Distributed transaction Solution

1.背景

随着服务化改造的逐渐上车，依据不同特性的独立性进行解耦，产生了 100 多个微服务。保证微服务间分布式事务的一致性；进行低侵入的改造；具备良好的可靠性和性能；针对这些诉求需要给出一个基于公司 PAAS2.0 平台的解决方案。

2.分布式事务基本概念

2.1 事务的隔离级别

隔离级别越低，数据库的读写效率越高，但是随之带来的异常发生的概率也越大。最高的事务隔离级别（串行化）下，虽然完全不会出现数据库事务异常，但是在高并发事务下数据库的性能是急剧劣化的。

以下是不同隔离级别出现的问题：

- 脏读：读取未提交的事务
- 不可重复读：一个事务两次读写同一个数据结果不同，期间其他事务提交。
- 幻读：和不可重复读类似，一个事务读两次，发现第二次读取的数据比第一次多

以下是不同隔离级别，以及其面临的问题：

- 可读取未提交事务：脏读、幻读、不可重复读
- 可读取已提交事务：不可重复读、幻读
- 可重复读取：幻读（不能控制其他事务 insert）
- 串行化：穿行执行

2.2CAP 理论

CAP 的概念：

- C:consistency,一致性，任意节点的数据表示的意义应该一致。
- A:availability,可用性，不能因为事务而阻塞请求的响应。
- P:partition tolerence，分区容忍性，节点间无法通信，要保证服务能正常使用。

假设有微服务 data1 和微服务 data2，一开始有个数据 number=1。之后向 data1 提交更新，将数据 number 设置为 2。场景分析：

场景 1：保证 C&p，为了保证数据一致性，data1 需要将数据复制给 data2，即 data1 和 data2 需要进行通信。但是由于网络是不可靠的，我们系统有保证了分区容忍性，也就是说这个系统是可以容忍网络的不可靠的。这时候 data2 就不一定能及时的收到 data1 的数据复制消息，当有请求向 data2 访问 number 数据时，为了保证数据的一致性，data2 只能阻塞等待数据真正同步完成后再返回，这时候就没办法保证高可用性了。**在保证 C 和 P 的情况下，是无法同时保证 A 的。**

场景 2：保证 A&P，为了保证高可用性，data1 和 data2 都有在有限时间内返回。同样由于网络的不可靠，在有限时间内，data2 有可能还没收到 data1 发来的数据更新消息，这时候返回给客户端的可能是旧的数据，和访问 data1 的数据是不一致的，也就是违法了 C。**在保证 A 和 P 的情况下，是无法同时保证 C 的。**

场景 3：如果要保证高可用和一致性，只有在网络情况良好且可靠的情况下才能实现。这样 data1 才能立即将更新消息发送给 data2。但是我们都知道网络是不可靠的，是会存在丢包的情况的。所以要满足即时可靠更新，只有将 data1 和 data2 放到一个区内才可以，也就丧失了 P 这个保证。其实这时候整个系统也不能算是一个分布式系统了。

