(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118796932 A
(43)申请公布日 2024.10.18

(21)申请号 202311338585.3

(22)申请日 2023.10.16

(71)申请人 中移动金融科技有限公司
地址 100032 北京市西城区阜成门外大街31号4层411B
申请人 中国移动通信集团有限公司

(72)发明人 孙宇 朱沛东 刘春游 金文萱 张清华 王飞 牛亚宾 严明 李杰

(74)专利代理机构 深圳市世纪恒程知识产权代理事务所 44287
专利代理师 张楠

(51)Int.Cl.
G06F 16/27(2019.01)

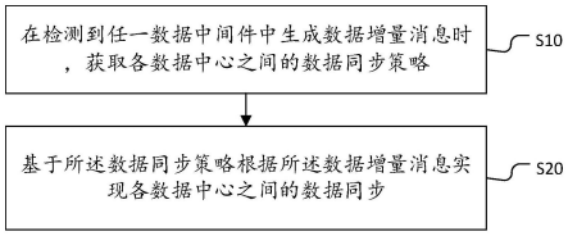
权利要求书2页 说明书12页 附图7页

(54)发明名称

数据同步方法、装置、设备及存储介质

(57)摘要

本发明属于数据处理技术领域，公开了一种数据同步方法、装置、设备及存储介质。本发明通过在检测到任一数据中间件中生成数据增量消息时，获取各数据中心之间的数据同步策略；基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。由于是在检测到数据中间件中生成数据增量消息时，根据对各数据中心之间的架构识别结果进行设置数据同步策略，再基于数据同步策略及数据增量消息实现各数据中心之间的数据同步，保证可支持多种不同的架构，使得仅需做少量处理即可切换架构，降低了架构切换的难度。



CN 118796932 A

1.一种数据同步方法,其特征在于,所述数据同步方法包括以下步骤:

在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略,所述各数据中心中分别设置有数据中间件,所述数据中间件在数据中心中发生数据变更时,根据变更的数据生成数据增量消息,所述数据同步策略根据对各数据中心之间的架构识别结果进行设置;

基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

2.如权利要求1所述的数据同步方法,其特征在于,所述数据同步策略包括单向同步策略;

所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

根据所述数据增量消息生成事务消息;

将所述事务消息添加至消息队列中,以使所述消息队列的消费者根据所述事务消息对所述消费者对应的数据中心进行数据同步。

3.如权利要求1所述的数据同步方法,其特征在于,所述数据同步策略包括双向同步策略;

所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

获取当前时刻信息,并生成随机数;

根据所述当前时刻信息及所述随机数构建高精度时间戳;

以所述高精度时间戳为排序依据将所述数据增量消息添加至有序集合中,以使所述有序集合对应的消费者根据所述有序集合的集合顺序对所述有序集合中的数据增量消息依次进行广播分发,对各数据中心进行数据同步。

4.如权利要求1所述的数据同步方法,其特征在于,所述数据同步策略包括星形同步策略;

所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

获取分布式锁,并检测共享缓存中是否存在所述分布式锁;

若不存在,则将所述分布式锁写入所述共享缓存中;

在写入成功后,将所述数据增量消息进行广播分发,以对各数据中心进行数据同步;

在同步完成后,将所述分布式锁从所述共享缓存中移除。

5.如权利要求4所述的数据同步方法,其特征在于,所述获取分布式锁的步骤,包括:

获取所述数据增量消息对应的数据增量类型;

根据所述数据增量类型确定锁构建字段及锁构建方式;

获取各锁构建字段对应的字段数据;

基于所述锁构建方式将所述字段数据进行拼接,获得分布式锁。

6.如权利要求1-5任一项所述的数据同步方法,其特征在于,所述获取各数据中心之间的数据同步策略的步骤,包括:

获取连接的各数据中心对应的数据中心代码,所述数据中心代码为标识数据中心唯一性的标识代码;

将各数据中心对应的数据中心代码进行聚合,获得中心代码集合;

对所述中心代码集合进行数据去重,获得去重代码集合;

检测所述去重代码集合中的代码数量;

若所述代码数量大于预设数量,则判定各数据中心之间的数据同步策略为星形同步策略。

7.如权利要求6所述的数据同步方法,其特征在于,所述检测所述去重代码集合中的代码数量的步骤,包括:

若所述代码数量小于或等于预设数量,则对各数据中心中的主从配置文件进行解析,获得各数据中心对应的主从配置信息;

根据所述主从配置信息确定各数据中心之间的主从指向关系;

若所述主从指向关系为单向指向关系,则判定各数据中心之间的数据同步策略为单向同步策略;

若所述主从指向关系为双向指向关系,则判定各数据中心之间的数据同步策略为双向同步策略。

8.一种数据同步装置,其特征在于,所述数据同步装置包括以下模块:

检测模块,用于在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略,所述各数据中心中分别设置有数据中间件,所述数据中间件在数据中心中发生数据变更时,根据变更的数据生成数据增量消息,所述数据同步策略根据对各数据中心之间的架构识别结果进行设置;

同步模块,用于基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

9.一种数据同步设备,其特征在于,所述数据同步设备包括:处理器、存储器及存储在所述存储器上并可在所述处理器上运行的数据同步程序,所述数据同步程序被处理器执行时实现如权利要求1-7中任一项所述的数据同步方法的步骤。

10.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质上存储有数据同步程序,所述数据同步程序执行时实现如权利要求1-7中任一项所述的数据同步方法的步骤。

数据同步方法、装置、设备及存储介质

技术领域

[0001] 本发明涉及数据处理技术领域,尤其涉及一种数据同步方法、装置、设备及存储介质。

背景技术

[0002] 多中心系统架构是指在多个不同地理位置的数据中心之间建立的一套完整、相对独立运行的系统,并且系统间数据低延时同步,以确保系统的高可用性和数据一致性,如今主流的方案一般是核心数据中心、单元化分库分表、分布式数据库等几种,必要时,还会采用人工干预解决数据冲突。

[0003] 但是,核心数据中心采用单向同步数据方式虽然能解决同步行冲突问题,通常作为主从架构(灾备、冷备)方案,它无法满足多主系统间同步数据的需求,且因数据强一致性要求,所有修改数据的操作都要在主库上完成,这会增大主库压力,若主力节点宕机,那么从节点就面临着故障的问题;单元化数据分片虽然能解决同步行冲突问题且满足多数据中心需求,但是增加了聚合操作等复杂查询的响应时间,进行数据迁移和重建索引等操作的复杂度也会升高;分布式数据库各方面性能很高,但是改变现有服务体系,投入到分布式数据库的大流中需要相当长的时间,成本也非常高昂,整体实现难度过高,且条件过于苛刻,对大部分非重构需求是不切实际的;而人工干预需要项目上常备运维人员,人力成本极为高昂。

[0004] 由此可见,各主流方案在应用时各有优劣,在企业面对不同场景时,需要对数据中心之间的连接方案进行调整,而常见的数据系统建设都是以某一种架构、同步方案为主,当项目面临性能瓶颈、技术重构、多中心建设等各种需求时常需要申请大量资源,做出很大的系统变更,比如双中心系统建设,项目中常以主备(异地灾备、同城灾备)等方案,如果因业务发展等因素需要调整为双活系统,则原系统需要给出大量的资源、人力去协调,比如数据库网络的贯通性、容器或主机资源的测算申请、同步带宽、专线、数据双向同步的建设、数据冲突带来的额外建设等等,升级工作非常繁琐,架构的调整十分困难。

发明内容

[0005] 本发明的主要目的在于提供一种数据同步方法、装置、设备及存储介质,旨在解决现有技术在进行数据中心架构切换时,流程繁琐,导致架构切换困难的技术问题。

[0006] 为实现上述目的,本发明提供了一种数据同步方法,所述方法包括以下步骤:

[0007] 在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略,所述各数据中心中分别设置有数据中间件,所述数据中间件在数据中心中发生数据变更时,根据变更的数据生成数据增量消息,所述数据同步策略根据对各数据中心之间的架构识别结果进行设置;

[0008] 基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

[0009] 可选的,所述数据同步策略包括单向同步策略;

[0010] 所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

[0011] 根据所述数据增量消息生成事务消息;

[0012] 将所述事务消息添加至消息队列中,以使所述消息队列的消费者根据所述事务消息对所述消费者对应的数据中心进行数据同步。

[0013] 可选的,所述数据同步策略包括双向同步策略;

[0014] 所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

[0015] 获取当前时刻信息,并生成随机数;

[0016] 根据所述当前时刻信息及所述随机数构建高精度时间戳;

[0017] 以所述高精度时间戳为排序依据将所述数据增量消息添加至有序集合中,以使所述有序集合对应的消费者根据所述有序集合的集合顺序对所述有序集合中的数据增量消息依次进行广播分发,对各数据中心进行数据同步。

[0018] 可选的,所述数据同步策略包括星形同步策略;

[0019] 所述基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步的步骤,包括:

[0020] 获取分布式锁,并检测共享缓存中是否存在所述分布式锁;

[0021] 若不存在,则将所述分布式锁写入所述共享缓存中;

[0022] 在写入成功后,将所述数据增量消息进行广播分发,以对各数据中心进行数据同步;

[0023] 在同步完成后,将所述分布式锁从所述共享缓存中移除。

[0024] 可选的,所述获取分布式锁的步骤,包括:

[0025] 获取所述数据增量消息对应的数据增量类型;

[0026] 根据所述数据增量类型确定锁构建字段及锁构建方式;

[0027] 获取各锁构建字段对应的字段数据;

[0028] 基于所述锁构建方式将所述字段数据进行拼接,获得分布式锁。

[0029] 可选的,所述获取各数据中心之间的数据同步策略的步骤,包括:

[0030] 获取连接的各数据中心对应的数据中心代码,所述数据中心代码为标识数据中心唯一性的标识代码;

[0031] 将各数据中心对应的数据中心代码进行聚合,获得中心代码集合;

[0032] 对所述中心代码集合进行数据去重,获得去重代码集合;

[0033] 检测所述去重代码集合中的代码数量;

[0034] 若所述代码数量大于预设数量,则判定各数据中心之间的数据同步策略为星形同步策略。

[0035] 可选的,所述检测所述去重代码集合中的代码数量的步骤,包括:

[0036] 若所述代码数量小于或等于预设数量,则对各数据中心中的主从配置文件进行解析,获得各数据中心对应的主从配置信息;

[0037] 根据所述主从配置信息确定各数据中心之间的主从指向关系;

[0038] 若所述主从指向关系为单向指向关系,则判定各数据中心之间的数据同步策略为

单向同步策略；

[0039] 若所述主从指向关系为双向指向关系，则判定各数据中心之间的数据同步策略为双向同步策略。

[0040] 此外，为实现上述目的，本发明还提出一种数据同步装置，所述数据同步装置包括以下模块：

[0041] 检测模块，用于在检测到任一数据中间件中生成数据增量消息时，获取各数据中心之间的数据同步策略，所述各数据中心中分别设置有数据中间件，所述数据中间件在数据中心中发生数据变更时，根据变更的数据生成数据增量消息，所述数据同步策略根据对各数据中心之间的架构识别结果进行设置；

[0042] 同步模块，用于基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

[0043] 此外，为实现上述目的，本发明还提出一种数据同步设备，所述数据同步设备包括：处理器、存储器及存储在所述存储器上并可在所述处理器上运行的数据同步程序，所述数据同步程序被处理器执行时实现如上所述的数据同步方法的步骤。

[0044] 此外，为实现上述目的，本发明还提出一种计算机可读存储介质，所述计算机可读存储介质上存储有数据同步程序，所述数据同步程序执行时实现如上所述的数据同步方法的步骤。

[0045] 本发明通过在检测到任一数据中间件中生成数据增量消息时，获取各数据中心之间的数据同步策略；基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。由于是在检测到数据中间件中生成数据增量消息时，根据对各数据中心之间的架构识别结果进行设置数据同步策略，再基于数据同步策略及数据增量消息实现各数据中心之间的数据同步，保证可支持多种不同的架构，使得仅需做少量处理即可切换架构，降低了架构切换的难度。

附图说明

[0046] 图1是本发明实施例方案涉及的硬件运行环境的电子设备的结构示意图；

[0047] 图2为本发明数据同步方法第一实施例的流程示意图；

[0048] 图3为本发明一实施例的设备结构示意图；

[0049] 图4为本发明一实施例的日志解析对象属性示意图；

[0050] 图5为本发明一实施例的单向同步流程示意图；

[0051] 图6为本发明一实施例的双向同步流程示意图；

[0052] 图7为本发明一实施例的星形同步流程示意图；

[0053] 图8为本发明数据同步方法第二实施例的流程示意图；

[0054] 图9为本发明数据同步装置第一实施例的结构框图。

[0055] 本发明目的的实现、功能特点及优点将结合实施例，参照附图做进一步说明。

具体实施方式

[0056] 应当理解，此处所描述的具体实施例仅用以解释本发明，并不用于限定本发明。

[0057] 参照图1，图1为本发明实施例方案涉及的硬件运行环境的数据同步设备结构示意

图。

[0058] 如图1所示,该电子设备可以包括:处理器1001,例如中央处理器(Central Processing Unit,CPU),通信总线1002、用户接口1003,网络接口1004,存储器1005。其中,通信总线1002用于实现这些组件之间的连接通信。用户接口1003可以包括显示屏(Display)、输入单元比如键盘(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(如无线保真(Wireless-Fidelity,WI-FI)接口)。存储器1005可以是高速的随机存取存储器(Random Access Memory,RAM),也可以是稳定的非易失性存储器(Non-Volatile Memory,NVM),例如磁盘存储器。存储器1005可选的还可以是独立于前述处理器1001的存储装置。

[0059] 本领域技术人员可以理解,图1中示出的结构并不构成对电子设备的限定,可以包括比图示更多或更少的部件,或者组合某些部件,或者不同的部件布置。

[0060] 如图1所示,作为一种存储介质的存储器1005中可以包括操作系统、网络通信模块、用户接口模块以及数据同步程序。

[0061] 在图1所示的电子设备中,网络接口1004主要用于与网络服务器进行数据通信;用户接口1003主要用于与用户进行数据交互;本发明电子设备中的处理器1001、存储器1005可以设置在数据同步设备中,所述电子设备通过处理器1001调用存储器1005中存储的数据同步程序,并执行本发明实施例提供的数据同步方法。

[0062] 本发明实施例提供了一种数据同步方法,参照图2,图2为本发明一种数据同步方法第一实施例的流程示意图。

[0063] 本实施例中,所述数据同步方法包括以下步骤:

[0064] 步骤S10:在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略。

[0065] 需要说明的是,本实施例的执行主体可以是所述数据同步设备,或由多个数据同步设备组成的集群,所述数据同步设备可以是个人电脑、服务器等电子设备,还可以是其他可实现相同或相似功能的电子设备,本实施例对不加以限制,在本实施例及下述各实施例中,以数据同步设备为例对本发明数据同步方法进行说明。

[0066] 需要说明的是,数据同步设备可以同时与多个数据中心进行连接,各数据中心中可以设置有数据中间件,在数据中心中发生数据变更时,数据中间件会根据变更的数据生成数据增量消息。数据同步设备可以通过对连接的各数据中心之间的连接架构进行自动识别,并根据架构识别结果设置对应的数据同步策略。

