(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114510539 B
(45)授权公告日 2022.06.24

(21)申请号 202210401171.X
(22)申请日 2022.04.18
(65)同一申请的已公布的文献号
申请公布号 CN 114510539 A
(43)申请公布日 2022.05.17
(73)专利权人 北京易鲸捷信息技术有限公司
地址 100089 北京市海淀区知春路128号4层401-7
(72)发明人 刘博 范振勇 李东卫 何振兴 莫荻 武新
(74)专利代理机构 四川言己律师事务所 51349
专利代理师 罗韬
(51)Int.Cl.
G06F 16/27(2019.01)
G06F 16/23(2019.01)
G06F 9/46(2006.01)
(56)对比文件
CN 113193947 A,2021.07.30
CN 113193947 A,2021.07.30
CN 112764888 A,2021.05.07
CN 109977171 A,2019.07.05
CN 114079660 A,2022.02.22
US 2012/0102006 A1,2012.04.26
审查员 王妍

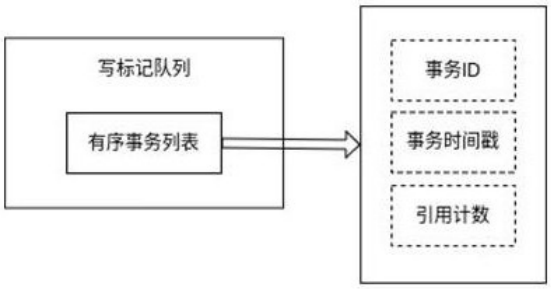
权利要求书2页 说明书6页 附图5页

(54)发明名称

分布式数据库一致性检查点的生成及应用方法

(57)摘要

本发明公开了一种分布式数据库一致性检查点的生成及应用方法，属及一种分布式数据库一致性检查方法，方法包括以数据库中的主键为关键字，对数据库中的数据进行分片，得到多个数据分片；每个数据分片通过其维护的事务写标记队列，辅助生成数据分片的一致性时间戳；当事务提交完成，数据成功写入后，由数据分片将所述逻辑指令传输至处理器；由汇聚器生成全局时间戳作为整个分布式数据库的一致性检查点。通过采用时钟机制生成全局时间戳作为整个分布式数据库的一致性检查点，避免了在全局事务管理器中的网络收发瓶颈，解决在主备模式下数据同步带来的性能消耗问题，更利于提升分布式数据库的性能。



CN 114510539 B

1.一种分布式数据库一致性检查点的生成方法,其特征在于所述的方法包括:

以分布式数据库中的主键为关键字,对分布式数据库中的存储单元进行分片,得到多个数据分片;

每个数据分片通过其维护的事务写标记队列,辅助生成数据分片的一致性时间戳,所述事务写标记队列的列表中包含事务ID、事务时间戳与引用计数;

当事务提交完成,数据成功写入后,针对每个数据分片生成逻辑指令,由数据分片将所述逻辑指令传输至处理器;

由所述处理器基于当前逻辑指令生成事件,将事件发送至事务写标记队列,然后所述事件发送至事务写标记队列通过消费逻辑指令的事件向前推动数据分片的一致性时间戳,得到新时间戳后,将其传输至汇聚器;

由汇聚器将多个所述新时间戳进行去重与排序处理后,生成全局时间戳,以所述全局时间戳作为整个分布式数据库的一致性检查点;

所述事务写标记队列包括消费逻辑指令与重计算时间戳;

所述消费逻辑指令为在事务中发生数据更新时,在有序的事务列表中找到对应的元素,并更新其引用计数,使当前数据分片的一致性时间戳向前推进;

所述重计算时间戳为在事务写标记队列的列表中取出时间戳最小的元素,与当前一致性时间戳进行比较,如经比较有向前推进,则更新所述一致性时间戳。

2.根据权利要求1所述的分布式数据库一致性检查点的生成方法,其特征在于:系统实时检测每个数据分片中的数据量,当一个数据分片中的数量过大时,将其分裂成两个新的分片;当两个在范围上相邻的两个数据分片数据量过少时,将两个数据分片合并为一个数据分片;

所述数据分片中数据量的大小通过预设的阈值确定。

3.根据权利要求1所述的分布式数据库一致性检查点的生成方法,其特征在于:所述消费逻辑指令在事务的引用计数变为0时,从列表中删除对应的事务。

4.根据权利要求1或3所述的分布式数据库一致性检查点的生成方法,其特征在于:在所述重计算时间戳的操作中,还采用自然时间戳推进所述一致性时间戳。

5.根据权利要求1所述的分布式数据库一致性检查点的生成方法,其特征在于:所述处理器在启动后首先等待数据分片注册,数据分片向处理器输出注册请求,数据分片在处理器完成注册后,由处理器处理数据分片传输的逻辑指令。

