

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2022 年 11 月 10 日 (10.11.2022)



(10) 国际公布号
WO 2022/233274 A1

- (51) 国际专利分类号:
G06F 16/27 (2019.01) *G06F 16/22* (2019.01)
G06Q 40/04 (2012.01) *G06F 16/23* (2019.01)
- (21) 国际申请号: PCT/CN2022/090294
- (22) 国际申请日: 2022 年 4 月 29 日 (29.04.2022)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
202110495070.9 2021年5月7日 (07.05.2021) CN
- (71) 申请人: 支付宝(杭州)信息技术有限公司 (ALIPAY (HANGZHOU) INFORMATION TECHNOLOGY CO., LTD.) [CN/CN]; 中国浙江省杭州市西湖区西溪路 556 号 8 层 B 段 801-11, Zhejiang 310000 (CN)。蚂蚁区块链

科技(上海)有限公司 (ANT BLOCKCHAIN TECHNOLOGY (SHANGHAI) CO., LTD.) [CN/CN]; 中国上海市黄浦区外马路 618 号 8 层 803 室, Shanghai 200010 (CN)。

(72) 发明人: 陆钟豪 (LU, Zhonghao); 中国浙江省杭州市西湖区西溪路 556 号 8 层 B 段 801-11, Zhejiang 310000 (CN)。卓海振 (ZHUO, Haizhen); 中国浙江省杭州市西湖区西溪路 556 号 8 层 B 段 801-11, Zhejiang 310000 (CN)。俞本权 (YU, Benquan); 中国浙江省杭州市西湖区西溪路 556 号 8 层 B 段 801-11, Zhejiang 310000 (CN)。

(74) 代理人: 北京博思佳知识产权代理有限公司 (BEIJING BESTIPR INTELLECTUAL PROPERTY LAW CORPORATION); 中国北京市

(54) Title: BLOCK CHAIN DATA STORAGE METHOD AND APPARATUS, AND ELECTRONIC DEVICE

(54) 发明名称: 区块链数据存储方法及装置、电子设备

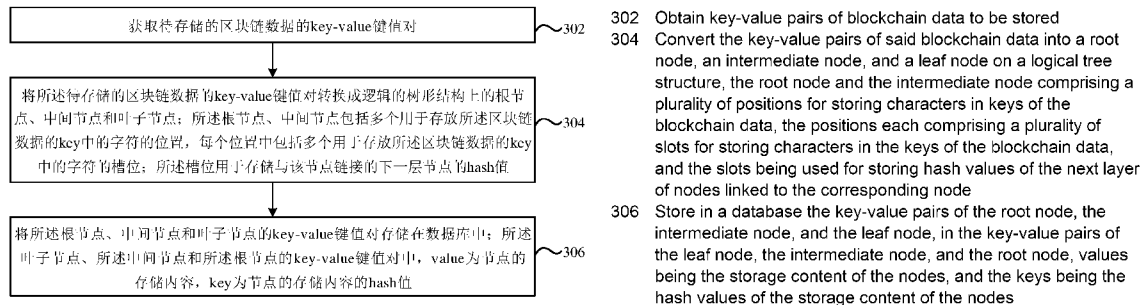


图 3

(57) Abstract: A block chain data storage method and apparatus, and an electronic device. The method comprises: obtaining key-value pairs of blockchain data to be stored; converting the key-value pairs of said blockchain data into a root node, an intermediate node, and a leaf node on a logical tree structure, the root node and the intermediate node comprising a plurality of positions for storing characters in keys of the blockchain data, the positions each comprising a plurality of slots for storing characters in the keys of the blockchain data, and the slots being used for storing hash values of the next layer of nodes linked to the corresponding node; and storing in a database the key-value pairs of the root node, the intermediate node, and the leaf node, in the key-value pairs of the leaf node, the intermediate node, and the root node, values being the storage content of the nodes, and the keys being the hash values of the storage content of the nodes.

(57) 摘要: 一种区块链数据存储方法及装置、电子设备, 方法包括: 获取待存储的区块链数据的key-value键值对; 将所述待存储的区块链数据的key-value键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点; 所述根节点、中间节点包括多个用于存放所述区块链数据的key中的字符的位置, 每个位置中包括多个用于存放所述区块链数据的key中的字符的槽位; 所述槽位用于存储与该节点链接的下一层节点的hash值; 将所述根节点、中间节点和叶子节点的key-value键值对存储在数据库中; 所述叶子节点、所述中间节点和所述根节点的key-value键值对中, value为节点的存储内容, key为节点的存储内容的hash值。

海淀区上地三街9号嘉华大厦B座409,
Beijing 100085 (CN)。

(81) 指定国(除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) 指定国(除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第21条(3))。

区块链数据存储方法及装置、电子设备

技术领域

[01] 本说明书一个或多个实施例涉及区块链技术领域，尤其涉及一种区块链数据存储方法及装置、电子设备。

背景技术

[02] 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。

[03] 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。

[04] 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

[05] 本说明书提出一种区块链中存储数据的方法，包括：获取待存储的区块链数据的 key-value 键值对；将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点；所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置，每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位；所述槽位用于存储与该节点链接的下一层节点的 hash 值；将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中；所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中，value 为节点的存储内容，key 为节点的存储内容的 hash 值。

[06] 本说明书还提出一种区块链中存储数据的装置，包括：获取模块，获取待存储的区块链数据的 key-value 键值对；转换模块，将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点；所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置，每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位；所述槽位用于存储与该节点链接的下一层节点的 hash 值；存储模块，将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中；所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中，value 为节点的存储内容，key 为节点的存储内容的 hash 值。

[07] 以上技术方案中，一方面，由于逻辑的树形结构中的每一个根节点和中间节点，均包括分别代表不同字符的多个位置；并且，各位置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，使得每一个根节点和中间节点将具有更大的数据存储容量和数据承载能力，从而在将上述逻辑的树形结构中的根节点和中间节点写入数据库进行存储时；或者，在访问数据库中存储的逻辑的树形结构中的根节点和中间节点时，可以使得根节点和中间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；而且，上述逻辑的树形结构上节点的数据存储容量和数据承载能力的提升，势必也会导致上述逻辑的树形结构整体的数据存储容量和数据承载能力的提升，使得上述逻辑的树形结构上可以存储更多的区块链数据；第二方面，由于逻辑的树形结构中的每一个根节点和中间节点均采用统一的数据结构；对于逻辑的树形结构中的根节点和中间节点来说，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，可以避免由于根节点和中间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，从而可以确保逻辑的树形结构中所包含的根节点和中间节点的层数，始终处于一个相对稳定的状态；第三方面，由于通过这种设计，每一个根节点和中间节点将具有更大的数据存储容量和数据承载能力；并且，上述逻辑的树形结

构的树形结构中包含的根节点和中间节点的层数，将处于一个相对稳定的状态；因此，根节点和中间节点存储容量的提升和根节点和中间节点的层数相对稳定，在某种程度上可以确保上述逻辑的树形结构的根节点和中间节点将具有更少的层数；从而，在根节点和中间节点具有更大的数据存储容量和数据承载能力，以及上述逻辑的树形结构的根节点和中间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，将上述逻辑的树形结构的前 N 层的根节点和中间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 层的根节点和中间节点读取到内存中时的 IO 读取次数，以及将逻辑的树形结构的前 N 层的根节点和中间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。

附图说明

[08]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；

[09]图 2 是一示例性实施例提供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；

[10]图 3 是一示例性实施例提供的一种区块链数据存储方法的流程图；

[11]图 4 是一示例性实施例提供的一种 FDMT 树的树形结构图；

[12]图 5 是一示例性实施例提供的另一种 FDMT 树的树形结构图；

[13]图 6 是一示例性实施例提供的一种 Tree node 的结构图；

[14]图 7 是一示例性实施例提供的一种 bucket 数据桶的结构图；

[15]图 8 是一示例性实施例提供的一种为 FDMT 树上的节点设置节点 ID 的示意图；

[16]图 9 是一示例性实施例提供的一种将账户状态数据写入 FDMT 树的示意图；

[17]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；

[18]图 11 是一示例性实施例提供的一种区块链数据存储装置的框图。

具体实施方式

[19]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。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。

[20]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。

[21]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。

[22]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。

[23]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事务（transaction）格式，然后发布至区块链，由区块链中的节点设备对该事务进行共识，并在达成共识后，由区块链中作为记账节点的节点设备，将这笔事务打包进区块，在区块链中进行持久化存证。

[24]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

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）。

[25]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。

[26]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：Balance 字段，用于维护账户目前的账户余额；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；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。

[27]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。

[28]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。

[29]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。

[30]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。

[31]以下以使用 MPT 树，来存储区块链数据为例进行说明；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。

[32]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 MPT 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。

[33]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。

[34]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。

[35]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。

[36]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。

[37]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。

[38]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。

[39]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。

[40]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。

[41]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）。

[42]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）。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。

[43]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。

[44]假设需要组织成 MTP 状态树的账户状态数据如下表 1 所示：

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

表 1

[45]其中，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。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。

[46]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。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。

[47]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。

[48]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)。

[49]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。

[50]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。

[51]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; 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。

[52]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。

[53]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。

[54]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。

[55]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。

[56]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:

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

表 2

[57]合约账户的存储空间中存储的合约数据, 通常可为状态变量的形式; 在进行存储时, 可将状态变量名以 key-value 键值对的形式组织成如图 2 所示的 storage 树进行存储。

[58]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。

[59]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。

[60]在实际应用中，MPT 树上存储的所有数据的 key 的共享字符前缀的字符长度，通常并不固定；而且，当 MPT 树上写入了新增的数据之后，可能导致上述共享字符前缀的字符长度，也会随之发生变化；因此，这就可能会导致 MPT 树上的扩展节点发生分裂，分裂出新的分支节点；也即，MPT 树上的节点的分裂条件是，上述共享字符前缀的字符长度发生变化；例如，以图 1 示出的 MPT 状态树为例，假设上述 MPT 状态树上新增了一个账户地址的前两位字符为“a8”的账户地址的账户状态数据，那么图 1 中示出的作为根节点的扩展节点的“Shared nibble”字段存储的共享字符前缀，就会由“a7”，改变为“a”；按照 MPT 状态树的节点的分裂条件，这个作为根节点的扩展节点，会分裂成一个存储的共享字符前缀为“a”的扩展节点；和一个字符“8”被占用的分支节点。

[61]一旦 MPT 树上的扩展节点发生分裂，可能会导致 MPT 树的节点层数也会发生变化，造成 MPT 树的节点层数不够稳定。由于上述 MPT 树上存储的所有数据的 key 的共享字符前缀的字符长度，会随着上述 MPT 树写入了新的数据，而频繁的变化；因此，按照以上示出的节点分裂方式，会造成节点的频繁分裂，进而会影响向 MPT 树写入新的数据时的数据存储效率。

[62]有鉴于此，本说明书提出一种新的逻辑的树形结构设计方案。

[63]在实现时，上述逻辑的树形结构仍然可以包括根节点、中间节点和叶子节点；其中，上述根节点、中间节点用于存储区块链数据的 key 中的字符；上述叶子节点用于存储区块链数据的 value。

[64]与上述 MPT 树不同的是，上述根节点、中间节点可以包括多个用于存放区块链数据的 key 中的字符的位置，每个位置中可以进一步包括多个用于存放区块链数据的 key 中的字符的槽位；而上述槽位则可以用于存储与该节点链接的下一层节点的 hash 值。

[65]区块链中的节点设备在存储区块链数据时，可以获取待存储的区块链数据的 key-value 键值对，再将该待存储的区块链数据转换成上述逻辑的树形结构上的根节点、中间节点和叶子节点；然后，将上述根节点、中间节点和叶子节点的 key-value 键值对再进一步存储在数据库中；上述叶子节点、所述中间节点和所述根节点的 key-value 键值对中，value 可以为节点的存储内容，key 可以为节点的存储内容的 hash 值。

