

请求处理方法、装置、计算设备及存储介质

本申请要求于 2021 年 08 月 30 日提交中国专利局、申请号为 2021110037642、申请名称为“请求处理方法、装置、计算设备及存储介质”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。

5 技术领域

本申请涉及数据库技术领域，特别涉及请求处理技术。

背景技术

随着数据库技术的发展，传统的单机数据库逐渐不能适应大数据、云计算等业务场景，集群数据库变得越来越普及。目前，不论是 MySQL、PostgreSQL 等单机数据库，还是 IBM、Oracle 等集群数据库，均采用基于强同步锁机制的并发控制方法，强同步锁机制是指：申请锁的进程需要在集群内全部的计算设备中都获得相应级别的锁，才算申请锁成功，否则需要等待，只要集群内任一计算设备申请锁失败，或者等待超时，本进程都无法申请锁成功。

在集群数据库中，强同步锁机制要求集群内全部的计算设备都获得相应级别的锁，随着集群内计算设备的数量增加，申请锁的开销也显著增加，进而导致集群数据库系统的处理能力较差。因此，亟需一种能够提升集群数据库系统的处理能力的方法。

发明内容

本申请实施例提供了一种请求处理方法、装置、计算设备及存储介质，能够提升集群数据库系统的处理能力。该技术方案如下：

一方面，提供了一种请求处理方法，由集群数据库中的第一计算设备执行，所述方法包括：

响应于数据请求，确定所述集群数据库的锁同步模式；

在所述锁同步模式为弱同步模式的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求；

在所述锁同步模式不为弱同步模式的情况下，获取所述集群数据库中的至少一个第二计算设备对所述数据资源的加锁反馈信息，所述加锁反馈信息用于表征所述第二计算设备是否对所述数据资源加锁成功；在所获取的所述加锁反馈信息满足所述锁同步模式对应的目标条件的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求。

一方面，提供了一种请求处理方法，由集群数据库中的第二计算设备执行，所述方法包括：

响应于与数据请求关联的加锁请求，将所述加锁请求添加到加锁等待队

-2-

列中；所述加锁请求由所述集群数据库中的第一计算设备在锁同步模式不为弱同步模式的情况下发送；

处理所述加锁等待队列中的所述加锁请求，得到加锁反馈信息；

向所述第一计算设备发送所述加锁反馈信息。

5 一方面，提供了一种请求处理装置，所述装置包括：

确定模块，用于响应于数据请求，确定集群数据库的锁同步模式；

加锁执行模块，用于在所述锁同步模式为弱同步模式的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求；

10 第一获取模块，用于在所述锁同步模式不为弱同步模式的情况下，获取所述集群数据库中的至少一个第二计算设备对所述数据资源的加锁反馈信息，所述加锁反馈信息用于表征所述第二计算设备是否对所述数据资源加锁成功；

所述加锁执行模块，还用于在所获取的所述加锁反馈信息满足所述锁同步模式对应的目标条件的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求。

15 一方面，提供了一种请求处理装置，所述装置包括：

添加模块，用于响应于与数据请求关联的加锁请求，将所述加锁请求添加到加锁等待队列中；所述加锁请求由所述集群数据库中的第一计算设备在锁同步模式不为弱同步模式的情况下发送；

20 处理模块，用于处理所述加锁等待队列中的所述加锁请求，得到加锁反馈信息；

发送模块，用于向所述第一计算设备发送所述加锁反馈信息。

25 一方面，提供了一种计算设备，该计算设备包括一个或多个处理器和一个或多个存储器，该一个或多个存储器中存储有至少一条计算机程序，该至少一条计算机程序由该一个或多个处理器加载并执行以实现如上述任一种可能实现方式的请求处理方法。

一方面，提供了一种存储介质，该存储介质中存储有至少一条计算机程序，该至少一条计算机程序由处理器加载并执行以实现如上述任一种可能实现方式的请求处理方法。

30 一方面，提供一种计算机程序产品或计算机程序，所述计算机程序产品或所述计算机程序包括一条或多条程序代码，所述一条或多条程序代码存储在计算机可读存储介质中。计算设备的一个或多个处理器能够从计算机可读存储介质中读取所述一条或多条程序代码，所述一个或多个处理器执行所述一条或多条程序代码，使得计算设备能够执行上述任一种可能实施方式的请求处理方法。

35 本申请实施例提供的技术方案带来的有益效果至少包括：

通过对锁同步模式设置灵活的多个级别，在弱同步模式下，直接对数据请求对应的数据资源加锁，并执行该数据请求，能够保证数据请求优先推进，

无需等待第二计算设备的响应，而在不是弱同步模式下，则获取各个第二计算设备的加锁反馈信息，在所获取的加锁反馈信息满足目标条件的情况下，才对该数据资源加锁并执行该数据请求，支持在不同业务场景下选取不同的锁同步模式，从而保证集群数据库能够满足不同业务场景的处理需求，提升了集群数据库系统的处理能力。

附图说明

- 图 1 是本申请实施例提供的一种请求处理方法的实施环境示意图；
- 图 2 是本申请实施例提供的一种锁同步机制模型的原理性示意图；
- 图 3 是本申请实施例提供的一种请求处理方法的流程图；
- 10 图 4 是本申请实施例提供的一种请求处理方法的交互流程图；
- 图 5 是本申请实施例提供的一种加锁流程的原理性示意图；
- 图 6 是本申请实施例提供的一种释放锁流程的原理性示意图；
- 图 7 是本申请实施例提供的一种死锁检测流程的原理性示意图；
- 图 8 是本申请实施例提供的一种请求处理装置的结构示意图；
- 15 图 9 是本申请实施例提供的一种请求处理装置的结构示意图；
- 图 10 是本申请实施例提供的一种计算设备的结构示意图；
- 图 11 是本申请实施例提供的一种计算设备的结构示意图。

具体实施方式

为使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

本申请中术语“第一”“第二”等字样用于对作用和功能基本相同的相同项或相似项进行区分，应理解，“第一”、“第二”、“第 n”之间不具有逻辑或时序上的依赖关系，也不对数量和执行顺序进行限定。

本申请中术语“至少一个”是指一个或多个，“多个”的含义是指两个或两个以上，例如，多个第一位置是指两个或两个以上的第一位置。

在介绍本申请实施例之前，需要引入一些云技术领域内的基本概念。

云技术 (Cloud Technology)：是指在广域网或局域网内将硬件、软件、网络等系列资源统一起来，实现数据的计算、储存、处理和共享的一种托管技术，其是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台技术、应用技术等的总称，可以组成资源池，按需所用，灵活便利。云计算技术为云技术领域的重要支撑。技术网络系统的后台服务需要大量的计算、存储资源，如视频网站、图片类网站和更多的门户网站。伴随着互联网行业的高度发展和应用，将来每个物品都有可能存在自己的识别标志，都需要传输到后台系统进行逻辑处理，不同程度级别的数据将会分开处理，各类行业数据皆需要强大的系统后盾支撑，均能通过云计算来实现。

云存储 (Cloud Storage)：是在云计算概念上延伸和发展出来的一个新的

-4-

概念，分布式云存储系统（以下简称存储系统）是指通过集群应用、网格技术以及分布存储文件系统等功能，将网络中大量不同类型的存储设备（存储设备也称之为存储节点）通过应用软件或应用接口集合起来协同工作，共同对外提供数据存储和业务访问功能的一个存储系统。

- 5 数据库（Database）：简而言之可视为一种电子化的文件柜——存储电子文件的处所，用户可以对文件中的数据进行新增、查询、更新、删除等操作。所谓“数据库”是以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。

- 10 以下，对本申请实施例所涉及的术语进行解释说明。

 计算设备：即计算节点，是指数据库（通常指集群数据库，对于单机数据库来说单机设备本身就是一个计算设备）中用来处理用户特定计算请求，并主要负责执行用户请求的节点设备，亦可称为 SQL 引擎（SQLEngine），其中，SQL 的英文全称为 Structured Query Language，中文全称为结构化查询语言。

- 15 锁机制：在集群数据库运行期间，在不同的计算设备或者同一计算设备上，会存在同时有不同的进程或者线程需要修改同一种数据资源（例如：数据库、数据表、数据列、数据索引等）的现象，通过锁机制可以保证该数据资源在不同并发的进程或者线程中的修改正确性。其中，集群数据库可划分为中央集群数据库和分布式集群数据库，中央集群数据库是指拥有中心化控制模块的集群数据库，分布式集群数据库是指去中心化的集群数据库。

 强同步锁机制：即申请锁的进程需要在集群内全部的计算设备中都获得相应级别的锁，才算申请锁成功，否则需要等待，只要集群内任一计算设备申请锁失败，或者等待超时，本进程都无法申请锁成功。

- 20 强同步锁机制对于节点间（即计算设备间）的通信性能要求极高，容易因为单个计算设备的不响应或故障，导致集群内其他计算设备的执行被这个计算设备阻塞，对外显示整个集群的业务处理能力明显下降，例如，整个集群的业务处理被阻塞（被 Hang 住）。通常情况下，强同步锁机制主要用在单机数据库与中央集群数据库场景中。

- 30 弱同步锁机制：某些进程或者线程（例如，数据定义语言 DDL 请求的进程或线程）在执行过程中，如果需要对某类数据资源进行加锁，不用向其他计算设备申请锁，直接推进，其中，操作的正确性可由一种特定机制 Writing Fence（写屏障）来保证。在写屏障机制的保证下，DDL 请求的进程或线程无需申请锁就能够直接推进执行，并推进写屏障，使得其他业务进程或线程在操作数据的时候，都需要对比本次操作持有的写屏障版本是否与最新数据存储的写屏障版本一致，如果不一致的话，则需要回滚业务进程或线程对应的业务事务，以保证数据处理的正确性和一致性。
- 35

弱同步锁机制在某种意义上是保证 DDL 请求优先, 因为一旦写屏障推进后, 对应操作的数据操纵语言 DML 请求可能会被回滚掉, 即, 弱同步锁机制对于 DML 请求非常不友好。上述弱同步锁机制可以看作是某种类型的进程或者线程 (如 DDL 请求的进程) 优先执行; 这种弱同步模式, 需要采用辅助手段 (例如写屏障或者租期机制) 来保证整个机制推进的正确性, 这种弱同步模式虽然适合分布式集群数据库场景, 但是存在对用户的表现行为与传统数据库的执行行为不一致的情况, 且对于 DML 请求不友好。同时, 因为在弱同步模式下的 DDL 进程或者线程被默认为已经获得了锁, 这样会造成大量其他类型的 DML 进程或者线程所执行的事务被回滚掉。

- 5 半强同步锁机制: 即上述两种锁机制 (强同步锁机制和弱同步锁机制) 的一种折中实现, 由于上述两种锁机制每一种都有很明显的优缺点, 使用在分布式集群数据库中的表现都会遇到非常大的挑战, 因此半强同步锁机制的概念对于大规模分布式集群的高性能处理用户业务就变得非常必要了。即, 某一进程或者线程在需要针对某一种数据资源申请锁的时候, 也采取向所有计算设备发送锁申请的方式, 同时开始计时, 在等待一段时间内, 根据指定的目标数量的计算设备的反馈情况 (如加锁反馈信息), 来确定是否申请锁成功, 这种半强同步模式, 既不会因为某些计算设备的不响应而导致整个集群处理能力明显下降, 甚至整个集群处理能力作废, 也不会因为假设某个进程或者线程天生拥有锁而导致其他 DML (往往是用户业务事务, 是用来衡量数据库提供数据服务质量, 性能的重要指标) 请求遭受回滚的不公平待遇。

DDL (Data Definition Language, 数据定义语言) 请求: 即 DDL 语句操作, 是用来修改数据库中数据对象 (如: 数据库、数据表、数据列、数据索引等对象) 的定义的修改操作语句。

- 25 DML (Data Manipulate Language, 数据操纵语言) 请求: 即 DML 语句操作, 指用来修改数据库中存储的用户数据的操作语句, 通常对应用户业务事务, 也即 DML 请求通常是业务请求, 往往是用户业务事务, 是用来衡量数据库提供数据服务质量、性能的重要指标。