6.一种分布式数据库一致性检查点的应用方法,其特征在于所述的方法包括:

将权利要求1至5任意一项所获得的一致性检查点,连同序列号,通过多数派一致性协议一并传输至各个数据库节点;

所述数据库节点在接收到一致性检查点与序列号后,在暂存队列中判断序列号是否大于前一次接收到的一致性检查点的序列号,如判断结果为否,将当前一致性检查点与序列号暂存至接收队列中;如判断结果为是,则保存本次的一致性检查点并清空暂存队列。

7.根据权利要求6所述的分布式数据库一致性检查点的应用方法,其特征在于所述保存本次的一致性检查点为:在判断序列号大于前一次接收到的一致性检查点的序列号后,继续判断暂存队列中一致性检查点的数据分片数量与当前数据库节点上的分片数量是否一致;

如判断结果为是,则将当前的一致性检查点保存至当前数据库节点上;

如判断结果为否,则继续判断是否因当前数据库节点的数据分片未初始化导致的分片数量是否一致;

如判断结果为是,则将当前的一致性检查点保存至当前数据库节点上;

如判断结果为否,则继续判断是否因当前数据库节点上的数据分片在等待GC清理,或者在一致性检查点传输的过程中发生分裂,导致的分片数量是否一致;

如判断结果为是,则将当前的一致性检查点保存至当前数据库节点上;

如判断结果为否,则放弃保存本次的一致性检查点。

分布式数据库一致性检查点的生成及应用方法

技术领域

[0001] 本发明涉及一种分布式数据库技术，更具体的说，本发明主要涉及一种分布式数据库一致性检查点的生成及应用方法。

背景技术

[0002] 一致性检查点是数据库领域一个非常重要的技术。在传统的数据库产品中，它的主要形式是一个全局的事务ID，工作方式是将某个ID之前的数据全部刷到磁盘上，保证事务的一致性。对于单机的数据库来说，产生一个全局事务ID很简单，系统中维护一个自增的序列号就可以了，并发事务执行时都从这个单线程中获取到事务ID，不会产生冲突。

[0003] 对于分布式数据库，全局事务ID通常由一个全局事务管理器提供。在一个分布式的数据库集群中，只能同时有一个全局事务管理器提供服务，而为了避免单点故障，需要以主备的方式实现全局事务管理器的高可用。数据库集群中的其他节点或者服务，通过网络请求连接到全局事务管理器获取全局事务ID，全局事务管理器返回一个ID供其他服务使用。无论单机还是集群版本的数据库，都以全局事务ID作为一致性检查点，然后使用这个一致性检查点去实现其他的功能，比如快照、数据恢复等。然而这种方法存在不可避免的缺点：首先是网络收发瓶颈。一个分布式数据库中通常有多个数据节点，而只有一个全局事务管理器。在并发高的场景中，多个数据节点同时向全局事务管理器发送请求获取全局事务ID，导致全局事务管理器的网络达到瓶颈，而此时数据节点往往还有剩余的网络资源和计算资源，造成即使增加并发，也无法提高数据库的整体性能。其次是全局事务管理器自身的实现。为了避免单点故障导致数据丢失，需要以主备的方式部署全局事务管理器。当主服务分配了事务ID时，需要把这个ID同步到备服务上，这样才能保证在主服务故障时，备服务可以继续提供一致性的服务。而事务ID从主服务同步到备服务的过程，也占用了系统资源，进一步限制了数据库性能。因此有必要针对分布式数据库的一致性检查点的生成与应用作进一步的研究和改进。

发明内容

[0004] 本发明的目的之一在于针对上述不足，提供一种基于时钟机制的分布式数据库一致性检查点的生成方法，以期望解决现有技术中基于全局事务ID的一致性检查点方法在使用时容易出现网络收发瓶颈，主备的方式部署全局事务管理器占用过多系统资源，进一步限制了数据库的性能技术问题。

[0005] 为解决上述的技术问题，本发明采用以下技术方案：

[0006] 本发明一方面提供了一种分布式数据库一致性检查点的生成方法，所述的方法包括如下步骤。

[0007] 步骤A、以分布式数据库中的主键为关键字，对分布式数据库中的存储单元进行分片，得到多个数据分片。

[0008] 步骤B、每个数据分片通过其维护的事务写标记队列，辅助生成数据分片的一致性

时间戳，所述事务写标记队列的列表中包含事务ID、事务时间戳与引用计数。

[0009] 步骤C、当事务提交完成，数据成功写入后，针对每个数据分片生成逻辑指令，由数据分片将所述逻辑指令传输至处理器。

[0010] 步骤D、由所述处理器基于当前逻辑指令生成事件，将事件发送至事务写标记队列，然后所述事件发送至事务写标记队列通过消费逻辑指令的事件向前推动数据分片的一致性时间戳，得到新时间戳后，将其传输至汇聚器。

