分布式事务

前言

- 底层原理或者是算法？No.
- 神秘的行业“技术壁垒”
- 不在于讲透，在于引起深入探讨

目录

1.引入场景

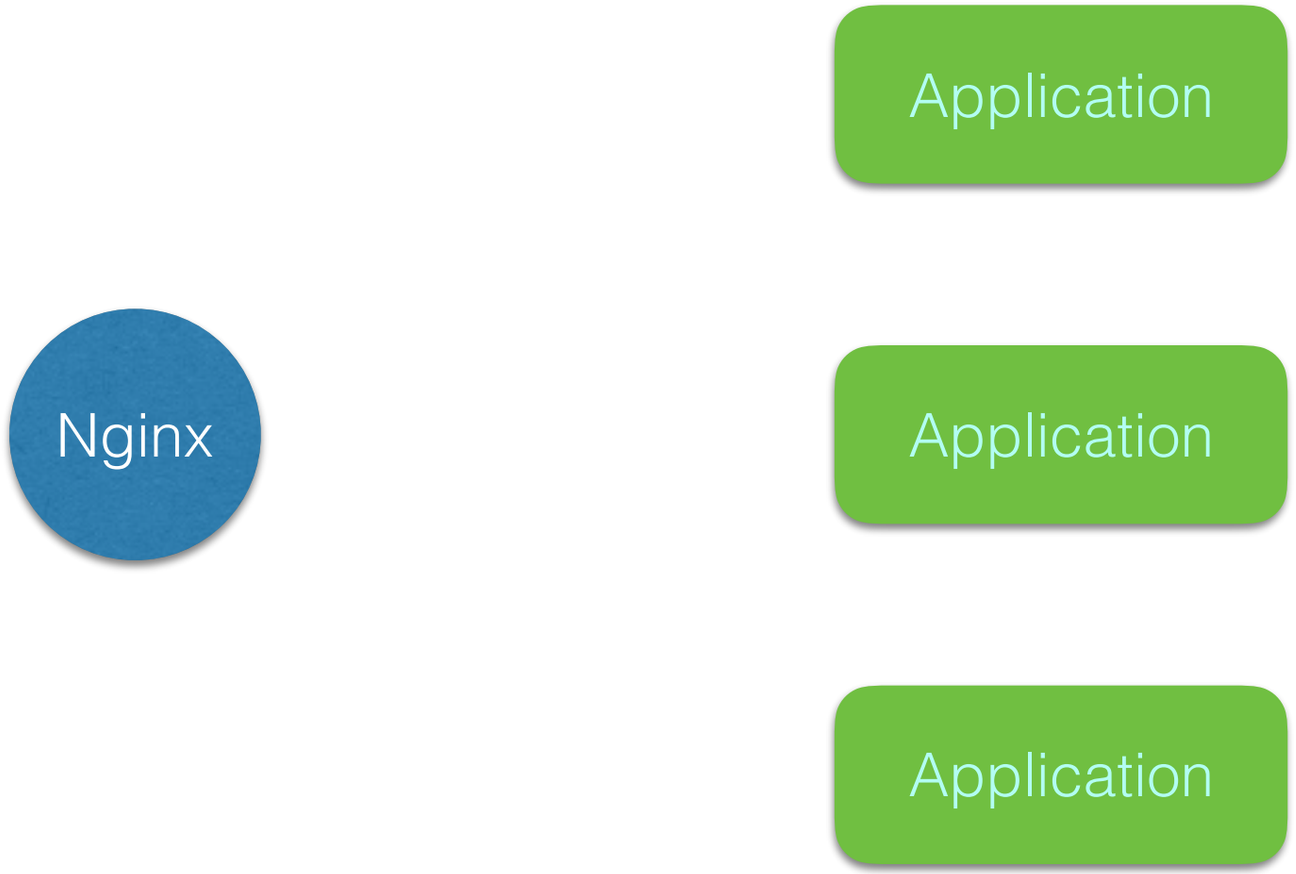
- 电商购物系统
- 钱账户(account)与积分账户(point)混合消费
- 消费后记录日志系统（record）