- 30 本申请实施例应用于集群数据库 (即集群数据库系统) 中, 集群数据库包括中央集群数据库和分布式集群数据库, 中央集群数据库是指拥有中心化控制模块的集群数据库, 在集群内配置有中心节点来部署全局的事务管理器和全局锁管理模块, 而分布式集群数据库是指去中心化的集群数据库, 在集群内没有中心节点来部署全局的事务管理器和全局锁管理模块, 由集群内所有计算节点通过某种算法达成全局事务与全局锁的实现。例如, 分布式集群数据库为分布式数据库系统、分布式大数据处理系统以及采用类似架构的分布式数据库管理系统。

在集群数据库中包括至少一个计算设备 (即计算节点), 每个计算设

-6-

备的数据库中可以存储有多个数据表，每个数据表可以用于存储一个或多个数据行（记录），其中，每一个数据行由一组按照特定位置索引排列的字段集组成，即数据字段列。其中，计算设备的数据库可以为任一类型的集群数据库，可以包括关系型数据库或者非关系型数据库中至少一项，例如 SQL
5 （Structured Query Language，结构化查询语言）数据库、MySQL、NoSQL、NewSQL（泛指各种新式的可拓展/高性能数据库）等，在本申请实施例中对数据库的类型不作具体限定。

在一些实施例中，本申请实施例还可以应用于基于区块链技术的数据库系统（以下简称为“区块链系统”），上述区块链系统在本质上属于一种去中心
10 化式的分布式数据库系统，采用共识算法保持区块链上不同计算设备所记载的账本数据一致，通过密码算法保证不同计算设备之间账本数据的加密传送以及不可篡改，通过脚本系统来拓展账本功能，通过网络路由来进行不同计算设备之间的相互连接。

在区块链系统中可以包括一条或多条区块链，区块链是一串使用密码学
15 方法产生的相关联的数据块，每一个数据块中包括一批次网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

区块链系统中计算设备之间可以组成点对点（Peer To Peer，P2P）网络，P2P 协议是一个运行在传输控制协议（Transmission Control Protocol，TCP）
20 协议之上的应用层协议。在区块链系统中，任一计算设备可以具备如下功能：
1）路由，计算设备具有的基本功能，用于支持计算设备之间的通信；2）应用，用于部署在区块链中，根据实际业务需求而实现特定业务，记录与特定业务相关的数据形成账本数据，在账本数据中携带数字签名以表示数据来源，将账本数据发送至区块链系统中的其他计算设备，供其他计算设备在验证账本数据来源以及完整性成功时，将账本数据添加至临时区块中，其中，应用
25 实现的业务可以包括钱包、共享账本、智能合约等；3）区块链，包括一系列按照先后的时间顺序相互接续的区块，新区块一旦加入到区块链中就不会再被移除，区块中记录了区块链系统中计算设备提交的账本数据。

在一些实施例中，每个区块中可以包括本区块存储交易记录的哈希值（本
30 区块的哈希值）以及前一区块的哈希值，各区块通过哈希值连接形成区块链，另，区块中还可以包括有生成区块时的时间戳等信息。

以下，对本申请实施例的系统架构进行说明。

图 1 是本申请实施例提供的一种请求处理方法的实施环境示意图。参见
图 1，以集群数据库为分布式集群数据库为例进行说明，分布式集群数据库
35 系统中包括应用客户端 101、网关服务器 102 以及分布式存储集群 103，在分布式存储集群 103 中包括一个或多个计算设备。

应用客户端 101 是指用户侧的终端上安装和运行的、能够发起数据请求

的客户端，该数据请求可以是 DDL 请求或者 DML 请求等，本申请实施例对此不进行具体限定。可选地，应用客户端 101 的类型包括但不限于：支付应用、社交应用、音视频应用、直播应用、购物应用、外卖应用或者打车应用等，本申请实施例不对应用客户端 101 的类型进行具体限定。在一些实施例中，用户侧的终端也称为用户设备、终端设备、用户终端、移动终端、智能终端、通信设备等。可选地，终端的设备类型包括：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、车载终端、智能家电、智能语音交互设备等，但并不局限于此。

应用客户端 101 与网关服务器 102 之间能够通过有线或无线的通信方式进行直接或间接地连接，本申请在此不做限制。

网关服务器 102 用于接收外部的数据请求，并将数据请求对应的读写事务分发至分布式存储集群 103。示意性地，用户在终端上登录应用客户端 101，触发应用客户端 101 生成 DDL 请求，例如，DDL 请求为修改数据表 A 的表名，进而应用客户端 101 调用分布式集群数据库系统提供的 API (Application Programming Interface, 应用程序编程接口) 将该 DDL 请求发送至网关服务器 102, 比如, 该 API 可以是 MySQL API (一种关系型数据库系统提供的 API)。又例如，在智慧交通场景下，新增停车位的请求为 DDL 请求，而查询已有停车位的请求为 DML 请求。

在一些实施例中，该网关服务器 102 可以与分布式存储集群 103 中的任何一个计算设备合并并在同一个物理机上，也即是，让分布式存储集群 103 中的某个计算设备充当网关服务器 102。

分布式存储集群 103 包括一个或多个计算设备，对于一部分数据请求如 DDL 请求来说，需要在执行 DDL 请求时，为该 DDL 请求添加一个全局的锁，以保证 DDL 请求对应的 DDL 事务的操作正确性，其中，处理 DDL 请求的计算设备称为第一计算设备，除了第一计算设备之外的其他计算设备称为第二计算设备。可选地，第一计算设备与第二计算设备的划分是针对不同 DDL 请求而言的，并且，第二计算设备的数量可以是一个或多个，本申请实施例不对分布式存储集群 103 中计算设备的数量进行具体限定，例如，计算设备的数量为 m 个，m 为大于或等于 1 的整数。

可选地，每个计算设备采用主备结构（也即是为一主多备集群），如图 1 所示，以计算设备为一主两备集群为例进行示意，每个计算设备中包括一个主机和两个备机，可选地，每个主机或备机都对应配置有代理 (Agent) 设备，代理设备可以与主机或备机是物理独立的，当然，代理设备还可以作为主机或备机上的一个代理模块，以计算设备 1 为例，计算设备 1 包括一个主数据库及代理设备（主 database+agent，简称主 DB+agent），此外还包括两备数据库及代理设备（备 database+agent，简称备 DB+agent）。

在一个示例性场景中，每个计算设备对应的主机或备机的数据库实例集

合称为一个 SET (集合), 例如, 假设某一计算设备为单机设备, 那么该计算设备的 SET 仅为该单机设备的数据库实例, 假设某一计算设备为一主两备集群, 那么该计算设备的 SET 为主机数据库实例以及两个备机数据库实例的集合。

- 5 在一些实施例中, 上述网关服务器 102 以及分布式存储集群 103 所构成的分布式集群数据库系统, 可以视为一种向用户终端提供数据服务的服务器, 该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 (Content Delivery Network, 内容分发网络)、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。

- 下面以集群数据库为分布式集群数据库为例, 介绍本申请实施例提供的一种锁同步机制模型, 图 2 是本申请实施例提供的一种锁同步机制模型的原理性示意图, 如 200 所示, 分布式集群数据库中包括第一计算设备和三个第二计算设备 A、B、C, 假设某一时刻第一计算设备接收到一个 DDL 请求, 该 DDL 请求需要在分布式集群数据库中进行加锁才能保证操作的正确性, 以强同步模式为例, 在强同步模式下, 需要在第一计算设备和第二计算设备 A、B、C 上均申请对该 DDL 请求对应的数据资源的锁, 在所有计算设备均申请锁成功之后, 执行 DDL 请求的处理逻辑, 即完成对应 DDL 事务, 然后第一计算设备释放本 DDL 请求的锁, 并通知第二计算设备 A、B、C 也释放本 DDL 请求的锁。下面将对整体的锁同步流程进行详细说明:

步骤一、第一计算设备开始处理 DDL 请求。

- 25 步骤二、第一计算设备进行 DDL 准备, 例如, 在本地向需要加锁的数据资源完成本地锁申请。

步骤三、第一计算设备通过加锁模块或加锁函数, 分别向第二计算设备 A、B、C 发送加锁申请, 第二计算设备 A、B、C 各自处理该加锁申请, 并向第一计算设备返回加锁反馈信息。

- 30 步骤四、第一计算设备执行 DDL 处理逻辑, 例如, 执行 DDL 请求对应的 DDL 事务。

步骤五、第一计算设备通过释放锁模块或释放锁函数, 分别向第二计算设备 A、B、C 发送释放锁申请, 第二计算设备 A、B、C 各自处理该释放锁申请, 并向第一计算设备返回释放锁反馈信息。

- 35 步骤六、第一计算设备结束 DDL 请求的处理进程 (简称为 DDL 进程)。
在上述过程中, 可以看出, 只需要在第一计算设备中增加加锁模块或加锁函数, 即可保证某一 DDL 请求的锁能够同步到本集群内的其他第二计算设备上, 比如, 每个第二计算设备接收加锁请求后, 带领在本地申请锁, 并将

加锁反馈信息返回给申请锁的远端节点（即第一计算设备），同时释放锁过程也是由锁持有端（即发起加锁请求的第一计算设备）先发送释放锁请求给本集群内的其他第二计算设备，每个第二计算设备接收到释放锁请求后，将对应的锁在本地释放，并将释放锁反馈信息返回给远端节点（即第一计算设备）。

5

在上述实施例中，简略介绍了锁同步机制模型的大体框架，而在本申请实施例中将对加锁过程中第一计算设备的各项操作进行说明，图 3 是本申请实施例提供的一种请求处理方法的流程图。参见图 3，该实施例由集群数据库中的第一计算设备执行，该实施例包括下述步骤：

10 301、第一计算设备响应于数据请求，确定集群数据库的锁同步模式。

可选地，集群数据库为中央集群数据库，或者为分布式集群数据库，均能够适用于该请求处理方法，本申请实施例仅以分布式集群数据库为例说明。集群数据库中包括第一计算设备和第二计算设备，第一计算设备是指用于处理该数据请求的计算设备，集群内除了第一计算设备以外的所有计算设备称为第二计算设备。

15

在一些实施例中，用户可以在终端上登录应用客户端，触发应用客户端生成该数据请求，可选地，该数据请求为数据定义语言 DDL 请求，或者，该数据请求为数据操纵语言 DML 请求，在本申请实施例中以数据请求为 DDL 请求为例说明，例如，DDL 请求为修改数据表 A 的表名。应用客户端在生成数据请求之后，可调用 API 将该数据请求发送至该第一计算设备。

20

在一些实施例中，第一计算设备接收该数据请求，解析该数据请求的头字段，当该头字段指示该数据请求为指定类型的请求如 DDL 请求时，确定该集群数据库的锁同步模式。

在一些实施例中，该锁同步模式可由数据库系统（或数据库引擎）配置的锁同步模式参数 lock_level_mode 来指示，锁同步模式参数 lock_level_mode 用于表征集群数据库当前所处的锁同步模式，该锁同步模式包括但不限于：强同步模式（Strong），半强同步模式（Middle）以及弱同步模式（Feak）。第一计算设备可从配置文件中查询锁同步模式参数 lock_level_mode 的取值，如果锁同步模式参数 lock_level_mode = Strong，确定该锁同步模式为强同步模式，如果锁同步模式参数 lock_level_mode = Middle，确定该锁同步模式为半强同步模式，如果锁同步模式参数 lock_level_mode = Feak，确定该锁同步模式为弱同步模式。

25

30

在一些实施例中，如上述的该锁同步模式参数 lock_level_mode 是被添加在该配置文件中的，在集群数据库启动时，集群内每个计算设备的数据库引擎都会载入该配置文件。

