

事务处理方法、装置、设备及计算机存储介质

相关申请的交叉引用

本申请基于申请号为201911223449.3、申请日为2019年12月3日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此引入本申请作为参考。

5 技术领域

本申请涉及数据库领域，尤其涉及事务处理方法、装置、设备及计算机存储介质。

背景技术

在相关技术的数据库中，在事务A在操作数据且提交（commit）或回滚（rollback）之前，其他事务不能看到事务A对数据的修改。对于长事务而言，如果事务不提交将会导致操作相同资源的请求阻塞或处理超时。如此，子事务的提交依赖于数据库本地事务，使得整体服务的性能受限于单机数据库的处理能力，处理效率较低。

发明内容

本申请实施例提供一种事务处理方法、装置、设备及计算机存储介质，能够通过自动的对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

15 本申请实施例的技术方案是这样实现的：

第一方面，本申请实施例提供一种事务处理方法，所述方法应用于事务处理的设备，包括：将从数据库获取的待处理事务，划分为至少两个子事务；按照每一子事务所属的网络接口和所述子事务实现的各功能，将每一所述子事务划分为具有关联关系的N个部分，N为大于1的整数；按照所述关联关系对每一所述子事务的N个部分进行处理，得到
20 所述N个部分中最后被执行的各部分的处理结果；如果根据所述处理结果确定出有异常子事务，确定与所述异常子事务的异常原因相匹配的处理策略；采用所述处理策略对所述异常子事务进行处理，以得到所述待处理事务的最终处理结果。

第二方面，本申请实施例提供一种事务处理装置，包括：第一划分模块，配置为将从数据库获取的待处理事务，划分为至少两个子事务；第二划分模块，配置为按照每一
25 子事务所属的网络接口和所述子事务实现的各功能，将每一所述子事务划分为具有关联关系的N个部分，N为大于1的整数；第一处理模块，配置为按照所述关联关系对每一所述子事务的N个部分进行处理，得到所述N个部分中最后被执行的各部分的处理结果；第一确定模块，配置为如果根据所述处理结果确定出有异常子事务，确定与所述异常子事务的异常原因相匹配的处理策略；第二处理模块，配置为采用所述处理策略对所述异常子

事务进行处理，以得到所述待处理事务的最终处理结果。

第三方面，本申请实施例提供一种事务处理的设备，包括：存储器，配置为存储可执行指令；处理器，配置为执行所述存储器中存储的可执行指令时，实现上述事务处理方法。

- 5 第四方面，本申请实施例提供一种存储介质（非暂时性存储介质），存储有可执行指令，配置为引起处理器执行时，实现本申请实施例提供的事务处理方法。

本申请实施例具有以下有益效果：对于待处理事务，通过基于子事务所属的网络接口将子事务划分成多个部分，针对每一部分进行处理以解决整体服务的性能受限于单机数据库的处理能力的问题；对于处理结果异常的异常子事务，基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；如此，自动对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

10

附图说明

此处的附图被并入说明书中并构成本说明书的一部分，这些附图示出了符合本申请的实施例，并与说明书一起用于说明本申请的技术方案。

- 15 图1是本申请实施例提供的事务处理系统的一个可选的架构示意图；
图2A是本申请实施例提供的事务处理系统的另一个可选的架构示意图；
图2B是本申请实施例提供的事务处理系统的结构示意图；
图3是本申请实施例提供的事务处理方法的实现流程示意图；
图4是本申请实施例提供的事务处理方法的另一实现流程示意图；
20 图5是本申请实施例提供的事务处理系统的框架图；
图6是本申请实施例提供的长事务处理的另一框架图；
图7是本申请实施例提供的事务处理方法的实现流程示意图；
图8是本申请实施例提供的长事务流程状态的执行路径图；
图9是本申请实施例提供的事务处理方法的另一实现流程示意图；
25 图10是本申请实施例消息中间件存储事务状态的实现流程示意图；
图11是本申请实施例实现转账请求的实施流程示意图；
图12是本申请实施例提供的事务处理方法的实现流程示意图。

具体实施方式

为了使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请作进一

步地详细描述，所描述的实施例不应视为对本申请的限制，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它实施例，都属于本申请保护的范围。

在以下的描述中，涉及到“一些实施例”，其描述了所有可能实施例的子集，但是可以理解，“一些实施例”可以是所有可能实施例的相同子集或不同子集，并且可以在不冲突的情况下相互结合。

在以下的描述中，所涉及的术语“第一\第二\第三”仅仅是是区别类似的对象，不表示针对对象的特定排序，可以理解地，“第一\第二\第三”在允许的情况下可以互换特定的顺序或先后次序，以使在一些实施例中描述的本申请实施例能够以除了在一些实施例中图示或描述的以外的顺序实施。

除非另有定义，本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述本申请实施例的目的，不是旨在限制本申请。

对本申请实施例进行进一步详细说明之前，对本申请实施例中涉及的名词和术语进行说明，本申请实施例中涉及的名词和术语适用于如下的解释。

1) ACID特性: 事务具有四个特性，分别是原子性 (Atomicity)、一致性 (Consistency)、隔离性 (Isolation) 和持久性 (Durability)，简称为事务的ACID特性。

2) 长事务: 一个事务花费很长时间不能够结束，就是一个长的事务，简称长事务 (Long Transaction)。在数据库中，运行时间超过6秒的事务就被视为长事务。Informix把占用整个逻辑日志空间在一定比例以上的事务 (事务占用整个逻辑日志空间的百分比超过“长事务深水线比例”这个参数限定的值)，叫做“长事务”。

3) 发布-订阅机制 (Publish-Subscribe, PubSub): 消息传递模式介绍是一种历史悠久的经典消息传递模式；订阅发布模式定义了一种一对多的依赖关系，让多个订阅者对象同时监听某一个主题对象。这个主题对象在自身状态变化时，会通知所有订阅者对象，使它们能够自动更新自己的状态。将一个系统分割成一系列相互协作的类有一个很不好的副作用，那就是需要维护相应对象间的一致性，这样会给维护、扩展和重用都带来不便。当一个对象的改变需要同时改变其他对象，而且当不知道具体有多少对象需要改变时，就可以使用订阅发布模式。

4) 集群服务器 (Broker): 集群中的一台或多台服务器的统称，Broker没有副本机制，一旦broker宕机，该broker的消息将都不可用。Broker不保存订阅者的状态，由订阅者自己保存。

5) 记账者 (BookKeeper), 是一个提供日志条目流存储持久化的服务框架。特别适合日志流存储, 一个比较经典的应用是作为消息队列的持久框架。

6) 预写式日志 (Write-Ahead Logging): 事物日志可以帮助提高事物的效率。使用事物日志, 存储引擎在修改标的数据时只需要修改其内存拷贝, 再把修改行为记录到持久在硬盘上的事务日志中, 而不用每次都修改的数据本身持久到磁盘中。事物日志采用的是追加的方式, 因此写日志的操作是磁盘上一小块区域内的顺序I/O, 而不像随机I/O需要在磁盘的多个地方移动磁头, 所以采用事物日志的方式相对来说要快得多。事务日志持久以后, 内存中被修改的数据在后台可以慢慢地刷回磁盘。

