不用分布式事务会发生什么:
包含了库存和订单两个独立的微服务，每个微服务维护了自己的数据库。在交易系统的业务逻辑中，一个商品在下单之前需要先调用库存服务，进行扣除库存，再调用订单服务，创建订单记录。
正常情况下，两个数据库各自更新成功，两边数据维持着一致性。
但是，在非正常情况下，有可能库存的扣减完成了，随后的订单记录却因为某些原因插入失败。这个时候，两边数据就失去了应有的一致性。
什么是分布式事务:
分布式事务用于在分布式系统中保证不同节点之间的数据一致性
分布式事务的实现:
XA协议包含两阶段提交（2PC）和三阶段提交（3PC）两种实现
在XA协议中包含着两个角色：事务协调者和事务参与者

在XA分布式事务的第一阶段，作为事务协调者的节点会首先向所有的参与者节点发送Prepare请求。
在接到Prepare请求之后，每一个参与者节点会各自执行与事务有关的数据更新，写入Undo Log和Redo Log。如果参与者执行成功，暂时不提交事务，而是向事务协调节点返回“完成”消息。
当事务协调者接到了所有参与者的返回消息，整个分布式事务将会进入第二阶段。
在XA的第一阶段，如果某个事务参与者反馈失败消息，说明该节点的本地事务执行不成功，必须回滚

在XA分布式事务的第二阶段，如果事务协调节点在之前所收到都是正向返回，那么它将会向所有事务参与者发出Commit请求。
接到Commit请求之后，事务参与者节点会各自进行本地的事务提交，并释放锁资源。当本地事务完成提交后，将会向事务协调者返回“完成”消息。
当事务协调者接收到所有事务参与者的“完成”反馈，整个分布式事务完成。

三点问题:

1.性能问题　2.协调者单点故障问题 3.丢失消息导致的不一致问题。

解决方案

1.XA三阶段提交

XA三阶段提交在两阶段提交的基础上增加了CanCommit阶段，并且引入了超时机制。一旦事物参与者迟迟没有接到协调者的commit请求，会自动进行本地commit。这样有效解决了协调者单点故障的问题。但是性能问题和不一致的问题仍然没有根本解决。

2.MQ事务

利用消息中间件来异步完成事务的后一半更新，实现系统的最终一致性。这个方式避免了像XA协议那样的性能问题。

3.TCC事务

TCC事务是Try、Confirm、Cancel三种指令的缩写，其逻辑模式类似于XA两阶段提交，但是实现方式是在代码层面来人为实现。