本课时我们来讨论两阶段提交和三阶段提交协议的过程以及应用。

在分布式系统中，各个节点之间在物理上相互独立，通过网络进行沟通和协调。在关系型数据库中，由于存在事务机制，可以保证每个独立节点上的数据操作满足 ACID。但是，相互独立的节点之间无法准确的知道其他节点中的事务执行情况，所以在分布式的场景下，如果不添加额外的机制，多个节点之间理论上无法达到一致的状态。

在分布式事务中，两阶段和三阶段提交是经典的一致性算法，那么两阶段和三阶段提交的具体流程是怎样的，三阶段提交又是如何改进的呢？

协调者统一调度

在分布式事务的定义中，如果想让分布式部署的多台机器中的数据保持一致性，那么就要保证在所有节点的数据写操作，要么全部都执行，要么全部都不执行。但是，一台机器在执行本地事务的时候无法知道其他机器中本地事务的执行结果，节点并不知道本次事务到底应该 Commit 还是 Rollback。

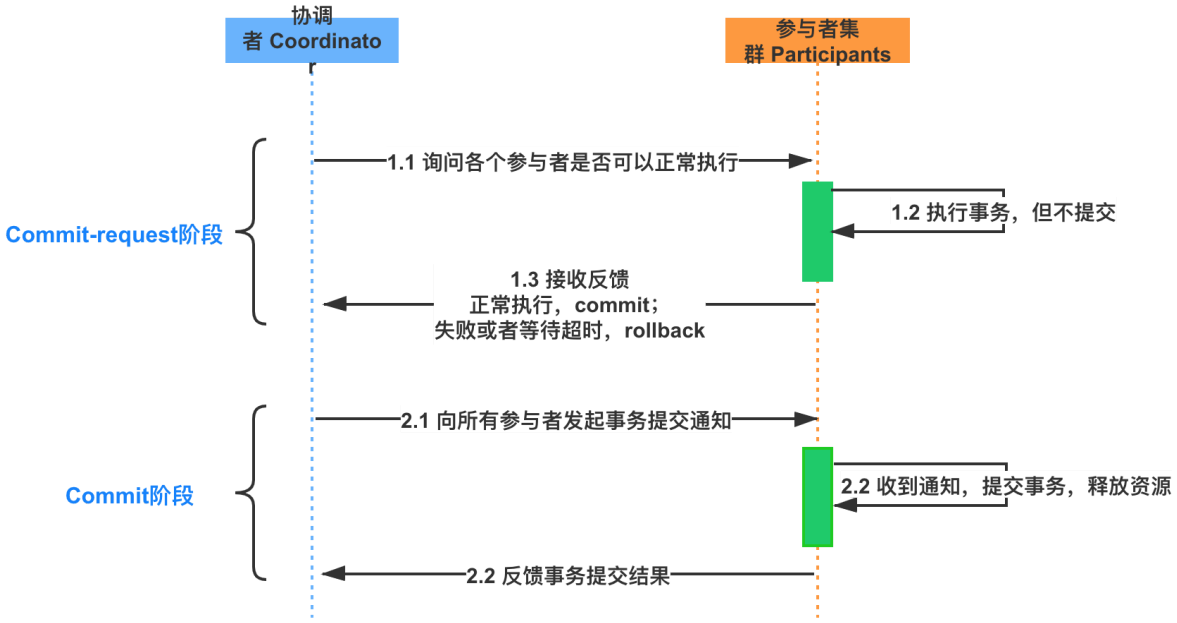
在前面介绍过的几种一致性算法中，都是通过一个 Leader 进程进行协调，在 2PC（两阶段）和 3PC（三阶段）中也是一样的解决办法。二阶段和三阶段提交协议都是引入了一个协调者的组件来统一调度所有分布式节点的执行，让当前节点知道其他节点的任务执行状态，通过通知和表决的方式，决定执行 Commit 还是 Rollback 操作。

