(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 110471795 B
(45)授权公告日 2020.10.02

(21)申请号 201910703814.4

(22)申请日 2019.07.31

(65)同一申请的已公布的文献号
申请公布号 CN 110471795 A

(43)申请公布日 2019.11.19

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

(72)发明人 卓海振 俞本权 陆钟豪

(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 11/14(2006.01)
G06F 16/22(2019.01)
G06F 16/27(2019.01)

(56)对比文件
CN 108923932 A,2018.11.30
CN 109919756 A,2019.06.21
CN 108197226 A,2018.06.22
US 2019228386 A1,2019.07.25
CN 108282474 A,2018.07.13
US 2019103973 A1,2019.04.04
覃应接.《网络日志管理与分析技术研究与实现》.《中国优秀硕士学位论文全文数据库(电子期刊)信息科技辑》.2015,第2015年卷(第10期),第I139-39页.

审查员 吴朝烨

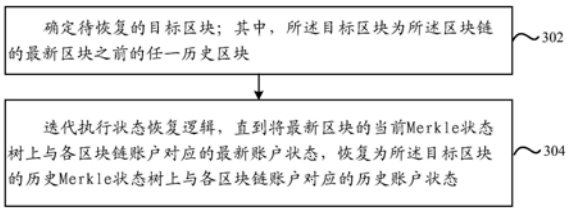
权利要求书3页 说明书21页 附图5页

(54)发明名称

区块链状态数据恢复方法及装置、电子设备

(57)摘要

一种区块链状态数据恢复方法,区块链中的账户状态数据被组织成Merkle状态树在数据库中存储;Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树;以及,由各区块链账户的历史账户状态组织成的历史Merkle状态树;包括:确定待恢复的目标区块;其中,所述目标区块为所述区块链的最新区块之前的任一历史区块;迭代执行状态恢复逻辑,直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态,恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态。



CN 110471795 B

1.一种区块链状态数据恢复方法,所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储;所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树;以及,由各区块链账户的历史账户状态组织成的历史Merkle状态树;所述方法包括:

确定待恢复的目标区块;其中,所述目标区块为所述区块链的最新区块之前的任一历史区块;

迭代执行状态恢复逻辑,直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态,恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态;

所述状态恢复逻辑包括:

确定与所述最新区块中的交易相关的目标账户,并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态;

将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态,修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态,并在修改完成后将所述上一区块重新确定为所述最新区块。

2.根据权利要求1所述的方法,所述确定与所述最新区块中的交易相关的目标账户,包括:

重新执行所述最新区块中的交易,以确定所述最新区块中的交易相关的目标账户;或者,查询与所述最新区块对应的读写集,以确定所述最新区块中的交易相关的目标账户;

所述查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态,包括:

在所述最新区块的上一区块的历史Merkle状态树上查询与所述目标账户对应的历史账户状态;或者,在与所述最新区块的上一区块对应的读写集中,查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态。

3.根据权利要求2所述的方法,所述迭代执行状态恢复逻辑之前,还包括:

确定从所述区块链的创世块至所述目标区块所包含的交易的总数量,是否大于从所述目标区块至所述最新区块所包含的交易的总数量;如果是,进一步迭代执行所述状态恢复逻辑。

4.根据权利要求1所述的方法,还包括:

在所述区块链中的任一区块中的交易执行完毕后,基于与所述区块中的交易相关的目标账户的最新账户状态,生成与所述区块的当前Merkle状态树对应的更新数据节点和与所述区块的历史Merkle状态树对应的历史数据节点;

基于生成的所述更新数据节点对所述区块的上一区块的当前Merkle状态树上与所述目标账户对应的数据节点进行修改更新,以得到所述区块的当前Merkle状态树;以及,

基于生成的所述历史数据节点和复用的所述区块的上一区块的历史Merkle状态树上与所述目标账户对应的数据节点以外的其它数据节点,为所述区块创建历史Merkle状态树。

5.根据权利要求4所述的方法,所述当前Merkle状态树上的数据节点被组织成B+树的数据结构在数据库中存储;所述历史Merkle状态树上的数据节点被组织成LSM树的数据结

构在数据库中存储。

6.根据权利要求1所述的方法,所述数据库为Key-Value数据库;

所述Merkle状态树上的数据节点以Key-Value键值对的形式存储在所述数据库中;其中,所述当前Merkle状态树上的数据节点的key为所述数据节点的节点ID;所述历史Merkle状态树上的数据节点的key为所述数据节点包含的数据内容的hash值。

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

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

9.根据权利要求1所述的方法,所述Merkle状态树为融合了Trie字典树的树形结构的Merkle状态树变种。

10.根据权利要求9所述的方法,所述 Merkle状态树为Merkle Patricia Tree状态树。

11.一种区块链状态数据恢复装置,所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储;所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树;以及,由各区块链账户的历史账户状态组织成的历史Merkle状态树;所述装置包括:

确定模块,确定待恢复的目标区块;其中,所述目标区块为所述区块链的最新区块之前的任一历史区块;

恢复模块,迭代执行状态恢复逻辑,直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态,恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态;

所述状态恢复逻辑包括:

确定与所述最新区块中的交易相关的目标账户,并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态;

将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态,修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态,并在修改完成后将所述上一区块重新确定为所述最新区块。

12.根据权利要求11所述的装置,所述确定模块:

重新执行所述最新区块中的交易,以确定所述最新区块中的交易相关的目标账户;或者,查询与所述最新区块对应的读写集,以确定所述最新区块中的交易相关的目标账户;

所述恢复模块:

在所述最新区块的上一区块的历史Merkle状态树上查询与所述目标账户对应的历史账户状态;或者,在与所述最新区块的上一区块对应的读写集中,查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态。

13.根据权利要求12所述的装置,所述恢复模块进一步:

在迭代执行状态恢复逻辑之前,确定从所述区块链的创世块至所述目标区块所包含的交易的总数量,是否大于从所述目标区块至所述最新区块所包含的交易的总数量;如果是,进一步迭代执行所述状态恢复逻辑。

14.根据权利要求11所述的装置,还包括:

生成模块,在所述区块链中的任一区块中的交易执行完毕后,基于与所述区块中的交

易相关的目标账户的最新账户状态,生成与所述区块的当前Merkle状态树对应的更新数据节点和与所述区块的历史Merkle状态树对应的历史数据节点;

修改模块,基于生成的所述更新数据节点对所述区块的上一区块的当前Merkle状态树上与所述目标账户对应的数据节点进行修改更新,以得到所述区块的当前Merkle状态树;

创建模块,基于生成的所述历史数据节点和复用的所述区块的上一区块的历史Merkle状态树上与所述目标账户对应的数据节点以外的其它数据节点,为所述区块创建历史Merkle状态树。

15.根据权利要求14所述的装置,所述当前Merkle状态树上的数据节点被组织成B+树的数据结构在数据库中存储;所述历史Merkle状态树上的数据节点被组织成LSM树的数据结构在数据库中存储。

16.根据权利要求11所述的装置,所述数据库为Key-Value数据库;

所述Merkle状态树上的数据节点以Key-Value键值对的形式存储在所述数据库中;其中,所述当前Merkle状态树上的数据节点的key为所述数据节点的节点ID;所述历史Merkle状态树上的数据节点的key为所述数据节点包含的数据内容的hash值。

17.根据权利要求11所述的装置,所述数据库为LevelDB数据库;或者基于LevelDB架构的数据库。

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

19.根据权利要求11所述的装置,所述Merkle状态树为融合了Trie字典树的树形结构的Merkle状态树变种。

20.根据权利要求19所述的装置,所述 Merkle状态树为Merkle Patricia Tree状态树。

21.一种电子设备,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1-10中任一项所述的方法。

22.一种计算机可读存储介质,其上存储有计算机指令,其特征在于,该指令被处理器执行时实现如权利要求1-10中任一项所述方法的步骤。

区块链状态数据恢复方法及装置、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，尤其涉及一种区块链状态数据恢复方法及装置、电子设备。

背景技术

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

发明内容

[0003] 本说明书提出一种区块链状态数据恢复方法，所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储；所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树；以及，由各区块链账户的历史账户状态组织成的历史Merkle状态树；所述方法包括：

[0004] 确定待恢复的目标区块；其中，所述目标区块为所述区块链的最新区块之前的任一历史区块；

[0005] 迭代执行状态恢复逻辑，直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态；

[0006] 所述状态恢复逻辑包括：

[0007] 确定与所述最新区块中的交易相关的目标账户，并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态；

[0008] 将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态，修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态，并在修改完成后将所述上一区块重新确定为所述最新区块。

[0009] 可选的，所述确定与所述最新区块中的交易相关的目标账户，包括：

[0010] 重新执行所述最新区块中的交易，以确定所述最新区块中的交易相关的目标账户；或者，查询与所述最新区块对应的读写集，以确定所述最新区块中的交易相关的目标账户。

[0011] 所述查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态，包括：

[0012] 在所述最新区块的上一区块的历史Merkle状态树上查询与所述目标账户对应的历史账户状态；或者，在与所述最新区块的上一区块对应的读写集中，查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态。

[0013] 可选的，所述迭代执行状态恢复逻辑之前，还包括：

[0014] 确定从所述区块链的创世块至所述目标区块所包含的交易的总数量，是否大于从所述目标区块至所述最新区块所包含的交易的总数量；如果是，进一步迭代执行所述状态恢复逻辑。

[0015] 可选的，还包括：

[0016] 在所述区块链中的任一区块中的交易执行完毕后，基于与所述区块中的交易相关的目标账户的最新账户状态，生成与所述区块的当前Merkle状态树对应的更新数据节点和与所述区块的历史Merkle状态树对应的历史数据节点；

