

基于区块链的数据处理方法、装置、设备及可读存储介质

本申请要求于 2022 年 2 月 24 日提交中国专利局、申请号 202210174690.7、申请名称为“基于区块链的数据处理方法、装置、设备及可读存储介质”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及计算机技术领域，尤其涉及一种基于区块链的数据处理方法、装置、设备及可读存储介质。

背景技术

区块链系统是由多个区块链节点组成的区块链网络，每个区块链节点会存储该区块链网络对应的最新上链世界状态，即区块链中的最大区块高度的区块上链后，区块链网络中各个账户所拥有的状态数据共同组成的世界状态。

目前的区块链网络中，区块内交易的执行会修改若干账户所拥有的部分状态数据，因此每当有一个区块上链后，就会有对应有一个新的最新上链世界状态。新区块在共识时，又需要依赖其父区块上链后对应的最新上链世界状态来确定自己上链后对应的最新上链世界状态，因此目前的共识机制中往往仅允许一个区块进行共识。若采用多个区块同时进行共识的共识机制，则往往无法保证区块链系统的状态数据的一致性。

发明内容

本申请实施例提供了一种数据处理方法、装置、设备及可读存储介质，可以维护多个待上链区块的世界状态，保证区块链系统的状态数据的一致性。

本申请实施例一方面提供了一种基于区块链的数据处理方法，该方法由计算机设备执行，包括：

生成初始区块，创建初始区块对应的第一世界状态；第一世界状态具有指向第二世界状态的链式关系，第二世界状态为初始区块的父区块对应的世界状态；第一世界状态和第二世界状态存在于世界状态链中；若父区块为已上链区块中具有最大区块高度的区块，则第二世界状态为共识根世界状态；世界状态链中的第一个世界状态为最新上链世界状态；共识根世界状态具有指向最新上链世界状态的链式关系；最新上链世界状态用于记录已上链区块对应的最新世界状态；

根据初始区块中的交易更新第一世界状态，得到初始更新状态，根据世界状态链对初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，更新后的第一世界状态包括干净现世界状态树；

将干净现世界状态树对应的第一世界状态根写入初始区块，得到待上链区块，将待上链区块发送至共识节点，以使共识节点基于第一世界状态根对待上链区块进行共识处理，得到共识结果；

若共识结果为共识通过结果，则将待上链区块写入区块链中，通过世界状态链将更新后的第一世界状态逐级提交至最新上链世界状态，将更新后的第一世界状态确定为新的共识根世界状态，新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

本申请实施例一方面提供了一种基于区块链的数据处理装置，该装置部署在计算机设备上，包括：

初始化模块，用于生成初始区块；

状态创建模块，用于创建初始区块对应的第一世界状态；第一世界状态具有指向第二世界状态的链式关系，第二世界状态为初始区块的父区块对应的世界状态；第一世界状态和第二世界状态存在于世界状态链中；若父区块为已上链区块中具有最大区块高度的区块，则第二世界状态为共识根世界状态；世界状态链中的第一个世界状态为最新上链世界状态；共识根世界状态具有指向最新上链世界状态的链式关系；最新上链世界状态用于记录已上链区块对应的最新世界状态；

第一状态更新模块，用于根据初始区块中的交易更新第一世界状态，得到初始更新状态；

第二状态更新模块，用于根据世界状态链对初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，更新后的第一世界状态包括干净现世界状态树；

共识模块，用于将干净现世界状态树对应的第一世界状态根写入初始区块，得到待上链区块，将共识区块发送至共识节点，以使共识节点基于第一世界状态根对待上链区块进行共识处理，得到共识结果；

状态提交模块，用于若共识结果为共识通过结果，则将待上链区块写入区块链中，通过世界状态链将更新后的第一世界状态逐级提交至最新上链世界状态，将更新后的第一世界状态确定为新的共识根世界状态，新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

本申请实施例一方面提供了一种计算机设备，包括：处理器、存储器、网络接口；

上述处理器与上述存储器、上述网络接口相连，其中，上述网络接口用于提供数据通信网元，上述存储器用于存储计算机程序，上述处理器用于调用上述计算机程序，以执行本申请实施例中的方法。

本申请实施例一方面提供了一种计算机可读存储介质，上述计算机可读存储介质中存储有计算机程序，上述计算机程序适于由处理器加载并执行本申请实施例中的方法。

本申请实施例一方面提供了一种计算机程序产品，该计算机程序产品包括计算机程序/指令，该计算机程序/指令存储在计算机可读存储介质中，计算机设备的处理器从计算机可读存储介质读取该计算机程序/指令，处理器执行该计算机程序/指令，使得该计算机设备执行本申请实施例中的方法。

本申请实施例中，通过构建世界状态链，可以支持同时维护不同区块高度或者同一区块高度不同分支的世界状态，即每创建一个新的区块，都会为该新区块创建一个世界状态，将该世界状态加入世界状态链中，世界状态链中的第一个世界状态为已上链的最新的区块内的交易执行过后的最新上链世界状态，世界状态链中的第二个世界状态为是上一次被共识确认的共识根世界状态，再往后的世界状态即为正在共识的区块对应的世界状态，共识根世界状态具有指向最新上链世界状态的链式关系，正在共识的区块对应的世界状态具有指向其父区块对应的世界状态的链式关系，若其父区块为已上链区块，则其具有指向共识

根世界状态的链式关系，通过每个世界状态具有的链式关系，从而形成世界状态链的链式结构。因此可以保证无论同时共识多少区块，在区块共识通过后，其对应的世界状态均会按照世界状态链的链式结构逐级提交至最新上链世界状态，保证区块链系统的状态数据的一致性。

附图说明

为了更清楚地说明本申请实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图1是本申请实施例提供的一种区块链节点系统的结构示意图；

图2a是本申请实施例提供的一种世界状态的结构示意图；

图2b是本申请实施例提供的一种账户状态的结构示意图；

图2c是本申请实施例提供的一种世界状态管理的场景示意图；

图2d是本申请实施例提供的一种世界状态链的结构示意图；

图3是本申请实施例提供的一种基于区块链的数据处理方法的流程示意图；

图4a-图4c是本申请实施例提供的一种基于区块链的数据处理的场景示意图；

图5是本申请实施例提供的一种基于交易的第一世界状态更新方法的流程示意图；

图6是本申请实施例提供的一种干净世界状态树构建方法的流程示意图；

图7是本申请实施例提供的一种基于区块链的数据处理装置的结构示意图；

图8是本申请实施例提供的一种计算机设备的结构示意图。

具体实施方式

下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

为便于理解，以下将对区块链及其相关概念进行阐述：

区块链：区块链是一种分布式数据存储、点对点传输、共识机制以及加密算法等计算机技术的新型应用模式，主要用于对数据按时间顺序进行整理，并加密成账本，使其不可被篡改和伪造，同时可进行数据的验证、存储和更新。区块链本质上是一个去中心化的数据库，该数据库中的每个节点均存储一条相同的区块链，区块链网络可以将节点区分为共识节点和业务节点，其中共识节点负责区块链全网的共识。对于区块链网络中交易数据被写入账本的过程可以为：客户端发送交易数据至业务节点，随后该交易数据以接力棒的方式在区块链网络中的业务节点之间传递，直到共识节点收到该交易数据，共识节点再将该交易数据打包进区块，与其他共识节点之间进行共识，在共识通过后，将携带该交易数据的区块写入账本。

区块：是在区块链网络上承载交易数据（即交易业务）的数据包，是一种被标记上时间戳和之前一个区块的哈希值的数据结构，区块经过网络的共识机制验证并确定区块中的交易。

哈希值：也称作信息特征值或特征值，哈希值是通过哈希算法将任意长度的输入数据转换为密码并进行固定输出而生成的，不能通过解密哈希值来检索原始输入数据，它是一个单向的加密函数。在区块链中，每个区块（除了初始区块）都包含前继区块的哈希值，前继区块被称为当前区块的父区块。哈希值是区块链技术中的潜力核心基础和最重要的方面，它保留了记录和查看数据的真实性，以及区块链作为一个整体的完整性。

交易：区块链账户发送的交易，内有交易哈希作为唯一标识，包含有账户地址标识所属发送该交易的区块链账户。

状态快照：表示一定范围内的所有状态数据的快照，固定长度的哈希值，只要任何状态数据有任何变化，状态快照均发生变化，以此可以判断所有状态数据的完整性。

账户：区块链账户，内有账户地址作为唯一标识，包含有余额作为现世界状态下该账户的数字资产，每个账户有其若干状态数据，账户含有账户状态根作为其所有状态数据的状态快照。

状态数据：存储于区块链中的账户的数据，例如存储某个账户有多少资产，或者其它智能合约需要读取或者更新的数据，每个状态数据以键值对的形式表示。

上链：每个区块包含若干个交易，当一个区块经过共识插入到每个区块链节点内的区块链数据结构的过程，称为上链。

世界状态：区块链内各个账户所拥有的持久化数据共同组成的视图，由于交易的执行本身会修改若干个账户的部分持久化数据，因此每个区块内的每个交易执行后会有一个新的世界状态，每个区块上链后更是会更新每个区块链节点的世界状态。在同一区块上链后，每个区块链节点的世界状态必然一致。

MPT：Merkle Patricia Trie，默克尔帕特里夏树，一种组织状态数据的树形结构，内含数据节点、拓展节点、分支节点三种节点。其中，状态数据包含有状态数据键和状态数据值。其中，树据节点是树型结构的叶子节点，只会出现在 MPT 底部，存储着实际的状态数据值；拓展节点是仅有一个子节点的父节点，包含一个任意长度字符串代表状态数据键的一部分，和另一个指向子节点的树节点哈希；分支节点可以有 1 至 16 个子节点的父节点，有一个容量为 16 的哈希值数组，数组中的这 16 个位置分别对应 16 进制下的 0-9-a-f，并且分别有可能作为树节点哈希而指向一个子节点。其中，树节点哈希是 MPT 中子节点的哈希值，存储在父节点。

状态树：本申请实施例中包含有账户状态树和世界状态树，均是以 MPT 作为组织区块链的状态数据的数据结构，可以提供以下操作：

(1) 查询：可以根据状态数据键，从状态树中查询出状态数据值，此时需要向状态树提供树节点查询来源。其中，树节点查询来源是帮助状态树以树节点哈希为条件从中快速查找树节点的数据结构。状态树中的每个树节点，在数据库中是以键值对的形式存储，存储键为树节点的哈希值，存储值为树节点，因此可以根据树节点哈希从数据库中快速查找树节点。而因为状态树中每个父节点均包含其所以子节点的树节点哈希，所以可以由父节点从数据库中拓展出其所有子节点。

(2) 更新：可以根据状态数据键将状态数据值更新到状态树中。

(3) 获取状态树根：计算并获取状态树的树根的哈希值。

(4) 提交：把状态树的每个树节点提交到树节点更新缓存中。因为状态树中的每个树节点，在数据库中是以键值对的形式存储，存储键为树节点的哈希值，存储值为树节点。因此，当状态树进行提交操作时，需要一个以树节点哈希为键和以树节点为值，接收键值对并存储的数据结构，此数据结构称为树节点更新缓存。