35

在一些实施例中，该锁同步模式参数 lock_level_mode 还可以在整个集群运行时由计算设备随机进行修改，计算设备对锁同步模式参数

lock_level_mode 修改完毕后，会自动将修改后的锁同步模式参数 lock_level_mode 同步到集群内所有计算设备上。

- 5 在一些实施例中，如果第一计算设备确定的该锁同步模式为弱同步模式，则执行下述步骤 302，如果第一计算设备确定的该锁同步模式为强同步模式或者半强同步模式，则执行下述步骤 303-304。

在另一些实施例中，第一计算设备在本地向需要加锁的数据资源完成本地锁申请后，再进入到加锁流程中，接着，判断该锁同步模式是否为弱同步模式，如果该锁同步模式是弱同步模式，则执行下述步骤 302，如果该锁同步模式不是弱同步模式，则执行下述步骤 303-304。

- 10 302、在该锁同步模式为弱同步模式的情况下，第一计算设备对该数据请求对应的数据资源加锁，执行该数据请求。

- 15 如果该锁同步模式是弱同步模式，意味着第一计算设备不用向其他的第二计算设备申请锁，就能够直接推进当前的数据请求如 DDL 请求，其中，操作的正确性可由一种特定机制 Writing Fence（写屏障）来保证。通过上述机制，能够保证在多核多线程环境中，在某一个时间点上，只能有一个线程进入临界区代码，从而保证临界区中操作数据的一致性。

需要说明的是，本申请实施例所涉及的“锁”，可以理解为内存中的一个内存状态参数，例如，该内存状态参数可以是整型数，该内存状态参数与数据资源关联，代表是否对所关联的数据资源加锁。

- 20 可选地，该内存状态参数拥有两种状态：空闲状态和上锁状态。加锁时，判断锁是否空闲，如果空闲，修改为上锁状态，同时自身转变为锁持有端，返回锁申请成功；如果已经上锁，则返回锁申请失败。解锁时，则把上锁状态修改为空闲状态。

- 25 可选地，该内存状态参数可拥有两种以上状态：空闲状态、读锁（即共享锁，S 锁）、写锁（即排他锁，X 锁）和更新锁，其中，读锁、写锁和更新锁用于区分在上锁状态下其他 DML 请求对加锁的数据资源能够执行的操作，比如，某一 DDL 请求对用户表 T1 加锁 T1_lock，如果 T1_lock 为读锁，那么其他 DML 请求虽然无法对用户表 T1 执行写操作，但仍然能够正常读取用户表 T1 中的数据项。

- 30 在弱同步模式下，由于写屏障机制的保证，第一计算设备在本地锁申请成功后，即可直接认为已经获得锁，即第一计算设备转变为锁持有端，无需向其他的各个第二计算设备发送加锁申请，并可确定对该数据请求对应的数据资源加锁成功，加锁流程结束，然后执行该数据请求对应的数据库事务即可，例如，通过 DDL 线程执行 DDL 请求对应的 DDL 事务。

- 35 303、在该锁同步模式不为弱同步模式的情况下，第一计算设备获取该集群数据库中的至少一个第二计算设备对该数据资源的加锁反馈信息，该加锁反馈信息用于表征该第二计算设备是否对该数据资源加锁成功。

如果锁同步模式不是弱同步模式，那么锁同步模式可能是强同步模式或者半强同步模式，在上述强同步模式或者半强同步模式下，第一计算设备需要向远端的各个第二计算设备申请锁，只是两种模式判定锁申请成功的目标条件不同，在强同步模式下需要所有的第二计算设备均申请锁成功，才认为
5 对该数据资源的锁全局申请成功，而在半强同步模式下只需要不少于目标数量的第二计算设备申请锁成功，就认为对该数据资源的锁全局申请成功。

在一些实施例中，第一计算设备向各个第二计算设备申请锁，以获取各个第二计算设备的加锁反馈信息，可选地，该加锁反馈信息包括：加锁成功和加锁失败，加锁失败的原因通常是多种多样的，比如，等待超时、陷入死
10 锁等等。

在一些实施例中，上述向各个第二计算设备申请锁的过程包括：第一计算设备向该至少一个第二计算设备发送与该数据请求关联的加锁请求；第一计算设备接收该至少一个第二计算设备基于该加锁请求返回的该加锁反馈信息。

15 可选地，第一计算设备和每个第二计算设备上，均可在后台创建一个消息处理线程，该消息处理线程可专门用于处理各种与锁机制相关的消息，比如，监听其他计算设备的加锁请求、释放锁请求、加锁反馈信息、释放锁反馈信息等，以及时收发各种与锁机制相关的消息，保障计算设备间的通信性能。

20 基于上述情况，第一计算设备在发送加锁请求时，可以通过该第一计算设备上的消息处理线程，分别向该至少一个第二计算设备上的消息处理线程发送该加锁请求；同理，第一计算设备在接收加锁反馈信息时，可以通过该第一计算设备上的消息处理线程，接收该至少一个第二计算设备上的消息处理线程返回的该加锁反馈信息。其中，第二计算设备的锁申请流程将在下个
25 实施例中详述，这里不展开描述。

在一些实施例中，由于第一计算设备会向集群内每个第二计算设备均发送该加锁请求，而每个第二计算设备的加锁反馈信息返回至第一计算设备的时间可能并不相同，在一些示例性场景中，如果第一计算设备上该数据请求的处理进程（如 DDL 进程）崩溃，那么在该 DDL 进程崩溃后，第一计算设备
30 对于其接收到的加锁反馈信息，应当视情况来及时指示加锁成功的第二计算设备释放掉对应的锁，避免阻塞在该数据资源上的其他 DML 进程，以优化集群数据库的处理性能。

针对上述情况，第一计算设备可以响应于接收到任一第二计算设备的加锁反馈信息，在该数据请求处于活跃状态的情况下，通过该消息处理线程将
35 该加锁反馈信息发送至该数据请求的锁申请线程；在该数据请求不处于活跃状态，且该加锁反馈信息表征加锁成功的情况下，向该第二计算设备发送释放锁请求，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数

据资源。

可选地，第一计算设备在通过消息处理线程接收到任一加锁反馈信息之后，判断本地的该数据请求是否活跃，以 DDL 请求为例，可判断 DDL 请求对应的 DDL 进程是否存在，由于有可能存在多个 DDL 请求复用同一 DDL 进程的情况，导致在当前 DDL 请求已失败（比如不满足目标条件，DDL 事务被回滚）的情况下，DDL 进程仍有可能存在，因此针对 DDL 进程判断 DDL 请求是否活跃可能存在一定误判情况。可选地，还可直接判断 DDL 请求对应的 DDL 事务是否处于活跃状态，这样能够有效降低上述误判情况，提高判断准确度。

- 10 如果该数据请求不活跃，即该数据请求不处于活跃状态，判断该加锁反馈信息是否表征加锁成功，如果该加锁反馈信息表征加锁成功，说明本地的 DDL 请求已经结束，但在远端的第二计算设备上却已申请成功了对应数据资源的锁，此时需要指示该第二计算设备及时释放掉对应数据资源的锁，因此，第一计算设备可通过消息处理线程向该第二计算设备的消息处理线程发送释放锁请求，以指示该第二计算设备释放该数据请求对应的数据资源，例如，
- 15 将该数据资源对应的锁的内存状态参数修改为空闲状态；此外，如果该加锁反馈信息表征加锁失败，代表远端不需要及时释放的数据资源，那么无需进行任何处理。

- 20 如果该数据请求活跃，即该数据请求处于活跃状态，那么可通过该消息处理线程将该加锁反馈信息发送至该数据请求的锁申请线程，以使得锁申请线程记录本次接收到的加锁反馈信息，并基于下述步骤 304 来判断是否符合目标条件，从而方便了该锁申请线程及时统计接收到的加锁反馈信息是否符合当前锁同步模式对应的目标条件。

- 25 在一些实施例中，第一计算设备还可以在向各个第二计算设备发送加锁请求之后开始计时，这一计时统计的是对各个加锁反馈信息的等待时长，并可以在配置文件中设置强同步模式对应的第一等待阈值和半强同步模式对应的第二等待阈值。第一等待阈值用于表征计算设备在强同步模式下对加锁反馈信息的最大等待时长，如果在强同步模式下的等待时长超过第一等待阈值，那么可以直接视为本次加锁失败，并向申请锁成功的各个第二计算设备发送释放锁请求，以指示第二计算设备释放该数据请求对应的数据资源，然后结束加锁流程，其中，该第一等待阈值为任一大于 0 的数值，例如第一等待阈值为 15 秒。第二等待阈值用于表征计算设备在半强同步模式下对加锁反馈信息的最大等待时长，如果在半强同步模式下的等待时长超过第二等待阈值，那么可以直接视为本次加锁失败，并向申请锁成功的各个第二计算设备发送释放锁请求，以指示第二计算设备释放该数据请求对应的数据资源，然后结束加锁流程，其中，该第二等待阈值为任一大于 0 的数值，例如第二等待阈值为 10 秒，需要说明的是，第一等待阈值和第二等待阈值可以相同或者不同，
- 30
- 35

本申请实施例对此不作限定。

在上述过程中，通过配置第一等待阈值和第二等待阈值，并在发送加锁请求后开始计时，能够避免第一计算设备长时间地等待其他第二计算设备的加锁反馈信息，比如加锁反馈信息传输中发生丢包情况，那么此时第一计算设备在强同步模式下的等待时长最多达到第一等待阈值，在半强同步模式下的等待时长最多达到第二等待阈值，就可以推进处理后续其他的数据请求，能够提升集群数据库的性能。

304、在所获取的该加锁反馈信息满足该锁同步模式对应的目标条件的情况下，第一计算设备对该数据请求对应的数据资源加锁，执行该数据请求。
锁同步模式对应的目标条件，是用于根据所获取的加锁反馈信息衡量是否对数据请求对应的数据资源加锁的条件，通常情况下，强同步模式对应的目标条件比半强同步模式对应的目标条件要求更高。

在一些实施例中，如果集群数据库未设置超时机制，在该锁同步模式为强同步模式的情况下，该目标条件为该至少一个第二计算设备的加锁反馈信息均表征加锁成功；在该锁同步模式为半强同步模式的情况下，该目标条件为该至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。

在一些实施例中，如果集群数据库设置了超时机制，那么除了需要满足上述基本条件之外，还需要保证在强同步模式下的等待时长不超过第一等待阈值，在半强同步模式下的等待时长不超过第二等待阈值，也即是说，在该锁同步模式为强同步模式的情况下，该目标条件为等待时长不超过第一等待阈值，且该至少一个第二计算设备的加锁反馈信息均表征加锁成功；在该锁同步模式为半强同步模式的情况下，该目标条件为等待时长不超过第二等待阈值，且该至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。其中，该第一等待阈值和第二等待阈值均为大于 0 的数值，该第一等待阈值和第二等待阈值均可以添加至配置文件中，并在集群数据库启动时，由每个计算设备的数据库引擎载入。

在一些实施例中，该目标数量可以由数据库引擎所配置的锁通过概率参数 `lock_succ_node_num` 来指示，锁通过概率参数 `lock_succ_node_num` 用于表征集群数据库在半强同步模式下总共需要多少计算设备申请锁成功才能视为全局申请锁成功，可选地，锁通过概率参数 `lock_succ_node_num` 等于目标数量加一所得的数值，其中，目标数量代表总共需要多少第二计算设备申请锁成功，而目标数量加上第一计算设备自身即等于集群内所需申请锁成功的总计算设备数量。锁通过概率参数 `lock_succ_node_num` 的有效取值范围为：大于或等于 1、且小于或等于集群内所有计算设备的数量，若锁通过概率参数 `lock_succ_node_num` 小于 1 或者大于集群内所有计算设备的数量，则说明此时集群的锁通过概率参数 `lock_succ_node_num` 不处于有效取值范围内，那么

可以视为一律采用强同步模式或弱同步模式(由管理员设定),以在锁通过概率参数 lock_succ_node_num 不合法的情况下保证集群能够正常运转,例如,一律采用强同步模式。