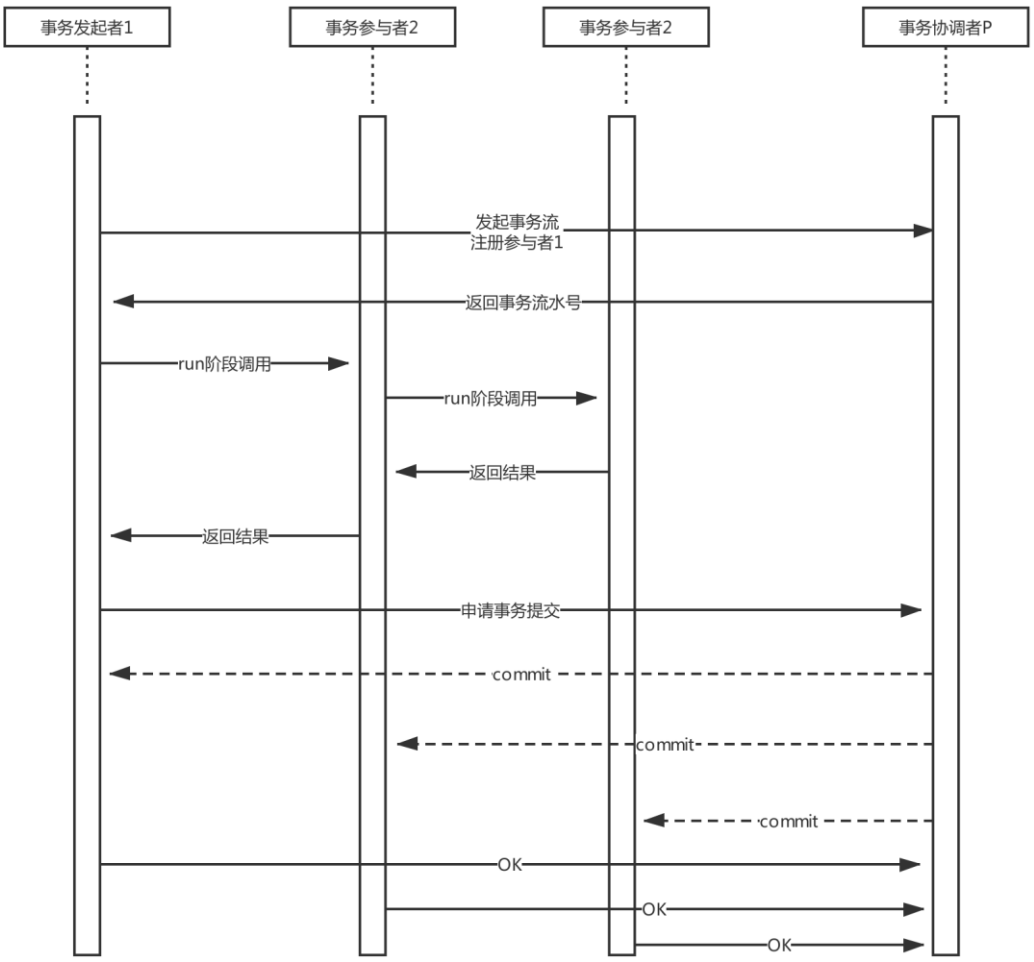
2.3 术语

- DTMS:Distributed Transaction Management Service 分布式事务管理服务
- 事务参与者：一个分布式事务下，涉及的微服务（除了 DTMS）
- 事务协调者：即 DTMS
- 事务发起者：事务的发起人
- TID：分配的事务流水号，用来联系同一事务不同进程间的数据库连接。
- 最终一致性：分布式系统各节点的数据总会在一段时间后完全一致。
- 2P&3P：run 阶段、precommit 阶段、commit 阶段，precommit 给了缓冲确认的时机。
- 可靠性：事务重试机制。
- 有状态的&无状态的：多次请求是否需要绑定到同一个实例。

3.解决方案

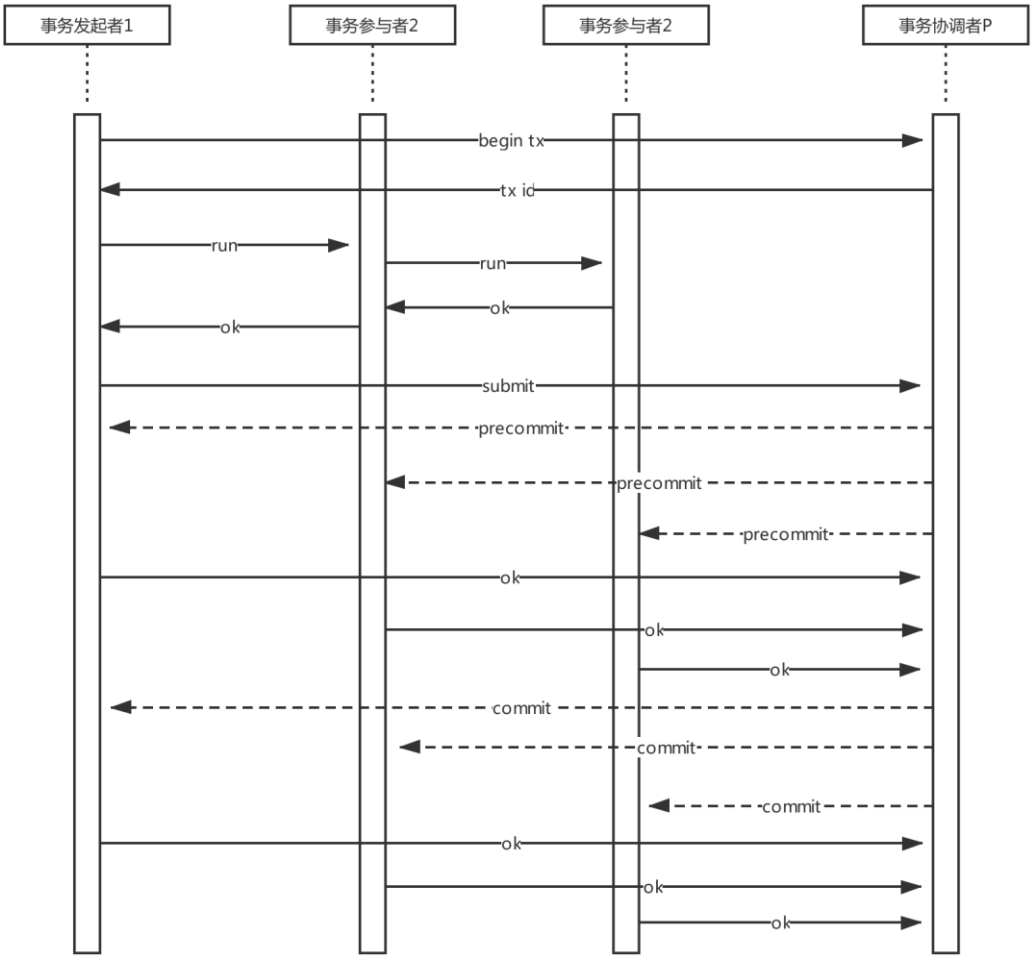
3.1 设计时序图

二阶段提交



第一阶段为 run 阶段，也可以叫表决阶段，这一阶段决定第二阶段是 commit 还是 rollback。第二阶段为 commit&rollback 阶段，根据第一阶段的表决结果决定是提交事务还是回滚事务。

三阶段提交



最基本的分布式事务的时序图应该如上面两张图所示，分别对应二阶段和三阶段。如何理解二阶段、三阶段、以及它们之间的区别？