请参见图 1，图 1 是本申请实施例提供的一种区块链节点系统的结构示意图。如图 1 所示的区块链节点系统可以对应于区块链网络，该区块链网络可以包括但不限于联盟链所对应的区块链网络。区块链节点系统是指用于进行区块链节点与区块链节点之间数据共享的系统，该区块链节点系统中可以包括多个区块链节点，多个区块链节点具体可以包括区块链节点 10a、区块链节点 10b、区块链节点 10c、区块链节点 10d、…、区块链节点 10n。其中，每个区块链节点在进行正常工作时可以接收到外界发送的数据，并基于接收到的数据进行区块上链处理，也可以向外界发送数据。为了保证各个区块链节点之间的数据互通，每个区块链节点之间可以存在数据连接，例如区块链节点 10a 和区块链节点 10b 之间存在数据连接，区块链节点 10a 和区块链节点 10c 之间存在数据连接，区块链节点 10b 和区块链节点 10c 之间存在数据连接。

可以理解的是，区块链节点之间可以通过上述数据连接进行数据或者区块传输。区块链网络可以基于节点标识实现区块链节点之间的数据连接，对于区块链网络中的每个区块链节点，均具有与其对应的节点标识，而且上述每个区块链节点均可以存储与自身有相连关系的其他区块链节点的节点标识，以便后续根据其他区块链节点的节点标识，将获取到的数据或生成的区块广播至其他区块链节点，例如区块链节点 10a 中可以维护一个如表 1 所示的节点标识列表，该节点标识列表保存着其他节点的节点名称和节点标识：

表 1

节点名称	节点标识
节点 10a	117.114.151.174
节点 10b	117.116.189.145
节点 10c	117.114.151.183
节点 10d	117.117.125.169
...	...
节点 10n	117.116.189.125

其中，节点标识可为网络之间互联的协议（Internet Protocol，IP）地址以及其他任意一种能够用于标识区块链网络中节点的信息，表 1 中仅以 IP 地址为例进行说明。例如，区块链节点 10a 可以通过节点标识 117.116.189.145 向区块链节点 10b 发送信息（例如，区块），且区块链节点 10b 可以通过节点标识 117.114.151.174 确定该信息是由区块链节点 10a 所发送的。

在区块链中，在将一个区块进行上链之前，该区块必须经过区块链网络中的共识节点进行共识，在共识通过后才能将该区块添加到区块链上。可以理解的是，当区块链被用于政府或者商业机构的一些场景中时，并非区块链中的所有参与节点（即上述区块链节点系统中的区块链节点）都有足够的资源和必要性成为区块链的共识节点。例如，在图 1 所示的区块链节点系统中，可以将区块链节点 10a、区块链节点 10b、区块链节点 10c 和区块链节点 10d 作为该区块链节点系统中的共识节点。区块链节点系统中的共识节点参与共识，也就是对区块（包含一批交易）进行共识，包括生成区块，对区块进行投票；而非共识节点不参与共识，但会帮助传播区块和投票消息，以及相互同步状态等。

其中，区块链节点可以包含以下模块：

交易池：每个区块链节点会接收用户发送的交易，并缓存至交易池中，根据共识机制某个区块链节点被选为出块节点时，该区块链节点会在内存中创建新区块，并把交易池中的交易打包至新区块内。

共识模块：对交易进行打包生成候选区块，并在各个区块链节点之间，基于共识机制达成共识，产生新区块。

智能合约容器：共识机制执行过程中需要执行区块内的每个交易，本质是执行交易中的智能合约的过程，执行智能合约中可能需要读取或者更新状态数据，本模块为智能合约执行提供执行环境。

世界状态管理：负责对每个共识中区块的世界状态进行维护管理。

键值数据库：以键值对的方式持久化存储数据，可以根据键快速查找值，在本申请中负责持久化存储每个区块的数据和树节点存储，树节点存储即状态树的每个树节点的哈希值为键，每个树节点为值，因此可以根据树节点哈希快速查找树节点。

其中，可以理解的是，上述的数据连接不限定连接方式，可以通过有线通信方式进行直接或间接地连接，也可以通过无线通信方式进行直接或间接地连接，还可以通过其他连接方式，本申请在此不做限制。

可以理解的是，本申请实施例所提供的数据处理方法可以由计算机设备执行，计算机设备包括但不限于上述区块链节点（可以为服务器或者终端）。上述服务器可以是独立的物理服务器，也可以是多个物理服务器构成的服务器集群或者分布式系统，还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN、以及大数据和人工智能平台等基础云计算服务的云服务器。上述终端可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表等，但并不局限于此。

可以理解的是，本申请实施例可应用于各种场景，包括但不限于云技术、人工智能、智慧交通、辅助驾驶等场景。

可以理解的是，在本申请的具体实施方式中，涉及到的状态数据等相关的数据，当本申请以上实施例运用到具体产品或技术中时，需要获得用户许可或者同意，且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。

如图 1 所示, 区块链网络中的出块节点, 例如, 上述区块链节点系统中的区块链节点 10a, 在生成初始区块后, 可以创建该初始区块对应的第一世界状态, 该第一世界状态具有指向第二世界状态的链式关系, 第二世界状态为初始区块的父区块对应的世界状态。此时第一世界状态和第二世界状态存在于世界状态链中, 若父区块为已上链区块中具有最大区块高度的区块, 则第二世界状态为共识根世界状态, 共识根世界状态具有指向最新上链世界状态的链式关系, 最新上链世界状态为世界状态链中的第一个世界状态, 最新上链世界状态用于记录已上链区块对应的最新世界状态; 然后, 根据初始区块中的交易更新第一世界状态, 得到初始更新状态, 根据世界状态链对初始更新状态进行构建干净状态树处理, 得到包含干净现世界状态树的更新后的第一世界状态; 将干净现世界状态树对应的第一世界状态根写入初始区块, 得到待上链区块, 将共识区块发送至共识节点, 以使共识节点基于第一世界状态根对待上链区块进行共识处理, 得到共识结果; 若共识结果为共识通过结果, 则将待上链区块写入区块链中, 通过世界状态链将更新后的第一世界状态逐级提交至最新上链世界状态, 将更新后的第一世界状态确定为新的共识根世界状态, 新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

为更好地理解上述世界状态管理的过程, 请先参见图 2a, 图 2a 是本申请实施例提供的一种世界状态的结构示意图, 如图 2a 所示, 世界状态 20 可以包含有前世界状态指针 201、脏账户地址集合 202、账户状态映射 203、世界状态树 204、树节点缓存 205。其中, 前世界状态指针 201 指向世界状态 20 的前世界状态, 可以通过前世界状态指针 201 在内存中定位世界状态 20 的前世界状态, 上述最新上链世界状态的前世界状态指针为空。其中, 脏账户地址集合 202 是指世界状态 20 内经过更新但还未更新到世界状态树 204 中的账户状态的账户地址的集合。其中, 账户状态映射 203 是指由账户地址映射到账户状态的映射结构, 在账户状态映射 203 中, 可以根据账户地址快速查找账户状态。其中, 世界状态树 204 是指以账户地址为键, 以账户状态根为值进行组织的 MPT 数据结构。其中, 账户状态根是指账户地址对应的账户状态中包含的账户状态树的树根的哈希值。其中, 树节点缓存 205 包含有前树节点缓存指针 2051 和树节点映射 2052, 前树节点缓存指针 2051 指向世界状态 20 的前世界状态的树节点缓存, 上述最新上链世界状态的树节点缓存的前树节点缓存指针指向键值数据库中的树节点存储; 树节点映射 2052 为由树节点哈希映射到树节点的映射结构, 在树节点映射 2052 中可以根据树节点哈希快速查找树节点。树节点缓存 205 会作为世界状态 20 的世界状态树 204 和账户状态映射 203 中的每个账户状态的账户状态树的树节点查询来源, 世界状态树 204 和每个账户状态的账户状态树可以根据树节点哈希从树节点缓存的树节点映射中查找树节点, 当未能找到树节点时, 会根据前树节点缓存指针在内存中定位前树节点缓存, 并继续根据树节点哈希从前树节点缓存的树节点映射中查找树节点, 仍然找不到则继续向前定位, 如此类推, 直到最前的树节点缓存的前树节点缓存指针为指向键值数据库的树节点存储, 则继续根据树节点哈希从树节点存储中查找树节点。树节点缓存 205 还会作为世界状态树 204 和账户状态映射 203 中的每个账户状态的账户状态树的树节点更新缓存, 世界状态树 204 和账户状态树可以进行提交操作, 即以树节点哈希为键和以树节点为值更新到树节点缓存的树节点映射中。

请一并参见图 2b, 图 2b 是本申请实施例提供的一种账户状态的结构示意图。如图 2b 所示, 账户状态 21 可以包含有账户地址 211、账户状态根 212、脏状态数据键集合 213、状态数据映射 214 以及账户状态树 215。其中, 账户地址 211 是账户的唯一标识。其中, 账户状态根 212 是账户状态树 215 的状态树根, 也就是其树根的哈希值。其中, 脏状态数据键集合 213 是指账户状态 21 中经过更新但未更新到账户状态树 215 的状态数据的键的集合。其中, 状态数据映射 214 是指由状态数据键映射到状态数据值的映射结构, 在状态数据映射 214 中可以根据状态数据键快速查找状态数据值。其中, 账户状态树 214 是以状态数据键为键, 以状态数据值为值进行组织的 MPT 数据结构。

本申请实施例提供的方法, 可以通过上述图 2a 所示的结构来构建世界状态, 通过上述图 2b 所示的结构来构建账户状态, 然后对构建出的账户状态和世界状态进行管理, 为便于理解, 请一并参见图 2c, 图 2c 是本申请实施例提供的一种世界状态管理的场景示意图。如图 2a 所示, 出块节点 22 (可以为上述图 1 所示的区块链节点系统中的任一区块链节点, 例如区块链节点 10b) 的键值数据库 23 中存储着已上链的区块链数据结构 231, 该区块链数据结构 231 中包含了所有已经上链的区块。此外, 键值数据库 23 中还包含有树节点存储 232, 树节点存储 232 中, 以树节点哈希值为键、树节点为值对区块链网络中每个账户对应的账户状态树以及最新上链世界状态对应的世界状态树中的每个树节点及其哈希值进行存储。其中, 最新上链世界状态是指已上链的最新的区块内的交易执行过后的世界状态。

如图 2c 所示, 出块节点可以在内存 24 中维护正在共识中的若干不同高度的区块, 例如区块高度 $n+1$ 的区块 $n+1$, 区块高度为 $n+2$ 的区块 $(n+2)$ a 等等。当然按照共识机制的执行还有可能同一高度下不同区块分支, 例如, 区块高度为 $n+2$ 的区块 $(n+2)$ a 和区块 $(n+2)$ b, 但最终只会有其中一个分支在区块链网络的区块链节点中形成共识并完成上链操作进入到键值数据库 21。除此以外, 在内存 21 中还维护着不同的世界状态, 除了包含有最新上链世界状态, 还维护有共识根世界状态以及正在共识的区块对应的世界状态。其中, 共识根世界状态是指上一次被共识确认的区块对应的世界状态; 正在共识的区块对应的世界状态, 顾名思义, 就是对于每个共识中的区块, 会为其创建相应的世界状态, 代表该区块内的交易执行过程中和执行后的状态数据的查询来源和更新缓存。由图 2c 可知, 正在共识的区块对应的世界状态均具有指向其父区块对应的世界状态的链式关系, 出块节点通过某个正在共识的区块对应的世界状态具有的链式关系, 就可以获取到其父区块对应的世界状态, 然后根据其父区块对应的世界状态具有的链式关系, 就可以继续获取到其父区块的父区块对应的世界状态。例如, 区块 $(n+2)$ a 对应的世界状态 $(n+2)$ a 具有指向其父区块 $n+1$ 对应的世界状态 $n+1$ 的链式关系, 出块节点 20 获取区块 $(n+2)$ a 对应的世界状态 $(n+2)$ a 后, 就可以获取世界状态 $(n+2)$ a 具有的链式关系, 然后根据该链式关系获取世界状态 $n+1$ 。当正在共识的区块的父区块为上一轮共识通过的区块时, 该正在共识的区块的世界状态的链式关系就会指向共识根世界状态, 因此正在共识的区块的世界状态必定是由共识根世界状态直接或间接延伸创建出来。而在共识根世界状态中, 则具有指向最新上链世界状态的链式关系。可以理解, 新的区块被共识后, 新的区块对应的世界状态就会被确定为共

