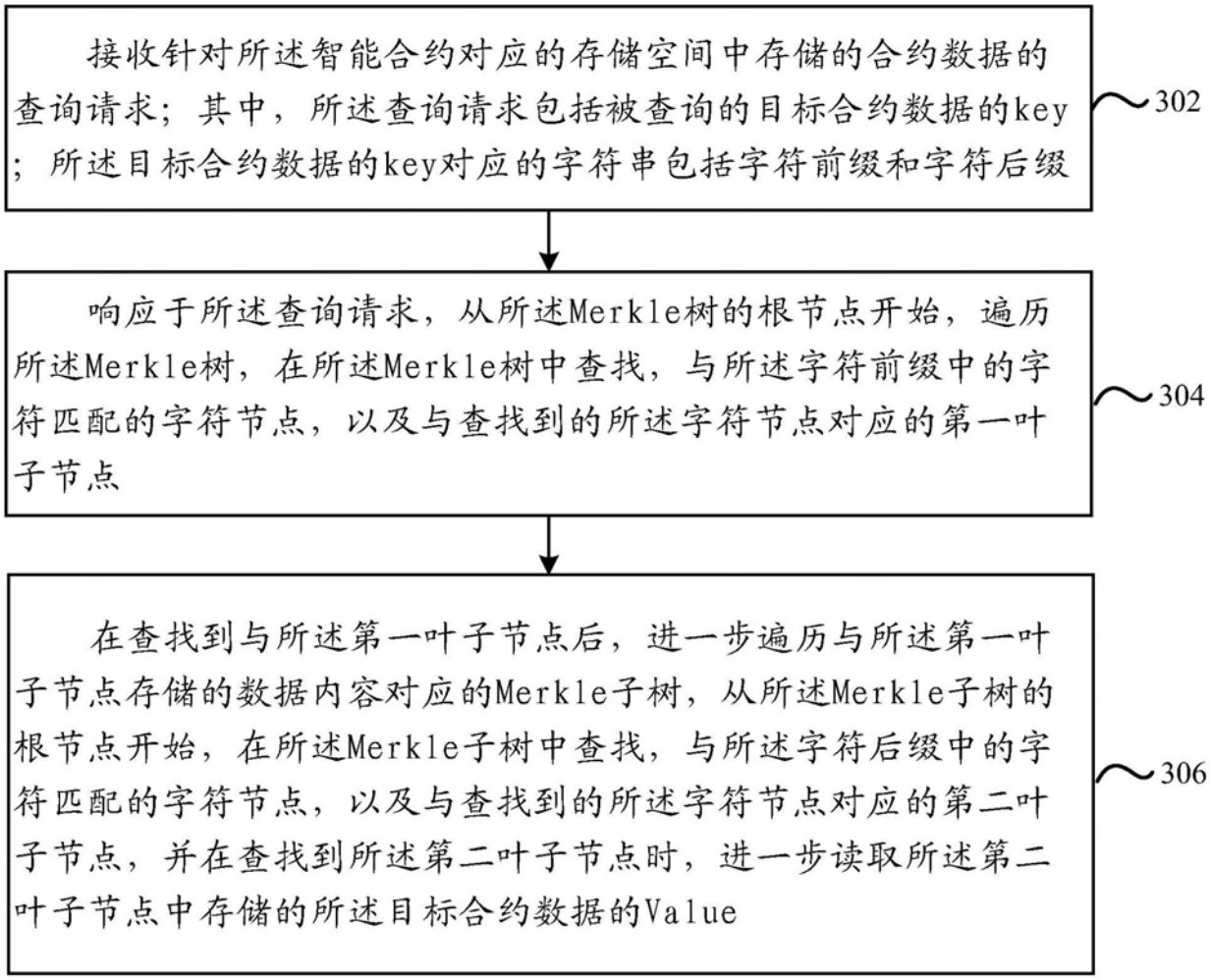

接收针对所述智能合约对应的存储空间中存储的合约数据的查询请求；其中，所述查询请求包括被查询的目标合约数据的key；所述目标合约数据的key对应的字符串包括字符前缀和字符后缀
302
响应于所述查询请求，从所述Merkle树的根节点开始，遍历所述Merkle树，在所述Merkle树中查找，与所述字符前缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第一叶子节点
304
在查找到与所述第一叶子节点后，进一步遍历与所述第一叶子节点存储的数据内容对应的Merkle子树，从所述Merkle子树的根节点开始，在所述Merkle子树中查找，与所述字符后缀中的字符匹配的字符节点，以及与查找到的所述字符节点对应的第二叶子节点，并在查找到所述第二叶子节点时，进一步读取所述第二叶子节点中存储的所述目标合约数据的Value
306