[66]在以上技术方案中，一方面，由于逻辑的树形结构中的每一个根节点和中间节点，均包括分别代表不同字符的多个位置；并且，各位置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，使得每一个根节点和中间节点将具有更大的数据存储容量和数据承载能力，从而在将上述逻辑的树形结构中的根节点和中间节点写入数据库进行存储时；或者，在访问数据库中存储的逻辑的树形结构中的根节点和中间节点时，可以使得根节点和中间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；而且，上述逻辑的树形结构上节点的数据存储容量和数据承载能力的提升，势必也会导致上述逻辑的树形结构整体的数据存储容量和数据承载能力的提升，使得上述逻辑的树形结构上可以存储更多的区块链数据；第二方面，由于逻辑的树形结构中的每一个根节点和中间节点均采用统一的数据结构；对于逻辑的树形结构中的根节点和中间节点来说，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，可以避免由于根节点和中间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，从而可以确保逻辑的树形结构中所包含的根节点和中间节点的层数，始终处于一个相对稳定的状态；第三方面，由于通过这种设计，每一个根节点和中间节点将具有更大的数据存储容量和数据承载能力；并且，上述逻辑的树形结构的树形结构中包含的根节点和中间节点的层数，将处于一个相对稳定的状态；因此，根节点和中间节点存储容量的提升和根节点和中间节点的层数相对稳定，在某种程度上可以确保上述逻辑的树形结构的根节点和中间节点将具有更少的层数；从而，在根节点和中间节点具有更大的数据存储容量和数据承载能力，以及上述逻辑的树形结构的根节点和中间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，将上述逻辑的树形结构的前 N 层的根节点和中间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 层的根节点和中间节点读取到内存中时的 IO 读取次数，以及将逻辑的树形结构的前 N 层的根

节点和中间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。

[67]请参见图 3，图 3 是一示例性实施例提供的一种区块链数据存储方法的流程图。所述方法应用于区块链节点设备；所述方法包括以下步骤：步骤 302，获取待存储的区块链数据的 key-value 键值对；步骤 304，将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点；所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置，每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位；所述槽位用于存储与该节点链接的下一层节点的 hash 值；步骤 306，将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中；所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中，value 为节点的存储内容，key 为节点的存储内容的 hash 值。

[68]在本说明书中，为了避免节点频繁发生分裂，提出一种前 N 层的用于存储区块链数据的 key 中的字符节点的层数固定的逻辑的树状结构。

[69]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。

[70]上述逻辑的树状结构，具体可以包括根节点、中间节点和叶子节点；其中，上述逻辑的树状结构上的节点，可以通过其自身的 hash 值与上一层的节点进行链接。上述根节点和中间节点，具体用于存储区块链数据的 key-value 键值对中的 key 中的至少一个字符；上述叶子节点具体用于存储区块链数据的 value（即区块链数据的具体内容）。中间节点的层数可以为一层也可以为多层，在本说明书中不进行特别限定。

[71]例如，在一个例子中，上述区块链数据的 key，仍然可以包括字符前缀部分（Shared nibble）和字符后缀部分（key-end）；在这种情况下，根节点和中间节点可以用于存储上述字符前缀中的字符；而上述叶子节点则可以用于存储上述字符后缀和区块链数据的 value。

[72]一方面，由于上述逻辑的树形结构中的根节点和中间节点，可以存放区块链数据的 key 中的字符；因此，上述逻辑的树形结构具有 Trie 字典树的特点；另一方面，上述逻辑的树状结构上的节点，可以通过其自身的 hash 值与上一层的节点进行链接；因此，上述逻辑的树形结构也具有 Merkle 树的特点。综上，本说明书描述的逻辑的树形结构，实际上是一种类似于 MPT 树的融合了 Trie 字典树的树形结构的 Merkle 树变种。

[73]请参见图 4，图 4 为本说明书示出的一种 FDMT（Fixed Depth Merkle Tree）树的树形结构图。上述 FDMT 树是一种融合了 Trie 字典树的树形结构的 Merkle 树变种。

[74]如图 4 所示，在本说明书示出的 FDMT 树的树形结构中，包含前 N 层（图 4 示出的为 3 层，仅为示意性的）的 Tree node，和最后一层的 Leaf node（即叶子节点）。其中，在前 N 层的 Tree node 中，第一层为作为根节点的 Tree node，其它层为作为中间节点的 Tree node。

[75]其中，与以上描述的 MPT 树不同的是，上述 FDMT 树前 N 层的各个 Tree node（即根节点和中间节点），将采用统一的数据结构，从而使得上述 FDMT 树前 N 层的各个 Tree node，可以不再因为新写入了数据导致存储的字符的长度发生变化而发生节点分裂。

[76]如图 4 所示，上述 FDMT 树上的前 N 层的 Tree node，均可以包括分别代表不同字符的多个 block；上述 block 为用于存放区块链数据的 key 中的字符的“位置”。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；上述槽位也用于存放区块链数据的 key 中的字符。

[77]例如，图 4 示出的为每一个 Tree node 包括 N 个 block；每个 block 进一步包括 N 个 slot。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，上述 FDMT 树中的节点，通过其自身的 hash 值，链接至上一层的节点。相应的，上述槽位具体可以用于填充当前的 Tree node 所链接的下一层节点的 hash 值。Tree node 的下一层节点，具体仍然可以是 Tree node，也可以是 Leaf node。

[78] 其中,在实际应用中,当上述 FDMT 树上的 Tree node 中的任一 block 中的任一槽位填充的 hash 值发生更新后,通常需要重新计算该 Tree node 的 hash 值,并根据计算出的该 hash 值与上一层的节点重新进行链接。

[79] 而在计算该 Tree node 的 hash 值时,通常需要将 Tree node 中填充的所有数据都作为计算参数进行 hash 计算;因此,在 Tree node 采用了如图 4 所示的数据结构的情况下,如果需要计算该 Tree node 的 hash 值,则需要先计算出该 Tree node 包含的各个 block 的 hash,再将各 block 的 hash 拼接起来,再针对拼接起来的 hash 执行二次的 hash 计算。

[80] 通过这种方式,当上述 FDMT 树上的 Tree node 中的任一 block 中的任一槽位填充的 hash 值发生更新,在重新计算该 Tree node 的 hash 值时,即便该 Tree node 中其它的各个 block 未发生数据更新,也仍然需要作为计算参数参与 hash 计算,显然存在 hash 计算的计算量较大的问题。

[81] 基于此,在本说明书中,为了降低在计算 Tree node 的 hash 时的计算量,上述 Tree node 具体可以采用主 block (即主位置)和多个子 block (即子位置)的数据结构。

[82] 请参见图 5,图 5 为本说明书示出的另一种 FDMT 树的树形结构图。

[83] 如图 5 所示,在本说明书示出的 FDMT 树的树形结构中,仍然包含前 N 层的 Tree node,和最后一层的叶子节点。

[84] 其中,与图 4 示出的上述 FDMT 树的结构不同的是,如图 5 中示出的上述 FDMT 树上的前 N 层的 Tree node,均可以包括主 block(即图 5 中示出的 root block)和分别代表不同字符的多个子 block;而每一个 block 还可以进一步包括多个槽位;例如,图 5 示出的为每一个 Tree node 均包括一个主 block 和 N 个子 block;而每个子 block 又进一步包括 N 个 slot。

[85] 其中,上述主 block 和子 block 包含的槽位的功能,具体可以有所不同;如图 5 所示,对于主 block 而言,可以包含分别与各子 block 对应的多个槽位,每一个槽位具体可以用于填充对应的子 block 中存储的数据内容的 hash;例如,在计算任一子 block 的 hash 时,具体可以将该子 block 中各槽位填充的 hash 值拼接起来,再针对拼接起来的 hash 执行二次的 hash 计算,得到该子 block 的 hash。

[86] 而对于子 block 而言,可以包含分别代表不同的字符的多个槽位,每一个槽位具体可以用于填充该 Tree node 链接的下一层节点的 hash 值。

[87] 在本说明书中,按照图 5 示出的 Tree node 的结构,则可以用该 Tree node 中的主 block 的 hash 值来表示该 Tree node 的 hash 值;从而,在这种情况下,当上述 FDMT 树上的 Tree node 中的任一子 block 中的槽位填充的 hash 值发生更新后,则在重新计算该 Tree node 的 hash 值时,只需要将该发生数据更新的该子 block 中各个槽位中填充的 hash 值进行拼接,并将拼接出的 hash 值作为计算参数重新进行 hash 计算,再将计算出的 hash 值填充至该 Tree node 中的主 block 中与该子 block 对应的槽位,然后再对主 block 中的各槽位填充的 hash 进行拼接,再将拼接出的 hash 值作为计算参数重新进行 hash 计算即可得到该 Tree node 的 hash。

[88] 在整个过程中,对于未发生数据更新的其它子 block 中各个槽位填充的 hash 值,将不再需要进行拼接后作为计算参数参与 hash 计算,从而可降低在重新计算该 Tree node 的 hash 值时的 hash 计算量和计算时长,提升 hash 计算的计算效率。

[89] 需要解释的是,图 4 和图 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。

[90] 请继续参见图 4 和图 5,图 4 和图 5 示出的上述 FDMT 树上的各 Tree node,均可以用于存储上述区块链数据的 key 中的至少部分字符。

[91] 对于图 4 示出的上述 FDMT 树上的各 Tree node,实际存储的字符,具体可为该 Tree node 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。

[92] 而对于图 5 示出的上述 FDMT 树上的各 Tree node,实际存储的字符,具体可以是该 Tree node 中的子 block 所代表的字符,与该 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。

[93] 其中，在实际应用中，Tree node 中的每一个 block，可以仅代表一位字符；也即，基于图 4 和 5 示出的 Tree node 的存储格式，每一个 Tree node 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。

[94] 例如，请参见图 6，图 6 为本说明书示出的一种 Tree node 的结构图；如图 6 所示，该 Tree node 包含 16 个代表不同的 16 进制字符的子 block；每一个子 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 Tree node 中的子 block6（代表了 16 进制字符 6）为非空 block，该子 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 Tree node 链接的下一层节点的 hash 值的非空 slot；则该 Tree node 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”。

[95] 其中，上述 Tree node 中所包含的子 block 的数量，以及每一个子 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 Tree node 所包含的子 block 的数量；以及，子 block 所包含的槽位的数量；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 Tree node 中所包含的子 block 的数量，以及每一个子 block 所包含的槽位的数量都可以为 16。

[96] 在实际应用中，上述 Tree node 所包含的子 block 的数量，和上述子 block 所包含的槽位的数量，可以保持相同；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 Tree node 可以均包括 16 个分别代表不同的 16 进制字符的子 block；而每一个子 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。

[97] 通过这种方式，则上述 FDMT 树前 N 层每一层中的单个 Tree node，可以具有 $16 \times 16 = 256$ 个槽位，显然图 4 示出的 FDMT 树上的 Tree node 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。

[98] 当前，在实际应用中，上述 Tree node 所包含的子 block 的数量，和上述子 block 所包含的槽位的数量，具体也可以不相同；

[99] 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 Tree node 可以均包括 16 个分别代表不同的 16 进制字符的子 block；而每一个子 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 Tree node 可以均包括 10 个分别代表不同的 10 进制字符的子 block；而每一个子 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。

[100] 在示出的一种实施方式中，上述 FDMT 树包含的 Tree node 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 Tree node，并且包含的 Tree node 的层数相对固定的 Merkle 树。

[101] 例如，在一个例子中，以上述区块链数据的 key 为区块链账户地址为例，假设区块链系统支持的区块链账户地址被设计为，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 Tree node 存储的区块链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 FDMT 树可以被设计成包含三层 Tree node 的树状结构。

[102] 基于图 4 和图 5 示出的 FDMT 树的树形结构设计，一方面，由于 FDMT 树中的每一个 Tree node，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 Tree node 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 Tree node 写入数据库进行存储时；或者，在访问数据库中存储的上述 FDMT 树中的 Tree node 时，可以使得

Tree node 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;而且,上述 FDMT 树上单个 Tree node 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;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。

[103]但是,如果采用如图 4 和图 5 所示出的 FDMT 树的树形结构设计,假设上述前 N 层的 Tree node 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 前 N 层每一层中的单个 Tree node,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 Tree node 的最大存储容量则是 $256*32\text{byte}=8192\text{byte}=8\text{kb}$,刚好是两个扇区的容量。显而易见的,图 4 和图 5 示出的 FDMT 的树形结构中的每一个 Tree node 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。