二阶段提交协议

二阶段提交算法的成立是基于以下假设的：

- 在该分布式系统中，存在一个节点作为**协调者**（Coordinator），其他节点作为**参与者**（Participants），且节点之间可以进行网络通信；
- 所有节点都采用预写式日志，日志被写入后被保存在可靠的存储设备上，即使节点损坏也不会导致日志数据的丢失；
- 所有节点不会永久性损坏，即使损坏后仍然可以恢复。

两阶段提交中的两个阶段，指的是 **Commit-request 阶段**和 **Commit 阶段**，两阶段提交的流程如下：



提交请求阶段

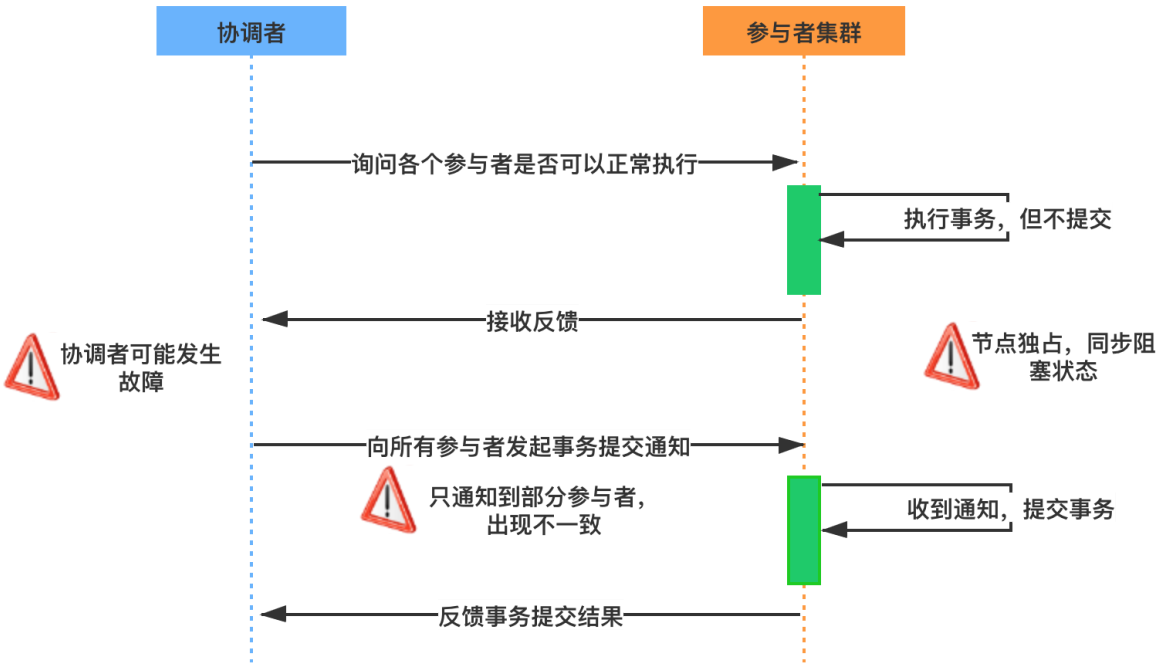
在提交请求阶段，协调者将通知事务参与者准备提交事务，然后进入表决过程。在表决过程中，参与者将告知协调者自己的决策：同意（事务参与者本地事务执行成功）或取消（本地事务执行故障），在第一阶段，参与节点并没有进行Commit操作。

提交阶段

在提交阶段，协调者将基于第一个阶段的投票结果进行决策：提交或取消这个事务。这个结果的处理和前面基于半数以上投票的一致性算法不同，必须当且仅当所有的参与者同意提交，协调者才会通知各个参与者提交事务，否则协调者将通知各个参与者取消事务。