5 可选地,该锁通过概率参数 lock_succ_node_num 也可添加在配置文件中,并在集群数据库启动时,集群内每个计算设备的数据库引擎都会载入该配置文件。

10 可选地,该锁通过概率参数 lock_succ_node_num 还可在整个集群运行时由计算设备随机进行修改,计算设备对锁通过概率参数 lock_succ_node_num 修改完毕后,会自动将修改后的锁通过概率参数 lock_succ_node_num 同步到集群内所有计算设备上。

在一些实施例中,第一计算设备在该锁同步模式为半强同步模式的情况下,获取该目标数量;在该目标数量小于一或者大于该集群数据库中全部第二计算设备的数量的情况下,将该锁同步模式切换为该强同步模式。

在上述过程中,第一计算设备可以从配置文件中查询该锁通过概率参数
15 lock_succ_node_num 的取值,并将该锁通过概率参数 lock_succ_node_num 的取值减一所得的数值确定为该目标数量。进一步地,判断该目标数量是否合法,即,如果该目标数量大于或等于 1、且小于或等于该集群数据库中全部第二计算设备的数量,说明该目标数量是合法的,那么该锁同步模式按照半强同步模式来判断目标条件,而如果该目标数量小于 1、或者大于该集群数
20 据库中全部第二计算设备的数量,说明该目标数量不合法,那么可以强制将该锁同步模式切换为强同步模式。在一些实施例中,也可以强制将锁同步模式切换至弱同步模式,本申请实施例对此不作限定。

在另一些实施例中,第一计算设备在配置文件中查询得到该锁通过概率参数 lock_succ_node_num 的取值之后,判断该锁通过概率参数
25 lock_succ_node_num 的取值是否合法,如果锁通过概率参数 lock_succ_node_num 合法(即大于或等于 1、且小于或等于集群内所有计算设备的数量),那么代表该目标数量也是合法的,如果锁通过概率参数 lock_succ_node_num 不合法(即小于 1、或者大于集群内所有计算设备的数量),那么代表该目标数量也不合法,需要强制将锁同步模式切换为强同步模式或者弱同步模式,在本申请实施例中,仅以将锁同步模式切换为强同步模式
30 为例进行说明。

在一些实施例中,如果锁同步模式参数 lock_level_mode $\neq$ Feak,代表锁同步模式不是弱同步模式,此时通过消息处理线程向各个第二计算设备发送加锁申请,并开始计时,通过消息处理线程接收各个第二计算设备返回的加
35 锁反馈信息。可选地,如果锁同步模式参数 lock_level_mode = Strong,确定该锁同步模式为强同步模式,那么判断是否所有第二计算设备的加锁反馈信息均表征加锁成功,此时可分三种情况进行讨论:1)如果尚未等待超时,即

等待时长不超过第一等待阈值，且接收到的所有第二计算设备的加锁反馈信息均表征加锁成功，则符合目标条件，视为全局加锁成功；2) 如果接收到的任一第二计算设备的加锁反馈信息表征加锁失败，则不符合目标条件，视为全局加锁失败；3) 如果等待时长超过第一等待阈值时仍存在第二计算设备未返回加锁反馈信息，代表已经等待超时，则不符合目标条件，视为全局加锁失败。

5 可选地，如果锁同步模式参数 `lock_level_mode = Middle`，确定锁同步模式为半强同步模式，从配置文件中读取锁通过概率参数 `lock_succ_node_num`，判断锁通过概率参数 `lock_succ_node_num` 是否合法，如果锁通过概率参数
10 `lock_succ_node_num` 大于或等于 1、且小于或等于集群内所有计算设备的数量，即锁通过概率参数 `lock_succ_node_num` 合法，则按照半强同步模式继续处理，如果锁通过概率参数 `lock_succ_node_num` 小于 1、或者大于集群内所有计算设备的数量，即按照强同步模式继续处理。在半强同步模式下，同样可分三种情况进行讨论：A) 如果尚未等待超时，即等待时长不超过第二等
15 待阈值，且接收到的加锁反馈信息表征加锁成功的第二计算设备的数量大于或等于目标数量，则符合目标条件，视为全局加锁成功；在另一种可能实施方式中，如果接收到的加锁反馈信息表征加锁成功的第二计算设备的数量加一所得的数值大于或等于锁通过概率参数 `lock_succ_node_num`，同样判定符合目标条件，视为全局加锁成功；B) 如果接收到的加锁反馈信息表征加锁失
20 败的第二计算设备的数量大于集群内全部第二计算设备的数量减去该目标数量所得的数值，则不符合目标条件，视为全局加锁失败；在另一种可能实施方式中，如果接收到的加锁反馈信息表征加锁失败的第二计算设备的数量加一所得的数值大于集群内所有计算设备减去锁通过概率参数
25 `lock_succ_node_num` 所得的数值，则同样判定不符合目标条件，视为全局加锁失败；C) 如果等待时长超过第二等待阈值时仍存在较多的第二计算设备还未返回加锁反馈信息，导致既不满足情况 A) 也不满足情况 B)，代表已经等待超时，则不符合目标条件，视为全局加锁失败。

30 在一些实施例中，如果判定所获取的加锁反馈信息符合目标条件，则执行与上述步骤 302 类似的操作，即对该数据请求对应的数据资源加锁，并执行该数据请求，这里不做赘述。

在一些实施例中，在所获取的加锁反馈信息不符合该目标条件或者该数据请求执行完毕的情况下，向该至少一个第二计算设备发送释放锁请求，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数据资源。

35 在上述过程中，如果所获取的加锁反馈信息不符合目标条件，那么此时全局加锁失败，但是很有可能当前数据请求在某些第二节点设备上已经申请锁成功了，因此第一计算设备需要及时向加锁反馈信息表征加锁成功的各个第二计算设备发送释放锁请求，从而及时释放掉对应数据资源的锁。可选地，

还可以直接向所有第二计算设备广播该释放锁请求，加锁成功的第二计算设备会释放掉对应数据资源的锁，但加锁失败的第二计算设备不作任何处理。

5 可选地，第一计算设备无需等待各个第二计算设备返回释放锁反馈信息，这是由下述实施例中的探活机制来保证的，用以提高系统的请求处理效率，当然，第一计算设备也可以等待各个第二计算设备返回释放锁反馈信息之后才完成加锁流程，以及时探测是否由于网络丢包导致某些第二计算设备未能收到释放锁请求，本申请实施例对此不进行限定。另一种场景下，如果所获取的加锁反馈信息符合目标条件，在第一计算设备全局加锁成功，并对数据请求如 DDL 请求执行完毕之后，同样需要通知其他的各个第二计算设备及时
10 释放掉对应数据资源的锁，释放锁请求的发送方式与上述情况类似，这里不做赘述。

上述所有可选技术方案，能够采用任意结合形成本公开的可选实施例，在此不再一一赘述。

本申请实施例提供的方法，通过对锁同步模式设置灵活的多个级别，在
15 弱同步模式下，直接对数据请求对应的数据资源加锁，并执行该数据请求，能够保证数据请求优先推进，无需等待第二计算设备的响应，而在不是弱同步模式下，则获取各个第二计算设备的加锁反馈信息，在所获取的加锁反馈信息符合目标条件的情况下，才对该数据资源加锁并执行该数据请求，使得在不同业务场景下选取不同的锁同步模式，从而保证集群数据库能够满足不
20 同业务场景的处理需求，提升了集群数据库系统的处理能力。

在上述实施例中，介绍了在请求处理过程中，第一计算设备如何发起加锁请求并基于加锁反馈信息来判断是否符合目标条件，而在本申请实施例中，针对第一计算设备和第二计算设备在加锁过程中的消息交互，以及各自的本地加锁逻辑进行详细说明。
25

图 4 是本申请实施例提供的一种请求处理方法的交互流程图，如图 4 所示，该实施例应用于集群数据库，集群数据库中包括第一计算设备和至少一个第二计算设备，第一计算设备是用于处理该数据请求的计算设备，集群内除了第一计算设备以外的所有计算设备称为第二计算设备，该实施例包括下
30 述步骤：

401、应用客户端向集群数据库中的第一计算设备发送数据请求。

可选地，集群数据库为中央集群数据库，或者为分布式集群数据库，均能够适用于该请求处理方法，本申请实施例仅以分布式集群数据库为例说明。

35 在一些实施例中，用户在终端上登录应用客户端，触发应用客户端生成该数据请求，可选地，该数据请求为 DDL 请求，或者，该数据请求为 DML 请求，在本申请实施例中以数据请求为 DDL 请求为例说明，例如，DDL 请求为修改数据表 A 的表名。应用客户端在生成数据请求之后，可调用 API 将

该数据请求发送至该第一计算设备。

402、第一计算设备响应于该数据请求，确定该集群数据库的锁同步模式。

上述步骤 402 与上述步骤 301 类似，这里不做赘述。

403、在该锁同步模式为弱同步模式的情况下，第一计算设备对该数据请求
5 求对应的数据资源加锁，执行该数据请求。

上述步骤 403 与上述步骤 302 类似，这里不做赘述。

404、在该锁同步模式不为弱同步模式的情况下，第一计算设备向该集群数据库中的至少一个第二计算设备发送与该数据请求关联的加锁请求。

如果锁同步模式不是弱同步模式，那么锁同步模式可能是强同步模式或者半强同步模式，在上述强同步模式或者半强同步模式下，第一计算设备均
10 需要向远端的各个第二计算设备申请锁，只是两者判定锁申请成功的目标条件不同，在强同步模式下需要所有的第二计算设备均申请锁成功，才认为对该数据资源的锁全局申请成功，而在半强同步模式下只需要不少于目标数量的第二计算设备申请锁成功，就认为对该数据资源的锁全局申请成功。

15 在一些实施例中，第一计算设备向各个第二计算设备申请锁，以获取各个第二计算设备的加锁反馈信息，可选地，该加锁反馈信息包括：加锁成功和加锁失败，加锁失败的原因通常是多种多样的，比如，等待超时、陷入死锁等等。

20 可选地，第一计算设备和每个第二计算设备上，均可在后台创建一个消息处理线程，该消息处理线程可专门用于处理各种与锁机制相关的消息，比如，监听其他计算设备的加锁请求、释放锁请求、加锁反馈信息、释放锁反馈信息等，以及时收发各种与锁机制相关的消息，保障计算设备间的通信性能。

25 基于上述情况，第一计算设备在发送加锁请求时，可以通过该第一计算设备上的消息处理线程，分别向该至少一个第二计算设备上的消息处理线程发送该加锁请求。

405、对任一第二计算设备，响应于与该数据请求关联的加锁请求，将该加锁请求添加到加锁等待队列中。

30 其中，该加锁请求由该集群数据库中的第一节点设备在锁同步模式不为弱同步模式的情况下发送。

35 在一些实施例中，第二计算设备通过消息处理线程接收该加锁请求；通过该消息处理线程启动与该加锁请求对应的锁代理线程；通过该锁代理线程将该加锁请求添加到该加锁等待队列中。可选地，每个加锁请求对应于一个锁代理线程，以降低每个加锁请求的等待时延，可选地，多个加锁请求可复用同一个锁代理线程，此时后到达的加锁请求需要存储在加锁等待队列中，由锁代理线程按照顺序处理该加锁等待队列中存储的各个加锁请求。

在上述过程中，通过在第一计算设备和第二计算设备上均配置专门负责

收发与锁相关的消息的消息处理线程，使得在申请锁的过程中，能够由消息处理线程启动锁代理线程，来维护每个第二计算设备上远端锁的生命周期。此外，有可能会出现第二计算设备的加锁反馈信息到达第一计算设备时，第一计算设备上的锁申请线程已经退出的情况，此时由于配置了消息处理线程，能够由第一计算设备的消息处理线程通过下述步骤 408 中的操作，判断当前数据请求是否活跃并判断加锁反馈信息是否表征加锁成功，及时在数据请求不活跃的情况下，向加锁成功的第二计算设备返回释放锁请求，触发释放远端已经申请成功的锁，以达到不阻塞第二计算设备上业务的目的。