[0017] 基于生成的所述更新数据节点对所述区块的上一区块的当前Merkle状态树上与所述目标账户对应的数据节点进行修改更新，以得到所述区块的当前Merkle状态树；以及，

[0018] 基于生成的所述历史数据节点和复用的所述区块的上一区块的历史Merkle状态树上与所述目标账户对应的数据节点以外的其它数据节点，为所述区块创建历史Merkle状态树。

[0019] 可选的，所述当前Merkle状态树上的数据节点被组织成B+树的数据结构在数据库中存储；所述历史Merkle状态树上的数据节点被组织成LSM树的数据结构在数据库中存储。

[0020] 可选的，所述数据库为Key-Value数据库；

[0021] 所述Merkle状态树上的数据节点以Key-Value键值对的形式存储在所述数据库中；其中，所述当前Merkle状态树上的数据节点的key为所述数据节点的节点ID；所述历史Merkle状态树上的数据节点的key为所述数据节点包含的数据内容的hash值。

[0022] 可选的，所述数据库为LevelDB数据库；或者基于LevelDB架构的数据库。

[0023] 可选的，所述数据库为基于LevelDB架构的Rocksdb数据库。

[0024] 可选的，所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种。

[0025] 可选的，所述Merkle状态树为Merkle Patricia Tree状态树。

[0026] 本说明书还提出一种区块链状态数据恢复装置，所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储；所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树；以及，由各区块链账户的历史账户状态组织成的历史Merkle状态树；所述装置包括：

[0027] 确定模块，确定待恢复的目标区块；其中，所述目标区块为所述区块链的最新区块之前的任一历史区块；

[0028] 恢复模块，迭代执行状态恢复逻辑，直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态；

[0029] 所述状态恢复逻辑包括：

[0030] 确定与所述最新区块中的交易相关的目标账户，并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态；

[0031] 将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态，修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态，并在修改完成后将所述上一区块重新确定为所述最新区块。

[0032] 可选的，所述确定模块：

[0033] 重新执行所述最新区块中的交易，以确定所述最新区块中的交易相关的目标账

户;或者,查询与所述最新区块对应的读写集,以确定所述最新区块中的交易相关的目标账户。

[0034] 所述恢复模块:

[0035] 在所述最新区块的上一区块的历史Merkle状态树上查询与所述目标账户对应的历史账户状态;或者,在与所述最新区块的上一区块对应的读写集中,查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态。

[0036] 可选的,所述恢复模块进一步:

[0037] 在迭代执行状态恢复逻辑之前,确定从所述区块链的创世块至所述目标区块所包含的交易的总数量,是否大于从所述目标区块至所述最新区块所包含的交易的总数量;如果是,进一步迭代执行所述状态恢复逻辑。

[0038] 可选的,还包括:

[0039] 生成模块,在所述区块链中的任一区块中的交易执行完毕后,基于与所述区块中的交易相关的目标账户的最新账户状态,生成与所述区块的当前Merkle状态树对应的更新数据节点和与所述区块的历史Merkle状态树对应的历史数据节点;

[0040] 修改模块,基于生成的所述更新数据节点对所述区块的上一区块的当前Merkle状态树上与所述目标账户对应的数据节点进行修改更新,以得到所述区块的当前Merkle状态树;

[0041] 创建模块,基于生成的所述历史数据节点和复用的所述区块的上一区块的历史Merkle状态树上与所述目标账户对应的数据节点以外的其它数据节点,为所述区块创建历史Merkle状态树。

[0042] 可选的,所述当前Merkle状态树上的数据节点被组织成B+树的数据结构在数据库中存储;所述历史Merkle状态树上的数据节点被组织成LSM树的数据结构在数据库中存储。

[0043] 可选的,所述数据库为Key-Value数据库;

[0044] 所述Merkle状态树上的数据节点以Key-Value键值对的形式存储在所述数据库中;其中,所述当前Merkle状态树上的数据节点的key为所述数据节点的节点ID;所述历史Merkle状态树上的数据节点的key为所述数据节点包含的数据内容的hash值。

[0045] 可选的,所述数据库为LevelDB数据库;或者基于LevelDB架构的数据库。

[0046] 可选的,所述数据库为基于LevelDB架构的Rocksdb数据库。

[0047] 可选的,所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种。

[0048] 可选的,所述Merkle状态树为Merkle Patricia Tree状态树。

[0049] 以上技术方案中,可以在支持将账户状态数据组织成当前Merkle状态树和历史Merkle状态树的区块链模型中,将各区块链账户在最新区块的当前Merkle状态树上对应的最新账户状态,稳定高效的回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态。

附图说明

[0050] 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;

[0051] 图2是一示例性实施例提供的一种MPT状态树上的node复用的示意图;

[0052] 图3是一示例性实施例提供的一种区块链状态数据恢复方法的流程图；

[0053] 图4是一示例性实施例提供的一种对当前Merkle状态树进行状态回滚的示意图；

[0054] 图5是一示例性实施例提供的一种电子设备的结构示意图；

[0055] 图6是一示例性实施例提供的一种区块链状态恢复装置的框图。

具体实施方式

[0056] 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。

[0057] 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。

[0058] 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。

[0059] 其中，去中心化程度最高的是公有链。公有链以比特币、以太坊为代表，加入公有链的参与者(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。

[0060] 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。

[0061] 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。

[0062] 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。

[0063] 对于物理世界产生的真实数据，可以将其构建成区块链所支持的标准的交易(transaction)格式，然后发布至区块链，由区块链中的节点设备对收到的交易进行共识处理，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。

[0064] 其中，区块链中支持的共识算法可以包括：

[0065] 第一类共识算法，即节点设备需要争夺每一轮的记账周期的记账权的共识算法；例如，工作量证明(Proof of Work,POW)、股权证明(Proof of Stake,POS)、委任权益证明(Delegated Proof of Stake,DPOS)等共识算法；

[0066] 第二类共识算法，即预先为每一轮记账周期选举记账节点(不需要争夺记账权)的共识算法；例如，实用拜占庭容错(Practical Byzantine Fault Tolerance，PBFT)等共识算法。

[0067] 在采用第一类共识算法的区块链网络中，争夺记账权的节点设备，都可以在接收到交易后执行该笔交易。争夺记账权的节点设备中可能有一个节点设备在本轮争夺记账权的过程中胜出，成为记账节点。记账节点可以将收到的交易与其它交易一起打包以生成最新区块，并将生成的最新区块或者该最新区块的区块头发送至其它节点设备进行共识。

[0068] 在采用第二类共识算法的区块链网络中，具有记账权的节点设备在本轮记账前已经商定好。因此，节点设备在接收到交易后，如果自身不是本轮的记账节点，则可以将该交易发送至记账节点。对于本轮的记账节点，在将该交易与其它交易一起打包以生成最新区块的过程中或者之前，可以执行该交易。记账节点在生成最新区块后，可以将该最新区块或者该最新区块的区块头发送至其它节点设备进行共识。

[0069] 如上所述，无论区块链采用以上示出的哪种共识算法，本轮的记账节点都可以将接收到的交易打包以生成最新区块，并将生成的最新区块或者该最新区块的区块头发送至其它节点设备进行共识验证。如果其它节点设备接收到最新区块或者该最新区块的区块头后，经验证没有问题，可以将该最新区块追加到原有的区块链末尾，从而完成区块链的记账过程。其它节点验证记账节点发来的新的区块或区块头的过程中，也可以执行该区块中的包含的交易。

[0070] 在区块链领域，有一个重要的概念就是账户(Account)；以以太坊为例，以太坊通常将账户划分为外部账户和合约账户两类；外部账户就是由用户直接控制的账户，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)。

[0071] 当然，对于一些基于以太坊的架构而衍生出的区块链模型(比如蚂蚁区块链)，还可以对区块链支持的账户类型，进行进一步的扩展，在本说明书中不进行特别限定。

[0072] 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。

[0073] 以以太坊为例，账户的结构体通常包括Balance，Nonce，Code和Storage等字段。其中：

[0074] Balance字段，用于维护账户目前的账户余额；

[0075] 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；

[0076] 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。

[0077] Storage字段，用于维护该账户的存储内容(默认字段值为空)；对于合约账户而言，通常会分配一个独立的存储空间，用以存储该合约账户的存储内容；该独立的存储空间通常称之为该合约账户的账户存储。合约账户的存储内容通常会构建成MPT(Merkle Patricia Trie)树的数据结构存储在上述独立的存储空间之中；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；因此，Storage字段通常也称之为StorageRoot字段。

[0078] 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。

[0079] 对于大多数区块链模型，通常都会使用Merkle树；或者，基于Merkle树的数据结构，来存储和维护数据。以以太坊为例，以太坊使用了MPT树(一种Merkle树变种)，作为数据组织形式，用来组织和管理账户状态、交易信息等重要数据。

[0080] 以太坊针对区块链中需要存储和维护的数据，设计了三棵MPT树，分别是MPT状态树、MPT交易树和MPT收据树。其中，除了以上三棵MPT树以外，实际上还存在一棵基于合约账户的存储内容构建的Storage树。

[0081] MPT状态树，是由区块链中所有账户的账户状态(state)数据组织成的MPT树；MPT交易树，是由区块链中的交易(transaction)数据组织成的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)收据组织成的MPT树。以上示出的MPT状态树、MPT交易树和MPT收据树的根节点的hash值，最终都会被添加至对应区块的区块头中。

[0082] 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。

[0083] 对于组织成的MPT交易树、MPT收据树和MPT状态树，最终都会在采用多级数据存储结构的Key-Value型数据库(比如，LevelDB)中进行存储。