[0011] 步骤E、由汇聚器将多个所述新时间戳进行去重与排序处理后，生成全局时间戳，以所述全局时间戳作为整个分布式数据库的一致性检查点。

[0012] 作为优选，进一步的技术方案是：在步骤A中，系统实时检测每个数据分片中的数据量，当一个数据分片中的数量过大时，将其分裂成两个新的分片；当两个在范围上相邻的两个数据分片数据量过少时，将两个数据分片合并为一个数据分片；所述数据分片中数据量的大小通过预设的阈值确定。

[0013] 更进一步的技术方案是：在步骤B中，事务写标记队列包括消费逻辑指令与重计算时间戳；所述消费逻辑指令为在事务中发生数据更新时，在有序的事务列表中找到对应的元素，并更新其引用计数，使当前数据分片的一致性时间戳向前推进；重计算时间戳为在事务写标记队列的列表中取出时间戳最小的元素，与当前一致性时间戳进行比较，如经比较有向前推进，则更新所述一致性时间戳。

[0014] 更进一步的技术方案是：在步骤B中，当消费逻辑指令在事务的引用计数变为0时，从列表中删除对应的事务。

[0015] 更进一步的技术方案是：在步骤B中，重计算时间戳的操作中，还采用自然时间戳推进所述一致性时间戳。

[0016] 更进一步的技术方案是：在步骤D中，处理器在启动后首先等待数据分片注册，数据分片向处理器输出注册请求，数据分片在处理器完成注册后，由处理器处理数据分片传输的逻辑指令。

[0017] 本发明另一方面提供了一种分布式数据库一致性检查点的应用方法，该方法包括如下步骤。

[0018] 步骤F、将上述所获得的一致性检查点，连同序列号，通过多数派一致性协议一并传输至各个数据库节点。

[0019] 步骤G、所述数据库节点在接收到一致性检查点与序列号后，在暂存队列中判断序列号是否大于前一次接收到的一致性检查点的序列号，如判断结果为否，将当前一致性检查点与序列号暂存至接收队列中；如判断结果为是，则保存本次的一致性检查点并请空暂存队列。

[0020] 作为优选，更进一步的技术方案是：所述步骤G中保存本次的一致性检查点为：在判断序列号大于前一次接收到的一致性检查点的序列号后，继续判断暂存队列中一致性检查点的数据分片数量与当前数据库节点上的分片数量是否一致；如判断结果为是，则将当前的一致性检查点保存至当前数据库节点上；如判断结果为否，则继续判断是否因当前数据库节点的数据分片未初始化导致的分片数量是否一致。

[0021] 如判断结果为是，则将当前的一致性检查点保存至当前数据库节点上；如判断结果为否，则继续判断是否因当前数据库节点上的数据分片在等待GC清理，或者在一致性检

查点传输的过程中发生分裂,导致的分片数量是否一致。

[0022] 如判断结果为是,则将当前的一致性检查点保存至当前数据库节点上;如判断结果为否,则放弃保存本次的一致性检查点。

[0023] 与现有技术相比,本发明的有益效果之一是:通过采用时钟机制生成全局时间戳作为整个分布式数据库的一致性检查点,避免了在全局事务管理器中的网络收发瓶颈,解决在主备模式下数据同步带来的性能消耗问题,更利于提升分布式数据库的性能。并且采用对称式的集群架构实现分布式事务时间戳的生成,生成过程对业务是透明的,用户感知不到一致性检查点的工作任务,由此提升数据库产品给的用户带来的体验感。

附图说明

[0024] 图1为用于说明本发明一个实施例中事务写标记队列原理图。

[0025] 图2为用于说明本发明一个实施例中一致性检查点的生成流程图。

[0026] 图3为用于说明本发明另一个实施例中一致性检查点传输流程图。

[0027] 图4为用于说明本发明另一个实施例中数据库节点对一致性检查点的接收流程图。

[0028] 图5为用于说明本发明另一个实施例中数据库节点对一致性检查点的保存流程图。

具体实施方式

[0029] 本发明所称的名词解释如下。

[0030] MVCC:MVCC 的全称是多版本并发控制(Multi-Version Concurrency Control),即数据存储时加入版本号的信息,在做查询时通过版本号来控制可见性。版本号的实现方案主要有两种,一种是通过全局事务ID,辅助以事务的活跃性;另一种是时间戳,每个数据都要加入时间信息。