406、第二计算设备处理该加锁等待队列中的该加锁请求，得到加锁反馈信息，该加锁反馈信息用于表征该第二计算设备是否对该数据资源加锁成功。

在一些实施例中，第二计算设备启动本地的锁代理线程，并调用本地的申请锁逻辑，来处理加锁等待队列中缓存的各个加锁请求。锁代理线程遍历该加锁等待队列，如果对本次第一计算设备发送的该数据请求对应的数据资源加锁完毕，那么说明已经对该加锁请求处理完毕，将该加锁反馈信息确定为加锁成功；否则，需要在加锁等待队列中继续等待，直到等待超时或者陷入死锁，确定该加锁反馈信息为加锁失败，也即是说，在该加锁请求在该加锁等待队列中的等待时长超过第三等待阈值，或者死锁检测不通过的情况下，将该加锁反馈信息确定为加锁失败。其中，该第三等待阈值为任一大于 0 的数值，例如第三等待阈值为 5 秒，该第三等待阈值也可添加到配置文件中，并在集群数据库启动时，由每个计算设备的数据库引擎载入。

在上述过程中，可以看出，加锁失败至少存在如下两种情况：a) 等待超时，即该加锁请求在该加锁等待队列中的缓存时长超过第三等待阈值，那么可认为加锁失败；b) 死锁检测不通过，说明在死锁检测中发现死锁，且第二计算设备自身被选中为死锁牺牲者，那么无法完成锁申请。

在一些实施例中，第二计算设备的死锁检测流程如下：响应于该第一计算设备的死锁检测消息，第二计算设备对该加锁请求进行本地死锁检测；在本地死锁检测不通过的情况下，确定死锁检测不通过；在本地死锁检测通过的情况下，确定该加锁请求的锁等待链，该锁等待链包括与该加锁请求具有依赖关系的锁信息；在该锁等待链包括远端锁的情况下，向发起该远端锁的第三计算设备发送死锁检测消息，该远端锁为已在该第二计算设备中注册、但并非由该第二计算设备发起的锁信息；在发送该死锁检测消息的任一计算设备又接收到该死锁检测消息的情况下，确定死锁检测不通过。

可选地，死锁检测消息可以由第一计算设备发起，比如第一计算设备过了比较长的时间还没有接收到第二计算设备的加锁反馈信息，但此时也并未等待超时，那么第一计算设备的消息处理线程可以向第二计算设备的消息处理线程发送死锁检测消息。第二计算设备接收到该死锁检测消息，先进行本地死锁检测，如果本地死锁检测不通过，那么无需遍历锁等待链，直接向第

一计算设备返回加锁反馈信息，该加锁反馈信息指示加锁失败，且失败类型为自身被选中为死锁牺牲者。如果本地死锁检测通过，但此时仍不能完全确保不存在死锁，因此还需要遍历对应的锁等待链，对锁等待链上的每个锁，判断是否为远端锁，如果是远端锁，那么向远端锁的锁持有端（即第三计算设备）发送死锁检测消息，触发第三计算设备执行与第二计算设备类似的死锁检测流程，即第三计算设备先进行本地死锁检测，再遍历锁等待链上的远端锁并再次发送死锁检测消息。如果任一计算设备，本身作为死锁检测消息的发送端，又接收到了其他计算设备发送的同一死锁检测消息，说明陷入死锁，那么这一计算设备向第一计算设备返回加锁反馈信息，该加锁反馈信息指示加锁失败，且失败类型为自身被选中为死锁牺牲者。比如，第一计算设备向第二计算设备发送死锁检测消息，而第二计算设备在锁等待链上存在远端锁，远端锁的锁持有端为第一计算设备，那么第二计算设备向第一计算设备发送死锁检测消息，此时出现了依赖环现象，即锁 A 等待锁 B，锁 B 又等待锁 A，即第一计算设备作为死锁检测消息的发送端，又接收到第二计算设备发送的同一死锁检测消息，说明发现死锁，本次加锁失败。

在一些实施例中，如果第二计算设备加锁成功，那么在对数据请求对应的数据资源加锁的情况下，第二计算设备的锁代理线程在消息处理线程中注册与该数据请求对应的锁信息；每间隔目标时长，确定已注册的该锁信息是否为活跃锁信息；在该锁信息为活跃锁信息、并且该数据请求对应的数据资源的加锁时长超过探活阈值的情况下，向该第一计算设备发送探活请求，该探活请求用于请求该第一计算设备指示是否释放该活跃锁信息所加锁的数据资源。其中，该探活阈值可以是任一大于 0 的数值，例如探活阈值为 20 秒。

在上述过程中，如果加锁成功，锁代理线程在消息处理线程注册对应锁信息，是为了方便消息处理线程管理本地已申请成功的各个锁，并且消息处理线程可以周期性地检查注册表中的活跃锁信息，如果超过探活阈值的情况下，活跃锁信息仍然没有被释放，例如，此时存在如下几种可能性：i) 第一计算设备的数据请求执行完毕，但发送的释放锁请求产生丢包；ii) 第一计算设备的锁申请线程崩溃；iii) 第一计算设备正常处理，但本次的数据请求如 DDL 事务本身耗时就很长。这时，如果属于上述情况 i) 和 ii)，通过发送探活请求，能够触发第一计算设备再次发送释放锁消息，指示第二计算设备释放掉对应的锁，如果属于上述情况 iii)，那么第二计算设备不能释放掉对应的锁，仍需要继续保持对应的锁。

407、第二计算设备向该第一计算设备发送该加锁反馈信息。

在一些实施例中，第二计算设备的消息处理线程向第一计算设备的消息处理线程发送该加锁反馈信息。

上述步骤 405-407，以单个第二计算设备为例介绍了申请远端锁的流程，对集群内每个第二计算设备均可执行与步骤 405-407 类似的操作，这里不做

赘述。

408、第一计算设备接收该至少一个第二计算设备基于该加锁请求返回的该加锁反馈信息。

- 5 可选地，第一计算设备在接收加锁反馈信息时，可以通过该第一计算设备上的消息处理线程，接收该至少一个第二计算设备上的消息处理线程返回的该加锁反馈信息。

- 10 在一些实施例中，由于第一计算设备会向集群内每个第二计算设备均发送该加锁请求，而每个第二计算设备的加锁反馈信息返回至第一计算设备的时间可能并不相同，在一些示例性场景中，如果第一计算设备上该数据请求的处理进程（如 DDL 进程）崩溃，在该 DDL 进程崩溃后，第一计算设备对于其接收到的加锁反馈信息，应当视情况来及时指示加锁成功的第二计算设备释放掉对应的锁，避免阻塞在该数据资源上的其他 DML 进程，以优化集群数据库的处理性能。

- 15 针对上述情况，第一计算设备可以响应于接收到任一第二计算设备的加锁反馈信息，在该数据请求处于活跃状态的情况下，通过该消息处理线程将该加锁反馈信息发送至该数据请求的锁申请线程；在该数据请求不处于活跃状态，且该加锁反馈信息为加锁成功的情况下，向该第二计算设备发送释放锁请求，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数据资源。

- 20 可选地，第一计算设备在通过消息处理线程接收到任一加锁反馈信息之后，判断本地的该数据请求是否活跃，以 DDL 请求为例，可判断 DDL 请求对应的 DDL 进程是否存在，由于有可能存在多个 DDL 请求复用同一 DDL 进程的情况，导致在当前 DDL 请求已失败（比如不满足目标条件，DDL 事务被回滚）的情况下，DDL 进程仍有可能存在，因此针对 DDL 进程判断 DDL 请求是否活跃是存在一定误判情况的。可选地，还可直接判断 DDL 请求对应的 DDL 事务是否处于活跃状态，这样能够有效降低上述误判情况，提高判断准确度。

- 30 如果该数据请求不活跃，即该数据请求不处于活跃状态，判断该加锁反馈信息是否表征加锁成功，如果该加锁反馈信息表征加锁成功，说明本地的 DDL 请求已经结束，但在远端的第二计算设备上却已申请成功了对应数据资源的锁，此时需要指示该第二计算设备及时释放掉对应数据资源的锁，因此，第一计算设备可通过消息处理线程向该第二计算设备的消息处理线程发送释放锁请求，以指示该第二计算设备释放该数据请求对应的数据资源，例如，将该数据资源对应的锁的内存状态参数修改为空闲状态；此外，如果该加锁反馈信息为加锁失败，代表远端不存在需要及时释放的数据资源，那么无需进行任何处理。

如果该数据请求活跃，即该数据请求处于活跃状态，那么可通过该消息

处理线程将该加锁反馈信息发送至该数据请求的锁申请线程，以使得锁申请线程记录本次接收到的加锁反馈信息，并基于下述步骤 409 来判断是否符合目标条件，从而方便了该锁申请线程及时统计接收到的加锁反馈信息是否符合当前锁同步模式对应的目标条件。

- 5 在上述步骤 404-408 中，示出了第一计算设备如何获取该集群数据库中的至少一个第二计算设备对该数据资源的加锁反馈信息，在一些实施例中，第一计算设备还可以直接在集群数据库中广播或组播的方式发送该加锁请求，并接收各个第二计算设备返回的加锁反馈信息，也可以不设置专门的消息处理线程，而是由第一计算设备上的锁申请线程和第二计算设备上的锁代理线程两者进行直接通信，可以降低线程间的通信开销。

409、第一计算设备在所获取的加锁反馈信息满足与该锁同步模式对应的目标条件的情况下，对该数据请求对应的数据资源加锁，执行该数据请求。

上述步骤 409 与上述步骤 304 类似，这里不做赘述。

- 15 410、在所获取的加锁反馈信息不满足该目标条件或者该数据请求执行完毕的情况下，第一计算设备向该至少一个第二计算设备发送释放锁请求。

其中，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数据资源。

- 20 可选地，第一计算设备的消息处理线程向各个第二计算设备的消息处理线程发送释放锁请求。释放锁请求的发送过程与加锁请求的发送过程同理，这里不做赘述。

411、对任一第二计算设备，响应于与该数据请求关联的释放锁请求，释放该数据请求对应的数据资源。

- 25 可选地，第二计算设备的消息处理线程接收该释放锁请求，并通知锁代理线程释放对应的数据资源，例如，锁代理线程将锁的内存状态参数修改为空闲状态，并在注册表中清除对应的锁信息，至此消息处理完毕，丢弃接收到的各个消息即可节约存储空间。

上述所有可选技术方案，能够采用任意结合形成本公开的可选实施例，在此不再一一赘述。

- 30 本申请实施例提供的方法，通过对锁同步模式设置灵活的多个级别，在弱同步模式下，直接对数据请求对应的数据资源加锁，并执行该数据请求，能够保证数据请求优先推进，无需等待第二计算设备的响应，而在不是弱同步模式下，则获取各个第二计算设备的加锁反馈信息，在所获取的加锁反馈信息符合目标条件的情况下，才对该数据资源加锁并执行该数据请求，使得在不同业务场景下选取不同的锁同步模式，从而保证集群数据库能够满足不同业务场景的处理需求，提升了集群数据库系统的处理能力。
- 35

图 5 是本申请实施例提供的一种加锁流程的原理性示意图，如 500 所示，

集群数据库中包括计算节点 1 和计算节点 2，计算节点 1 为第一计算设备，计算节点 1 上运行有锁申请线程和消息处理线程，计算节点 2 为第二计算设备，计算节点 2 上运行有消息处理线程和锁代理线程。以数据请求为 DDL 请求为例说明，在计算节点 1 的锁申请线程在本地申请锁成功之后，进入到全局加锁流程中，下面对全局加锁流程进行详述：

5 步骤 1、先完成本地锁申请，再进入加锁流程。

步骤 2、计算节点 1 的锁申请线程判断是否为弱同步模式，如果为弱同步模式，则无需向其他节点发送加锁申请，直接认为锁已经获取（正确性由写屏障机制保障），跳转至步骤 28；如果不为弱同步模式，进入步骤 3。