图3

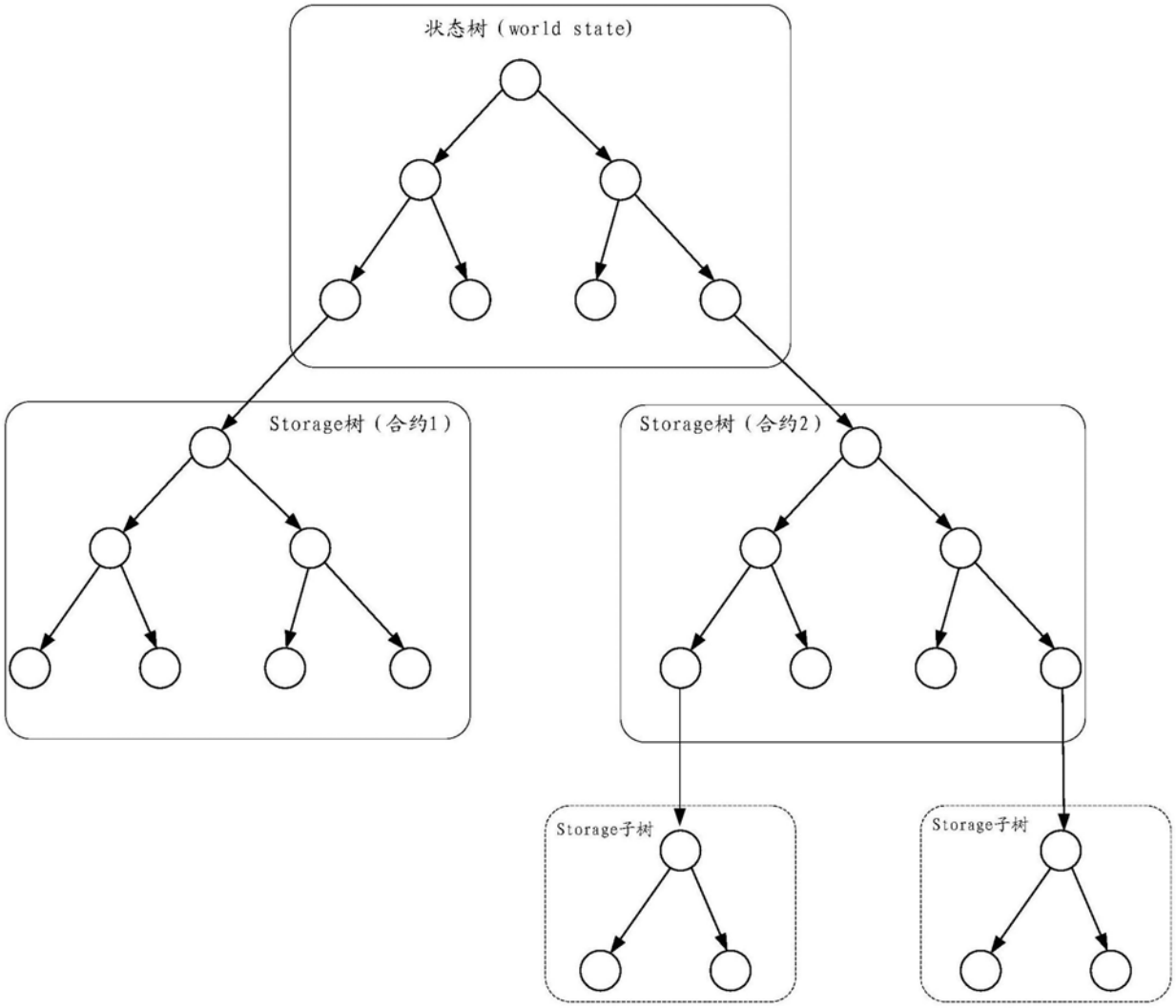

状态树（world state）
Storage树（合约1）
Storage树（合约2）
Storage子树
Storage子树

图4

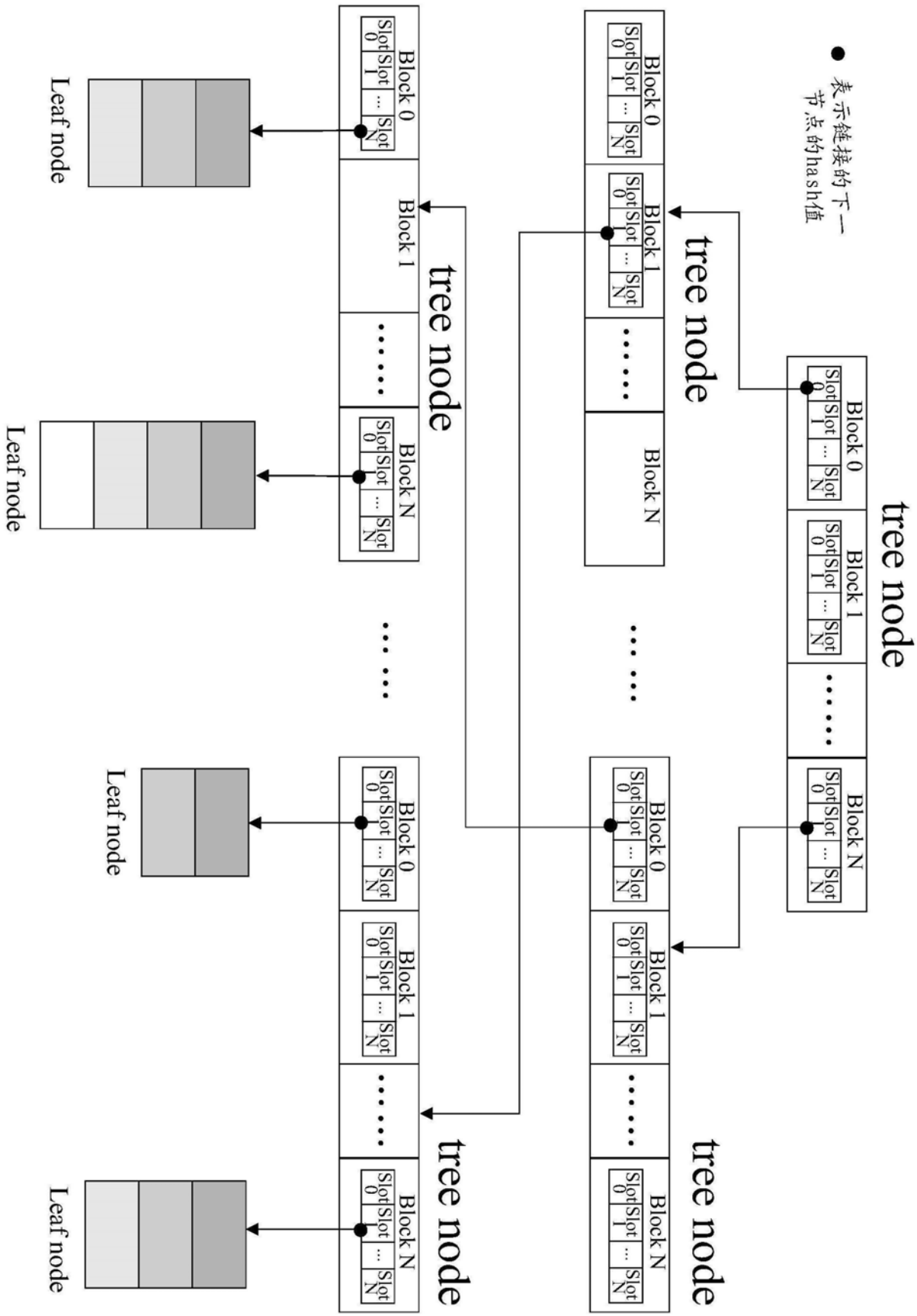

表示链接的下一节点的hash值
tree node
Block 0
Block 1
Block N
Slot 0
Slot 1