7) 区块链 (Blockchain): 由区块 (Block) 形成的加密的、链式的交易的存储结构。

8) 区块链网络 (Blockchain Network): 通过共识的方式将新区块纳入区块链的一系列节点的集合。

9) 云技术 (Cloud technology) 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。云计算技术将变成重要支撑。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, 只能通过云计算来实现。

10) 数据库 (Database, DB), 简而言之可视为电子化的文件柜——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。

11) 数据库管理系统 (Database Management System, DBMS) 是为管理数据库而设计的电脑软件系统, 一般具有存储、截取、安全保障、备份等基础功能。数据库管理系统可以依据它所支持的数据库模型来作分类, 例如关系式、XML (Extensible Markup Language, 即可扩展标记语言); 或依据所支持的计算机类型来作分类, 例如服务器群集、移动电话; 或依据所用查询语言来作分类, 例如SQL (结构化查询语言 (Structured Query Language)、XQuery; 或依据性能冲量重点来作分类, 例如最大规模、最高运行速度; 亦或其他分类方式。不论使用哪种分类方式, 一些DBMS能够跨类别, 例如, 同时支持多种查询语言。

在相关技术中，由于子事务的提交依赖数据库本地事务，因此整体服务的性能受限于单机数据库的处理能力；长事务的状态需要和业务的数据库操作作为一个本地数据库事务提交，因此对业务数据库存在侵入性；而且，对于一些无法提供补偿事务的操作，存在一定的使用限制。

- 5 基于此，本申请实施例提供一种事务处理方法、装置、设备及计算机存储介质（非暂时性存储介质），在需要对长事务进行处理的场景下，首选，将获取的待处理事务，划分为至少两个子事务；然后，按照每一子事务所属的网络接口，将每一所述子事务划分为具有关联关系的多个部分；并且，按照所述关联关系对每一所述子事务的多个部分进行处理，得到所述多个部分中最后被执行的部份的处理结果；最后，如果根据所述处理
- 10 结果确定出有异常子事务，确定与所述异常子事务的异常原因相匹配的处理策略；采用所述处理策略对所述异常子事务进行处理，以得到所述待处理事务的最终处理结果；如此，对于待处理事务，通过基于子事务所属的网络接口将子事务划分成多个部分，以解决整体服务的性能受限于单机数据库的处理能力的问题；然后，针对每一部分进行处理，对于处理结果异常的异常子事务，基于异常原因选择匹配的处理策略对异常子事务进行
- 15 处理，以得到最终的处理结果；如此，自动对异常子事务采用匹配的处理策略进行处理，能够提高容错性以及处理事务的能力。

- 下面说明本申请实施例提供的事务处理的设备的示例性应用，本申请实施例提供的设备可以实施为笔记本电脑，平板电脑，台式计算机，机顶盒，移动设备（例如，移动电话，便携式音乐播放器，个人数字助理，专用消息设备，便携式游戏设备）等各种类型的用户设备，也可以实施为服务器。下面，将说明设备实施为设备或服务器时示例性应用。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、以及大数据和人工智能平台等基础
- 20 云计算服务的云服务器。终端可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表等，但并不局限于此。终端以及服务器可以通过有线或无线通信方式进行直接或间接地连接，本申请在此不做限制。

- 参见图1，图1是本申请实施例提供的事务处理系统的一个可选的架构示意图，为实现支撑一个示例性应用，首先，当接收到终端10发送的事务处理请求时，将该事务处理请求指定的待处理事务11，划分为多个子事务12；然后，按照每一个子事务所属的网络
- 30 接口，对该子事务再次进行划分，得到相互关联的多个部分13；接下来，按照该多个部

分13之间的关联关系对这多个部分依次进行处理，得到所述多个部分13中最后被执行的部分的处理结果101；如果该处理结果101存在异常，基于异常原因确定相匹配的处理策略102；最后，采用该处理策略对异常子事务103进行处理，以得到所述待处理事务11的最终处理结果104；如此，通过基于子事务所属的网络接口将子事务划分成多个部分，针
5 对每一部分进行处理，解决了整体服务的性能受限于单机数据库的处理能力的问题；对于处理结果异常的异常子事务，基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；这样，自动的对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

参见图2A，图2A是本申请实施例提供的事务处理系统的另一个可选的架构示意图，
10 包括区块链网络20（示例性示出了作为原生节点的服务器200）、监测系统30（示例性示出归属于监测系统30的设备300及其图形界面301），下面分别进行说明。

区块链网络20的类型是灵活多样的，例如可以为公有链、私有链或联盟链中的任意一种。以公有链为例，任何业务主体的电子设备例如用户设备和服务器，都可以在不需要授权的情况下接入区块链网络20；以联盟链为例，业务主体在获得授权后其下辖的电子
15 设备（例如设备/服务器）可以接入区块链网络20，此时，成为区块链网络20中的一类特殊的节点即客户端节点。

需要指出地，客户端节点可以只提供支持业务主体发起交易（例如，用于上链存储数据或查询链上数据）功能，对于区块链网络20的原生节点的功能，例如下文所述的排序功能、共识服务和账本功能等，客户端节点可以缺省或者有选择性（例如，取决于业务主体的具体业务需求）地实现。从而，可以将业务主体的数据和业务处理逻辑最大程度迁移到区块链网络20中，通过区块链网络20实现数据和业务处理过程的可信和可追溯。
20

区块链网络20接收来自业务主体（例如图2A中示出的监测系统30）的客户端节点（例如，图2A中示出的归属于监测系统30的设备300）提交的交易，执行交易以更新账本或者查询账本，并在设备的用户界面（例如，设备300的图形界面301）显示执行交易的各种
25 中间结果或最终结果。

下面以监测系统接入区块链网络以实现事务处理的上链为例说明区块链网络的示例性应用。

监测系统30的设备300接入区块链网络20，成为区块链网络20的客户端节点。设备300通过传感器获取待处理事务，将该待处理事务，划分为至少两个子事务；然后，按照每一子事务所属的网络接口，将子事务划分为具有关联关系的多个部分；以及，按照关联
30

关系对子事务的多个部分进行处理，得到最后被执行的部份的处理结果；最后，如果处理结果表明存在异常子事务，确定与异常原因相匹配的处理策略；采用该处理策略对异常子事务进行处理，从而得到待处理事务的最终处理结果；并且，将最终处理结果传递给区块链网络20中的服务器200或者保存在设备300中；在已对设备300部署上传逻辑或用
5 户进行操作的情况下，设备300根据待处理事务/同步时间查询请求，生成对应更新操作/查询操作的交易，在交易中指定了实现更新操作/查询操作需要调用的智能合约、以及向智能合约传递的参数，交易还携带了监测系统30签署的数字签名（例如，使用监测系统30的数字证书中的私钥，对交易的摘要进行加密得到），并将交易广播到区块链网络20。其中，数字证书可由监测系统30向认证中心31进行登记注册得到。