10 步骤 3、在不是弱同步模式的情况下，向其他节点发送加锁请求，进入步骤 4，并进入步骤 26 开始计时。

步骤 4、计算节点 2 的消息处理线程接收加锁请求。

其中，该消息处理线程在后台专门监听其他节点的加锁请求、释放锁请求以及加锁反馈信息，负责收发各类锁相关消息。在申请锁的阶段，消息处理线程作为消息接收端，调起锁代理线程来维护计算节点 2 上远端锁的生命周期。在返回加锁反馈信息阶段，可能出现加锁反馈信息到达申请端（计算节点 1）的时候，申请端的锁申请线程已经崩溃或退出，那么计算节点 1 上的消息处理线程将会根据锁申请线程是否退出、以及加锁反馈信息是否为加锁成功，确定是否需要向计算节点 2 发送释放锁请求，以释放远端已经申请成功的远端锁。

20 步骤 5、计算节点 2 启动本地锁代理线程，调用本地申请锁逻辑。

步骤 6、判断 DDL 请求对应的锁是否申请成功，如果申请成功进入步骤 8，如果申请不成功，包括如下情况：6.1) 加锁请求仍然位于加锁等待队列中在排队等待，此时继续等待，并进入步骤 7 判断是否超时；6.2) 死锁检测中发现死锁，自身被选中为死锁牺牲者，进入步骤 8。

步骤 7、判断远端锁申请是否超时，如果未超时，继续等待；如果超时，进入步骤 8。

例如，在判断是否超时时，判断等待时长是否超过第三等待阈值。

步骤 8、向计算节点 1 的消息处理线程发送加锁反馈信息，进入步骤 10。

30 可选地，加锁反馈信息可分为三类：加锁成功、锁等待超时、死锁牺牲者。

可选地，加锁反馈信息可分为两类：加锁成功、加锁失败，其中，加锁失败包括：锁等待超时、死锁牺牲者。

35 步骤 9、计算节点 2 的消息处理线程的消息反馈接口，这里主要处理步骤 10-13 的探活机制所引发的探活请求，将探活请求发送至计算节点 1 的消息处理线程。

其中，探活请求针对的是分布式环境下，锁申请端（计算节点 1）的锁

申请线程已经退出，但是计算节点 2 的远端锁一直没有得到释放的情况。在超过一定时长（如探活阈值）远端锁仍然没有被释放的情况下，发送探活请求到锁申请端即锁持有者，进行探活。

5 步骤 10、计算节点 2 的锁代理线程在消息处理线程中注册所有远端锁的锁信息与锁状态。

锁信息包括但不限于：锁类型（读锁、写锁、更新锁等），锁针对的数据资源（如数据库名，数据表名，数据索引 ID，数据列名等），加锁时间等。

锁状态包括但不限于：活跃状态、非活跃状态等。

10 步骤 11、计算节点 2 的消息处理线程周期性检查注册表中各个远端锁中的活跃锁。

例如，遍历注册表中的每个锁信息，查询处于活跃状态的远端锁。

步骤 12、判断是否存在超过探活阈值的远端活跃锁，如果存在，进入步骤 13，如果不存在，则继续周期性监听注册表中的锁信息。

步骤 13、准备反馈探活请求，进入步骤 9。

15 步骤 14、计算节点 1 的消息处理线程接收加锁反馈信息以及探活请求。

步骤 15、判断本地的锁申请线程是否活跃，如果不活跃，进入步骤 16；如果活跃，进入步骤 21。

其中，在活跃情况下，可划分为三种情况：15.1) 当前锁申请线程活跃，处于申请锁阶段，那么更新对应锁申请结构数据，发送至锁申请线程处理；20 15.2) 当前锁申请线程在半强同步模式下，已经达到锁申请成功的目标条件，更新对应锁申请结构数据，完成操作；15.3) 当前锁申请线程处于锁退出，或者锁申请失败退出（例如半强同步模式下已判定不符合目标条件）情况，更新对应锁申请结构数据，完成操作。

25 可选地，该锁申请结构数据用于记录各个计算节点 2 的加锁反馈信息，例如采取哈希表、数组、集合、向量等形式，比如采取集合形式，集合中的每个元素代表一个计算节点是加锁成功还是加锁失败，元素取 0 代表加锁失败，元素取 1 代表加锁成功，元素为空 (null) 代表还未接收到加锁反馈信息。

30 步骤 16、如果此时加锁反馈信息为加锁成功，或者接收到的是探活请求，进入步骤 17；否则，不做任何处理，进入步骤 32 即消息处理完毕，丢弃接收到的本条消息。

步骤 17、计算节点 1 的消息处理线程向计算节点 2 的消息处理线程发送释放锁请求。

步骤 18、计算节点 2 的消息处理线程接收释放锁请求。

步骤 19、计算节点 2 的消息处理线程通知锁代理线程释放锁。

35 步骤 20、计算节点 2 的锁代理线程释放锁。

步骤 21、计算节点 1 的锁申请线程接收消息处理线程发过来的加锁反馈信息。这里包括步骤 8 中涉及的几类加锁反馈信息。

步骤 22、判断当前的锁同步模式，如果是强同步模式，进入步骤 25，如果是半强同步模式，进入步骤 23。

5 步骤 23、检验锁通过概率参数 lock_succ_node_num 是否被设置为有效值（即大于或等于 1、且小于或等于集群内所有计算设备的数量），如果是有效值，按照半强同步模式进入步骤 24；如果不是有效值，按照强同步模式进入步骤 25。

10 步骤 24、判断是否反馈加锁成功的远端节点加上本地节点的数量超过锁通过概率参数 lock_succ_node_num，如果超过，进入步骤 28；如果不超过，确定是否申请锁超时，即等待时长是否超过第二等待阈值，如果尚未等待超时则继续等待，如果等待超时则加锁失败；如果反馈加锁失败的远端节点数量已超过集群内节点总数量减去锁通过概率参数 lock_succ_node_num 所得的数值，则判定加锁失败，进入步骤 29。

15 步骤 25、强同步模式下，需要判断是否所有节点都反馈加锁成功，如果满足上述条件，进入步骤 28；如果不满足但尚未等待超时，即等待时长不超过第一等待阈值，则继续等待；如果任一节点反馈加锁失败，则判定加锁失败，进入步骤 29。

步骤 26、开始计算锁申请时间。

步骤 27、判断锁申请时间是否超时，如果超时，进入步骤 29 即加锁失败；如果不超时，继续返回步骤 21 检查逻辑。

20 可选地，强同步模式下超时阈值为第一等待阈值，半强同步模式下超时阈值为第二等待阈值，两者可以取值相同或不同。

步骤 28、全局加锁成功。

步骤 29、全局加锁失败。

步骤 30、向所有其他节点发送释放锁请求。

25 可选地，由于其他节点上存在步骤 11-20 的探活逻辑，因此计算节点 1 在发送释放锁请求完毕后，不需要等待其他节点返回释放锁反馈信息，即可结束流程。

步骤 31、完成加锁流程。

步骤 32、消息处理完毕，丢弃消息即可。

30 在本申请实施例中，详细介绍了加锁流程，能够在分布式数据库中实现一种灵活的锁机制，来确保各个进程或者线程对于某一种数据资源进行修改时的操作正确性。同时，通过这种灵活的锁机制的实现，能够在弱同步模式、半强同步模式、强同步模式下实现灵活切换，让分布式数据库集群能够根据自身数据库应用场景的需求，选择不同级别的锁机制，来满足用户在不同业务场景下的业务需求。

35

图 6 是本申请实施例提供的一种释放锁流程的原理性示意图，如 600 所

-25-

示, 集群数据库中包括计算节点 1 和计算节点 2, 计算节点 1 为第一计算设备, 计算节点 2 为第二计算设备, 计算节点 2 上运行有消息处理线程和锁代理线程。在计算节点 1 全局加锁成功之后, 执行对应的数据请求如 DDL 请求, 在本地对 DDL 事务处理逻辑完成后, 可进入到释放锁流程中, 下面对释放锁流程进行详述:

5 步骤 1、DDL 事务的本地处理逻辑完成后, 进入释放锁流程。

当持有锁的计算节点 1 完成了 DDL 事务的计算逻辑之后, 就需要全局释放锁了, 在本申请实施例中, 以先释放远端锁, 再释放本地锁的顺序为例, 来介绍全局释放锁的流程。在这种情况下, 加锁以及释放锁的整体过程为:

10 本地加锁→远端加锁→远端释放锁→本地释放锁。

步骤 2、判断是否为弱同步模式, 如果是弱同步模式, 进入步骤 10, 无需释放远端锁, 如果不是弱同步模式, 进入步骤 3。

步骤 3、向其他节点发送释放锁请求。

步骤 4、计算节点 2 的消息处理线程接收释放锁请求。

15 步骤 5、消息处理线程启动锁代理线程在本地释放对应的远端锁。

例如, 如果锁代理线程处于挂起状态, 则消息处理线程唤醒锁代理线程, 并进行释放锁操作。

步骤 6、锁代理线程在本地释放对应的远端锁, 如果释放成功, 进入步骤 7, 如果释放失败, 重新释放。

20 步骤 7、准备清理注册表中对应的锁信息。

步骤 8、清理注册表中已注册的锁信息。

步骤 9、锁代理线程退出。

步骤 10、释放锁成功。

步骤 11、释放锁流程结束。

25 在本申请实施例中, 详细介绍了在 DDL 请求执行完毕以后, 如何进行释放锁流程, 需要说明的是, 如果是探活机制探测出 DDL 线程已崩溃, 或者是不符合锁同步模式下的目标条件, 那么触发的释放锁流程与本申请实施例类似, 这里不做赘述。

30 图 7 是本申请实施例提供的一种死锁检测流程的原理性示意图, 如 700 所示, 集群数据库中包括计算节点 1 和计算节点 2, 计算节点 1 为第一计算设备, 计算节点 2 为第二计算设备, 计算节点 2 上运行有消息处理线程。下面对死锁检测流程进行说明:

步骤 1、计算节点 1 希望对数据资源 A 加锁。

35 步骤 2、向其他节点发送死锁检测消息, 并进入步骤 3 和步骤 10。

步骤 3、计算节点 2 的消息处理线程接收死锁检测消息。

例如, 死锁检测消息包括相关的锁信息。

-26-

步骤 4、应用本地死锁检测逻辑，进行本地死锁检测，如果死锁检测通过，进入步骤 5；如果死锁检测不通过，返回指示自身被选中为死锁牺牲者的消息。

步骤 5、遍历锁等待链。

- 5 步骤 6、判断锁等待链中当前的锁是否为远端锁，如果是远端锁，进入步骤 7；如果不是远端锁，遍历锁等待链上的下一个锁。

步骤 7、向锁持有者发送死锁检测消息。

在本申请实施例中，以锁持有者为计算节点 1 为例说明。

步骤 8、接收死锁检测消息。

- 10 步骤 9、发现死锁检测消息形成闭环，则有死锁产生。

步骤 10、向本地的消息处理线程注册锁信息。

- 在本申请实施例中，不但需要进行本地死锁检测逻辑，还针对分布式数据库进行了适配，在遍历锁等待链时，通过判断是否为远端锁，如果不是远端锁，那么在本地死锁检测逻辑中就能够发现是否产生死锁，如果是远端锁，那么通过向锁持有者（即锁持有线程所在计算节点）发送死锁检测消息，如果持有锁节点和发起死锁检测的计算节点是同一个节点，那么代表形成了闭环，即产生了死锁，例如本申请实施例中，计算节点 1 等待计算节点 2，计算节点 2 等待计算节点 1，形成闭环，在发生死锁的情况下，当前的锁申请操作会被拒绝，此时可由数据库引擎间隔一段时间后进行自动重试，或者拒绝消息返回至用户，让用户在等待一段时间后，主动发起重试。
- 15 20