Slot N
Leaf node

图5

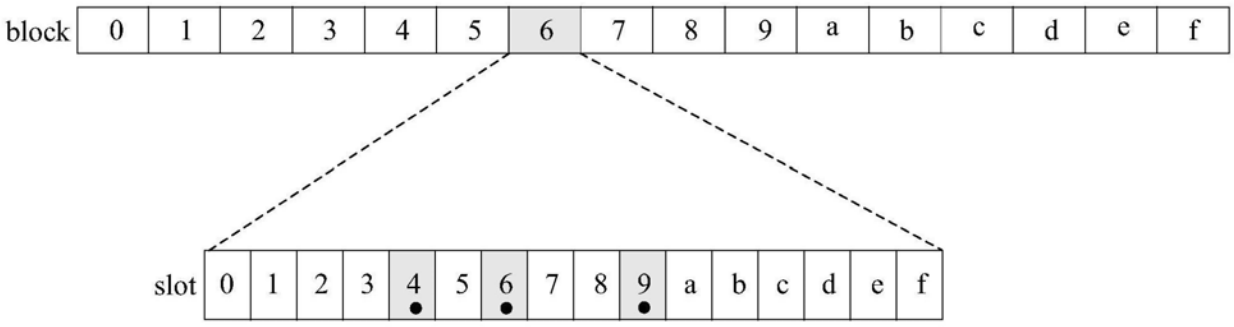


图6

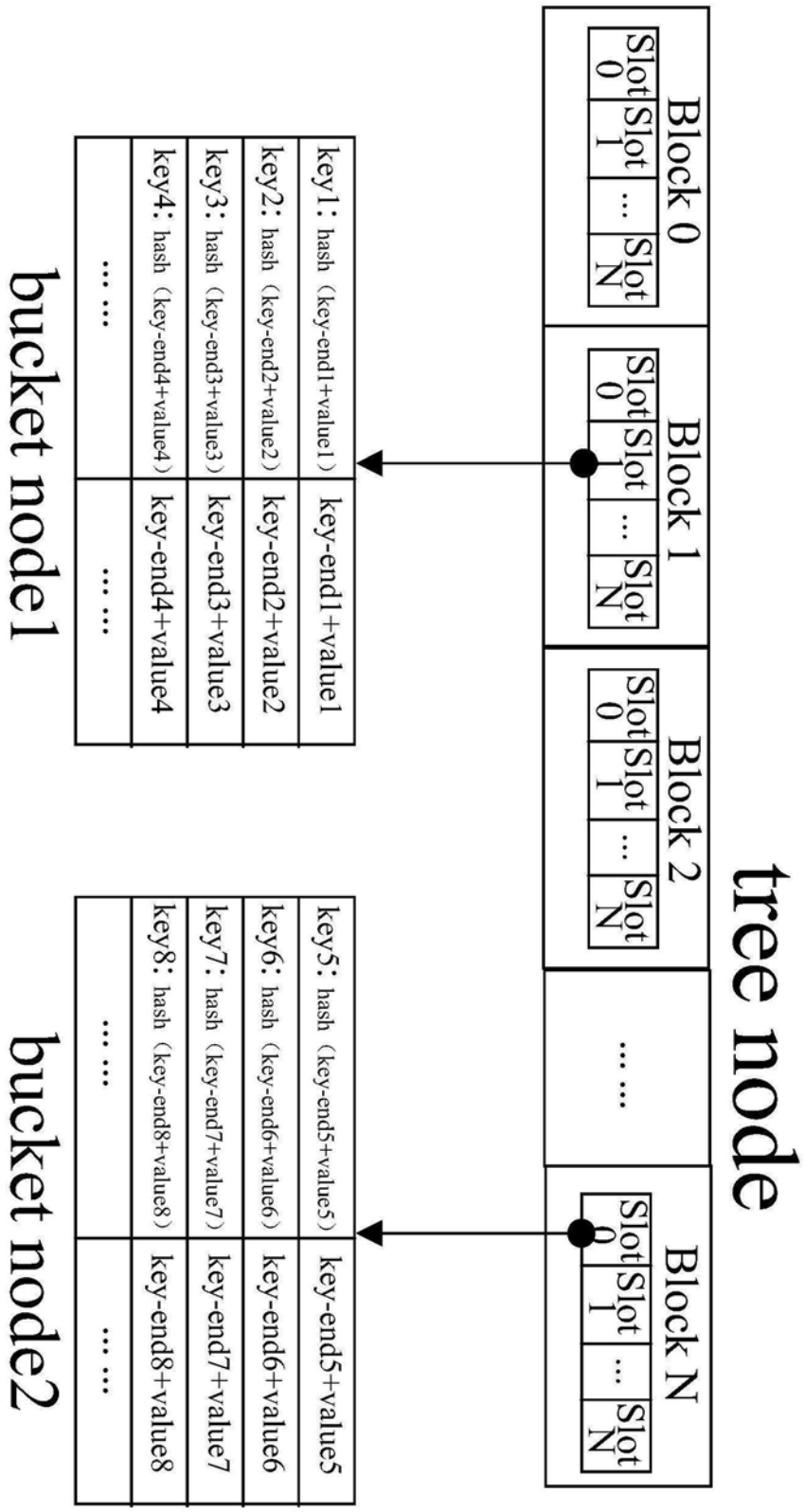

tree node
Block 0
Slot 0
Slot 1
...
Slot N
Block 1
Slot 0
Slot 1
...
Slot N
Block 2
Slot 0
Slot 1
...
Slot N
... ...
Block N
Slot 0
Slot 1
...
Slot N
bucket node1
key1: hash（key-end1+value1）
key-end1+value1
key2: hash（key-end2+value2）
key-end2+value2
key3: hash（key-end3+value3）
key-end3+value3
key4: hash（key-end4+value4）
key-end4+value4
... ...
... ...
bucket node2
key5: hash（key-end5+value5）
key-end5+value5
key6: hash（key-end6+value6）
key-end6+value6
key7: hash（key-end7+value7）
key-end7+value7
key8: hash（key-end8+value8）
key-end8+value8
... ...
... ...

