

一种事务的处理方法、设备及存储介质

本申请要求于 2022 年 08 月 09 日提交中国专利局、申请号为 202210952643.0、申请名称为“一种事务的处理方法、设备及存储介质”的中国专利申请的优先权，其全部内容
5 通过引用结合在本申请中。

技术领域

本申请涉及数据库技术领域，尤其涉及一种事务的处理方法、设备及存储介质。

10 背景技术

数据库事务(transaction)是一组不可分割的 SQL 语句，这组 SQL 语句是一个逻辑工作单元。其中，涉及到分布式数据库中多个分片的数据库事务可被称为分布式事务，分布式事务可包括多个子事务，不同子事务可能需要访问不同的分片，每个子事务下可能包含至少一条语句逻辑。

15 分布式事务的一个例子是将钱从一个银行账号中转到另外一个银行账号中。此时通常包含两步操作：一条 UPDATE 语句负责从一个银行账号的总额中减去一定的钱数，另外一条 UPDATE 语句负责向另外一个银行账号中增加相应的钱数。减少和增加这两个操作必须永久性地记录到相关分片中，否则钱就会丢失。如果钱的转帐有问题，则必须同时取消减少和增加这两个操作。也即是，对分布式事务的处理需保证原子性。

20 目前，通常采用两阶段提交协议（two phase commit, 2PC）来保证分布式事务的原子性。两阶段提交协议的第一阶段是表决阶段，在该阶段，所有参与者（分片）都将当前子事务能否成功执行的反馈信息发送给协调者（分片）；第二阶段是提交阶段，在该阶段，协调者根据所有参与者发送的反馈信息，通知所有参与者步调一致地进行当前子事务的提交，或者进行当前子事务的回滚。因此，两阶段提交协议需要在事务处理路径中引入多个
25 网络往返，而且两阶段提交协议中的分布式事务持锁时间过长，导致对分布式事务的吞吐量不佳。

发明内容

本申请的多个方面提供一种事务的处理方法、设备及存储介质，用以改善对事务的处理效率。
30

本申请实施例提供一种事务的处理方法，适用于事务协调端，所述方法包括：
接收针对目标事务的处理请求；

若确定所述目标事务为分布式事务，则在所述目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁；

35 计算所述目标事务的写集；

释放所述读集中无需执行写操作的数据记录上的排它锁；

在所述目标事务的写环节中，基于排它锁完成针对所述写集的写操作后，释放相

关数据记录上的排它锁。

本申请实施例还提供一种事务的处理方法，适用于事务参与端，所述方法包括：

在目标事务为分布式事务的情况下，接收所述目标事务的事务协调端在所述目标事务的读环节发起的远程读操作；

5 在所述读环节中对所述远程读操作请求的数据记录添加排它锁；

在所述目标事务的写环节中，若接收到所述事务协调端发起的远程写操作，则响应所述远程写操作并释放对应数据记录上的排它锁。

本申请实施例还提供一种事务协调端，包括存储器、处理器和通信组件；

所述存储器用于存储一条或多条计算机指令；

10 所述处理器与所述存储器和所述通信组件耦合，用于执行所述一条或多条计算机指令，以用于：

通过所述通信组件接收针对目标事务的处理请求；

若确定所述目标事务为分布式事务，则在所述目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁；

15 计算所述目标事务的写集；

释放所述读集中无需执行写操作的数据记录上的排它锁；

在所述目标事务的写环节中，基于排它锁完成针对所述写集的写操作后，释放相关数据记录上的排它锁。

本申请实施例还提供一种事务参与端，包括包括存储器、处理器和通信组件；

20 所述存储器用于存储一条或多条计算机指令；

所述处理器与所述存储器和所述通信组件耦合，用于执行所述一条或多条计算机指令，以用于：

在目标事务为分布式事务的情况下，通过所述通信组件接收所述目标事务的事务协调端在所述目标事务的读环节发起的远程读操作；

25 在所述读环节中对所述远程读操作请求的数据记录添加排它锁；

在所述目标事务的写环节中，若接收到所述事务协调端发起的远程写操作，则响应所述远程写操作并释放对应数据记录上的排它锁。

本申请实施例还提供一种存储计算机指令的计算机可读存储介质，当所述计算机指令被一个或多个处理器执行时，致使所述一个或多个处理器执行前述的事务的处理方法。

30 在本申请实施例中，针对分布式事务，创新性地提出可在读环节分别为其读集中所涉及到的数据记录添加排它锁；在写环节可基于这些排它锁直接执行针对写集的写操作，并在完成写操作后立即释放相关数据记录上的排它锁。在写环节中为数据记录添加的排它锁可保持分布式事务对这些数据记录的独占性，这可避免写环节中发生写操作中止问题，也即是，分布式事务的所有写操作必然成功，这保证了分布式事务处理的原子性，另外，由于在写操作完成后立即释放排它锁，因此，分布式事务的持锁时间仅为执行其事务逻辑固有所需的时间，这大大缩短了分布式事务的持锁时间，从而可有效提高对分布式事务的吞吐量，改善事务处理效率。

附图说明

此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中：

- 图 1 为本申请一示例性实施例提供的一种事务的处理系统的结构示意图；
- 5 图 2 为本申请一示例性实施例提供的传统两阶段提交协议与本实施例的技术方案之间的对比示意图；
- 图 3 为本申请一示例性实施例提供的一种示例性的应用方案示意图；
- 图 4 为本申请一示例性实施例提供的一种持久化检查方案的应用示意图；
- 图 5 为本申请另一示例性实施例提供的一种事务的处理方法的流程示意图；
- 10 图 6 为本申请另一示例性实施例提供的另一种事务的处理方法的流程示意图；
- 图 7 为本申请又一示例性实施例提供的一种事务协调端的结构示意图；
- 图 8 为本申请又一示例性实施例提供的一种事务参与端的结构示意图。

具体实施方式

- 15 为使本申请的目的、技术方案和优点更加清楚，下面将结合本申请具体实施例及相应的附图对本申请技术方案进行清楚、完整地描述。显然，所描述的实施例仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有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。

目前，常用的两阶段提交协议需要在事务处理路径中引入多个网络往返，而且两阶段提交协议中的分布式事务持锁时间过长，导致对分布式事务的吞吐量不佳。为此，本申请的一些实施例中：针对分布式事务，可在读环节分别为其读集中所涉及到的数据记录添加排它锁；在写环节可基于这些排它锁直接执行针对写集的写操作，并在完成写操作后立即释放相关数据记录上的排它锁。在写环节中为数据记录添加的排它锁可保持分布式事务对这些数据记录的独占性，这可避免写环节中发生写操作中止问题，也即是，分布式事务的所有写操作必然成功，这保证了分布式事务处理的原子性，另外，由于在写操作完成后立即释放排它锁，因此，分布式事务的持锁时间仅为执行其事务逻辑固有所需的时间，这大大缩短了分布式事务的持锁时间，从而可有效提高对分布式事务的吞吐量，改善事务处理效率。

- 20 在开始描述技术方案之前，首先对可能涉及到的技术概念进行简要解释：
- 30 分布式系统：系统中的数据分别在不同的局部存储区（可称为分片）中存储、由不同的管理系统进行管理、在不同的机器上运行、由不同的操作系统支持、被不同的通信网络连接在一起。

事务：可以由一个有限的操作序列构成，操作序列中的元素为一个个的事务逻辑。事务可包括本地事务和分布式事务。在不同的应用场景中，事务的类型可不同。例如，在数据库场景中，对应的是数据库事务。又例如，在流处理场景中，对应的可以是流处理事务。

本地事务：仅限于对单一存储资源的访问控制。例如，对于分布式数据库来说，可将仅对某一个分片进行访问的数据库事务理解为是该分片上的本地事务。

分布式事务：是指是指事务的参与者、支持事务的服务器、资源服务器以及事务管理

器分别位于不同的分布式系统的不同节点之上。分布式事务中可包含多条语句逻辑，不同语句逻辑所访问的数据记录可能分布在不完全相同的分片上。

分布式事务的原子性：可理解为需要保证分布式事务中的多条语句逻辑要么全部提交，要么全部回滚，分布式事务需保证不同分片上的数据一致性。

- 5 排它锁：也称为X锁，若事务T对数据记录A加上排它锁，事务T可以读A也可以修改A，其他事务不能再对A加任何锁，直到T释放A上的排它锁。

共享锁：也称为S锁，若事务T对数据记录A加上共享锁，则事务T可以读A但不能修改A，在读A时，需要将S锁升级为X锁；其他事务只能再对A加共享锁，而不能加排它锁，直到T释放A上的共享锁。