参与者在接收到协调者发来的消息后将执行对应的操作，也就是本地 Commit 或者 Rollback。

两阶段提交存在的问题



两阶段提交协议有几个明显的问题，下面列举如下。

- 资源被同步阻塞

在执行过程中，所有参与节点都是事务独占状态，当参与者占有公共资源时，那么第三方节点访问公共资源会被阻塞。

- 协调者可能出现单点故障

一旦协调者发生故障，参与者会一直阻塞下去。

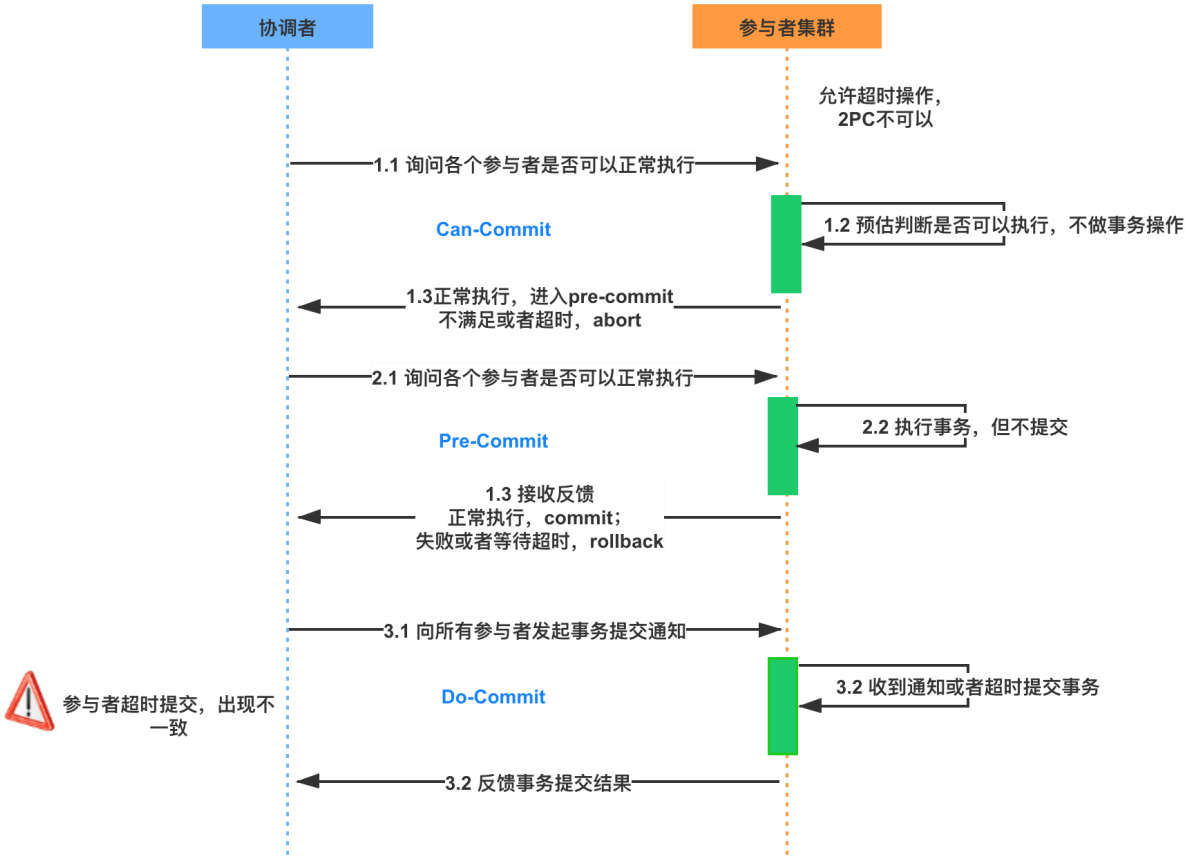
- 在 Commit 阶段出现数据不一致

在第二阶段中，假设协调者发出了事务 Commit 的通知，但是由于网络问题该通知仅被一部分参与者所收到并执行 Commit，其余的参与者没有收到通知，一直处于阻塞状态，那么，这段时间就产生了数据的不一致性。