[0031] 写标记:代表事务临时未提交的状态。创建写标记后,数据库会检查是否存在事务冲突,冲突就重启当前事务。如果事务由于其他原因结束(比如违反约束),就终止事务。

[0032] 逻辑指令:事务写入完成后,会产生对本次事务的统计信息,包括事务ID、事务时间戳、写标记等。

[0033] 分片变更反馈:数据库存储数据按照主键进行分片,分片之间互相独立。当分片上面有数据修改时,会产生相应的反馈动作,表现为生成数据对应的写标记,并且将其记录在分片变更内部维护的队列中,该队列记录着每个写事务的时间戳和写标记的计数。

[0034] 下面结合附图对本发明作进一步阐述。

[0035] 在基于时钟机制的分布式数据库中,不使用全局事务ID而是使用了时间戳作为一致性检查点。每个节点都以本节点的时间来运行事务,在分布式事务中,通过节点之间的始终对齐来保证事务时间戳的有效以及事务的一致性。这样就避免了一个单点的全局服务造成的性能问题。

[0036] 本发明中提供的分布式数据库一致性检查点的生成方法是针对基于时钟机制的分布式数据库设计和实现的,并且数据库使用KV结构的存储引擎。

[0037] 数据库中的事务ACID特性,指的是原子性、一致性、隔离性和持久性。利用时钟的

机制，可以保证其中的原子性和隔离性。原子性是指事务是完整的，即事务中的数据，要么全部不写入，要么全部写入。隔离性是指事务之间是隔离的、独立的。时间戳记录了事务的执行顺序，可以按因果关系排序，也实现了数据的MVCC多版本记录，允许读写相同数据，实现并发，在某些场景下大幅提高性能。

[0038] 新型分布式数据库几乎都采用了KV结构的存储引擎，是由于它的弹性、可扩展性、部署和管理简单等诸多优点，而且经过大量优化后性能上也可以得到保证。在数据库保存一份数据时，以主键+时间戳的组合作为key，value就是序列化之后的数据内容。

[0039] 本发明的设计思想与架构为：事务有三种状态：Pending、Committed与Aborted。当事务创建时，事务的状态为Pending，并将其对数据的引用计数加n；当事务提交或者取消时，将其状态设置为Committed或者Aborted，相应的引用计数减n。其中n为该事务中涉及的KV的写标记数。因为数据库对写标记的管理是异步进行的，所以这里的计数器要独立于事务本身来做维护。因此本发明所涉及方法的主要思想就是通过分布式数据库对事务的控制机制，识别出每个事务的执行状态，使用时间戳将事务进行严格划分，划分的最后结果就是检查点。当事务状态发生变化时，时间戳逐渐向前推动。每次推动时间戳，取出最小值将其汇总保存起来，作为一致性检查点。

[0040] 基于上述设计思想与架构，本发明主要通过数据分片、事务写标记队列、数据分片变更反馈，来生成上述的一致性检查点。

[0041] 步骤S1为数据分片。

[0042] 数据库采用主键作为关键字，对数据的存储进行分片。主键可以由系统自动生成，也可以通过用户指定。实际使用中，使用系统自动生成的方式更加友好，提升易用性。

[0043] KV存储数据中的key值是有序存储的，因而数据分片很容易实现通过范围的方式来划分。这种方式，一方面能够加入对业务更好的支持，另一方面相比与Hash方式的分配，可以更高效地扫描数据。

[0044] 在本步骤中，优选的是数据分片可以完成自动的分裂与合并。当一个分片中的数据过大时，系统会检测到，并完成将其分裂成两个新的分片。这样可以保证每个分片中的数据较为均衡。而两个在范围上相邻的数据量较少的分片，会由系统自动的完成合并操作。分片数据量的大小通过预设的阈值参数控制。

[0045] 数据分片还可以根据负载做自动化调度。如果同一个节点上的多个数据分片成为业务的访问热点，导致某个节点的压力很大，此时就需要将该节点上高负载的数据分片迁移到其他节点上，降低该节点的运行压力。

[0046] 数据均衡和负载均衡是数据分片的重要功能，而分片作为数据访问的最小单元，支撑了上层复杂的事务实现，也提供了一致性检查点的生成环境。

[0047] 步骤S2为事务写标记队列。

[0048] 上述已经提到，写标记代表的是事务临时未提交的状态；对应到本步骤中，每个数据分片都维护一个事务写标记队列，其作用是辅助生成对应分片的一致性时间戳，结构如图1所示。在事务写标记队列中有一个有序的事务列表，按照事务的时间戳进行从小到大的排序，即列表中的第一个元素为时间戳最小的事务。列表中的每个元素都包括三个重要的字段：事务ID、事务时间戳和引用计数。