[0084] 而采用多级数据存储结构的上述数据库，通常采用多级数据存储的结构，可以被划分为n级数据存储；例如，各级数据存储可以依次设为L0，L1，L2，L3....L(n-1)；对于上述数据库中的各级数据存储而言，等级编号越小通常级别越高；例如，L0存储的是最新的若干区块的数据，L1存储的是次新的若干区块的数据，以此类推。

[0085] 其中，各级数据存储对应的存储介质的读写性能，通常也可以存在性能差异；例如，级别高(即等级编号较小的)的数据存储对应的存储介质的读写性能，可以高于级别低的数据存储对应的存储介质的读写性能。在实际应用中，级别高的数据存储，可以使用存储成本较高，存储性能较优的存储介质；而级别低的数据存储，可以使用单位成本低，且容量较大的存储介质。

[0086] 在实际应用中，随着区块链的区块号的增长(也称之为区块高度)，在数据库中存储的数据，会包含很多历史数据；而且，区块号越小的区块中的数据越久远，越不重要。因此，为了降低整体的存储成本，通常可以对不同区块高度的数据进行“区别对待”；例如，可以将区块号较小的区块中的数据，存储至成本较低的存储介质上；而将区块号较大的区块中的数据，存储在成本较高的存储介质上。

[0087] 需要说明的是，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；

[0088] 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。

[0089] 而节点设备在区块链产生的最新区块中的交易执行完毕后，由于当前区块链中的账户状态发生了变化，因此节点设备需要根据区块链中所有账户当前的账户状态数据，来

构建MPT状态树，用于维护区块链中所有账户的最新状态。

[0090] 也即，每当区块链中产生一个最新区块，并且该最新区块中的交易执行完毕后，导致区块链中的账户状态发生了变化，节点设备都需要基于区块链中所有账户最新的账户状态数据，重新构建一棵MPT状态树。

[0091] 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。

[0092] 请参见图1，图1为本说明书示出的一种将区块链的账户状态数据组织成MPT状态树的示意图。

[0093] MPT树，是一种经过改良的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。

[0094] 在MPT树中通常包括三种数据节点，分别为叶子节点(leaf node)，扩展节点(extension node)和分支节点(branch node)。

[0095] 叶子节点，是表示为[key,value]的一个键值对，其中key是种特殊的十六进制编码字符，value是该叶子节点对应的账户地址的状态数据(即以上示出的结构体)。扩展节点，也是表示为[key,value]的一个键值对，其中key也是种特殊的十六进制编码字符，但是value是其他节点的hash值(hash指针)，也就是说可以通过hash指针链接到其他节点。

[0096] 分支节点，包含17个元素，前16个元素对应着key中的16个可能的十六进制字符；一个字符对应一个nibble(半字节)。如果有一个[key,value]对在这个分支节点终止，则该分支节点可以充当叶子节点的角色，最后一个元素则代表叶子节点的value值；反之，分支节点的最后一个元素，可以为空值。

[0097] 由于在MPT树上，从根节点到一个叶子节点的搜索路径上的字符，组成一个完整的账户地址；因此，对于分支节点而言，其既可以是上述搜索路径的终止节点，也可以是上述搜索路径的中间节点。

[0098] 假设需要组织成MTP状态树的账户状态数据如下表1所示：

[0099]

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

[0100] 表1

[0101] 在表1中，账户地址是由若干16进制的字符构成的字符串。账户状态state，是由上述Balance，Nonce，Code和Storage等字段构成的结构体。

[0102] 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个扩展节点构成。

[0103] 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。

[0104] 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。

[0105] 而分支节点,由于其是并列单nibble的前缀节点,因此分支节点不具有上述prefix字段。

[0106] 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)。

[0107] 分支节点中的16进制字符0～f字段,对应该分支节点所包含的键值对的key值;如果该分支节点为账户地址在MPT树上的搜索路径上的中间节点,则该分支节点的Value字段可以为空值。0～f字段中用于填充下一个节点的hash值。

[0108] 叶子节点中的Key-end,对应该叶子节点所包含的键值对的key值,表示账户地址的最后几个字符。从根节点搜索到叶子节点的搜索路径上的各个节点的key值,构成了一个完整的账户地址。该叶子节点的Value字段填充账户地址对应的账户状态数据;例如,可以对上述Balance,Nonce,Code和storage等字段构成的结构体进行编码后,填充至叶子节点的Value字段。

[0109] 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;

[0110] 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。

[0111] 也即,在将MPT状态树上的node存储至数据库时,可以计算该node所包含的数据内容的hash值(即对node整体进行hash计算),并将计算出的hash值作为key,将该node所包含的数据内容作为value,生成Key-Value键值对;然后,将生成的Key-Value键值对存储至数据库中。

[0112] 由于MPT状态树上的node,是以node所包含的数据内容的hash值为Key,node所包含的数据内容为value进行存储;因此,在需要查询MPT状态树上的node时,通常可以基于node所包含的数据内容的hash值作为key来进行内容寻址。而采用“内容寻址”,对于一些“内容重复”的node,则通常可以进行“复用”,以节约数据存储的存储空间。如图2所示,图2为本说明书示出的一种MPT状态树上的node复用的示意图。

[0113] 需要说明的是,在实际应用中,当区块链产生的一个最新区块中的交易执行完毕后,可能仅仅会导致部分账户的账户状态发生变化;因此,在构建MPT状态树时,并不需要基于区块链中所有的账户当前的状态数据,重新构建一棵完整的MPT状态树,而只需要在该最新区块之前的区块对应的MPT状态树的基础上,对部分账户状态发生变化的账户对应的node进行更新即可。

[0114] 而对于MPT状态树上与账户状态未发生变化的账户对应的node,由于这些node未发生数据更新,因此可以直接复用该最新区块之前的区块对应的MPT状态树上相应的node即可。

[0115] 如图2所示,假设表1中的账户状态数据,为Block N中的交易执行完毕后,区块链上所有账户的最新账户状态;基于表1中的账户状态数据组织成的MPT状态树,仍如图1所

示。

[0116] 假设当Block N+1中的交易执行完毕后，导致上述表1中的账户地址为“a7f9365”的账户状态，由“state3”更新为“state5”；此时，在Block N+1更新MPT状态树时，并不需要在Block N+1中的交易执行完毕后，基于区块链中所有的账户当前的状态数据重新构建一棵MPT状态树。

[0117] 在这种情况下，可以仅将Block N对应的MPT状态树上(即图1示出的MPT状态树)，“key-end”为“9365”的叶子节点中的Value，由“state3”更新为“state5”，并继续更新从root节点到该叶子节点的路径上的所有节点的hash指针；

[0118] 也即，当MPT状态树上的叶子节点发生更新，由于该叶子节点整体的hash值发生更新，那么从根节点到该叶子节点的路径上的所有节点的hash指针也会随之发生更新。

[0119] 例如，请继续参见图2，除了需要更新“key-end”为“9365”的叶子节点中的Value值以外，还需要更新该叶子节点的上一个分支节点(Branch Node)的f字段中填充的，指向该叶子节点的hash指针；进一步的，还可以继续向根节点追溯，继续更新该分支节点的上一个根节点(Root Extension Node)的“Next Node”字段中填充的，指向该分支节点的hash指针。

[0120] 而除了以上发生更新的节点以外，其它未发生更新的节点，都可以直接复用Block N的MPT状态树上对应的节点即可；

[0121] 其中，由于Block N对应的MPT树，最终需要作为历史数据进行保留；因此，在Block N+1更新MPT状态树时，对于这些发生更新的node，并不是在Block N对应的MPT状态树上原来的node的基础上，直接进行修改更新，而是在Block N+1对应的MPT树上重新创建这些发生更新的node。

[0122] 也即，在与Block N+1对应的MPT状态树上，实际上只需要重新创建少量发生更新的node，对于其它未发生更新的node，可以通过直接复用Block N对应的MPT状态树上对应的节点。

[0123] 例如，如图2所示，对于Block N+1对应的MPT状态树上，实际上只需要重新创建一个作为根节点的扩展节点、一个分支节点和一个叶子节点；对于未发生更新的node，可以通过在该MPT状态树上这些重新创建的node中，添加指向Block N对应的MPT状态树上的相应node的hash指针来完成node的“复用”。而Block N对应的MPT状态树上那些更新之前的node，将作为历史账户状态数据进行保存；比如，图2示出的“key-end”为“9365”，且Value为“state3”的叶子节点，将作为历史数据进行保留。

[0124] 在以上例子中，以Block N+1的MPT状态树上的少量node发生内容更新，从而可以“复用”上一个区块Block N的大多数node为例进行了说明。而在实际应用中，Block N+1的MPT状态树上也可能会较上一个区块Block N新增node。在这种情况下，该新增的node虽然无法直接从上一个区块Block N的MPT树中进行“复用”，但有可能从更早之前的区块的MPT状态树上进行“复用”；

[0125] 例如，Block N+1的MPT状态树上新增的node，虽然没在Block N的MPT状态树上出现过，但可能出现在更早的Block的MPT状态树上；比如，出现在Block N-1的MPT状态树上；因此，Block N+1的MPT状态树上新增的node，可以直接复用Block N-1的MPT状态树上对应的node即可。

[0126] 以MPT树为代表的众多采用“内容寻址”的Merkle树，虽然可以通过复用一些“内容重复”的数据节点的方式，来节约区块链中的账户状态数据在数据库中存储时的存储空间；但随着数据的不断增加，大量的历史状态数据过于冗余，势必会导致针对Merkle状态树的访问性能降低，最终会影响区块链平台的TPS(Transactions Per Second)指标；