10 区块链网络20中的原生节点，例如服务器200在接收到交易时，对交易携带的数字签名进行验证，数字签名验证成功后，根据交易中携带的监测系统30的身份，确认监测系统30是否是具有交易权限，数字签名和权限验证中的任何一个验证判断都将导致交易失败。验证成功后签署原生节点自己的数字签名（例如，使用原生节点的私钥对交易的摘要进行加密得到），并继续在区块链网络20中广播。

15 区块链网络20中具有排序功能的节点接收到验证成功的交易后，将交易填充到新的区块中，并广播到区块链网络中20提供共识服务的节点。

区块链网络20中的提供共识服务的节点对新区块进行共识过程以达成一致，提供账本功能的节点将新区块追加到区块链的尾部，并执行新区块中的交易：对于提交待处理事务的处理结果的交易，更新状态数据库中待处理事务对应的键值对；对于查询同步时
20 间的交易，从状态数据库中查询同步时间对应的键值对，并返回查询结果。对于得到的同步时间，可显示于设备300的图形界面301中。

区块链网络20中的原生节点可从区块链中读取待处理事务，并将待处理事务呈现于原生节点的监测页面，原生节点也可以利用在区块链存储的待处理事务，对该待处理事务进行处理，比如，通过基于子事务所属的网络接口将子事务划分成多个部分，解决了
25 整体服务的性能受限于单机数据库的处理能力的问题；对于处理结果异常的异常子事务，基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；这样，自动的对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

在实际应用中，可为区块链网络20的不同原生节点设置不同的功能，例如设置服务器200具有事务处理功能和记账功能，比如，通过基于子事务所属的网络接口将子事务划
30

分成多个部分，针对每一部分进行处理，确定出处理结果异常的异常子事务；然后，基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果。对于该情况，可在交易过程中，服务器200接收设备300发送的事务处理请求，采用服务器200基于事务处理请求，对待处理事务进行划分，得到多个子事务；然后，基于子事务所属的网络接口将子事务划分成多个部分；最后，针对每一部分进行处理，对于处理结果异常的异常子事务，基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；这样，既解决了整体服务的性能受限于单机数据库的处理能力的问题，而且通过自动的对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

10 参见图2B，图2B是本申请实施例提供的事务处理系统的结构示意图，图2B所示的设备400包括：至少一个处理器410、存储器450、至少一个网络接口420和用户接口430。设备400中的各个组件通过总线系统440耦合在一起。可理解，总线系统440用于实现这些组件之间的连接通信。总线系统440除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图2B中将各种总线都标为总线系统440。

15 处理器410可以是一种集成电路芯片，具有信号的处理能力，例如通用处理器、数字信号处理器，或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等，其中，通用处理器可以是微处理器或者任何常规的处理器等。

用户接口430包括使得能够呈现媒体内容的一个或多个输出装置431，包括一个或多个扬声器和/或一个或多个视觉显示屏。用户接口430还包括一个或多个输入装置432，包括有助于用户输入的用户接口部件，在一些示例中键盘、鼠标、麦克风、触屏显示屏、摄像头、其他输入按钮和控件。

存储器450可以是可移除的，不可移除的或其组合。示例性的硬件设备包括固态存储器，硬盘驱动器，光盘驱动器等。存储器450可选地包括在物理位置上远离处理器410的一个或多个存储设备。

25 存储器450包括易失性存储器或非易失性存储器，也可包括易失性和非易失性存储器两者。非易失性存储器可以是只读存储器（Read Only Memory，ROM），易失性存储器可以是随机存取存储器（Random Access Memory，RAM）。本申请实施例描述的存储器450旨在包括任意适合类型的存储器。

30 在一些实施例中，存储器450能够存储数据以支持各种操作，这些数据的示例包括程序、模块和数据结构或者其子集或超集，其中，“模块”可以使用硬件（例如处理电路和

/或存储器)，软件（例如使用编程语言开发的软件）或其组合来实现。下面示例性说明。

操作系统451，包括配置为处理各种基本系统服务和执行硬件相关任务的系统程序，例如框架层、核心库层、驱动层等，配置为实现各种基础业务以及处理基于硬件的任务；

网络通信模块452，配置为经由一个或多个（有线或无线）网络接口420到达其他计算设备，示例性的网络接口420包括：蓝牙、无线相容性认证、和通用串行总线（Universal Serial Bus，USB）等；

呈现模块453，配置为经由一个或多个与用户接口430相关联的输出装置431（例如，显示屏、扬声器等）使得能够呈现信息（例如，用于操作外围设备和显示内容和信息的用户接口）；

输入处理模块454，配置为对一个或多个来自一个或多个输入装置432之一的一个或多个用户输入或互动进行检测以及翻译所检测的输入或互动。

在一些实施例中，本申请实施例提供的装置可以采用软件方式实现，图2B示出了存储在存储器450中的事务处理的服务器455，其可以是程序和插件等形式的软件，包括以下软件模块：第一划分模块4551、第二划分模块4552、第一处理模块4553、第一确定模块4554和第二处理模块4555；这些模块是逻辑上的，因此根据所实现的功能可以进行任意的组合或进一步拆分。将在下文中说明各个模块的功能。

在另一些实施例中，本申请实施例提供的装置可以采用硬件方式实现，作为示例，本申请实施例提供的装置可以是采用硬件译码处理器形式的处理器，其被编程以执行本申请实施例提供的事务处理方法，例如，硬件译码处理器形式的处理器可以采用一个或多个应用专用集成电路（Application Specific Integrated Circuit，ASIC）、DSP、可编程逻辑器件（Programmable Logic Device，PLD）、复杂可编程逻辑器件（Complex Programmable Logic Device，CPLD）、现场可编程门阵列（Field-Programmable Gate Array，FPGA）或其他电子元件。

将结合本申请实施例提供的设备的示例性应用和实施，说明本申请实施例提供的事务处理方法。

参见图3，图3是本申请实施例提供的事务处理方法的实现流程示意图，应用于事务处理的设备，结合图3示出的步骤进行说明。

步骤S301，将从数据库获取的待处理事务，划分为至少两个子事务。

在一些实施例中，所述待处理事务可以是长事务，即需要跨网络实现的事务，比如，腾讯计费事务。在一些实施例中，可以是按照待处理事务需要调用的资源接口的类别，

对该待处理事务进行划分，比如，资源接口的类别为3类，那么将待处理事务至少划分为3个子事务，即一个资源接口的类别对应一个子事务。

在本申请实施例中，待处理事务可以是事务处理的设备接收客户端发送的事务，还可以是事务处理设备从客户端或者服务器中读取的事务。比如，客户端向事务处理设备
5 发送游戏计费事务或结账事务等。

步骤S302，按照每一子事务所属的网络接口和子事务实现的各功能，将每一子事务划分为具有关联关系的N个部分。

在一些实施例中，N为大于1的整数。每一子事务所属的网络接口可以理解为实现该子事务需要调用的网络接口，比如，需要调用三个网络接口，将该子事务划分为具有关
10 联关系的三个部分。所述关联关系可以是这三个部分在执行的过程中的先后关系，比如，该子事务是A和B实现通话，因为在通话过程中需要调用三个网络接口，所以划分为三个部分，这三个部分可以是，第一部分是，主叫端A查询被叫端B的电话号码，为实现通话做准备工作；第二个部分是，基于查询到的号码，拨通该号码，基于网络通信建立A与B的之间的网络连接，基于该网络连接进行通话；第三部分是，对于通话的结果进行处理。

