Atitit 分布式事务

Tcc

### 两阶段提交/XA

### TCC

关于 TCC（Try-Confirm-Cancel）的概念，最早是由 Pat Helland 于 2007 年发表的一篇名为《Life beyond Distributed Transactions:an Apostate’s Opinion》的论文提出。 TCC 事务机制相比于上面介绍的 XA，解决了其几个缺点：

1. 解决了协调者单点，由主业务方发起并完成这个业务活动。业务活动管理器也变成多点，引入集群。
2. 同步阻塞：引入超时，超时后进行补偿，并且不会锁定整个资源，将资源转换为业务逻辑形式，粒度变小。
3. 数据一致性，有了补偿机制之后，由业务活动管理器控制一致性

TCC(Try Confirm Cancel)  
Try 阶段：尝试执行，完成所有业务检查（一致性）, 预留必须业务资源（准隔离性）  
Confirm 阶段：确认执行真正执行业务，不作任何业务检查，只使用 Try 阶段预留的业务资源，Confirm 操作满足幂等性。要求具备幂等设计，Confirm 失败后需要进行重试。  
Cancel 阶段：取消执行，释放 Try 阶段预留的业务资源 Cancel 操作满足幂等性 Cancel 阶段的异常和 Confirm 阶段异常处理方案基本上一致。

在 Try 阶段，是对业务系统进行检查及资源预览，比如订单和存储操作，需要检查库存剩余数量是否够用，并进行预留，预留操作的话就是新建一个可用库存数量字段，Try 阶段操作是对这个可用库存数量进行操作。  
基于 TCC 实现分布式事务，会将原来只需要一个接口就可以实现的逻辑拆分为 Try、Confirm、Cancel 三个接口，所以代码实现复杂度相对较高。

## 可靠消息最终一致性

### 尽最大努力通知

最大努力通知是最简单的一种柔性事务，适用于一些最终一致性时间敏感度低的业务，且被动方处理结果 不影响主动方的处理结果。

这个方案的大致意思就是：

1. 系统 A 本地事务执行完之后，发送个消息到 MQ；
2. 这里会有个专门消费 MQ 的服务，这个服务会消费 MQ 并调用系统 B 的接口；
3. 要是系统 B 执行成功就 ok 了；要是系统 B 执行失败了，那么最大努力通知服务就定时尝试重新调用系统 B, 反复 N 次，最后还是不行就放弃。

## 分布式事务实战

### 两阶段提交/XA

目前支付宝使用两阶段提交思想实现了分布式事务服务 (Distributed Transaction Service, DTS) ，它是一个分布式事务框架，用来保障在大规模分布式环境下事务的最终一致性。具体可参考支付宝官方文档：[https://tech.antfin.com/docs/2/46887](https://tech.antfin.com/docs/2/46887" \t "https://xiaomi-info.github.io/2020/01/02/distributed-transaction/_blank)

### TCC

TCC 需要事务接口提供 try, confirm, cancel 三个接口，提高了编程的复杂性。依赖于业务方来配合提供这样的接口，推行难度大，所以一般不推荐使用这种方式。

### 可靠消息最终一致性

目前市面上支持该方案的 mq 只有阿里的 rocketmq, 该方案应用场景也比较多，比如用户注册成功后发送邮件、电商系统给用户发送优惠券等需要保证最终一致性的场景

### 本地消息表

跨行转账可通过该方案实现。  
用户 A 向用户 B 发起转账，首先系统会扣掉用户 A 账户中的金额，将该转账消息写入消息表中，如果事务执行失败则转账失败，如果转账成功，系统中会有定时轮询

### 三阶段提交协议 3PC

与两阶段提交不同的是，三阶段提交有两个改动点。

* 引入超时机制。同时在协调者和参与者中都引入超时机制。
* 在第一阶段和第二阶段中插入一个准备阶段。保证了在最后提交阶段之前各参与节点的状态是一致的。

也就是说，除了引入超时机制之外，3PC把2PC的准备阶段再次一分为二，这样三阶段提交就有CanCommit、PreCommit、DoCommit三个阶段。

## 分布式事务的解决方案

分布式事务的解决方案有如下几种：

* 全局消息
* 基于可靠消息服务的分布式事务
* TCC
* 最大努力通知

### 方案1：全局事务（DTP模型）

全局事务基于DTP模型实现。DTP是由X/Open组织提出的一种分布式事务模型——X/Open Distributed Transaction Processing Reference Model。它规定了要实现分布式事务，需要三种角色：

### 方案3：最大努力通知（定期校对）

最大努力通知也被称为定期校对，其实在方案二中已经包含，这里再单独介绍，主要是为了知识体系的完整性。这种方案也需要消息中间件的参与，其过程如下：

### 方案4：TCC（两阶段型、补偿型）

TCC即为Try Confirm Cancel，它属于补偿型分布式事务。顾名思义，TCC实现分布式事务一共有三个步骤：

* Try：尝试待执行的业务
  + 这个过程并未执行业务，只是完成所有业务的一致性检查，并预留好执行所需的全部资源
* Confirm：执行业务
  + 这个过程真正开始执行业务，由于Try阶段已经完成了一致性检查，因此本过程直接执行，而不做任何检查。并且在执行的过程中，会使用到Try阶段预留的业务资源。
* Cancel：取消执行的业务
  + 若业务执行失败，则进入Cancel阶段，它会释放所有占用的业务资源，并回滚Confirm阶段执行的操作。

作者：大闲人柴毛毛  
链接：https://juejin.cn/post/6844903573667446797  
来源：掘金  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。