- 10 以下结合附图，详细说明本申请各实施例提供的技术方案。

图1为本申请一示例性实施例提供的一种事务的处理系统的结构示意图，参考图1，该系统可包括：事务协调端和至少一个事务参与端。其中，事务协调端和事务参与端是基于工作角色而定义的。对于分布式系统来说，包含多个分片，每个分片可针对不同的事务适应性地承担协调端或参与端的工作角色。而对于不同的事务，所涉及到的事务协调端和事务参与端可能不同，所涉及到的分片的数量也可能不同。举例来说，某个分片对于事务T1来说可能是事务协调端，但该分片也可能是事务T2的事务参与端。实际应用中，可将拥有事务所需读取的第一条数据记录的分片作为该事务的事务协调端，而参与该事务处理工作的其它分片则可作为该事务的事务参与端，当然，这仅是示例性的，本实施例并不限于这一种工作角色定义方案。

- 20 对于一个事务来说，都可以按照读环节和写环节来执行。本实施例提供的事务的处理方案旨在避免在写环节中发生中止，以确保分布式事务在读环节之后可必然提交，从而保证分布式事务处理的原子性，并且摒弃了传统的两阶段提交方案而提出单阶段提交方案来缩短分布式事务的持锁时间，进而提高吞吐量。

为了便于描述，本实施例中将以目标事务为例，描述事务协调端和事务参与端在协同处理目标事务过程中实施方案，但应当理解的是，除了目标事务之外，本实施例提供的事务的处理系统中还可采用同样的技术方案处理其它事务，支持对事务的并发处理。

为此，参考图1，针对目标事务的处理请求可被发送到事务协调端上，这里，处理请求由目标事务的发起方发起。对于事务协调者来说，可判断目标事务是否为分布式事务。通常，一个事务是否为分布式事务，在执行之前是无法识别的，为此，在一种示例性方案中，事务协调端可在目标事务的读环节中，运行目标事务中的各条语句逻辑；若各条语句逻辑中存在远程读逻辑，则确定目标事务为分布式事务。举例来说，目标事务中可能包含3条语句逻辑，第一条语句逻辑中的读逻辑为本地读逻辑，也即需要读取的是属于事务协调端的本地数据记录，此时尚无法确定目标事务是否为本地事务；继续执行第二条语句逻辑，第二条语句逻辑中的读逻辑为远程读逻辑，也即需要读取的是目标事务的某个事务参与端上的数据记录，而不再是事务协调端的本地数据记录，则即可确定目标事务为分布式事务。其中，事务中的每条语句逻辑对应一条SQL查询语句，在分布式事务中，不同语句逻辑所对应的数据记录可能分布在不完全相同的分片上。

若目标事务为分布式事务，则可在目标事务的读环节中分别为其读集所涉及到的各条

数据记录获取排它锁。也即是，目标事务的每一次读操作，都会使得对应的数据记录上产生排它锁，排它锁可保证目标事务享有对这些数据记录的独占权。

在为读集所涉及到的各条数据记录获取排它锁的过程中，对于事务协调端来说，可确定读集中属于事务协调端的本地数据记录；为本地数据记录添加排它锁。事务协调端还可触发读集中各条非本地数据记录各自所属的事务参与端对相应的数据记录添加排它锁。为此，事务协调端可向事务参与端发起远程读操作，而对于事务参与端来说，可在读环节中对远程读操作请求的数据记录添加排它锁，以保证目标事务对这些数据记录的独占性。这样，通过目标事务的事务协调端及事务参与端之间的协同操作，可实现为目标事务的读集所涉及到的各条数据记录获取排它锁。

图2为本申请一示例性实施例提供的传统两阶段提交协议与本实施例的技术方案之间的对比示意图。参考图2，可明显地发现，在传统的两阶段协议下，分布式事务的读环节处于表决阶段中，且在读环节中为分布式事务的读集中涉及到的数据记录获取的是共享锁，分布式事务的写环节处于两阶段协议下的提交阶段中，参考图2，两阶段协议下，至少需要两个阶段的网络往返才能完成分布式事务的执行。而本实施例的技术方案中，则创新性地提出单阶段提交的方式，分布式事务的读环节和写环节均处于该单阶段中，且在分布式事务的读环节中为其读集中涉及到的数据记录获取的是排它锁，而非共享锁。参考图2，本实施例的技术方案中，只需单阶段的网络往返即可完成分布式事务的处理，这无形中就缩短了分布式事务所需的网络往返代价，因此可有效提高分布式事务的处理效率。

本实施例中，可在目标事务的读环节中运行完目标事务所有的语句逻辑，从而产生目标事务的完整读集，目标事务的读集可缓存在事务协调端上的指定数据结构中。另外，本实施例的技术方案中，允许目标事务中的读逻辑和写逻辑交织进行，这样，在读环节中运行目标事务的所有语句逻辑后，还可产生目标事务的写集，并可缓存在事务协调端中，供写环节调用。本实施例中，我们可假设目标事务的读集可覆盖其写集，也即是，写集所涉及到的数据记录是读集所涉及到的数据记录中的部分或全部。这种情况下，对读集获取的排它锁可保证写集的无中止写入。对于违反这种假设的情况，我们可称之为盲写，也即是，写集涉及到了读集中未涉及到的数据记录，这种情况下，本实施例提出了救济方案。一种示例性的救济方案可以是：事务协调者可检查目标事务的读集是否覆盖写集，如果不是，则可针对写集涉及到而读集未涉及到的数据记录执行虚拟读取，并为这类数据记录获取排它锁。也即是，在目标事务的读环节中，并不实际地读取这类数据记录，而仅是为这类数据记录获取排它锁。其中，若这类数据记录为不属于事务协调端的非本地数据记录，则可与其它正常的远程读操作一同批量执行获取排它锁的操作，在不存在其它正常的远程读操作的情况下，则需要为这类数据记录增加额外的往返行程作为违反假设的代价。

这样，本实施例中，可保证目标事务的写集涉及到的每条数据记录，均已在读环节中获得排它锁。

基于此，对于事务协调端来说，可在目标事务的写环节中，基于排它锁完成针对写集的写操作后，释放相关数据记录上的排它锁。其中，写操作是指按照写集中为数据记录计算出的新值对该数据记录进行更新。这里，事务协调端可对目标事务的写集进行批量处理，但是，应当理解的是，写集中涉及到的各条数据记录的排它锁释放过程是相互独立的，无任何依赖或同步关系，在单条数据记录的写入操作完成后，其上的排它锁可立即被释放，

而无需依赖其它数据记录上的锁释放情况。

在目标事务的写环节中，对于事务协调端来说，可确定写集中属于事务协调端的本地数据记录；基于本地数据记录上的排它锁完成写操作后，释放其排它锁。事务协调端还可针对写集中各条非本地数据记录向其各自所属的事务参与端发起写操作，以使事务参与端在响应写操作后释放相应数据记录上的排它锁。为此，事务协调端可向事务参与端发起远程写操作，而对于事务参与端来说，若接收到事务协调端发起的远程写操作，则可响应远程写操作并释放对应数据记录上的排它锁。这样，无论是本地写操作还是远程写操作，都可保证在写操作完成后立即释放对应数据记录上的排它锁。

可知，本实施例的技术方案通过在读环节中获取排它锁而确保分布式事务一旦开始写入就不会回滚，且在写入之后将立即释放排它锁。这样，分布式事务的持锁时间仅为其事务逻辑的实际执行时间。继续参考图 2，在传统的两阶段协议下，分布式事务 T1 对分片 P1 上对应数据记录的持锁时间为：读操作-结果-写请求-验证-同意-提交，时间非常长。再看本实施例的技术方案，分布式事务 T1 对分片 P1 上对应数据记录的持锁时间仅为：读操作-结果-写操作，持锁时间比传统的两阶段协议节省了一半。而单个事务的持锁时间减少，则可为更多的其它事务腾出处理时间，因此，可有效提高分布式事务处理的吞吐量。