三阶段提交协议

为了解决二阶段协议中的同步阻塞等问题，三阶段提交协议在协调者和参与者中都引入了超时机制，并且把两阶段提交协议的第一个阶段拆分成了两步：询问，然后再锁资源，最后真正提交。

三阶段中的 Three Phase 分别为 CanCommit、PreCommit、DoCommit 阶段。



CanCommit 阶段

3PC 的 CanCommit 阶段其实和 2PC 的准备阶段很像。协调者向参与者发送 Can-Commit 请求，参与者如果可以提交就返回 Yes 响应，否则返回 No 响应。

PreCommit 阶段

协调者根据参与者的反应情况来决定是否可以继续事务的 PreCommit 操作。根据响应情况，有以下两种可能。

A. 假如协调者从所有的参与者获得的反馈都是 Yes 响应，那么就会进行事务的预执行：

- **发送预提交请求**，协调者向参与者发送 PreCommit 请求，并进入 Prepared 阶段；
- **事务预提交**，参与者接收到 PreCommit 请求后，会执行事务操作；
- **响应反馈**，如果参与者成功执行了事务操作，则返回 ACK 响应，同时开始等待最终指令。

B. 假如有任何一个参与者向协调者发送了 No 响应，或者等待超时之后，协调者都没有接到参与者的响应，那么就中断事务：

- **发送中断请求**，协调者向所有参与者发送 abort 请求；
- **中断事务**，参与者收到来自协调者的 abort 请求之后，执行事务的中断。

DoCommit 阶段

该阶段进行真正的事务提交，也可以分为以下两种情况。

A. 执行提交

- 发送提交请求。协调者接收到参与者发送的 ACK 响应后，那么它将从预提交状态进入到提交状态，并向所有参与者发送 doCommit 请求。
- 事务提交。参与者接收到 doCommit 请求之后，执行正式的事务提交，并在完成事务提交之后释放所有事务资源。
- 响应反馈。事务提交完之后，向协调者发送 ACK 响应。
- 完成事务。协调者接收到所有参与者的 ACK 响应之后，完成事务。

B. 中断事务

协调者没有接收到参与者发送的 ACK 响应，可能是因为接受者发送的不是 ACK 响应，也有可能响应超时了，那么就会执行中断事务。

C.超时提交

参与者如果没有收到协调者的通知，超时之后会执行 Commit 操作。

三阶段提交做了哪些改进

引入超时机制

在 2PC 中，**只有协调者拥有超时机制**，如果在一定时间内没有收到参与者的消息则默认失败，3PC 同时在协调者和参与者中都引入超时机制。

添加预提交阶段

在 2PC 的准备阶段和提交阶段之间，插入一个准备阶段，使 3PC 拥有 CanCommit、PreCommit、DoCommit 三个阶段，PreCommit 是一个缓冲，保证了在最后提交阶段之前各参与节点的状态是一致的。

三阶段提交协议存在的问题

三阶段提交协议同样存在问题，具体表现为，在阶段三中，如果参与者接收到了 PreCommit 消息后，出现了不能与协调者正常通信的问题，在这种情况下，参与者依然会进行事务的提交，这就出现了数据的不一致性。

两阶段和三阶段提交的应用

两阶段提交是一种比较精简的一致性算法/协议，很多关系型数据库都是采用两阶段提交协议来完成分布式事务处理的，典型的比如 MySQL 的 XA 规范。

在事务处理、数据库和计算机网络中，两阶段提交协议提供了分布式设计中的数据一致性的保障，整个事务的参与者要么一致性全部提交成功，要么全部回滚。MySQL Cluster 内部数据的同步就是用的 2PC 协议。

MySQL 的主从复制

在 MySQL 中，二进制日志是 server 层，主要用来做**主从复制**和**即时点恢复**时使用的；而事务日志（Redo Log）是 InnoDB 存储引擎层，用来保证事务安全的。