[0049] 在本步骤中，优选的是上述事务写标记队列主要包括消费逻辑指令和重计算时间

戳两个操作。

[0050] 其中消费逻辑指令发生在事务中发生数据更新时，比如事务写入一条数据或者异步清理写标记。一旦有数据更新，就会在有序的事务列表中找到对应的元素，并更新其引用计数。并且在该操作中，如果引用计数变为0，从列表中删除对应的事务。此时该数据分片的一致性时间戳就会向前推进。

[0051] 重计算时间戳操作为每个数据分片的一致性时间戳的计算通过跟踪写标记队列，从队列中取出第一个元素，即时间戳最小的元素，与旧的一致性时间戳做比较，如果有推进，则更新一致性时间戳。如果有序事务列表中没有事务时，会有另外一个自然时间戳推动一致性时间戳，避免一致性时间戳在没有业务运行时不会推进的问题。

[0052] 步骤S3为数据分片变更反馈。

[0053] 上述已经提到的，分片变更反馈为数据库存储数据按照主键进行分片，分片之间互相独立。在本步骤中，当事务提交完成，数据成功写入后，会针对每个涉及到的数据分片生成逻辑指令，该指令传入分片变更反馈模块。

[0054] 数据分片变更反馈从逻辑上划分成三个大模块：数据分片请求、处理器和汇聚器。

[0055] 如图2所示，变更反馈模块先启动一个处理器，等待数据分片的注册。注册到处理器的数据分片发送的请求才会被接收并处理。逻辑指令传到处理之后，经过生成事件、消费和推动时间戳等步骤，产生对应数据分片的事务一致性时间戳，即新时间戳。每个数据分片的新时间戳会发送到汇聚器中，由汇聚器做统一的去重、排序等处理，最终生成一个全局的时间戳。该全局的时间戳即为当前分布式数据库的一致性检查点。

[0056] 通过上述的方式得到分布式数据库的一致性检查点后，本发明的另一个实施例是一种分布式数据库一致性检查点的应用方法，该方法是将生成的一致性检查点发送至分布式数据库中的各个节点数据库中保存，并以此作为各个节点数据库的一致性检查依据。

[0057] 步骤S4为一致性检查点传输，分布式数据库通常使用多数派一致性协议(比如Paxos、Raft协议)传输用户数据。该协议采用多数派的算法，当集群中大多数节点达成一致后，就认为数据写入成功。

[0058] 在一致性检查点生成后，需要传输到个节点。如果使用普通的网络协议传输并保存，那么会导致尚未达成一致的少数派节点，接收到一致性时间戳时，事务的状态更新还没有同步到这些节点上，此时发生故障就会导致最终数据的不一致。基于前述的原因，一致性检查点也需要使用多数派一致性协议进行传输，具体如图3所示。

[0059] 步骤S5为一致性检查点保存，每个节点都会启动一个一致性点接收器，接收从数据分片发送过来的一致性检查点和序列号。在本步骤中，保存一致性检查点分接收流程和保存流程。

[0060] 上述的接收流程如图4所示。

[0061] 步骤S511、接收器接收到一致性检查点和序列号。

[0062] 步骤S512、如果序列号大于上一个接收到的值，那么表明本批次可能已经结束，尝试保存本批次的一致性检查点；否则将接收到的数据保存在暂存队列中，然后继续接收。

[0063] 步骤S513、尝试做本批次的一致性检查点的保存操作，具体的流程见下一节的流程。

[0064] 步骤S514、清空暂存队列中的数据，准备接收下一个批次的数据。

[0065] 步骤S515、结束本流程。

[0066] 上述的保存流程始于从接收器判断一个批次的一致性检查点接收完成，开始尝试保存操作，具体如图5所示：

[0067] 步骤S521、暂存队列中的一致性检查点也是从数据分片接收过来的，所以需要判断暂存队列中的分片数量与本节点上的分片数量是否一致。

[0068] 步骤S522、如果完全匹配，那么直接可以将一致性点保存至本节点。

[0069] 步骤S523、如果不匹配，再判断是否都是因为本节点的某些数据分片未初始化导致的，如果是，那么也可以将一致性检查点保存至本节点。

[0070] 步骤S524、如果不是因为本节点的某些数据分片未初始化导致的，那么继续判断是否是因为节点上的分片在等待GC清理或者是因为在一致性检查点传输的过程中发生的分裂，导致了分片不一致，如果是，那么也可以将一致性检查点保存至本节点。

