

数据归档方法、装置、设备、存储介质及计算机程序产品

相关申请的交叉引用

本申请实施例基于申请号为 202110921637.4、申请日为 2021 年 08 月 11 日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此引入本申请实施例作为参考。

技术领域

本申请涉及互联网技术领域，具体涉及计算机技术领域，尤其涉及一种数据归档方法、装置、设备、计算机可读存储介质及计算机程序产品。

背景技术

随着区块链技术的发展，区块头链应运而生，所谓的区块头链是指：区块链节点中，由区块头依次连接而组成的链型结构；一个区块头指向位于区块链节点的本地空间中的一个状态树，所谓的状态树是指用于组织存储用户的状态数据的数据结构，该状态数据可以是某个用户的资产（如数字资源、虚拟电子资源等）。
相关技术中，采用节点复用的方式来构建每个区块头所指向的状态树，使得相邻两个区块头所指向的状态树之间可复用部分树节点，以减少树节点的数量，从而节省区块链节点的存储空间。由于节点复用的方式，使得历史状态树中的部分树节点可能仍然被新区块所指向的新状态树引用，因此各个状态树所存储的状态数据不能随便删除。随着区块链网络的运行时间的增长，区块链节点中所存储的状态数据也在不断膨胀，大量的状态数据给区块链节点带来了存储空间的压力，相关技术尚无减轻区块链节点的存储压力的有效解决方案。

发明内容

本申请实施例提供了一种数据归档方法、装置、设备、计算机可读存储介质及计算机程序产品，可以实现对区块链节点中所存储的状态数据进行归档处理，来释放区块链节点的存储空间，减轻区块链节点的存储压力，从而提升区块链节点的运行性能，进而保证区块链节点在运行过程中的稳定性。
本申请实施例提供了一种数据归档方法，应用于计算机设备，所述方法包括：
当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树；
其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；
根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树；
其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；
根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树；
其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点

复用；

将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

本申请实施例提供了一种数据归档装置，所述装置包括：

- 5 查找单元，配置为当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树，其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；

- 10 第一重构单元，配置为根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树，其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；

- 15 第二重构单元，配置为根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树，其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用；

归档单元，配置为将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

本申请实施例提供了一种设备，所述计算机设备包括输入设备、输出设备，所述计算机设备还包括：

- 20 处理器，适于实现一条或多条指令；以及，

计算机可读存储介质，所述计算机可读存储介质存储有一条或多条指令，所述一条或多条指令适于由所述处理器加载并执行如下步骤：

当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树；

- 25 其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；

根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树；

- 30 其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；

根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树；

其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用；

- 35 将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

本申请实施例提供了一种计算机可读存储介质，所述计算机可读存储介质存储有一条或多条指令，所述一条或多条指令适于由所述处理器加载并执行如下步骤：

- 40 当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树；

其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；

根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树；

其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；

根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树；

5 其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用；

将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

10 本申请实施例提供了一种计算机程序产品，计算机程序产品包括计算机程序；该计算机程序存储在计算机可读存储介质中，计算机程序被计算机设备的处理器执行时，执行上述的数据归档方法。

在本申请实施例中，由于构建的目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据，该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在复用的树节点，即该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。并且，由于目标区块头之后各个第二区块头所指向的新状态树是基于目标区块头指向的新状态树所构建的，任一第二区块头所指向的新状态树可与该任一第二区块头的后一区块头所指向的新状态树存在节点复用，而与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。因此，在重构新状态树后，由于新状态树不存在节点复用，可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力，以有效保障区块链节点运行的稳定性。

附图说明

- 25 图 1a 是本申请实施例提供的一种状态树的结构示意图；
图 1b 是本申请实施例提供的一种状态树之间的节点复用情况的结构示意图；
图 1c 是本申请实施例提供的一种数据归档系统的架构示意图；
图 2 是本申请实施例提供的一种数据归档方法的流程示意图；
图 3 是本申请实施例提供的一种确定存在归档需求的结构示意图；
图 4 是本申请实施例提供的一种数据归档方法的流程示意图；
30 图 5a 是本申请实施例提供的一种目标区块头所指向的新状态树的结构示意图；
图 5b 是本申请实施例提供的一种第二区块头所指向的新状态树的结构示意图；
图 6a 是本申请实施例提供的一种数据归档方法的流程示意图；
图 6b 是本申请实施例提供的一种数据归档系统的架构示意图；
图 7 是本申请实施例提供的一种数据归档装置的结构示意图；
35 图 8 是本申请实施例提供的一种数据归档设备的结构示意图。

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做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。

40 本申请实施例涉及区块链网络，下面对区块链网络的相关术语及概念进行简要介绍：

本申请实施例涉及的交易，又称交易请求，交易包括了需要提交到区块链网络执行的操作以及对应的交易结果。本申请实施例涉及的交易并不局限于商业语境中的交易，即交易数据（又称状态数据）并不局限于数字货币等相关数据。例如，交易可以是部署（Deploy）交易、调用（Invoke）交易，部署交易用于将智能合约部署至区块链网络的节点中，并准备好被调用；调用交易用于对账本中的状态数据库进行查询操作（即读操作）或更新操作（即写操作，包括增加、删除和修改）。

区块链网络可以理解为一个用于进行节点与节点之间数据共享的网络，位于区块链网络中的节点可称为区块链节点；所谓的区块链节点是指：区块链网络中进行数据处理的计算机设备，该计算机设备可以是终端设备或者服务器，对此不做限定。此处所提及的终端设备可以是智能手机、平板电脑、笔记本电脑、台式电脑、车载计算机、智能家居、可穿戴电子设备、虚拟现实（VR, Virtual Reality）/增强现实（AR, Augmented Reality）设备等等；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、内容分发网络（CDN, Content Delivery Network）、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，等等。

为了保证区块链网络内的信息互通，每个区块链节点之间可以存在信息连接，以使区块链节点之间可以实现信息传输，进而实现信息共享；任意两个区块链节点之间可以实现点对点（P2P, Peer To Peer）通信，具体可以通过有线通信链路或无线通信链路进行 P2P 通信。对于每个区块链节点，均具有与其对应的节点标识，而且，区块链网络中的每个区块链节点均可以存储有区块链网络中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，将生成的区块广播至区块链网络中的其他区块链节点。任一区块链节点的节点标识可为互联网协议（IP, Internet Protocol）地址，或者其他任一种能够用于标识该区块链节点的信息。以节点标识为 IP 地址为例，每个区块链节点中可维护一个如下表 1 所示的节点标识列表：

表 1

节点名称	节点标识
节点 1	117.114.151.174
节点 2	117.116.189.145
...	...
节点 N	XX.XX.XX.XX

针对任一区块链节点（区块链网络中进行数据处理的计算机设备，即目标设备）而言，其内部可维护一条区块头链，该区块头链可由区块头连接组成的链形结构，每个区块头可以指向区块链节点的本地空间中的一个状态树，每个状态树中的各个数据节点均存储有状态数据。与完整的区块链不同的是，区块头链不需要保留所有账号完整的状态数据，仅在区块头中存储状态快照即可。所谓的状态快照是指用于指向状态树的数据，其中，状态快照可以表示目标设备中所存储的状态数据的固定长度的数据，只要任何状态数据有任何变化，状态快照均可以发生变化，以此可以判断所有状态数据的完整性。其中，目标设备可以根据区块头中的状态快照所指向的状态树来确定区块头所指向的状态树。状态数据可以存储在与区块头对应的区块体中，一个区块头和一个区块体可以组成区块链节点中所包括的一个区块。在区块体中，可以采用树型数据结构来对所有账号对应的状态数据进行存储，即状态数据可以以状态树的数据结构进行存储，该状态树中的叶子节点中可以存储有具体的状态数据。

在本申请实施例中，状态树可以是默克尔帕特里夏树（MPT, Merkle Patricia Trie），MPT 是一种组织数据的树形结构。MPT 在存储状态数据时，是以键值对（Key-Value）

数据的形式进行存储的。MPT 中的树节点可以包括数据节点、拓展节点以及分支节点三种类型的节点。数据节点是树型结构的叶子节点，只会出现在 MPT 底部，存储着具体的状态数据。拓展节点是指具有一个子节点的父节点，包含一个任意长度字符串 (Key)，
5 和一个指向子节点的哈希指针。分支节点是指具有 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 0-9a-f，并且分别有可能作为哈希指针而指向一个子节点，哈希指针是指 MPT 中子节点的哈希值，存储在父节点中。

为更好的理解本申请实施例所提供的 MPT，下面结合图 1a 所示的 MPT 进一步说明。如图 1a 所示，本申请实施例中的 MPT 可以是目标设备的本地空间中的任一状态树，
10 其中，表 2 所示为需要利用 MPT 进行数据存储的状态数据：

表 2

	Key	Value
状态数据 1	ab567cd	12.34
状态数据 2	abc1235	fenghm
状态数据 3	abc12b5	100

每个状态数据可以键值对 (Key-Value) 的形式进行组织。Key 可以是状态数据对应的关键字字符串，该关键字字符串可以是例如某一用户在业务上的账户标识，Value 可以是状态数据对应的具体值。区块头对应的状态数据的数量可以包括一个或多个。例如，表
15 2 所示中的 3 个状态数据为某一区块头对应的状态数据。如状态数据 1 对应的关键字字符串可以为“ab567cd”，该状态数据 3 对应的具体值可以为 12.34。又如，状态数据 2 对应的关键字字符串可以为“abc1235”，该状态数据 2 对应的具体值可以为 fenghm。又如，状态数据 3 对应的关键字字符串可以为“abc12b5”，该状态数据 1 对应的具体值可以为 100。

可以理解的是，在 MPT 中，哈希指针是指 MPT 中子节点的哈希值，且存储在父节点中，则可以根据每个树节点的哈希指针来确定当前树节点 (即父节点) 对应的下一层树节点 (即子节点)。其中，表 2 所示的每个状态数据中的关键字字符串 (Key) 可以对应一条从 MPT 中的树根节点到相应数据节点的节点路径。上述描述中的状态快照对应的具体值可以是树根节点中的哈希指针。基于此，可以根据每个树节点中所存储的哈希指针确定针对各个状态数据所对应的节点路径。例如，针对表 2 中的状态数据 1，其中，
20 状态数据 1 对应的节点路径可以包括拓展节点 11、分支节点 12、拓展节点 13 以及数据节点 14。其中，拓展节点 13 中的哈希指针是对数据节点 14 中所存储的状态数据 12.34 进行哈希计算得到的哈希值，分支节点 12 中的字符“5”的哈希指针是对拓展节点 13 中存储的数据进行哈希计算得到的哈希值。从上述可知，拓展节点 11 的哈希指针指向分支节点 12，分支节点 12 中的字符“5”的哈希指针指向拓展节点 13，拓展节点 13 的
25 哈希指针指向数据节点 14。同理可知，状态数据 2 和状态数据 3 分别对应的节点路径。