[104]而且,上述 FDMT 上单个 Tree node 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 整体的数据存储容量和数据承载能力的提升,使得上述 FDMT 上可以存储更多的区块链数据;例如,假设图 4 和图 5 示出的 FDMT 的树形结构包含 3 层 Tree node,每一层的 Tree node 可以均包括 16 个分别代表不同的 16 进制字符的子 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 Tree node,可以具有 $16*16=256$ 个槽位;则三层 Tree node 共计可以承载 $256*256*256=16.77\text{M}$ 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 FDMT 树最多可以承载 $16.77\text{M}*16$ 条区块链数据;显而易见的,相较于图 1 示出的 MPT 树,图 4 和图 5 示出的上述 FDMT 可以存储更多的区块链数据,具有更大的数据承载能力。

[105]第二方面,由于 FDMT 中的每一个 Tree node 均采用统一的数据结构;对于上述 FDMT 中各层的 Tree node 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 Tree node 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 的树形结构中所包含的 Tree node 的层数,始终处于一个相对稳定的状态。

[106]第三方面,由于通过这种设计,每一个 Tree node 将具有更大的数据存储容量和数据承载能力;并且,上述 FDMT 的树形结构中包含的 Tree node 的层数,将处于一个相对稳定的状态;因此,Tree node 存储容量的提升和 Tree node 的层数相对稳定,在某种程度上可以确保上述 FDMT 的 Tree node 将具有更少的层数;从而,在 Tree node 具有更大的数据存储容量和数据承载能力,以及上述 FDMT 的 Tree node 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 的前 N 层的 Tree node 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 Tree node 读取到内存中时的 IO 读取次数,以及将 FDMT 的前 N 层的 Tree node 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。

[107]例如,以图 4 和图 5 示出的 FDMT 为 3 层 Tree node 固定,且每一层的 Tree node 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 Tree node 的最大存储容量则是 $256*32\text{byte}=8192\text{byte}=8\text{kb}$;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 Branch Node 节点的存储容量 512byte,远小于图 4 和图 5 示出的 FDMT 树上的 Tree node;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 FDMT 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。

[108] 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 4 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 4 和图 5 示出的 FDMT 树的 IO 读取次数的至少 8 倍。

[109] 显而易见的，系统在冷启动时，针对图 4 和图 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 4 和图 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。

[110] 在示出的一种实施方式中，上述区块链数据的 key 对应的字符串仍然可以包括字符前缀和字符后缀；在这种情况下，上述 Tree node 可以用于存储所述区块链数据的 key 的字符前缀中的字符；上述叶子节点可以用于存储所述区块链数据的 key 的字符后缀和上述区块链数据的 Value。

[111] 其中，由于上述叶子节点实际存储的数据，相对于上述 Tree node 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。

[112] 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；在示出的一种实施方式中，上述叶子节点具体可以是 bucket 数据桶的形式；其中，上述 bucket 数据桶具体可以是用于存储数据的容器或者存储空间。

[113] 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；如图 7 所示，在上述 bucket 数据桶（即图 7 中示出的 bucket node）中，可以包括若干条数据记录；其中，上述 bucket 数据桶中包含的上述若干条数据记录，在逻辑上可以不是一个整体，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 value；也即，一条数据记录，是指一条存储的数据内容包括所述区块链数据的 value 的存储记录。

[114] 上述 bucket 数据桶中包含的上述若干条数据记录，在逻辑上不是一个整体，是指各条数据记录都可以分别对应一个独立的查询键值（key），从而可以基于各条数据记录的查询键值，来针对上述 bucket 数据桶中的各条数据记录分别进行精确查询，而并不需要对上述 bucket 数据桶中存储的所有数据记录进行整体的读取。

[115] 其中，各条数据记录分别对应的查询键值的具体形式和具体内容，在本说明书中不进行特别限定，可以是任意形式的能够作为各条数据记录的查询索引的字符串。

[116] 在示出的一种实施方式中，各条数据记录对应的查询键值具体可以是各条数据记录包含的数据内容的 hash 值；上述 bucket 数据桶中包含的上述数据记录，则具体可以包括由上述区块链数据的 value 的 hash 值，和上述区块链数据的 value 对应的数据内容构成的 key-value 键值对。

[117] 当然，在实际应用中，各条数据记录对应的查询键值也可以是 hash 值以外的其它形式的能够作为各条数据记录的查询索引的字符串，在本说明中不进行特别的限定；例如，在一个例子中，各条数据记录对应的查询键值具体也可以是由节点设备为各条数据记录分别设置的一个唯一的标识（比如编号）。

[118] 在示出的一种实施方式中，如果上述 Tree node 用于存储上述区块链数据的 key 的字符前缀中的字符；则上述叶子节点用于存储上述区块链数据的 key 的字符后缀和上述区块链数据的 Value；相应的，上述 bucket 数据桶中包含的上述数据记录，具体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 对应的数据内容整体进行 hash 计算得到的 hash 值，和上述区块链数据的 key 的字符后缀和上述区块链数据的 value 对应的数据内容构成的 key-value 键值对。

[119] 在实际应用中，上述数据记录具体也可以是 key-value 键值对以外的其它类型的数据形式，在本说明书中不再进行一一列举。

[120] 通过以上实施方式，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 FDMT 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;例如,以上述区块链数据的 value 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。

[121] 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;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。

[122] 其中,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。

[123] 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。

[124] 其中,上述 bucket 数据桶中所存储的数据记录的总存储容量,在本说明书中也不进行特别限定;在示出的一种实现方式中,上述 bucket 数据桶中所存储的数据记录的总存储容量,具体可以由用户进行自定义配置。

[125] 例如,在实现时,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,在这种情况下,用户可以将上述 bucket 数据桶中所存储的数据记录的最大存储容量,设置为 4KB;或者,4KB 的整数倍,以使上述 bucket 数据桶的最大存储容量,能够与上述存储介质的单个物理扇区的存储容量相适配。

[126] 在本说明书中,如前所述,上述 FDMT 树上前 N 层的各 Tree node,由于其采用统一的数据结构;因此,上述 FDMT 树上前 N 层的各 Tree node,将不再发生节点分裂。而上述 FDMT 树上的叶子节点,仍然可以在满足一定的分裂条件时进行节点分裂。

[127] 与上述 MPT 树不同的是,在本说明书中,上述 FDMT 树上的叶子节点在进行节点分裂时,可以不再按照共享字符前缀的字符长度发生变化来进行分裂,而是按照叶子节点的存储容量进行分裂。当上述 FDMT 树上的叶子节点的存储容量满足节点分裂条件时,再从该叶子节点中分裂出中间节点(即图 5 中示出的 Extend node)。分裂出的中间节点,为上述叶子节点的上层节点。对同一个叶子节点而言,可以进行多次分裂。

[128] 其中,上述节点分裂条件,可以包含与上述叶子节点的存储容量相关的任意形式的条件,在本说明书中不进行特别限定;在示出的一种实施方式中,上述节点分裂条件,具体可以是以下示出的条件 1 和条件 2 中的任意一个;或者,也可以是以下示出的条件

1 和条件 2 中的组合：条件 1：上述叶子节点中存储的数据记录的总数量大于阈值；条件 2：上述叶子节点中存储的数据记录的总存储容量大于阈值。

[129] 例如，在一个例子中，可以将上述节点分裂条件设置为，当上述叶子节点中存储的数据记录的总数量大于 64 条；和/或，上述叶子节点中存储的数据记录的总存储容量大于 4KB 时，则从该叶子节点中分裂出中间节点。

[130] 在本说明书中，上述 Extend node，也是上述 FDMT 树的中间节点，可以采用与以上描述的 Tree node 完全相同的数据结构，具体用于存储从上述叶子节点存储的字符后缀中分裂出的字符；例如，请参见图 4 和图 5，从叶子节点中分裂出的 Extend node，也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；该槽位具体可以用于填充该节点链接的下一层节点的 hash 值。比如，对于 Extend node 而言，其下一层节点为上述叶子节点；因此，Extend node 中的 block 所包含的槽位中，可以用于填充该 Extend node 链接的下一层叶子节点的 hash 值。

[131] 在本说明书中，对于从上述叶子节点中分裂出的 Extend node 而言，还可以按照 Extend node 的存储容量与其链接的下层的叶子节点进行合并。当上述 FDMT 树上的 Extend node 的存储容量满足节点合并条件时，可以将该 Extend node 合并至与其链接的下层的叶子节点。

[132] 其中，将 Extend node 合并至与其链接的下层的叶子节点，是指将该 Extend node 中存储的字符作为上述字符后缀，写入与该 Extend node 链接的下层的一个或者多个叶子节点。

[133] 其中，上述节点合并条件，也可以包含与上述 Extend node 的存储容量相关的任意形式的条件，在本说明书中不进行特别限定；在示出的一种实施方式中，上述节点合并条件，具体可以是以下示出的条件 3 和条件 4 中的任意一个；或者，也可以是以下示出的条件 3 和条件 3 中的组合：条件 1：上述 Extend node 中存储的字符的总数量小于或等于阈值；也即，上述 Extend node 中各个 block 中非空 slot（即填充了 hash 值的 slot）的总数量；条件 2：上述 Extend node 中存储的字符的总存储容量小于或等于阈值。也即，上述 Extend node 中各个 block 中非空 slot 中填充的 hash 值的总存储容量；例如，在一个例子中，可以将上述节点合并条件设置为，当上述 Extend node 中存储的字符的总数量小于或者等于 1 条；和/或，上述 Extend node 中存储的字符的总存储容量小于或者等于 1KB 时，则将该 Extend node 存储的字符作为字符后缀，写入该 Extend node 链接的下层的一个或者多个叶子节点。

[134] 通过以上描述可知，本说明书中描述的 Extend node，可以是上述 FDMT 树上动态可伸缩的中间节点；当上述 FDMT 树上的任一叶子节点的存储容量满足节点分裂条件时，可以从该叶子节点中分裂出至少一个 Extend node，以及至少一个叶子节点；当任一 Extend node 的存储容量满足节点合并条件时，可以将该 Extend node 合并至与其链接的下层的叶子节点。

[135] 在本说明书中，待存储的区块链数据，具体可以包括至少以下四类数据：区块中收录的交易；在区块中的交易执行完毕后，与所述区块中收录的交易对应的交易收据；在区块中的交易执行完毕后，与所述区块链中的区块链账户对应的最新账户状态数据；智能合约账户的存储内容；相应的，上述 FDMT 树具体也可以包括：用于存储区块中收录的交易的交易树；用于存储与区块中收录的交易对应的交易收据的收据树；用于存储与所述区块链中的区块链账户对应的最新账户状态数据的状态树；用于存储智能合约账户的存储内容的存储树。

[136] 当然，在实际应用中，也可以只利用上述 FDMT 树的树形结构来存储上述四种数据中的部分类型的数据；比如，只利用上述 FDMT 树来存储上述区块链账户对应的最新账户状态数据，其他类型的数据可以利用其他形式的二叉树来存储（比如 MPT 或者其他形式的二叉树）。

[137] 其中，上述交易树、收据树和状态树的根节点的 hash 值，可以存储在区块头中；上述存储树的根节点的 hash 值，可以存储在与该存储树对应的合约账户的结构体中的 Storage 字段中。

[138] 上述区块链数据的 key，具体可以是指区块链数据在数据库中对应的查找键值；相

应的,上述区块链数据的 Value,具体可以是上述区块链数据的原始内容;其中,在实际应用中,上述查找键值具体可以是一个与区块链数据对应的字符串;当上述区块链数据为不同类型的数据时,其对应的 key 也会存在一定的差异;例如,当上述区块链数据为区块中收录的交易;或者,在区块中的交易执行完毕后,与该区块中收录的交易对应的交易收据时,此时与该区块链数据对应的 key,具体可以是交易在区块中的序号;或者,其它形式的交易标识。

[139] 当上述区块链数据为在区块中的交易执行完毕后,与区块链中的区块链账户对应的最新账户状态数据时,此时与该区块链数据对应的 key,具体可以是区块链账户的账户地址。