15 步骤S303，按照关联关系对每一所述子事务的N个部分进行处理，得到N个部分中最后被执行的部分的处理结果。

在一些实施例中，按照该关联关系对每一部分进行处理，最终得到最后被执行的部分的处理结果。比如，以上述A与B实现通话为例，最终得到通话的结果。将该通话的结果作为该子事务的处理结果。

20 步骤S304，如果根据处理结果确定出有异常子事务，确定与异常子事务的异常原因相匹配的处理策略。

在一些实施例中，所述异常子事务可以理解为是处理结果与预期结果不符的子事务。比如，以A与B之间实现通信为例，最后得到的通话结果如果表明通话失败，那么说明该子事务是异常子事务。所述与所述异常子事务的异常原因相匹配的处理策略，可以理解
25 为是基于该异常原因，确定是对该异常子事务进行数据回滚还是继续进行处理，从而能够更加准确的得到该异常子事务的处理结果。在一些实施例中，如果所述异常原因因为用户端在处理所述异常子事务时发生异常，将所述异常子事务的上一个子事务的处理结果作为所述异常子事务的处理结果。在一个具体例子中，用户端在处理所述异常子事务时发生异常可以是本次发生异常之后，不能再次执行或者是即使再次执行仍然会发生异常，
30 比如，用户由于选择的银行卡的余额不足导致的付款失败，那么即使再次执行该页面的

事务，仍然会付款失败，需要对事务进行回滚，以回滚到重新选择付款方式的页面，以使用户可以重新选择其他银行卡。如果所述异常原因为服务器端在处理所述异常子事务时发生异常，按照所述关联关系再次对所述异常子事务的N个部分进行处理。一个具体例子中，服务器端在处理所述异常子事务时发生异常可以是本次发生异常之后，需要再次

5 执行即可完成该子事务，比如，用户付款成功之后，发货失败，那么不能进行数据回滚，需要给用户重新发货，直至发货成功；如此，基于异常原因选择相匹配的处理策略对异常子事务进行处理，能够及时的对异常子事务进行自动处理，提高了事务的处理能力。

步骤S305，采用处理策略对异常子事务进行处理，以得到待处理事务的最终处理结果。

10 在一些实施例中，对于每一个子事务得到正确的处理结果，从而得到待处理事务的最终处理结果，这样，自动对异常子事务采用匹配的处理策略进行处理，既可以保证多个子事务进行处理的一致性，还提高了容错性以及处理事务的能力。

在本实施例中，对于需要处理的长事务，先划分为多个子事务，然后，将子事务划分成多个部分，针对每一部分进行处理，得到每一个子事务的处理结果，这样，解决了

15 整体服务的性能受限于单机数据库的处理能力的问题；基于发生异常的处理结果的异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；如此，自动对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。

在一些实施例中，为了保证对于每一个部分进行处理的高效性，所述步骤S302可以通过以下步骤实现：

20 步骤S321，确定每一子事务所属的网络接口的类别集合。

比如，以B向A转账为例，需要调用A发起转账请求的网络接口、A和B分别对应的银行的网络接口以及B接收转账请求并实现转账的网络接口等。

步骤S322，根据网络接口的类别集合和对应子事务实现的各功能，将对应子事务划分为N个部分。

25 在一些实施例中，根据网络接口的类别集合，将每一子事务至少划分为和类别集合中类别数量相同的多个部分，并且结合对应子事务实现的各功能，确定将子事务划分为部分的划分点，比如，以上述的B向A转账为例，有三个类别且实现的各功能是提交订单、实施转账操作和转账成功通知，那么将该子事务至少划分为三个部分。如果A和B对应的是同一个银行，那么可以将该子事务划分为三个部分，如果A和B对应的不是同一个银行，

30 那么可以将该子事务划分为四个部分。

步骤S323, 根据每一子事务实现的各功能, 确定N个部分之间的关联关系, 以形成具有关联关系的N个部分。

比如, 以上述的B向A转账为例, 基于这三个功能(即提交订单、实施转账操作和转账成功通知), 将子事务划分为三个部分, 并确定这三个部分之间的关联关系。

5 在一些实施例中, 为了保证对于多个子事务处理的一致性, 所述步骤S303包括以下两种情况:

情况一: 如果所述至少两个子事务之间是独立的, 按照所述关联关系对每一所述子事务的N个部分进行处理, 得到所述N个部分中最后被执行的部的处理结果。

10 在一些实施例中, 所述至少两个子事务之间是独立的, 即这些子事务之间没有交集, 比如, 一个子事务是实现通信, 一个子事务是实现银行接口的调用, 那么对于这样没有交集的子事务, 可以不考虑子事务之间的处理顺序, 对于每一子事务的多个部分, 按照这些部分之间的关联关系, 对这多个部分进行处理, 以得到最后被执行部的处理结果。

情况二: 首先, 如果所述至少两个子事务之间存在交集, 按照所述交集和预设白名单, 确定所述至少两个子事务之间的处理路径。

15 在一些实施例中, 以待处理事务为腾讯计费为例, 一个子事务是产品批价, 一个子事务是订单支付, 一个子事务是发货, 那么这三个子事务之间是存在交集的, 即, 需要先进行产品批价, 基于产品批价的结果实现订单支付, 最后, 基于订单支付的结果实现发货成功或者失败。所述预设的白名单可以认为是实现设定的事务之间的合理的处理路径, 比如, 需要是先进行订单支付再进行发货, 如果是先进行发货, 后进行订单支付, 20 则确定这一处理路径不满足预设白名单。

然后, 按照所述处理路径, 依次对每一子事务的N个部分按照所述关联关系进行处理, 以得到每一所述子事务的处理结果。

25 比如, 按照该处理路径, 首先确定需要执行的第一个子事务, 对于该第一个子事务的多个部分按照所述关联关系进行处理, 以得到第一个子事务的处理结果; 然后, 基于第一个子事务的处理结果, 对第二个子事务的多个部分按照所述关联关系进行处理, 以得到第二个子事务的处理结果; 依次类推, 对所有子事务按照该处理路径处理完毕, 得到待处理事务的最终处理结果。如此, 通过对多个子事务划分为多个部分进行处理, 保证了子事务的实时一致性。通过引入白名单机制, 保证多个子事务之间状态流转的合理性。

30 在一个具体例子中, 如果所述子事务所属的网络接口的类别集合中包含M个类别,

将所述子事务划分为至少M个部分，M为大于1的整数，所述步骤S303可以通过以下步骤实现：

第一步，根据所述关联关系，确定所述至少M个部分之间的串行执行顺序。

比如，以B向A转账为例，若划分为三个部分：创建转账订单，执行转账操作和转账结果，确定这三个部分之间的执行顺序。

第二步，按照所述串行执行顺序，基于所述至少M个部分在所述串行执行顺序中的第M-1个部分的处理结果，执行所述至少M个部分在所述串行执行顺序中的第M个部分，以得到最后被执行的部门的处理结果。