可以理解的是，在构建每个区块头所指向的状态树时，可以采用节点复用的方式来构建，则可以使相邻两个区块头所指向的状态树之间可复用部分树节点，非相邻两个区块头所指向的状态树之间也可复用部分树节点，在本申请实施例不作限定。例如，当前区块头对应的某一状态数据执行交易后，可以得到更新后的新状态数据，对于新区块头所指向的新状态树，该新状态树可以根据该更新后的新状态数据进行构建得到，该新
30 状态树中的数据节点用于存储更新后的新状态数据。未更新过的历史状态数据可以被新区块头中的状态快照指向，而根据该指向即可以确定该新区块头所指向的新状态树与当前区块头所指向的状态树之间的节点复用情况。通过上述可知，可以根据节点复用情况确定任意相邻的两个区块头中的后一区块头所指向的状态树，和前一区块头所指向的状态树中，被复用的树节点，而该复用的树节点即是复用树节点。
40

例如，图 1b 所示，目标设备在区块链节点中产生的新区块头 100，该区块头 100 所指的状态树中具有一个数据节点 24，该数据节点 24 存储的状态数据为 43.21，从区块头 99 所指的状态树中可以看出，当前区块头 99 中的状态数据包括 12.34、fenghm、100，其中，区块头 100 所对应的状态数据 43.21 是根据区块头 99 所对应的状态数据为 12.34 执行交易后所更新得到的新状态数据，即可以根据状态数据 43.21 构建区块头 100 所指的状态树。而针对区块头 99 中的未更新过的历史状态数据（fenghm 和 100），可以被区块头 100 中所存储的状态快照指向，区块头 99 中对应的未更新过的历史状态数据所在的分支中的树节点可以被区块头 100 指向的状态树所复用。例如，在图 1b 中，区块头 99 所指的状态树中的拓展节点 15、分支节点 16、拓展节点 17、数据节点 18、拓展节点 19、数据节点 20 是被区块头 100 指向的状态树所复用的树节点，这些复用的树节点也就是复用树节点。

基于上述区块链网络的相关介绍，本申请实施例提出了一种基于状态树的数据归档方法以及相应的数据归档系统，以实现区块链节点中所存储的过期的状态数据进行归档处理，来释放区块链节点的存储空间，减轻区块链节点的存储压力，从而提升区块链节点的运行性能，进而保证区块链节点在运行过程中的稳定性。请参见图 1c，是本申请实施例提供的一种数据归档系统的架构示意图，该数据归档系统包括区块链网络 100，以及位于区块链网络 100 外的归档服务器。其中，区块链网络 100 中可以包括多个区块链节点，一个区块链节点可与一个或多个归档服务器进行通信；所谓的归档服务器是指：基于与区块链节点的通信，为相应区块链节点提供数据存储服务和数据查询服务的服务器。图 1c 中所示的区块链节点的数目以及归档服务器的数目仅仅是示意性的，根据实际需要，可以部署任意数目的区块链节点和与每个区块链节点进行通信的归档服务器。

在本申请实施例中，为了释放区块链节点的存储空间，本申请实施例基于图 1c 所示的数据归档系统提供了一种数据归档方法，在具体实现中，该数据归档方法可由计算机设备执行，该计算机设备可以是图 1c 所示中的任一区块链节点。该数据归档方法的实现过程如下：当存在针对目标区块头的归档需求时，可以在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。其中，任一区块头指向的当前状态树是指：当存在归档需求时，任一区块头指向的状态树。进而可以根据查找到的各个当前状态树和查找到的各个当前状态树之间的节点复用情况，重构出目标区块头指向的新状态树，目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据的。可以基于目标区块头指向的新状态树，和位于目标区块头之后的各个第二区块头对应的目标状态数据，重构出各个第二区块头指向的新状态树。其中，第二区块头是指：位于目标区块头之后的区块头；任一新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用，即相邻的两个新状态树存在节点复用，非相邻的两个新状态树也可以存在节点复用。在重构出目标区块头指向的新状态树和各个第二区块头指向的新状态树之后，可以将本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在成功归档后，删除本地空间中的各个当前状态树。

作为示例，用户 A、用户 B、用户 C 先后均购买了同一件商品，用户 A 通过“路径 1-路径 2-路径 3-路径 4”这一途径 A 购买了商品，用户 B 通过“路径 1-路径 2-路径 5-路径 6”这个途径 B 购买了商品，用户 C 通过“路径 7-路径 8-路径 3-路径 4”这个途径购买了商品，用户 A、用户 B、用户 C 通过各自对应的计算机设备将各自的商品购买途径上链，则按照购买顺序，第一区块头指向的当前状态树存储有途径 A 的状态数据，当前区块头指向的当前状态树存储有途径 B 的状态数据，第二区块头指向的当前状态树存储有途径 C 的状态数据，则目标区块头指向的状态树与第一区块头所指向的状态树存在节点复用（即路径 1-路径 2），目标区块头指向的状态树与第二区块头所指向的状态树

存在节点复用（即路径 3-路径 4），由于存在节点复用，因此各个状态树所存储的状态数据不能随便删除。当区块链网络中不需要存储用户 A、用户 B、用户 C 购买该商品的购物记录时，通过本申请实施例的数据归档方法，在重构新状态树后，由于新状态树不存在节点复用，可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力，以有效保障区块链节点运行的稳定性。

作为示例，用户 A、用户 B、用户 C 进行针对物品推荐模型的联邦学习任务，用户 A 通过“样本 1-样本 2-样本 3-样本 4”这一样本集合 A 训练物品推荐模型，用户 B 通过“样本 1-样本 2-样本 5-样本 6”这个样本集合 B 训练物品推荐模型，用户 C 通过“样本 7-样本 8-样本 3-样本 4”这一样本集合 C 训练物品推荐模型，用户 A、用户 B、用户 C 通过各自对应的计算机设备将各自的样本集合训练物品推荐模型的过程上链，则按照训练顺序，第一区块头指向的当前状态树存储有样本集合 A 的状态数据，当前区块头指向的当前状态树存储有样本集合 B 的状态数据，第二区块头指向的当前状态树存储有样本集合 C 的状态数据，则目标区块头指向的状态树与第一区块头所指向的状态树存在节点复用（即样本 1-样本 2），目标区块头指向的状态树与第二区块头所指向的状态树存在节点复用（即样本 3-样本 4），由于存在节点复用，因此各个状态树所存储的状态数据不能随便删除。当区块链网络中不需要存储用户 A、用户 B、用户 C 训练物品推荐模型的训练记录时，通过本申请实施例的数据归档方法，在重构新状态树后，由于新状态树不存在节点复用，可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力，以有效保障区块链节点运行的稳定性，并且基于联邦学习的数据隐私保护的特性，通过各自的训练数据独立训练模型，提高区块链的数据安全性。

基于上述数据归档方案的相关描述可知，本申请实施例中所构建的目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据，该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在复用的树节点，即该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。并且，由于目标区块头之后各个第二区块头所指向的新状态树是基于目标区块头指向的新状态树所构建的，那么同理，任一第二区块头所指向的新状态树可与该任一第二区块头的后一区块头所指向的新状态树存在节点复用，而与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。那么，后续可以在将本地空间中存储的各个当前状态树作为归档数据，并成功归档至对应的归档服务器中之后，可以删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史状态数据而给存储空间带来的压力，进而可以有效保障区块链节点运行的稳定性。

如上所示，本申请实施例提供了一种数据归档方法，该数据归档方法可由上述所提及的目标设备（用于数据归档的计算机设备，即数据归档设备）执行。请参阅图 2，该数据归档方法可包括以下步骤 S201-S204：

S201，当存在针对目标区块头的归档任务时，在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。

其中，任一区块头指向的当前状态树表示当存在归档任务（又称归档需求）时，任一区块头指向的状态树。

例如，可以是在目标设备（区块链网络中进行数据处理的区块链节点）接收到一个针对目标区块头的归档请求时，则确定当前存在针对目标区块头的归档需求。例如，归档管理员可以向目标设备发送一个针对目标区块头的归档请求，以使目标设备接收该归

档请求,而在目标设备接收到该归档请求之后,也就确定存在针对目标区块头的归档任务(又称归档需求)。在一种实现方式中,当归档管理员需要对目标设备的本地空间存储的状态数据进行数据归档时,归档管理员可以通过在终端的用户操作界面执行相关操作,以向目标设备发送针对目标区块头的归档请求。例如参见图3,归档管理员所使用的终端的屏幕上显示一个用户操作界面,该用户操作界面可以至少包括由301标记的区块头设置区域,以及由302标记的确认控件。若归档管理员想要对第100个区块头所指示的当前状态树存储的状态数据进行归档处理,则该归档管理员可在该区块头设置区域301中输入第100个区块头的相关信息(如区块头号),然后对该确认控件302执行触发操作(如点击操作、按压操作等),从而触发归档管理员所使用的终端根据在区块头设置区域301中获取到的相关信息所指示的区块头,确定目标区块头,并向目标设备发送针对该目标区块头的归档请求。

例如,归档任务(又称归档需求)还可以是触发归档定时任务所产生的。如可以设置一个归档定时任务,该归档定时任务中指示了对目标设备在本地空间中的状态数据进行归档的触发条件。例如,该触发条件可以是当前时间到达预设归档时间,或者,目标设备在本地空间的剩余存储空间到达了预设剩余存储空间等等。

例如,当存在针对目标区块头的归档需求时,可以对针对目标区块头的归档需求进行需求校验,以校验该归档需求是否是可实现的,如果校验结果是可实现的,则可以进行后续的步骤,如果校验结果是不可实现的,则不可以进行后续的步骤。其中,可以根据目标区块头的区块高度对目标区块头的归档需求进行需求校验,其实现过程可以为下述描述:可以在区块头链中确定当前区块头,并在区块头链中确定最新区块头。其中,该当前区块头可以是指在本本地空间中指向有状态树、且区块高度最小的区块头;该最新区块头可以是指在本本地空间中指向有状态树、且区块高度最大的区块头。而在确定区块头链中的当前区块头和最新区块头之后,可以根据目标区块头的区块高度、以及当前区块头和最新区块头分别对应的区块高度对归档需求进行需求校验。例如,如果目标区块头的区块高度大于当前区块头的区块高度、且小于最新区块头的区块高度,则表明当前存在可归档的数据,也可以表明针对目标区块头的归档需求是可实现的,即可以执行在本本地空间中查找目标区块头指向的当前状态树,以及位于目标区块头之前的各个第一区块头指向的当前状态树的步骤。如果目标区块头的区块高度小于当前区块头的区块高度,或大于最新区块头的区块高度,则表明当前不存在可归档的数据,也可以表明针对目标区块头的归档需求是不可实现的,则无需进行后续的步骤。在这种情况下,目标设备还可以向归档管理设备返回一个报错信息,该报错信息可以用于指示当前归档需求不满足归档要求,以便于归档管理设备对应的归档管理员对归档需求进行调整。例如,该报错信息可以显示在图3中的用户操作界面上的第二区域,例如图3中用户操作界面上显示的数据归档失败可以是一个报错信息。