图7

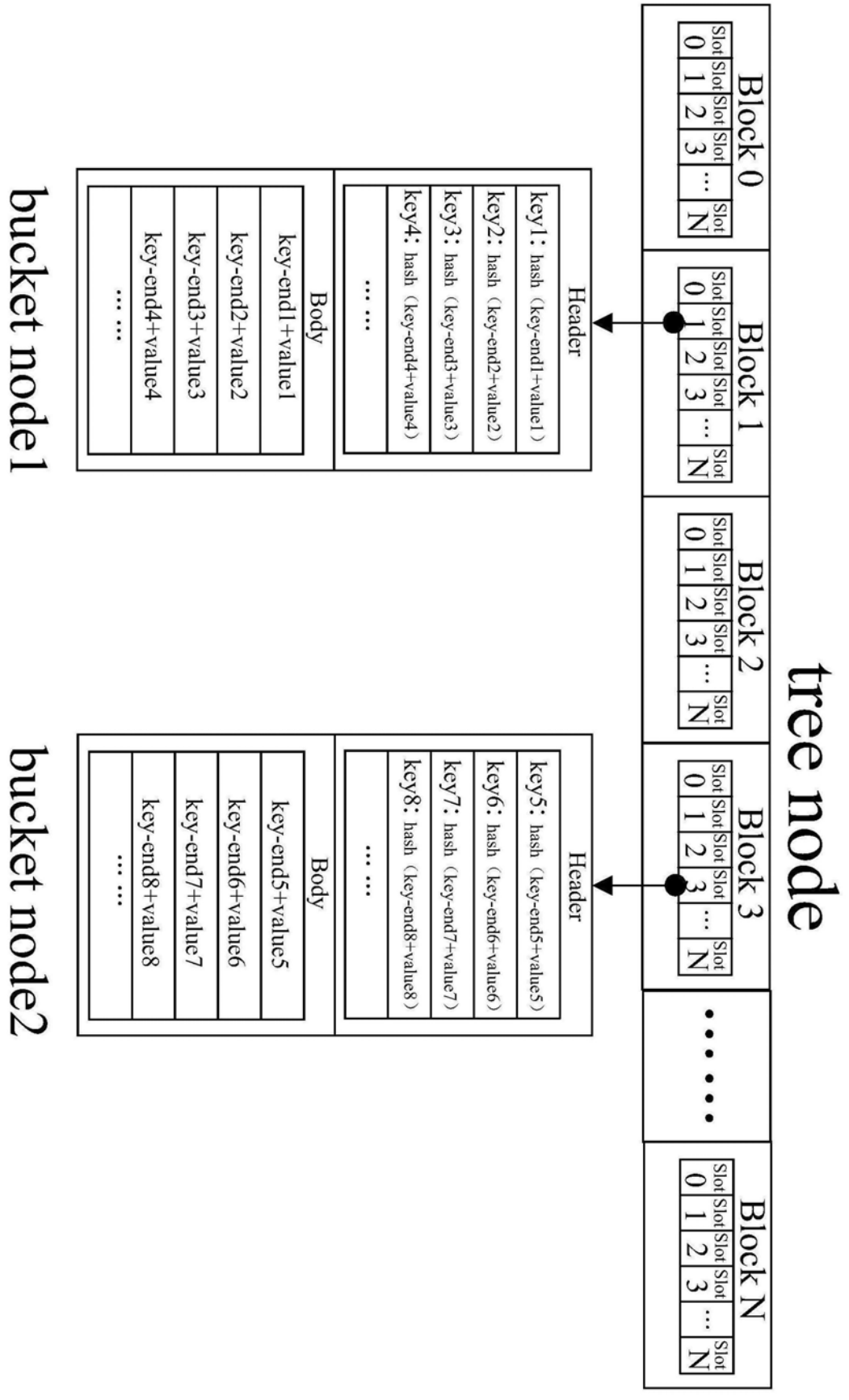

tree node
Block 0
Block 1
Block 2
Block 3
Block N
Slot 0
Slot 1
Slot 2
Slot 3
Slot N
Header
key1: hash（key-end1+value1）
key2: hash（key-end2+value2）
key3: hash（key-end3+value3）
key4: hash（key-end4+value4）
Body
key-end1+value1
key-end2+value2
key-end3+value3
key-end4+value4
bucket node1
Header
key5: hash（key-end5+value5）
key6: hash（key-end6+value6）
key7: hash（key-end7+value7）
key8: hash（key-end8+value8）
Body
key-end5+value5
key-end6+value6
key-end7+value7
key-end8+value8
bucket node2