[0067] 例如:假设数据中心中设置的数据库集群为mysql数据集群,则此时在数据中心中设置的数据中间件可以是Canal中间件,Canal中间件会伪装自己为MySQL slave,将数据中心中的数据库集群作为MySQL master,模拟MySQL slave的交互协议,向数据库集群发送同步(dump)请求,MySQL master收到dump请求,会推送数据变更日志(binary log,也可以简称为binlog,可以为二进制日志)给Canal中间件,Canal中间件可以对binary log进行解析,将其解析为binary log对象,之后可以根据binary log对象生成数据增量消息。

[0068] 其中,根据binary log对象生成数据增量消息可以是将binary log对象格式化,从而生成数据增量消息,例如:将binary log对象格式化为json格式,将生成的json字符串作为数据增量消息。

[0069] 为了便于理解,现结合图3和4进行说明,但不对本方案进行限定。图3为本实施例的设备结构示意图,图4为本实施例的日志解析对象属性示意图。

[0070] 如图3所示,数据同步设备或由多个数据同步设备组成的集群中,可以包括rabbitmq集群、process模块、redis集群以及多个Agent,通过Agent与各数据中心的canal中间件以及数据库集群连接;

[0071] 其中,数据库集群:包含主流的关系型数据库Mysql、Oracle等,原则上使用binlog日志同步的数据库都支持,因使用了canal中间件收集数据库binlog日志,故默认数据库为Mysql;

[0072] Canal中间件:基于MySQL数据库增量日志解析,提供增量数据订阅和消费;

[0073] Agent组件可以由java语言实现,是数据同步的关键组件。在Agent组件内部维护了一个存储sql的队列与process的节点信息。与process同步模块之间定时探活获取节点状态,对接canal中间件,在canal解析binlog后会生成对应的数据输出流,Agent组件可以在捕获数据输出流后确定操作对应的table表、执行sql等,缓存数据到内部队列后发送请求到redis集群,根据table表名做哈希计算,确定数据落点,存入zset数据结构。Agent组件会发送数据处理请求到Process模块,Process加工完数据后需要回调系统内所有Agent完成数据同步;

[0074] Process同步模块:Process模块由java语言实现,是同步处理中心,拉取数据、分发数据、释放资源、日志记录都是由该模块处理。内部采用cas乐观锁控制同步逻辑,并且内置了ecache进程内缓存框架存储数据,Process模块每5秒钟探活一次Agent组件,查看系统状态,在收到Agent的数据处理请求后,可以按顺序同步给系统内所有agent组件。

[0075] Redis Cluster集群:redis集群可以采用六主六从结构,数据被分成16384个哈希槽(hash slot),每个哈希槽都有一个编号,从0到16383,这些哈希槽被分配到不同的Redis节点上,Agent向redis存储数据时,可以根据键值(表名)计算哈希值(可以采用CRC16算法计算)判断落在哪个槽位上。

[0076] 如图4所示,解析之后的binary log对象可以包括多个字段,其中,database字段存储表示发生变化的数据库标识,table字段用于存储发生数据变化的数据表表名,type字段用于表示数据变化的类型(可以包括INSERT、UPDATE、DELETE三种类型),es字段用于存储数据发生变化的时间,ts用于存储数据库集群同步该数据变化消耗的时长,sql可以用于存储令数据发生变化的sql语句,data字段用于存储发生变化后的数据,old用于存储发生变化前的数据。

[0077] 步骤S20:基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

[0078] 需要说明的是,数据同步策略可以包括单向同步策略、双向同步策略及星形同步策略等,当然,根据实际需要,还可以设置更多的同步策略,本实施例对此不加以限制。

[0079] 在实际使用中,不同的数据同步策略,可以对应不同的数据同步方式,基于数据同步策略根据数据增量消息实现各数据中心之间的数据同步可以是根据数据同步策略对应的数据同步方式对数据增量消息进行处理,从而实现对各数据中心之间的数据同步。

[0080] 在具体实现中,若数据同步策略为单向同步策略,则表示此时数据中心是以主从方式进行数据同步(即一个数据中心为主节点,其他数据中心为从节点,数据由主节点同步

至从节点),此时本实施例所述步骤S20,可以包括:

[0081] 根据所述数据增量消息生成事务消息;

[0082] 将所述事务消息添加至消息队列中,以使所述消息队列的消费者根据所述事务消息对所述消费者对应的数据中心进行数据同步。

[0083] 需要说明的是,消息队列可以是设置在rabbitmq集群中的消息队列,根据数据增量消息生成事务消息可以是将数据增量消息转化为可以存入消息队列中的数据结构,从而生成事务消息。消费者对应的数据中心可以是与消费者对接的数据中心。

[0084] 在实际使用中,在采用单向同步策略时,此时数据中心以主从方式进行数据同步,此时生成数据增量消息数据中心为主节点,其对接的为消息队列的生产者,其他需要同步的数据中心为从节点,其对接的为消息队列的消费者,在将事务消息存入消息队列之后,消息队列的消费者会从消息队列中提取数据增量消息,并根据数据增量消息对消费者对应的数据中心进行数据同步(如控制消费者对应的数据中心执行数据增量消息中的SQL语句)。

[0085] 为了便于理解,以上述图3为基础,结合图5进行说明,图5为本实施例的单向同步流程示意图,如图5所示,各数据中心与Agent对接,此时Agent作为MQ客户端启动,不同的是主节点(作为主节点的数据中心)对接的Agent接入的是消息生产者,从节点(作为从节点的数据中心)对接的Agent接入的是消息消费者。

[0086] 主节点对接的Agent在获取到canal中间件传过来的数据增量消息,处理后将生成的事务消息发送到MQ服务端(即图3中的rabbitmq集群),存储至消息队列中。其中,对于业务量较多、写入请求较高的数据表在MQ服务端中可以预设队列,令一个表对应一个队列,利用队列的有序性做数据的顺序同步。

[0087] 消费者(从节点对接的Agent)可以监听消息队列,从消息队列中拉取事务消息,根据事务消息做本地同步(即根据事务消息对对接的从节点进行数据同步)。其中,为了保证消息的准确投递以及精准消费,消息队列可以关闭消息的自动ACK并开启事务消息,由消费者Agent在本地数据同步后发起提交,才将事务消息从消息队列中移除,为了保证数据同步的可靠性,消费者Agent还可以保存每个表的同步进度信息,通过异步请求发给Process模块。

[0088] 在具体实现中,若数据同步策略为双向同步策略,则表示此时同步方式为两数据中心互相进行同步,此时本实施例所述步骤S20,可以包括:

[0089] 获取当前时刻信息,并生成随机数;

[0090] 根据所述当前时刻信息及所述随机数构建高精度时间戳;

[0091] 以所述高精度时间戳为排序依据将所述数据增量消息添加至有序集合中,以使所述有序集合对应的消费者根据所述有序集合的集合顺序对所述有序集合中的数据增量消息依次进行广播分发,对各数据中心进行数据同步。

[0092] 需要说明的是,当前时刻信息可以是当前时刻对应的时间戳,有序集合可以是redis中的zset集合。

[0093] 在实际使用中,若数据同步策略为双向同步策略,则表示此时是两个数据中心之间进行双向同步,例如:假设数据中心包括A和B,若数据同步策略为双向同步策略,则此时数据可以从A同步至B,也可以从B同步至A。

[0094] 而数据同步是不断在进行过程中的,为了避免数据紊乱,需要保证数据的时序正

确性(即数据正常更新的前后顺序),则此时在可以获取当前时刻信息,而由于在同一时刻,可能会有多个数据发生变化,为了保证针对此种数据也进行区分,可以在获取到当前时刻信息的同时,生成随机数,之后根据当前时刻信息及随机数构建高精度时间戳,其中,根据当前时刻信息及随机数构建高精度时间戳可以是将随机数拼接在当前时刻信息之后,例如:假设当前时刻信息可以为“A”,随机数为“336”,则此时高精度时间戳可以为“A336”。