其中,为了方便后续描述,区块头可以按照区块高度从低到高的顺序排列,因此可以将当前区块头以及当前区块头和目标区块头之间的各个区块头,均被作为第一区块头;可以将最新区块头以及最新区块头和目标区块头之间的各个区块头,均被作为第二区块头。

例如,当存在针对目标区块头的归档需求时,目标设备可以在本地空间中查找目标区块头指向的当前状态树,以及位于目标区块头之前的各个第一区块头指向的当前状态树。其中,在查找当前状态树时,可以根据目标区块头中所存储的状态快照来进行查找,每一个状态快照可以指向一个状态树。同理,位于目标区块头之前的各个第一区块头指向的当前状态树也可以根据各个第一区块头所存储的状态快照来进行查找。

S202,根据查找到的各个当前状态树和各个当前状态树之间的节点复用信息,重构

出目标区块头指向的新状态树。

其中，目标区块头指向的新状态树独立存储目标区块头对应的所有状态数据，即目标区块头指向的新状态树与各个第一区块头指向的当前状态树之间不存在节点复用。

5 在一些实施例中，目标设备可以根据查找到的各个当前状态树之间的节点复用信息（又称节点复用情况），从各个当前状态树中确定至少一个复用树节点。其中，复用树节点表示在目标区块头指向的当前状态树和任一第一区块头指向的当前状态树中，被复用的树节点。而在确定复用树节点之后，可以根据复用树节点以及目标区块头指向的当前状态树的各个树节点，确定目标区块头指向的新状态树。例如，目标设备可以复制至少一个复用树节点，以及复制目标区块头指向的当前状态树中的各个树节点，根据复制得到的树节点，重构出目标区块头指向的新状态树。其中，复制得到的树节点可以包括：复制得到的各个复用树节点，以及复制得到的目标区块头指向的当前状态树中的各个树节点。

15 需要说明的是，由于构建的目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据，该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在复用的树节点，即该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用，由于新状态树不存在节点复用，后续可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力。

20 S203，根据目标区块头指向的新状态树，和位于目标区块头之后的各个第二区块头对应的目标状态数据，重构出各个第二区块头指向的新状态树。

其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用。

25 其中，假设第二区块头的数量为 M 个， M 为正整数。第 m 个第二区块头对应的目标状态数据表示第 m 个第二区块头对应的多个状态数据中，与第 m 个第二区块头的前一区块头对应的各个状态数据均不同的状态数据；也可以说是在第 m 个第二区块头对应的多个状态数据的基础上，进行状态数据的更新，而产生的新的状态数据。例如，第 m 个第二区块头的前一区块头对应的状态数据包括：状态数据 A、状态数据 B、状态数据 C，第 m 个第二区块头对应的状态数据包括：状态数据 D、状态数据 B、状态数据 C，30 从上述数据可知，第 m 个第二区块头对应的目标状态数据为状态数据 D。其中， $m \in [1, M]$ 。

35 在一些实施例中，针对各个第二区块头中的任一第二区块头，例如，以该任一第二区块头为第 m 个第二区块头为例进行说明，重构出第 m 个第二区块头指向的新状态树的实施过程可以是：目标设备可以基于第 m 个第二区块头对应的目标状态数据，生成新叶子节点，该新叶子节点也就是新数据节点。再将第 m 个第二区块头对应的目标状态数据，存储至新叶子节点中。以根据存储目标状态数据后的新叶子节点，以及第 m 个第二区块头的前一区块头所指向的新状态树，重构第 m 个第二区块头指向的新状态树。

40 通过上述方法，可以重构出每一个第二区块头所指向的新状态树。其中，针对任一第二区块头所指向的新状态树，可以与相应区块头的后一第二区块头所指向的新状态树存在节点复用。例如，第 m 个第二区块头指向的新状态树和前一区块头指向的新状态树之间，复用至少一个叶子节点（数据节点）。在一种实现中，当 m 等于 1 时，前一区块头为目标区块头。

需要说明的是，由于目标区块头之后各个第二区块头所指向的新状态树是基于目标区块头指向的新状态树所构建的，任一第二区块头所指向的新状态树可与该任一第二区

块头的后一区块头所指向的新状态树存在节点复用，而与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用，由于新状态树不存在节点复用，后续可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力。

5 S204，将本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除本地空间中的各个当前状态树。

例如，在通过上述步骤重构出目标区块头指向的新状态树、以及重构出各个第二区块头指向的新状态树之后，即可以将归档数据归档至目标服务器，该归档数据可以是本地空间中存储的各个当前状态树。而在目标归档服务器成功归档之后，则可以删除本地空间中的各个当前状态树，而在删除本地空间中的当前状态树之后，可以使得本地空间的当前存储空间变大，从而可以减小本地空间的存储压力。

10 在一些实施例中，在将归档数据归档至目标归档服务器时，也需要将针对数据归档处理的开始区块号和结束区块号归档至目标归档服务器，以便于后续目标设备在接收到针对指定区块头的查询请求时，可以根据开始区块号和结束区块号进行数据的查询。在将归档数据归档至目标归档服务器时，可以将各个第一区块头中，区块高度最小的第一区块头的区块号作为开始区块号，并将各个第一区块头中，区块高度最大的第一区块头的区块号作为结束区块号。那么，在确定开始区块号和结束区块号之后，可以将本地空间中存储的各个当前状态树作为归档数据，并根据归档数据、开始区块号以及结束区块号生成归档请求。进而将该归档请求发送至目标归档服务器，以请求目标归档服务器对归档数据进行哈希运算，得到归档数据哈希值，并在存储归档数据哈希值、归档数据、开始区块号以及结束区块号后，返回归档数据哈希值。

15 其中，哈希运算可以是摘要哈希运算，也可以是其他方式的哈希运算，在本申请实施例中不作限定。需要说明的是，基于开始区块号和结束区块号机数据归档后，后续可以开始区块号和结束区块号快速进行数据查询，提高数据查询效率。

20 在上述将归档数据归档至目标归档服务器的过程中，如果目标设备接收到目标归档服务器发送的归档数据哈希值，则可以确定归档数据已被目标归档服务器成功归档。而在目标设备确定归档数据被成功归档之后，则可以将归档数据哈希值、归档时间戳、开始区块号、结束区块号、以及目标归档服务器的服务器标识记录在本地空间的归档记录中。以便于后续可以根据本地空间的归档记录进行数据查询。其中，归档时间戳可以是指将归档数据归档至目标归档服务器时的时间，目标归档服务器的服务器标识可以是指目标归档服务器对应的 IP 地址，也可以是其他能够用于标识该目标归档服务器的信息，在本申请实施例中不作限定。在下述描述中，以服务器标识为 IP 地址为例进行相关的说明。

30 在一些实施例中，在目标归档服务器接收到目标设备发送的归档数据，并根据所述归档数据进行哈希运算，得到归档数据哈希值之后，也可以将归档数据哈希值、归档数据、开始区块号、结束区块号以及归档时间戳记录在目标归档服务器中的归档记录中，以便于可以根据目标归档服务器中的归档记录进行数据查询。

40 举例来说，假设归档数据以列表的形式进行记录，则上述的归档记录可以是归档记录列表，即可以将归档数据等信息记录在本地空间的归档记录列表中，对应的，也可以将归档数据等信息记录在目标归档服务器中的归档记录中。例如，表 3 所示为在本地空间的归档记录列表：

表 3

归档时间戳	归档数据哈希值	开始区块号	结束区块号	服务器标识
T1	A	A1	A2	IP1

T2	B	B1	B2	IP2
T3	C	C1	C2	IP3
...	...	...	...	...

表 3 中的每一行是一次数据归档处理的归档相关信息，该归档记录列表中可以记录每一次数据归档处理的归档相关信息，且可以将每一次数据归档处理的归档相关信息记录在当前归档记录列表的末尾，以保证归档记录列表中所记录内容的有序性，方便后续用户查看归档记录列表。本地空间的归档记录列表中的归档相关信息可以包括归档时间戳、归档数据哈希值、开始区块号、结束区块号、服务器标识。如表 3 所示，某一次数据归档处理后在归档记录列表中所记录的归档相关信息包括：归档时间戳为 T1、归档数据哈希值为 A、开始区块号 A1、结束区块号 A2、服务器标识 IP1。

又如，表 4 所示为在目标归档服务器的归档记录列表：

表 4

归档时间戳	归档数据哈希值	开始区块号	结束区块号	归档数据
T4	D	D1	D2	归档数据 D
T2	B	B1	B2	归档数据 B
T5	E	E1	E2	归档数据 E
...	...	...	...	...

表 4 中的每一行是一次数据归档处理的归档相关信息，目标归档服务器的归档记录列表也可以记录每一次数据归档处理的归档相关信息，且可以将每一次数据归档处理的归档相关信息记录在当前归档记录列表的末尾，以保证归档记录列表中所记录内容的有序性，方便后续用户查看归档记录列表。目标归档服务器的归档记录列表中的归档相关信息可以包括归档时间戳、归档数据哈希值、开始区块号、结束区块号、归档数据。如表 4 所示，某一次数据归档处理后在归档记录列表中所记录的归档相关信息包括：归档时间戳为 T1、归档数据哈希值为 A、开始区块号 A1、结束区块号 A2、归档数据 A。

其中，目标归档服务器的归档记录列表与在本地空间的归档记录列表是同步记录的，即本地空间的归档记录列表中的记录在目标归档服务器的归档记录列表中存在相应的记录。例如，如果目标设备可以与多个归档服务器进行通信，即目标设备在本地空间中的每一次所要进行归档的归档数据可以归档至不同的归档服务器中。根据表 3 中的服务器标识，也可以确定某一次记录的归档数据具体是归档到了哪一个归档服务器。例如，表 3 中的第三行所记录的信息是某一次数据归档处理的归档相关信息，假设表 4 所示的归档记录列表对应的目标归档服务器的服务器标识为 IP2，则根据表 3 可知，该某一次数据归档处理得到归档数据是归档到服务器标识为 IP2 的目标归档服务器，则可以在服务器标识为 IP2 的目标归档服务器的归档记录列表中查询到与该某一次数据归档处理的归档相关信息，例如，表 4 中的第三行中所记录的信息即是该某一次数据归档处理的归档相关信息。

在一些实施例中，目标设备在每一次进行数据归档时，是可以将与归档数据相关的信息记录在本地空间的归档记录中、以及目标归档服务器的归档记录中的。那么，后续目标设备在接收到例如数据查询请求时，可以根据上述的两个归档记录中记录的信息进行查询。例如，目标设备可以响应于针对指定区块头的数据查询请求，以进行数据查询。在目标设备响应该数据查询请求之后，可以在本地空间的归档记录中读取结束区块号，以根据该结束区块号确定是在本地空间对指定区块头的数据进行查询，还是在目标归档服务器中对指定区块头的数据进行查询。例如，当指定区块头的区块号小于或等于结束区块号时，则可以在目标归档服务器中进行查询；当指定区块头的区块号大于结束区块号时，则可以在本地空间中进行查询。