正如前文提及的，我们假设了目标事务的读集可覆盖写集，因此，读集中可能不存在需要执行写操作的数据记录。本实施例中，可及时释放这部分数据记录上的排它锁，以减少面部事务对这部分数据记录的持锁时间。本实施例中，可在能够确定出读集中哪些数据记录无需执行写操作后，立即释放这些数据记录上的排它锁。一种示例性的释放时机可以是：在目标事务的写环节中，若确定读集中的任意本地数据记录无需执行写操作，则释放其上的排它锁。本实施例中，在进入写环节后，可及时发现读集中无需执行写操作的数据记录，从而可立即释放其上的排它锁。当然，本实施例中，释放读集中无需执行写操作的数据记录上的排它锁的时机并不限于此，还可设置为其它时机。

这样，可保证在读环节中为读集所涉及到的各条数据记录所获取的排它锁，都可及时得到释放，使得分布式事务对数据记录的持锁时间足够短。

据此，本实施例中，针对分布式事务，可在读环节分别为其读集中所涉及到的数据记录添加排它锁；在写环节可基于这些排它锁直接执行针对写集的写操作，并在完成写操作后立即释放相关数据记录上的排它锁。在写环节中为数据记录添加的排它锁可保持分布式事务对这些数据记录的独占性，这可避免写环节中发生写操作中止问题，也即是，分布式事务的所有写操作必然成功，这保证了分布式事务处理的原子性，另外，由于在写操作完成后立即释放排它锁，因此，分布式事务的持锁时间仅为执行其事务逻辑固有所需的时间，这大大缩短了分布式事务的持锁时间，从而可有效提高对分布式事务的吞吐量，改善事务处理效率。

本实施例提出的单阶段提交方案，对分布式事务非常友好，可有效避免分布式事务中的写中止问题。但是，发明人在研究过程中发现，单阶段提交方案中获取的排它锁会阻止其它事务对数据记录的访问，尤其是无辜的本地事务，而本地事务在所有事务中的占比还是非常可观的，本地事务的读环节会收到本实施例提出的单阶段提交方案中引入的排它锁的影响。

为此，在一种优选的设计方案中，提出针对本地事务和分布式事务采用不同处理模式

的技术构思。

承接上文中判断目标事务是否为分布式事务的处理逻辑，这里，若目标事务的各条语句逻辑中不存在远程读逻辑，则确定目标事务为本地事务；在目标事务为本地事务的情况下，可在目标事务的读环节中按照无锁的方式执行读操作，以获得目标事务的读集；在目标事务的写环节中，为其写集涉及到的各条数据记录获取排它锁以执行写操作。

这样，对于本地事务，可按照无锁的方式执行读操作，也即是，在读环节中无需为本地事务而获取任何锁，也无需关注所需读取的数据记录上是否存在锁，而是可默认数据记录上无锁以无障碍读取。因此，按照无锁的方式执行读操作可避免分布式事务处理过程中引入的排它锁对本地事务的读环节的影响。

本实施例中，可采用乐观并发控制（Optimistic Concurrency Control, OCC）技术来实现针对本地事务的读环节。采用的实现方案可包括但不限于 TICTOC、多版本并发控制 MVCC 等。

由于本地事务的读环节采用了无锁的方式，因此，可能在本地事务读的读操作发生后到写操作发生时这段时间内发生相关数据记录被更新的情况，而出现数据不一致问题。为了避免这一问题，本实施例中，可在本地事务的处理过程中增加验证环节，在验证环节中可对本地事务的读集进行一致性验证。本实施例中，可基于时间戳来实现对本地事务的读集进行一致性验证。为此，本实施例中提出可在分布式事务的处理过程中，引入时间戳，来支持本地事务和分布式事务之间的协调性。本实施例中，为分布式事务引入的时间戳可包括但不限于事务的逻辑时间戳、数据记录的读时间戳或数据记录的写时间戳等。

另外，这里有一种特殊情况，在引入对本地事务采用无锁方式执行读操作的方案后，对于一个分布式事务来说，在其包含的多条语句逻辑中的第一条远程读逻辑发生之前的本地读逻辑会被默认按照无锁的方式执行，这导致这部分本地读逻辑对应的数据记录上会缺失排它锁。为此，本实施例中，可对这部分本地读逻辑后补排它锁。举例来说，目标事务中有 3 条语句逻辑，其中的第一条语句逻辑为本地读逻辑，这导致在执行第一条语句逻辑时默认采用无法方式执行读逻辑，因此并未针对第一条语句逻辑执行任何加锁操作，导致第一条语句逻辑对应的数据记录上并未获得排它锁，而后我们根据第二条语句逻辑（远程读逻辑）才确定出目标事务为分布式事务，为保证分布式事务的读集所涉及到的各条数据记录均获取到排它锁，事务协调端可在第二条语句逻辑执行后，为第一条语句逻辑补加排它锁。另外，由于是对第一条语句逻辑对应的数据记录补加的排它锁，则需要对第一条语句逻辑对应的数据记录进行一致性验证，也即是，需要验证这条/些数据记录在第一条语句逻辑执行完读操作之后是否发生了更新，如果未发生更新，则可正常处理目标事务；而如果发生了更新，则需要中止或重试目标事务，以避免因第一条语句逻辑对应的数据记录发生更新而导致目标事务的处理结果错误。

本实施例中，目标事务的事务协调端可在目标事务的写环节，为目标事务分配逻辑时间戳。逻辑时间戳是基于事务之间的执行顺序而分配的，也即是，逻辑时间戳用于表征事务之间的执行顺序，其实质可理解为是一种编号。本实施例中，事务协调端在分配逻辑时间戳的过程中，需遵循一定的分配规则，分配规则可包括但不限于逻辑时间戳大于读集中各条数据记录中读时间戳和写时间戳中的最大值，等等，这可保证为事务分配的逻辑时间戳是单调增长的。这里应当理解的是，在进行逻辑时间戳分配时无需区分本地事务和分

布式事务，而是两种类型的事务混合在一起进行分配。

基于目标事务的逻辑时间戳，事务协调端可在写环节中，配置其本地数据记录对应的读时间戳和写时间戳。为此，本实施例中，若读集中属于事务协调端的本地数据记录的读时间戳小于目标事务的逻辑时间戳，则事务协调端可将本地数据记录的读时间戳更新为目标事务的逻辑时间戳；事务协调端还可将写集中属于事务协调端的本地数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。这可保证读集和写集中属于事务协调端的本地数据记录的读时间戳和写时间戳的新鲜度。另外，事务协调端还可将目标事务的逻辑时间戳发送给写集涉及到的各个事务参与端。则对于目标事务的事务参与端来说，可获取到事务协调端为目标事务分配的逻辑时间戳；在目标事务的读环节，若事务协调端发起的远程读操作对应的数据记录的读时间戳小于目标事务的逻辑时间戳，则将远程读操作对应的数据记录的读时间戳更新为目标事务的逻辑时间戳；在目标事务的写环节，将事务协调端发起的远程写操作对应的数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。这样，可保证目标事务的事务参与方上相关的数据记录的读时间戳和写时间戳的新鲜度。

基于此，分布式事务针对数据记录所产生的读时间戳和写时间戳可作为本地事务进行一致性验证的依据。这样，若确定目标事务为本地事务，则事务协调端可基于其读集中属于事务协调端的本地数据记录所拥有的读时间戳和/或写时间戳进行一致性验证。同理，对于目标事务的事务参与端来说，若事务参与端作为其它事务的事务协调端，则在其它事务为本地事务的情况下，可基于该其它事务的读集中的本地数据记录的读时间戳和/或写时间戳进行一致性验证。从而可及时发现本地事务的读集是否存在一致性问题的，如果存在，则可中止或重试本地事务。