[0095] 在具体实现中,以高精度时间戳为排序依据将数据增量消息添加至有序集合中可以是将高精度时间戳作为分值(score),将数据增量消息添加至有序集合中,则之后有序集合对应的消费者(如上图3中Process模块)可以根据有序集合的集合顺序依次从有序集合中提取数据增量消息,并依次进行广播分发,对各数据中心进行数据同步,由此则可以实现各数据中心之间的数据同步,且保证数据中心进行数据双向同步过程中的时序正确性。

[0096] 为了便于理解,现以上述图3为基础,结合图6进行说明,图6为本实施例的双向同步流程示意图,如图6所示,Agent可以使用time命令从Redis集群中获取时间戳(即当前时刻信息),结合本地随机数生成高精度时间戳,然后将数据增量消息(也可以仅为数据增量消息中的sql语句)发送到Redis中的zset(有序集合)进行存储,存储时,以时间戳来做分值(score),zset根据分值对存储的数据进行排序,操作数据所属数据表可以按照哈希算法分配到Redis的各个节点中,之后Process开始从有序集合中拉取数据并通过广播分发到系统内各个Agent节点进行数据同步。其中,Process在拉取数据时可以使用ZREVRANGEBYSCORE命令从有序集合中获取最新的时间戳对应的数据。

[0097] 在具体实现中,若数据同步策略为星形同步策略,则表示此时是三个或三个以上的数据中心之间进行数据同步,此时本实施例所述步骤S20,可以包括:

[0098] 获取分布式锁,并检测共享缓存中是否存在所述分布式锁;

[0099] 若不存在,则将所述分布式锁写入所述共享缓存中;

[0100] 在写入成功后,将所述数据增量消息进行广播分发,以对各数据中心进行数据同步;

[0101] 在同步完成后,将所述分布式锁从所述共享缓存中移除。

[0102] 需要说明的是,共享缓存可以是上述的redis集群。若数据同步策略为星形同步策略,则表示此时是三个或三个以上的数据中心之间进行数据同步,例如:假设数据中心包括A、B和C,此时数据可能是从A同步至B和C,也可能是从B同步至A和C,还有可能是从C同步至A和B,此时可以通过分布式锁来避免各数据中心之间的数据同步冲突问题,因此,可以先获取分布式锁,然后检测共享缓存中是否存在分布式锁。

[0103] 可以理解的是,若共享缓存中不存在该分布式锁,则表示此时该行数据并未被进行同步,因此,可以执行数据同步流程,此时为了避免同步的冲突,可以将分布式锁写入共享缓存中,则此后其他同步执行流程在检测到共享缓存中的分布式锁时,会暂停对该行数据的同步,因此,在分布式锁成功写入共享缓存之后,可以将数据增量消息进行广播分发,从而对各数据中心进行数据同步,而在同步完成之后,为了避免对后续同步的影响,可以将分布式锁从共享缓存中移除。

[0104] 在具体实现中,为了保证可以避免行冲突问题,分布式锁可以根据不同场景获取不同的字段进行组成,则此时本实施例所述获取分布式锁的步骤,可以包括:

[0105] 获取所述数据增量消息对应的数据增量类型;

[0106] 根据所述数据增量类型确定锁构建字段及锁构建方式;

[0107] 获取各锁构建字段对应的字段数据;

[0108] 基于所述锁构建方式将所述字段数据进行拼接,获得分布式锁。

[0109] 需要说明的是,数据增量类型可以包括数据修改、表结构修改、设置服务端命令等多种,其中,数据修改还可以进一步分为insert、delete和update三种,每种不同的数据增量类型,可以对应不同的锁构建字段以及锁构建方式,数据增量类型与锁构建字段以及锁构建方式之间的对应关系,可以由数据同步设备的管理人员根据实际需要预先进行设置。

[0110] 例如:针对insert、delete等数据增量类型,锁构建字段可以包括表名、行ID两种,锁构建方式可以设置为“表名:行ID”,此时假设表名为User,行ID为1000,则此时分布式锁为“User:1000”,其中,若insert时,数据库设置了自增操作,则需要先获取自增序列(secquence);

[0111] 针对update数据增量类型,锁构建字段可以包括表名、行ID、字段三种,锁构建方式可以设置为“表名:行ID:字段”,且字段为多个时,字段值之间通过下划线拼接,假设表名为User,行ID为1000,字段有三个,对应的值分别为A、B和C,则此时分布式锁为“User:1000:A_B_C”;

[0112] 针对表结构修改数据增量类型,可以以表名为分布式锁;

[0113] 针对设置服务端命令(如设置会话、全局属性等),可以以属性为分布式锁。

[0114] 为了便于理解,现以上述图3为基础,结合图7进行说明,图7为本实施例的星形同步流程示意图,如图7所示,Agent分析场景,生成分布式锁,并请求redis判断分布式锁是否已存在,若已存在,则判定获取分布式锁失败说明有进程在同步该行数据,此时可以开启数据强一致性模式,自旋等待重新获取锁;

[0115] 而若是若不存在,则表示没有进程在同步改行数据判定获取分布式锁成功,之后,可以将分布式锁写入redis集群,并通知Process模块开始处理数据同步,Process接到通知后,可以从ecache缓存中获取待同步数据(即数据增量消息)做广播分发,Agent接收到分发的数据增量消息后,根据数据增量消息进行数据同步,并提交本地事务,在提交本地事务之后返回ACK,在节点均返回ACK后认为本次同步完成,Process释放本地缓存、分布式锁。

[0116] 其中,若大部分节点返回ACK后,也可以判定本次同步完成,Process模块内部采用cas乐观锁限制同步数据逻辑。

[0117] 在此过程中利用了redis的多路复用技术和单线程模型,保障了数据操作的高性能以及线程安全。通过redis cluster集群实现一致性哈希算法,将每一张表通过哈希计算映射到哈希环上,可以防止数据堆积。

[0118] 本实施例通过在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略;基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。由于是在检测到数据中间件中生成数据增量消息时,根据对各数据中心之间的架构识别结果进行设置数据同步策略,再基于数据同步策略及数据增量消息实现各数据中心之间的数据同步,保证可支持多种不同的架构,使得仅需做少量处理即可切换架构,降低了架构切换的难度。

[0119] 参考图8,图8为本发明一种数据同步方法第二实施例的流程示意图。

[0120] 基于上述第一实施例,本实施例数据同步方法的所述步骤S10,包括:

[0121] 步骤S101:在检测到任一数据中间件中生成数据增量消息时,获取连接的各数据中心对应的数据中心代码。

[0122] 需要说明的是,数据中心代码可以为标识数据中心唯一性的标识代码。

[0123] 在实际使用中,获取连接的各数据中心对应的数据中心代码可以是获取与数据同步设备相连接的各数据中心的数据中心代码。

[0124] 例如:如上图3所示,数据中心与Agent相连,每一个Agent都有全局唯一ID以及与其相连的数据中心的数据中心代码,则此时可以从Agent中读取数据中心代码,从而获得连接的各数据中心对应的数据中心代码。

[0125] 步骤S102:将各数据中心对应的数据中心代码进行聚合,获得中心代码集合。

[0126] 在实际使用中,将各数据中心对应的数据中心代码进行聚合,获得中心代码集合可以是将各数据中心对应的数据中心代码添加至同一集合中,从而获得中心代码集合。

[0127] 步骤S103:对所述中心代码集合进行数据去重,获得去重代码集合。

[0128] 可以理解的是,同一数据中心在实际应用时,可能会连接多个Agent,为了准确的区分到底有多个各数据中心与数据同步设备连接,可以对中心代码集合进行数据去重,令集合中的数据中心代码不再重复,之后,将去重处理后的中心代码集合作为去重代码集合。