比如，先执行创建转账订单，得到创建成功的转账订单；然后，基于该转账订单，执行B向A转账；最后，判断该结果是成功还是失败，以通知用户。

在本申请实施例中，通过上述第一步至第四步实现了单个子事务的处理流程，按照多个部分之间的执行顺序对这多个部分进行逐一执行，以得到该子事务的处理结果；从而提高了事务的处理速度。

在一些实施例中，在对于多个子事务进行处理的过程中，对于发生异常的异常子事务，将待处理事务的当前事务状态存储在消息中间件中，提高了数据层的可靠性，而且保证存储的数据不会丢失，在步骤S305之前，所述方法还包括以下步骤，如图4所示，图4是本申请实施例提供的事务处理方法的另一实现流程示意图，结合图3进行以下说明：

步骤S401，采用处理策略对异常子事务进行处理的结果，确定新的处理结果。

比如，该异常子事务为发货失败，那么相匹配的处理策略为继续为已付款成功的用户进行发货，得到新的发货结果。

步骤S402，如果新的处理结果异常子事务仍处于异常状态，确定当前已处理的子事务的处理结果集合。

步骤S403，采用消息中间件，存储处理结果集合。

比如，如果新的发货结果表明仍然是发货失败，那么说明可能是当前网络出现故障，这样的情况下，为了保证事务消息不丢，先确定当前已处理的子事务的处理结果集合，即待处理事务的当前事务状态信息，然后，将该当前事务状态信息存储在消息中间件中，提高了数据层的高可靠性。

步骤S404，当接收到输入的读取指令时，从消息中间件中读取处理结果集合，并继续对异常子事务的N个部分进行处理，以得到待处理事务的最终处理结果。

在一些实施例中，所述接收到输入的读取指令可以理解为，当需要继续对待处理事

务进行处理时，比如，网络从异常状态变为正常状态，那么从该消息中间件中，利用消息订阅机制实现事务的恢复，继续对异常子事务的多个部分进行处理，以得到正常的处理结果。

5 在一些实施例中，对于不同资源相混合的待处理事务，为了保证这样的待处理事务中不同资源的子事务的一致性，所述方法以下步骤实现。

第一步，确定实现所述待处理事务需要调用的资源接口集合。

在一些实施例中，以待处理事务为腾讯计费为例，实现该事务需要调用的资源接口集合至少包括：实现产品批价的资源接口，实现订单支付的用户端资源接口，实现发货的资源接口等。

10 第二步，将待处理事务划分为与资源接口集合相匹配的多个子事务。

比如，将腾讯计费划分为与实现产品批价的资源接口，实现订单支付的用户端资源接口，实现发货的资源接口相匹配的三个子事务。

第三步，响应于接收到的处理执行指令，启动每一资源接口。

15 在一些实施例中，当接收到需要对待处理事务进行处理的处理执行指令时，进入分布式事务（外部XA）能力事务第一阶段，即启动每一个资源接口，以促使每一个资源接口做好处理子事务的准备工作。

第四步，当接收到资源接口的反馈的准备完成信息时，在资源接口中对相匹配的子事务的N个部分按照关联关系进行处理，以得到N个部分中最后被执行的部份的处理结果。

20 在一些实施例中，当资源接口准备完成之后，向处理器反馈准备完成信息，在资源接口中对相匹配的子事务进行处理，得到处理结果，然后进入分布式事务（外部XA）能力的第二阶段，即对于处理结果进行提交，并向用户反馈处理完成的通知信息；这样，通过采用多个资源接口对多个子事务进行处理，保证了不同子事务的一致性。

25 下面，将说明本申请实施例在一个实际的应用场景中的示例性应用，以对计费场景下的长事务进行处理为例，进行说明。

30 在一些实施例中，一般事务可以通过数据库的本地事务来保证。比如，在MySQL默认可重复读（Repeatable read, RR）的隔离级别下，当事务A在操作数据且提交或回滚前，其他事务不能看到事务A对数据的修改。对于短事务而言，数据库锁的时间比较短，而对于长事务，如果事务不提交将会导致操作相同资源的请求阻塞或处理超时。因此，长事务一般不适合直接使用数据库事务实现。在一些实施例中，将一个长事务分解成一串子

事务序列并依次执行，若其中某子事务执行失败，则对之前执行成功的子事务执行相应的补偿事务。其中，每个子事务都是作为本地数据库事务提交，并通过另一个后台服务（Daemon）保证整个长事务的原子性。如此，在保证长事务一致性的基础上，解决了传统基于数据库的长事务性能低的问题，从而能够改善整体服务的性能。处理过程如图5所示，图5是本申请实施例提供的事务处理系统的框架图，结合图5进行以下说明：

首先，启动处理流程51，将长事务分为多个子事务501、502、50j和50n，对应于 T_1 至 T_n ，其中， C_{151} 和 C_{252} 为子事务501至50n对应的补偿事务。通过对多个子事务501至50n进行处理，若均执行成功，则结束流程52；对于执行失败的子事务，采用补偿事务进行补偿，当执行成功时，结束流程53。

由此可见，实现图5所示的方案需要具备以下两个条件：

条件1：需要DBMS的事务特性支持。

条件2：能够将一个长事务可以分解为多个子事务。

执行过程如下：

（1）假设将一个长事务 T 分解为多个子事务 $\{T_1, T_2, \dots, T_N\}$ ，每个子事务都是相对独立的，且可以保证原子性。

（2）为了保证整个长事务 T 的原子性，若出现部分执行成功的情况，每个子事务需要提供对应的补偿事务 $\{C_1, C_2, \dots, C_N\}$ 。因此，长事务执行的序列如下：（其中， $0 \leq j \leq N$ ）

正常执行序列： $T_1, T_2, \dots, T_j, \dots, T_N$ ；

异常执行序列： $T_1, T_2, \dots, T_j, \dots, C_j, \dots, C_2, C_1$ ；

（3）每个子事务的执行是相互独立的，在长事务执行过程中，不具备事务隔离性，事务的中间状态可以被外部读取。

（4）应用程序通过开启事务（begin-saga）命令字开启一个长事务并生成一个唯一事务标识（ID），通过开始处理（begin-transaction）和结束处理（end-transaction）命令字定义每个子事务的边界，最后通过结束事务（end-saga）命令字完成整个事务的提交。

（5）在执行每个子事务时，会同时将当前子事务的状态信息作为一个数据库本地事务提交保存。与数据库事务异常（比如，宕机（Crash））处理机制不同，数据库事务在出现异常时通过回滚日志（undo log）可以对事务进行自动回滚。而处理异常的机制是，需要应用程序通过信息中继（Message Relay）服务saga后台服务（saga daemon）继续完

成后续子事务 (forward recovery), 或者执行补偿事务 (backward recovery), 以保证长事务的原子性。如图6所示, 图6是本申请实施例提供的长事务处理的另一框架图, 结合图6进行以下说明:

第一步, 将应用程序 (Application, APP) 61输出的待处理事务, 划分为多个子事务
5 T₁62、T₂63至T_N64。

在一些实施例中, APP 61向系统服务69输出待处理事务。该待处理事务为长事务。该多个子事务T₁62、T₂63至T_N64之间可以是独立的, 也可以是存在串联关系的。

第二步, 对于每个业务操作同时记录下来, 以保证两个操作的一致性。

在一些实施例中, 在执行每一个事务时, 因为是基于数据库进行的, 所以要保证每
10 一个操作和基于的事务是一致性的; 其中, APP表65表示业务的逻辑, 处理事件表66配置为表明事务的状态信息, 整体思路为, 需要完成业务操作的时候, 同时会把整个事务的操作同时都记录下来, 这样, 以保证两个表是要么全部执行要么全部失败, 即是表明两个操作是同时生效的。

第三步, 基于表66, 确定执行完子事务T₁62之后下一个待执行的子事务, 采用信息
15 中继67 (message delay) 驱动执行该待执行的子事务。

在一些实施例中, 信息中继67用于一个子事务执行完之后, 驱动去执行下一个子事务。

第四步, 将该待执行的子事务进行发布到系统中, 以使继续执行该子事务。在一些实施例中, 数据库68表示数据库存储。

20 由此可见, 由于子事务的提交依赖数据库本地事务, 整体服务的性能受限于单机数据库的处理能力; 而且长事务的状态需要和业务的数据库操作作为一个本地数据库事务提交, 对业务数据库存在侵入性; 对于一些无法提供补偿事务的操作, 存在一定的使用限制。

在本申请实施例中, 基于腾讯计费的应用场景, 进行以下说明, 首先, 腾讯计费是
25 孵化于支撑腾讯内部业务千亿级营收的互联网计费平台, 在如此庞大的业务体量下, 腾讯计费要支撑业务的快速增长, 同时还要保证每笔交易不错账。然而腾讯计费的一笔交易流程通常涉及几十次的网络调用, 在分布式场景下要满足ACID, 保证所有操作均执行成功才提交结果, 随着分布式规模的扩大要达成一致性的时间周期也就越长。比如, 比特币网络的可扩展性 (Scalability) 问题, 为了满足一致性一笔交易的平均确认时间需要
30 10分钟。因此, 为了适应复杂的业务场景出现了Base理论, 使用最终一致性来代替强一

致性。然而，对于腾讯计费日均过亿的交易量，采用最终一致性或离线补偿性的方案往往会带来较多的处理风险及投诉，需要做到实时的一致性。因此，本申请实施例提供一种事务处理方法，将复杂的分布式一致性问题交给引擎平台处理，主要实现分布式事务的原子性，保证交易实时高一致；自动的异常处理，提高容错性；以及基于状态机的流程管理，加强流程的可约束性。其中，长事务在腾讯计费是指，一次完整的交易流程，通常包括，用户登录态校验，风控检查，原始物品批价，营销活动批价，用户支付，物品发货，活动赠送等流程，需要保证整体流程的一致性，做到全部成功或者全部失败。通过引入有限状态机（Finite State Machine，FSM）和可靠的消息中间件实现基于应用层的长事务。

图7是本申请实施例提供的事务处理方法的实现流程图，结合图7进行以下说明：

首先，发起方接收APP端700提交的待处理事务，并将该待处理事务划分为多个子事务 T_1 701， T_2 702和 T_N 703。

在一些实施例中，发起方基于算子条件1确定执行子事务 T_1 701之后，执行 T_2 702；基于算子条件3，确定执行子事务 T_1 701之后，执行 T_N 703；即算子条件2确定执行子事务 T_2 702之后，执行 T_N 703。资源方可以理解为数据库或者网络接口的调用。箭头71和72分别表示子事务执行失败的话，进行一个逆向操作。

然后，向资源方73发送启动信息，以使资源方对一个子事务 T_1 701开始处理。

在一些实施例中，把一个子事务分为三个独立部分（即准备远程过程调用协议（prepare-Remote Procedure Call Protocol, pre-rpc）704，rpc 705和提交rpc（post-rpc）706），若其中某个子事务在处理过程中发生异常，事务恢复可以由业务根据rpc应答的结果信息，通过多种算子条件（Conditions）组合指定采取forward recovery还是backward recovery。

再次，按照每一个部分之间的关联关系，执行每一个部分，并将最后被执行的部分的处理结果反馈给资源方74。

最后，向资源方75反馈子事务 T_1 701执行完成，并最终反馈给APP端700。

在本申请实施例中，通过引入FSM白名单机制，保证交易状态流转的合理性。例如，不会出现已经支付的订单，被再次提交支付。图8是本申请实施例提供的长事务流程状态的执行路径图，如图8所述，节点中的数字表示某种执行状态，其中，比如，01表示初始化，50表示批价成功，100表示下单成功，那么批价成功之后，就要进行下单，100就表示提交了一笔支付，表示调用微信支付这个窗口调用成功了，比如，给微信弹出一个弹窗，让用户输入密码，用于可以输入也可以不输入，如果输入就支付成功进入101支付成

功，然后就到610表示发货成功，901表示发货失败；205表示没有支付未成功；601表示用户发起退款，602表示给用户退款成功；602到910的箭头表示，退款成功之后，会同时更新支付单的状态，然后标记该支付单已经发起退款。实线路径表示可以正常执行的路径，而虚线路径801表示受限的执行路径（即，节点50不能直接流转至节点901，即批价到发货失败是不合理的状态流转，中间需要经过支付完成状态节点101）。如此，长事务内部执行会分为多步操作，并且有些步骤是有先后顺序的，因此需要做到对状态流程可约束，通过有限状态机白名单的方式可以控制事务流程合理的流转执行。

在本申请实施例中，通过采用高可靠消息中间件（pub-sub），将事务的状态存储在消息中间件中，并通过消息订阅机制实现事务的恢复。如此，解决了业务侵入性的问题，可以应用在更多的场景，同时提供了灵活的订阅能力，使得并发处理能力更高，延迟也在毫秒级，比较适合在线长事务的处理场景。如图9所示，图9是本申请实施例提供的事务处理方法的另一实现流程示意图，结合图9进行以下说明：

首先，APP端91向系统服务92提交待处理事务。

然后，系统服务92对待处理事务中的子事务 T_1 93进行处理，基于条件算子1，确定执行完 T_1 93之后，执行子事务 T_2 94，以此类推，顺序执行每一个子事务，直至执行最后一个子事务 T_N 95，以对该待处理事务处理完成，得到最终的处理结果。

在一些实施例中，在处理这些子事务的过程中，可以将处理子事务 T_1 93的事务状态保存在事务状态存储区96中，并且从事务状态存储区96还可以向系统服务反馈 T_1 93的事务状态。同理，将处理子事务 T_2 94的事务状态保存在事务状态存储区97中，并且从事务状态存储区97还可以向系统服务反馈 T_2 94的事务状态；将处理子事务 T_N 95的事务状态保存在事务状态存储区98中，并且从事件存储区98还可以向系统服务反馈 T_N 95的事务状态。