另外，发明人在研究过程中发现，本实施例的技术方案中可能会发生死锁问题。死锁问题可理解为两个分布式事务均需要等待对方释放某个锁后才能继续执行。针对死锁问题，本实施例提供了一种示例性的解决方案：若目标事务的读环节中，在任意数据记录上发生死锁问题，则判断该数据记录上的当前事务的启动顺序编码是否小于目标事务的启动顺序编码，如果是，则目标事务中止或重试。同理，如果否，则可等待该数据记录上的锁释放，以继续处理目标事务，这是由于，该数据记录上的当前事务将因其启动顺序编码小于目标事务的启动顺序编码而被中止或重试，从而使该数据记录上的锁释放，也就解除了目标事务的执行障碍。其中，启动顺序编码是事务协调端按照其收到的事务启动请求而分配的，启动时间越来的事务可被分配到更小的启动顺序编码，启动顺序编码具有唯一性，这里未使用逻辑时间戳来处理死锁问题，主要是考虑到逻辑时间戳是在写环节中才分配的，而死锁问题是早在读环节中发生的。应当理解的是，本实施例中，仅在读环节中可能发生死锁问题，因为分布式事务可在写环节中基于排它锁而无中止写入，本地事务也可在写环节中通过添加排它锁而实现写入，因此，写环节中不会出现死锁问题。

本实施例中，我们可将针对分布式事务提出的单阶段提交方案作为分布式模式，而将针对本地事务采用的按照无锁方式执行写操作的方案作为本地模式。据此，可对本地事务和分布式事务采用不同的处理模式，以缓解分布式事务中引入的排它锁对本地事务读环节的影响，从而保证本地事务的处理效率；另外，通过在分布式事务的处理过程中引入时间戳，可支持本地事务对其读集进行一致性验证，从而支持本地事务与分布式事务之前的协

调性。

图3为本申请一示例性实施例提供的一种示例性的应用方案示意图。以下将结合图3来说明本实施例提供的事务处理逻辑。

参考图3，事务T读取x、y和z，并需更新y和z。其中，x、y和z分别为与分片1、
5 2、3上。分片1作为T的事务协调端，分片2和3作为T的事务参与端：

在T的读环节中，分片1首先以本地模式 (TicToc) 执行T，按照无锁的方式读取x，
其中，x的wts（写时间戳）=1，rts（读时间戳）=3；

当T尝试执行远程读取时，将切换到分布式模式，并补回x上的排它锁；

对x进行一致性验证，如果x发生了更新，则在分布式模式下直接中止和重试事务T；
10 否则，成功进入分布式模式并远程读取y和z；

分片2和分片3分别获取相应的排它锁，并返回包含数据记录的wts和rts的读结果，
以形成T的读集；参考图3，返回的数据记录y的wts和rts为[1, 3]，数据记录z的wts
和rts为[2, 4]；

按照事务逻辑，T计算其写集（在图3中为{y=10, z=20}），然后进入写环节；

15 在T的写环节中，分片1为T分配逻辑时间戳Ts，应该大于x的wts以及y和z的
rts，结果Ts=5；由于x的rts<Ts，因此，可将x的rts更新为5；

之后，分片1解锁x，并向分片2发送针对y的远程写入以及Ts=5，向分片3发送
针对z的远程写入以及Ts=5；

分片2在收到针对y的远程写入时，将y的值更新为10，并初始化y的rts=wts=5，
20 并解锁y；

分片3在收到针对z的远程写入时，将z的值更新为20，并初始化z的rts=wts=5，
并解锁z。

以上示例性地呈现了在事务T为分布式事务的情况下的事务处理过程，分布式事务处
理过程中配置的各种时间戳可支持其它本地事务的处理过程。可知，分布式事务对上述的
25 数据记录x、y和z的持锁时间已经最小化，结合前述实施例中提出的对本地事务采用无锁
方式进行写操作的方案，可有效改善分布式事务对本地事务的影响，因此，整体上可有效
提升事务处理的吞吐量。

正如前文提及的，本实施例为分布式事务提供的单阶段提交方案中，分布式事务在执
行完写操作后立即释放相关排它锁，而无需等待其日志记录被持久化。为此，本实施例中，
30 还提出了针对分布式事务的持久化检查环节。一种示例性的持久化检查方案可以是：基于
前述为事务分配的逻辑时间戳，检查目标事务在其事务协调端和事务参与端上是否全部完
成持久化。其中，持久化是指：针对目标事务，其事务协调端和事务参与端上的日志管理
器会自动地将目标事务对应的日志记录持久化到硬盘中，而只有完成日志记录的持久化后，
才认为目标事务完成了全局持久化。但是由于不同分片上的日志管理器的工作效率不同等
35 原因，导致不同分片对目标事务的持久化工作的进度不同，因此，需要检查目标事务是否
其所涉及到的所有分片上完成全局持久化。

在该示例性的方案中，以目标事务的事务协调端为例，可基于事务协调端上所处

理的各事务各自对应的逻辑时间戳，确定事务协调端上未持久化事务中的最小逻辑时间戳，作为事务协调端上的分片时间戳；获取其它分片所提供的分片时间戳；将所有分片各自对应的分片时间戳中的最小值，作为全局时间戳；若目标事务的逻辑时间戳小于全局时间戳，则确定目标事务已完成全局持久化处理。这里，分布式系统中的每个分片均可确定出自身的分配时间戳，并可获取所有其它分配提供的分片时间戳，并计算出自己认为的全局时间戳。值得说明的是，由于处理的事务不同，处理进度不同等原因，分布式系统中的不同分片各自计算出的全局时间戳可能并不相同，这是正常的。

其中，对于单个分片来说，可定期检查本地未持久化事务中的最小逻辑时间戳，以产生分片时间戳。而在分片上的分片时间戳发生变化的情况下，可向其它分片提供新的分片时间戳，这样，在分片的自身发生分片时间戳变化或者接收到其它分片提供的新的分片时间戳的情况下，可触发分片重新计算自己认为的全局时间戳，从而可保证各个分片上的全局时间戳的新鲜度。

正如前文提及的，逻辑时间戳可用于表征事务之间的执行顺序。本实施例中，分片时间戳是逻辑时间戳的阈值，因此，分片时间戳具有单调性。这种单调性可保证各个分片计算出的全局时间戳能够准确地反映出分布式系统各个分片上的事务持久化情况，进而保证能够准确地确定出目标事务是否已完成全局的持久化。

但是，基于逻辑时间戳进行持久化检查的过程中，可能存在两种问题：第一种是，某些未持久化事务可能尚未分配到逻辑时间戳，导致其所处分片无法确定出最小逻辑时间戳；第二种是，在分片上计算出分片时间戳后，分片上新的事务可能被分配一个小于分片时间戳的逻辑时间戳，导致破坏了分片时间戳的单调性。

针对第一种问题，还是以目标事务的事务协调端为例，一种示例性的解决方案可以是：若目标事务的事务协调端上存在未分配逻辑时间戳的特殊事务，则将特殊事务第一个访问的数据记录上的写时间戳作为特殊事务的逻辑时间戳下限，逻辑时间戳下限代替逻辑时间戳而作为确定分片时间戳的依据。当然，对于分布式系统中的其它分片，也可采用该示例性的解决方案，以解决上述的第一种问题。

针对第二种问题，一种示例性的解决方案中，可强制为新的事务分配的逻辑时间戳大于其所在分片上的分片时间戳。为此，我们考虑两种情况，还是以目标事务的事务协调端为例：一种情况下，在确定出事务协调端上的分片时间戳后，若事务协调端后续作为新的事务的协调端，则其为新的事务分配的逻辑时间戳不小于分片时间戳；另一种情况下，若事务协调端后续作为新的事务的参与端，则将其在新的事务写集中的所承载的数据记录的写时间戳更新为不小于其自身的分片时间戳，以驱使新的事务的协调端为新的事务分配不小于分片时间戳的逻辑时间戳。这里与前文中为事务分配逻辑时间戳时的逻辑呼应，也即是在为事务分片逻辑时间戳时，需保证分配的逻辑时间戳大于读集中数据记录的写时间戳，因此，这里通过合理维护新的事务涉及到的数据记录的写时间戳来间接影响为新的事务所分配的逻辑时间戳。