[0129] 步骤S104:检测所述去重代码集合中的代码数量。

[0130] 需要说明的是,检测去重代码集合中的代码数量可以是统计去重代码集合中的数据中心代码的数量,从而获得代码数量。

[0131] 步骤S105:若所述代码数量大于预设数量,则判定各数据中心之间的数据同步策略为星形同步策略。

[0132] 需要说明的是,预设数量可以由数据同步设备的管理人员预先进行设置。例如:数据中心一般在两个时,才会以主从架构或双活架构进行连接,在超过两个时,一般是采用多活架构连接,因此可以将预设数量设置为2。

[0133] 可以理解的是,若代码数量大于预设数量,则表示此时数据同步设备连接的数据中心是以多活架构进行连接,此时是三个或三个以上的数据中心之间进行数据同步,因此,可以判定各数据中心之间的数据同步策略为星形同步策略。

[0134] 而若是代码数量小于或等于预设数量,则此时还需要进一步区分各数据中心之间的连接架构是主从架构还是双活架构,则此时本实施例所述步骤S104之后,还可以包括:

[0135] 若所述代码数量小于或等于预设数量,则对各数据中心中的主从配置文件进行解析,获得各数据中心对应的主从配置信息;

[0136] 根据所述主从配置信息确定各数据中心之间的主从指向关系;

[0137] 若所述主从指向关系为单向指向关系,则判定各数据中心之间的数据同步策略为单向同步策略;

[0138] 若所述主从指向关系为双向指向关系,则判定各数据中心之间的数据同步策略为双向同步策略。

[0139] 需要说明的是,主从配置文件可以是设置在数据中心集群中的配置文件,如my.cnf配置文件。主从配置文件中至少包括server-id,log-bin,binlog-do-db,replicate-do-db,read_only等字段;

[0140] 其中,server-id用于指定服务器节点唯一值;log-bin用于指定是否启用二进制

记录;read_only用于指定是否只读控制,一般主节点是读写权限,从节点只读权限,1是只读,0是读写;replicate-do-db用于指定需要复制的数据库名。

[0141] 在必要时,还可以包括额外字段:master-host用于指定主机地址或主机名称;master-user用于指定主机用户名;master-password用于指定主机密码;master-port用于指定主机端口。

[0142] 在实际使用中,在获取到各数据中心的主从配置信息之后,可以根据主从配置信息确定各数据中心之间的主从指向关系,而若是主从指向关系为单向指向关系,则表示其中一个数据中心单向指定另一数据中心为主机,此时数据中心之间是通过主从架构连接,因此,可以判定各数据中心之间的数据同步策略为单向同步策略;

[0143] 而若是主从指向关系为双向指向关系,则表示此时两个数据中心是互相指定对方为主机,此时数据中心之间是通过双活架构连接,因此,可以判定各数据中心之间的数据同步策略为双向同步策略。

[0144] 例如:Agent在数据中心A的节点上执行SHOW MASTER STATUS命令,记下返回值,如File和Position。在数据中心B的节点上执行SHOW SLAVE STATUS命令,检查Master_Host、Master_User、Master_Log_File、Read_Master_Log_Pos等参数。如果这些参数对应的值与数据中心A的主节点相匹配,那么数据中心A到B的同步是单向的。

[0145] 反之,在数据中心B的主节点上执行SHOW MASTER STATUS命令,记下File和Position的值。在数据中心A的从节点上执行SHOW SLAVE STATUS,检查Master_Host、Master_User、Master_Log_File、Read_Master_Log_Pos等参数。如果这些参数对应的值与数据中心B的主节点相匹配,那么数据中心B到A的同步是单向的。

[0146] 而如果数据中心A到B和数据中心B到A都是指向性的,那么两个数据中心之间的主从指向关系为双向指向关系,此时表示两个数据中心之间采用的是双活架构。反之,如果只有一个方向的同步,那么它们之间采用的是主备架构。

[0147] 针对主备架构采用单向同步策略;双活架构采用双向同步策略;多活架构采用星形同步策略。

[0148] 为了避免每次均需要获取同步策略,Process在确定整体架构后会缓存策略信息到内存中,同时调用Agent通知接口,Agent会缓存策略信息到内存中,为后续同步数据做准备。

[0149] 本实施例通过获取连接的各数据中心对应的数据中心代码;将各数据中心对应的数据中心代码进行聚合,获得中心代码集合;对中心代码集合进行数据去重,获得去重代码集合;检测去重代码集合中的代码数量;若代码数量大于预设数量,则判定各数据中心之间的数据同步策略为星形同步策略。由于可根据数据中心代码确定具体与数据同步设备连接的数据中心的数量,根据数量识别数据中心是否以多活架构进行连接,并根据识别结果设置对应的数据同步策略,保证了可根据数据中心之间的连接架构选择合适的数据同步策略。

[0150] 此外,本发明实施例还提出一种存储介质,所述存储介质上存储有数据同步程序,所述数据同步程序被处理器执行时实现如上文所述的数据同步方法的步骤。

[0151] 参照图9,图9为本发明数据同步装置第一实施例的结构框图。

[0152] 如图9所示,本发明实施例提出的数据同步装置包括:

[0153] 检测模块10,用于在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略,所述各数据中心中分别设置有数据中间件,所述数据中间件在数据中心中发生数据变更时,根据变更的数据生成数据增量消息,所述数据同步策略根据对各数据中心之间的架构识别结果进行设置;

[0154] 同步模块20,用于基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。

[0155] 本实施例通过在检测到任一数据中间件中生成数据增量消息时,获取各数据中心之间的数据同步策略;基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步。由于是在检测到数据中间件中生成数据增量消息时,根据对各数据中心之间的架构识别结果进行设置数据同步策略,再基于数据同步策略及数据增量消息实现各数据中心之间的数据同步,保证可支持多种不同的架构,使得仅需做少量处理即可切换架构,降低了架构切换的难度。

[0156] 进一步的,所述数据同步策略包括单向同步策略;

[0157] 所述同步模块20,还用于根据所述数据增量消息生成事务消息;将所述事务消息添加至消息队列中,以使所述消息队列的消费者根据所述事务消息对所述消费者对应的数据中心进行数据同步。

[0158] 进一步的,所述数据同步策略包括双向同步策略;

[0159] 所述同步模块20,还用于获取当前时刻信息,并生成随机数;根据所述当前时刻信息及所述随机数构建高精度时间戳;以所述高精度时间戳为排序依据将所述数据增量消息添加至有序集合中,以使所述有序集合对应的消费者根据所述有序集合的集合顺序对所述有序集合中的数据增量消息依次进行广播分发,对各数据中心进行数据同步。

[0160] 进一步的,所述数据同步策略包括星形同步策略;

[0161] 所述同步模块20,还用于获取分布式锁,并检测共享缓存中是否存在所述分布式锁;若不存在,则将所述分布式锁写入所述共享缓存中;在写入成功后,将所述数据增量消息进行广播分发,以对各数据中心进行数据同步;在同步完成后,将所述分布式锁从所述共享缓存中移除。

[0162] 进一步的,所述同步模块20,还用于获取所述数据增量消息对应的数据增量类型;根据所述数据增量类型确定锁构建字段及锁构建方式;获取各锁构建字段对应的字段数据;基于所述锁构建方式将所述字段数据进行拼接,获得分布式锁。

[0163] 进一步的,所述检测模块10,还用于获取连接的各数据中心对应的数据中心代码,所述数据中心代码为标识数据中心唯一性的标识代码;将各数据中心对应的数据中心代码进行聚合,获得中心代码集合;对所述中心代码集合进行数据去重,获得去重代码集合;检测所述去重代码集合中的代码数量;若所述代码数量大于预设数量,则判定各数据中心之间的数据同步策略为星形同步策略。