例如，当指定区块头的区块号小于或等于结束区块号时，可以根据本地空间内的归

档记录中的归档数据哈希值、以及指定区块头的区块号生成指定数据查询请求。而在生成指定数据查询请求之后，可以将指定数据查询请求发送至目标归档服务器，以使该目标归档服务器根据指定数据查询请求携带的归档数据哈希值，确定并返回指定区块头对应的状态数据，其中，归档数据哈希值指示归档数据中的当前状态树、以及指定区块头的区块号。目标设备可以接收目标归档服务器返回的指定区块头对应的状态数据，并输出接收到的状态数据。

当指定区块头的指定区块号大于结束区块号时，可以在本地空间中确定指定区块头所指向的新状态树、以及确定位于指定区块头之前的各个区块头所指向的新状态树。例如，可以基于确定的各个新状态树，获取指定区块头对应的状态数据，并输出获取到的状态数据。

在一些实施例中，目标设备可以与一个或多个归档服务器进行通信。当目标设备与一个归档服务器进行通信时，则目标归档服务器即是与目标设备进行通信的归档服务器。当目标设备与多个归档服务器进行通信时，可以从多个归档服务器中确定一个目标归档服务器，以利用该目标归档服务器对归档数据进行归档。其中，可以从多个归档服务器中随机选择一个归档服务器作为目标归档服务器，或者，也可以是按目标选择规则从多个归档服务器中选择一个归档服务器作为目标归档服务器。例如，目标选择规则可以是指从多个归档服务器中选择一个最优的归档服务器作为目标归档服务器。其中，最优的评价标准可以是根据每个归档服务器的归档优先级来判定，而最优的归档服务器可以是指归档优先级最高的归档服务器，即指归档优先级最高的归档服务器可以作为目标归档服务器。每个归档服务器的归档优先级可以根据每个归档服务器的归档性能参数来确定。

需要说明的是，根据每个归档服务器的归档性能参数来确定目标归档服务器，确定出的目标归档服务器的归档性能较强，能够进行快速的数据归档，从而尽快减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力。

例如，目标设备可以获取每个归档服务器的归档性能参数，该归档性能参数可以包括剩余存储数据量、传输带宽、安全系数中的至少一种。而在获取到每个归档服务器的归档性能参数之后，即可以根据归档性能参数确定每个归档服务器的归档优先级，并将最高归档优先级对应的归档服务器确定为目标归档服务器。

以归档性能参数为剩余存储数据量、传输带宽、安全系数中的任一种为例进行说明。

假设归档性能参数为剩余存储数据量，则可以获取每个归档服务器的剩余存储数据量。其中，剩余存储数据量越小，其对应的归档服务器的归档优先级也就越低，对应的，剩余存储数据量越大，其对应的归档服务器的归档优先级也就越高。也就是说，最大剩余存储数据量对应的归档服务器的归档优先级最高，则可以将最大剩余存储数据量对应的归档服务器确定为目标归档服务器。在一种实现中，获取每个归档服务器的剩余存储数据量的具体实现方式可以是：目标设备可以向每个归档服务器发送一个剩余数据量获取请求，以请求每个归档服务器返回各自的剩余存储数据量。而当各个归档服务器接收该剩余数据量获取请求之后，可以确定自身的剩余存储数据量，并将该剩余存储数据量返回至目标设备，则目标设备可以接收到各个归档服务器所返回的剩余存储数据量。

假设归档性能参数为传输带宽，则可以获取每个归档服务器的传输带宽。其中，传输带宽越小，其对应的归档服务器的归档优先级也就越低，对应的，传输带宽越大，其对应的归档服务器的归档优先级也就越高。也就是说，最大传输带宽对应的归档服务器的归档优先级最高，则可以将最大传输带宽对应的归档服务器确定为目标归档服务器。在一种实现中，获取每个归档服务器的归档性能参数所包括的传输带宽的具体实现过程可以是：目标设备可以向每个归档服务器发送带宽探测请求，带宽探测请求用于指示每个归档服务器返回指定数据量的目标数据。那么，在每个归档服务器返回指定数据量的

目标数据之后，目标设备可以接收每个归档服务器返回的目标数据。目标设备可以基于每个归档服务器返回目标数据所消耗的时长，来确定每个归档服务器的传输带宽。

假设归档性能参数为安全系数，则可以获取每个归档服务器的安全系数。其中，安全系数越小，其对应的归档服务器的归档优先级也就越低，对应的，安全系数越大，其对应的归档服务器的归档优先级也就越高。也就是说，最大安全系数对应的归档服务器的归档优先级最高，则可以将最大安全系数对应的归档服务器确定为目标归档服务器。在一种实现中，获取每个归档服务器的归档性能参数所包括的安全系数的具体实现过程可以是：目标设备可以向每个归档服务器发送一个安全系数获取请求，以请求每个归档服务器返回各自的安全系数。而当各个归档服务器接收该安全系数获取请求之后，可以确定自身的安全系数，并将该安全系数返回至目标设备，则目标设备可以接收到各个归档服务器所返回的安全系数。其中，该安全系数可以用于衡量归档服务器被窃取数据的概率。归档服务器确定自身的安全系数的具体实施方式可以是：归档服务器可以统计数据被归档的次数和归档数据被成功窃取的数据量，将归档数据被成功窃取的数据量与数据被归档的数据量之间的比值作为安全系数。或者也可以是：归档服务器可以统计数据被归档的数据量和归档数据被成功窃取的数据量，将归档数据被成功窃取的数据量与数据被归档的数据量之间的比值作为安全系数。还可以利用其他方式确定归档服务器的安全系数，在本申请实施例不作限定。

例如，可以预先设置归档性能参数与归档优先级的对应关系，那么，目标设备在获取到归档性能参数之后，可以根据获取到的归档性能参数和预先设置的对应关系确定归档性能参数对应的归档优先级。其中，该对应关系可以是指归档性能参数与归档优先级呈正相关，也可以是指归档性能参数与归档优先级之间具体的数值对应关系，在本申请实施例不作限定。例如，在归档性能参数包括剩余存储数据量、传输带宽、安全系数中的一种时，该对应关系可以是指归档性能参数与归档优先级呈正相关，如上述描述的剩余存储数据量越小，其对应的归档服务器的归档优先级也就越低。又如上述描述的传输带宽越小，其对应的归档服务器的归档优先级也就越低。又如上述描述的安全系数越小，其对应的归档服务器的归档优先级也就越低。假设归档性能参数为剩余存储数据量，与目标设备进行通信的归档服务器的数量为3，且3个归档服务器分别对应的剩余存储数据量的大小关系为： $P1$ （归档服务器1） $>P2$ （归档服务器2） $>P3$ （归档服务器3）。根据剩余存储数据量的大小关系可知，3个归档服务器的归档优先级的高低关系为：归档服务器1 $>$ 归档服务器2 $>$ 归档服务器3，即归档服务器1的归档优先级最高。

以归档性能参数为剩余存储数据量、传输带宽、安全系数为例进行说明，针对多个归档服务器中的任一归档服务器而言，目标设备可以获取该归档服务器的剩余存储数据量、传输带宽、安全系数，其中，剩余存储数据量、传输带宽以及安全系数的获取方式可以参考上述描述，在本申请实施例不作限定。例如，目标设备可以根据该归档服务器的剩余存储数据量确定第一优先级，剩余存储数据量越大，对应的第一优先级也就越高。同理，也可以根据该归档服务器的传输带宽确定第二优先级，也可以根据该归档服务器的安全系数确定第三优先级。而在确定每个归档服务器对应的第一优先级、第二优先级和第三优先级之后，可以根据该3个优先级确定每个归档服务器的归档优先级。例如，针对一个归档服务器而言，可以将第一优先级、第二优先级和第三优先级的和值作为该归档服务器的归档优先级。又如，也可以将各个优先级与其对应的参数权重分别进行加权处理之后，将各个加权处理的和值作为该归档服务器的归档优先级。其中，每个优先级对应的参数权重可以预先设置，可以根据实际的业务需求设置对应的参数权重，可以将业务需求较为看重的归档性能参数对应的参数权重设为较大，各个参数权重之间的和值为1。通过上述方式，可以确定每一个归档服务器的归档优先级，那么，在确定每一

个归档服务器的归档优先级之后，则可以将最高归档优先级对应的归档服务器确定为目标归档服务器。

其中，本申请实施例中的优先级（例如上述描述归档优先级、第一优先级、第二优先级、第三优先级）的高低可以理解为数值越大，优先级越高，上述的优先级也可以是其他数值，可以体现优先级和优先级的高低即可。

在上述描述中可知，目标设备可以预先设置归档性能参数与归档优先级的对应关系，同理，在归档性能参数包括剩余存储数据量、传输带宽、安全系数中的多种时，可以设置每一种归档性能参数与其优先级的对应关系。例如，可以设置剩余存储数据量与第一优先级之间的对应关系、传输带宽与第二优先级之间的对应关系、以及安全系数与第三优先级之间的对应关系。为了方便后续根据第一优先级、第二优先级和第三优先级确定归档优先级，此处的对应关系可以是指归档性能参数与归档优先级之间的数值对应关系。例如表 5 所示为剩余存储数据量与第一优先级之间的对应关系：

表 5

剩余存储数据量	第一优先级
0~K1	1
K1~K2	2
K2~K3	3
...	...

又如表 6 所示为传输带宽与第二优先级之间的对应关系：

表 6

传输带宽	第二优先级
0~Q1	1
Q1~Q2	2
Q2~Q3	3
...	...

又如表 7 所示为安全系数与第三优先级之间的对应关系：

表 7

安全系数	第一优先级
R1	1
R2	2
R3	3
...	...