2.集群应用实践

定义：

- 标准MVC结构
- 编译完后只有一个war包
- 部署在多个servlet容器中

2.集群应用实践



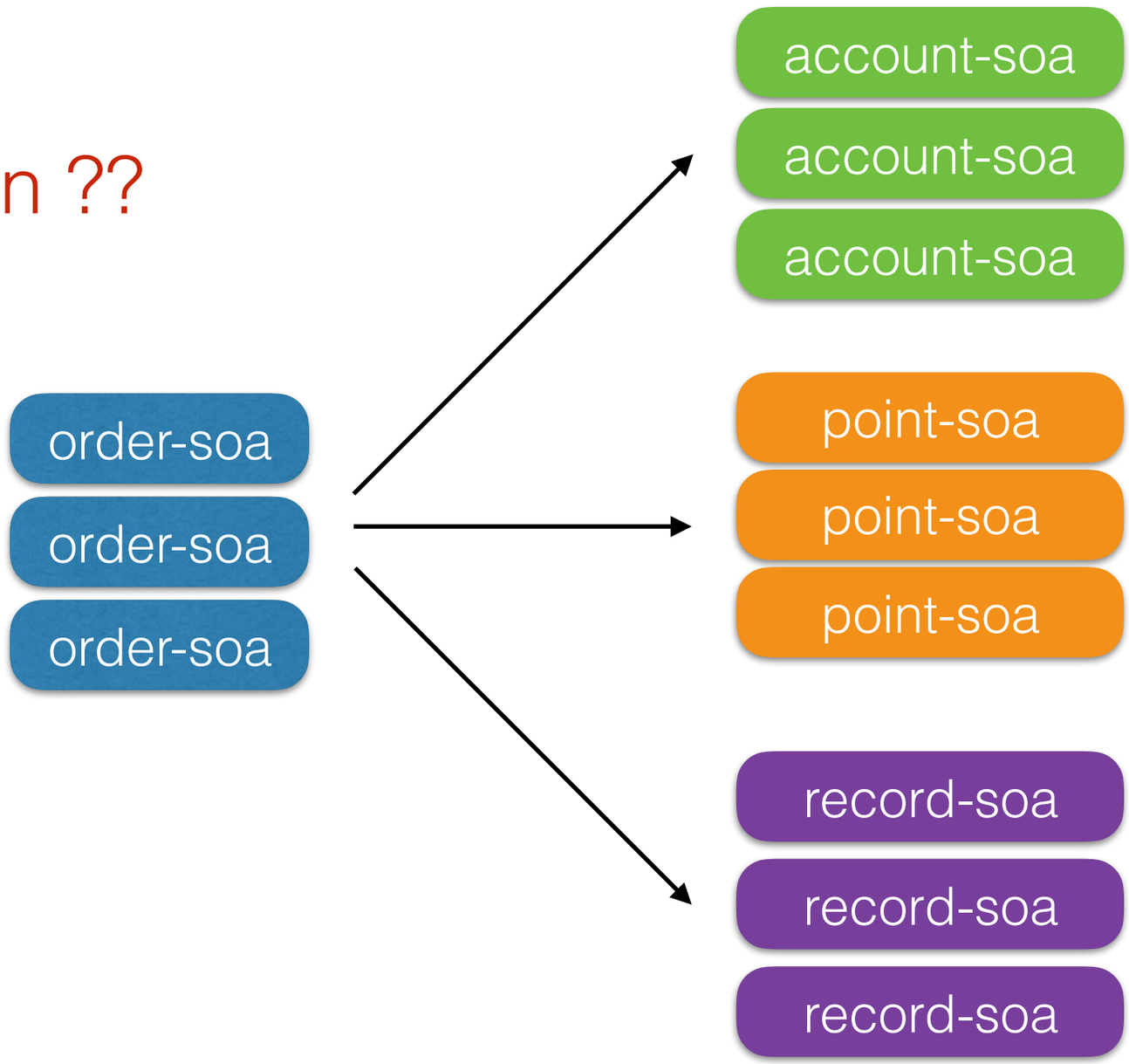
2.集群应用实践

Application

```
@Transactional(rollbackFor=Exception.class)
makePaymentOrder() {
    accountPay()
    pointPay()
    recordActivity()
}
```

3.分布式/微服务应用实践

Transaction ??