[0164] 进一步的,所述检测模块10,还用于若所述代码数量小于或等于预设数量,则对各数据中心中的主从配置文件进行解析,获得各数据中心对应的主从配置信息;根据所述主从配置信息确定各数据中心之间的主从指向关系;若所述主从指向关系为单向指向关系,则判定各数据中心之间的数据同步策略为单向同步策略;若所述主从指向关系为双向指向关系,则判定各数据中心之间的数据同步策略为双向同步策略。

[0165] 应当理解的是,以上仅为举例说明,对本发明的技术方案并不构成任何限定,在具体应用中,本领域的技术人员可以根据需要进行设置,本发明对此不做限制。

[0166] 需要说明的是,以上所描述的工作流程仅仅是示意性的,并不对本发明的保护范围构成限定,在实际应用中,本领域的技术人员可以根据实际的需要选择其中的部分或者全部来实现本实施例方案的目的,此处不做限制。

[0167] 另外,未在本实施例中详尽描述的技术细节,可参见本发明任意实施例所提供的数据同步方法,此处不再赘述。

[0168] 此外,需要说明的是,在本文中,术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含,从而使得包括一系列要素的过程、方法、物品或者系统不仅包括那些要素,而且还包括没有明确列出的其他要素,或者是还包括为这种过程、方法、物品或者系统所固有的要素。在没有更多限制的情况下,由语句“包括一个……”限定的要素,并不排除在包括该要素的过程、方法、物品或者系统中还存在另外的相同要素。

[0169] 上述本发明实施例序号仅仅为了描述,不代表实施例的优劣。

[0170] 通过以上的实施方式的描述,本领域的技术人员可以清楚地了解到上述实施例方法可借助软件加必需的通用硬件平台的方式来实现,当然也可以通过硬件,但很多情况下前者是更佳的实施方式。基于这样的理解,本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质(如只读存储器(Read Only Memory,ROM)/RAM、磁碟、光盘)中,包括若干指令用以使得一台终端设备(可以是手机,计算机,服务器,或者网络设备等)执行本发明各个实施例所述的方法。

[0171] 以上仅为本发明的优选实施例,并非因此限制本发明的专利范围,凡是利用本发明说明书及附图内容所作的等效结构或等效流程变换,或直接或间接运用在其他相关的技术领域,均同理包括在本发明的专利保护范围内。

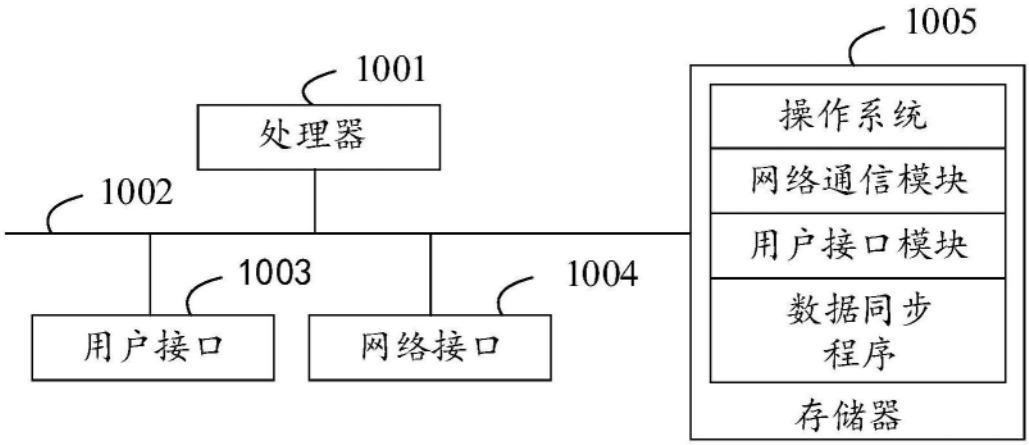

1005
1001
处理器
操作系统
网络通信模块
1002
用户接口模块
1003
1004
数据同步
程序
用户接口
网络接口
存储器

图1

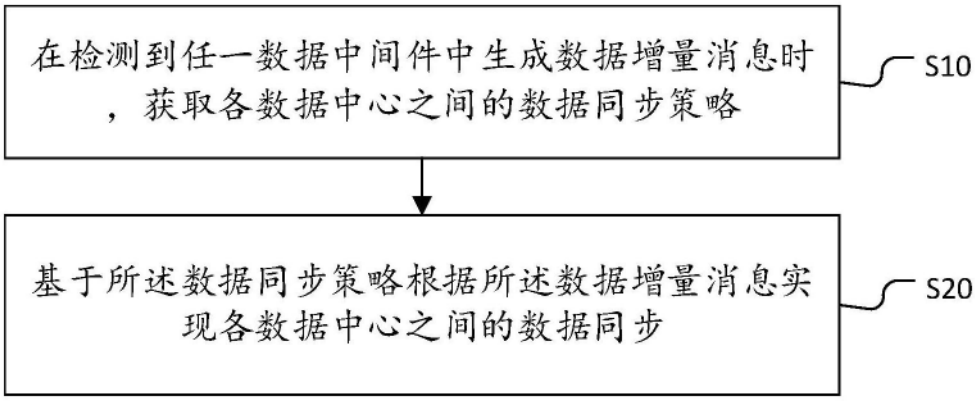

在检测到任一数据中间件中生成数据增量消息时，获取各数据中心之间的数据同步策略
S10
基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步
S20

图2

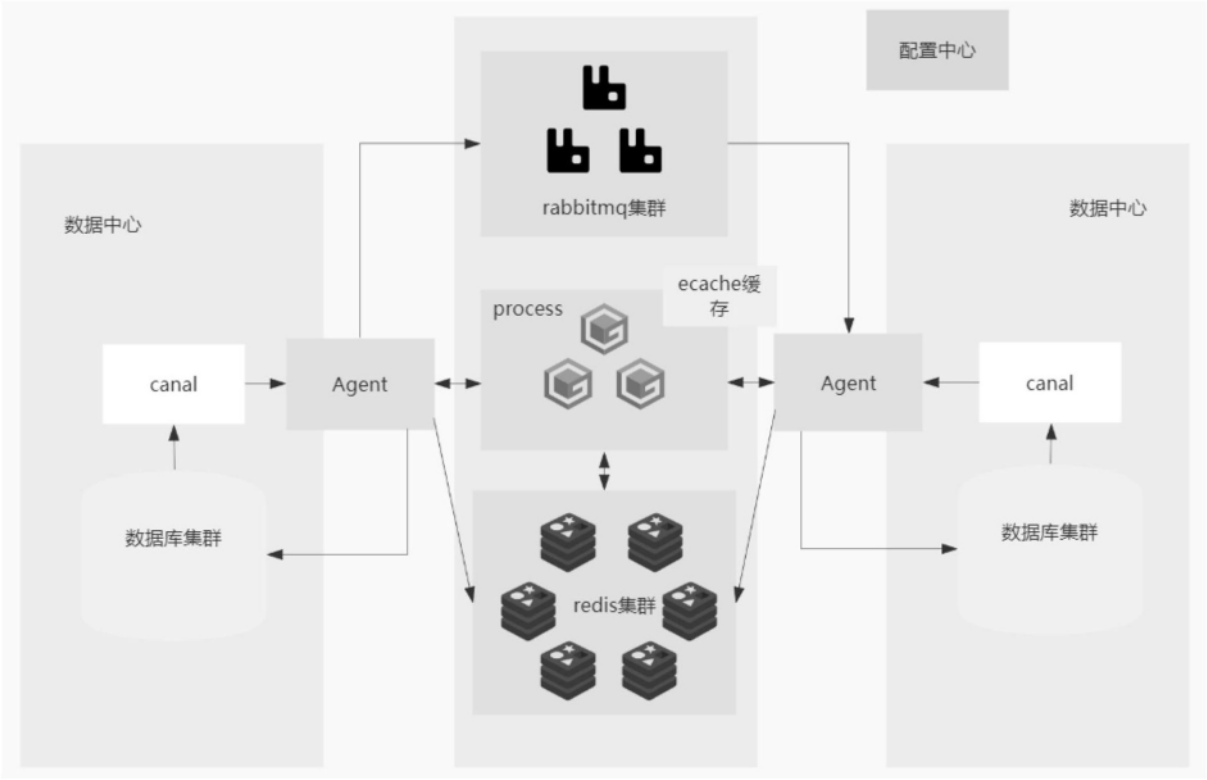

配置中心
rabbitmq集群
数据中心
数据中心
ecache缓存
process
canal
Agent
Agent
canal
数据库集群
redis集群
数据库集群