举例来说，假设目标设备与归档服务器进行通信的数量为 3（归档服务器 1、归档服务器 2、归档服务器 3），归档性能参数包括剩余存储数据量、传输带宽、安全系数。目标设备在获取到每个归档服务器的归档性能参数之后，可以根据归档性能参数所包括的剩余存储数据量，和如表 5 所示的剩余存储数据量与第一优先级之间的对应关系确定第一优先级，同理，可以根据传输带宽和如表 6 所示的对应关系确定第二优先级，可以根据安全系数和如表 7 所示的对应关系确定第三优先级。假设，归档服务器 1 的第一优先级为 3、第二优先级为 2、第三优先级为 5，则归档服务器 1 的归档优先级可以为 10（3+2+5）。归档服务器 2 的第一优先级为 5、第二优先级为 1、第三优先级为 3，则归档服务器 2 的归档优先级可以为 9（5+1+3）。归档服务器 3 的第一优先级为 1、第二优先级为 4、第三优先级为 2，则归档服务器 3 的归档优先级可以为 7（1+4+2）。从上述可知，归档服务器 1 的归档优先级最高，则可以将归档服务器 1 确定为目标归档服务器。

在本申请实施例中，当存在归档需求时，目标设备可以在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。

进而可以根据查找到的各个当前状态树和查找到的各个当前状态树之间的节点复用情况，重构出目标区块头指向的新状态树，目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据的。基于目标区块头指向的新状态树，和位于目标区块头之后的各个第二区块头对应的目标状态数据，重构出各个第二区块头指向的新状态树。在重构出目标区块头指向的新状态树和各个第二区块头指向的新状态树之后，将本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在成功归档后，删除本地空间中的各个当前状态树。通过上述实施方法，可以构建目标区块头指向的新状态树以及目标区块头之后各个第二区块头所指向的新状态树，由于目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据，则该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在复用的树节点，即该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。并且，由于目标区块头之后各个第二区块头所指向的新状态树是基于目标区块头指向的新状态树所构建的，则任一第二区块头所指向的新状态树可与该任一第二区块头的后一区块头所指向的新状态树存在节点复用，而与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。因此，在重构新状态树后，由于新状态树不存在节点复用，可以将本地空间中存储的各个当前状态树进行归档，删除本地空间中的各个当前状态树，从而释放了区块链节点的存储空间，减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力，以有效保障区块链节点运行的稳定性。

20 本申请实施例提供了一种数据归档方法，该数据归档方法可由上述所提及的目标设备执行。请参阅图 4，该数据归档方法可包括以下步骤 S401-S408：

S401，当存在针对目标区块头的归档任务时，在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。

25 S402，根据查找到的各个当前状态树之间的节点复用信息，从查找到的各个当前状态树中确定至少一个复用树节点。

其中，复用树节点表示在目标区块头指向的当前状态树和任一第一区块头指向的当前状态树中，被复用的树节点。

30 例如，在图 1b 中，区块头 99 指向的当前状态树中的拓展节点 51、分支节点 52、拓展节点 53、数据节点 54、拓展节点 55、数据节点 56 是被区块头 100 指向的当前状态树所复用的树节点，则这些复用的树节点也就是复用树节点。

S403，复制至少一个复用树节点，以及复制查找到的目标区块头指向的当前状态树中的各个树节点。

S404，根据复制得到的树节点，重构出目标区块头指向的新状态树。

35 在步骤 S403 和步骤 S404 中，可以复制至少一个复用树节点，以及复制查找到的目标区块头指向的当前状态树中的各个树节点，并采用复制得到的树节点，重构出目标区块头指向的新状态树。该复制得到的树节点可以包括：复制得到的各个复用树节点，以及复制得到的目标区块头指向的当前状态树中的各个树节点。

40 例如，以图 1b 为例，假设目标区块头为区块头 100，目标区块头指向的当前状态树中的各个树节点可以如图 1b 中的拓展节点 21、分支节点 22、拓展节点 23、数据节点 24，且区块头 100 指向的当前状态树所复用的树节点是区块头 99 指向的当前状态树中的树节点。即上述描述的区块头 99 指向的当前状态树中的拓展节点 15、分支节点 16、拓展节点 17、数据节点 18、拓展节点 19、数据节点 20。那么，根据上述的各个树节点可以重构出目标区块头指向的新状态树，该目标区块头指向的新状态树可以如图 5a 所示。

45 S405，针对第 m 个第二区块头，基于第 m 个第二区块头对应的目标状态数据，生

成新叶子节点。

其中，第二区块头的数量为 M 个， M 为正整数， $m \in [1, M]$ 。第 m 个第二区块头对应的目标状态数据表示第 m 个第二区块头对应的多个状态数据中，与第 m 个第二区块头的前一区块头对应的各个状态数据均不同的状态数据。

5 例如，第 m 个第二区块头对应的目标状态数据的获取方式可以是通过交易得到，也可以是从该第 m 个第二区块头所指向的当前状态树中得到。

在一种实现方式中，通过交易得到第二区块头对应的目标状态数据的实施过程可以包括：目标设备可以获取位于目标区块头之后的各个第二区块头所对应的交易数据，其中，每一个第二区块头所对应的交易数据可以用于指示针对至少一个状态数据的交易操作。例如，以第 m 个第二区块头为例，第 m 个第二区块头中的交易数据指示了对第 m 个第二区块头的前一区块头中的状态数据的交易操作，假设 m 等于 1，则第 m 个第二区块头中的交易数据指示了对目标区块头中的状态数据的交易操作。那么，可以基于目标区块头指向的新状态树存储的所有状态数据，按照区块高度从低到高的顺序，依次执行各个第二区块头对应的交易数据所指示的交易操作，得到各个第二区块头的目标状态数据。可以理解的是，假设 m 等于 1，则第 m 个第二区块头的目标状态数据是基于目标区块头指向的新状态树存储的所有状态数据，执行第 m 个第二区块头对应的交易数据所指示的交易操作所得。第 $m+1$ 个第二区块头的目标状态数据是基于第 m 个第二区块头指向的新状态树存储的所有状态数据，执行第 $m+1$ 个第二区块头对应的交易数据所指示的交易操作所得。其中，第 m 个第二区块头指向的新状态树的构建可以参考下述描述。

另一种具体实现中，通过第二区块头所指向的当前状态树得到该第二区块头对应的目标状态数据的具体实施过程可以包括：目标设备可以遍历位于目标区块头之后的各个第二区块头，并确定当前遍历的当前第二区块头，在确定当前第二区块头之后，可以从该当前第二区块头指向的当前状态树中，复制得到当前第二区块头的目标状态数据。

25 假设当前第二区块头为第 m 个第二区块头，复制得到第 m 个第二区块头的目标状态数据的具体实施过程可以为如下描述。可以理解的是，所述第 m 个第二区块头所指向的当前状态树中所包括的树节点存储有用于指向下一层树节点的哈希指针，则目标设备可以根据当前状态树所包括的树节点中的哈希指针，确定多条节点路径，而每条节点路径所包括的最后一个节点（数据节点）中存储的数据即为目标状态数据。例如，假设将当前状态树中的树节点划分为根节点（可以是状态快照）、中间节点（中间节点可以包括拓展节点和分支节点）和叶子节点（数据节点），则从当前状态树中确定多条节点路径中的任一条节点路径的实现过程可以如下：目标设备可以根据根节点对应的第一哈希指针确定该第一哈希指针所指向的目标中间节点，在确定目标中间节点之后，可以根据该目标中间节点对应的第二哈希指针确定该第二哈希指针所指向的目标叶子节点，而根节点、目标中间节点和目标叶子节点所构成的路径即可以确定为节点路径。

35 S406，将第 m 个第二区块头对应的目标状态数据，存储至新叶子节点。

S407，基于存储目标状态数据后的新叶子节点，以及第 m 个第二区块头的前一区块头的新状态树，重构第 m 个第二区块头指向的新状态树。

在步骤 S406 和步骤 S407 中，在确定第 m 个第二区块头对应的目标状态数据之后，可以根据第 m 个第二区块头对应的目标状态数据、以及第 m 个第二区块头的前一区块头的新状态树，重构第 m 个第二区块头指向的新状态树。

40 例如，以 m 等于 1 为例进行说明，则第 m 个第二区块头的前一区块头的新状态树也就是目标区块头所指向的新状态树。目标设备可以将第 m 个第二区块头对应的目标状态数据，存储至新叶子节点，该新叶子节点也就是针对第 m 个第二区块头所指向的新状

态树中的数据节点。基于存储了目标状态数据的新叶子节点,以及第 m 个第二区块头的前一区块头的新状态树,重构第 m 个第二区块头指向的新状态树。通过上述方法,则可以重构每一个第二区块头指向的新状态树。其中,针对第二区块头指向的新状态树而言,任一新状态树可以与相应区块头的后一区块头所指向的新状态树存在节点复用。例如,第 m 个第二区块头指向的新状态树与第 m+1 个第二区块头指向的新状态树存在节点复用。

例如,假设目标区块头为区块头 100,则第 m 个第二区块头为区块头 101,图 5b 所示中的数据节点 51 中的 AAA 即是区块头 101 对应的目标状态数据,那么可以根据该数据节点计算一个哈希值,而该哈希值可以作为该数据节点的父节点中的哈希指针,该数据节点的父节点也就是拓展节点 52,则可以将该哈希值填充至拓展节点 52 中,同理,根据填充后的拓展节点 52 再次计算一个哈希值,并将该哈希值填充至分支节点 53 中字符“5”对应的位置。依次类推,可以计算得到区块头 101 中所存储的状态快照。而区块头 100 中对应的未更新过的历史状态数据所在的分支中的树节点可以被区块头 101 指向的状态树所复用。

S408,将本地空间中存储的各个当前状态树作为归档数据,归档至目标归档服务器,并在归档成功后,删除本地空间中的各个当前状态树。

其中,步骤S401和S408的具体实施方式可以参见上述实施例步骤S201和S204的具体描述,此处不再赘述。

在本申请实施例中,所构建的目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据,该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在复用的树节点,即该目标区块头指向的新状态树与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。并且,由于目标区块头之后各个第二区块头所指向的新状态树是基于目标区块头指向的新状态树所构建的,任一第二区块头所指向的新状态树可与该任一第二区块头的后一区块头所指向的新状态树存在节点复用,而与目标区块头之前的第一区块头所指向的当前状态树不存在节点复用。因此,在重构新状态树后,由于新状态树不存在节点复用,可以将本地空间中存储的各个当前状态树进行归档,删除本地空间中的各个当前状态树,从而释放了区块链节点的存储空间,减小了区块链节点中因大量历史的状态数据而给存储空间带来的压力,以有效保障区块链节点运行的稳定性。

为更好的理解本申请实施例所提供的数据归档方法,下面结合图 6a 和图 6b 进行进一步说明,图 6a 中所示的数据归档方法涉及了归档管理设备、目标设备(目标设备即是区块链节点,下述描述中以区块链节点为例进行描述)以及目标归档服务器,本申请实施例主要描述了归档管理设备、目标设备以及目标归档服务器之间的交互过程。如图 6a 所示,数据归档的具体实现过程包括以下步骤 s1-s12:

s1,当归档管理设备有针对目标区块头的归档需求时,归档管理设备可以向区块链节点发送归档数据请求,该归档数据请求中可以包括目标区块号,该目标区块号可以是上述描述中的目标区块头的区块头号。

s2,区块链节点判断目标区块号是否大于当前区块号、且小于或等于最新区块号。如果判断结果为目标区块号大于当前区块号、且小于或等于最新区块号,则继续以下操作。如果判断结果不是上述结果,则可以返回一个报错信息,以便于归档管理员可以根据该报错信息对归档数据请求进行调整。其中,该当前区块号可以是上述描述中的当前区块头的区块号,也可以是上述描述中的各个第一区块头中,区块高度最小的第一区块头的区块号,最新区块号可以是上述描述中的最新区块头的区块号。

s3,区块链节点将当前区块号作为开始区块号,将目标区块号减一作为结束区块号。

其中，开始区块号可以是上述描述中的各个第一区块头中，区块高度最小的第一区块头的区块号，开始区块号也就是当前区块号；结束区块号可以是上述描述中的各个第一区块头中，区块高度最大的第一区块头的区块号。