3.分布式/微服务应用实践

全局事务：

基于两段式提交，典型的有XA，JTA等

3.分布式/微服务应用实践

BASE理论：

BA: Basic Availability 基本可用性

S: Soft state 柔性

E: Eventual consistency 最终一致性

柔性与全局事务相对

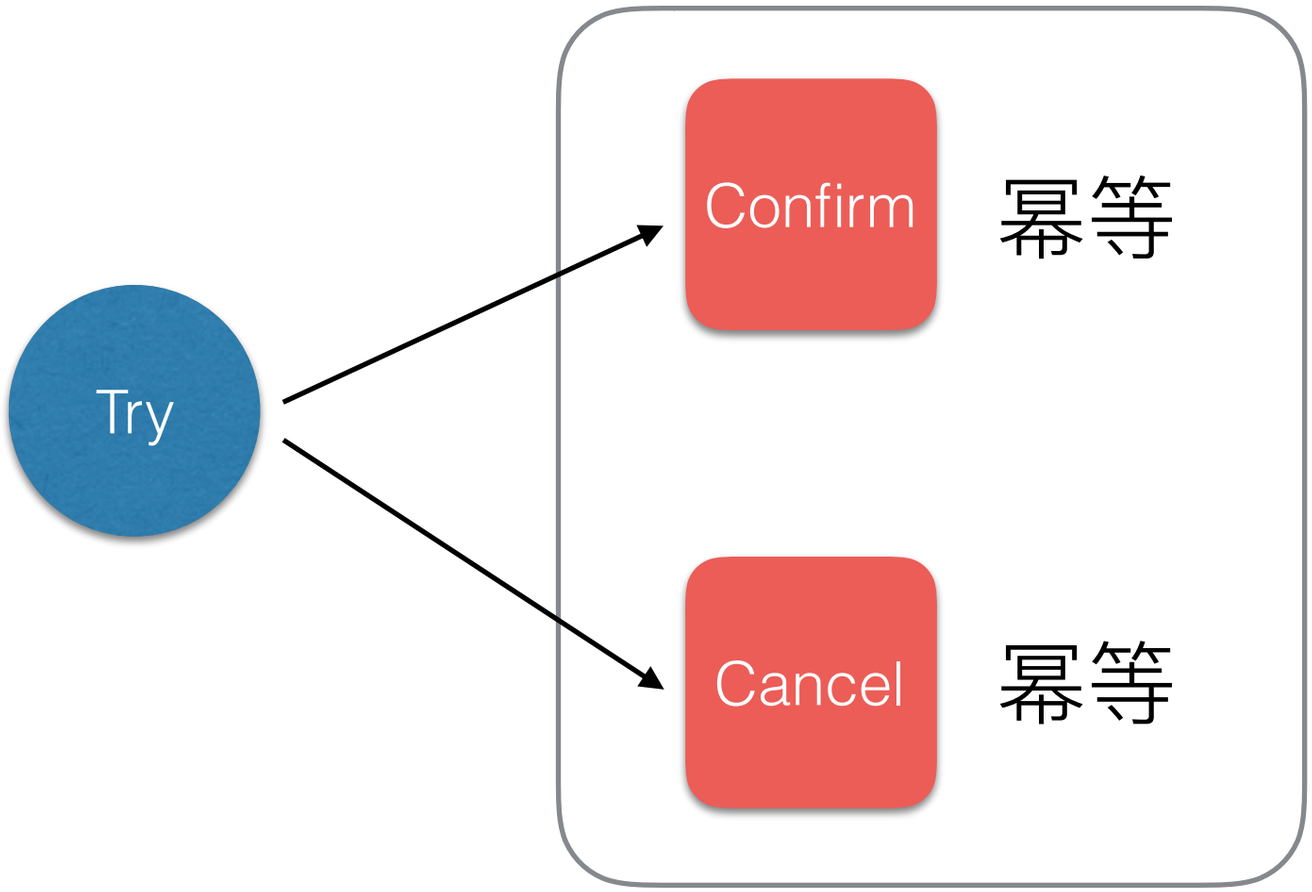
3.分布式/微服务应用实践