可知，通过上述两种问题下提出的解决方案，可保证各个分片上计算出的分片时间戳的单调性。这保证了分片时间戳的合理性，进而可保证全局时间戳的合理性。

图4为本申请一示例性实施例提供的一种持久化检查方案的应用示意图。参考图4，

分布式系统中的分片 P1、分片 P2 和分片 P3 上分别维护一个最新的分片时间戳表格，该表格中记录了分片 P1、分片 P2 和分片 P3 提供的最新的分片时间戳，各个分片上的表格可持续更新，并通过获取表格中最小的数字来计算全局时间戳。例如，分区 P1 接收到分片 P2 提供的分片时间戳 $Wp_2 = 5$ 和分片 P3 提供的分片时间戳 $Wp_3 = 6$ 后，结合自己的分片时间戳 $Wp_1 = 4$ ，P1 计算出全局时间戳 $Wg = 4$ 。分片时间戳不必同步广播（例如，P3 可能比其他分片晚广播 $Wp_3 = 14$ ），并且消息也被允许延迟。尽管如此，每个分片不会阻止来自其他分片的等待消息，因为它们中的每个分片都独立地处理和保留事务，并且仅异步地通信“分片时间戳”。

这里，对于目标事务来说，仅由其事务协调端来判断目标事务的逻辑时间戳是否小于事务协调端计算出的全局时间戳，若是，则事务协调端可确定目标事务已完成全局持久化处理，而不再需要目标事务的事务参与端来执行重复的判断。但是，目标事务的事务参与端也会计算自己认为的全局时间戳，这是由于目标事务的事务参与端有可能还作为其它事务的事务协调端，其计算出的全局时间戳需作为针对这类其它事务的持久化检查过程的依据。

可知，本实施例中，分布式系统中的各个分片能够独立进行全局时间戳的计算，并且只要求每个分片定期广播其最新持久化事务的时间戳即可。这样，每个分片都能够学习其他分片的分片时间戳并计算全局时间戳，即所有分片的分片时间戳中的最小值。随着时间戳单调增长，可保证时间戳低于全局时间戳的事务在所有分片上都已完成持久化。总体而言，通过从分布式事务的持锁期间摒弃掉持久化检查逻辑，而通过额外的单独过程来实现持久化检查，可有效减少分布式事务的持锁时间。而且，通过允许分片独立分组提交，通过这种已补的组提交方式可避免全局协调环节，减少了持久化校验过程的代价，这可有效提高分布式系统的可扩展性，在分布式系统发生扩展的情况下，也不会因为新增分片而导致持久化校验过程的代价出现增加。

另外，发明人在研究过程中发现，分布式系统中可能出现局部故障问题，也即是部分分片上可能发生宕机等故障。本实施例中提出，可基于前述的全局时间戳而解决分布式系统中的故障恢复问题。一种示例性的解决方案可以是：在发生分片故障的情况下，获取恢复时间戳，恢复时间戳为所有分片上确定出的全局时间戳中的最大值；将恢复时间戳之后处理的事务进行回滚。其中，恢复时间戳选用所有分片上确定出的全局时间戳中的最小值的原因在于：每个分片在正常操作期间独立地维护自己视角的全局时间戳，所有全局时间戳可以作为有效的候选者；为了减少要回滚的事务数，所有分区都采用了最大的一个作为约定的恢复时间戳。在该示例性方案中，可采用 Zookeeper 中的共识机制确保所有分区采用相同的恢复时间戳。然后，具有逻辑时间戳 $Ts \geq$ 恢复时间戳的事务可以安全地回滚。对于发生故障的分片，则在恢复后重新加入 Zookeeper 集群以获取约定的恢复时间戳，然后为所有 $Ts <$ 恢复时间戳的事务重新进行持久化，并回滚 $Ts \geq$ 恢复时间戳的事务。为了限制恢复的时间，每个分片都会定期保留其状态的检查点，这样它只需要在检查点之后进行重新持久化即可恢复。

本实施例中，还可能涉及到事务处理过程中的一些典型问题，包括但不限于针对只读事务的处理问题、针对数据范围的读环节的处理问题、约束检查问题等。这些问题的解决方案与本实施例的技术方案可以是相交的，在此仅进行简要的示例性说明，

而不过多详述。例如，针对只读事务的处理问题，一种示例性的解决方案可以是：目标事务的事务协调端根据其事务逻辑判断其为只读事务的情况下，可为目标事务配置只读标识，并按照无锁的方式执行目标事务的读环节。可采用快照等技术优化只读事务，实现无锁读取。再例如，针对数据范围的读环节的处理问题，一种示例性的解决方案可以是：若目标事务为分布式事务且其读环节中涉及到对数据范围的读操作，则其事务协调者可在目标事务的读环节中，为对应的数据范围获取排它的范围锁（predicate locks），以保证目标事务对该数据范围的独占性。

应当理解的是，在事务处理过程中的典型问题并不仅限于此，还可能存在更多问题，这些问题的解决方案可以本实施例提供的技术方案相互适配，相互协作，在此不再对这些典型问题进行穷举。

图5为本申请另一示例性实施例提供的一种事务的处理方法的流程示意图，该方法可由数据处理装置执行，该数据处理装置可实现为软件和/或硬件的结合，该数据处理装置可集成在事务的事务协调端中。参考图5，该方法可包括：

步骤500、接收针对目标事务的处理请求；

15 步骤501、若确定目标事务为分布式事务，则在目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁；

步骤502、计算目标事务的写集；

步骤503、释放读集中无需执行写操作的数据记录上的排它锁；

20 步骤504、在目标事务的写环节中，基于排它锁完成针对写集的写操作后，释放相关数据记录上的排它锁。

在一可选实施例中，步骤在目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁，包括：

确定读集中属于事务协调端的本地数据记录；

为本地数据记录添加排它锁；

25 触发读集中各条非本地数据记录各自所属的事务参与端对相应的数据记录添加排它锁。

在一可选实施例中，步骤释放读集中无需执行写操作的数据记录上的排它锁，包括：

30 在目标事务的写环节中，若确定读集中的任意本地数据记录无需执行写操作，则释放其上的排它锁。

在一可选实施例中，该方法还可包括：

在目标事务的读环节中，运行目标事务中的各条语句逻辑；

若各条语句逻辑中存在远程读逻辑，则确定目标事务为分布式事务。

在一可选实施例中，该方法还包括：

35 若各条语句逻辑中不存在远程读逻辑，则确定目标事务为本地事务；

在目标事务为本地事务的情况下，在目标事务的读环节中按照无锁的方式执行读操作，以获得目标事务的读集；

在目标事务的写环节中，为其写集涉及到的各条数据记录获取排它锁以执行写操作。

在一可选实施例中，步骤计算目标事务的写集，包括：

在目标事务的读环节中，计算目标事务的写集；

将写集缓存在事务协调端的本地，以供写环节调用。

- 5 在一可选实施例中，步骤在目标事务的写环节中，基于排它锁完成针对写集的写操作后，释放相关数据记录上的排它锁，包括：

在目标事务的写环节中，确定写集中属于事务协调端的本地数据记录；

基于本地数据记录上的排它锁完成写操作后，释放其排它锁；

针对写集中各条非本地数据记录向其各自所属的事务参与端发起写操作，以使事务参与端在响应写操作后释放相应数据记录上的排它锁。

- 10 在一可选实施例中，该方法还包括：

在目标事务的写环节，为目标事务分配逻辑时间戳，逻辑时间戳是基于事务之间的执行顺序而分配的；

若读集中属于事务协调端的本地数据记录的读时间戳小于目标事务的逻辑时间戳，则将本地数据记录的读时间戳更新为目标事务的逻辑时间戳；

- 15 将写集中属于事务协调端的本地数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。

在一可选实施例中，该方法还包括：

20 将目标事务的逻辑时间戳发送给写集涉及到的事务参与端，以供事务参与端基于目标事务的逻辑时间戳更新其在读集和/或写集中承载的数据记录的读时间戳和/或写时间戳。

在一可选实施例中，该方法还包括：

若确定目标事务为本地事务，则基于其读集中属于事务协调端的本地数据记录所拥有的读时间戳和/或写时间戳进行一致性验证。

在一可选实施例中，该方法还包括：

- 25 基于事务协调端上所处理的各事务各自对应的逻辑时间戳，确定事务协调端上未持久化事务中的最小逻辑时间戳，作为事务协调端上的分片时间戳；

获取其它分片所提供的分片时间戳；

将所有分片各自对应的分片时间戳中的最小值，作为全局时间戳；

- 30 若目标事务的逻辑时间戳小于全局时间戳，则确定目标事务已完成全局持久化处理。

在一可选实施例中，该方法还包括：

若事务协调端上存在未分配逻辑时间戳的特殊事务，则将特殊事务第一个访问的数据记录上的写时间戳作为特殊事务的逻辑时间戳下限，逻辑时间戳下限代替逻辑时间戳而作为确定分片时间戳的依据。