[140] 当上述区块链数据为智能合约账户的存储内容时,此时与该区块链数据对应的 key,具体可以是合约账户的账户地址和上述存储内容在合约账户的账户存储中的存储位置的 hash 值;比如,在一个例子中,上述存储内容通常可以是状态变量,则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。

[141] 在本说明书中,当区块链中的节点设备在存储区块链数据时,可以先获取待存储的区块链数据的 key-value 键值对;例如,在一个例子中,节点设备可以在获取到待存储的区块链数据时,将该区块链数据处理成 key-value 键值对;在获取到待存储的区块链数据的 key-value 键值对之后,可以将该待存储的区块链数据的 key-value 键值对转换成上述逻辑的树形结构上的根节点、中间节点和叶子节点;例如,在一个例子中,上述节点设备可以搭载一个与上述 FDMT 树对应的存储接口或者存储服务,该存储接口或者存储服务具体可以用于将待存储的区块链数据的 key-value 键值对,转换成 FDMT 树上的节点;则该节点设备在获取到待存储的区块链数据的 key-value 键值对之后,可以通过调用该存储接口或者存储服务,按照图 4 或者图 5 示出的 FDMT 树的树形结构,将待存储的区块链数据的 key-value 键值对,转换成图 4 或者图 5 上的根节点、中间节点和叶子节点。

[142] 在将存储的区块链数据的 key-value 键值对转换成上述逻辑的树形结构上的根节点、中间节点和叶子节点之后,可以将这些根节点、中间节点和叶子节点以 key-value 键值对的形式存储在数据库中;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 FDMT 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 FDMT 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。

[143] 其中,上述数据库的具体类型,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;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。

[144] 上述 FDMT 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,具体可以是上述 FDMT 树上的节点的节点 ID;其中,上述节点 ID 具体可以包括能够唯一标识上述 FDMT 树上的节点的标识信息;例如,在一种实现方式中,上述节点 ID 具体可以是上述 FDMT 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 FDMT 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。

[145] 在另一种实现方式中,上述节点 ID 具体也可以包括上述 FDMT 树上的节点在上述 FDMT 树中的路径信息;也即,将上述 FDMT 树上的节点在上述 FDMT 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 FDMT 树上的位置的任意形式的信息;例如,请参见图 8,图 8 为本说明书示出的一种为 FDMT 树上的节点设置节点 ID 的示意图;对于如图 8 所示包含三层 Tree node 的 FDMT 树,假设作为根节点的 tree node 的节点 ID 用 0x00 表示,那么图 8 示出的 bucket node 的节点 ID,可以表示成 0x00123456。

[146] 其中,0x00 为根节点的节点 ID;123456 是指上述 bucket node 在 FDMT 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。

[147] 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 FDMT 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。

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

[149] 在另一种实现方式中, 上述节点 ID 具体也可以包括上述 FDMT 树上的节点在上述 FDMT 树中的相对位置, 和该节点所包含的数据内容的 hash 值 (此时该节点 ID 也可以作为该节点的 hash 标识); 也即, 将上述 FDMT 树上的节点在上述 FDMT 树中的相对位置, 和该节点所包含的数据内容的 hash 值作为该节点的节点 ID。

[150] 例如, 在实现时, 可以将节点在上述 FDMT 树中的相对位置, 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相对位置和上述 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。

[151] 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 FDMT 树上存储的节点时, 可以精确定位到该节点在 FDMT 树上所在的具体槽位。

[152] 其中, 上述节点设备上用于承载上述数据库的存储介质, 具体可以是持久性存储介质; 例如, 可以是能够对数据进行持久性存储的磁盘、内存或者其它形式的存储介质, 在本说明书中不再进行一一列举。

[153] 在本说明书中, 在将上述 FDMT 树上的节点的 key-value 键值对分别写入上述数据库之前, 还可以预先对上述 FDMT 树上的节点进行编码处理, 然后将编码处理后的节点的 key-value 键值对在数据库中进行存储。

[154] 其中, 由于上述 FDMT 树上的 Tree node 和 Extend node 均采用了, 包含多个 block, 每一个 block 又进一步包含多个槽位的数据结构; 因此, 在本说明书中, 在针对上述 FDMT 树进行编码时, 可以针对上述 FDMT 树上的 Tree node 和 Extend node 中的 block 执行 bitmap 编码。

[155] 在针对上述 FDMT 树上的 Tree node 和 Extend node 中的 block (可以包括主 block 和子 block) 执行 bitmap 编码时, 具体可以统计出该 Tree node 和 Extend node 中的 block 中的各槽位是否填充了 hash 值, 然后再将统计结果利用特殊的编码字符表示出来得到 bitmap 编码信息, 最后再将该 bitmap 编码信息添加至 block 中, 完成针对该 block 的 bitmap 编码过程; 例如, 在一个例子中, 上述 bitmap 编码信息, 具体可以是一个用于标识 block 中的各槽位是否填充了 hash 值的 16 进制字符串; 假设待编码的 block (可以是主 block, 也可以是子 block) 共计包含 16 个槽位; 在这 16 个槽位中第 0 位、第 8 位、第 9 位、第 10 位、第 11 位分别填充了 hash 值; 则该 block 中各个槽位是否填充了 hash 的统计结果, 可以用二进制字符串 0000 1111 0000 0001 表示; 其中, 该二进制字符串最左边是最高位第 15 位, 最右边是最低位第 0 位; 最后, 可以将该二进制字符串转换成为 16 进制字符串 0x0f01; 此时该 16 进制字符串 0x0f01 即为该 block 的 bitmap 编码信息。

[156] 其中, 在针对上述 FDMT 树上的 Tree node 和 Extend node 中的 block 执行 bitmap 编码时, 无论是主 block 还是子 block, 采用 bitmap 编码处理过程是完全相同的。

[157] 通过这种方式, 由于编码后的上述 FDMT 树中的 Tree node 和 Extend node 包含的 block 中被添加了, 能够指示该 block 中的各槽位是否填充了 hash 值的 bitmap 编码信息; 因此, 在需要查询数据库中存储的各 Tree node 中的 block 里填充的数据时, 通过该 block 中的 bitmap 编码信息, 就可以确定该 block 里的非空槽位, 而不再需要遍历该 block 中的各个槽位, 从而可以提升查找效率。

[158] 其中, 由于上述 FDMT 树上的叶子节点采用了与 Tree node 和 Extend node 完全不

同的数据结构；因此，在对上述 FDMT 树上的叶子节点进行编码时，也可以采用与上述 Tree node 和 Extend node 完全不同的编码方式。

[159] 在示出的一种实施方式中，在针对上述 FDMT 树上的叶子节点进行编码时，具体仍然可以采用 RLP (Recursive Length Prefix, 递归长度前缀编码) 编码；其中，RLP 编码是如图 1 示出的 MPT 树常用的一种编码方式，其具体的编码过程，在本说明书中不再进行详述。

[160] 当然，在实际应用中，针对上述 FDMT 树上的叶子节点进行编码时所采用的编码方式，也可以是 RLP 编码以外的其它形式的编码方式，在实际应用中，可以灵活的选择，在本说明书中不再进行一一列举。

[161] 在本说明书中，在针对上述 FDMT 树上的节点编码完成后，可以将上述 FDMT 树上的节点的 key-value 键值对存储至上述数据库。

[162] 其中，在示出的一种实施方式中，对于上述 FDMT 树上的 Tree node 和 Extend node 而言，在将编码完成的 Tree node 和 Extend node 的 key-value 键值对存储到数据库之前，可以根据编码处理后的 Tree node 和 Extend node 中的 block 被添加的 bitmap 编码信息，来确定该 block 中未填充 hash 值的空槽位，然后将确定出的空槽位从该 block 中删除。

[163] 通过这种方式，可以在将各 Tree node 和 Extend node key-value 键值对存储至数据库中时，删除各 block 中的空槽位，可以进一步节约将各 Tree node 和各 Extend node 存储至数据库时所占用的存储空间，提升存储效率。

[164] 其中，对于编码后的 block，在计算该 block 的 hash 值时，可以将编码后的 block 所包含的 bitmap 编码信息和非空字符槽位填充的 hash 值作为一个整体进行 hash 计算；当然，也可以将 bitmap 编码信息排除在外。

[165] 在本说明书中，在将上述 FDMT 树上的节点 key-value 键值对分别写入上述数据库之后，如果上述 FDMT 树上写入的某一条区块链数据对应的 value 发生更新，此时可能需要对上述 FDMT 树上用于存储该区块链数据的字符后缀和 value 的叶子节点，从该叶子节点中分裂出的 Extend node，以及用于存储该区块链数据的 key 的字符前缀的 Tree node 分别进行更新。

[166] 当然，如果用于存储该区块链数据的字符后缀和 value 的叶子节点没有发生过分裂，可以只需要对上述 FDMT 树上用于存储该区块链数据的字符后缀和 value 的叶子节点，以及用于存储该区块链数据的 key 的字符前缀的 Tree node 分别进行更新即可。

[167] 在这种情况下，节点设备可以在上述数据库中查找与上述需要数据更新的区块链数据对应的节点；其中，具体的查找方式不再赘述；再将查找到的节点从上述数据库读取到内存中，在内存中对这些节点进行修改更新，然后再将更新后的节点，进一步写入上述数据库对原有的更新节点进行更新。

[168] 例如，对上述 FDMT 树上发生数据更新的叶子节点进行更新，具体可以包括对该叶子节点中存储的上述区块链数据的 value 进行更新。在更新完成后，可以再重新计算该叶子节点的 hash 值，并基于该 hash 值与该节点上一层的节点进行重新链接。

[169] 而对上述 FDMT 树上发生数据更新的叶子节点的上层的 Extend node 或者 Tree node 进行更新，具体可以包括对该 Extend node 或者 Tree node 中填充了该叶子节点的 hash 值的槽位进行更新；在更新完成后，可以再重新计算 Extend node 或者 Tree node 的 hash 值，并基于该 hash 值与该节点上一层的节点继续进行重新链接。

[170] 在示出的一种实施方式中，针对上述 FDMT 树上的叶子节点的分裂判断及具体的节点分裂操作，可在重新计算该叶子节点的 hash 值，并基于该 hash 值与该叶子节点上一层的节点（可以是 Tree node，也可以是 Extend node）进行重新链接的阶段来完成。

[171] 在这种情况下，节点设备可以确定上述 FDMT 树上的叶子节点是否发生数据更新，如果上述 FDMT 树上的任一叶子节点发生数据更新，则节点设备可以在重新计算该叶子节点的 hash 值，并基于该 hash 值与该叶子节点上层的 Tree node 或者 Extend node 进行重新链接之前，进一步确定该叶子节点的存储容量是否满足节点分裂条件；如果该叶子节点的存储容量满足节点分裂条件，可以进一步从该叶子节点分裂出至少一个 Extend node。

[172] 例如，在一个例子中，在叶子节点的存储容量满足节点分裂条件时，为了快速降低叶子节点的存储容量，可以将该叶子节点分裂为，一个 Extend node 和多个叶子节点。

[173] 其中，从上述叶子节点中分裂出 Extend node 的分裂策略，在本说明书中也不进行特别限定，在实际应用中，用户可以根据具体的分裂需求，来进行自定义设置；在示出的一种实施方式中，上述分裂策略具体可以包括：将在当前的成块周期内写入叶子节点的最新的数据记录，从该叶子节点中分裂出去，并基于该最新的数据来额外创建 Extend node 和叶子节点。

[174] 在这种情况下，在对任一目标叶子节点进行节点分裂时，可以确定在当前的成块周期内写入该目标叶子节点的最新的数据记录；再将这些最新的数据记录从该目标数据节点中删除，并从这些最新的数据记录所包含的字符后缀中分裂出字符前缀；例如，在一个例子中，默认可以将这些最新的数据记录的前两位分裂出去；或者，在这些最新的数据记录包含的字符后缀中存在共享字符前缀，并且该共享字符前缀的长度达到两位时，将该共享字符前缀分裂出去。

[175] 然后，再进一步创建用于存储分裂出的该字符前缀的至少一 Extend node，以及用于存储分裂出所述字符前缀之后的所述最新的数据记录的至少一叶子节点。