图10是本申请实施例消息中间件存储事务状态的实现流程示意图，如图10所示，APP11、APP 12和APPN 13分别用于提交待处理事务。在一个具体例子中，如果APP12提交的待处理事务在处理过程中出现异常，那么可以将APP12提交的待处理事务的事务状态存储在消息中间件1004中，在该消息中间件1004中，包括生产端1005和消费端1006，其中，生产端1005用于将存储的事务状态进行入队，消费端1006用于将存储的事务状态进行出队。接下来，对于消息中间件中的要存储的事务状态，因为强调的是高可靠的消息件，要保证消息存储下来不会丢失，所以基于作为底层存储的控制分发器1007（dispatcher），在接收请求之后，控制分发器1007根据请求，确定将事务状态存储在哪里，在该控制分发器1007底层存储中包括计算引擎1008（broker），和存储引擎1009

(bookkeeper, BK), 以及分布式协调件1010 (zookeeper)。在计算引擎1008中通过负载均衡1011 (load balance)、管理分类1012 (managed ledger)、存储器1013 (cache) 和BK客户端1014, 来确定请求存储的事务状态如何存储。总体来说, 控制分发器1007实现的是计算和存储的分离, 即计算这个事务状态怎么存储, 存储在哪里, 存储是指存储引擎

5 1009保证事务状态的高可靠的存储。

在本实施例中, 高可靠的消息中间件。为了保证事务消息不丢, 消息中间件在数据层是基于多副本的方式, 保证数据层的高可靠; 另外, 在生产端通过WAL顺序写日志的方式, 提高了系统的处理能力, 可以达到毫秒级的延迟; 在架构上通过计算和存储分离的方式, 可以灵活的水平扩展; 最后, 为了便于管理我们实现了一套易于操作的管理台,

10 可以实现对任意时刻执行的事务进行检查。

在一些实施例中, 在腾讯计费的实际业务场景中, 一次交易处理过程处理调用RPC接口还可能会直接操作数据库或其他资源接口。以用户转账为例: 一次转账操作流程, 涉及了图11中的三个步骤。图11是本申请实施例实现转账请求的实施流程示意图, 结合图11进行以下说明:

15 第一步, 对于APP端1101提交的转账请求, 计入订单服务1102, 以创建转账订单。

第二步, 订单创建完成之后, 进入DB服务1103, 以执行转账DB操作, 同时对用户的DB数据进行修改。

在一些实施例中, 执行转账操作的过程中对于需要调用两个不同网络接口的情况, 需要通过两个组1104和1105来实现。在组1104和1105中M和S分别表示: 数据库一套设置

20 中的主和备。在一个具体例子中, 用户A给用户B转账, 但是两个用户的开户行不同, 一个建设银行一个工商银行, 这两个银行的数据存储是不在一个服务器上的, 这种情况下, 通过采用两个组1104和1105, 实现不同银行之间的转账。

第三步, 转账完成后, 进入通知服务1106, 以生成转账成功信息, 进行转账成功通知, 并将转账成功信息返回给APP端1101。

25 在一些实施例中, 虚线1107和虚线1108分别表示当前步骤失败时, 进行数据回滚。

在本实施例中, 消息中间件的读写性能高于数据库的处理性能, 并通过本地存储和远程高一致消息中间件相结合的方式保证事务消息不丢失。长事务的状态信息通过消息中间件进行持久化存储, 不需要与业务的存储耦合, 通过pub-sub的方式驱动整个事务的执行, 保证了数据库的无侵入性。

30 在一些实施例中, 由于数据库的事务是基于连接的, 连接断开事务就会回滚, 所以

假设这样几个场景：

场景1：若DB事务先提交成功，其他事务失败了。

场景2：若其他事务先提交成功，此时DB连接断了。

场景3：若其他事务先提交成功，此时DB部分成功，部分失败。

5 场景4：若存在某个异步服务，此时DB操作提交还是不提交。

由此可见，当长事务中涉及DB操作时，很难实现或提供补偿事务操作，例如，一些复杂的SQL操作无法实现对应的反操作。从而无法实现长事务的一致性处理。当上面四种场景出现时，都会发生子事务部分提交部分失败的情况。因此，为了解决上面的问题，本申请实施例结合MySQL外部XA能力，实现了原生的分布式事务。由于MySQL提供了
10 外部XA事务的能力，允许准备成功的事务跨连接。通过两个阶段提交的方式来实现跨数据库的分布式事务。由于对业务SQL不存在限制，使用场景更广。下面以转账为例来说明长事务中涉及混合资源操作时是保证不同子事务一致性的过程，如图12所示，图12是本申请实施例提供的事务处理方法的实现流程示意图，结合图12进行以下说明：

步骤S1201，开启事务。

15 在一些实施例中，APP端向中间件服务引擎（TDXA）发送开启事务的指令；APP端发起转账请求给TDXA，TDXA用于实现一致性的中间件服务引擎。

步骤S1202，创建转账订单。

在一些实施例中，Try表示创建订单，创建订单服务（order service）之后，发起sql转账操作，进入转账操作的第一阶段。

20 步骤S1203，向数据库1发送sql指令“xa start xd1”。

在一些实施例中，sql1，对子事务T1进行更新，可以通过sql语句“@sql1 = update T1 set balance = balance + 10 where user='g1'”实现。sql2，对子事务T1再次进行更新，可以通过sql语句“@sql2 = update T1 set balance = balance -10 where user='g2' and balance >= 10”实现。

25 步骤S1204，向数据库2发送sql指令“xa start xd2”。

在一些实施例中，通过分别向数据库1和数据库2发送sql指令“xa start xd1”和“xa start xd2”（其中，这两个sql指令作为协调者，用于保证双方数据库达成一致，以使数据库1和数据库2都准备就绪），保持数据库1和数据库2处理事务的一致性。

步骤S1205，向数据库1发送sql指令“xa prepare xd1”。

30 步骤S1206，向数据库2发送sql指令“xa prepare xd2”。

在一些实施例中，通过分别向数据库1和数据库2发送sql指令“xa prepare xd1”和“xa prepare xd2”，使得两个DB均准备就绪，以对事务的处理，即对转账订单进行处理。

步骤S1207，确定转账结果，并向用户发送通知消息。

5 在一些实施例中，可以采用“do”获取转账结果，如果 order service在创建转账订单的过程中失败了，对该创建的订单重新设置状态，比如，创建订单完成时，设定的状态为1，如果失败了就修改该订单的状态；do的意思是通知用户转账失败或者成功。在本实施例中，如果订单创建成功，但是转账结果失败，就要把创建的订单的状态改成订单失效。完成步骤S1207之后，进入多个事务之间状态的第二阶段。

在其他实施例中，数据库事务处理过程中，在需要回滚的场景下，通过记录sql的undo
10 log方式进行逆向操作（例如， $a=a+1$ 的逆向操作是 $a=a-1$ ）。除了使用数据库作为存储介质，也可以采用本地文件作为存储，比如，按行存储每一个子事务的状态信息，然后通过一个异步发货检查文件中的事务状态信息以决定后续的操作。