识根世界状态，原来的共识根世界状态就可以被清空了，但是新区块上链后只需对最新上链世界状态进行更新，得到新的最新上链世界状态即可，其在内存中的存储地址无需更改。

可以理解，世界状态的链式关系可以由其前树世界状态指针和前树节点缓存指针体现，每个世界状态通过前世界状态指针指向前世界状态，并且通过树节点缓存的前树节点缓存指针指向前世界状态的树节点缓存，最前的世界状态的前世界状态指针为空和树节点缓存的前树节点缓存指针指向键值数据库的树节点存储，以此形成世界状态链。为便于理解，可以一并参见图 2d，图 2d 是本申请实施例提供的一种世界状态链的结构示意图。如图 2d 所示，在世界状态链中，第一个世界状态就为最新上链世界状态，最新上链世界状态没有前世界状态了，因此最新上链世界状态的前世界状态指针为空，此外，最新上链世界状态的前树节点缓存指针也无法指向前世界状态的树节点缓存，但是最新上链世界状态的前树节点缓存指针会指向树节点存储（例如，上述图 2c 所示的树节点存储 232）。世界状态链中的第二个世界状态为共识根世界状态，共识根世界状态的前世界状态为最新上链世界状态，其树节点缓存的前树节点缓存指针，指向最新上链世界状态的树节点缓存。世界状态链从第三个世界状态开始，就是共识中的区块对应的世界状态，其前世界状态指针和树节点缓存的前树节点缓存指针，分别指向该区块的父区块的世界状态和父区块的世界状态的树节点缓存。当然，也允许两个不同的共识中的区块的世界状态的前世界状态指针和树节点缓存的前树节点缓存指针，指向同一个前世界状态和其内的树节点缓存。换言之，世界状态链在共识根世界状态之后，可以出现分支。

对于正在共识的区块对应的世界状态，最终会被提交至最新上链世界状态，当正在共识的区块完成共识后，其对应的世界状态就会成为新的共识根世界状态，具有指向最新上链世界状态的链式关系，为便于理解本申请实施例对世界状态管理的具体过程，进一步地，请参见图 3，图 3 是本申请实施例提供的一种基于区块链的数据处理方法的流程示意图。其中，该方法可以由出块节点（例如，上述图 1 所对应实施例中的区块链节点系统 10 中的任一区块链节点）执行。以下将以本方法由出块节点执行为例进行说明，其中，该基于区块链的数据处理方法至少可以包括以下 S101-S104：

S101，生成初始区块，创建所述初始区块对应的第一世界状态。

每个区块链节点可以接收交易，然后将交易缓存至交易池中。区块链网络中可以根据共识机制来选择出块节点，当某个区块链节点被选作出块节点时，就会在内存中创建新区块，新区块中包含有其父区块哈希值，通过父区块哈希值，就可以查找其父区块，然后出块节点可以把交易池中的交易打包至新区块内，就得到了初始区块。此时，出块节点就可以创建初始区块对应的第一世界状态了。

在一种可能的实现方式中，创建初始区块对应的第一世界状态的过程可以是出块节点先获取其父区块对应的第二世界状态，然后基于其父区块的第二世界状态来进行世界状态初始化，得到第一世界状态。需要说明的是，如果初始区块的父区块已经被写入了区块链数据结构中，此时初始区块的父区块只可能是区块链中最新上链的区块，父区块对应的第二世界状态即为上一轮共识通过的世界状态，即此时父区块对应的第二世界状态为上述图 2c 所示的共识根世界状态；如果初始区块的父区块还未被写入区块链数据结构中，此时初

始区块的父区块应该正在共识中，其对应的世界状态即为上述图 2c 所示的正在共识的区块对应的世界状态，例如，世界状态 $n+1$ 。可以理解，第一世界状态和第二世界状态均位于世界状态链中，世界状态链中的第一个世界状态为最新上链世界状态，最新上链世界状态用于记录已上链区块对应的最新世界状态；第二个共识根世界状态为共识根世界状态，共识根世界状态具有指向最新上链世界状态的链式关系，换言之，第二世界状态可能是世界状态链中的第二个世界状态，也可能是第二个世界状态后的世界状态，第一世界状态在世界状态链中就位于第二世界状态之后。

在一种可能的实现方式中，创建初始区块对应的第一世界状态的一个可行实施方法为：创建初始区块对应的初始世界状态，该初始世界状态包含有第一前世界状态指针和第一树节点缓存，该第一树节点缓存包含有第一前树节点缓存指针；然后，将第一前世界状态指针指向初始区块的父区块对应的第二世界状态，将第一前树节点缓存指针指向第二世界状态包含的第二树节点缓存，得到初始区块对应的第一世界状态。需要说明的是，此时第一世界状态的结构组成依然可以参见上述图 2a 所示的世界状态 20，此时出块节点仅初始化了第一世界状态的第一前世界状态指针和第一前树节点缓存指针，脏账户地址集合、账户状态映射、世界状态树以及树节点映射等结构可以先初始化为空集合，也可以后续使用时在添加进第一世界状态中，本申请实施例在此不做限制。不过后续统一以世界状态在创建时就会按照如图 2a 所示的世界状态结构来生成世界状态，没有对应的数据就默认创建为空集合为例进行阐述。

S102，根据所述初始区块中的交易更新所述第一世界状态，得到初始更新状态，根据世界状态链对所述初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，所述更新后的第一世界状态包括干净现世界状态树。

初始区块中的交易执行会修改区块链中若干个账户包含的部分状态数据，出块节点可以逐步执行初始区块中的每笔交易，确定每笔交易执行后需要修改的状态数据，作为待更新数据，然后根据待更新数据来更新第一世界状态，最终得到初始更新状态。此时的初始更新状态的账户状态映射中就将包含有需要更新的账户的账户地址以及更新后的账户状态，同一账户的账户地址和其更新后的账户状态之间具有映射关系，后续出块节点在访问第一世界状态时，就可以根据账户的账户地址，在账户状态映射中快速获取该账户对应的更新后的账户状态。同时，初始更新状态的脏账户地址集合中会包含有在初始区块中的交易执行完毕后，状态数据发生更新的账户的账户地址。有了初始更新状态的脏账户地址集合和账户状态映射，就可以去构建初始更新状态对应的干净现世界状态树了。

此时的初始更新状态中的世界状态树为空，也就是说初始更新状态中的世界状态树还未被构建，此时出块节点会根据初始更新状态中的第一前世界状态指针查找到第二世界状态，先获取到第二世界状态的干净的世界状态树，作为干净前世界状态树，然后根据第一树节点缓存和现状态树根构建初始更新状态对应的第二世界状态树，将第二世界状态树添加到初始更新状态，得到过渡初始更新状态；对过渡初始更新状态进行全链脏账户状态提交处理，得到更新后的第一世界状态。

S103, 将所述干净现世界状态树对应的第一世界状态根写入所述初始区块, 得到待上链区块, 将所述共识区块发送至共识节点, 以使所述共识节点基于所述第一世界状态根对所述待上链区块进行共识处理, 得到共识结果。

干净现世界状态树对应的第一世界状态根是指干净现世界状态树的树根的哈希值, 此时第一世界状态根可以作为初始区块执行后区块链网络中所有账户包含的状态数据的状态快照。当第一世界状态根被写入初始区块后, 出块节点对区块的打包完成, 就得到了待上链区块, 随后, 出块节点会将该待上链区块经过 P2P (Peer to Peer, 点对点) 网络传送到区块链网络中的共识节点, 随后共识节点可以先构建该待上链区块对应的世界状态, 然后对构建的世界状态进行上述 S102 至 S103 的过程, 得到构建的世界状态对应的干净现世界状态树, 再获取其树根的哈希值作为校验世界状态根, 共识节点会将校验世界状态根和待上链区块中的第一世界状态根进行比较, 若相同, 则校验通过; 若不同, 则校验失败, 待上链区块的共识失败。

S104, 若所述共识结果为共识通过结果, 则将所述待上链区块写入区块链中, 通过所述世界状态链将所述更新后的第一世界状态逐级提交至最新上链世界状态, 将所述更新后的第一世界状态确定为新的共识根世界状态, 所述新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

倘若待上链区块共识通过, 出块节点和区块链网络中的其他区块链节点均会将待上链区块写入区块链中, 上述提到, 新的区块上链后, 最新上链世界状态也会更新, 此时出块节点需要将更新后的第一世界状态按照世界状态链的顺序, 逐级提交至最新上链世界状态。提交完成后, 更新后的第一世界状态将被确定为新的共识根世界状态, 此时新的共识根世界状态中的第一前世界状态指针会从其父区块对应的世界状态指向最新上链世界状态, 新的共识根世界状态中的第一前树节点缓存指针也会指向最新上链世界状态中的前树节点缓存。例如, 上述图 2c 所示的世界状态链中, 出块节点 22 可以在区块 n+1 共识通过后, 将世界状态 n+1 提交至共识根世界状态, 然后将提交后的共识根世界状态提交至最新上链世界状态, 得到提交后的最新上链世界状态。提交完成后, 世界状态 n+1 就会被确认为新的共识根世界状态, 原本具有的指向共识根世界状态的链式关系会被更新为指向提交后的最新上链世界状态的链式关系。提交后的最新上链世界状态的存储地址依然和最新上链世界状态相同, 提交后的最新上链世界状态用于表征区块 n+1 内的交易执行后的世界状态。可选的, 原本的共识根世界状态就可以删除了。

在一种可能的实现方式中, 当更新后的第一世界状态最终提交至最新上链世界状态后, 出块节点还需要将提交后的最新上链世界状态提交至树节点存储中, 因为树节点存储属于键值数据库, 因此也可以认为是提交至树节点数据库, 具体过程可以为: 对提交后的最新上链世界状态进行全链脏账户状态提交处理, 得到干净最新上链世界状态, 干净最新上链世界状态中的树节点缓存中包括第一树节点映射; 将第一树节点映射包含的树节点哈希和树节点更新到树节点数据库; 清空干净最新上链世界状态中的脏账户地址集合、账户状态映射和第一树节点映射。

为便于理解上述过程，请一并参见图 4a-图 4c，图 4a-图 4c 是本申请实施例提供的一种基于区块链的数据处理的场景示意图。如图 4a 所示，在出块节点 40（可以为上述图 2c 所示的出块节点 22）中，区块 41 是区块链 42 中最新被上链的区块，则最新上链世界状态 400 记录的就是区块 42 上链后区块链网络中所有账户对应的状态数据的视图，此时最新上链世界状态 400 中除了前世界状态指针为空以外，脏账户地址集合、账户状态映射以及树节点映射均为空集合，最新上链世界状态 400 中的世界状态树即是区块链此时所有状态数据对应的世界状态树，最新上链世界状态 400 的前树节点缓存指针指向树节点存储 43，可以理解，最新上链世界状态 400 的世界状态树中的每个树节点以及树节点对应的树节点哈希均存储在树节点存储 43 中。此外，出块节点 40 的内存中还包含有共识根世界状态 402，该共识根世界状态 402 其实就是区块 41 共识提交后对应的世界状态，其前世界状态即为最新上链世界状态 400，由图 4a 可知，共识根世界状态 402 的前世界状态指针指向最新上链世界状态 400，共识根世界状态 402 的前树节点缓存指针指向树节点缓存 4005。