[0071] 步骤S525、如果都不是因为上述情况，那么就放弃本批次的一致性检查点的保存。

[0072] 步骤S526、结束流程。

[0073] 基于本发明上述的实施例可知，本发明最大的改进之一是一致性检查点的生成方式与原理，通过对每个数据分片维护一个事务的写标记引用计数队列，推动队列中事务时间戳的方式产生一致性检查点；同时在一致性检查点生成后，通过多数派一致性协议将一致性检查点传输给所有的节点数据库；并且在节点数据库端对一致性检查点保存时，需要判断与本节点上的数据分片状态是否一致，以及可以忽略哪些不一致的状态等。因此本发明致力所解决的技术问题是在时钟机制下的分布式数据库的一致性检查点的产生过程，以及如何将其一致性地保存至每个节点上的运行机制。

[0074] 除上述以外，还需要说明的是在本说明书中所谈到的“一个实施例”、“另一个实施例”、“实施例”等，指的是结合该实施例描述的具体特征、结构或者特点包括在本申请概括性描述的至少一个实施例中。在说明书中多个地方出现同种表述不是一定指的是同一个实施例。进一步来说，结合任一实施例描述一个具体特征、结构或者特点时，所要主张的是结合其他实施例来实现这种特征、结构或者特点也落在本发明的范围内。

[0075] 尽管这里参照本发明的多个解释性实施例对本发明进行了描述，但是，应该理解，本领域技术人员可以设计出很多其他的修改和实施方式，这些修改和实施方式将落在本申请公开的原则范围和精神之内。更具体地说，在本申请公开、附图和权利要求的范围内，可以对主题组合布局的组成部件和/或布局进行多种变型和改进。除了对组成部件和/或布局进行的变型和改进外，对于本领域技术人员来说，其他的用途也将是明显的。