图3

```
public class CommonMessage implements Serializable {

    private static final long        serialVersionUID = 2611556444074013268L;
    private String                   database;                                  // 数据库或 schema
    private String                   table;                                     // 表名
    private List<String>             pkNames;
    private Boolean                  isDdl;
    // 类型:INSERT/UPDATE/DELETE
    private String                   type;
    // binlog executeTime, 执行耗时
    private Long                     es;
    // dml build timeStamp, 同步时间
    private Long                     ts;
    // 执行的 sql,dml sql 为空
    private String                   sql;
    // 数据列表
    private List<Map<String, Object>> data;
    // 旧数据列表,用于 update,size 和 data 的 size 一一对应
    private List<Map<String, Object>> old;
}
```

图4

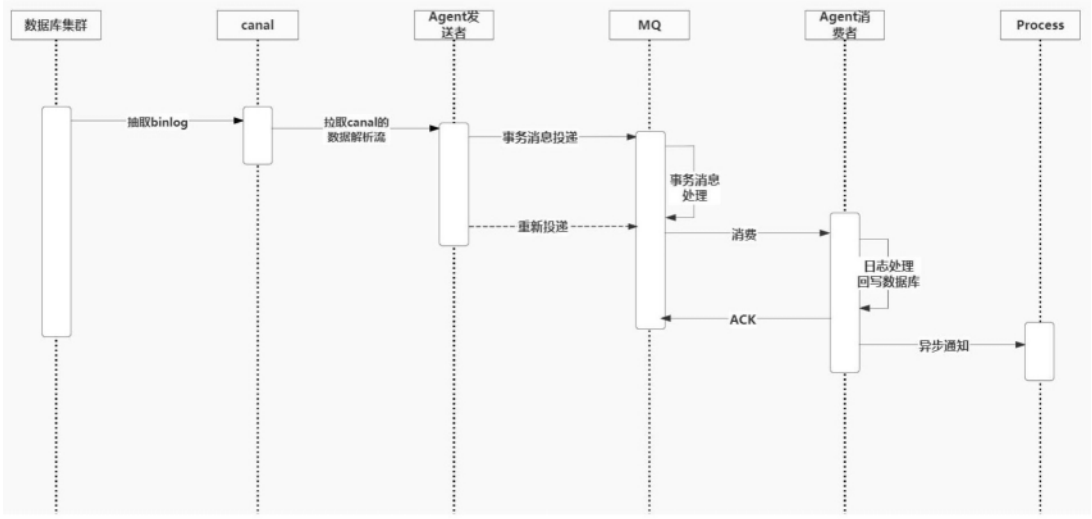


图5

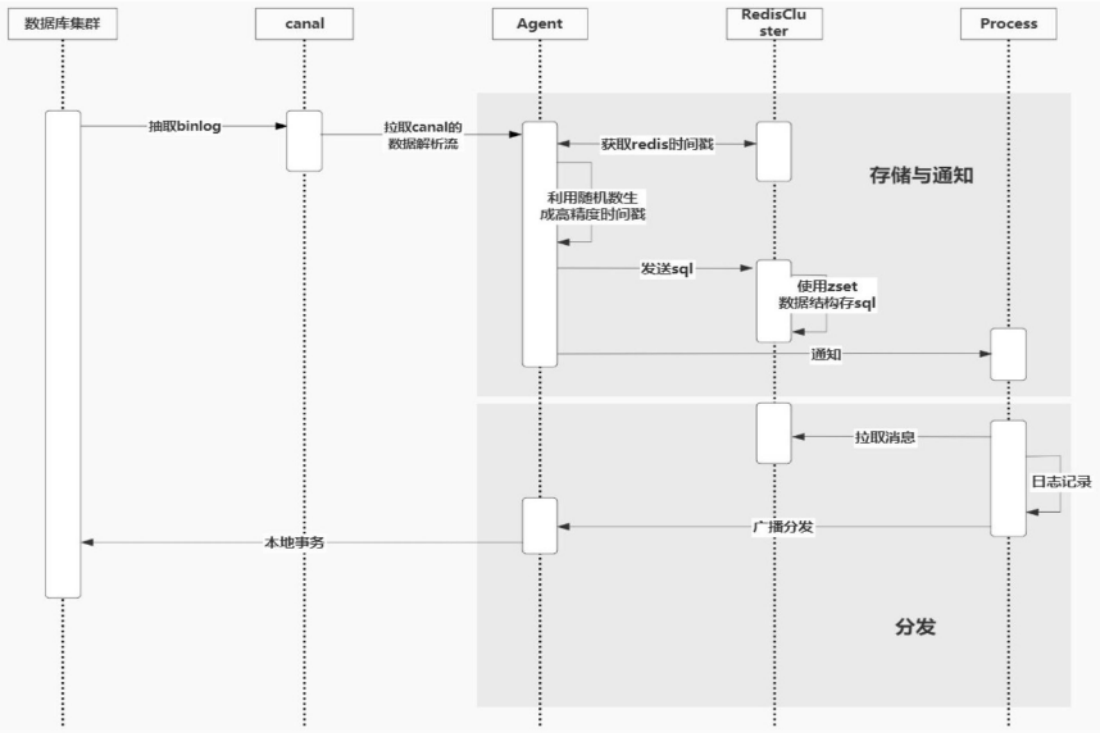

数据库集群
canal
Agent
RedisCluster
Process
抽取binlog
拉取canal的数据解析流
获取redis时间戳
存储与通知
利用随机数生成高精度时间戳
发送sql
使用zset数据结构存sql
通知
拉取消息
日志记录
广播分发
本地事务
分发