[0127] 例如，在实际应用中，由于以MPT树为代表的采用“内容寻址”的Merkle树，会存在节点复用的情况，最新区块的Merkle状态树上的很多数据节点，往往是复用之前的历史区块对应的Merkle状态树上的数据节点；这就会导致最新区块的Merkle状态树上维护的很多区块链账户的最新账户状态数据，实际上是“混杂”在众多的历史账户状态数据之中；因此，当需要查找各区块链账户的最新账户状态时，则需要遍历大量的历史状态数据，从而对Merkle状态树的访问性能造成影响。

[0128] 基于此，为了提升Merkle状态树的访问性能，在一些区块链模型中通常同时支持当前Merkle状态树和历史Merkle状态树。

[0129] 在这一类区块链模型中，区块链中的账户状态数据仍然可以被组织成Merkle状态树在数据库中存储；其中，上述Merkle状态树可以包括当前Merkle状态树(Current State Tree)和历史Merkle状态树(History State Tree)；

[0130] 当前Merkle状态树，是由各区块链账户的最新账户状态组织成的Merkle状态树；历史Merkle状态树是由各区块链账户的历史账户状态组织成的Merkle状态树。每一个区块都有一棵与之对应的当前Merkle状态树和历史Merkle状态树。

[0131] 进一步的，当区块链中的任一区块中的交易执行完毕后，一方面，可以基于与该区块中的交易相关的目标账户更新后的最新账户状态，生成与该区块的当前Merkle状态树对应的更新数据节点；另一方面，可以基于上述目标账户更新前的历史账户状态，生成与该区块的历史Merkle状态树对应的历史数据节点；

[0132] 当生成了上述更新数据节点和历史数据节点后，可以基于生成的更新数据节点对该区块的上一区块的当前Merkle状态树上，与上述目标账户对应的数据节点进行修改更新，以得到该区块的当前Merkle状态树；以及，

[0133] 基于生成的上述历史数据节点和复用的上述区块的上一区块的历史Merkle状态树上与该目标账户对应的数据节点以外的其它数据节点，为该区块创建历史Merkle状态树。

[0134] 通过这种方式，由于当前Merkle状态树和历史Merkle状态树之间，并不存在数据节点的复用关系；当前Merkle状态树上只维护各区块链账户的最新账户状态，历史Merkle状态树上只维护各区块链账户的历史账户状态；因此，当需要查找各区块链账户的最新账户状态时，只需要遍历上一区块的当前Merkle状态树，就可以查找到这部分账户在上一区块中的交易执行完毕后的最新账户状态，从而可以提升Merkle状态树的访问性能。

[0135] 而在实际应用中，当区块链中的节点设备发生节点设备宕机，则该节点设备在宕机恢复后，通常可能需要将各区块链账户在最新区块的当前Merkle状态树上对应的最新账户状态，回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态；

[0136] 因此，在同时支持当前Merkle状态树和历史Merkle状态树的区块链模型中，如何在区块链中的节点设备宕机恢复后，将各区块链账户在最新区块的当前Merkle状态树上对

应的最新账户状态，稳定高效的回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态，是这类区块链模型中亟待解决的问题。

[0137] 在相关技术中，如果想要将各区块链账户在最新区块的当前Merkle状态树上的最新账户状态，回滚至各区块链账户在最新区块之前的任一目标历史区块的历史Merkle状态树上对应的历史账户状态，通常可以通过以下示出的方法来实现：

[0138] 方法一：

[0139] 假设上述目标历史区块记为区块N，可以通过重新执行从创世块(即区块链的第一区块)到区块N中包含的所有交易，创建出区块N的历史Merkle状态树，再将上述最新区块的当前Merkle状态树，修改回区块N的当前Merkle状态树，完成状态数据的回滚。

[0140] 然而，通过这种方式，需要重新执行大量的交易，状态数据的回滚代价较高。

[0141] 方法二：

[0142] 假设上述目标历史区块仍然记为区块N，则可以通过遍历数据库中存储的所有历史Merkle状态树，查找到与区块N对应的历史Merkle状态树，再将上述最新区块的当前Merkle状态树，修改回区块N的当前Merkle状态树，完成状态数据的回滚。

[0143] 然而，通过这种方式，由于需要遍历数据库中所有历史Merkle状态树，将与区块N对应的历史Merkle状态树上的数据节点逐个查找出来；因此，状态数据的回滚代价仍然较高。

[0144] 有鉴于此，本说明书提出一种，在同时支持当前Merkle状态树和历史Merkle状态树的区块链模型中，将各区块链账户在最新区块的当前Merkle状态树上对应的最新账户状态，稳定高效的回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态的技术方案。

[0145] 在实现时，当节点设备宕机恢复后，首先可以确定待恢复的目标区块；其中，该目标区块可以为区块链中的最新区块之前的任一历史区块；

[0146] 例如，在实际应用中，管理员可以在节点设备宕机恢复后，通过向节点设备输入配置指令的形式，来指定需要恢复的目标区块的区块号。

[0147] 进一步的，节点设备可以通过迭代执行状态恢复逻辑，直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为上述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态；

[0148] 其中，上述状态恢复逻辑可以包括以下执行逻辑：

[0149] 确定与最新区块中的交易相关的目标账户，并查询上述目标账户在最新区块的上一区块的历史Merkle状态树上对应的历史账户状态；

[0150] 将当前Merkle状态树中与上述目标账户对应的最新账户状态，修改为上述上一区块的历史Merkle状态树中与上述目标账户对应的历史账户状态，并在修改完成后将上述上一区块重新确定为最新区块。

[0151] 以上技术方案中，可以在支持将账户状态数据组织成当前Merkle状态树和历史Merkle状态树的区块链模型中，将各区块链账户在最新区块的当前Merkle状态树上对应的最新账户状态，稳定高效的回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态。

[0152] 请参见图3，图3是一示例性实施例提供的一种区块链状态数据恢复方法的流程

图。所述方法应用于区块链节点设备;所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储;所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树;以及,由各区块链账户的历史账户状态组织成的历史Merkle状态树;所述方法包括以下步骤:

[0153] 步骤302,确定待恢复的目标区块;其中,所述目标区块为所述区块链的最新区块之前的任一历史区块;

[0154] 步骤304、迭代执行状态恢复逻辑,直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态,恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态;

[0155] 其中,所述状态恢复逻辑包括:确定与所述最新区块中的交易相关的目标账户,并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态;将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态,修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态,并在修改完成后将所述上一区块重新确定为所述最新区块。

[0156] 在示出的一种实施方式中,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;

[0157] 例如,在实际应用中,上述Merkle树仍然可以采用以以太坊为代表的公链采用的MPT树;或者,对于基于以太坊架构而衍生出的区块链模型而言,除了可以采用诸如MPT树等改良版的Merkle树以外,也可以采用其他形式的类似于MPT树的,融合了Trie字典树的树形结构的Merkle树变种,在本说明书中不再进行一一列举。

[0158] 区块链中的账户状态数据,可以被组织成Merkle状态树的数据结构,在上述数据库中进行存储;

[0159] 例如,上述Merkle状态树仍然可以是MPT树,可以采用MPT树的数据结构,将区块链的账户状态数据组织成MPT状态树。

[0160] 其中,在本说明书中,区块链中的账户状态数据组织成的Merkle状态树,具体可以包括当前Merkle状态树(Current State Tree)和历史Merkle状态树(History State Tree);当前Merkle状态树,是由各区块链账户的最新账户状态组织成的Merkle状态树;历史Merkle状态树,是由各区块链账户的历史账户状态组织成的Merkle状态树。

[0161] 其中,对于区块链中的每一个区块来说,都有一棵与之对应的当前Merkle状态树和历史Merkle状态树。

[0162] 需要说明的是,当前Merkle状态树和历史Merkle状态树,是两颗独立存在的Merkle状态树;当前Merkle状态树与历史Merkle状态树之间并不存在数据节点的复用关系。

[0163] 其中,所谓当前Merkle状态树与历史Merkle状态树之间并不存在数据节点的复用关系是指,区块链中的任一区块的当前Merkle状态树中,均会包含与所有区块链账户对应的数据节点,而并不需要复用该区块或者该区块之前的任一历史区块的历史Merkle状态树上的任何数据节点。

[0164] 例如,在实际应用中,在为某一目标区块创建当前Merkle状态树时,由于上一区块的当前Merkle状态树中会包含与所有区块链账户对应的数据节点,因此可以直接在上一区

块的当前Merkle状态树的基础上，对部分发生更新的数据节点进行修改更新，即可得到该目标区块的当前Merkle状态树，而并不需要复用该区块或者该区块之前的任一历史区块的历史Merkle状态树上的任何数据节点节点。

[0165] 而对于区块链中的任一区块的历史Merkle状态树来说，可以仅包含与该区块中的交易相关的账户对应的数据节点，而对于与该区块中的交易相关的账户以外的其它区块链账户对应的数据节点，则可以直接复用该区块之前的任一历史区块的历史Merkle状态树上的数据节点；其中，具体的数据节点复用方式，具体仍然可以参考图2以及与图2相关部分的描述。

[0166] 在本说明书中，接入区块链的用户客户端，可以将数据打包成区块链所支持的标准的交易格式，然后发布至区块链；而区块链中的节点设备，可以基于搭载的共识算法与其它节点设备一起，对用户客户端发布至区块链的这些交易进行共识，以此来为区块链产生最新区块；其中，具体的共识过程不再赘述。

[0167] 而区块链中的节点设备在执行了区块链中的任一区块中的交易之后，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，节点设备在该区块中的交易执行完毕后，可以根据上述目标账户发生账户更新前的历史账户状态(即最新区块中的交易执行前的账户状态)和发生账户更新后的最新账户状态(即最新区块中的交易执行后的账户状态)，来为该区块分别创建当前Merkle状态树和历史Merkle状态树。