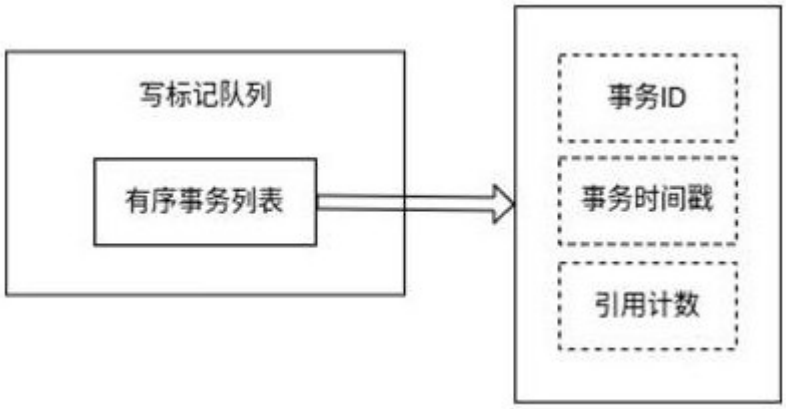


图1

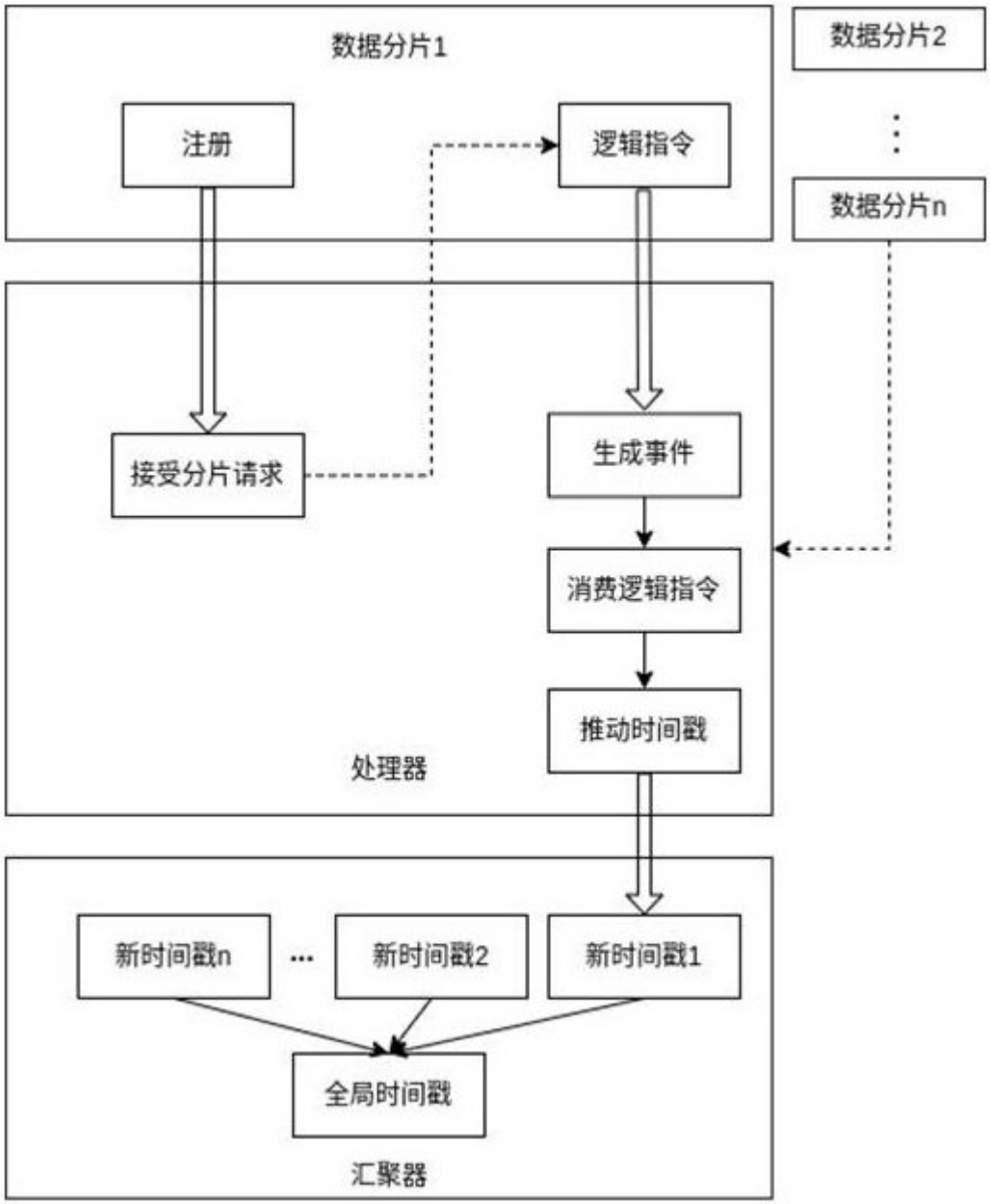

数据分片1
数据分片2
数据分片n
注册
逻辑指令
接受分片请求
生成事件
消费逻辑指令
推动时间戳
处理器
新时间戳n
...
新时间戳2
新时间戳1
全局时间戳
汇聚器