[176] 通过这种方式，由于可以将当前的成块周期内写入该目标叶子节点的最新的数据记录，从该目标叶子节点中删除，并基于这些删除的最新的的数据记录重新创建至少一个 Extend node 和至少一个叶子节点作为分裂出去的节点；因此，对于该目标叶子节点而言，在其进行节点分裂之后，该目标叶子节点实际存储的数据记录，将与上一个成块周期该目标叶子节点存储的数据记录完全一致；该目标叶子节点的 hash 值相对于上一个成块周期不会发生任何变化。从而，当该目标叶子节点在当前的成块周期写入了新的区块链数据，导致该目标叶子节点的存储容量满足了节点分裂条件时，只需要将新写入的数据记录分裂出去重新创建额外的 Extend node 和叶子节点即可，而不再需要重新计算该目标叶子节点的 hash 值与上一层的节点重新进行重新链接；通过这种分裂方式，不仅可以保证满足节点分裂条件的叶子节点的 hash 值处于一个稳定的状态，而且可以减少重新计算该叶子节点的 hash 值的计算次数。

[177] 在示出的另一种实施方式中，上述分裂策略具体可以包括：将叶子节点存储的若干条数据记录所包含的字符后缀之间的共享字符前缀，从该叶子节点中分裂出去，并基于该共享字符前缀来额外创建 Extend node。

[178] 在这种情况下，在对任一目标叶子节点进行节点分裂时，可确定目标叶子节点存储的若干条数据记录所包含的字符后缀之间是否存在共享字符前缀；如果该目标叶子节点存储的若干条数据记录所包含的字符后缀之间存在共享字符前缀，可将该共享字符前缀从上述若干条数据记录中删除，并创建用于存储该共享字符前缀的至少一 Extend node。

[179] 例如，由于上述 Extend node 也可以采用与 Tree node 相同的数据结构；因此，上述 Extend node 存储的字符，也可以是 block 代表的字符和 block 中填充了 hash 值的 slot 所代表的字符拼接起来的，长度为两位的字符串；则在目标叶子节点存储的若干条数据记录所包含的字符后缀之间存在共享字符前缀，并且该共享字符前缀的长度达到两位时，可以将该共享字符前缀从上述若干条数据记录中分裂出去。

[180] 在示出的一种实施方式中，针对上述 FDMT 上的 Extend node 的合并判断以及具体的节点合并操作，也可以在重新计算该 Extend node 的 hash 值，并基于该 hash 值与该 Extend node 上一层的节点进行重新链接的阶段来完成。

[181] 在这种情况下，节点设备可以确定上述 FDMT 树上的 Extend node 是否发生数据更新，如果上述 FDMY 树上的任一从叶子节点中分裂出的 Extend node 发生数据更新，则节点设备在重新计算该 Extend node 的 hash 值，并基于该 hash 值与该 Extend node 上层的节点进行重新链接之前，可以进一步确定该 Extend node 的存储容量是否满足节点合并条件；如果该 Extend node 的存储容量满足节点分裂条件，可以进一步将该 Extend node 合并至与该 Extend node 链接的下一层的叶子节点。

[182] 例如，在实现时，首先可以确定该 Extend node 链接的下一层的叶子节点；比如，可以根据该 Extend node 包含的 block 中的非空 slot 中填充的 hash 值，来确定该 Extend node 链接的下一层的叶子节点；然后，可以将该 Extend node 存储的字符作为上述字符

后缀的一部分，写入确定出的该 Extend node 链接的下一层的叶子节点。

[183] 在实际应用中，当一个叶子节点发生多次节点分裂时，此时该叶子节点可分裂出多层 Extend node；在这种情况下，当任一 Extend node 满足节点合并条件，如果其下层节点仍然是 Extend node，不是叶子节点，则该 Extend node 暂时无法进行节点合并；在这种情况下，可等待其下层的 Extend node 与更下层的叶子节点完成合并形成新的叶子节点之后，再与下层的该新的叶子节点进行二次合并，具体的合并方式不再赘述。

[184] 以下以上述区块链数据为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，对将上述区块链数据以 key-value 键值对的形式，写入采用如图 4 和图 5 所示的 FDMT 树的具体过程，进行详细描述。

[185] 在实现时，接入区块链的用户客户端，可以将交易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；其中，具体的共识过程不再赘述。

[186] 区块链中的节点设备在执行了目标区块中的交易之后，区块链中与这些被执行的交易相关的目标账户的账户状态，通常也会随之发生变化；因此，节点设备在目标区块中的交易执行完毕后，可以获取上述目标账户发生账户更新后的最新账户状态数据（即目标区块中的交易执行后的账户状态），并将获取到的上述目标账户的最新账户状态数据处理成 key-value 键值对；其中，该 key-value 键值对的 key 为该目标账户的账户地址；该 key-value 键值对的 value 为该目标账户的最新账户状态。

[187] 当将获取到的上述目标账户的最新账户状态数据处理成 key-value 键值对之后，可以将该 key-value 键值对转换成为上述 FDMT 树上的 Tree node、Extend node 和 Leaf node，并将上述 Tree node、Extend node 和 Leaf node 的 key-value 键值对在数据库中进行存储。

[188] 请参见图 9，图 9 为本说明书示出的一种将账户状态数据写入 FDMT 状态树的示意图；在图 9 示出的上述 FDMT 状态树中，前三层为 Tree node，最后一层为叶子节点；其中，上述叶子节点可以采用以上描述的 bucket 数据桶的存储结构。各层的 Tree node 均包括主 block 和 16 个分别代表不同的 16 进制字符 1~f 的子 block；并且，各子 block 进一步包括 16 个分别代表不同的 16 进制字符 1~f 的槽位。

[189] 假设上述目标账户的账户地址为“a71135125”，最新的账户状态数据为“statel”；此时，该账户地址的字符前缀(Shared nibble)为“a71135”；字符后缀（key-end）为“125”。

[190] 在将该账户地址“a71135125”的账户状态数据“statel”以 key-value 键值对的形式写入上述 FDMT 树时，首先可以在数据库中定位该 FDMT 树上作为根节点的 Tree node（比如根据区块头中填充的根节点的 hash 来定位根节点）；如果是首次数据写入，根节点尚未创建，此时可以创建根节点；再从第一层的根节点 Tree node1 开始，依次在三层 Tree node 中确定与该账户地址的字符前缀“a71135”对应的字符槽位。如图 9 所示，与该账户地址的字符前缀“a71135”对应的字符槽位，具体包括：Tree node1 的第 11 个子 block（代表字符 a）中的第 8 个槽位（代表字符 7）；Tree node2 的第 2 个子 block（代表字符 1）中的第 2 个槽位（代表字符 1）；Tree node3 的第 4 个子 block（代表字符 3）中的第 6 个槽位（代表字符 5）。

[191] 在确定以上槽位后，可以在 Tree node3 的第 4 个子 block 的第 6 个槽位链接的 bucket node 中，写入由字符后缀“125”和状态数据“statel”构成的数据记录；当然，如果 bucket node 中已经存在与字符后缀“125”对应的数据记录，表明 FDMT 树上写入过上述账户地址“a71135125”的历史账户状态数据对应的 value；此时可以将该数据记录中存储的 Value 更新为“statel”即可。

[192] 数据写入完成后，可重新计算该 bucket node 包含的数据内容的 hash 值，将该 hash 值填充到 Tree node3 的第 4 个 block 的第 6 个槽位中，对该槽位原有的 hash 值进行更新。

[193] 进一步的，在将 Tree node3 的第 4 个 block 的第 6 个槽位中原有的 hash 值进行更新之后，再重新计算 Tree node3 的主 block 包含的数据内容的 hash 值，将该 hash 值填充到 Tree node2 的第 2 个 block 的第 2 个槽位，对该槽位原有的 hash 值进行更新。

[194] 然后，在将 Tree node2 的第 2 个 block 的第 2 个槽位原有的 hash 值进行更新之后，

再重新计算 Tree node2 的主 block 包含的数据内容的 hash 值, 将该 hash 值继续填充到 Tree node1(即根节点)的第 11 个 block 的第 8 个槽位, 对该槽位原有 hash 值进行更新。

[195] 当根节点 Tree node1 的第 11 个 block 的第 8 个槽位原有的 hash 值也更新完成后, 此时再重新计算根节点 Tree node1 的主 block 包含的数据内容的 hash 值, 并基于该 hash 值对区块头里存储的 FDMT 状态树的根节点的 hash 值进行更新。当区块头里存储的该 FDMT 状态树的根节点的 hash 值更新完成后, 此时针对该 FDMT 状态树的更新完成, 上述账户地址“a71135512”和对应的账户状态数据“state1”构成的 key-value 键值对, 成功写入到 FDMT 状态树。

[196] 以上述区块链数据为区块链账户对应的最新账户状态数据为例, 仅为示例性的; 在实际应用中, 当上述区块链数据为以上描述的 4 种区块链数据的其它类型的区块链数据时, 将其写入相应的 FDMT 树的具体过程, 与以上述描述的实现过程类似, 在本说明书中不再进行一一赘述。

[197] 与上述方法实施例相对应, 本申请还提供了装置的实施例。

[198] 与上述方法实施例相对应, 本说明书还提供了一种区块链数据存储装置的实施例。

[199] 本说明书的区块链数据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。

[200] 从硬件层面而言, 如图 10 所示, 为本说明书的区块链数据存储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。

[201] 图 11 是本说明书一示例性实施例示出的一种区块链数据存储装置的框图。

[202] 请参考图 11, 所述区块链数据存储装置 110 可以应用在前述图 10 所示的电子设备中, 所述装置 110 包括: 获取模块 1101, 获取待存储的区块链数据的 key-value 键值对; 转换模块 1102, 将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点; 所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置, 每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位; 所述槽位用于存储与该节点链接的下一层节点的 hash 值; 存储模块 1103, 将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中; 所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中, value 为节点的存储内容, key 为节点的存储内容的 hash 值。

[203] 在本实施例中, 所述根节点、中间节点的层数为固定值。

[204] 在本实施例中, 所述根节点、中间节点用于存储所述区块链数据的 key 中的至少部分字符; 所述根节点、中间节点存储的所述至少部分字符, 为所述根节点、中间节点中的所述位置所代表的字符, 与所述位置中填充了 hash 值的槽位所代表的字符, 进行拼接生成的字符串。

[205] 在本实施例中, 所述叶子节点为 bucket 数据桶; 所述 bucket 数据桶包含若干条数据记录; 所述数据记录中存储的数据内容包括所述区块链数据的 value。

[206] 在本实施例中, 所述区块链数据的 key 对应的字符串包括字符前缀和字符后缀; 所述根节点、中间节点包括多个用于存放所述字符前缀中的字符的位置, 每个位置中包括多个用于存放所述字符前缀中的字符的槽位; 所述数据记录中存储的数据内容包括所述字符后缀和所述区块链数据的 value。

[207] 在本实施例中, 所述根节点、中间节点包括的所述位置的数量, 与所述位置包括的槽位的数量相同。

[208] 在本实施例中, 所述根节点、中间节点包括分别代表不同的 16 进制字符的 16 个位置; 所述位置包括 16 个分别代表不同的 16 进制字符的槽位。

[209] 在本实施例中, 所述区块链数据包括以下示出的任一: 区块中收录的交易; 与所

述区块中收录的交易对应的交易收据；与所述区块链中的区块链账户对应的最新账户状态数据；智能合约账户的存储内容。

[210] 在本实施例中，所述逻辑的树形结构包括以下示出的任一：用于存储区块中收录的交易的交易树；用于存储与区块中收录的交易对应的交易收据的收据树；用于存储与所述区块链中的区块链账户对应的最新账户状态数据的状态树；用于存储智能合约账户的存储内容的存储树。