为便于更好地阐述本方法，请一并参见图 4b。假设此时没有正在共识的区块。出块节点 40 在接收到出块命令后，就将生成以区块 41 为父区块的新区块 420，该新区块 420 中包含的父哈希即为区块 41 的哈希值，然后出块节点 40 将交易池中的部分交易按序打包进了新区块 420 中，就得到了初始区块 421，随后，出块节点 40 可以构建初始区块 421 对应的第一世界状态 402，第一世界状态 402 中的第一世界状态会指向初始区块 421 的父区块对应的第二世界状态，因为初始区块 421 的父区块为已上链区块，因此其对应的第二世界状态其实为共识根世界状态 401，因此第一世界状态 402 的第一前世界状态指针指向共识根世界状态 401，第一前树节点缓存指针指向共识根世界状态 401 的树节点缓存，此时，第一世界状态 402 的其他组成，例如脏账户地址集合、世界状态树等皆为空集合。然后，出块节点 40 会执行初始区块 421 中的交易，然后在执行初始区块 421 中的交易的过程中，对第一世界状态 402 进行更新，得到初始更新状态 403，其中，更新是指将需要更新的账户的账户地址以及更新后的账户状态写入第一世界状态 402 的账户状态映射中，将需要更新的账户的账户地址添加至第一世界状态 402 的脏账户地址集合中。例如，交易 1 执行后，会更改账户 A 的状态数据，因此账户 A 的账户状态会相应的更新，假设更新后的账户状态为账户状态 A，出块节点 40 会将账户 A 对应的账户地址 A 写入第一世界状态 402 的脏账户地址集合中，然后将账户地址 A 和账户状态 A 的映射关系添加至第一世界状态 402 的账户状态映射中。等到初始区块 421 中的所有交易执行完毕后，就得到初始更新状态 403 了。此时，初始更新状态 403 的世界状态树还没有被构建，然后出块节点 40 会构建初始更新状态 403 进行干净世界状态树构建处理，得到干净现世界状态树，最后，出块节点将干净现世界状态树对应的状态树根作为第一世界状态根，添加进初始区块 421 中，就得到待上链区块 422。随后，出块节点就可以将待上链区块 422 发送给区块链网络中的共识节点，让共识节点基于第一世界状态根对待上链区块 22 进行共识，得到共识结果。

请一并参见图 4c。图 4c 所示，当共识结果为共识通过结果，此时出块节点 40 中待上链区块 422 对应的世界状态为包含干净现世界状态树的更新后的第一世界状态 404，出块节点 40 会将更新后的第一世界状态 404 逐渐提交至最新上链世界状态，即将更新后的第一

世界状态 404 先提交至共识根世界状态 401，再将提交后的共识根世界状态提交至最新上链世界状态 400，最后将提交后的最新上链世界状态提交至树节点存储 43 中，待上链区块 422 也会被写入区块链 42 中，成为最新上链的区块，此时更新第一目标时间状态 404 会被作为新的共识根世界状态，出块节点 40 会将其第一前世界状态指针指向提交后的最新上链世界状态，将其第一前树节点缓存指针指向提交后的最新上链世界状态的树节点缓存。

采用本申请实施例提供的方法，在区块执行过程中更新状态数据时，不需要记录状态数据变更的旧值，而且对于执行失败的交易不需要进行状态数据的回滚操作，节省了系统开销。

请参见图 5，图 5 是本申请实施例提供的一种基于交易的第一世界状态更新方法的流程示意图。其中，该更新方法是上述图 3 所对应实施例中 S102 所述的根据初始区块中的交易更新第一世界状态，得到初始更新状态的一种可行实现方法。其中，该方法同样可以由出块节点（例如，上述图 1 所对应实施例中的区块链节点系统 10 中的任一区块链节点）执行。以下将以本方法由出块节点执行为例进行说明，其中，该第一世界状态更新方法至少可以包括以下 S201-S205：

S201，顺序遍历初始区块中的交易，获取所述初始区块中的第 k 个交易； k 为小于或等于 H 的正整数； H 为所述初始区块中的交易的总数量。

当初始区块中包含有多个交易时，出块节点会遍历初始区块，顺序获取第一个交易，然后执行第一个交易，再根据第一个交易的执行结果来更新第一世界状态，得到第一个提交世界状态，然后再获取第二个交易，执行第二个交易，再根据第二个交易的执行结果来更新第一个提交世界状态，得到第二个提交世界状态，以此类推，直至处理完初始区块中的所有交易，就得到初始更新状态了。为便于理解，以处理第 k 个交易为例进行说明。

S202，创建所述第 k 个交易对应的第 k 个交易世界状态；所述第 k 个交易世界状态具有指向第 $k-1$ 个提交世界状态的链式关系； k 为 1 时，所述第 $k-1$ 个提交世界状态为第一世界状态。

出块节点会先创建第 k 个交易对应的第 k 个交易世界状态，对于第 k 个交易世界状态来说，其前世界状态就为第 $k-1$ 个提交世界状态，此时第 k 个交易世界状态的创建可以参见上述 S101 中第一世界状态的创建过程，这里不再进行赘述。第 k 个交易世界状态具有指向第 $k-1$ 个提交世界状态的链式关系。可以理解，对于第一个交易来说，其前世界状态就为第一世界状态。

S203，执行所述第 k 个交易，在第 k 个交易执行的过程中对所述第 k 个交易世界状态进行状态更新处理，得到第 k 个更新交易世界状态。

由上述图 2b 所示的账户状态的结构可知，第 k 个交易世界状态可以包含有账户状态映射和脏账户地址集合，只是此时第 k 个交易世界状态的账户状态映射和脏账户地址集合皆为空。在第 k 个交易执行的过程中，出块节点在执行交易中的智能合约时，可以需要读取或者更新某个账户的状态数据，此时出块节点会确定待查询数据或者待更新数据。其中，待查询数据可以包含待查询账户地址和待查询状态数据键；待更新数据可以包括待更新账户地址、待更新状态数据键和待更新状态数据值。然后，出块节点可以基于待更新数据来

对第 k 个交易世界状态进行状态更新处理, 或者, 基于待查询数据来对第 k 个交易世界状态进行查询处理。

在一种可能的实现方式中, 执行第 k 个交易, 在第 k 个交易执行的过程中对第 k 个交易世界状态进行状态更新处理, 得到第 k 个更新交易世界状态的一个可行实施过程为: 执行第 k 个交易, 在第 k 个交易执行的过程中确定第 k 个交易对应的待更新数据, 即上述所说的待更新账户地址、待更新状态数据键和待更新状态数据值, 然后根据第 k 个交易世界状态获取待更新账户地址对应的账户状态, 作为待更新账户状态; 将待更新状态数据键和待更新状态数据值的映射关系更新到待更新状态数据映射中, 将待更新状态数据键插入到待更新脏状态数据键集合中, 得到现账户状态; 将待更新账户地址和现账户状态的映射关系更新到第 k 个交易世界状态包含的账户状态映射中, 将待更新账户地址插入到第 k 个交易世界状态包含的脏账户地址集合中, 得到第 k 个更新交易世界状态。

在一种可能的实现方式中, 根据第 k 个交易世界状态获取待更新账户地址对应的账户状态, 作为待更新账户状态的一个可行实施过程为: 根据待更新账户地址对第 k 个交易世界状态至最新上链世界状态进行账户状态映射顺序检索; 若在账户状态映射顺序检索的过程中检索到与待更新账户地址具有映射关系的账户状态, 则将最先检索到的与待更新账户地址具有映射关系的账户状态作为待更新账户状态; 若在账户状态映射顺序检索的过程中未检索到与待更新账户地址具有映射关系的账户状态, 则根据最新上链世界状态中的世界状态树获取待更新账户地址对应的待更新状态树根, 根据待更新账户地址和待更新状态树根建立待更新账户状态。具体来说, 出块节点可以根据第 k 个交易世界状态获取待更新账户状态的方法为, 先在第 k 个交易世界状态的账户状态映射中去查找是否存在与待更新账户地址具有映射关系的账户状态, 如果存在, 就将该账户状态作为待更新账户状态, 停止后续的查询检索; 如果不存在, 就根据第 k 个交易世界状态的前世界状态指针获取到第 k 个交易世界状态的前世界状态, 也就是第 k-1 个提交世界状态, 然后出块节点继续查找第 k-1 个提交世界状态的账户状态映射, 看是否存在与待更新账户地址具有映射关系的账户状态, 存在就直接获取, 停止后续的查询检索; 不存在就继续获取前世界状态, 以此类推, 直至出块节点一直查找到最新上链世界状态的账户状态映射, 倘若最新上链世界状态的账户状态映射中也不存在与待更新账户地址具有映射关系的账户状态, 上述可知, 最新上链世界状态没有前世界状态了, 此时出块节点就可以根据待更新账户地址, 从最新上链世界状态的世界状态树中去检索待更新地址对应的账户状态根, 然后在内存中新建一个账户状态, 并将新建账户状态的账户地址初始化为待更新账户地址, 将新建账户状态的账户状态根初始化为检索到的账户状态根, 然后将初始化后的新建账户状态作为待更新账户状态。

在一种可能的实现方式中, 执行第 k 个交易, 在第 k 个交易执行的过程中对第 k 个交易世界状态进行查询处理的一个可行实施过程为: 执行第 k 个交易, 在第 k 个交易执行的过程中确定第 k 个交易对应的待查询数据, 即上述所说的待查询账户地址和待查询状态数据键, 然后, 根据待查询账户地址在第 k 个交易世界状态的账户状态映射中进行检索, 如果检索得到待查询账户地址对应的账户状态, 就根据待查询状态数据键在检索得到的账户状态中继续检索, 如果检索得到待查询状态数据键对应的状态数据值, 就将该状态数据值

作为结果返回, 不再进行后续检索; 如果检索不到待查询状态数据键对应的状态数据值, 或者没有在第 k 个交易世界状态的账户状态映射中检索到待查询账户地址对应的账户状态, 就根据第 k 个交易世界状态的前世界状态指针判断是否存在前世界状态。如果第 k 个交易世界状态的前世界状态指针存在, 就根据前世界状态指针获取第 k 个交易世界状态, 即上述第 $k-1$ 个提交世界状态, 然后继续根据待查询地址和待查询状态数据键对第 $k-1$ 个提交世界状态进行检索, 如果在第 $k-1$ 个提交世界状态中检索到待查询状态数据键对应的状态数据值, 就停止后续的检索; 如果在第 $k-1$ 个提交世界状态中检索不到待查询状态数据键对应的状态数据值, 就继续根据第 $k-1$ 个提交世界状态前世界状态指针判断是否存在前世界状态。以此类推, 倘若一直检索到最新上链世界状态, 在最新上链世界状态中的账户状态映射中依然没有获取到待查询状态数据键对应的状态数据值, 就从最新上链世界状态的世界状态树中去检索待更新地址对应的账户状态根, 然后在内存中新建一个账户状态, 并将新建账户状态的账户地址初始化为待更新账户地址, 将新建账户状态的账户状态根初始化为检索到的账户状态根, 然后将初始化后的新建账户状态作为待查询账户状态。倘若待查询账户状态的账户状态根为空, 则确定待查询状态数据键对应的状态数据值为空, 停止后续的检索。倘若待查询账户状态的账户状态根不为空, 但是待查询账户状态的账户状态树为空, 则以最新上链世界状态的树节点缓存作为树节点查询来源, 以待查询账户状态的账户状态根作为状态树根, 在内存中新建账户状态树, 加入到待查询账户状态中; 然后在根据待查询状态数据键从待查询账户状态的账户状态树中检索出状态数据值, 检索结束。