5 s4, 区块链节点通过在磁盘中复制目标区块的状态快照所指向的状态树(MPT)的各个树节点(即状态数据), 得到目标区块的状态数据的集合。

例如, 图 6b 所示为本申请实施例提供的一种数据归档系统的架构示意图。图 6b 中区块头 100 可以是指目标区块对应的目标区块头, 而该区块头 100 中的状态快照所指向的状态树可以如图 6b 中的 MPT, 目标设备可以将存储在磁盘中的目标区块的状态快照所指向的状态树加载到内存中, 以在内存中根据该状态树得到目标区块的状态数据的集合, 在数据归档过程中, 可以将数据归档中所要利用的, 且在磁盘中的数据加载到内存中进行相关的处理。例如, 区块链节点可以查找目标区块头指向的当前状态树, 以及位于目标区块头之前的各个第一区块头指向的当前状态树, 从而可以根据查找到的各个当前状态树和查找到的各个当前状态树之间的节点复用情况, 重构出目标区块头指向的新状态树, 目标区块头指向的新状态树中存储的所有状态数据即是目标区块的状态数据的集合。

15 s5, 区块链节点可以在目标区块的状态数据的集合基础上按区块号顺序依次执行目标区块后各个区块内的交易, 以得到目标区块后各个区块内的状态数据的集合, 该区块号顺序可以是指区块高度从低到高的顺序。

例如, 区块链节点可以获取位于目标区块之后的各个第二区块所对应的交易数据, 一个交易数据用于指示针对至少一个状态数据的交易操作。在获取到交易数据之后, 可以基于目标区块指向的新状态树存储的所有状态数据, 按照区块高度从低到高的顺序, 依次执行各个第二区块头对应的交易数据所指示的交易操作, 得到各个第二区块头的目标状态数据。而在得到各个第二区块头的目标状态数据之后, 区块链节点还可以基于目标区块指向的新状态树, 和位于目标区块头之后的各个第二区块头对应的目标状态数据, 25 重构出各个第二区块头指向的新状态树。

s6, 区块链节点可以把开始区块号、结束区块号、历史数据传送至目标归档服务器。

其中, 将历史数据作为归档数据, 还可以将本地空间中存储的各个当前状态树作为归档数据。

30 s7, 目标归档服务器接收到区块链节点发送的归档数据后, 对归档数据进行哈希处理, 以得到归档数据的归档数据哈希值。

s8, 目标归档服务器将归档时间戳、归档数据哈希值、归档数据的开始区块号、结束区块号、归档数据记录到目标归档服务器的归档数据列表中。

s9, 目标归档服务器将归档数据哈希值返回至区块链节点。

35 s10, 区块链节点接收目标归档服务器返回归档数据哈希值后, 删除本地历史数据。其中, 本地历史数据可以是指本地空间中的各个当前状态树。

s11, 区块链节点将目标归档服务器返回的归档数据哈希值、以及归档时间戳、开始区块号、结束区块号、目标归档服务器的服务器标识(例如服务器的 IP 地址)记录到本地空间中的归档记录列表。

其中, 该归档记录列表可以是如图 6b 中所示的归档记录列表。

40 s12, 区块链节点将当前区块号更新为目标区块号。

通过上述方法, 可以在归档数据成功归档至对应的归档服务器中之后, 删除本地空间中的各个当前状态树, 从而释放区块链节点的存储空间, 减小区块链节点中因大量历史状态数据而给存储空间带来的压力, 进而可以有效保障区块链节点运行的稳定性。

45 基于上述数据归档方法实施例的描述, 本申请实施例还公开了一种数据归档装置,

该数据归档装置可以是运行于上述所提及的目标设备中的一个计算机程序(包括程序代码)。该数据归档装置可以执行图 2 或图 4 所示的方法。请参见图 7, 所述数据归档装置可以运行如下单元:

5 查找单元 701, 配置为当存在针对目标区块头的归档任务时, 在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树, 其中, 任一区块头指向的当前状态树表示当存在所述归档任务时, 所述任一区块头指向的状态树;

10 第一重构单元 702, 配置为根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息, 重构出所述目标区块头指向的新状态树, 其中, 所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据;

第二重构单元 703, 配置为根据所述目标区块头指向的新状态树, 和位于所述目标区块头之后的各个第二区块头对应的目标状态数据, 重构出所述各个第二区块头指向的新状态树, 其中, 任一所述新状态树, 与相应区块头的后一区块头所指向的新状态树存在节点复用;

15 归档单元 704, 配置为将所述本地空间中存储的各个当前状态树作为归档数据, 归档至目标归档服务器, 并在归档成功后, 删除所述本地空间中的各个当前状态树。

20 在一些实施例中, 所述第一重构单元 702, 在根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息, 重构出所述目标区块头指向的新状态树时, 还配置为: 根据各个所述当前状态树之间的节点复用信息, 从所述各个当前树中确定至少一个复用树节点; 其中, 所述复用树节点表示在所述目标区块头指向的当前状态树和任一所述第一区块头指向的当前状态树中, 被复用的树节点; 复制所述至少一个复用树节点, 以及复制所述目标区块头指向的当前状态树中的各个树节点; 根据复制得到的树节点, 重构出所述目标区块头指向的新状态树。

25 在一些实施例中, 所述第二区块头的数量为 M 个, M 为正整数; 所述第二重构单元 703, 在根据所述目标区块头指向的新状态树, 和位于所述目标区块头之后的各个第二区块头对应的目标状态数据, 重构出所述各个第二区块头指向的新状态树时, 还配置为: 针对第 m 个第二区块头执行以下处理: 基于所述第 m 个第二区块头对应的目标状态数据, 生成新叶子节点; 其中, 所述第 m 个第二区块头对应的目标状态数据表示所述第 m 个第二区块头对应的多个状态数据中, 与所述第 m 个第二区块头的前一区块头对应的各个状态数据均不同的状态数据; 其中, $m \in [1, M]$; 将所述第 m 个第二区块头对应的目标状态数据, 存储至所述新叶子节点; 根据存储所述目标状态数据后的新叶子节点, 以及所述第 m 个第二区块头的前一区块头的新状态树, 重构所述第 m 个第二区块头指向的新状态树; 其中, 所述第 m 个第二区块头指向的新状态树和所述前一区块头指向的新状态树之间, 复用至少一个叶子节点; 其中, 当 m 等于 1 时, 所述前一区块头为所述目标区块头。

30 40 在一些实施例中, 所述第二重构单元 703, 还配置为: 获取位于所述目标区块头之后的各个第二区块头所对应的交易数据, 一个所述交易数据用于指示针对至少一个状态数据的交易操作; 根据所述目标区块头指向的新状态树存储的所有状态数据, 按照区块高度从低到高的顺序, 依次执行所述各个第二区块头对应的交易数据所指示的交易操作, 得到所述各个第二区块头的目标状态数据。

在一些实施例中, 所述第二重构单元 703, 还配置为: 遍历位于所述目标区块头之后的各个第二区块头, 并确定当前遍历的当前第二区块头; 从所述当前第二区块头指向的当前状态树中, 复制得到所述当前第二区块头的目标状态数据。

在一些实施例中, 所述归档单元 704, 在将所述本地空间中存储的各个当前状态树

作为归档数据, 归档至目标归档服务器时, 还配置为: 将所述各个第一区块头中, 区块高度最小的第一区块头的区块号作为开始区块号; 将所述各个第一区块头中, 区块高度最大的第一区块头的区块号作为结束区块号; 将所述本地空间中存储的各个当前状态树作为归档数据, 并根据所述归档数据、所述开始区块号以及所述结束区块号生成归档请求; 将所述归档请求发送至目标归档服务器; 其中, 所述目标归档服务器根据所述归档请求对所述归档数据进行哈希运算, 得到归档数据哈希值, 并在存储所述归档数据哈希值、所述归档数据、所述开始区块号以及所述结束区块号后, 返回所述归档数据哈希值。

在一些实施例中, 所述归档单元 704, 还配置为: 当接收到所述目标归档服务器发送的所述归档数据哈希值时, 确定所述归档数据被成功归档; 将所述归档数据哈希值、归档时间戳、所述开始区块号、所述结束区块号、以及所述目标归档服务器的服务器标识记录至所述本地空间的归档记录中。

在一些实施例中, 所述装置还包括查询单元 705, 所述查询单元 705, 配置为: 响应于针对指定区块头的的数据查询请求, 在所述本地空间的归档记录中读取所述结束区块号; 当所述指定区块头的区块号小于或等于所述结束区块号时, 根据所述本地空间内的归档记录中的归档数据哈希值、以及所述指定区块头的区块号生成指定数据查询请求; 将所述指定数据查询请求发送至所述目标归档服务器; 其中, 所述目标归档服务器根据所述指定数据查询请求携带的归档数据哈希值, 确定并返回所述指定区块头对应的状态数据, 所述归档数据哈希值指示所述归档数据中的当前状态树、以及所述指定区块头的区块号; 接收所述目标归档服务器返回的所述指定区块头对应的状态数据, 并输出所述指定区块头对应的状态数据。

在一些实施例中, 所述查询单元 705, 还配置为: 当所述指定区块头的指定区块号大于所述结束区块号时, 在所述本地空间中确定所述指定区块头所指向的新状态树、以及确定位于所述指定区块头之前的各个区块头所指向的新状态树; 根据确定的各个所述新状态树, 获取所述指定区块头对应的状态数据, 并输出获取到的所述状态数据。

在一些实施例中, 所述目标设备与多个归档服务器进行通信, 所述归档单元 704, 还配置为: 获取每个所述归档服务器的归档性能参数, 所述归档性能参数包括剩余存储数据量、传输带宽、安全系数中的至少一种, 所述安全系数用于衡量归档服务器被窃取数据的概率; 根据所述归档性能参数确定每个所述归档服务器的归档优先级, 并将最高归档优先级对应的归档服务器确定为所述目标归档服务器。

在一些实施例中, 所述归档性能参数包括传输带宽; 所述归档单元 704, 在用于获取每个归档服务器的归档性能参数时, 还配置为: 向每个所述归档服务器发送带宽探测请求, 其中, 所述带宽探测请求用于指示所述归档服务器返回指定数据量的目标数据;

接收每个所述归档服务器返回的所述目标数据, 并基于每个所述归档服务器返回所述目标数据所消耗的时长, 确定每个所述归档服务器的传输带宽。

在一些实施例中, 所述查找单元 701, 还配置为: 当存在针对所述目标区块头的归档任务时, 在区块头链中确定当前区块头, 所述当前区块头表示在所述本地空间中指向有状态树、且区块高度最小的区块头; 在所述区块头链中确定最新区块头, 所述最新区块头表示在所述本地空间中指向有状态树、且区块高度最大的区块头; 当所述目标区块头的区块高度大于所述当前区块头的区块高度、且小于所述最新区块头的区块高度时, 执行所述在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树的步骤; 其中, 所述第一区块头为所述当前区块头、以及所述当前区块头与所述目标区块头之间的各个区块头, 所述第二区块头为所述最新区块头、以及所述最新区块头与所述目标区块头之间的各个区块头。