- 35 在一可选实施例中，该方法还包括：

在确定出事务协调端上的分片时间戳后，若事务协调端后续作为新的事务的协调端，则其为新的事务分配的逻辑时间戳不小于分片时间戳；

若事务协调端后续作为新的事务的参与端，则将其在新的事务写集中的所承载的数据记录的写时间戳更新为不小于其自身的分片时间戳，以驱使新的事务的协调端为

新的事务分配不小于分片时间戳的逻辑时间戳。

在一可选实施例中，该方法还包括：

在发生分片故障的情况下，获取恢复时间戳，恢复时间戳为所有分片上确定出的全局时间戳中的最大值；

- 5 将恢复时间戳之后处理的事务进行回滚。

值得说明的是，上述关于事务的处理方法各实施例中的技术细节，可参考前述的系统实施例中关于事务协调端的相关描述，为节省篇幅，在此不再赘述，但这不应造成本申请保护范围的损失。

- 10 图6为本申请另一示例性实施例提供的另一种事务的处理方法的流程示意图，该方法可由数据处理装置执行，该数据处理装置可实现为软件和/或硬件的结合，该数据处理装置可集成在事务的事务参与端中。参考图6，该方法可包括：

步骤600、在目标事务为分布式事务的情况下，接收目标事务的事务协调端在目标事务的读环节发起的远程读操作；

步骤601、在读环节中对远程读操作请求的数据记录添加排它锁；

- 15 步骤602、在目标事务的写环节中，若接收到事务协调端发起的远程写操作，则响应远程写操作并释放对应数据记录上的排它锁。

在一可选实施例中，该方法还包括：

获取事务协调端为目标事务分配的逻辑时间戳，逻辑时间戳是基于事务之间的执行顺序而分配的；

- 20 在读环节，若远程读操作对应的数据记录的读时间戳小于目标事务的逻辑时间戳，则将远程读操作对应的数据记录的读时间戳更新为目标事务的逻辑时间戳；

在写环节，将远程写操作对应的数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。

在一可选实施例中，该方法还包括：

- 25 若事务参与端作为其它事务的事务协调端，则在其它事务为本地事务的情况下，基于其读集中的本地数据记录的读时间戳和/或写时间戳进行一致性验证。

在一可选实施例中，该方法还包括：

基于事务参与端上所处理的各事务各自对应的逻辑时间戳，确定事务参与端上未持久化事务中的最小逻辑时间戳，作为事务参与端上的分片时间戳；

- 30 将自身的分片时间戳提供给目标事务的事务协调端，以供事务协调端将所有分片各自对应的分片时间戳中的最小值作为全局时间戳，并在目标事务的逻辑时间戳小于全局时间戳的情况下确定目标事务已完成全局持久化处理。

在一可选实施例中，该方法还包括：

- 35 在确定出分片时间戳后，若事务参与端后续作为新的事务的协调端，则为其分配的逻辑时间戳不小于分片时间戳；

若事务参与端后续作为新的事务的参与端，则将其在新的事务写集中的所承载的数据记录的些时间戳更新为不小于分片时间戳，以驱使新的事务的协调端为新的事务分配不小于分配时间戳的逻辑时间戳。

值得说明的是，上述关于事务的处理方法各实施例中的技术细节，可参考前述的

系统实施例中关于事务参与端的相关描述，为节省篇幅，在此不再赘述，但这不应造成本申请保护范围的损失。

另外，在上述实施例及附图中的描述的一些流程中，包含了按照特定顺序出现的多个操作，但是应该清楚了解，这些操作可以不按照其在本文中出现的顺序来执行或并行执行，操作的序号如 501、502 等，仅仅是用于区分开各个不同的操作，序号本身不代表任何的执行顺序。另外，这些流程可以包括更多或更少的操作，并且这些操作可以按顺序执行或并行执行。

图 7 为本申请又一示例性实施例提供的一种事务协调端的结构示意图。如图 7 所示，该事务协调端可包括：存储器 70、处理器 71 以及通信组件 72。

10 处理器 71，与存储器 70 和通信组件 72 耦合，用于执行存储器 70 中的计算机程序，以用于：

通过通信组件 72 接收针对目标事务的处理请求；

若确定目标事务为分布式事务，则在目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁；

15 计算目标事务的写集；

释放读集中无需执行写操作的数据记录上的排它锁；

在目标事务的写环节中，基于排它锁完成针对写集的写操作后，释放相关数据记录上的排它锁。

20 在一可选实施例中，处理器 71 在目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁的过程中，可用于：

确定读集中属于事务协调端的本地数据记录；

为本地数据记录添加排它锁；

触发读集中各条非本地数据记录各自所属的事务参与端对相应的数据记录添加排它锁。

25 在一可选实施例中，处理器 71 在释放读集中无需执行写操作的数据记录上的排它锁的过程中，可用于：

在目标事务的写环节中，若确定读集中的任意本地数据记录无需执行写操作，则释放其上的排它锁。

在一可选实施例中，处理器 71 还可用于：

30 在目标事务的读环节中，运行目标事务中的各条语句逻辑；

若各条语句逻辑中存在远程读逻辑，则确定目标事务为分布式事务。

在一可选实施例中，处理器 71 还可用于：

若各条语句逻辑中不存在远程读逻辑，则确定目标事务为本地事务；

35 在目标事务为本地事务的情况下，在目标事务的读环节中按照无锁的方式执行读操作，以获得目标事务的读集；

在目标事务的写环节中，为其写集涉及到的各条数据记录获取排它锁以执行写操作。

在一可选实施例中，处理器 71 在计算目标事务的写集的过程中，可用于：

在目标事务的读环节中，计算目标事务的写集；

将写集缓存在事务协调端的本地，以供写环节调用。

在一可选实施例中，处理器 71 在在目标事务的写环节中，基于排它锁完成针对写集的写操作后，释放相关数据记录上的排它锁的过程中，可用于：

在目标事务的写环节中，确定写集中属于事务协调端的本地数据记录；

- 5 基于本地数据记录上的排它锁完成写操作后，释放其排它锁；

针对写集中各条非本地数据记录向其各自所属的事务参与端发起写操作，以使事务参与端在响应写操作后释放相应数据记录上的排它锁。

在一可选实施例中，处理器 71 还可用于：

- 10 在目标事务的写环节，为目标事务分配逻辑时间戳，逻辑时间戳是基于事务之间的执行顺序而分配的；

若读集中属于事务协调端的本地数据记录的读时间戳小于目标事务的逻辑时间戳，则将本地数据记录的读时间戳更新为目标事务的逻辑时间戳；

将写集中属于事务协调端的本地数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。

- 15 在一可选实施例中，处理器 71 还可用于：

将目标事务的逻辑时间戳发送给写集涉及到的事务参与端，以供事务参与端基于目标事务的逻辑时间戳更新其在读集和/或写集中承载的数据记录的读时间戳和/或写时间戳。

在一可选实施例中，处理器 71 还可用于：

- 20 若确定目标事务为本地事务，则基于其读集中属于事务协调端的本地数据记录所拥有的读时间戳和/或写时间戳进行一致性验证。

在一可选实施例中，处理器 71 还可用于：

基于事务协调端上所处理的各事务各自对应的逻辑时间戳，确定事务协调端上未持久化事务中的最小逻辑时间戳，作为事务协调端上的分片时间戳；

- 25 获取其它分片所提供的分片时间戳；

将所有分片各自对应的分片时间戳中的最小值，作为全局时间戳；

若目标事务的逻辑时间戳小于全局时间戳，则确定目标事务已完成全局持久化处理。

在一可选实施例中，处理器 71 还可用于：

- 30 若事务协调端上存在未分配逻辑时间戳的特殊事务，则将特殊事务第一个访问的数据记录上的写时间戳作为特殊事务的逻辑时间戳下限，逻辑时间戳下限代替逻辑时间戳而作为确定分片时间戳的依据。