S204, 若所述第 k 个交易执行成功, 则将所述第 k 个更新交易世界状态提交至所述第 $k-1$ 个提交世界状态, 得到第 k 个提交世界状态。

在一种可能的实现方式中, 若第 k 个交易执行成功, 则将第 k 个更新交易世界状态向第 $k-1$ 个提交世界状态进行提交, 得到第 k 个提交世界状态的一个可行实施过程为: 遍历第 k 个更新交易世界状态包含的脏账户地址集合中的账户地址, 顺序获取第 j 个账户地址; j 为小于或等于 I 的正整数; I 为第 k 个更新交易世界状态包含的脏账户地址集合中的账户地址的总数量; 从第 k 个更新交易世界状态包含的账户状态映射中获取第 j 个账户地址对应的现账户状态; 根据第 $k-1$ 个提交世界状态获取第 j 个账户地址的前账户状态; 将第 j 个账户地址对应的现账户状态提交至第 j 个账户地址的前账户状态, 将第 j 个账户地址插入到第 $k-1$ 个提交世界状态包含的脏账户地址集合, 得到第 j 个过渡提交世界状态; 当 j 等于 I 时, 清空第 k 个更新交易世界状态包含的脏账户地址集合和账户状态映射, 将第 I 个过渡提交世界状态作为第 k 个提交世界状态。其中, 根据第 $k-1$ 个提交世界状态获取第 j 个账户地址的前账户状态的过程, 可以为: 根据第 j 个账户地址在第 $k-1$ 个提交世界状态中的账户状态映射中进行检索, 如果检索得到与第 j 个账户地址具有映射关系的账户状态, 则将其作为参考账户状态, 然后在内存中新建账户状态, 并根据参考账户状态的账户地址和账户状态根, 来初始化刚新建的账户状态的账户地址和账户状态根, 把刚新建的账户状态作为第 j 个账户地址的前账户状态; 如果没有检索到与第 j 个账户地址具有映射关系的账户状态, 就继续在第 $k-1$ 个提交世界状态的前世界状态的账户状态映射进行检索, 继续上述检索过程, 倘若一直检索到最新上链世界状态的账户状态映射, 都没有检索到与第 j 个账户地址

具有映射关系的账户状态，就在最新上链世界状态的世界状态树中进行检索，得到第 j 个账户地址对应的账户状态根，然后在内存中新建账户状态，并根据第 j 个账户地址和从世界状态树中获取的账户状态根，来初始化刚新建的账户状态的账户地址和账户状态根，把刚新建的账户状态作为第 j 个账户地址的前账户状态。

在一种可能的实现方式中，第 j 个账户地址对应的现账户状态包含第一状态数据映射，第 j 个账户地址的前账户状态包含第二状态数据映射，将第 j 个账户地址对应的现账户状态提交至第 j 个账户地址的前账户状态，将第 j 个账户地址插入到第 $k-1$ 个提交世界状态包含的脏账户地址集合，得到第 j 个过渡提交世界状态的一个可行实施过程为：将第一状态数据映射中的状态数据键和状态数据值的映射关系，更新到第二状态数据映射中；将第一状态数据映射中的状态数据键插入到第 j 个账户地址的前账户状态包含的脏状态数据键集合中，得到第 j 个过渡提交世界状态。

S205，当 k 等于 H 时，将第 H 个提交世界状态作为初始更新状态。

通常情况下，初始更新状态中将包含有初始区块内所有交易执行完后，需要更新的账户地址以及该账户地址更新后的账户状态，初始更新状态的脏账户地址集合中将包含有所有需要更新的账户地址。

请参见图 6，图 6 是本申请实施例提供的一种干净世界状态树构建方法的流程示意图。其中，该干净世界状态树构建方法是上述图 3 所对应实施例中的 S102 所述的根据世界状态链对初始更新状态进行构建干净状态树处理，得到包含干净现世界状态树的更新后的第一世界状态的一种可行实现方法。其中，该方法同样可以由出块节点（例如，上述图 1 所对应实施例中的区块链节点系统 10 中的任一区块链节点）执行。以下将以本方法由出块节点执行为例进行说明，其中，该干净世界状态树构建方法至少可以包括以下 S301-S306：

S301，对初始更新状态进行世界状态树查询。

出块节点管理的所有世界状态，都可以构建对其对应的干净的世界状态树，该干净世界状态树构建方法适用于所有世界状态，而某些世界状态中可能已经包含有世界状态树了，因此在该干净世界状态树构建方法中，需要先确定需要构建干净世界状态树的世界状态中是否包含有世界状态树。当出块节点针对初始更新状态进行构建干净世界状态树处理时，出块节点会对初始更新状态进行世界状态树查询。

S302，若所述初始更新状态包含第一世界状态树，则对所述初始更新状态进行全链脏账户状态提交处理，得到所述更新后的第一世界状态。

倘若初始更新状态中包含有第一世界状态树，则出块节点将直接对初始更新状态进行全链脏账户状态提交处理，得到包含干净现世界状态树的更新后的第一世界状态。该提交过程可以参考下述 S306，这里先不进行展开描述。

S303，若所述初始更新状态未包含所述第一世界状态树，则根据所述第一前世界状态指针查找第二世界状态。

倘若初始更新状态中没有包含第一世界状态树，则出块节点可以根据世界状态链逐个向前进行查询，依赖世界状态链中之前的世界状态的干净世界状态来构建初始更新状态的第二世界状态树。首先，出块节点会先根据第一前世界状态指针获取到初始更新状态的前

世界状态，也就是第二世界状态，然后先获取第二世界状态对应的干净现世界状态树，将其作为干净前世界状态树。

S304，根据世界状态链对所述第二世界状态进行构建干净状态树处理，得到干净前世界状态树，将所述干净前世界状态树的前状态树根作为现状态树根。

出块节点会对第二世界状态进行构建干净状态树处理，即对第二世界状态进行S301-S306的处理过程，得到第二世界状态对应的干净现世界状态树，然后将其作为初始更新状态的干净前世界状态树。然后，出块节点会将干净前世界状态树的前状态树根作为现状态树根。

S305，根据第一树节点缓存和所述现状态树根构建所述初始更新状态对应的第二世界状态树，将所述第二世界状态树添加到所述初始更新状态，得到过渡初始更新状态。

出块节点会将第一树节点缓存作为树节点查询来源，然后基于现状态树根构建初始更新状态对应的第二世界状态树，此时初始更新状态的脏账户地址集合中的账户地址对应的更新账户状态还未被提交，因此需要将其进行提交，才能得到干净的第二世界状态树，即干净现世界状态树。

S306，对所述过渡初始更新状态进行全链脏账户状态提交处理，得到所述更新后的第一世界状态。

通常情况下，无论是获取到了初始更新状态对应的第一世界状态树或者第二世界状态树后，出块节点均会进行全链脏账户状态提交处理。以包含第二世界状态树的过渡初始更新状态为例，出块节点会顺序遍历世界状态链，获取第 x 个世界状态包含的脏账户地址集合，作为第 x 个脏账户地址集合； x 为小于或等于 Z 的正整数； Z 为过渡初始更新状态在世界状态链中的顺序序号；遍历第 x 个脏账户地址集合中的账户地址，顺序获取第 m 个账户地址； m 为小于或等于第 x 个脏账户地址集合中账户地址的总数量的正整数；获取第 m 个账户地址对应的现账户状态和前账户状态；根据第 m 个账户地址的前账户状态对第 m 个账户地址的现账户状态进行账户脏状态数据提交处理；当第 x 个脏账户地址集合中每个脏账户地址的现账户状态均进行账户脏状态数据提交处理后，得到第 x 个提交后的世界状态；将第 x 个脏账户地址集合中每个脏账户地址添加至第 $x-1$ 个链上脏账户地址集合，得到第 x 个链上脏账户地址集合； x 为1时，第 $x-1$ 个链上脏账户地址集合为空集合；当 x 等于 Z 时，将第 x 个链上脏账户地址集合作为全链脏账户地址集合，将第 x 个提交后的世界状态作为提交后的过渡初始更新状态。

在一种可能的实现方式中，根据第 m 个账户地址的前账户状态对第 m 个账户地址的现账户状态进行账户脏状态数据提交处理的一个可行实施过程为：对第 m 个账户地址的现账户状态进行账户状态树查询；若第 m 个账户地址的现账户状态不包含有账户状态树，或第 m 个账户地址的现账户状态的账户状态根不等于第 m 个账户地址的前账户状态的账户状态根，则根据第 m 个账户地址的现账户状态的树节点缓存和第 m 个账户地址的前账户状态的账户状态根建立指定账户状态树，将指定账户状态树添加至第 m 个账户地址的现账户状态，将第 m 个账户地址的现账户状态的账户状态根更新为第 m 个账户地址的前账户状态的账户状态根，得到指定账户状态；若第 m 个账户地址的现账户状态包含有账户状态树，且

第 m 个账户地址的现账户状态的账户状态根等于第 m 个账户地址的前账户状态的账户状态根，则将第 m 个账户地址的现账户状态作为指定账户状态；若指定账户状态的第一脏状态数据键集合不为空，则从指定账户状态的状态数据映射中获取第一脏状态数据键集合中脏状态数据键对应的状态数据值，将第一脏状态数据键集合中脏状态数据键和第一脏状态数据键集合中脏状态数据键对应的状态数据值更新到指定账户状态的账户状态树，得到包含更新账户状态树的指定账户状态；将包含更新账户状态树的指定账户状态的账户状态根更新为将更新账户状态树的状态树根；清空包含更新账户状态树的指定账户状态的脏状态数据键集合；将更新账户状态树的更新树节点哈希和更新树节点的关联关系存储至第 x 个世界状态。

在一种可能的实现方式中，上述过程的实际实现可以为：出块节点可以将过渡初始更新状态作为现世界状态，然后针对现世界状态执行“提交全链脏账户的状态数据”的方法，该方法的具体步骤为：根据现世界状态的前世界状态指针判断是否存在前世界状态，如果存在前世界状态则把前世界状态视为现世界状态继续执行“提交全链脏账户的状态数据”的方法，将得到的全链脏账户地址集合作为初始全链脏账户地址集合；如果不存在前世界状态则在内存中新建为空的初始全链脏账户地址集合。在得到初始全链脏账户地址集合后，出块节点可以遍历现世界状态的脏账户地址集合，对于每个账户地址，分别执行以下操作：根据账户地址从账户状态映射上检索出账户状态，作为现账户状态；根据现世界状态的前世界状态指针判断是否存在前世界状态，如果存在前世界状态则根据账户地址在前世界状态中获取前账户状态，如果不存在前世界状态则把现账户状态同时作为前账户状态；把前账户状态的账户状态根作为前账户状态根，并根据现世界状态的树节点缓存，现账户状态作为指定账户状态执行“提交单个账户的状态数据”的方法；然后把账户地址插入到处理全链脏账户地址集合中；当脏账户地址集合中的所有账户地址均完成上述操作后，就得到了全链脏账户地址集合和提交后的过渡初始更新状态。