步骤S1208，对前述步骤进行确认。

15 在一些实施例中，多个事务之间状态的第二阶段，把前面步骤S1201至步骤S1207中已经协商好的协议进行确认。

步骤S1209，向数据库1发送“xa commit xid1”，和向数据库2发送“xa commit xid2”。

在一些实施例中，通过分别向数据库1和数据库2发送“xa commit xid1”和“xa commit xid2”，确认数据库1和数据库2对最后的处理结果（比如，转账成功）进行提交，已反馈给APP端。

20 步骤S1210，向APP端返回事务结束消息。

在本申请实施例中，支持多种不同资源的事务类型，应用场景更广，一个长事务流程可能涉及RPC数据库，KV等多种资源操作，需要做到混合资源操作的一致性，实现了通用性。

下面继续说明本申请实施例提供的事务处理的服务器455的实施为软件模块的示例
25 性结构，在一些实施例中，如图2B所示，存储在存储器450的事务处理的服务器455中的软件模块可以包括：第一划分模块4551，配置为将从数据库获取的待处理事务，划分为至少两个子事务；第二划分模块4552，配置为按照每一子事务所属的网络接口和所述子事务实现的各功能，将每一所述子事务划分为具有关联关系的N个部分，N为大于1的整数；第一处理模块4553，配置为按照所述关联关系对每一所述子事务的N个部分进行处理，
30 得到所述N个部分中最后被执行的的部分的处理结果；第一确定模块4554，配置为如果根据

所述处理结果确定出有异常子事务，确定与所述异常子事务的异常原因相匹配的处理策略；第二处理模块4555，配置为采用所述处理策略对所述异常子事务进行处理，以得到所述待处理事务的最终处理结果。在一些实施例中，所述第一划分模块4551，还配置为：确定每一所述子事务所属的网络接口的类别集合；根据所述网络接口的类别集合和对应子事务实现的各功能，将所述对应子事务划分为N个部分；根据每一所述子事务实现的各功能，确定所述N个部分之间的关联关系，以形成具有关联关系的N个部分。在一些实施例中，所述第一处理模块4553，还配置为：如果所述至少两个子事务之间是独立的，按照所述关联关系对每一所述子事务的N个部分进行处理，得到所述N个部分中最后被执行的各部分的处理结果。在一些实施例中，所述第一处理模块4553，还配置为：如果所述至少两个子事务之间存在交集，按照所述交集和预设白名单，确定所述至少两个子事务之间的处理路径；按照所述处理路径，依次对每一子事务的N个部分按照所述关联关系进行处理，以得到每一所述子事务的处理结果。在一些实施例中，如果所述子事务所属的网络接口的类别集合中包含M个类别，将所述子事务划分为至少M个部分，M为大于1的整数，所述第一处理模块4553，还配置为：根据所述关联关系，确定所述至少M个部分之间的串行执行顺序；按照所述串行执行顺序，基于所述至少M个部分在所述串行执行顺序中的第M-1个部分的处理结果，执行所述至少M个部分在所述串行执行顺序中的第M个部分，以得到最后被执行的各部分的处理结果。在一些实施例中，所述第一处理模块4553，还配置为：采用所述处理策略对所述异常子事务进行处理的结果，确定新的处理结果；如果所述新的处理结果所述异常子事务仍处于异常状态，确定当前已处理的子事务的处理结果集合；采用消息中间件，存储所述处理结果集合；当接收到输入的读取指令时，从所述消息中间件中读取所述处理结果集合，并继续对异常子事务的N个部分进行处理。在一些实施例中，所述第二处理模块4555，还配置为：如果所述异常原因为用户端在处理所述异常子事务时发生异常，将所述异常子事务的上一个子事务的处理结果作为所述异常子事务的处理结果；如果所述异常原因为服务器端在处理所述异常子事务时发生异常，按照所述关联关系再次对所述异常子事务的N个部分进行处理。在一些实施例中，所述第一划分模块4551，还配置为：确定实现所述待处理事务需要调用的资源接口集合；将所述待处理事务划分为与资源接口集合相匹配的多个子事务；对应地，响应于接收到的处理执行指令，启动每一资源接口；当接收到资源接口的反馈的准备完成信息时，在所述资源接口中对相匹配的子事务的N个部分按照所述关联关系进行处理，以得到所述N个部分中最后被执行的各部分的处理结果。

本申请实施例提供一种存储有可执行指令的存储介质，其中存储有可执行指令，当可执行指令被处理器执行时，将引起处理器执行本申请实施例提供的方法。在一些实施例中，存储介质可以是闪存、磁表面存储器、光盘、或光盘存储器等存储器；也可以是包括上述存储器之一或任意组合的各种设备。在一些实施例中，可执行指令可以采用程序、软件、软件模块、脚本或代码的形式，按任意形式的编程语言（包括编译或解释语言，或者声明性或过程性语言）来编写，并且其可按任意形式部署，包括被部署为独立的程序或者被部署为模块、组件、子例程或者适合在计算环境中使用的其它单元。作为示例，可执行指令可以但不一定对应于文件系统中的文件，可以可被存储在保存其它程序或数据的文件的一部分，例如，存储在超文本标记语言（Hyper Text Markup Language, HTML）文档中的一个或多个脚本中，存储在专用于所讨论的程序的单个文件中，或者，存储在多个协同文件（例如，存储一个或多个模块、子程序或代码部分的文件）中。作为示例，可执行指令可被部署为在一个车载计算设备上执行，或者在位于一个地点的多个计算设备上执行，又或者，在分布在多个地点且通过通信网络互连的多个计算设备上执行。综上所述，本申请实施例在对长事务进行处理的过程中，对于待处理事务，通过基于子事务所属的网络接口将子事务划分成多个部分，针对每一部分进行处理，对于处理结果异常的异常子事务，解决了整体服务的性能受限于单机数据库的处理能力的问题；基于异常原因选择匹配的处理策略对异常子事务进行处理，以得到最终的处理结果；如此，自动对异常子事务采用匹配的处理策略进行处理，提高了容错性以及处理事务的能力。以上所述，仅为本申请的实施例而已，并非用于限定本申请的保护范围。凡在本申请的精神和范围之内所作的任何修改、等同替换和改进等，均包含在本申请的保护范围之内。

工业实用性

本申请实施例中，将从数据库获取的待处理事务，划分为至少两个子事务；按照每一子事务所属的网络接口和所述子事务实现的各功能，将每一所述子事务划分为具有关联关系的N个部分；按照所述关联关系对每一所述子事务的N个部分进行处理，得到所述N个部分中最后被执行的部份的处理结果；如果根据所述处理结果确定出有异常子事务，确定与所述异常子事务的异常原因相匹配的处理策略；采用所述处理策略对所述异常子事务进行处理，以得到所述待处理事务的最终处理结果；这样，通过自动化部署的方式，采用匹配的处理策略处理异常子事务，提高了容错性以及处理事务的能力。