在一可选实施例中，处理器 71 还可用于：

- 35 在确定出事务协调端上的分片时间戳后，若事务协调端后续作为新的事务的协调端，则其为新的事务分配的逻辑时间戳不小于分片时间戳；

若事务协调端后续作为新的事务的参与端，则将其在新的事务写集中的所承载的数据记录的写时间戳更新为不小于其自身的分片时间戳，以驱使新的事务的协调端为新的事务分配不小于分片时间戳的逻辑时间戳。

在一可选实施例中，处理器 71 还可用于：

在发生分片故障的情况下，获取恢复时间戳，恢复时间戳为所有分片上确定出的全局时间戳中的最大值；

将恢复时间戳之后处理的事务进行回滚。

进一步，如图 7 所示，该事务协调端还包括：电源组件 73 等其它组件。图 7 中仅示意性给出部分组件，并不意味着事务协调端只包括图 7 所示组件。

值得说明的是，上述关于事务协调端各实施例中的技术细节，可参考前述的系统实施例中关于事务协调端的相关描述，为节省篇幅，在此不再赘述，但这不应造成本申请保护范围的损失。

图 8 为本申请又一示例性实施例提供的一种事务参与端的结构示意图。如图 8 所示，该事务参与端可包括：存储器 80、处理器 81 以及通信组件 82。

处理器 81，与存储器 80 和通信组件 82 耦合，用于执行存储器 80 中的计算机程序，以用于：

在目标事务为分布式事务的情况下，通过通信组件 82 接收目标事务的事务协调端在目标事务的读环节发起的远程读操作；

在读环节中对远程读操作请求的数据记录添加排它锁；

在目标事务的写环节中，若接收到事务协调端发起的远程写操作，则响应远程写操作并释放对应数据记录上的排它锁。

在一可选实施例中，处理器 81 还可用于：

获取事务协调端为目标事务分配的逻辑时间戳，逻辑时间戳是基于事务之间的执行顺序而分配的；

在读环节，若远程读操作对应的数据记录的读时间戳小于目标事务的逻辑时间戳，则将远程读操作对应的数据记录的读时间戳更新为目标事务的逻辑时间戳；

在写环节，将远程写操作对应的数据记录的读时间戳和写时间戳更新为目标事务的逻辑时间戳。

在一可选实施例中，处理器 81 还可用于：

若事务参与端作为其它事务的事务协调端，则在其它事务为本地事务的情况下，基于其读集中的本地数据记录的读时间戳和/或写时间戳进行一致性验证。

在一可选实施例中，处理器 81 还可用于：

基于事务参与端上所处理的各事务各自对应的逻辑时间戳，确定事务参与端上未持久化事务中的最小逻辑时间戳，作为事务参与端上的分片时间戳；

将自身的分片时间戳提供给目标事务的事务协调端，以供事务协调端将所有分片各自对应的分片时间戳中的最小值作为全局时间戳，并在目标事务的逻辑时间戳小于全局时间戳的情况下确定目标事务已完成全局持久化处理。

在一可选实施例中，处理器 81 还可用于：

在确定出分片时间戳后，若事务参与端后续作为新的事务的协调端，则为其分配的逻辑时间戳不小于分片时间戳；

若事务参与端后续作为新的事务的参与端，则将其在新的事务写集中的所承载的数据记录的些时间戳更新为不小于分片时间戳，以驱使新的事务的协调端为新的事务分配不小于分配时间戳的逻辑时间戳。

进一步，如图 8 所示，该事务参与端还包括：电源组件 83 等其它组件。图 8 中仅示意性给出部分组件，并不意味着事务参与端只包括图 8 所示组件。

值得说明的是，上述关于事务参与端各实施例中的技术细节，可参考前述的系统实施例中关于事务参与端的相关描述，为节省篇幅，在此不再赘述，但这不应造成本

5 申请保护范围的损失。

相应地，本申请实施例还提供一种存储有计算机程序的计算机可读存储介质，计算机程序被执行时能够实现上述方法实施例中可由事务协调端或事务参与端执行的各步骤。

上述图 7-8 中的存储器，用于存储计算机程序，并可被配置为存储其它各种数据以支持在计算平台上的操作。这些数据的示例包括用于在计算平台上操作的任何应用程序或方法的指令，联系人数据，电话簿数据，消息，图片，视频等。存储器可以由任何类型的易失性或非易失性存储设备或者它们的组合实现，如静态随机存取存储器（SRAM），电可擦除可编程只读存储器（EEPROM），可擦除可编程只读存储器（EPROM），可编程只读存储器（PROM），只读存储器（ROM），磁存储器，快闪存储器，磁盘或光盘。

上述图 7-8 中的通信组件，被配置为便于通信组件所在设备和其他设备之间有线或无线方式的通信。通信组件所在设备可以接入基于通信标准的无线网络，如 WiFi，2G、3G、4G/LTE、5G 等移动通信网络，或它们的组合。在一个示例性实施例中，通信组件经由广播信道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中，所述通信组件还包括近场通信（NFC）模块，以促进短程通信。例如，在 NFC 模块可基于射频识别（RFID）技术，红外数据协会（IrDA）技术，超宽带（UWB）技术，蓝牙（BT）技术和其他技术来实现。

上述图 7-8 中的电源组件，为电源组件所在设备的各种组件提供电力。电源组件可以包括电源管理系统，一个或多个电源，及其他与为电源组件所在设备生成、管理和分配电力相关联的组件。

本领域内的技术人员应明白，本申请的实施例可提供为方法、系统、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器、CD-ROM、光学存储器等）上实施的计算机程序产品的形式。

本申请是参照根据本申请实施例的方法、设备（系统）、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装

置的制造品，该指令装置实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他
5 可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能的步骤。

在一个典型的配置中，计算设备包括一个或多个处理器 (CPU)、输入/输出接口、网络接口和内存。

内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器 (RAM) 和/或非
10 易失性内存等形式，如只读存储器 (ROM) 或闪存(flash RAM)。内存是计算机可读介质的示例。

计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存 (PRAM)、静态随机存取存储器 (SRAM)、
15 动态随机存取存储器 (DRAM)、其他类型的随机存取存储器 (RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器 (CD-ROM)、数字多功能光盘 (DVD) 或其他光学存储、磁盒式磁带，磁带式磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体 (transitory media)，如调制的数据信号和载波。
20

还需要说明的是，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、商品或者设备不仅包括那些要素，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、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素
25 的过程、方法、商品或者设备中还存在另外的相同要素。

以上所述仅为本申请的实施例而已，并不用于限制本申请。对于本领域技术人员来说，本申请可以有各种更改和变化。凡在本申请的精神和原理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 书