根据本申请实施例, 图 2 或图 4 所示的方法所涉及的各个步骤均可以是由图 7 所示

的数据归档装置中的各个单元执行的。例如，图 2 所示的步骤 S201 由图 7 中所示的查找单元 701 来执行，S202 由图 7 中所示的第一重构单元 702 来执行，S203 由图 7 中所示的第二重构单元 703 来执行，S204 由图 7 中所示的归档单元 704 来执行。又如，图 4 所示的步骤 S401 由图 7 中所示的查找单元 701 来执行，步骤 S402 至步骤 S404 由图 7 中所示的第一重构单元 702 来执行，步骤 S405 至步骤 S407 由图 7 中所示的第二重构单元 703 来执行，S408 由图 7 中所示的归档单元 704 来执行。

根据本申请实施例，图 7 所示的数据归档装置中的各个单元可以分别或者全部合并为一个或若干个另外的单元来构成，或者其中的某个（些）单元还可以再拆分为功能上更小的多个单元来构成，这可以实现同样的操作，而不影响本申请实施例的技术效果的实现。上述单元是基于逻辑功能划分的，在实际应用中，一个单元的功能也可以是由多个单元来实现，或者多个单元的功能由一个单元实现。在本申请的其他实施例中，基于数据归档装置也可以包括其他单元，在实际应用中，这些功能也可以由其他单元协助实现，并且可以由多个单元协作实现。

根据本申请实施例，可以通过包括中央处理单元（CPU，Central Processing Unit），随机存取存储介质（RAM）、只读存储介质（ROM）等处理元件和存储元件。例如计算机的通用计算设备上运行能够执行如图 2 或图 4 中所示的相应方法所涉及的各步骤的计算机程序（包括程序代码），来构造如图 7 所示的数据归档装置，以及来实现本申请实施例的数据归档方法。所述的计算机程序可以记载于例如计算机可读记录介质上，并通过计算机可读记录介质装载于上述目标设备中，并在其中运行。

在本申请实施例中，当存在归档需求时，目标设备可以在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。进而可以根据查找到的各个当前状态树和查找到的各个当前状态树之间的节点复用情况，重构出目标区块头指向的新状态树，目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据的。可以基于目标区块头指向的新状态树，和位于目标区块头之后的各个第二区块头对应的目标状态数据，重构出各个第二区块头指向的新状态树。其中，任一新状态树，与相应区块头的后一区块头所指向的状态树存在节点复用。在重构出目标区块头指向的新状态树和各个第二区块头指向的新状态树之后，可以将本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在成功归档后，删除本地空间中的各个当前状态树。通过上述实施方法，可以实现对区块链节点中所存储的过期的状态数据（即当前状态树）进行归档处理，来释放区块链节点的存储空间，减轻区块链节点的存储压力，从而提升区块链节点的运行性能，进而保证区块链节点在运行过程中的稳定性。

基于上述数据归档方法实施例的描述，本申请实施例还公开了一种数据归档设备（即用于数据归档的计算机设备），该数据归档设备可以是上述实施例中的目标设备，请参见图 8，该数据归档设备至少可包括处理器 801、输入设备 802、输出设备 803 以及计算机可读存储介质 804。其中，数据归档设备内的处理器 801、输入设备 802、输出设备 803 以及计算机可读存储介质 804 可通过总线或其他方式连接。

所述计算机可读存储介质 804 是数据归档设备中的记忆设备，用于存放程序和数据。可以理解的是，此处的计算机可读存储介质 804 既可以包括数据归档设备的内置存储介质，当然也可以包括数据归档设备支持的扩展存储介质。计算机可读存储介质 804 提供存储空间，该存储空间存储了数据归档设备的操作系统。并且，在该存储空间中还存放了适于被处理器 801 加载并执行的一条或多条指令，这些指令可以是一个或一个以上的计算机程序（包括程序代码）。需要说明的是，此处的计算机可读存储介质可以是高速 RAM 存储器；还可以是至少一个远离前述处理器的计算机可读存储介质、所述处理器

可以称为中央处理单元 (CPU, Central Processing Unit), 是数据归档设备的核心以及控制中心, 适于被实现一条或多条指令, 具体加载并执行一条或多条指令从而实现相应的方法流程或功能。

5 在一种实施例中, 可由处理器 801 加载并执行计算机可读存储介质中存放的一条或多条第一指令, 以实现上述有关数据归档方法实施例中的方法的相应步骤; 计算机可读存储介质中的一条或多条第一指令由处理器 801 加载并执行如下操作:

10 当存在针对目标区块头的归档任务时, 在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树; 其中, 任一区块头指向的当前状态树表示当存在所述归档任务时, 所述任一区块头指向的状态树; 根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息, 重
15 构出所述目标区块头指向的新状态树; 其中, 所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据; 根据所述目标区块头指向的新状态树, 和位于所述目标区块头之后的各个第二区块头对应的目标状态数据, 重构出所述各个第二区块头指向的新状态树; 其中, 任一所述新状态树, 与相应区块头的后一区块头所指向的新状态树存在节点复用; 将所述本地空间中存储的各个当前状态树作为归档数据, 归档至目标
20 归档服务器, 并在归档成功后, 删除所述本地空间中的各个当前状态树。

在一些实施例中, 在根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息, 重构出所述目标区块头指向的新状态树时, 所述一条或多条指令由处理器 801 加载并执行: 根据各个所述当前状态树之间的节点复用信息, 从所述各个当前
25 树中确定至少一个复用树节点; 其中, 所述复用树节点表示在所述目标区块头指向的当前状态树和任一所述第一区块头指向的当前状态树中, 被复用的树节点; 复制所述至少一个复用树节点, 以及复制所述目标区块头指向的当前状态树中的各个树节点; 根据复制得到的树节点, 重构出所述目标区块头指向的新状态树。

30 在一些实施例中, 所述第二区块头的数量为 M 个, M 为正整数; 在根据所述目标区块头指向的新状态树, 和位于所述目标区块头之后的各个第二区块头对应的目标状态数据, 重构出所述各个第二区块头指向的新状态树时, 所述一条或多条指令由处理器 801 加载并执行: 针对第 m 个第二区块头执行以下处理: 基于所述第 m 个第二区块头对应的目标状态数据, 生成新叶子节点; 其中, 所述第 m 个第二区块头对应的目标状态数据表示所述第 m 个第二区块头对应的多个状态数据中, 与所述第 m 个第二区块头的前一
35 区块头对应的各个状态数据均不同的状态数据; 其中, $m \in [1, M]$; 将所述第 m 个第二区块头对应的目标状态数据, 存储至所述新叶子节点; 根据存储所述目标状态数据后的新叶子节点, 以及所述第 m 个第二区块头的前一区块头的新状态树, 重构所述第 m 个第二区块头指向的新状态树; 其中, 所述第 m 个第二区块头指向的新状态树和所述前一区块头指向的新状态树之间, 复用至少一个叶子节点; 其中, 当 m 等于 1 时, 所述前一区块头为所述目标区块头。

40 在一些实施例中, 所述一条或多条指令由处理器 801 加载并还执行: 获取位于所述目标区块头之后的各个第二区块头所对应的交易数据, 一个所述交易数据用于指示针对至少一个状态数据的交易操作; 根据所述目标区块头指向的新状态树存储的所有状态数据, 按照区块高度从低到高的顺序, 依次执行所述各个第二区块头对应的交易数据所指示的交易操作, 得到所述各个第二区块头的目标状态数据。

在一些实施例中, 所述一条或多条指令由处理器 801 加载并还执行: 遍历位于所述目标区块头之后的各个第二区块头, 并确定当前遍历的当前第二区块头; 从所述当前第二区块头指向的当前状态树中, 复制得到所述当前第二区块头的目标状态数据。

在一些实施例中, 在将所述本地空间中存储的各个当前状态树作为归档数据, 归档

至目标归档服务器时,所述一条或多条指令由处理器 801 加载并还执行:将所述各个第一区块头中,区块高度最小的第一区块头的区块号作为开始区块号;将所述各个第一区块头中,区块高度最大的第一区块头的区块号作为结束区块号;将所述本地空间中存储的各个当前状态树作为归档数据,并根据所述归档数据、所述开始区块号以及所述结束区块号生成归档请求;将所述归档请求发送至目标归档服务器;其中,所述目标归档服务器根据所述归档请求对所述归档数据进行哈希运算,得到归档数据哈希值,并在存储所述归档数据哈希值、所述归档数据、所述开始区块号以及所述结束区块号后,返回所述归档数据哈希值。

在一些实施例中,所述一条或多条指令由处理器 801 加载并还执行:当接收到所述目标归档服务器发送的所述归档数据哈希值时,确定所述归档数据被成功归档;将所述归档数据哈希值、归档时间戳、所述开始区块号、所述结束区块号、以及所述目标归档服务器的服务器标识记录至所述本地空间的归档记录中。

在一些实施例中,所述一条或多条指令由处理器 801 加载并还执行:响应于针对指定区块头的的数据查询请求,在所述本地空间的归档记录中读取所述结束区块号;当所述指定区块头的区块号小于或等于所述结束区块号时,根据所述本地空间内的归档记录中的归档数据哈希值、以及所述指定区块头的区块号生成指定数据查询请求;将所述指定数据查询请求发送至所述目标归档服务器;其中,所述目标归档服务器根据所述指定数据查询请求携带的归档数据哈希值,确定并返回所述指定区块头对应的状态数据,所述归档数据哈希值指示所述归档数据中的当前状态树、以及所述指定区块头的区块号;接收所述目标归档服务器返回的所述指定区块头对应的状态数据,并输出所述指定区块头对应的状态数据。

在一些实施例中,所述一条或多条指令由处理器 801 加载并还执行:当所述指定区块头的指定区块号大于所述结束区块号时,在所述本地空间中确定所述指定区块头所指向的新状态树、以及确定位于所述指定区块头之前的各个区块头所指向的新状态树;根据确定的各个所述新状态树,获取所述指定区块头对应的状态数据,并输出获取到的所述状态数据。

在一些实施例中,所述目标设备与多个归档服务器进行通信,所述一条或多条指令由处理器 801 加载并还执行:获取每个所述归档服务器的归档性能参数,所述归档性能参数包括剩余存储数据量、传输带宽、安全系数中的至少一种,所述安全系数用于衡量归档服务器被窃取数据的概率;根据所述归档性能参数确定每个所述归档服务器的归档优先级,并将最高归档优先级对应的归档服务器确定为所述目标归档服务器。

在一些实施例中,所述归档性能参数包括传输带宽;在获取每个所述归档服务器的归档性能参数时,所述一条或多条指令由处理器 801 加载并还执行:向每个所述归档服务器发送带宽探测请求,其中,所述带宽探测请求用于指示所述归档服务器返回指定数据量的目标数据;接收每个所述归档服务器返回的所述目标数据,并基于每个所述归档服务器返回所述目标数据所消耗的时长,确定每个所述归档服务器的传输带宽。