图6

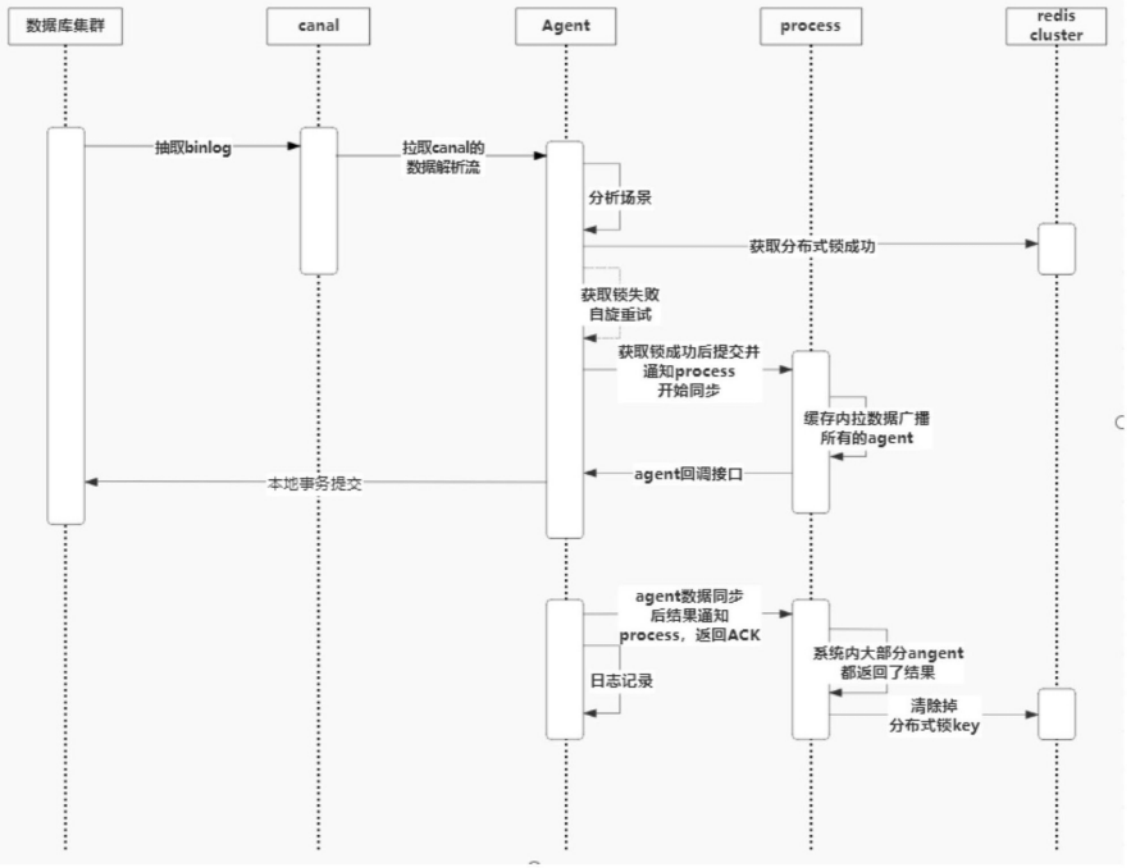

数据库集群
canal
Agent
process
redis cluster
抽取binlog
拉取canal的数据解析流
分析场景
获取分布式锁成功
获取锁失败自旋重试
获取锁成功后提交并通知process开始同步
缓存内拉数据广播所有的agent
agent回调接口
本地事务提交
agent数据同步后结果通知process，返回ACK
系统内大部分angent都返回了结果
日志记录
清除掉分布式锁key

图7

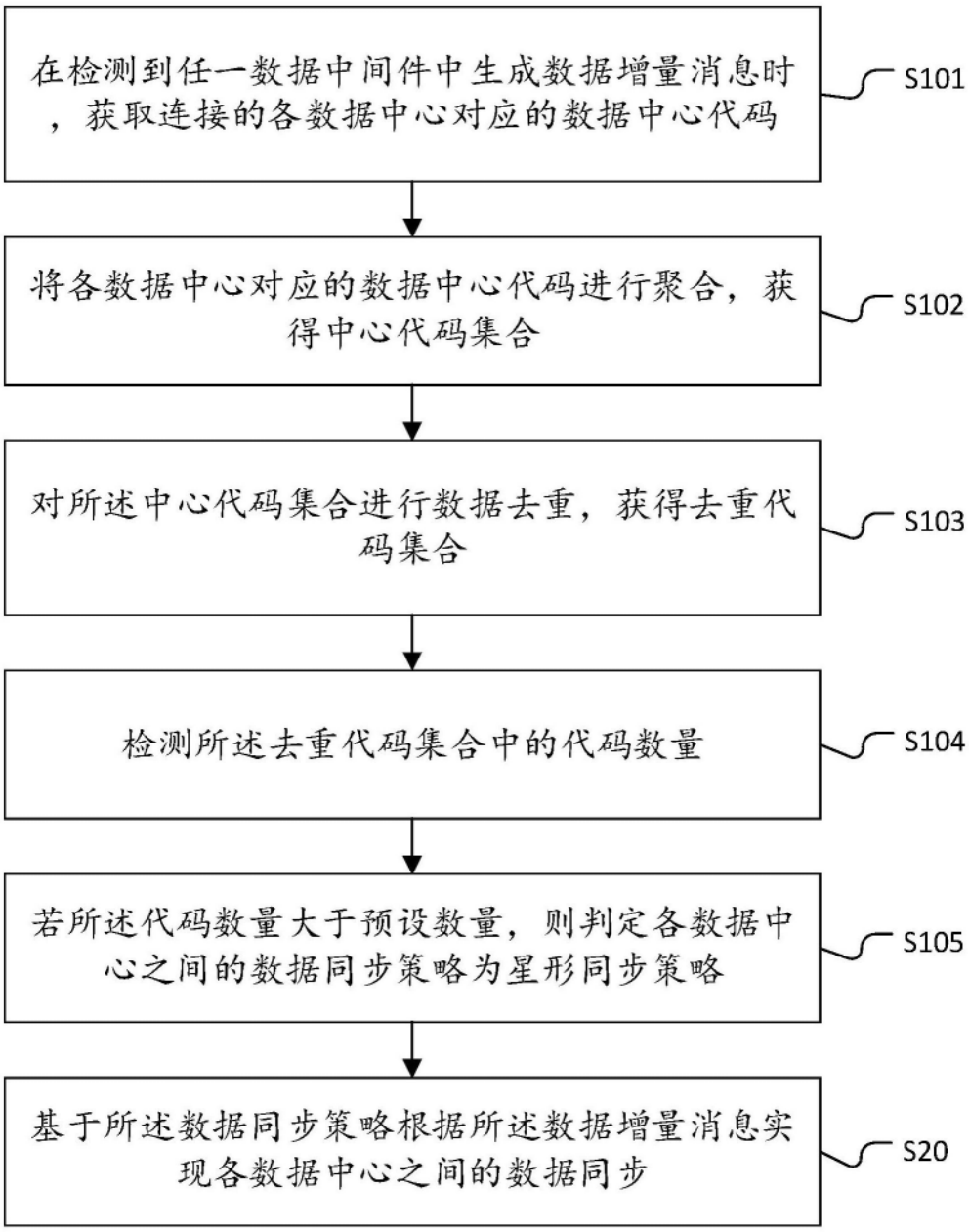

在检测到任一数据中间件中生成数据增量消息时，获取连接的各数据中心对应的数据中心代码
S101
将各数据中心对应的数据中心代码进行聚合，获得中心代码集合
S102
对所述中心代码集合进行数据去重，获得去重代码集合
S103
检测所述去重代码集合中的代码数量
S104
若所述代码数量大于预设数量，则判定各数据中心之间的数据同步策略为星形同步策略
S105
基于所述数据同步策略根据所述数据增量消息实现各数据中心之间的数据同步
S20

图8

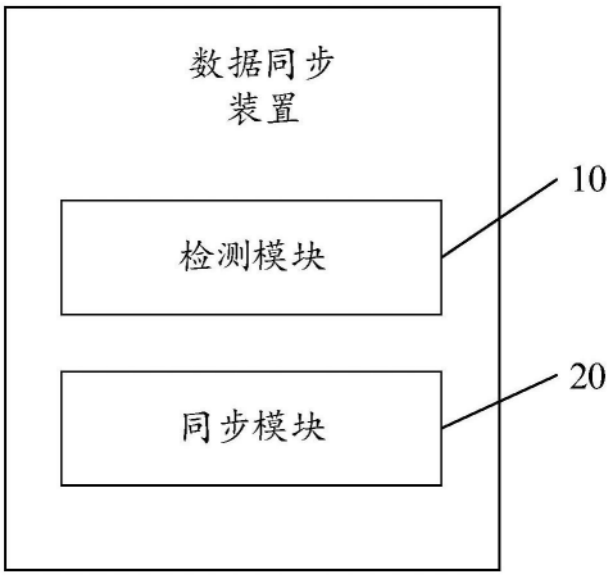


图9