[211] 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。

[212] 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。

[213] 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。

[214] 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。

[215] 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。

[216] 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。

[217] 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。

[218] 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”。

[219] 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。

权利要求书

1. 一种区块链中存储数据的方法, 包括:
获取待存储的区块链数据的 key-value 键值对;
将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点; 所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置, 每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位; 所述槽位用于存储与该节点链接的下一层节点的 hash 值;
将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中; 所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中, value 为节点的存储内容, key 为节点的存储内容的 hash 值。
2. 根据权利要求 1 所述的方法, 所述根节点、中间节点的层数为固定值。
3. 根据权利要求 1 所述的方法, 所述根节点、中间节点用于存储所述区块链数据的 key 中的至少部分字符; 所述根节点、中间节点存储的所述至少部分字符, 为所述根节点、中间节点中的所述位置所代表的字符, 与所述位置中填充了 hash 值的槽位所代表的字符, 进行拼接生成的字符串。
4. 根据权利要求 3 所述的方法, 所述叶子节点为 bucket 数据桶; 所述 bucket 数据桶包含若干条数据记录; 所述数据记录中存储的数据内容包括所述区块链数据的 value。
5. 根据权利要求 4 所述的方法, 所述区块链数据的 key 对应的字符串包括字符前缀和字符后缀; 所述根节点、中间节点包括多个用于存放所述字符前缀中的字符的位置, 每个位置中包括多个用于存放所述字符前缀中的字符的槽位; 所述数据记录中存储的数据内容包括所述字符后缀和所述区块链数据的 value。
6. 根据权利要求 1 所述的方法, 所述根节点、中间节点包括的所述位置的数量, 与所述位置包括的槽位的数量相同。
7. 根据权利要求 6 所述的方法, 所述根节点、中间节点包括分别代表不同的 16 进制字符的 16 个位置; 所述位置包括 16 个分别代表不同的 16 进制字符的槽位。
8. 根据权利要求 1 所述的方法, 所述区块链数据包括以下示出的任一:
区块中收录的交易;
与所述区块中收录的交易对应的交易收据;
与所述区块链中的区块链账户对应的最新账户状态数据;
智能合约账户的存储内容。
9. 根据权利要求 8 所述的方法, 所述逻辑的树形结构包括以下示出的任一:
用于存储区块中收录的交易的交易树;
用于存储与区块中收录的交易对应的交易收据的收据树;
用于存储与所述区块链中的区块链账户对应的最新账户状态数据的状态树;
用于存储智能合约账户的存储内容的存储树。
10. 一种区块链中存储数据的装置, 包括:
获取模块, 获取待存储的区块链数据的 key-value 键值对;
转换模块, 将所述待存储的区块链数据的 key-value 键值对转换成逻辑的树形结构上的根节点、中间节点和叶子节点; 所述根节点、中间节点包括多个用于存放所述区块链数据的 key 中的字符的位置, 每个位置中包括多个用于存放所述区块链数据的 key 中的字符的槽位; 所述槽位用于存储与该节点链接的下一层节点的 hash 值;
存储模块, 将所述根节点、中间节点和叶子节点的 key-value 键值对存储在数据库中; 所述叶子节点、所述中间节点和所述根节点的 key-value 键值对中, value 为节点的存储内容, key 为节点的存储内容的 hash 值。
11. 根据权利要求 10 所述的装置, 所述根节点、中间节点的层数为固定值。
12. 根据权利要求 10 所述的装置, 所述根节点、中间节点用于存储所述区块链数据的 key 中的至少部分字符; 所述根节点、中间节点存储的所述至少部分字符, 为所述根节点、中间节点中的所述位置所代表的字符, 与所述位置中填充了 hash 值的槽位所代表的字符, 进行拼接生成的字符串。
13. 根据权利要求 12 所述的装置, 所述叶子节点为 bucket 数据桶; 所述 bucket 数据桶包含若干条数据记录; 所述数据记录中存储的数据内容包括所述区块链数据的 value。
14. 根据权利要求 13 所述的装置, 所述区块链数据的 key 对应的字符串包括字符前缀和字符后缀; 所述根节点、中间节点包括多个用于存放所述字符前缀中的字符的位置, 每个位置中包括多个用于存放所述字符前缀中的字符的槽位; 所述数据记录中存储的数据内容包括所述字符后缀和所述区块链数据的 value。

15. 根据权利要求 10 所述的装置, 所述根节点、中间节点包括的所述位置的数量, 与所述位置包括的槽位的数量相同。

16. 根据权利要求 15 所述的装置, 所述根节点、中间节点包括分别代表不同的 16 进制字符的 16 个位置; 所述位置包括 16 个分别代表不同的 16 进制字符的槽位。

17. 根据权利要求 10 所述的装置, 所述区块链数据包括以下示出的任一:

区块中收录的交易;

与所述区块中收录的交易对应的交易收据;

与所述区块链中的区块链账户对应的最新账户状态数据;

智能合约账户的存储内容。

18. 根据权利要求 17 所述的装置, 所述逻辑的树形结构包括以下示出的任一:

用于存储区块中收录的交易的交易树;

用于存储与区块中收录的交易对应的交易收据的收据树;

用于存储与所述区块链中的区块链账户对应的最新账户状态数据的状态树;

用于存储智能合约账户的存储内容的存储树。

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, 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。