在一种可能的实现方式中，把前账户状态的账户状态根作为前账户状态根，并根据现世界状态的树节点缓存，现账户状态作为指定账户状态执行“提交单个账户的状态数据”的方法的具体步骤为：如果指定账户状态的账户状态树不为空并且指定账户状态的账户状态根等于前账户状态根则跳过本步骤，否则以树节点缓存作为树节点查询来源，以前账户状态根作为状态树根，在内存中新建账户状态树，加入到指定账户状态中，再把指定账户状态的账户状态根更新为前账户状态根，继续以下步骤；如果指定账户状态的第一脏状态数据键集合为空则返回且本流程结束，如果指定账户状态的第一脏状态数据键集合不为空则继续以下步骤；遍历指定账户状态的第一脏状态数据键集合，根据每个状态数据键，从指定账户状态的状态数据映射中检索出状态数据值，把每对状态数据键和状态数据值更新到指定账户状态的账户状态树，继续以下步骤；把树节点缓存作为树节点更新缓存，对指定账户状态的账户状态树进行提交操作，继续以下步骤；获取指定账户状态的账户状态树的状态根，并更新到指定账户状态的账户状态根；清空指定账户状态的第一脏状态数据键集合。

通过本申请实施例提供的方法，通过构建世界状态链来同时维护不同区块高度或者同一区块高度不同分支的世界状态，兼容需要多区块同时共识的共识机制，可以保证区块链系统的状态数据的一致性

请参见图 7，图 7 是本申请实施例提供的一种基于区块链的数据处理装置的结构示意图。该数据处理装置可以是运行于计算机设备的一个计算机程序（包括程序代码），例如该数据处理装置为一个应用软件；该装置可以用于执行本申请实施例提供的数据处理方法中的相应步骤。如图 7 所示，该数据处理装置 1 可以包括：初始化模块 11、状态创建模块 12、第一状态更新模块 13、第二状态更新模块 14、共识模块 15 以及状态提交模块 16。

初始化模块 11，用于生成初始区块；

状态创建模块 12，用于创建初始区块对应的第一世界状态；第一世界状态具有指向第二世界状态的链式关系，第二世界状态为初始区块的父区块对应的世界状态；第一世界状态和第二世界状态存在于世界状态链中；若父区块为已上链区块中具有最大区块高度的区块，则第二世界状态为共识根世界状态；世界状态链中的第一个世界状态为最新上链世界状态；共识根世界状态具有指向最新上链世界状态的链式关系；最新上链世界状态用于记录已上链区块对应的最新世界状态；

第一状态更新模块 13，用于根据初始区块中的交易更新第一世界状态，得到初始更新状态；

第二状态更新模块 14，用于根据世界状态链对初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，更新后的第一世界状态包括干净现世界状态树；

共识模块 15，用于将干净现世界状态树对应的第一世界状态根写入初始区块，得到待上链区块，将共识区块发送至共识节点，以使共识节点基于第一世界状态根对待上链区块进行共识处理，得到共识结果；

状态提交模块 16，用于若共识结果为共识通过结果，则将待上链区块写入区块链中，通过世界状态链将更新后的第一世界状态逐级提交至最新上链世界状态，将更新后的第一世界状态确定为新的共识根世界状态，新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

其中，初始化模块 11、状态创建模块 12、第一状态更新模块 13、第二状态更新模块 14、共识模块 15 以及状态提交模块 16 的具体实现方式可以参见上述图 3 所对应实施例中对 S101-S104 的描述，这里不再进行赘述。

其中，状态创建模块 12，包括：创建单元 121 以及指向单元 122。

创建单元 121，用于创建初始区块对应的初始世界状态；初始世界状态包含有第一前世界状态指针和第一树节点缓存；第一树节点缓存包含有第一前树节点缓存指针；

指向单元 122，用于将第一前世界状态指针指向初始区块的父区块对应的第二世界状态，将第一前树节点缓存指针指向第二世界状态包含的第二树节点缓存，得到初始区块对应的第一世界状态。

其中，创建单元 121 以及指向单元 122 的具体实现方式可以参见上述图 3 所对应实施例中对 S101 的描述，这里不再进行赘述。

其中，第一状态更新模块 13，包括：第一交易获取单元 131、交易状态创建单元 132、交易状态更新单元 133、交易状态提交单元 134 以及世界状态确定单元 135。

第一交易获取单元 131，用于顺序遍历初始区块中的交易，获取初始区块中的第 k 个交易； k 为小于或等于 H 的正整数； H 为初始区块中的交易的总数量；

交易状态创建单元 132，用于创建第 k 个交易对应的第 k 个交易世界状态；第 k 个交易世界状态具有指向第 $k-1$ 个提交世界状态的链式关系； k 为 1 时，第 $k-1$ 个提交世界状态为第一世界状态；

交易状态更新单元 133，用于执行第 k 个交易，在第 k 个交易执行的过程中对第 k 个交易世界状态进行状态更新处理，得到第 k 个更新交易世界状态；

交易状态提交单元 134，用于若第 k 个交易执行成功，则将第 k 个更新交易世界状态提交至第 $k-1$ 个提交世界状态，得到第 k 个提交世界状态；

世界状态确定单元 135，用于当 k 等于 H 时，将第 H 个提交世界状态作为初始更新状态。

其中，第一交易获取单元 131、交易状态创建单元 132、交易状态更新单元 133、交易状态提交单元 134 以及世界状态确定单元 135 的具体实现方式可以参见上述图 5 所对应实施例中对 S201-S205 的描述，这里不再进行赘述。

其中，第 k 个交易世界状态包含有账户状态映射和脏账户地址集合；

交易状态更新单元 133，包括：执行子单元 1331、账户状态获取子单元 1332、第一映射更新子单元 1333 以及第二映射更新单元 1334。

执行子单元 1331，用于执行第 k 个交易，在第 k 个交易执行的过程中确定第 k 个交易对应的待更新数据；待更新数据包括待更新账户地址、待更新状态数据键和待更新状态数据值；

账户状态获取子单元 1332，用于根据第 k 个交易世界状态获取待更新账户地址对应的账户状态，作为待更新账户状态；待更新账户状态包含有待更新状态数据映射和待更新脏状态数据键集合；

第一映射更新子单元 1333，用于将待更新状态数据键和待更新状态数据值的映射关系更新到待更新状态数据映射中，将待更新状态数据键插入到待更新脏状态数据键集合中，得到现账户状态；

第二映射更新单元 1334，用于将待更新账户地址和现账户状态的映射关系更新到第 k 个交易世界状态包含的账户状态映射中，将待更新账户地址插入到第 k 个交易世界状态包含的脏账户地址集合中，得到第 k 个更新交易世界状态。

其中，账户状态获取子单元 1332，具体用于根据待更新账户地址对第 k 个交易世界状态至最新上链世界状态进行账户状态映射顺序检索；若在账户状态映射顺序检索的过程中检索到与待更新账户地址具有映射关系的账户状态，则将最先检索到的与待更新账户地址具有映射关系的账户状态作为待更新账户状态；若在账户状态映射顺序检索的过程中未检索到与待更新账户地址具有映射关系的账户状态，则根据最新上链世界状态中的世界状态

树获取待更新账户地址对应的待更新状态树根，根据待更新账户地址和待更新状态树根建立待更新账户状态。

其中，执行子单元 1331、账户状态获取子单元 1332、第一映射更新子单元 1333 以及第二映射更新单元 1334 的具体实现方式可以参见上述图 5 所对应实施例中对 S203 的描述，这里不再进行赘述。

其中，交易状态提交单元 134，包括：地址获取子单元 1341、现账户状态获取子单元 1342、前账户状态获取子单元 1343、世界状态提交子单元 1344 以及世界状态确定子单元 1345。

地址获取子单元 1341，用于遍历第 k 个更新交易世界状态包含的脏账户地址集合中的账户地址，顺序获取第 j 个账户地址；j 为小于或等于 I 的正整数；I 为第 k 个更新交易世界状态包含的脏账户地址集合中的账户地址的总数量；

现账户状态获取子单元 1342，用于从第 k 个更新交易世界状态包含的账户状态映射中获取第 j 个账户地址对应的现账户状态；

前账户状态获取子单元 1343，用于根据第 k-1 个提交世界状态获取第 j 个账户地址的前账户状态；

世界状态提交子单元 1344，用于将第 j 个账户地址对应的现账户状态提交至第 j 个账户地址的前账户状态，将第 j 个账户地址插入到第 k-1 个提交世界状态包含的脏账户地址集合，得到第 j 个过渡提交世界状态；

世界状态确定子单元 1345，用于当 j 等于 I 时，清空第 k 个更新交易世界状态包含的脏账户地址集合和账户状态映射，将第 I 个过渡提交世界状态作为第 k 个提交世界状态。

其中，第 j 个账户地址对应的现账户状态包含第一状态数据映射，第 j 个账户地址的前账户状态包含第二状态数据映射，世界状态提交子单元 1345，具体用于将第一状态数据映射中状态数据键和状态数据值的映射关系，更新到第二状态数据映射中；将第一状态数据映射中的状态数据键插入到第 j 个账户地址的前账户状态包含的脏状态数据键集合中，得到第 j 个过渡提交世界状态。

其中，地址获取子单元 1341、现账户状态获取子单元 1342、前账户状态获取子单元 1343、世界状态提交子单元 1344 以及世界状态确定子单元 1345 的具体实现方式可以参见上述图 5 所对应实施例中对 S204 的描述，这里不再进行赘述。

其中，第二状态更新模块 14，包括：树查询单元 141、第一查询处理单元 142、第二查询处理单元 143、第一树构建单元 144 以及第二树构建单元 145。

树查询单元 141，用于对初始更新状态进行世界状态树查询；

第一查询处理单元 142，用于若初始更新状态包含第一世界状态树，则对初始更新状态进行全链脏账户状态提交处理，得到更新后的第一世界状态；

第二查询处理单元 143，用于若初始更新状态未包含第一世界状态树，则根据第一前世界状态指针查找第二世界状态；

第一树构建单元 144，用于根据世界状态链对第二世界状态进行构建干净状态树处理，得到干净前世界状态树；

第二树构建单元 145, 用于将干净前世界状态树的前状态树根作为现状态树根;

第二树构建单元 145, 还用于根据第一树节点缓存和现状态树根构建初始更新状态对应的第二世界状态树, 将第二世界状态树添加到初始更新状态, 得到过渡初始更新状态;

脏状态提交单元 146, 用于对过渡初始更新状态进行全链脏账户状态提交处理, 得到更新后的第一世界状态。

其中, 树查询单元 141、第一查询处理单元 142、第二查询处理单元 143、第一树构建单元 144 以及第二树构建单元 145 的具体实现方式可以参见上述图 6 所对应实施例中对 S301-S306 的描述, 这里不再进行赘述。

其中, 脏状态提交单元 146, 包括: 脏数据提交子单元 1461、根获取子单元 1462、树更新子单元 1463 以及树提交子单元 1464。

脏数据提交子单元 1461, 用于对过渡初始更新状态进行全链脏状态数据提交处理, 得到提交后的过渡初始更新状态和全链脏账户地址集合;