图8

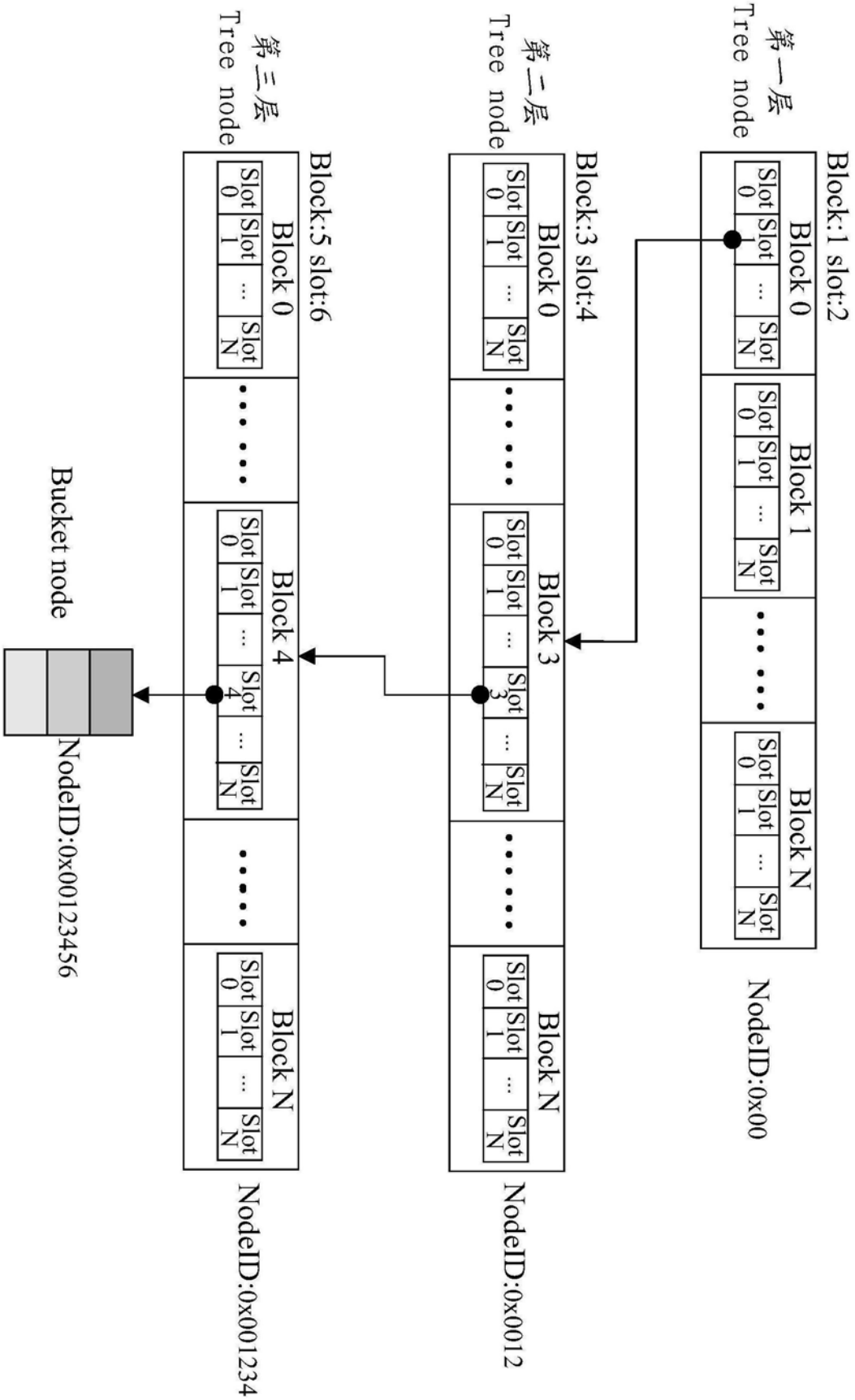

第一层
Tree node
Block:1 slot:2
Block 0
Block 1
Block N
Slot 0
Slot 1
Slot N
NodeID:0x00
第二层
Tree node
Block:3 slot:4
Block 0
Block 3
Block N
Slot 0
Slot 1
Slot 3
Slot N
NodeID:0x0012
第三层
Tree node
Block:5 slot:6
Block 0
Block 4
Block N
Slot 0
Slot 1
Slot 4
Slot N
NodeID:0x001234
Bucket node
NodeID:0x00123456