柔性事务：消息中间件

由于recordActivity的行为不需要与主事务强一致，所以可以用发消息的方式来触发

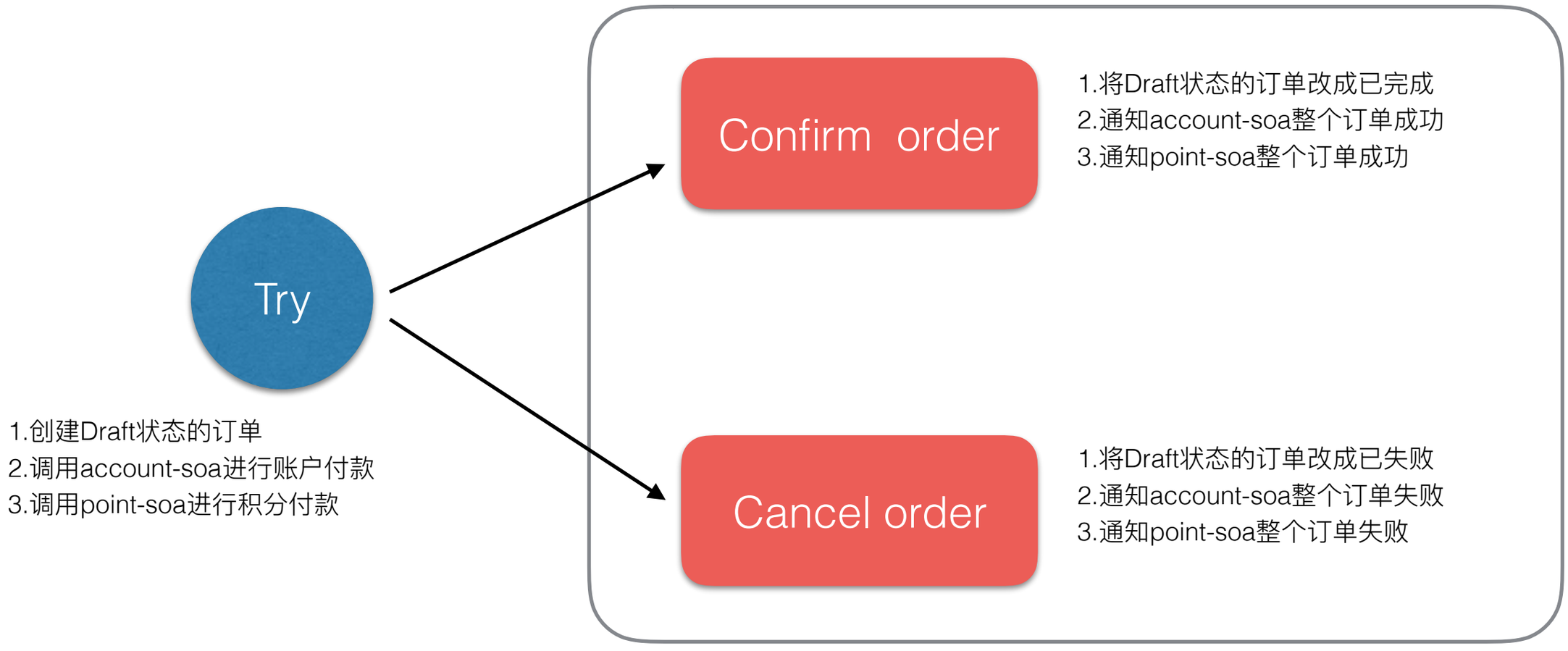
3.分布式/微服务应用实践

柔性事务：Try Confirm Cancel模型



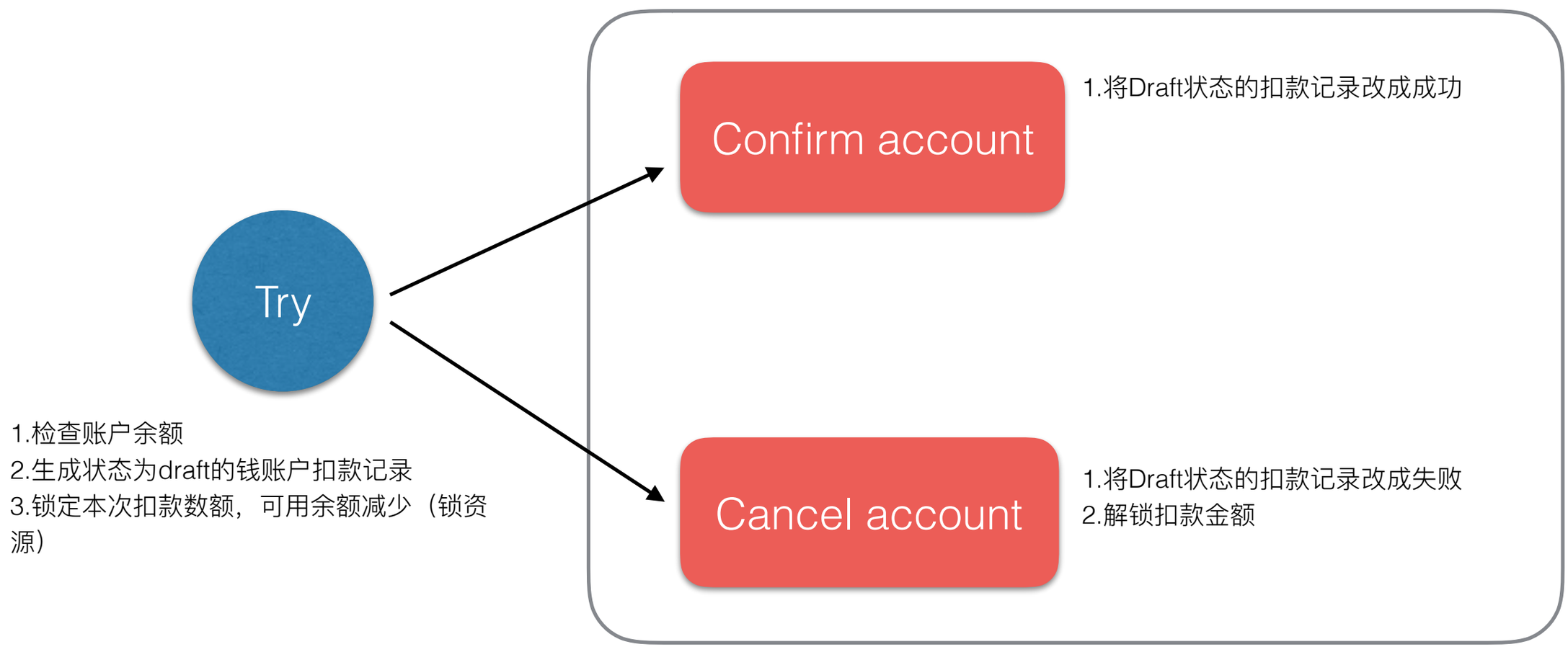
3.分布式/微服务应用实践