[0168] 其中，需要说明的是，在本说明书中，节点设备既可以执行区块链中产生的最新区块中的交易，也可以重新执行区块链中的任一历史区块中的交易；

[0169] 也即，在本说明书中，节点设备无论是执行了新产生的最新区块中的交易，还是重新执行了任一历史区块中的交易，都可以基于该区块中的交易执行完毕后，与这些被执行的交易相关的目标账户发生账户更新前的历史账户状态，和发生账户更新后的最新账户状态，为执行的该区块分别创建当前Merkle状态树和历史Merkle状态树。

[0170] 在本说明书中，当区块链上的任一区块中的交易执行完毕后，节点设备可以基于与该区块中的交易相关的目标账户更新后的最新账户状态，生成与该区块的当前Merkle状态树对应的更新数据节点；

[0171] 例如，在实现时，节点设备可以在上一区块的当前Merkle状态树上，查询与该区块中的交易相关的目标账户对应的数据节点，并复制查询到的这些数据节点；然后基于上述目标账户更新后的最新账户状态，对复制的这些数据节点的value进行修改更新，得到上述更新数据节点。其中，对数据节点的value进行查找和更新的过程，具体仍然可以参考图1以及与图1相关部分的描述。

[0172] 相应的，当区块中的交易执行完毕后，节点设备还可以基于与该区块中的交易相关的目标账户更新前的历史账户状态，生成与该区块的历史Merkle状态树对应的历史数据节点；

[0173] 例如，在实现时，节点设备也可以在上一区块的历史Merkle状态树上，查询与该区块中的交易相关的目标账户对应的数据节点，并复制查询到的这些数据节点；然后基于上述目标账户更新前的历史账户状态，对复制的这些数据节点的value进行修改更新，得到上述历史数据节点。

[0174] 在示出的一种实现方式中，上述更新数据节点和历史数据节点均可以表示成写入

集的形式。

[0175] 在区块中的交易的执行过程中,区块链中的节点设备首先可以生成与该区块中的交易对应的读写集(read-write set);对于生成的读写集,也可以在上述数据库中进行存储;例如,在实际应用中,对于生成的读写集,可以作为交易的执行日志保存到该交易对应的收据(receipt)中。

[0176] 其中,上述读写集具体用于记录该区块中的交易在执行前,与该交易相关的账户的账户状态(即账户更新前的历史账户状态);以及,该区块中的交易在执行完毕后,与该交易相关的账户的账户状态(即账户更新后的最新账户状态)。

[0177] 例如,以区块中的交易为转账交易为例,与该区块中的交易对应的读写集可以表示成<account,Balance1,Balance2>的形式;其中,account表示与区块中的交易相关的转账账户,Balance1表示该转账交易在执行前该转账账户的资金余额,Balance2表示该转账交易在执行后该转账账户的资金余额。

[0178] 当区块中的交易执行完毕后,节点设备可以根据生成的与该区块中的交易对应的读写集,进一步生成与该区块的Merkle状态树对应的写入集;该写入集具体用于描述需要写入该区块的Merkle状态树的数据节点。

[0179] 其中,需要说明的是,由于在本说明书中,对于区块链中的任一区块来说,都有当前Merkle状态树和历史Merkle状态树两棵Merkle状态树;因此,当区块中的交易执行完毕后,节点设备可以根据生成的与该区块中的交易对应的读写集,进一步生成双份修改集;其中一份写入集,为与该区块的当前Merkle状态树对应的写入集,该写入集即为需要向该区块的当前Merkle状态树写入的数据节点(即上述更新数据节点);另一份写入集,为与该区块的当前Merkle状态树对应的写入集,该写入集即为需要向该区块的历史Merkle状态树写入的数据节点(即上述历史数据节点)。

[0180] 在本说明书中,当节点设备基于与该区块中的交易相关的目标账户更新后的最新账户状态,生成了与该区块的当前Merkle状态树对应的更新数据节点;以及,基于与该区块中的交易相关的目标账户更新前的历史账户状态,生成了与该区块的历史Merkle状态树对应的历史数据节点之后,可以基于生成的更新数据节点和历史数据节点,来分别为该区块创建当前Merkle状态树和历史Merkle状态树;

[0181] 一方面,节点设备具体可以在上一区块的当前Merkle状态树的基础上,对部分发生更新的数据节点进行修改更新,来为该区块创建当前Merkle状态树。

[0182] 在这种情况下,生成的上述更新数据节点,即为需要对上一区块的当前Merkle状态树上进行修改更新的数据节点。节点设备可以在上述区块的上一区块的当前Merkle状态树上,查询与上述目标账户对应的数据节点,并基于生成的上述更新数据节点对查询到的这些数据节点进行修改更新。当修改更新完成后,即可得到该区块的当前Merkle状态树。

[0183] 例如,在实现时,如果上述更新数据节点表示成写入集的形式,则可以基于该写入集,对上述区块的上一区块的当前Merkle状态树上与上述目标账户对应的数据节点进行修改更新。

[0184] 另一方面,节点设备也可以在上一区块的历史Merkle状态树的基础上,重新创建添加部分发生更新的数据节点,并复用上一区块的历史Merkle状态树上与上述目标账户对应的数据节点以外的其它数据节点,为该区块创建历史Merkle状态树。

[0185] 在这种情况下,生成的上述历史数据节点,即为需要重新创建添加的数据节点。节点设备可以在上述区块的上一区块的历史Merkle状态树上,查询与上述目标账户对应的数据节点以外的其它数据节点,并复用这些查询到的其它数据节点;然后,可以基于生成的上述历史数据节点和复用的这些其它数据节点,为该区块创建历史Merkle状态树。其中,对数据节点的复用过程,具体仍然可以参考图2以及与图2相关部分的描述。

[0186] 例如,在实现时,如果上述历史数据节点也表示成写入集的形式,则可以基于该写入集,和复用的上述区块的上一区块的历史Merkle状态树上与上述目标账户对应的数据节点以外的其它数据节点,为该区块创建历史Merkle状态树。

[0187] 通过以上描述可知,对于历史Merkle状态树上而言,更多的是涉及向历史Merkle状态树上写入新的历史数据节点的操作,并不涉及到对数据节点进行修改更新的操作;因此,历史Merkle状态树本身对于读数据的性能的要求并不高;

[0188] 而对于当前Merkle状态树而言,更多的是涉及对数据节点进行修改更新的操作;而且,在实际应用中,节点设备在执行交易的过程中,通常需要频繁的调用当前Merkle状态树中维护的各区块链账户的最新账户状态;因此,当前Merkle状态树本身对于读数据的性能以及修改数据的性能要求较高;

[0189] 基于此,在实际应用中,将历史Merkle状态树在数据库中进行存储时,可以采对写入性能较高,而对读性能要求不高的数据结构;而将当前Merkle状态树在数据库中进行存储时,可以采对读性能和修改性能较高,对于写入性能要求不高的数据结构。

[0190] 在示出的一种实施方式中,可以将上述当前Merkle状态树上的数据节点组织成B+树(balance+tree)的数据结构在数据库中存储;将历史Merkle状态树上的数据节点组织成LSM树(Log-Structured Merge Tree)的数据结构在数据库中存储。其中,将当前Merkle状态树上的数据节点组织成B+树(balance+tree)的数据结构以及将历史Merkle状态树上的数据节点组织成LSM树的数据结构的具体方式,在本说明书中不再进行详述。

[0191] 在示出的一种实施方式中,上述数据库,具体可以是Key-Value型数据库;例如,在示出的一种实施方式中,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。

[0192] 在这种情况下,无论是当前Merkle状态树还是历史Merkle状态树,最终都会以Key-Value键值对的形式存储至上述数据库。

[0193] 如前所述,由于历史Merkle状态树仍然会存在复用之前区块的历史Merkle状态树上的数据节点的情况,而当前Merkle状态树则不必考虑数据节点复用的情况,更多的是涉及对数据节点的value进行修改;因此,基于这种特性上的差异,在将当前Merkle状态树和历史Merkle状态树上的数据节点以Key-Value键值对的形式存储至上述数据库时,可以为当前Merkle状态树和历史Merkle状态树设计差异化的key键。

[0194] 在示出的一种实施方式中,鉴于历史Merkle状态树仍然会存在复用之前区块的历史Merkle状态树上的数据节点的情况,则历史Merkle状态树上的数据节点在以Key-Value键值对的形式存储至上述数据库时,仍然可以采用该数据节点所包含的数据内容的hash值作为key。

[0195] 而鉴于当前Merkle状态树具有对数据节点的value进行频繁修改的需求,则当前

Merkle状态树上的数据节点在以Key-Value键值对的形式存储至上述数据库时，可以采用该数据节点的节点ID作为key；

[0196] 其中，在实际应用中，上述节点ID具体可以是上述Merkle状态树的根节点到该数据节点的路径对应的字符前缀；或者，基于上述Merkle状态树的根节点到该数据节点的路径对应的字符前缀映射得到的一个节点编号。

[0197] 在本说明书中，当区块链中的节点设备发生宕机，则该节点设备在宕机恢复时，可以将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为指定的目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态；其中，上述目标区块具体可以是指上述最新区块之前的任一历史区块。

[0198] 请参见图4，图4为本说明书示出的一种对当前Merkle状态树进行状态回滚的示意图。

[0199] 假设上述目标区块记为N1，上述最新区块记为N2；

[0200] 节点设备在宕机恢复后，首先可以确定待恢复的目标区块；

[0201] 例如，在实际应用中，管理员可以在节点设备宕机恢复后，可以通过输入配置指令的形式，来指定需要恢复的目标区块的区块号。而节点设备可以通过解析管理员输入的配置指令，来获取上述目标区块的区块号。