图2

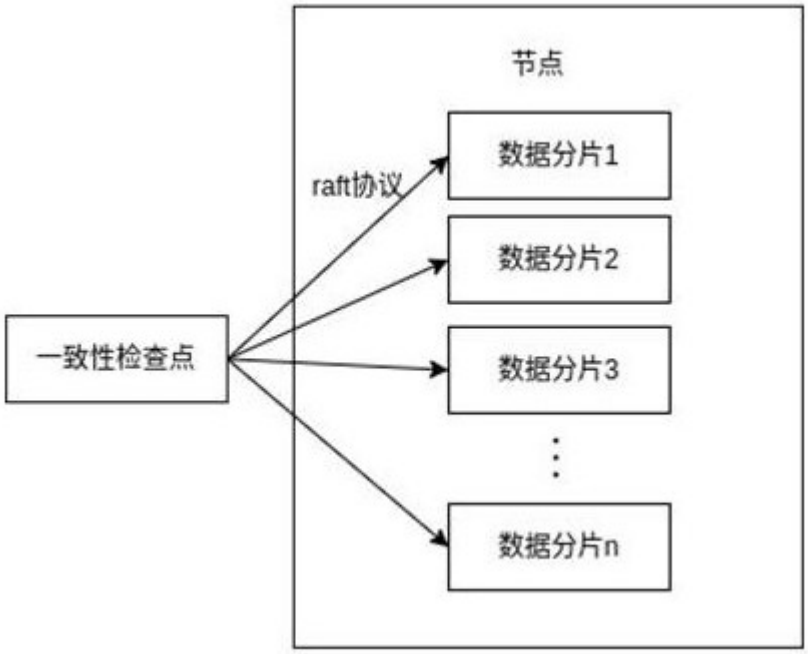


图3

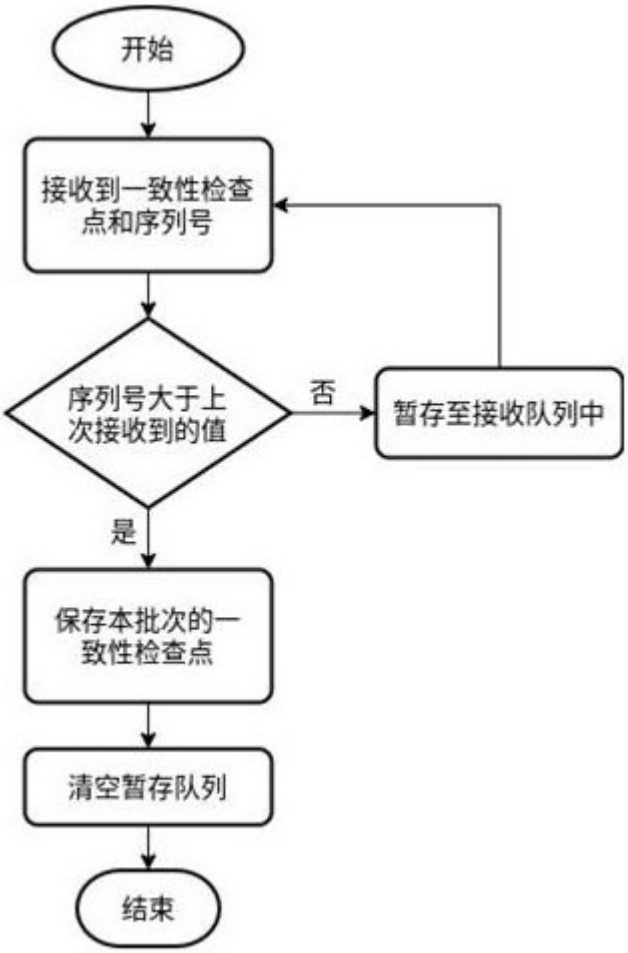

开始
接收到一致性检查点和序列号
序列号大于上次接收到的值
否
暂存至接收队列中
是
保存本批次的一致性检查点
清空暂存队列
结束

图4

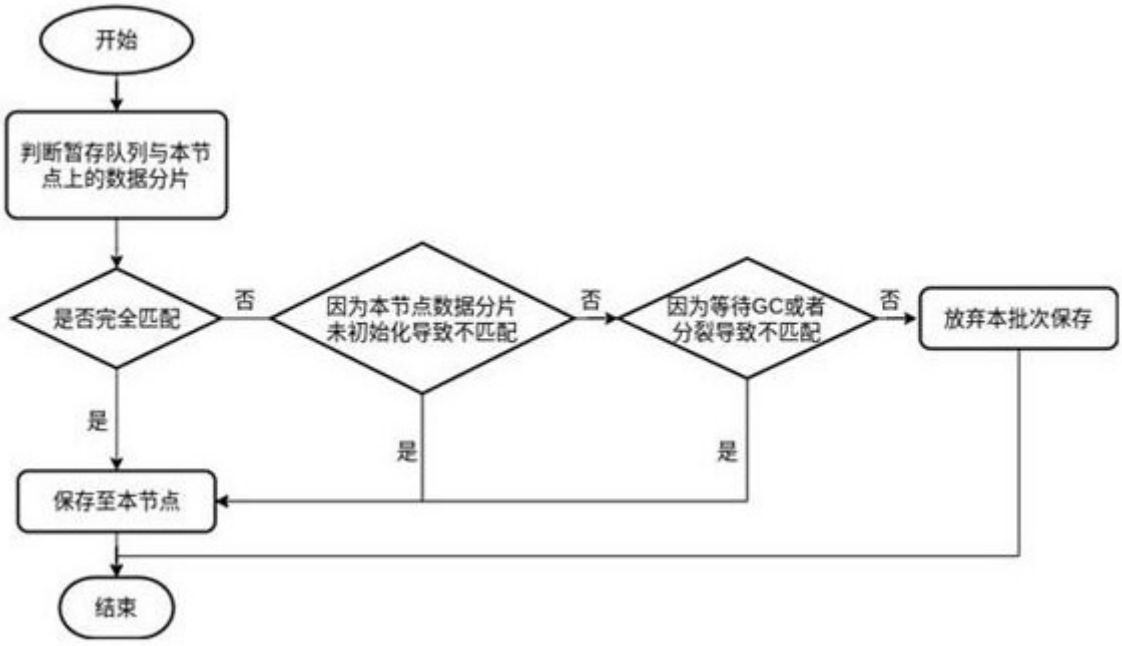

开始
判断暂存队列与本节点上的数据分片
是否完全匹配
否
因为本节点数据分片未初始化导致不匹配
否
因为等待GC或者分裂导致不匹配
否
放弃本批次保存
是
是
是
保存至本节点
结束

图5