图9

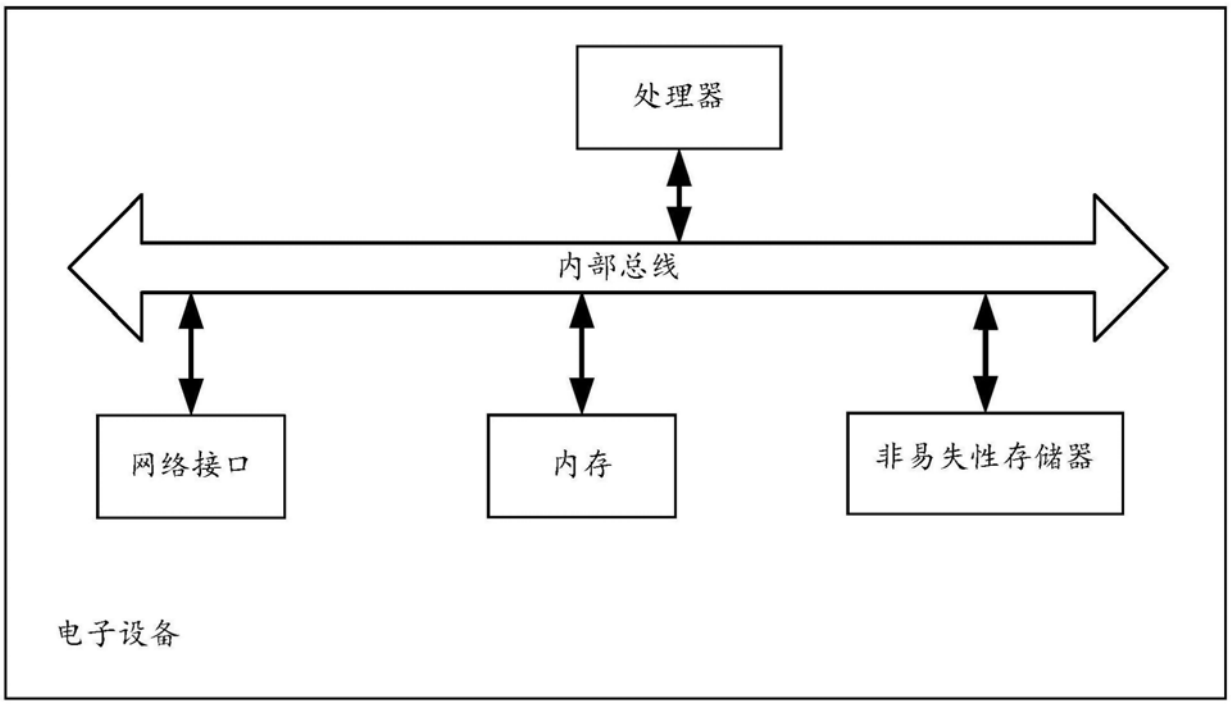

处理器
内部总线
网络接口
内存
非易失性存储器
电子设备

图10

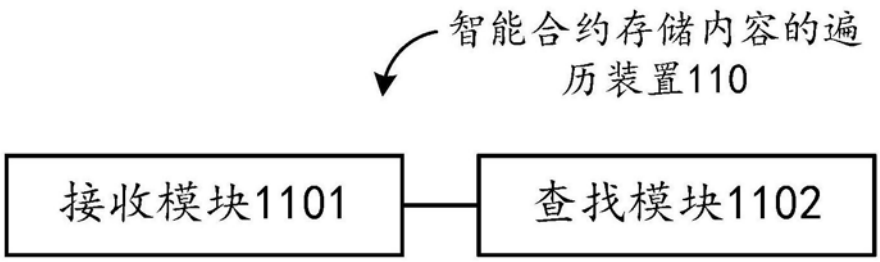

智能合约存储内容的遍历装置110
接收模块1101
查找模块1102

图11