根获取子单元 1462, 用于根据提交后的过渡初始更新状态获取全链脏账户地址集合中的每个全链脏账户地址对应的账户状态根;

树更新子单元 1463, 用于将每个全链脏账户地址和每个全链脏账户地址对应的账户状态根更新到第二世界状态树, 得到干净现世界状态树;

树提交子单元 1464, 用于将干净现世界状态树的更新树节点哈希和更新树节点的关联关系存储至提交后的过渡初始更新状态中的树节点缓存的树节点映射中, 得到更新后的第一世界状态。

其中, 脏数据提交子单元 1461, 具体用于顺序遍历世界状态链, 获取第 x 个世界状态包含的脏账户地址集合, 作为第 x 个脏账户地址集合; x 为小于或等于 Z 的正整数; Z 为过渡初始更新状态在世界状态链中的顺序序数; 遍历第 x 个脏账户地址集合中的账户地址, 顺序获取第 m 个账户地址; m 为小于或等于第 x 个脏账户地址集合中的账户地址的总数量的正整数; 获取第 m 个账户地址对应的现账户状态和前账户状态; 根据第 m 个账户地址的前账户状态对第 m 个账户地址的现账户状态进行账户脏状态数据提交处理; 当第 x 个脏账户地址集合中每个脏账户地址的现账户状态均进行账户脏状态数据提交处理后, 得到第 x 个提交后的世界状态; 将第 x 个脏账户地址集合中每个脏账户地址添加至第 $x-1$ 个链上脏账户地址集合, 得到第 x 个链上脏账户地址集合; x 为 1 时, 第 $x-1$ 个链上脏账户地址集合为空集合; 当 x 等于 Z 时, 将第 x 个链上脏账户地址集合作为全链脏账户地址集合, 将第 x 个提交后的世界状态作为提交后的过渡初始更新状态。

其中, 脏数据提交子单元 1461, 具体还用于对第 m 个账户地址的现账户状态进行账户状态树查询; 若第 m 个账户地址的现账户状态不包含有账户状态树, 或第 m 个账户地址的现账户状态的账户状态根不等于第 m 个账户地址的前账户状态的账户状态根, 则根据第 m 个账户地址的现账户状态的树节点缓存和第 m 个账户地址的前账户状态的账户状态根建立指定账户状态树, 将指定账户状态树添加至第 m 个账户地址的现账户状态, 将第 m 个账户地址的现账户状态的账户状态根更新为第 m 个账户地址的前账户状态的账户状态根, 得到指定账户状态; 若第 m 个账户地址的现账户状态包含有账户状态树, 且第 m 个账户地址的

现账户状态的账户状态根等于第 m 个账户地址的前账户状态的账户状态根, 则将第 m 个账户地址的现账户状态作为指定账户状态; 若指定账户状态的第一脏状态数据键集合不为空, 则从指定账户状态的状态数据映射中获取第一脏状态数据键集合中脏状态数据键对应的状态数据值, 将第一脏状态数据键集合中脏状态数据键和第一脏状态数据键集合中脏状态数据键对应的状态数据值更新到指定账户状态的账户状态树, 得到包含更新账户状态树的指定账户状态; 将包含更新账户状态树的指定账户状态的账户状态根更新为将更新账户状态树的状态树根; 清空包含更新账户状态树的指定账户状态的脏状态数据键集合; 将更新账户状态树的更新树节点哈希和更新树节点的关联关系存储至第 x 个世界状态。

其中, 脏数据提交子单元 1461、根获取子单元 1462、树更新子单元 1463 以及树提交子单元 1464 的具体实现方式可以参见上述图 3 所对应实施例中对 S306 的描述, 这里不再进行赘述。

其中, 上述数据处理装置 1, 还包括: 提交模块 17、更新模块 18 以及清空模块 19。

提交模块 17, 用于对提交后的最新上链世界状态进行全链脏账户状态提交处理, 得到干净最新上链世界状态, 干净最新上链世界状态中的树节点缓存中包括第一树节点映射;

更新模块 18, 用于将第一树节点映射包含的树节点哈希和树节点更新到树节点数据库;

清空模块 19, 用于清空干净最新上链世界状态中的脏账户地址集合、账户状态映射和第一树节点映射。

其中, 提交模块 17、更新模块 18 以及清空模块 19 的具体实现方式可以参见上述图 3 所对应实施例中的可选描述, 这里不再进行赘述。

请参见图 8, 图 8 是本申请实施例提供的一种计算机设备的结构示意图。如图 8 所示, 上述图 7 所对应实施例中的数据处理装置 1 可以应用于计算机设备 1000, 该计算机设备 1000 可以包括: 处理器 1001, 网络接口 1004 和存储器 1005, 此外, 上述计算机设备 1000 还可以包括: 用户接口 1003, 和至少一个通信总线 1002。其中, 通信总线 1002 用于实现这些组件之间的连接通信。其中, 用户接口 1003 可以包括显示屏(Display)、键盘(Keyboard), 可选用户接口 1003 还可以包括标准的有线接口、无线接口。网络接口 1004 可选的可以包括标准的有线接口、无线接口(如 WI-FI 接口)。存储器 1005 可以是高速 RAM 存储器, 也可以是非不稳定的存储器(non-volatile memory), 例如至少一个磁盘存储器。存储器 1005 可选的还可以是至少一个位于远离前述处理器 1001 的存储装置。如图 8 所示, 作为一种计算机可读存储介质的存储器 1005 中可以包括操作系统、网络通信模块、用户接口模块以及设备控制应用程序。

在如图 8 所示的计算机设备 1000 中, 网络接口 1004 可提供网络通讯网元; 而用户接口 1003 主要用于为用户提供输入的接口; 而处理器 1001 可以用于调用存储器 1005 中存储的设备控制应用程序, 以实现:

生成初始区块, 创建初始区块对应的第一世界状态; 第一世界状态具有指向第二世界状态的链式关系, 第二世界状态为初始区块的父区块对应的世界状态; 第一世界状态和第二世界状态存在于世界状态链中; 若父区块为已上链区块中具有最大区块高度的区块, 则第二世界状态为共识根世界状态; 世界状态链中的第一个世界状态为最新上链世界状态;

共识根世界状态具有指向最新上链世界状态的链式关系；最新上链世界状态用于记录已上链区块对应的最新世界状态；

根据初始区块中的交易更新第一世界状态，得到初始更新状态，根据世界状态链对初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，更新后的第一世界状态包括干净现世界状态树；

将干净现世界状态树对应的第一世界状态根写入初始区块，得到待上链区块，将待上链区块发送至共识节点，以使共识节点基于第一世界状态根对待上链区块进行共识处理，得到共识结果；

若共识结果为共识通过结果，则将待上链区块写入区块链中，通过世界状态链将更新后的第一世界状态逐级提交至最新上链世界状态，将更新后的第一世界状态确定为新的共识根世界状态，新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

应当理解，本申请实施例中所描述的计算机设备 1000 可执行前文图 3 任一个所对应实施例中对该数据处理方法的描述，在此不再赘述。另外，对采用相同方法的有益效果描述，也不再进行赘述。

此外，这里需要指出的是：本申请实施例还提供了一种计算机可读存储介质，且上述计算机可读存储介质中存储有前文提及的数据处理装置 1 所执行的计算机程序，且上述计算机程序包括程序指令，当上述处理器执行上述程序指令时，能够执行前文图 3 任一个所对应实施例中对上述数据处理方法的描述，因此，这里将不再进行赘述。另外，对采用相同方法的有益效果描述，也不再进行赘述。对于本申请所涉及的计算机可读存储介质实施例中未披露的技术细节，请参照本申请方法实施例的描述。

上述计算机可读存储介质可以是前述任一实施例提供的数据处理装置或者上述计算机设备的内部存储单元，例如计算机设备的硬盘或内存。该计算机可读存储介质也可以是该计算机设备的外部存储设备，例如该计算机设备上配备的插接式硬盘，智能存储卡（smart media card，SMC），安全数字（secure digital，SD）卡，闪存卡（flash card）等。进一步地，该计算机可读存储介质还可以既包括该计算机设备的内部存储单元也包括外部存储设备。该计算机可读存储介质用于存储该计算机程序以及该计算机设备所需的其他程序和数据。该计算机可读存储介质还可以用于暂时地存储已经输出或者将要输出的数据。

此外，这里需要指出的是：本申请实施例还提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括计算机指令，该计算机指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机指令，处理器执行该计算机指令，使得该计算机设备执行前文任一个所对应实施例提供的方法。

本申请实施例的说明书和权利要求书及附图中的术语“第一”、“第二”等是用于区别不同对象，而非用于描述特定顺序。此外，术语“包括”以及它们任何变形，意图在于覆盖不排他的包含。例如包含了一系列步骤或单元的过程、方法、装置、产品或设备没有限定于已列出的步骤或模块，而是可选地还包括没有列出的步骤或模块，或可选地还包括对于这些过程、方法、装置、产品或设备固有的其他步骤单元。

本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照网元一般性地描述了各示例的组成及步骤。这些网元究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的网元，但是这种实现不应认为超出本申请的范围。

以上所揭露的仅为本申请较佳实施例而已，当然不能以此来限定本申请之权利范围，因此依本申请权利要求所作的等同变化，仍属本申请所涵盖的范围。

权 利 要 求

1、一种基于区块链的数据处理方法，所述方法由计算机设备执行，所述方法包括：

生成初始区块，创建所述初始区块对应的第一世界状态；所述第一世界状态具有指向第二世界状态的链式关系，所述第二世界状态为所述初始区块的父区块对应的世界状态；所述第一世界状态和所述第二世界状态存在于世界状态链中；若所述父区块为已上链区块中具有最大区块高度的区块，则所述第二世界状态为共识根世界状态；所述世界状态链中的第一个世界状态为最新上链世界状态；所述共识根世界状态具有指向最新上链世界状态的链式关系；所述最新上链世界状态用于记录所述已上链区块对应的最新世界状态；

根据所述初始区块中的交易更新所述第一世界状态，得到初始更新状态，根据所述世界状态链对所述初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，所述更新后的第一世界状态包括干净现世界状态树；

将所述干净现世界状态树对应的第一世界状态根写入所述初始区块，得到待上链区块，将所述待上链区块发送至共识节点，以使所述共识节点基于所述第一世界状态根对所述待上链区块进行共识处理，得到共识结果；

若所述共识结果为共识通过结果，则将所述待上链区块写入区块链中，通过所述世界状态链将所述更新后的第一世界状态逐级提交至所述最新上链世界状态，将所述更新后的第一世界状态确定为新的共识根世界状态，所述新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

2、根据权利要求1所述的方法，所述创建所述初始区块对应的第一世界状态，包括：

创建所述初始区块对应的初始世界状态；所述初始世界状态包含有第一前世界状态指针和第一树节点缓存；所述第一树节点缓存包含有第一前树节点缓存指针；

将所述第一前世界状态指针指向所述初始区块的父区块对应的第二世界状态，将所述第一前树节点缓存指针指向所述第二世界状态包含的第二树节点缓存，得到所述初始区块对应的第一世界状态。

3、根据权利要求1所述的方法，所述根据所述初始区块中的交易更新所述第一世界状态，得到初始更新状态，包括：

顺序遍历所述初始区块中的交易，获取所述初始区块中的第k个交易；k为小于或等于H的正整数；H为所述初始区块中的交易的总数量；

创建所述第k个交易对应的第k个交易世界状态；所述第k个交易世界状态具有指向第k-1个提交世界状态的链式关系；k为1时，所述第k-1个提交世界状态为所述第一世界状态；