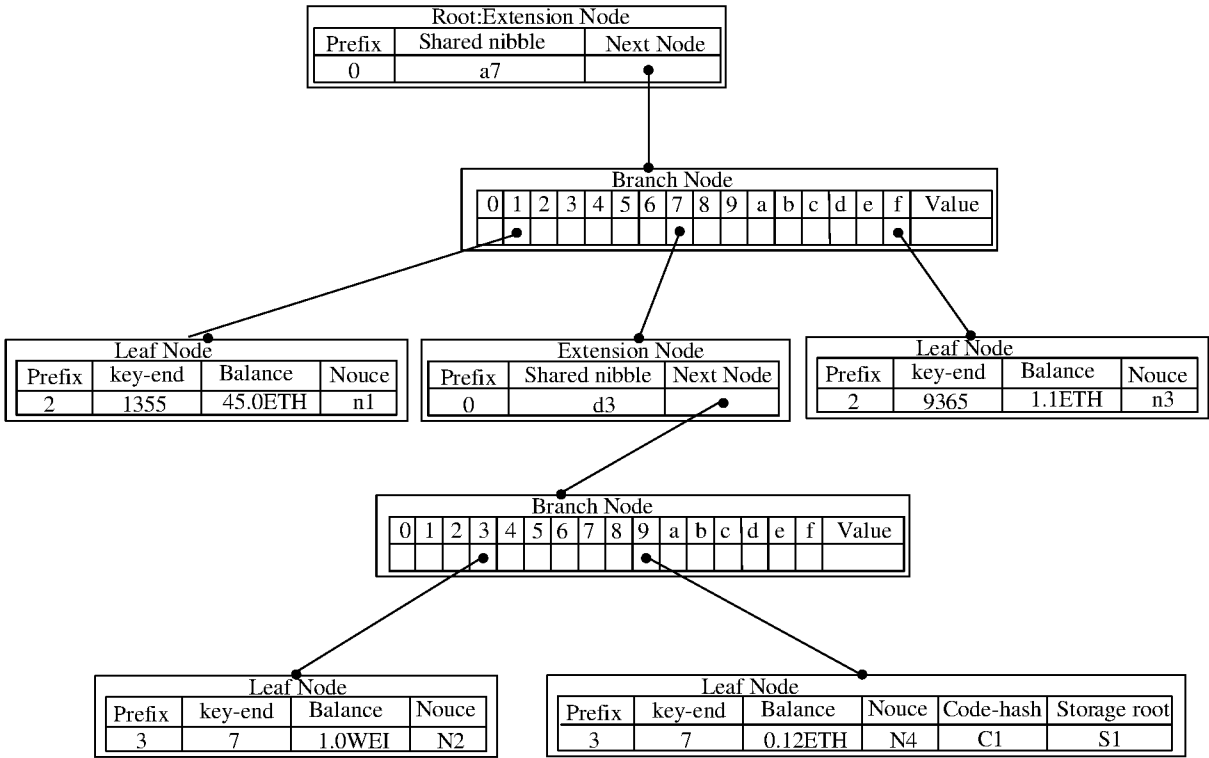


图 1

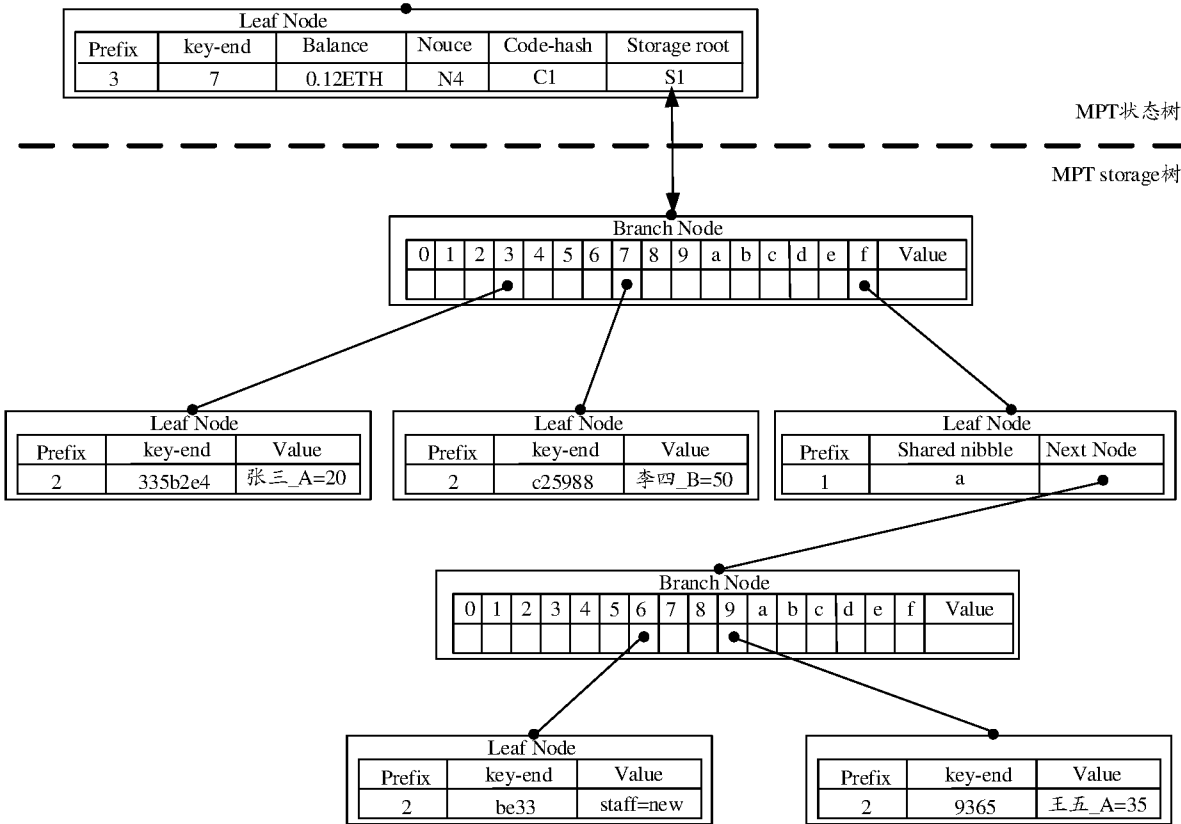


图 2

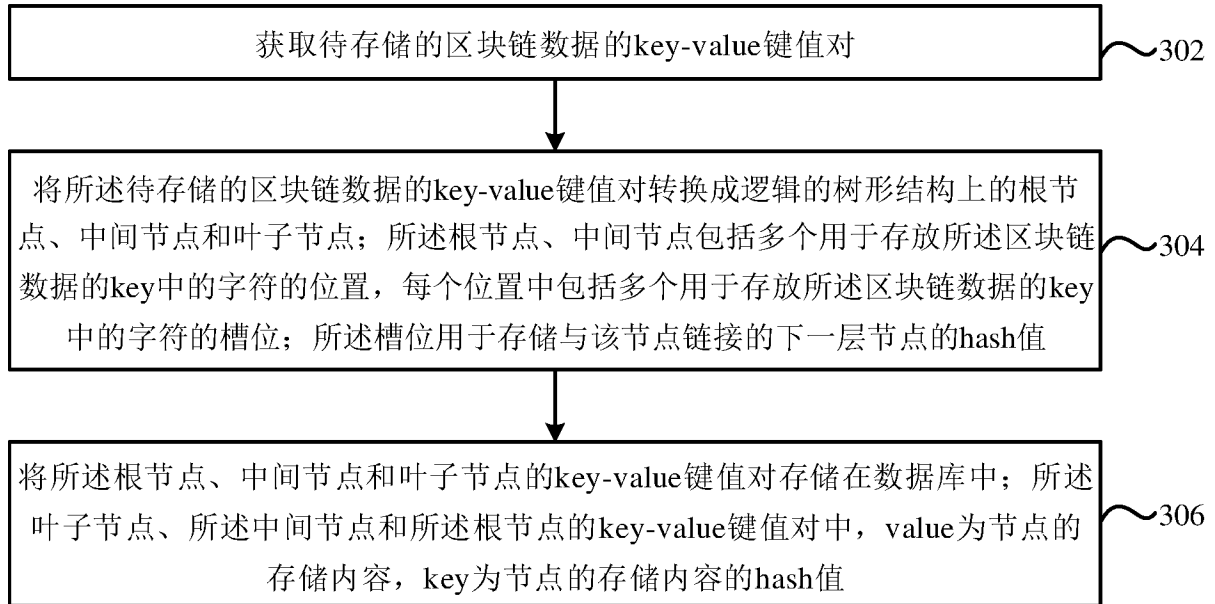


图 3

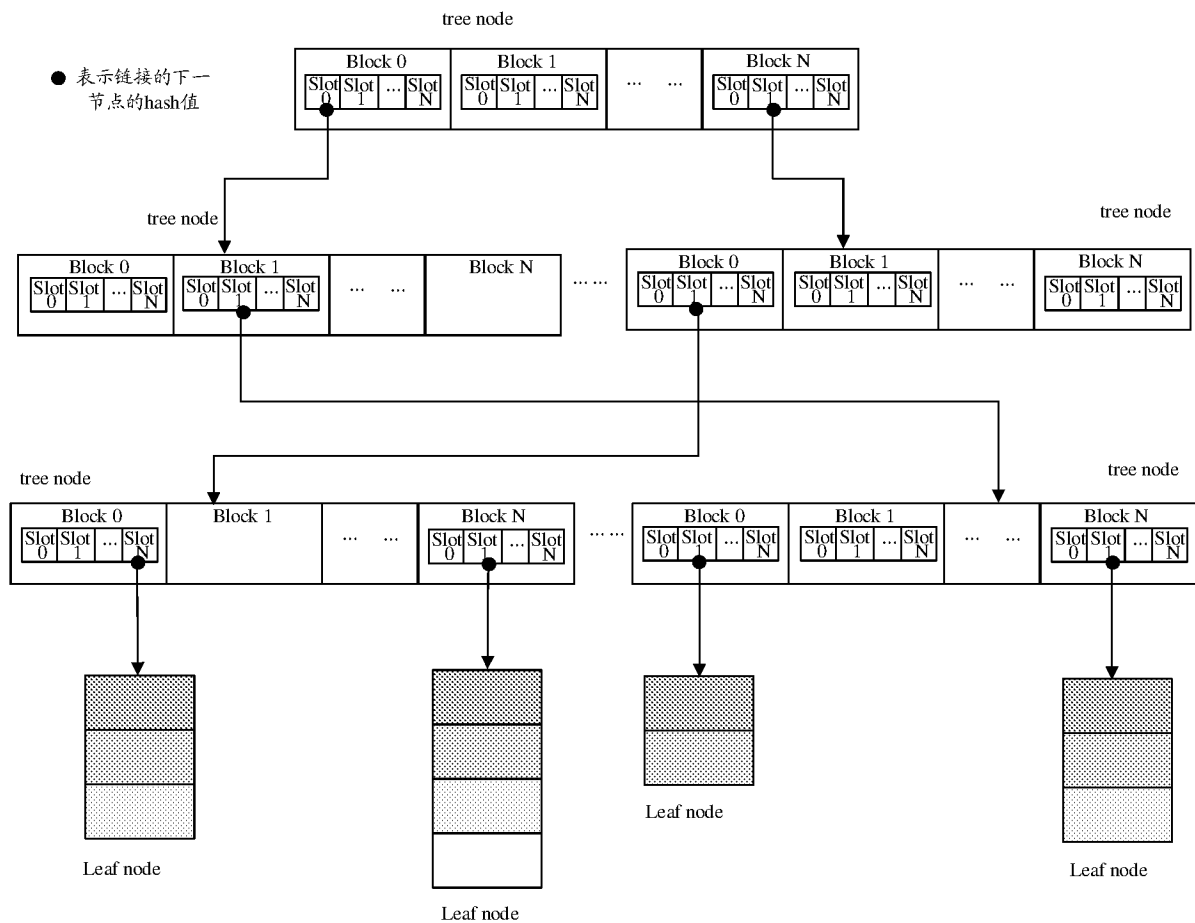


图 4

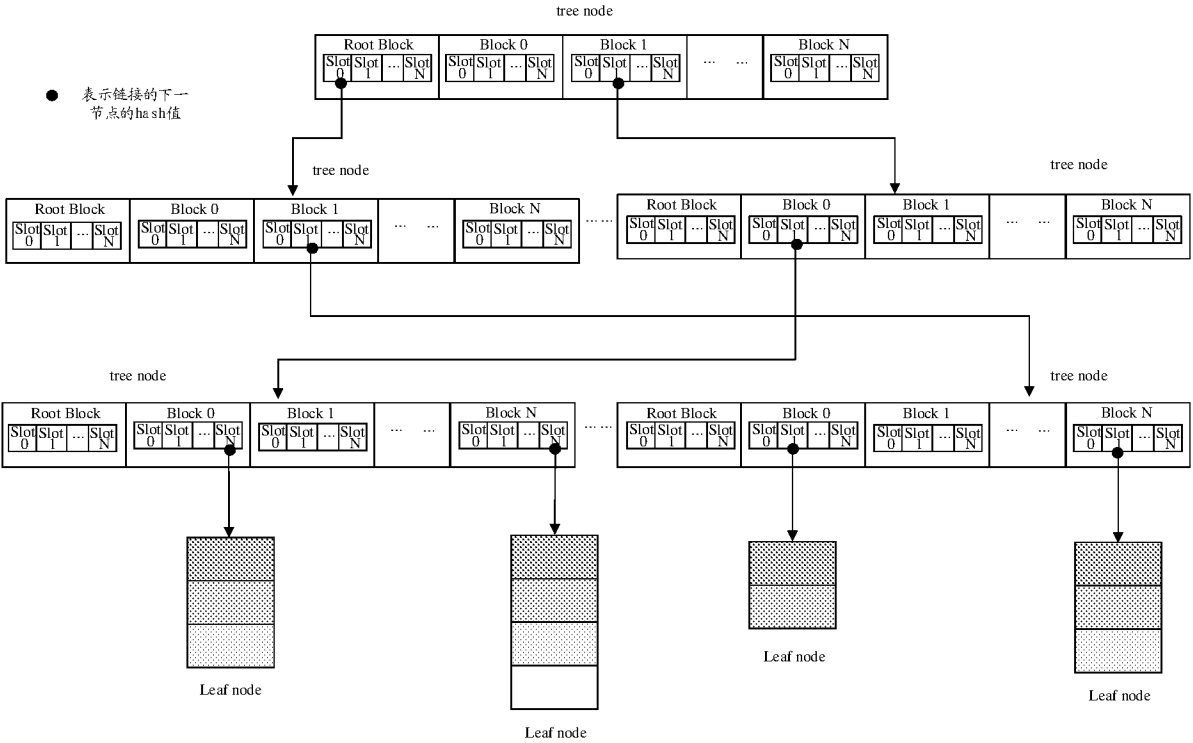


图 5

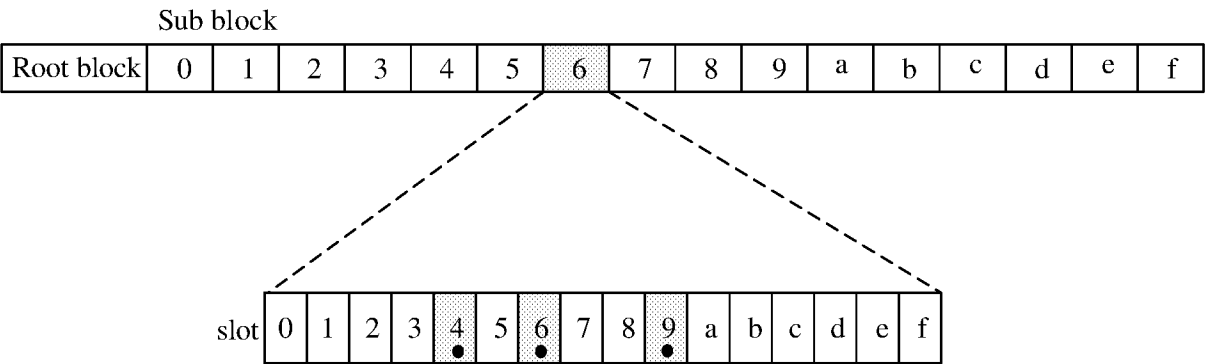


图 6

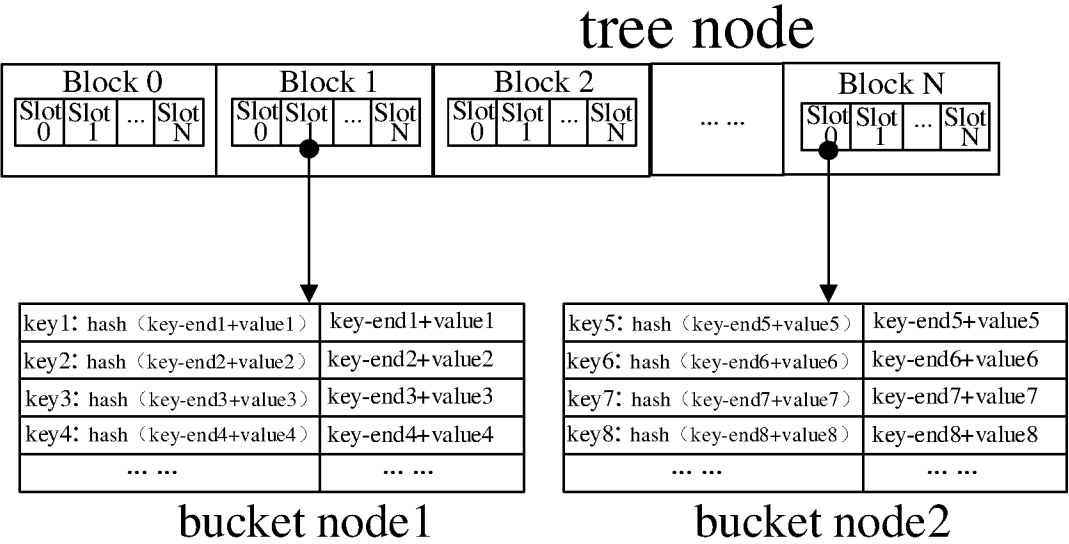


图 7

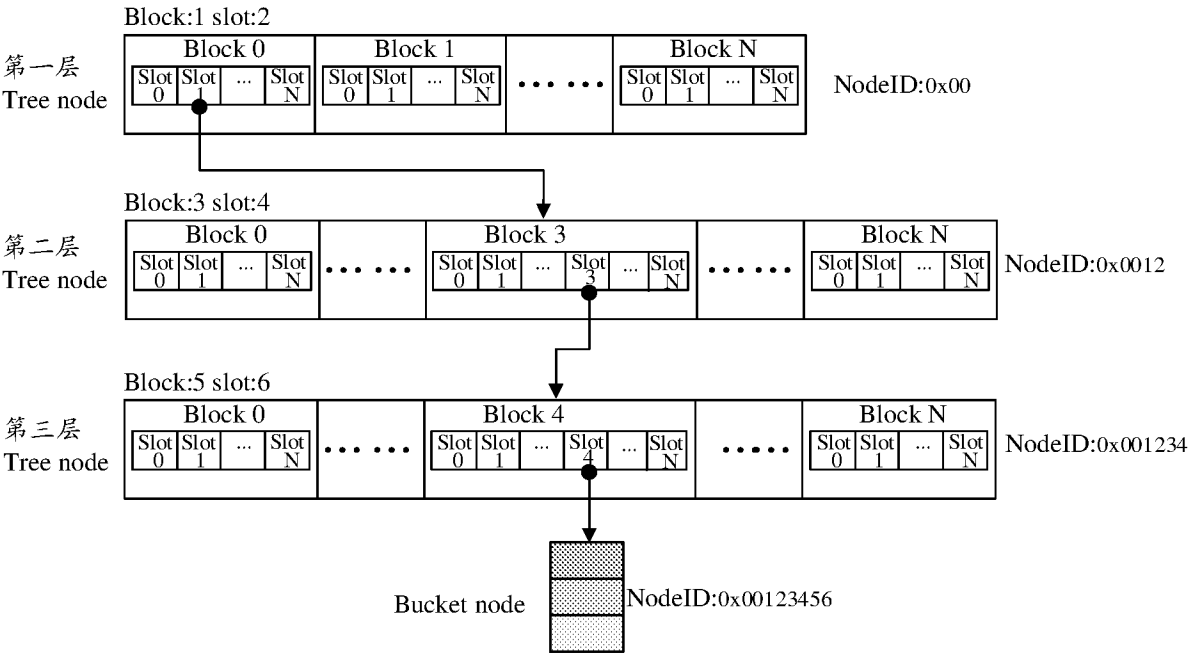


图 8

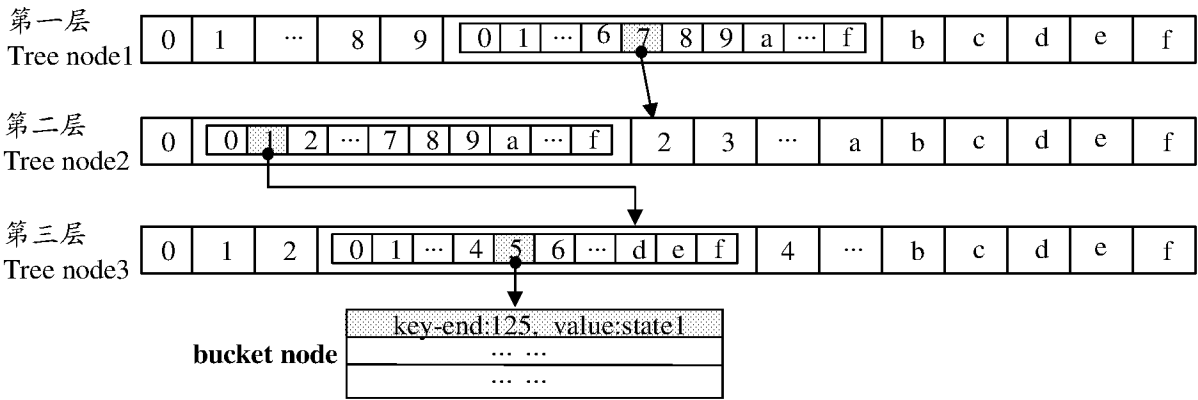


图 9

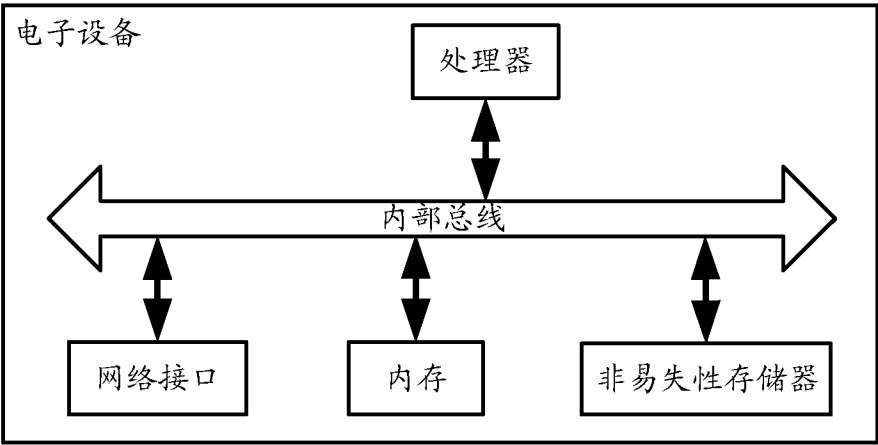


图 10

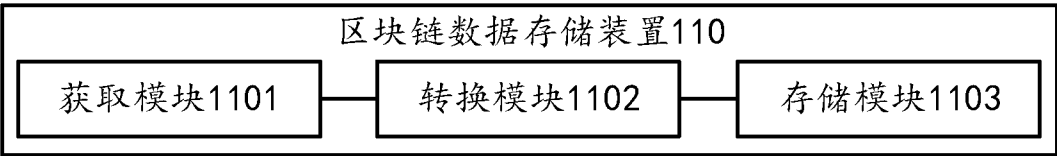


图 11

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2022/090294

A. CLASSIFICATION OF SUBJECT MATTER

G06F 16/27(2019.01)i; G06Q 40/04(2012.01)i; G06F 16/22(2019.01)n; G06F 16/23(2019.01)n

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F; G06Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, CNKI, WPI, EPODOC, IEEE: 区块链, 键值对, 槽位, 多个, N个, 索引, 哈希, 链接, 指针, 节点, 根节点, 叶子节点, 分支, 数据桶, 前缀, 后缀, 字符串, 拼接, 交易树, 收据树, 状态树, 存储树, 阿里巴巴, 支付宝, key, value, block chain, slot, hash, node, root, branch, leaf, extension, FDMT, MPT, Merkle, tree, Trie, bucket

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 112988910 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) claims 1-20	1-20
PX	CN 113220685 A (ALIPAY (HANGZHOU) INFORMATION TECHNOLOGY CO., LTD.) 06 August 2021 (2021-08-06) description, paragraphs 0052-0236	1-20
PX	CN 112988908 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) description, paragraphs 0029-0224	1-20
PX	CN 114153848 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 08 March 2022 (2022-03-08) description, paragraphs 0053-0325	1-20
PX	CN 112988761 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) description, paragraphs 0030-0173	1-20



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

08 July 2022

Date of mailing of the international search report

27 July 2022

Name and mailing address of the ISA/CN

China National Intellectual Property Administration (ISA/
CN)
No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing
100088, China

Facsimile No. (86-10)62019451

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2022/090294**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 112988909 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) description, paragraphs 0030-0223	1-20
PX	CN 112988911 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) description, paragraphs 0029-0222	1-20
PX	CN 112905607 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 04 June 2021 (2021-06-04) description, paragraphs 0030-0184	1-20
PX	CN 112988912 A (ALIPAY (HANGZHOU) TECHNOLOGY CO., LTD. et al.) 18 June 2021 (2021-06-18) description, paragraphs 0032-0224	1-20
A	CN 110334154 A (ALIBABA GROUP HOLDING LIMITED) 15 October 2019 (2019-10-15) description, paragraphs 0045-0126, and figures 1-5	1-20
A	CN 111837115 A (ADVANCED NEW TECHNOLOGIES CO., LTD.) 27 October 2020 (2020-10-27) entire document	1-20
A	CN 111488614 A (BEIJING RUICE TECHNOLOGY CO., LTD.) 04 August 2020 (2020-08-04) entire document	1-20
A	US 2016335299 A1 (APPLE INC.) 17 November 2016 (2016-11-17) entire document	1-20

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2022/090294

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	112988910	A	18 June 2021	None			
CN	113220685	A	06 August 2021	None			