[0202] 在示出的一种实施方式中，节点设备在确定待恢复的目标区块后，首先可以确定从上述区块链的创世块至上述目标区块N1，所包含的交易的总数量，是否大于从上述目标区块N1至上述最新区块N2所包含的交易的总数量；

[0203] 例如，在一种实现方式中，可以对上述区块链支持的区块格式进行改进，在每一个区块的区块头中添加一个用于指示当前区块的区块体中包含的交易总数的字段。而节点设备可以通过读取各个区块的区块头中的该字段所记录的数值，获取到各个区块中所包含的交易总数；进而，可以统计出从上述区块链的创世块至上述目标区块N1所包含的交易的总数量；以及，从上述目标区块N1至上述最新区块N2所包含的交易的总数量，来完成比较。

[0204] 请参见图4，如果节点设备确定出，从创世块至上述目标区块N1，所包含的交易的总数量，小于或者等于从上述目标区块N1至上述最新区块N2所包含的交易的总数量，此时节点设备仍然可以基于以上描述的相关技术中的方法一或者方法二来完成状态数据的回滚。

[0205] 反之，如果节点设备确定出，从创世块至上述目标区块N1，所包含的交易的总数量，大于从上述目标区块N1至上述最新区块N2所包含的交易的总数量，此时按照以上描述的相关技术中的方法一或者方法二来完成状态数据的回滚的代价较高；因此，可以通过迭代执行如图4所示出的状态恢复逻辑，来完成状态数据的回滚，直到将最新区块N2的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为上述目标区块N1的历史Merkle状态树上与各区块链账户对应的历史账户状态。

[0206] 以下参照图4，对迭代执行上述状态恢复逻辑进行状态数据回滚的过程进行详细描述。

[0207] 请继续参见图4，首先，节点设备可以将当前正在执行的区块N修改为上述最新区块N2；也即，当前正在执行的区块N的初始值为N2；其中，上述当前执行的区块N，是指上述状态恢复逻辑在软件层面正在进行执行处理的区块。

[0208] 在将上述最新区块N2确定为当前正在执行的区块N后，节点设备可以进一步确定与上述最新区块N2中的交易相关的目标账户；

[0209] 其中，节点设备确定与上述最新区块N2中的交易相关的目标账户的方式，在本说明书中不进行特别限定；

[0210] 在示出的一种实施方式中，节点设备可以通过重新执行上述最新区块N2中的交易，来确定上述最新区块中的交易相关的目标账户；

[0211] 例如，节点设备可以通过重新执行上述最新区块N2中的交易，来生成与上述最新区块中的交易对应的读写集，然后通过读取读写集中记录的账户地址，来明确上述最新区块中的交易相关的目标账户。

[0212] 在示出的另一种实施方式中，如果区块链模型支持将在最新区块中的交易的执行过程中生成的与该最新区块对应的读写集，在数据库中进行存储；那么，节点设备在确定上述最新区块中的交易相关的目标账户时，也可以不重新执行上述最新区块中的交易，而是直接在上述数据库中查询与该最新区块对应的读写集，然后直接读取该读写集中记录的数据，来明确上述最新区块中的交易相关的目标账户。

[0213] 请继续参见图4，当节点设备在确定出上述最新区块N2中的交易相关的目标账户之后，可以进一步查询上述目标账户在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态；

[0214] 其中，节点设备查询上述目标账户在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态的具体方式，在本说明书中也不进行特别限定；

[0215] 在示出的一种实施方式中，节点设备可以基于上述目标账户的账户地址在上述最新区块N2的上一区块的历史Merkle状态树上执行一次查询(相当于根据账户地址执行一次精确查找，并不需要遍历上一区块的历史Merkle状态树)，确定出上述目标账户，在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态。

[0216] 在示出的另一种实施方式中，如果区块链模型支持将在最新区块中的交易的执行过程中生成的与该最新区块对应的读写集，在数据库中进行存储；那么，节点设备在查询上述目标账户在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态时，也可以不在上述最新区块N2的上一区块的历史Merkle状态树上执行查询动作，而是在上述数据库中查询与该最新区块的上一区块对应的读写集，然后读取该读写集中记录的数据，来明确上述目标账户在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态。

[0217] 当确定出上述目标账户，在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态之后，节点设备可以将上述最新区块的当前Merkle状态树中与上述目标账户对应的最新账户状态，修改为上述最新区块N2的上一区块的历史Merkle状态树上与上述目标账户对应的历史账户状态；

[0218] 例如，在实现时，节点设备可以根据上述目标账户在上述最新区块N2的上一区块的历史Merkle状态树上对应的历史账户状态，生成对应于上述最新区块的当前Merkle状态树的写入集，然后基于生成的写入集，对上述目标账户在上述最新区块N2的当前Merkle状态树上对应的数据节点，进行修改更新。

[0219] 当节点设备将将上述最新区块的当前Merkle状态树中与上述目标账户对应的最

新账户状态，修改为上述最新区块N2的上一区块的历史Merkle状态树上与上述目标账户对应的历史账户状态之后，上述最新区块N2的当前Merkle状态树上与各区块链账户对应的最新账户状态，已经回滚至上述最新区块N2的上一区块的历史Merkle状态树上与各区块链账户对应的历史账户状态。

[0220] 此时，节点设备可以将当前正在执行的区块N再次修改为上述最新区块N2的上一区块；然后重新执行以上描述的过程，直到将当前正在执行的区块N修改为上述目标区块N1时停止；

[0221] 其中，需要说明的是，如图4所示，当前正在执行的区块N的取值，可以记为N＝N2——；表示当前正在执行的区块N以N2为初始值，每当迭代执行一次以上描述的状态恢复逻辑之后，当前正在执行的区块N的取值在N2的初始值的基础上减去1。而节点设备将当前正在执行的区块N修改为上述最新区块N2的上一区块的过程，相当于是将上述最新区块N2的上一区块，重新修改为最新区块；

[0222] 通过迭代以上描述的状态恢复逻辑，可以逐个区块进行反向的状态回滚，直到将最新区块N2的当前Merkle状态树上与各区块链账户对应的账户状态，修改为目标区块N1的历史Merkle状态树与各区块链账户对应的账户状态时停止。

[0223] 例如，假设N1＝90；N2＝100；在需要将第100号区块对应的当前Merkle状态树上与各区块链账户对应的最新账户状态，回滚至第90号区块对应的历史Merkle状态树上与各区块链账户对应的历史账户状态，按照以上描述的方法，首先将当前正在执行的区块N设置为100，当按照以上方法将第100号区块对应的当前Merkle状态树上与各区块链账户对应的最新账户状态，回滚至第99号区块对应的历史Merkle状态树上与各区块链账户对应的历史账户状态之后，再将当前正在执行的区块N修改为99(即将99号区块重新设置为最新区块)，并重复以上的过程，直到当前正在执行的区块N修改为90时停止。

[0224] 以上技术方案中，由于采用的是反向逐个重放区块中的交易，来完成状态数据的回滚；因此，当从创世块至上述目标区块N1，所包含的交易的总数量，显著大于从上述目标区块N1至上述最新区块N2所包含的交易的总数量时，采用以上技术方案，相较于以上描述的相关技术中的方法一和方法二，可以显著降低回放成本，可以在支持将账户状态数据组织成当前Merkle状态树和历史Merkle状态树的区块链模型中，将各区块链账户在最新区块的当前Merkle状态树上对应的最新账户状态，稳定高效的回滚至各区块链账户在最新区块之前的任一历史区块的历史Merkle状态树上对应的历史账户状态。

[0225] 与上述方法实施例相对应，本申请还提供了装置的实施例。

[0226] 与上述方法实施例相对应，本说明书还提供了一种区块链状态数据恢复装置的实施例。

[0227] 本说明书的区块链状态数据恢复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。

[0228] 从硬件层面而言，如图5所示，为本说明书的区块链状态数据恢复装置所在电子设备的一种硬件结构图，除了图5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。

[0229] 图6是本说明书一示例性实施例示出的一种区块链状态数据恢复装置的框图。

[0230] 请参考图6,所述区块链状态数据恢复装置60可以应用在前述图5所示的电子设备中,其中所述区块链中的账户状态数据被组织成Merkle状态树在数据库中存储;所述Merkle状态树包括由各区块链账户的最新账户状态组织成的当前Merkle状态树;以及,由各区块链账户的历史账户状态组织成的历史Merkle状态树;所述装置60包括:

[0231] 确定模块601,确定待恢复的目标区块;其中,所述目标区块为所述区块链的最新区块之前的任一历史区块;

[0232] 恢复模块602,迭代执行状态恢复逻辑,直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态,恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态;

[0233] 所述状态恢复逻辑包括:

[0234] 确定与所述最新区块中的交易相关的目标账户,并查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态;

[0235] 将所述最新区块的当前Merkle状态树中与所述目标账户对应的最新账户状态,修改为所述上一区块的历史Merkle状态树中与所述目标账户对应的历史账户状态,并在修改完成后将所述上一区块重新确定为所述最新区块。

[0236] 在本实施例中,所述确定模块601:

[0237] 重新执行所述最新区块中的交易,以确定所述最新区块中的交易相关的目标账户;或者,查询与所述最新区块对应的读写集,以确定所述最新区块中的交易相关的目标账户。

[0238] 所述恢复模块602:

[0239] 在所述最新区块的上一区块的历史Merkle状态树上查询与所述目标账户对应的历史账户状态;或者,在与所述最新区块的上一区块对应的读写集中,查询所述目标账户在所述最新区块的上一区块的历史Merkle状态树上对应的历史账户状态。

[0240] 在本实施例中,所述恢复模块602进一步:

[0241] 在迭代执行状态恢复逻辑之前,确定从所述区块链的创世块至所述目标区块所包含的交易的总数量,是否大于从所述目标区块至所述最新区块所包含的交易的总数量;如果是,进一步迭代执行所述状态恢复逻辑。

[0242] 在本实施例中,所述装置60还包括:

[0243] 生成模块603(图6中未示出),在所述区块链中的任一区块中的交易执行完毕后,基于与所述区块中的交易相关的目标账户的最新账户状态,生成与所述区块的当前Merkle状态树对应的更新数据节点和与所述最新区块的历史Merkle状态树对应的历史数据节点;

[0244] 修改模块604(图6中未示出),基于生成的所述更新数据节点对所述区块的上一区块的当前Merkle状态树上与所述目标账户对应的数据节点进行修改更新,以得到所述区块的当前Merkle状态树;

[0245] 创建模块605(图6中未示出),基于生成的所述历史数据节点和复用的所述区块的上一区块的历史Merkle状态树上与所述目标账户对应的数据节点以外的其它数据节点,为所述区块创建历史Merkle状态树。

[0246] 在本实施例中,所述当前Merkle状态树上的数据节点被组织成B+树的数据结构在数据库中存储;所述历史Merkle状态树上的数据节点被组织成LSM树的数据结构在数据库中存储。

[0247] 在本实施例中,所述数据库为Key-Value数据库;

[0248] 所述Merkle状态树上的数据节点以Key-Value键值对的形式存储在所述数据库中;其中,所述当前Merkle状态树上的数据节点的key为所述数据节点的节点ID;所述历史Merkle状态树上的数据节点的key为所述数据节点包含的数据内容的hash值。

[0249] 在本实施例中,所述数据库为LevelDB数据库;或者基于LevelDB架构的数据库。

[0250] 在本实施例中,所述数据库为基于LevelDB架构的Rocksdb数据库。

[0251] 在本实施例中,所述Merkle树为融合了Trie字典树的树形结构的Merkle树变种。

[0252] 在本实施例中,所述Merkle状态树为Merkle Patricia Tree状态树。

[0253] 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。

[0254] 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。

[0255] 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。

[0256] 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。

[0257] 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。

[0258] 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。

[0259] 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。

[0260] 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”。

[0261] 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。

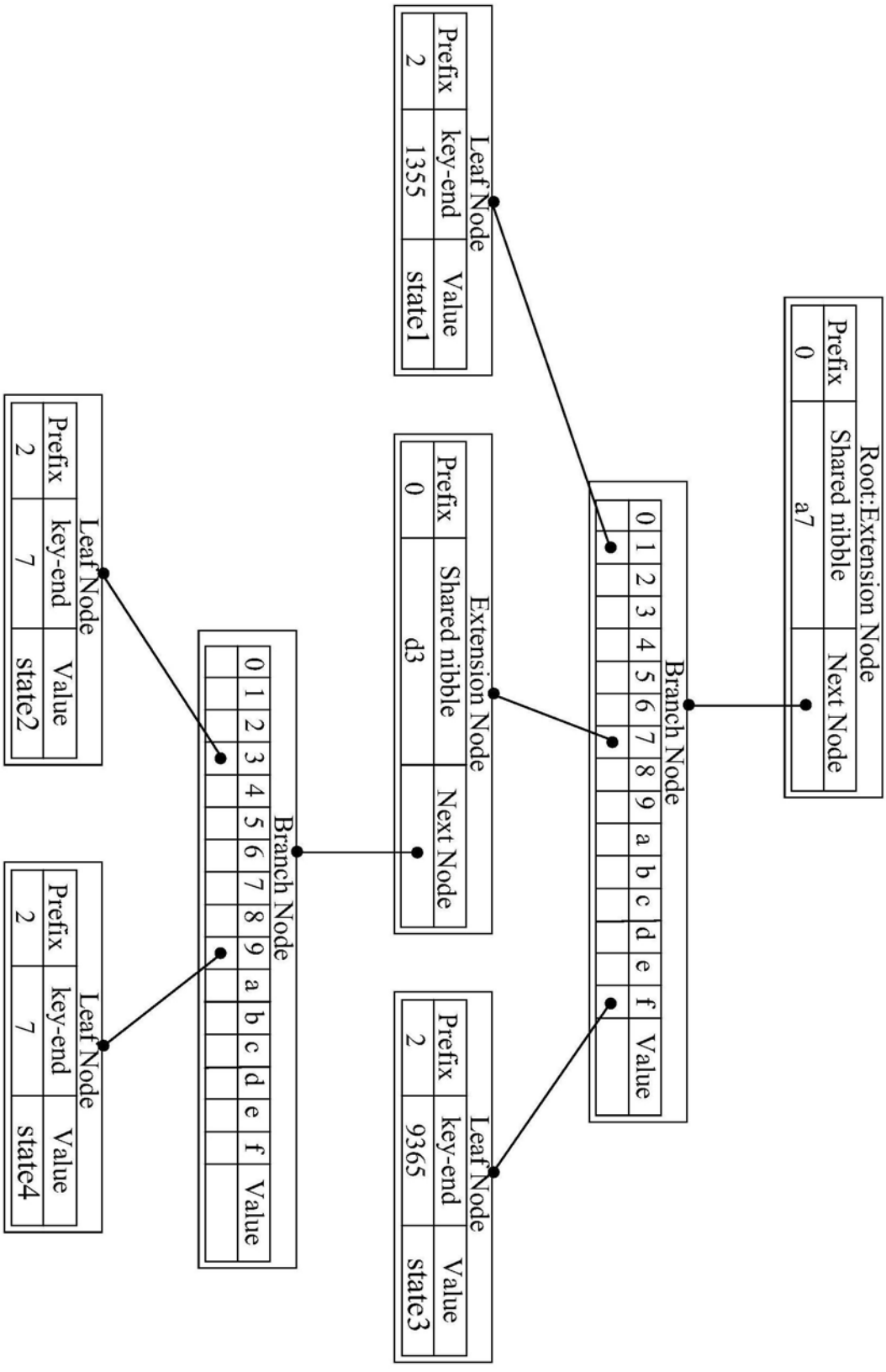

Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
Leaf Node
Prefix
key-end
Value
2
1355
state1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
9365
state3
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
Leaf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

图1

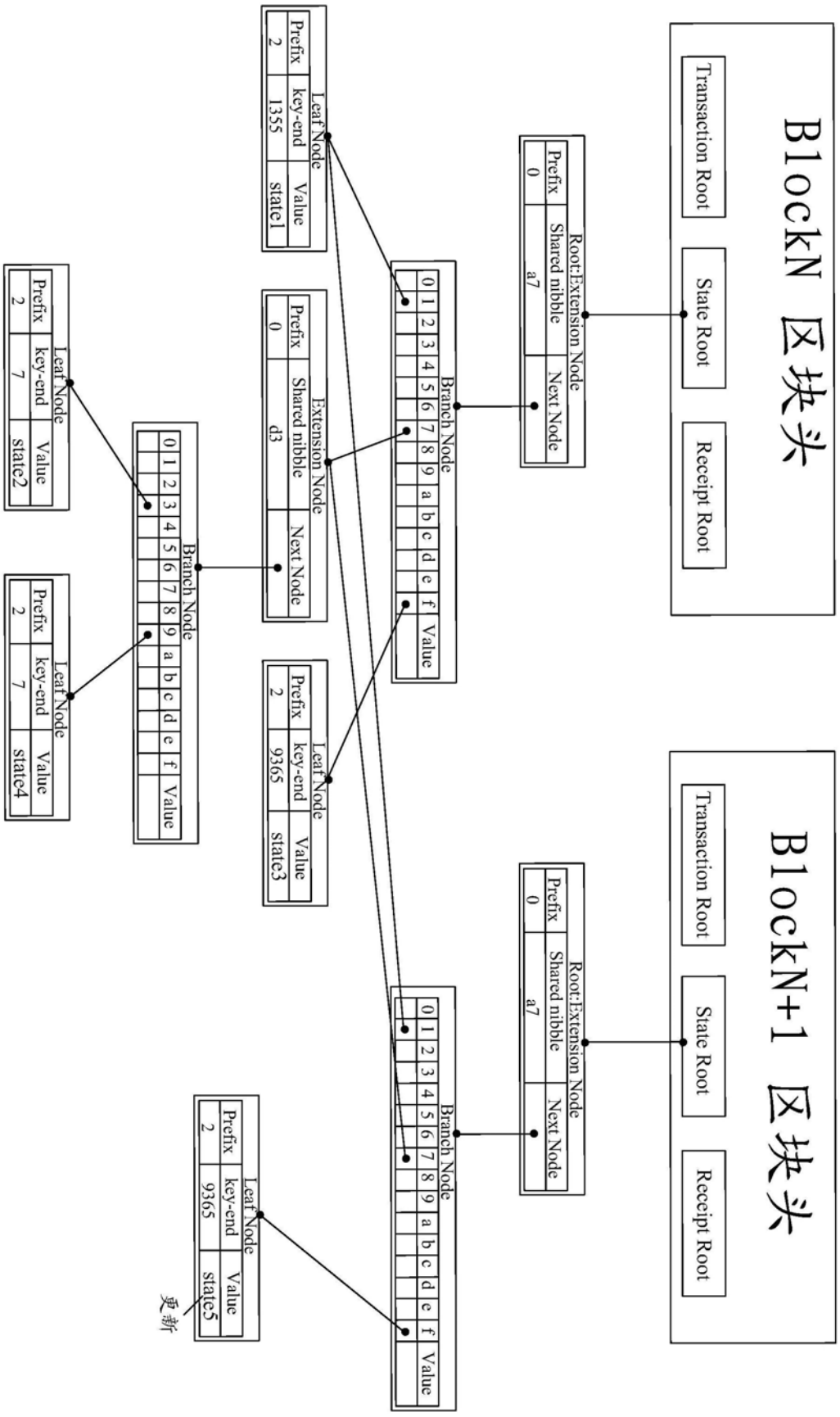

BlockN 区块头
Transaction Root
State Root
Receipt Root
Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
Leaf Node
Prefix
key-end
Value
2
1355
state1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
9365
state3
Leaf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