- 上述各个实施例中详细介绍了分布式数据库中的加锁流程、释放锁流程和死锁检测流程，而在本申请实施例中，将详细分析这一锁同步机制在异常场景下的表现，以说明上述锁同步机制如何应对分布式场景的计算节点网络分区、网络不稳定、网络延迟或者网络丢包等异常场景。
- 25

- 如果管理员选择弱同步模式，由于弱同步模式不需要向远端申请锁，对于资源操作的正确性，由写屏障机制来保证，因此相当于对需要申请锁的 DDL 事务具有较高优先级，但随着 DDL 事务的推进，有可能会将用户的 DML 事务给回滚掉，而通常 DML 事务又是用户的关键业务，因此在某些场景下用户体验较差，但适用于 DDL 事务较多的应用场景。
- 30

- 如果管理员选择强同步模式，可以确保平等对待 DDL 事务与 DML 事务，但是在网络状况不佳的情况下，DDL 事务由于需要申请全局锁，要等待集群内所有其他节点返回加锁反馈信息，会出现因为某一个或几个节点因为分区或网络延迟，加锁反馈信息返回的不及时，导致在这段等待期间内，整个分布式集群其他节点因为锁授予成功而拒绝 DML 事务的执行，最终导致整个集群的响应速度慢，带给用户的体验是整个集群不能提供服务；因此强同步模式在网络状况不好的情况下，用户体验同样不佳。
- 35

-27-

而如果管理员选择半强同步模式，能够提供一个折中选择，即在上述分布式系统网络问题带来的影响和用户对于业务执行成功的高需求之间，提供一个灵活且可设置的集群运行方式。半强同步模式需要设置一个锁通过概率参数 `lock_succ_node_num` 来确定判定多少节点返回算做全局申请锁成功：如果

5 果 `lock_succ_node_num = 0`，其实等同于弱同步模式；如果 `lock_succ_node_num = n`，`n` 为集群中节点即计算设备的总数量，其实等于强同步模式；如果 $0 < \text{lock_succ_node_num} < n$ ，即为半强同步模式。

通过提供锁通过概率参数 `lock_succ_node_num`，能够让管理员根据自己集群的网络情况设置一个合理值，最大程度的保证用户 DML 事务能够和 DDL

10 事务之间公平对待，同时也不会因为某一个或者多个节点不响应导致整个集群不可用，如被 hang 住。

此外，在上述加锁流程中的步骤 10-20 还提供了一种探活机制，确保在一个锁的锁持有者 (owner) 因为某种原因未能将释放锁请求发送到其他节点的情况下，远端节点能够通过探活机制去进行主动探活，从而释放掉远端锁。

15 在本申请实施例中，通过提供一种灵活的加锁机制，来为管理员提供一种能够根据分布式数据库集群运行的网络环境以及对于 DML 事务成功率的要求，来确定锁同步模式的灵活调节方式，能够最大限度让用户使用分布式集群的计算能力，同时也可以让分布式集群各节点运行过程中对外表现的整个像

20 一个单一节点的数据库系统的表现一致。

图 8 是本申请实施例提供的一种请求处理装置的结构示意图，请参考图 8，该装置包括：

确定模块 801，用于响应于数据请求，确定集群数据库的锁同步模式；

25 加锁执行模块 802，用于在该锁同步模式为弱同步模式的情况下，对该数据请求对应的数据资源加锁，执行该数据请求；

第一获取模块 803，用于在该锁同步模式不为弱同步模式的情况下，获取该集群数据库中的至少一个第二计算设备对该数据资源的加锁反馈信息，该加锁反馈信息用于表征该第二计算设备是否对该数据资源加锁成功；

30 该加锁执行模块 802，还用于在所获取的该加锁反馈信息满足该锁同步模式对应的目标条件的情况下，对该数据请求对应的数据资源加锁，执行该数据请求。

本申请实施例提供的装置，通过对锁同步模式设置灵活的多个级别，在弱同步模式下，直接对数据请求对应的数据资源加锁，并执行该数据请求，能够保证数据请求优先推进，无需等待第二计算设备的响应，而在不是弱同步模式

35 下，则获取各个第二计算设备的加锁反馈信息，在所获取的加锁反馈信息满足目标条件的情况下，对该数据资源加锁并执行该数据请求，使得在不同业务场景下选取不同的锁同步模式，从而保证集群数据库能够满足不同

业务场景的处理需求，提升了集群数据库系统的处理能力。

在一种可能实施方式中，基于图 8 的装置组成，该第一获取模块 803 包括：

5 发送单元，用于向该至少一个第二计算设备发送与该数据请求关联的加锁请求；

接收单元，用于接收该至少一个第二计算设备基于该加锁请求返回的该加锁反馈信息。

10 在一种可能实施方式中，该发送单元用于：通过该第一计算设备上的消息处理线程，分别向该至少一个第二计算设备上的消息处理线程发送该加锁请求；

该接收单元用于：通过该第一计算设备上的消息处理线程，接收该至少一个第二计算设备上的消息处理线程返回的该加锁反馈信息。

在一种可能实施方式中，基于图 8 的装置组成，该装置还包括：

15 发送模块，用于响应于接收到任一第二计算设备的加锁反馈信息，在该数据请求处于活跃状态的情况下，通过该消息处理线程将该加锁反馈信息发送至该数据请求的锁申请线程；

该发送模块，还用于在该数据请求不处于活跃状态，且该加锁反馈信息表征加锁成功的情况下，向该第二计算设备发送释放锁请求，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数据资源。

20 在一种可能实施方式中，在该锁同步模式为强同步模式的情况下，该目标条件为该至少一个第二计算设备的加锁反馈信息均表征加锁成功；在该锁同步模式为半强同步模式的情况下，该目标条件为该至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。

25 在一种可能实施方式中，在该锁同步模式为强同步模式的情况下，该目标条件为等待时长不超过第一等待阈值，且该至少一个第二计算设备的加锁反馈信息均表征加锁成功；在该锁同步模式为半强同步模式的情况下，该目标条件为等待时长不超过第二等待阈值，且该至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。

在一种可能实施方式中，基于图 8 的装置组成，该装置还包括：

30 第二获取模块，用于在该锁同步模式为半强同步模式的情况下，获取该目标数量；

切换模块，用于在该目标数量小于一或者大于该集群数据库中全部第二计算设备的数量的情况下，将该锁同步模式切换为该强同步模式。

在一种可能实施方式中，基于图 8 的装置组成，该装置还包括：

35 发送模块，用于在所获取的该加锁反馈信息不满足该目标条件或者该数据请求执行完毕的情况下，向该至少一个第二计算设备发送释放锁请求，该释放锁请求用于指示该第二计算设备释放该数据请求对应的数据资源。

在一种可能实施方式中，该数据请求为数据定义语言 DDL 请求。

上述所有可选技术方案，能够采用任意结合形成本公开的可选实施例，在此不再一一赘述。

需要说明的是：上述实施例提供的请求处理装置在处理数据请求时，仅
5 以上述各功能模块的划分进行举例说明，实际应用中，能够根据需要而将上述功能分配由不同的功能模块完成，即将计算设备的内部结构划分成不同的功能模块，以完成以上描述的全部或者部分功能。另外，上述实施例提供的请求处理装置与请求处理方法实施例属于同一构思，其具体实现过程详见请求处理方法实施例，这里不再赘述。

10

图 9 是本申请实施例提供的一种请求处理装置的结构示意图，请参考图 9，该装置包括：

添加模块 901，用于响应于与数据请求关联的加锁请求，将该加锁请求添加到加锁等待队列中；该加锁请求由该集群数据库中的第一计算设备在锁
15 同步模式不为弱同步模式的情况下发送；

处理模块 902，用于处理该加锁等待队列中的该加锁请求，得到加锁反馈信息；

发送模块 903，用于向该第一计算设备发送该加锁反馈信息。

本申请实施例提供的装置，通过对锁同步模式设置灵活的多个级别，只
20 有在不是弱同步模式的情况下，第一计算设备才会发送加锁请求，以使得响应于该加锁请求，返回对应的加锁反馈信息，触发第一计算设备在所获取的加锁反馈信息满足目标条件的情况下，对该数据资源加锁并执行该数据请求，使得集群数据库支持在不同业务场景下选取不同的锁同步模式，从而保证集群数据库能够满足不同业务场景的处理需求，提升了集群数据库系统的处理
25 能力。

在一种可能实施方式中，该处理模块 902 用于：

对该数据请求对应的数据资源加锁，将该加锁反馈信息确定为表征加锁成功；

在该加锁请求在该加锁等待队列中的等待时长超过第三等待阈值，或者
30 死锁检测不通过的情况下，将该加锁反馈信息确定为表征加锁失败。

在一种可能实施方式中，基于图 9 的装置组成，该装置还包括：

检测模块，用于响应于该第一计算设备的死锁检测消息，对该加锁请求进行本地死锁检测；

第一确定模块，用于在本地死锁检测不通过的情况下，确定死锁检测不
35 通过；

第二确定模块，用于在本地死锁检测通过的情况下，确定该加锁请求的锁等待链，该锁等待链包括与该加锁请求具有依赖关系的锁信息；

-30-

该发送模块 903, 还用于在该锁等待链包括远端锁的情况下, 向发起该远端锁的第三计算设备发送死锁检测消息, 该远端锁为已在该第二计算设备中注册、但并非由该第二计算设备发起的锁信息;

5 该第一确定模块, 还用于在发送该死锁检测消息的任一计算设备又接收到该死锁检测消息的情况下, 确定死锁检测不通过。

在一种可能实施方式中, 该添加模块 901 用于:

通过该第二计算设备上的消息处理线程接收该加锁请求;

通过该消息处理线程, 启动与该加锁请求对应的锁代理线程;

通过该锁代理线程, 将该加锁请求添加到该加锁等待队列中。

10 在一种可能实施方式中, 基于图 9 的装置组成, 该装置还包括:

注册模块, 用于在对该数据请求所对应的数据资源加锁的情况下, 注册与该数据请求对应的锁信息;

第三确定模块, 用于每间隔目标时长, 确定所述锁信息是否为活跃锁信息;

15 该发送模块 903, 还用于在该锁信息为活跃锁信息, 并且该数据请求对应的数据资源的加锁时长探活阈值的情况下, 向该第一计算设备发送探活请求, 该探活请求用于请求该第一计算设备指示是否释放该数据资源。

在一种可能实施方式中, 基于图 9 的装置组成, 该装置还包括:

20 释放模块, 用于响应于与该数据请求关联的释放锁请求, 释放该数据请求所对应的数据资源。

上述所有可选技术方案, 能够采用任意结合形成本公开的可选实施例, 在此不再一一赘述。

25 需要说明的是: 上述实施例提供的请求处理装置在处理数据请求时, 仅以上述各功能模块的划分进行举例说明, 实际应用中, 能够根据需要而将上述功能分配由不同的功能模块完成, 即将计算设备的内部结构划分成不同的功能模块, 以完成以上描述的全部或者部分功能。另外, 上述实施例提供的请求处理装置与请求处理方法实施例属于同一构思, 其具体实现过程详见请求处理方法实施例, 这里不再赘述。

30 图 10 是本申请实施例提供的一种计算设备的结构示意图, 以计算设备为终端 1000 为例说明。可选地, 该终端 1000 的设备类型包括: 智能手机、平板电脑、MP3 播放器 (Moving Picture Experts Group Audio Layer III, 动态影像专家压缩标准音频层面 3)、MP4 (Moving Picture Experts Group Audio Layer IV, 动态影像专家压缩标准音频层面 4) 播放器、笔记本电脑或台式电脑。
35 终端 1000 还可能被称为用户设备、便携式终端、膝上型终端、台式终端等其他名称。