- 1、一种事务的处理方法，适用于事务协调端，所述方法包括：
接收针对目标事务的处理请求；
若确定所述目标事务为分布式事务，则在所述目标事务的读环节中分别为其读集
5 所涉及到的各条数据记录获取排它锁；
计算所述目标事务的写集；
释放所述读集中无需执行写操作的数据记录上的排它锁；
在所述目标事务的写环节中，基于排它锁完成针对所述写集的写操作后，释放相关数据记录上的排它锁。
- 10 2、根据权利要求1所述的方法，所述在所述目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁，包括：
确定所述读集中属于所述事务协调端的本地数据记录；
为所述本地数据记录添加排它锁；
触发所述读集中各条非本地数据记录各自所属的事务参与端对相应的数据记录添
15 加排它锁。
- 3、根据权利要求2所述的方法，所述释放所述读集中无需执行写操作的数据记录上的排它锁，包括：
在所述目标事务的写环节中，若确定所述读集中的任意本地数据记录无需执行写
操作，则释放其上的排它锁。
- 20 4、根据权利要求1所述的方法，还包括：
在所述目标事务的读环节中，运行所述目标事务中的各条语句逻辑；
若各条语句逻辑中存在远程读逻辑，则确定所述目标事务为分布式事务。
- 5、根据权利要求4所述的方法，还包括：
若所述各条语句逻辑中不存在远程读逻辑，则确定所述目标事务为本地事务；
25 在所述目标事务为本地事务的情况下，在所述目标事务的读环节中按照无锁的方式执行读操作，以获得所述目标事务的读集；
在所述目标事务的写环节中，为其写集涉及到的各条数据记录获取排它锁以执行写操作。
- 6、根据权利要求1所述的方法，所述计算所述目标事务的写集，包括：
30 在所述目标事务的读环节中，计算所述目标事务的写集；
将所述写集缓存在所述事务协调端的本地，以供所述写环节调用。
- 7、根据权利要求1所述的方法，所述在所述目标事务的写环节中，基于排它锁完成针对所述写集的写操作后，释放相关数据记录上的排它锁，包括：
在所述目标事务的写环节中，确定所述写集中属于所述事务协调端的本地数据记
35 录；
基于所述本地数据记录上的排它锁完成写操作后，释放其排它锁；
针对所述写集中各条非本地数据记录向其各自所属的事务参与端发起写操作，以使事务参与端在响应写操作后释放相应数据记录上的排它锁。
- 8、根据权利要求1所述的方法，还包括：

在所述目标事务的写环节，为所述目标事务分配逻辑时间戳，所述逻辑时间戳是基于事务之间的执行顺序而分配的；

若所述读集中属于所述事务协调端的本地数据记录的读时间戳小于所述目标事务的逻辑时间戳，则将所述本地数据记录的读时间戳更新为所述目标事务的逻辑时间戳；

- 5 将所述写集中属于所述事务协调端的本地数据记录的读时间戳和写时间戳更新为所述目标事务的逻辑时间戳。

9、根据权利要求 8 所述的方法，还包括：

- 10 将所述目标事务的逻辑时间戳发送给所述写集涉及到的事务参与端，以供事务参与端基于所述目标事务的逻辑时间戳更新其在所述读集和/或所述写集中承载的数据记录的读时间戳和/或写时间戳。

10、根据权利要求 8 所述的方法，还包括：

若确定所述目标事务为本地事务，则基于其读集中属于所述事务协调端的本地数据记录所拥有的读时间戳和/或写时间戳进行一致性验证。

11、根据权利要求 1 所述的方法，还包括：

- 15 基于所述事务协调端上所处理的各事务各自对应的逻辑时间戳，确定所述事务协调端上未持久化事务中的最小逻辑时间戳，作为所述事务协调端上的分片时间戳；

获取其它分片所提供的分片时间戳；

将所有分片各自对应的分片时间戳中的最小值，作为全局时间戳；

- 20 若所述目标事务的逻辑时间戳小于所述全局时间戳，则确定所述目标事务已完成全局持久化处理。

12、根据权利要求 11 所述的方法，还包括：

若所述事务协调端上存在未分配逻辑时间戳的特殊事务，则将所述特殊事务第一个访问的数据记录上的写时间戳作为所述特殊事务的逻辑时间戳下限，所述逻辑时间戳下限代替逻辑时间戳而作为确定分片时间戳的依据。

- 25 13、根据权利要求 11 所述的方法，还包括：

在确定出所述事务协调端上的分片时间戳后，若所述事务协调端后续作为新的事务的协调端，则其为所述新的事务分配的逻辑时间戳不小于所述分片时间戳；

- 30 若所述事务协调端后续作为新的事务的参与端，则将其在所述新的事务写集中的所承载的数据记录的写时间戳更新为不小于其自身的分片时间戳，以驱使所述新的事务的协调端为所述新的事务分配不小于所述分片时间戳的逻辑时间戳。

14、根据权利要求 11 所述的方法，还包括：

在发生分片故障的情况下，获取恢复时间戳，所述恢复时间戳为所有分片上确定出的全局时间戳中的最大值；

将所述恢复时间戳之后处理的事务进行回滚。

35

15、一种事务的处理方法，适用于事务参与端，所述方法包括：

在目标事务为分布式事务的情况下，接收所述目标事务的事务协调端在所述目标事务的读环节发起的远程读操作；

在所述读环节中对所述远程读操作请求的数据记录添加排它锁；

在所述目标事务的写环节中，若接收到所述事务协调端发起的远程写操作，则响应所述远程写操作并释放对应数据记录上的排它锁。

16、根据权利要求15所述的方法，还包括：

5 获取所述事务协调端为所述目标事务分配的逻辑时间戳，所述逻辑时间戳是基于事务之间的执行顺序而分配的；

在所述读环节，若所述远程读操作对应的数据记录的读时间戳小于所述目标事务的逻辑时间戳，则将所述远程读操作对应的数据记录的读时间戳更新为所述目标事务的逻辑时间戳；

10 在所述写环节，将所述远程写操作对应的数据记录的读时间戳和写时间戳更新为所述目标事务的逻辑时间戳。

17、根据权利要求16所述的方法，还包括：

若所述事务参与端作为其它事务的事务协调端，则在所述其它事务为本地事务的情况下，基于其读集中的本地数据记录的读时间戳和/或写时间戳进行一致性验证。

18、根据权利要求15所述的方法，还包括：

15 基于所述事务参与端上所处理的各事务各自对应的逻辑时间戳，确定所述事务参与端上未持久化事务中的最小逻辑时间戳，作为所述事务参与端上的分片时间戳；

将自身的分片时间戳提供给所述目标事务的事务协调端，以供所述事务协调端将所有分片各自对应的分片时间戳中的最小值作为全局时间戳，并在所述目标事务的逻辑时间戳小于所述全局时间戳的情况下确定所述目标事务已完成全局持久化处理。

20 19、根据权利要求18所述的方法，还包括：

在确定出所述分片时间戳后，若所述事务参与端后续作为新的事务的协调端，则为其分配的逻辑时间戳不小于所述分片时间戳；

25 若所述事务参与端后续作为新的事务的参与端，则将其在所述新的事务写集中的所承载的数据记录的些时间戳更新为不小于所述分片时间戳，以驱使所述新的事务的协调端为所述新的事务分配不小于所述分配时间戳的逻辑时间戳。

20、一种事务协调端，包括存储器、处理器和通信组件；

所述存储器用于存储一条或多条计算机指令；

30 所述处理器与所述存储器和所述通信组件耦合，用于执行所述一条或多条计算机指令，以用于：

通过所述通信组件接收针对目标事务的处理请求；

若确定所述目标事务为分布式事务，则在所述目标事务的读环节中分别为其读集所涉及到的各条数据记录获取排它锁；

计算所述目标事务的写集；

35 释放所述读集中无需执行写操作的数据记录上的排它锁；

在所述目标事务的写环节中，基于排它锁完成针对所述写集的写操作后，释放相关数据记录上的排它锁。

21、一种事务参与端，包括包括存储器、处理器和通信组件；

所述存储器用于存储一条或多条计算机指令；

所述处理器与所述存储器和所述通信组件耦合，用于执行所述一条或多条计算机指令，以用于：

- 5 在目标事务为分布式事务的情况下，通过所述通信组件接收所述目标事务的事务协调端在所述目标事务的读环节发起的远程读操作；
- 在所述读环节中对所述远程读操作请求的数据记录添加排它锁；
- 在所述目标事务的写环节中，若接收到所述事务协调端发起的远程写操作，则响应所述远程写操作并释放对应数据记录上的排它锁。

- 10 22、一种存储计算机指令的计算机可读存储介质，当所述计算机指令被一个或多个处理器执行时，致使所述一个或多个处理器执行权利要求 1-19 任一项所述的事务的处理方法。