图2

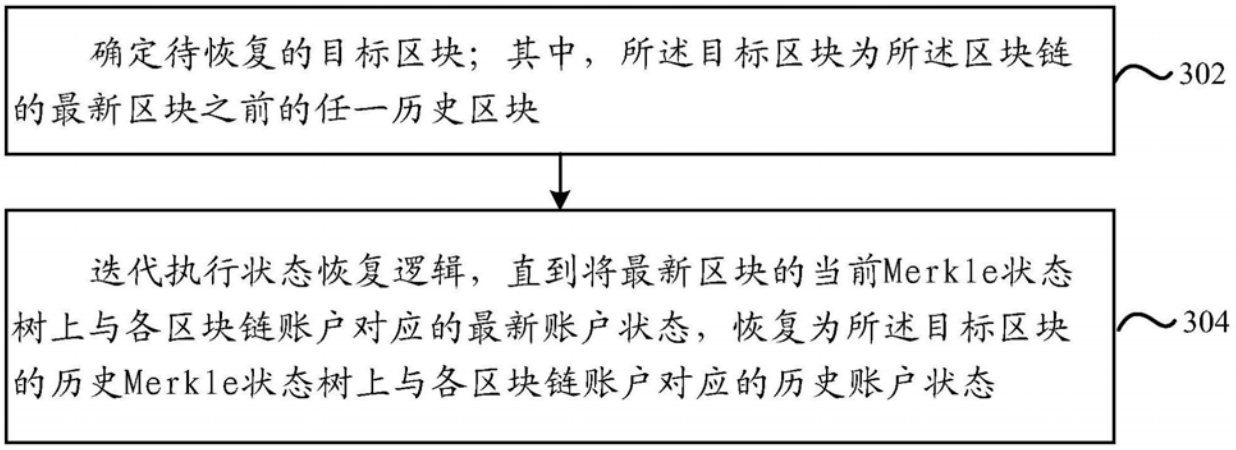

确定待恢复的目标区块；其中，所述目标区块为所述区块链的最新区块之前的任一历史区块
302
迭代执行状态恢复逻辑，直到将最新区块的当前Merkle状态树上与各区块链账户对应的最新账户状态，恢复为所述目标区块的历史Merkle状态树上与各区块链账户对应的历史账户状态
304

图3

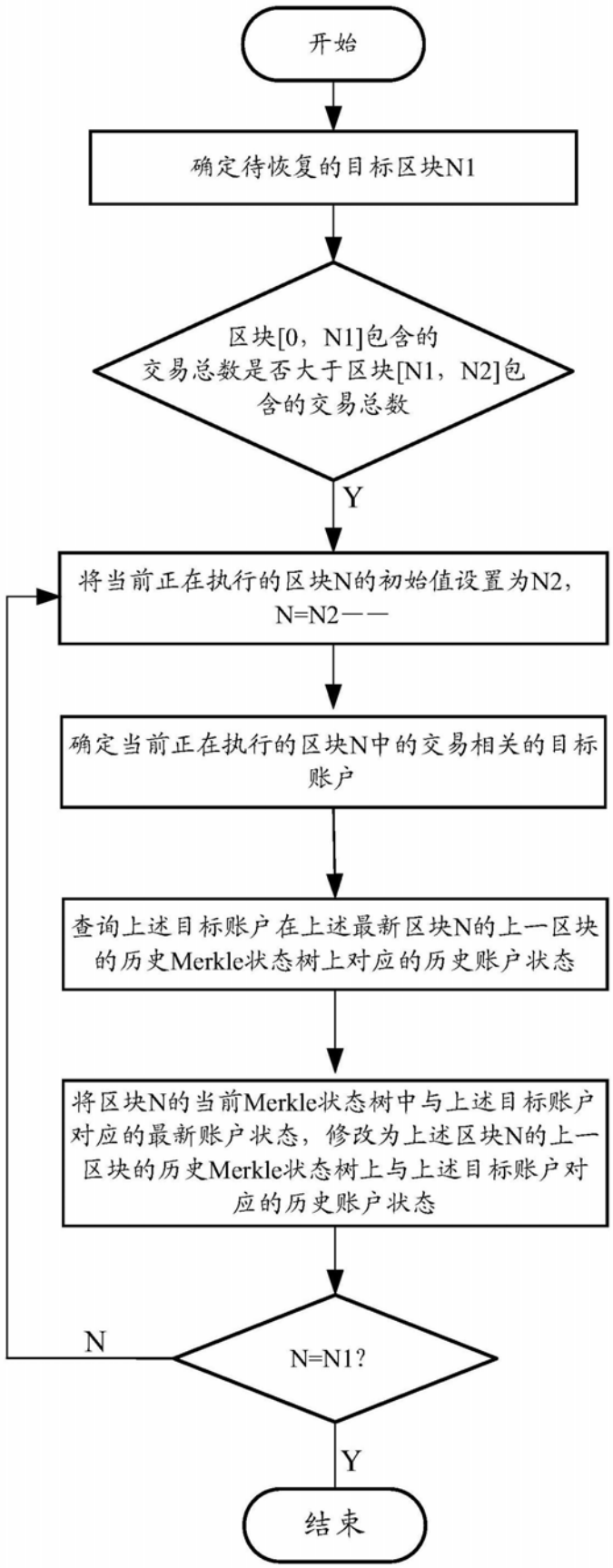

开始
确定待恢复的目标区块N1
区块[0，N1]包含的交易总数是否大于区块[N1，N2]包含的交易总数
Y
将当前正在执行的区块N的初始值设置为N2，N=N2——
确定当前正在执行的区块N中的交易相关的目标账户
查询上述目标账户在上述最新区块N的上一区块的历史Merkle状态树上对应的历史账户状态
将区块N的当前Merkle状态树中与上述目标账户对应的最新账户状态，修改为上述区块N的上一区块的历史Merkle状态树上与上述目标账户对应的历史账户状态
N
N=N1?
Y
结束

图4

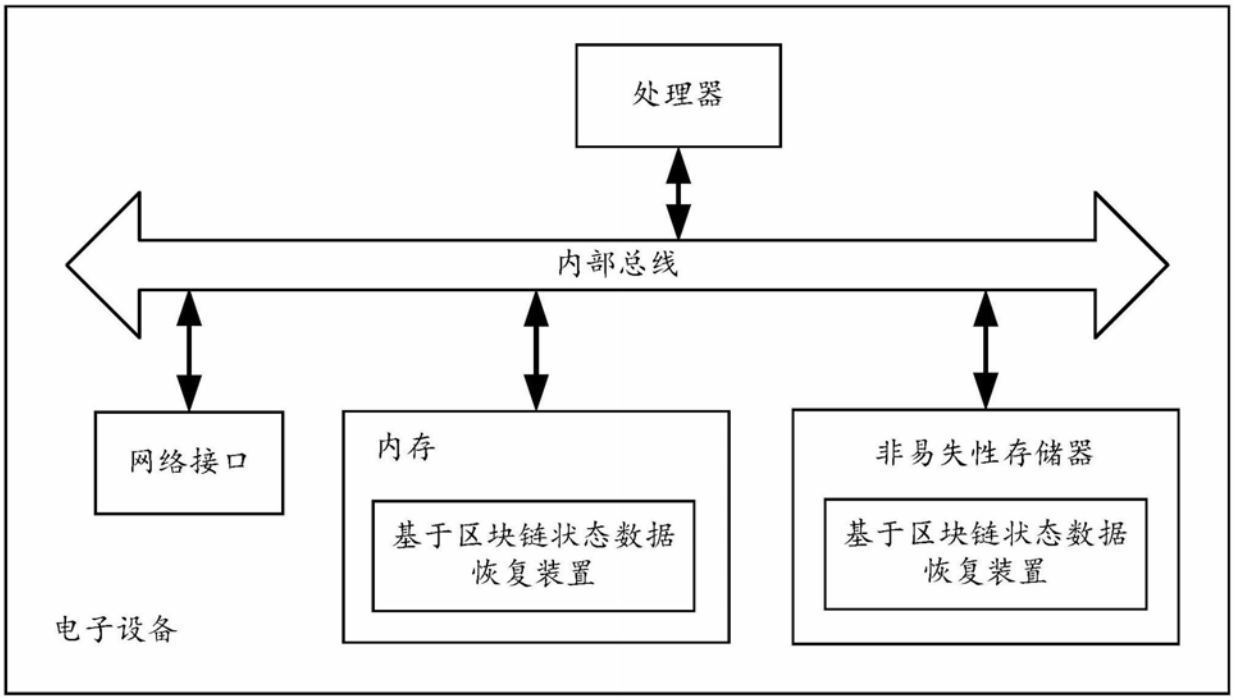

处理器
内部总线
网络接口
内存
基于区块链状态数据恢复装置
非易失性存储器
基于区块链状态数据恢复装置
电子设备

图5

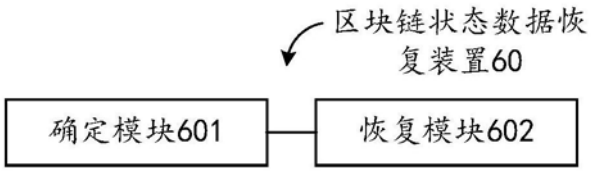

区块链状态数据恢复装置60
确定模块601
恢复模块602

图6