主事务：makePaymentOrder



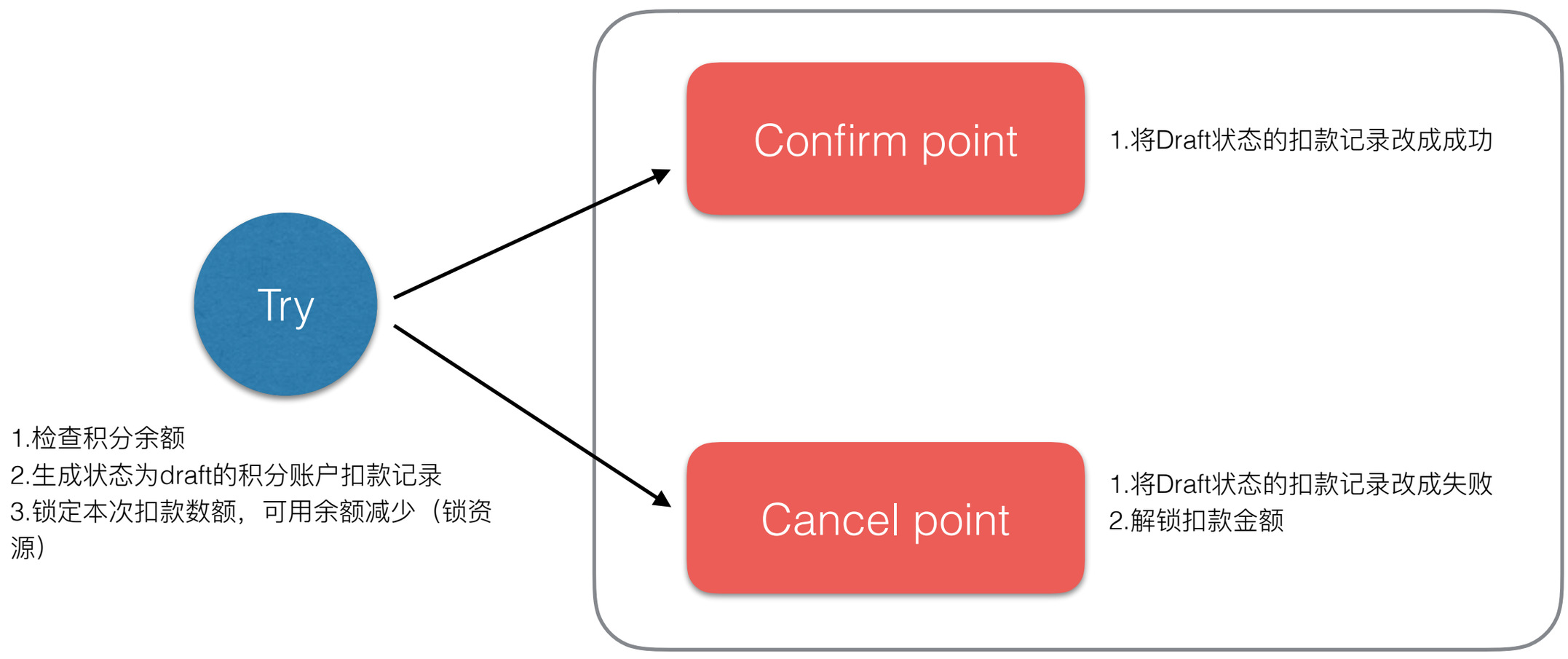
3.分布式/微服务应用实践

子事务：accountPay



3.分布式/微服务应用实践

子事务：pointPay

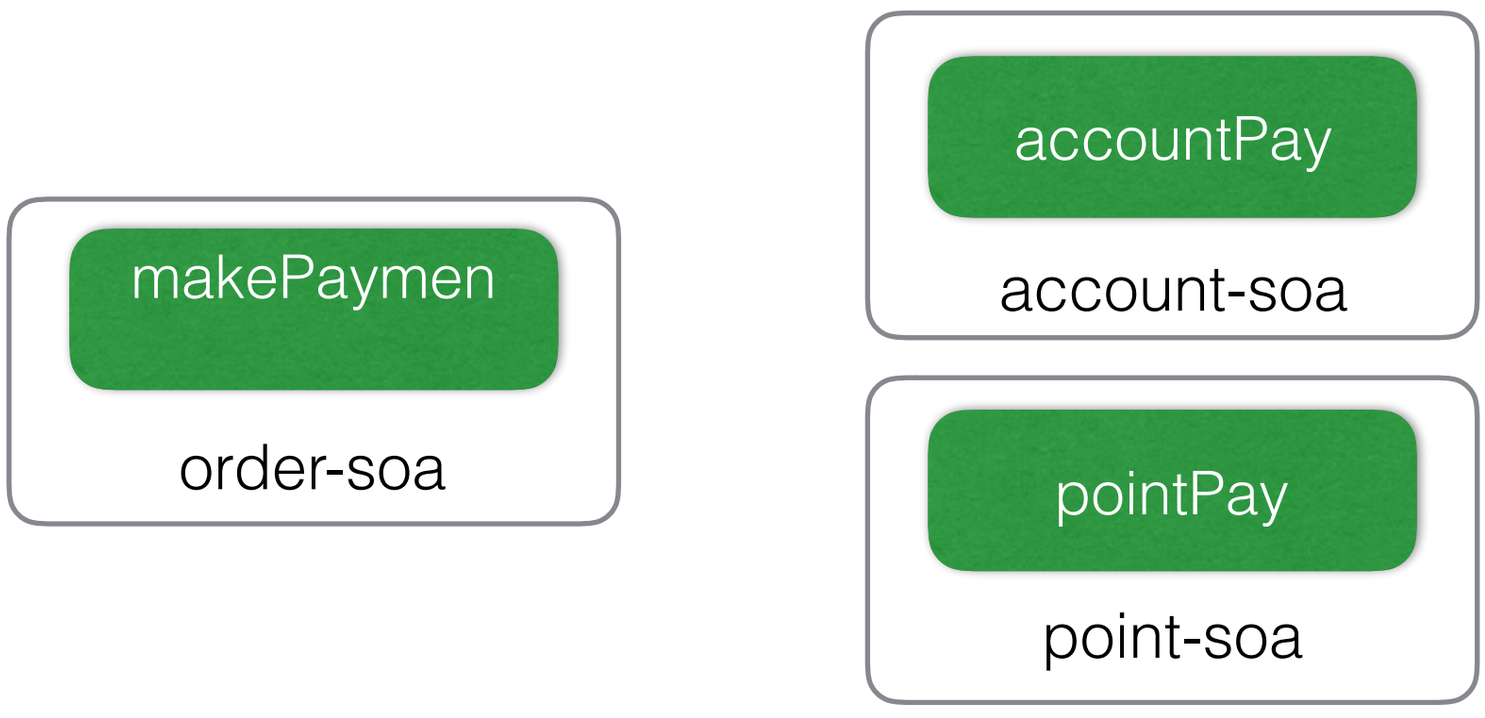


3.分布式/微服务应用实践

整体事务

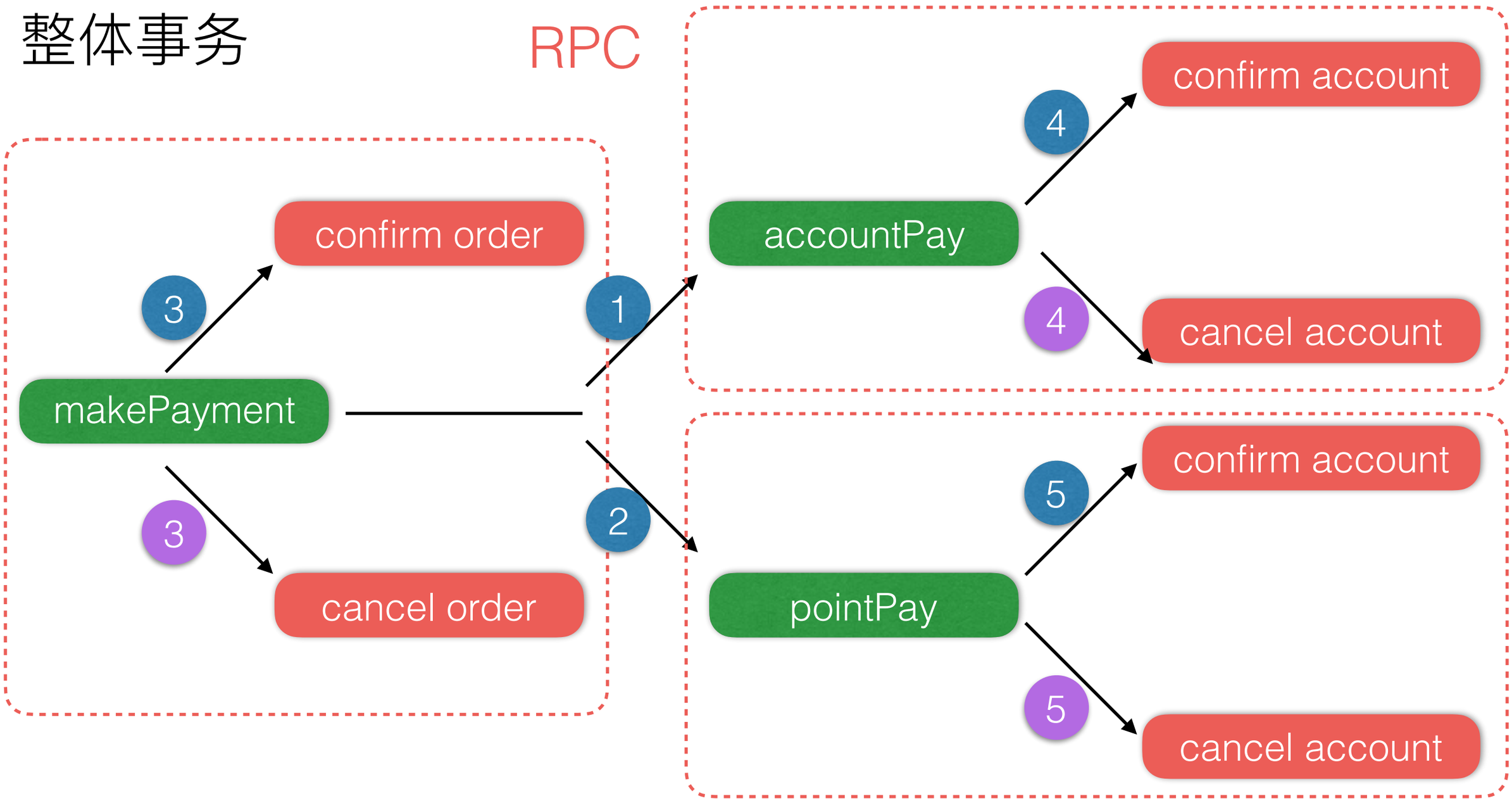
发起者

参与者



3.分布式/微服务应用实践

整体事务



3.分布式/微服务应用实践

最终一致性：

1.创建“全局事务”ID，作为“事务日志”的唯一主键，持久化（MySQL, Redis … 怎么高效怎么来）。

2.正式RPC前，先将服务标记为 Participant，持久化。

3.当事务被中断，通过Scheduler定时扫描持久化的未完成的事务进行补偿（要么confirm，要么cancel），达到最终一致。

4.蚂蚁金服DTS

分布式事务服务 (Distributed Transaction Service, DTS) 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。DTS 从架构上分为 xts-client 和 xts-server 两部分，前者是一个嵌入客户端应用的 JAR 包，主要负责事务数据的写入和处理；后者是一个独立的系统，主要负责异常事务的恢复。

https://www.cloud.alipay.com/docs/content/AntCloud/46881#/?_k=4toad8

5.TCC-Transaction工作原理

Github:https://github.com/changmingxie/tcc-transaction

5.TC 原理

Github:h ansaction