执行所述第k个交易，在所述第k个交易执行的过程中对所述第k个交易世界状态进行状态更新处理，得到第k个更新交易世界状态；

若所述第k个交易执行成功，则将所述第k个更新交易世界状态提交至所述第k-1个提交世界状态，得到第k个提交世界状态；

当k等于H时，将第H个提交世界状态作为所述初始更新状态。

4、根据权利要求3所述的方法，所述第k个交易世界状态包含有账户状态映射和脏账户地址集合，所述执行所述第k个交易，在所述第k个交易执行的过程中对所述第k个交易世界状态进行状态更新处理，得到第k个更新交易世界状态，包括：

执行所述第k个交易，在所述第k个交易执行的过程中确定所述第k个交易对应的待更新数据；所述待更新数据包括待更新账户地址、待更新状态数据键和待更新状态数据值；

根据所述第k个交易世界状态获取所述待更新账户地址对应的账户状态，作为待更新账户状态；所述待更新账户状态包含有待更新状态数据映射和待更新脏状态数据键集合；

将所述待更新状态数据键和所述待更新状态数据值的映射关系更新到所述待更新状态数据映射中，将所述待更新状态数据键插入到所述待更新脏状态数据键集合中，得到现账户状态；

将所述待更新账户地址和所述现账户状态的映射关系更新到所述第k个交易世界状态包含的账户状态映射中，将所述待更新账户地址插入到所述第k个交易世界状态包含的脏账户地址集合中，得到所述第k个更新交易世界状态。

5、根据权利要求4所述的方法，所述根据所述第k个交易世界状态获取所述待更新账户地址对应的账户状态，作为待更新账户状态，包括：

根据所述待更新账户地址对所述第k个交易世界状态至所述最新上链世界状态进行账户状态映射顺序检索；

若在账户状态映射顺序检索的过程中检索到与所述待更新账户地址具有映射关系的账户状态，则将最先检索到的与所述待更新账户地址具有映射关系的账户状态作为待更新账户状态；

若在账户状态映射顺序检索的过程中未检索到与所述待更新账户地址具有映射关系的账户状态，则根据所述最新上链世界状态中的世界状态树获取所述待更新账户地址对应的待更新状态树根，根据所述待更新账户地址和所述待更新状态树根建立待更新账户状态。

6、根据权利要求4所述的方法，所述若所述第k个交易执行成功，则将所述第k个更新交易世界状态向所述第k-1个提交世界状态进行提交，得到第k个提交世界状态，包括：

遍历所述第k个更新交易世界状态包含的脏账户地址集合中的账户地址，顺序获取第j个账户地址；j为小于或等于I的正整数；I为所述第k个更新交易世界状态包含的脏账户地址集合中的账户地址的总数量；

从所述第k个更新交易世界状态包含的账户状态映射中获取所述第j个账户地址对应的现账户状态；

根据所述第k-1个提交世界状态获取所述第j个账户地址的前账户状态；

将所述第j个账户地址对应的现账户状态提交至所述第j个账户地址的前账户状态，将所述第j个账户地址插入到所述第k-1个提交世界状态包含的脏账户地址集合，得到第j个过渡提交世界状态；

当j等于I时，清空第k个更新交易世界状态包含的脏账户地址集合和账户状态映射，将第I个过渡提交世界状态作为第k个提交世界状态。

7、根据权利要求6所述的方法，所述第j个账户地址对应的现账户状态包含第一状态数据映射，所述第j个账户地址的前账户状态包含第二状态数据映射，所述将所述第j个账户地址对应的现账户状态提交至所述第j个账户地址的前账户状态，将所述第j个账户地址插入到所述第k-1个提交世界状态包含的脏账户地址集合，得到第j个过渡提交世界状态，包括：

将所述第一状态数据映射中状态数据键和状态数据值的映射关系，更新到所述第二状态数据映射中；

将所述第一状态数据映射中的状态数据键插入到所述第j个账户地址的前账户状态包含的脏状态数据键集合中，得到第j个过渡提交世界状态。

8、根据权利要求2所述的方法，所述根据所述世界状态链对所述初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，包括：

对所述初始更新状态进行世界状态树查询；

若所述初始更新状态包含第一世界状态树，则对所述初始更新状态进行全链脏账户状态提交处理，得到所述更新后的第一世界状态；

若所述初始更新状态未包含所述第一世界状态树，则根据所述第一前世界状态指针查找所述第二世界状态；

根据所述世界状态链对所述第二世界状态进行构建干净状态树处理，得到干净前世界状态树；

将所述干净前世界状态树的前状态树根作为现状态树根；

根据所述第一树节点缓存和所述现状态树根构建所述初始更新状态对应的第二世界状态树，将所述第二世界状态树添加到所述初始更新状态，得到过渡初始更新状态；

对所述过渡初始更新状态进行全链脏账户状态提交处理，得到所述更新后的第一世界状态。

9、根据权利要求8所述的方法，所述对所述过渡初始更新状态进行全链脏账户状态提交处理，得到所述更新后的第一世界状态，包括：

对所述过渡初始更新状态进行全链脏状态数据提交处理，得到提交后的过渡初始更新状态和全链脏账户地址集合；

根据所述提交后的过渡初始更新状态获取所述全链脏账户地址集合中的每个全链脏账户地址对应的账户状态根；

将所述每个全链脏账户地址和所述每个全链脏账户地址对应的账户状态根更新到所述第二世界状态树，得到干净现世界状态树；

将所述干净现世界状态树的更新树节点哈希和更新树节点的关联关系存储至所述提交后的过渡初始更新状态中的树节点缓存的树节点映射中，得到所述更新后的第一世界状态。

10、根据权利要求9所述的方法，所述对所述过渡初始更新状态进行全链脏状态数据提交处理，得到提交后的过渡初始更新状态和全链脏账户地址集合，包括：

顺序遍历所述世界状态链，获取第 x 个世界状态包含的脏账户地址集合，作为第 x 个脏账户地址集合； x 为小于或等于 Z 的正整数； Z 为所述过渡初始更新状态在所述世界状态链中的顺序序数；

遍历所述第 x 个脏账户地址集合中的账户地址，顺序获取第 m 个账户地址； m 为小于或等于所述第 x 个脏账户地址集合中账户地址的总数量的正整数；

获取所述第 m 个账户地址对应的现账户状态和前账户状态；

根据所述第 m 个账户地址的前账户状态对所述第 m 个账户地址的现账户状态进行账户脏状态数据提交处理；

当所述第 x 个脏账户地址集合中每个脏账户地址的现账户状态均进行账户脏状态数据提交处理后，得到第 x 个提交后的世界状态；

将所述第 x 个脏账户地址集合中每个脏账户地址添加至第 $x-1$ 个链上脏账户地址集合，得到第 x 个链上脏账户地址集合； x 为 1 时，第 $x-1$ 个链上脏账户地址集合为空集合；

当 x 等于 Z 时，将第 x 个链上脏账户地址集合作为全链脏账户地址集合，将第 x 个提交后的世界状态作为提交后的过渡初始更新状态。

11、根据权利要求 10 所述的方法，所述根据所述第 m 个账户地址的前账户状态对所述第 m 个账户地址的现账户状态进行账户脏状态数据提交处理，包括：

对所述第 m 个账户地址的现账户状态进行账户状态树查询；

若所述第 m 个账户地址的现账户状态不包含有账户状态树，或所述第 m 个账户地址的现账户状态的账户状态根不等于所述第 m 个账户地址的前账户状态的账户状态根，则根据所述第 m 个账户地址的现账户状态的树节点缓存和所述第 m 个账户地址的前账户状态的账户状态根建立指定账户状态树，将所述指定账户状态树添加至所述第 m 个账户地址的现账户状态，将第 m 个账户地址的现账户状态的账户状态根更新为所述第 m 个账户地址的前账户状态的账户状态根，得到指定账户状态；

若所述第 m 个账户地址的现账户状态包含有账户状态树，且所述第 m 个账户地址的现账户状态的账户状态根等于所述第 m 个账户地址的前账户状态的账户状态根，则将所述第 m 个账户地址的现账户状态作为指定账户状态；

若所述指定账户状态的第一脏状态数据键集合不为空，则从所述指定账户状态的状态数据映射中获取所述第一脏状态数据键集合中脏状态数据键对应的状态数据值，将所述第一脏状态数据键集合中脏状态数据键和所述第一脏状态数据键集合中脏状态数据键对应的状态数据值更新到所述指定账户状态的账户状态树，得到包含更新账户状态树的指定账户状态；

将所述包含更新账户状态树的指定账户状态的账户状态根更新为将所述更新账户状态树的状态树根；

清空所述包含更新账户状态树的指定账户状态的脏状态数据键集合；

将所述更新账户状态树的更新树节点哈希和更新树节点的关联关系存储至所述第 x 个世界状态。

12、根据权利要求 11 所述的方法，所述方法还包括：

对提交后的最新上链世界状态进行全链脏账户状态提交处理，得到干净最新上链世界状态，所述干净最新上链世界状态中的树节点缓存中包括第一树节点映射；

将所述第一树节点映射包含的树节点哈希和树节点更新到树节点数据库；

清空所述干净最新上链世界状态中的脏账户地址集合、账户状态映射和所述第一树节点映射。

13、一种基于区块链的数据处理装置，所述装置部署在计算机设备上，所述装置包括：初始化模块，用于生成初始区块；

状态创建模块，用于创建所述初始区块对应的第一世界状态；所述第一世界状态具有指向第二世界状态的链式关系，所述第二世界状态为所述初始区块的父区块对应的世界状态；所述第一世界状态和所述第二世界状态存在于世界状态链中；若所述父区块为已上链区块中具有最大区块高度的区块，则所述第二世界状态为共识根世界状态；所述世界状态链中的第一个世界状态为最新上链世界状态；所述共识根世界状态具有指向最新上链世界状态的链式关系；所述最新上链世界状态用于记录所述已上链区块对应的最新世界状态；

第一状态更新模块，用于根据所述初始区块中的交易更新所述第一世界状态，得到初始更新状态；

第二状态更新模块，用于根据所述世界状态链对所述初始更新状态进行构建干净状态树处理，得到更新后的第一世界状态，所述更新后的第一世界状态包括干净现世界状态树；

共识模块，用于将所述干净现世界状态树对应的第一世界状态根写入所述初始区块，得到待上链区块，将所述共识区块发送至共识节点，以使所述共识节点基于所述第一世界状态根对所述待上链区块进行共识处理，得到共识结果；

状态提交模块，用于若所述共识结果为共识通过结果，则将所述待上链区块写入区块链中，通过所述世界状态链将所述更新后的第一世界状态逐级提交至所述最新上链世界状态，将所述更新后的第一世界状态确定为新的共识根世界状态，所述新的共识根世界状态具有指向提交后的最新上链世界状态的链式关系。

14、一种计算机设备，包括：处理器、存储器以及网络接口；

所述处理器与所述存储器、所述网络接口相连，其中，所述网络接口用于提供数据通信功能，所述存储器用于存储程序代码，所述处理器用于调用所述程序代码，以执行权利要求 1-12 任一项所述的方法。

15、一种计算机可读存储介质，所述计算机可读存储介质中存储有计算机程序，所述计算机程序适于由处理器加载并执行权利要求 1-12 任一项所述的方法。

16、一种计算机程序产品，包括计算机程序/指令，所述计算机程序/指令被处理器执行时，可以执行权利要求 1-12 任一项所述的方法。