CN	112988908	A	18 June 2021	None			
CN	114153848	A	08 March 2022	None			
CN	112988761	A	18 June 2021	None			
CN	112988909	A	18 June 2021	None			
CN	112988911	A	18 June 2021	None			
CN	112905607	A	04 June 2021	None			
CN	112988912	A	18 June 2021	None			
CN	110334154	A	15 October 2019	TW	202101249	A	01 January 2021
				WO	2020258855	A1	30 December 2020
CN	111837115	A	27 October 2020	None			
CN	111488614	A	04 August 2020	None			
US	2016335299	A1	17 November 2016	None			

A. 主题的分类 G06F 16/27(2019.01)i; G06Q 40/04(2012.01)i; G06F 16/22(2019.01)n; G06F 16/23(2019.01)n 按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类		
B. 检索领域 检索的最低限度文献(标明分类系统和分类号) G06F; G06Q 包含在检索领域中的除最低限度文献以外的检索文献 在国际检索时查阅的电子数据库(数据库的名称, 和使用的检索词(如使用)) CNPAT, CNKI, WPI, EP0DOC, IEEE: 区块链, 键值对, 槽位, 多个, N个, 索引, 哈希, 链接, 指针, 节点, 根节点, 叶子节点, 分支, 数据桶, 前缀, 后缀, 字符串, 拼接, 交易树, 收据树, 状态树, 存储树, 阿里巴巴, 支付宝, key, value, block chain, slot, hash, node, root, branch, leaf, extension, FDMT, MPT, Merkle, tree, Trie, bucket		
C. 相关文件		
类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
PX	CN 112988910 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 权利要求1-20	1-20
PX	CN 113220685 A (支付宝杭州信息技术有限公司) 2021年8月6日 (2021 - 08 - 06) 说明书0052-0236段	1-20
PX	CN 112988908 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 说明书0029-0224段	1-20
PX	CN 114153848 A (支付宝杭州信息技术有限公司 等) 2022年3月8日 (2022 - 03 - 08) 说明书0053-0325段	1-20
PX	CN 112988761 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 说明书0030-0173段	1-20
PX	CN 112988909 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 说明书0030-0223段	1-20
PX	CN 112988911 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 说明书0029-0222段	1-20
☒ 其余文件在C栏的续页中列出。 ☒ 见同族专利附件。		
* 引用文件的具体类型: “A” 认为不特别相关的表示了现有技术一般状态的文件 “E” 在国际申请日的当天或之后公布的在先申请或专利 “L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件(如具体说明的) “O” 涉及口头公开、使用、展览或其他方式公开的文件 “P” 公布日先于国际申请日但迟于所要求的优先权日的文件 “T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件 “X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性 “Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性 “&” 同族专利的文件		
国际检索实际完成的日期 2022年7月8日		国际检索报告邮寄日期 2022年7月27日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451		受权官员 宿渊源 电话号码 010-53961376

C. 相关文件

类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
PX	CN 112905607 A (支付宝杭州信息技术有限公司 等) 2021年6月4日 (2021 - 06 - 04) 说明书0030-0184段	1-20
PX	CN 112988912 A (支付宝杭州信息技术有限公司 等) 2021年6月18日 (2021 - 06 - 18) 说明书0032-0224段	1-20
A	CN 110334154 A (阿里巴巴集团控股有限公司) 2019年10月15日 (2019 - 10 - 15) 说明书0045-0126段, 附图1-5	1-20
A	CN 111837115 A (创新先进技术有限公司) 2020年10月27日 (2020 - 10 - 27) 全文	1-20
A	CN 111488614 A (北京瑞策科技有限公司) 2020年8月4日 (2020 - 08 - 04) 全文	1-20
A	US 2016335299 A1 (APPLE INC.) 2016年11月17日 (2016 - 11 - 17) 全文	1-20

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2022/090294

检索报告引用的专利文件			公布日 (年/月/日)	同族专利	公布日 (年/月/日)
CN	112988910	A	2021年6月18日	无	
CN	113220685	A	2021年8月6日	无	
CN	112988908	A	2021年6月18日	无	
CN	114153848	A	2022年3月8日	无	
CN	112988761	A	2021年6月18日	无	
CN	112988909	A	2021年6月18日	无	
CN	112988911	A	2021年6月18日	无	
CN	112905607	A	2021年6月4日	无	
CN	112988912	A	2021年6月18日	无	
CN	110334154	A	2019年10月15日	TW 202101249 A 2021年1月1日 WO 2020258855 A1 2020年12月30日	
CN	111837115	A	2020年10月27日	无	
CN	111488614	A	2020年8月4日	无	
US	2016335299	A1	2016年11月17日	无	