通常, 终端 1000 包括有: 处理器 1001 和存储器 1002。

可选地，处理器 1001 包括一个或多个处理核心，比如 4 核心处理器、8 核心处理器等。可选地，处理器 1001 采用 DSP (Digital Signal Processing, 数字信号处理)、FPGA (Field-Programmable Gate Array, 现场可编程门阵列)、PLA (Programmable Logic Array, 可编程逻辑阵列) 中的至少一种硬件形式来实现。在一些实施例中，处理器 1001 包括主处理器和协处理器，主处理器是用于对在唤醒状态下的数据进行处理的处理单元，也称 CPU (Central Processing Unit, 中央处理器)；协处理器是用于对在待机状态下的数据进行处理的低功耗处理器。在一些实施例中，处理器 1001 集成有 GPU (Graphics Processing Unit, 图像处理器)，GPU 用于负责显示屏所需要显示的内容的渲染和绘制。一些实施例中，处理器 1001 还包括 AI (Artificial Intelligence, 人工智能) 处理器，该 AI 处理器用于处理有关机器学习的计算操作。

在一些实施例中，存储器 1002 包括一个或多个计算机可读存储介质，可选地，该计算机可读存储介质是非暂态的。可选地，存储器 1002 还包括高速随机存取存储器，以及非易失性存储器，比如一个或多个磁盘存储设备、闪存存储设备。在一些实施例中，存储器 1002 中的非暂态的计算机可读存储介质用于存储至少一个程序代码，该至少一个程序代码用于被处理器 1001 所执行以实现本申请中各个实施例提供的请求处理方法。

在一些实施例中，终端 1000 还可选包括有：外围设备接口 1003 和至少一个外围设备。处理器 1001、存储器 1002 和外围设备接口 1003 之间能够通过总线或信号线相连。各个外围设备能够通过总线、信号线或电路板与外围设备接口 1003 相连。具体地，外围设备包括：射频电路 1004、显示屏 1005、摄像头组件 1006、音频电路 1007、定位组件 1008 和电源 1009 中的至少一种。

在一些实施例中，终端 1000 还包括有一个或多个传感器 1010。该一个或多个传感器 1010 包括但不限于：加速度传感器 1011、陀螺仪传感器 1012、压力传感器 1013、指纹传感器 1014、光学传感器 1015 以及接近传感器 1016。

本领域技术人员能够理解，图 10 中示出的结构并不构成对终端 1000 的限定，能够包括比图示更多或更少的组件，或者组合某些组件，或者采用不同的组件布置。

图 11 是本申请实施例提供的一种计算设备的结构示意图，该计算设备 1100 可因配置或性能不同而产生比较大的差异，该计算设备 1100 包括一个或一个以上处理器 (Central Processing Units, CPU) 1101 和一个或一个以上的存储器 1102，其中，该存储器 1102 中存储有至少一条计算机程序，该至少一条计算机程序由该一个或一个以上处理器 1101 加载并执行以实现上述各个实施例提供的请求处理方法。可选地，该计算设备 1100 还具有有线或无线网络接口、键盘以及输入输出接口等部件，以便进行输入输出，该计算设备 1100 还包括其他用于实现设备功能的部件，在此不做赘述。

在示例性实施例中，还提供了一种计算机可读存储介质，例如包括至少

一条计算机程序的存储器，上述至少一条计算机程序可由终端中的处理器执行以完成上述各个实施例中的请求处理方法。例如，该计算机可读存储介质包括 ROM(Read-Only Memory, 只读存储器)、RAM(Random-Access Memory, 随机存取存储器)、CD-ROM (Compact Disc Read-Only Memory, 只读光盘)、

5 磁带、软盘和光数据存储设备等。

在示例性实施例中，还提供了一种计算机程序产品或计算机程序，包括一条或多条程序代码，该一条或多条程序代码存储在计算机可读存储介质中。计算设备的一个或多个处理器能够从计算机可读存储介质中读取该一条或多条程序代码，该一个或多个处理器执行该一条或多条程序代码，使得计算设

10 备能够执行以完成上述实施例中的请求处理方法。

本领域普通技术人员能够理解实现上述实施例的全部或部分步骤能够通过硬件来完成，也能够通过程序来指令相关的硬件完成，可选地，该程序存储于一种计算机可读存储介质中，可选地，上述提到的存储介质是只读存储器、磁盘或光盘等。

15 以上所述仅为本申请的可选实施例，并不用以限制本申请，凡在本申请的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。

权利要求

1.一种请求处理方法，由集群数据库中的第一计算设备执行，所述方法包括：

响应于数据请求，确定所述集群数据库的锁同步模式；

5 在所述锁同步模式为弱同步模式的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求；

在所述锁同步模式不为所述弱同步模式的情况下，获取所述集群数据库中的至少一个第二计算设备对所述数据资源的加锁反馈信息，所述加锁反馈信息用于表征所述第二计算设备是否对所述数据资源加锁成功；在所获取的
10 所述加锁反馈信息满足所述锁同步模式对应的目标条件的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求。

2.根据权利要求1所述的方法，所述获取所述集群数据库中的至少一个第二计算设备对所述数据资源的加锁反馈信息，包括：

向所述至少一个第二计算设备发送与所述数据请求关联的加锁请求；

15 接收所述至少一个第二计算设备基于所述加锁请求返回的所述加锁反馈信息。

3.根据权利要求2所述的方法，所述向所述至少一个第二计算设备发送与所述数据请求关联的加锁请求，包括：

20 通过所述第一计算设备上的消息处理线程，分别向所述至少一个第二计算设备上的消息处理线程发送所述加锁请求；

所述接收所述至少一个第二计算设备基于所述加锁请求返回的所述加锁反馈信息，包括：

通过所述第一计算设备上的消息处理线程，接收所述至少一个第二计算设备上的消息处理线程返回的所述加锁反馈信息。

25 4.根据权利要求3所述的方法，所述方法还包括：

响应于接收到任一第二计算设备的加锁反馈信息，在所述数据请求处于活跃状态的情况下，通过所述消息处理线程将所述加锁反馈信息发送至所述数据请求的锁申请线程；

30 在所述数据请求不处于活跃状态，且所述加锁反馈信息表征加锁成功的情况下，向所述第二计算设备发送释放锁请求，所述释放锁请求用于指示所述第二计算设备释放所述数据请求对应的数据资源。

5.根据权利要求1所述的方法，在所述锁同步模式为强同步模式的情况下，所述目标条件为所述至少一个第二计算设备的加锁反馈信息均表征加锁成功；在所述锁同步模式为半强同步模式的情况下，所述目标条件为所述至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。

6.根据权利要求1所述的方法，在所述锁同步模式为强同步模式的情况

下, 所述目标条件为等待时长不超过第一等待阈值, 且所述至少一个第二计算设备的加锁反馈信息均表征加锁成功; 在所述锁同步模式为半强同步模式的情况下, 所述目标条件为等待时长不超过第二等待阈值, 且所述至少一个第二计算设备中不少于目标数量的第二计算设备的加锁反馈信息表征加锁成功。

5

7. 根据权利要求 5 或 6 所述的方法, 所述方法还包括:

在所述锁同步模式为所述半强同步模式的情况下, 获取所述目标数量;

在所述目标数量小于一或者大于所述集群数据库中全部第二计算设备的数量的情况下, 将所述锁同步模式切换为所述强同步模式。

10

8. 根据权利要求 1 所述的方法, 所述方法还包括:

在所获取的所述加锁反馈信息不满足所述目标条件或者所述数据请求执行完毕的情况下, 向所述至少一个第二计算设备发送释放锁请求, 所述释放锁请求用于指示所述第二计算设备释放所述数据请求对应的数据资源。

9. 根据权利要求 1 所述的方法, 所述数据请求为数据定义语言 DDL 请求。

15

10. 一种请求处理方法, 由集群数据库中的第二计算设备执行, 所述方法包括:

响应于与数据请求关联的加锁请求, 将所述加锁请求添加到加锁等待队列中; 所述加锁请求由所述集群数据库中的第一计算设备在锁同步模式不为弱同步模式的情况下发送;

20

处理所述加锁等待队列中的所述加锁请求, 得到加锁反馈信息;

向所述第一计算设备发送所述加锁反馈信息。

11. 根据权利要求 10 所述的方法, 所述处理所述加锁等待队列中的所述加锁请求, 得到加锁反馈信息, 包括:

25

对所述数据请求对应的数据资源加锁, 将所述加锁反馈信息确定为表征加锁成功;

在所述加锁请求在所述加锁等待队列中的等待时长超过第三等待阈值, 或者死锁检测不通过的情况下, 将所述加锁反馈信息确定为表征加锁失败。

12. 根据权利要求 10 所述的方法, 所述方法还包括:

30

响应于所述第一计算设备的死锁检测消息, 对所述加锁请求进行本地死锁检测;

在本地死锁检测不通过的情况下, 确定死锁检测不通过;

在本地死锁检测通过的情况下, 确定所述加锁请求的锁等待链; 所述锁等待链包括与所述加锁请求具有依赖关系的锁信息;

35

在所述锁等待链包括远端锁的情况下, 向发起所述远端锁的第三计算设备发送死锁检测消息; 所述远端锁为已在所述第二计算设备中注册、但并非由所述第二计算设备发起的锁信息;

在发送所述死锁检测消息的任一计算设备又接收到所述死锁检测消息的

情况下，确定死锁检测不通过。

13.根据权利要求 10 所述的方法，所述响应于与数据请求关联的加锁请求，将所述加锁请求添加到加锁等待队列中，包括：

- 5 通过所述第二计算设备上的消息处理线程，接收所述加锁请求；
通过所述消息处理线程，启动与所述加锁请求对应的锁代理线程；
通过所述锁代理线程，将所述加锁请求添加到所述加锁等待队列中。

14.根据权利要求 10 所述的方法，所述方法还包括：

在对所述数据请求对应的数据资源加锁的情况下，注册与所述数据请求对应的锁信息；

- 10 每间隔目标时长，确定所述锁信息是否为活跃锁信息；

在所述锁信息为活跃锁信息，并且所述数据请求对应的数据资源的加锁时长超过探活阈值的情况下，向所述第一计算设备发送探活请求；所述探活请求用于请求所述第一计算设备指示是否释放所述数据资源。

15.一种请求处理装置，所述装置包括：

- 15 确定模块，用于响应于数据请求，确定集群数据库的锁同步模式；

加锁执行模块，用于在所述锁同步模式为弱同步模式的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求；

- 20 第一获取模块，用于在所述锁同步模式不为所述弱同步模式的情况下，获取所述集群数据库中的至少一个第二计算设备对所述数据资源的加锁反馈信息，所述加锁反馈信息用于表征所述第二计算设备是否对所述数据资源加锁成功；

所述加锁执行模块，还用于在所获取的所述加锁反馈信息满足与所述锁同步模式对应的目标条件的情况下，对所述数据请求对应的数据资源加锁，执行所述数据请求。

- 25 16.一种请求处理装置，所述装置包括：

添加模块，用于响应于与数据请求关联的加锁请求，将所述加锁请求添加到加锁等待队列中；所述加锁请求由所述集群数据库中的第一计算设备在锁同步模式不为弱同步模式的情况下发送；

- 30 处理模块，用于处理所述加锁等待队列中的所述加锁请求，得到加锁反馈信息；

发送模块，用于向所述第一计算设备发送所述加锁反馈信息。

- 35 17.一种计算设备，其特征在于，所述计算设备包括一个或多个处理器和一个或多个存储器，所述一个或多个存储器中存储有至少一条计算机程序，所述至少一条计算机程序由所述一个或多个处理器加载并执行以实现如权利要求 1 至权利要求 9 或权利要求 10 至权利要求 14 中任一项所述的请求处理方法。

18.一种存储介质，其特征在于，所述存储介质中存储有至少一条计算机

—36—

程序，所述至少一条计算机程序由处理器加载并执行以实现如权利要求 1 至权利要求 9 或权利要求 10 至权利要求 14 中任一项所述的请求处理方法。

- 5 19.一种计算机程序产品，其特征在于，所述计算机程序产品包括至少一条计算机程序，所述至少一条计算机程序由计算设备的一个或多个处理器加载并执行以实现如权利要求 1 至权利要求 9 或权利要求 10 至权利要求 14 中任一项所述的请求处理方法。