在一些实施例中,所述一条或多条指令由处理器 801 加载并还执行:当存在针对所述目标区块头的归档需求时,在区块头链中确定当前区块头;所述当前区块头是指:在所述本地空间中指向有状态树,且区块高度最小的区块头;当存在针对所述目标区块头的归档任务时,在区块头链中确定当前区块头,所述当前区块头表示在所述本地空间中指向有状态树、且区块高度最小的区块头;在所述区块头链中确定最新区块头,所述最新区块头表示在所述本地空间中指向有状态树、且区块高度最大的区块头;当所述目标区块头的区块高度大于所述当前区块头的区块高度、且小于所述最新区块头的区块高度时,执行所述在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标

区块头之前的各个第一区块头指向的当前状态树的步骤；其中，所述第一区块头为所述当前区块头、以及所述当前区块头与所述目标区块头之间的各个区块头，所述第二区块头为所述最新区块头、以及所述最新区块头与所述目标区块头之间的各个区块头。

5 在本申请实施例中，当存在归档需求时，目标设备可以在本地空间中查找目标区块头指向的当前状态树，以及位于目标区块头之前的各个第一区块头指向的当前状态树。进而可以根据查找到的各个当前状态树和查找到的各个当前状态树之间的节点复用情况，重构出目标区块头指向的新状态树，目标区块头指向的新状态树是独立存储目标区块头对应的所有状态数据的。可以基于目标区块头指向的新状态树，和位于目标区块头之后的各个第二区块头对应的目标状态数据，重构出各个第二区块头指向的新状态树。
10 其中，任一新状态树，与相应区块头的后一区块头所指向的状态树存在节点复用。在重构出目标区块头指向的新状态树和各个第二区块头指向的新状态树之后，可以将本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在成功归档后，删除本地空间中的各个当前状态树。通过上述实施方法，可以实现对区块链节点中所存储的过期的状态数据进行归档处理，来释放区块链节点的存储空间，减轻区块链节点的存储压力，从而提升区块链节点的运行性能，进而保证区块链节点在运行过程中的稳定性。
15

需要说明的是，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。数据归档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，使得该数据归档设备执行上述数据归档方法实施例图 2 或图 4 中所执行的步骤。
20

本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程，是可以通
过计算机程序来指令相关的硬件来完成，所述的程序可存储于一计算机可读存储介质
中，该程序在执行时，可包括如上述各方法的实施例的流程。其中，所述的存储介质可
为磁碟、光盘、只读存储记忆体（ROM, Read-Only Memory）或随机存储记忆体（RAM,
25 Random Access Memory）等。

以上所揭露的仅为本申请一种较佳实施例而已，当然不能以此来限定本申请之权利
范围，本领域普通技术人员可以理解实现上述实施例的全部或部分流程，并依本申请权
利要求所作的等同变化，仍属于发明所涵盖的范围。

权利要求书

1、一种数据归档方法，应用于计算机设备，包括：

当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树；

5 其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；

根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树；

10 其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；

根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树；

其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用；

15 将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

2、如权利要求 1 所述的方法，其中，所述根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树，包括：

20 根据各个所述当前状态树之间的节点复用信息，从所述各个当前树中确定至少一个复用树节点；

其中，所述复用树节点表示在所述目标区块头指向的当前状态树和任一所述第一区块头指向的当前状态树中，被复用的树节点；

复制所述至少一个复用树节点，以及复制所述目标区块头指向的当前状态树中的各个树节点；

25 根据复制得到的树节点，重构出所述目标区块头指向的新状态树。

3、如权利要求 1 所述的方法，其中，

所述第二区块头的数量为 M 个， M 为正整数；

所述根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树，包括：

30 针对第 m 个第二区块头执行以下处理：

基于所述第 m 个第二区块头对应的目标状态数据，生成新叶子节点；

其中，所述第 m 个第二区块头对应的目标状态数据表示所述第 m 个第二区块头对应的多个状态数据中，与所述第 m 个第二区块头的前一区块头对应的各个状态数据均不同的状态数据，其中， $m \in [1, M]$ ；

35 将所述第 m 个第二区块头对应的目标状态数据，存储至所述新叶子节点；

根据存储所述目标状态数据后的新叶子节点，以及所述第 m 个第二区块头的前一区块头的新状态树，重构所述第 m 个第二区块头指向的新状态树；

其中，所述第 m 个第二区块头指向的新状态树和所述前一区块头指向的新状态树之间，复用至少一个叶子节点；其中，当 m 等于 1 时，所述前一区块头为所述目标区块头。

40 4、如权利要求 1 所述的方法，其中，所述方法还包括：

获取位于所述目标区块头之后的各个第二区块头所对应的交易数据，一个所述交易数据用于指示针对至少一个状态数据的交易操作；

根据所述目标区块头指向的新状态树存储的所有状态数据，按照区块高度从低到高的顺序，依次执行所述各个第二区块头对应的交易数据所指示的交易操作，得到所述各

个第二区块头的目标状态数据。

5、如权利要求 1 所述的方法，其中，所述方法还包括：

遍历位于所述目标区块头之后的各个第二区块头，并确定当前遍历的当前第二区块头；

5 从所述当前第二区块头指向的当前状态树中，复制得到所述当前第二区块头的目标状态数据。

6、如权利要求 1 所述的方法，其中，所述将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，包括：

将所述各个第一区块头中，区块高度最小的第一区块头的区块号作为开始区块号；

10 将所述各个第一区块头中，区块高度最大的第一区块头的区块号作为结束区块号；

将所述本地空间中存储的各个当前状态树作为归档数据，并根据所述归档数据、所述开始区块号以及所述结束区块号生成归档请求；

将所述归档请求发送至目标归档服务器；

15 其中，所述目标归档服务器根据所述归档请求对所述归档数据进行哈希运算，得到归档数据哈希值，并在存储所述归档数据哈希值、所述归档数据、所述开始区块号以及所述结束区块号后，返回所述归档数据哈希值。

7、如权利要求 6 所述的方法，其中，所述方法还包括：

当接收到所述目标归档服务器发送的所述归档数据哈希值时，确定所述归档数据被成功归档；

20 将所述归档数据哈希值、归档时间戳、所述开始区块号、所述结束区块号、以及所述目标归档服务器的服务器标识记录至所述本地空间的归档记录中。

8、如权利要求 7 所述的方法，其中，所述方法还包括：

响应于针对指定区块头的查询请求，在所述本地空间的归档记录中读取所述结束区块号；

25 当所述指定区块头的区块号小于或等于所述结束区块号时，根据所述本地空间内的归档记录中的归档数据哈希值、以及所述指定区块头的区块号生成指定数据查询请求；

将所述指定数据查询请求发送至所述目标归档服务器；

30 其中，所述目标归档服务器根据所述指定数据查询请求携带的归档数据哈希值，确定并返回所述指定区块头对应的状态数据，所述归档数据哈希值指示所述归档数据中的当前状态树、以及所述指定区块头的区块号；

接收所述目标归档服务器返回的所述指定区块头对应的状态数据，并输出所述指定区块头对应的状态数据。

9、如权利要求 8 所述的方法，其中，所述方法还包括：

35 当所述指定区块头的指定区块号大于所述结束区块号时，在所述本地空间中确定所述指定区块头所指向的新状态树、以及确定位于所述指定区块头之前的各个区块头所指向的新状态树；

根据确定的各个所述新状态树，获取所述指定区块头对应的状态数据，并输出获取到的所述状态数据。

10、如权利要求 1-9 任一项所述的方法，其中，

40 所述目标设备与多个归档服务器进行通信；

所述方法还包括：

获取每个所述归档服务器的归档性能参数，所述归档性能参数包括剩余存储数据量、传输带宽、安全系数中的至少一种，所述安全系数用于衡量归档服务器被窃取数据的概率；

根据所述归档性能参数确定每个所述归档服务器的归档优先级，并将最高归档优先级对应的归档服务器确定为所述目标归档服务器。

11、如权利要求 10 所述的方法，其中，

所述归档性能参数包括传输带宽；

5 所述获取每个所述归档服务器的归档性能参数，包括：

向每个所述归档服务器发送带宽探测请求，其中，所述带宽探测请求用于指示所述归档服务器返回指定数据量的目标数据；

接收每个所述归档服务器返回的所述目标数据，并基于每个所述归档服务器返回所述目标数据所消耗的时长，确定每个所述归档服务器的传输带宽。

10 12、如权利要求 1 所述的方法，其中，所述方法还包括：

当存在针对所述目标区块头的归档任务时，在区块头链中确定当前区块头，所述当前区块头表示在所述本地空间中指向有状态树、且区块高度最小的区块头；

在所述区块头链中确定最新区块头，所述最新区块头表示在所述本地空间中指向有状态树、且区块高度最大的区块头；

15 当所述目标区块头的区块高度大于所述当前区块头的区块高度、且小于所述最新区块头的区块高度时，执行所述在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树的步骤；

其中，所述第一区块头为所述当前区块头、以及所述当前区块头与所述目标区块头之间的各个区块头，所述第二区块头为所述最新区块头、以及所述最新区块头与所述目标区块头之间的各个区块头。

20 13、一种数据归档装置，包括：

查找单元，配置为当存在针对目标区块头的归档任务时，在本地空间中查找所述目标区块头指向的当前状态树、以及位于所述目标区块头之前的各个第一区块头指向的当前状态树，其中，任一区块头指向的当前状态树表示当存在所述归档任务时，所述任一区块头指向的状态树；

25 第一重构单元，配置为根据查找到的各个所述当前状态树和各个所述当前状态树之间的节点复用信息，重构出所述目标区块头指向的新状态树，其中，所述目标区块头指向的新状态树独立存储所述目标区块头对应的所有状态数据；

30 第二重构单元，配置为根据所述目标区块头指向的新状态树，和位于所述目标区块头之后的各个第二区块头对应的目标状态数据，重构出所述各个第二区块头指向的新状态树，其中，任一所述新状态树，与相应区块头的后一区块头所指向的新状态树存在节点复用；

归档单元，配置为将所述本地空间中存储的各个当前状态树作为归档数据，归档至目标归档服务器，并在归档成功后，删除所述本地空间中的各个当前状态树。

35 14、一种计算机设备，所述计算机设备包括输入设备、输出设备，所述计算机设备还包括处理器和存储介质，所述处理器用于获取存储介质中存储的一条或多条指令，以执行如权利要求 1-12 中任一项所述的方法。

15、一种计算机可读存储介质，所述计算机可读存储介质存储有一条或多条指令，所述一条或多条指令运行时执行如权利要求 1-12 中任一项所述的方法。

40 16、一种计算机程序产品，所述计算机程序产品包括计算机程序，所述计算机程序被处理器执行时执行如权利要求 1-12 中任一项所述的方法。