在数据库运行中，需要保证 Binlog 和 Redo Log 的一致性，如果顺序不一致， 则意味着 Master-Slave 可能不一致。

在开启 Binlog 后，如何保证 Binlog 和 InnoDB redo 日志的一致性呢？MySQL 使用的就是二阶段提交，内部会自动将普通事务当做一个 XA 事务（内部分布式事务）来处理：

- Commit 会被自动的分成 Prepare 和 Commit 两个阶段；
- Binlog 会被当做事务协调者（Transaction Coordinator），Binlog Event 会被当做协调者日志。

关于 XA 规范的具体实现，会在后面的课时中分享。

总结

两阶段和三阶段提交协议是众多分布式算法的基础，这一课时介绍了两阶段提交和三阶段提交的具体流程，两种协议的区别，以及两阶段提交在 MySQL 主从复制中的应用。

精选评论

**博:

> 为了实现分布式事务而设计的二阶段提交协议2P和三阶段提交协议3P；无论是2P还是3P都不能100%保证事务的正确性

**杰:

> 在执行过程中，所有参与节点都是事务独占状态，当参与者占有公共资源时，那么第三方节点访问公共资源会被阻塞。 想问一下这个公共资源是什么

**讲师回复:

> 这句话前后是解释关系，公共资源就是被参与节点「事务独占」的数据、文件等，是 2PC 和 3PC 协议维护的数据

**子:

> 3PC 中"参与者如果没有收到协调者的通知，超时之后会执行 Commit 操作"，超时了为啥还要commit,不应该是rollback吗?

**讲师回复:

> 进入第三阶段时，说明参与者在第二阶段已经收到了PreCommit请求，意味它知道大家其实都同意修改了。当进入第三阶段时，由于网络超时等原因，虽然参与者没有收到commit或者abort响应，但是此时成功提交的几率很大，所以会执行commit。

**7419:

> 三阶段提交如果在最后doCommit阶段提交了，但是由于网络原因ACK响应没有被协调者收到怎么办?

**讲师回复:

> 如果超时或者响应丢失，协调者会发起回滚动作，中断事务的进行

*林:

> 二阶段提交协议为什么第一步是协调者先发出询问？整个事件操作最开始的入口不应该是某个参与者先发起的吗?

**讲师回复:

> 协调者收到提案以后进入准备阶段

**文:

> 老师好，请问一下三阶段的第一阶段只做服务状态的确认不做事务相关操作 改怎么理解这句话呢？ 或者从实现的角度来说应该是怎样的一种过程?

**讲师回复:

> 你可以从一致性的角度，整体理解三阶段和两阶段，以及paxos这些协议。举个例子，现在开会评审一个需求，先问各方能不能支持，就是一个「确认」操作。

***州：**

> “发送提交请求。协调者接收到参与者发送的 ACK 响应后，那么它将从预提交状态进入到提交状态，并向所有参与者发送 doCommit 请求”协调者接收到参与者发送的 ACK 响应后 是所有参与者的ack 还是部分ack 呢?

讲师回复：

> 在docommit阶段，协调者需要接到所有参与者发送的ack响应，才进入提交状态，如果有参与者返回异常，会中断事务。

****航：**

> 事务独占是指参与结点同一时间只能执行一个事务?

讲师回复：

> 这么理解也可以，换个说法更贴切：不是只能，是只在执行，此处的独占更多的描述一种「状态」。

****方：**

> “3PC 的 CanCommit 阶段其实和 2PC 的准备阶段很像。协调者向参与者发送 Commit 请求，参与者如果可以提交就返回 Yes 响应，否则返回 No 响应。”--这句话不对话吧！3PC的CanCommit阶段应该只是做服务状态的确认，而2PC的“一阶段”流程已经是执行事务提交了（只是没commit），两者者不一样。而且你上面的3PC流程图画着2.2、3.2步骤是跟2PC一样的。

讲师回复：

> 应该是“协调者向参与者发送Can-Commit请求”，已经订正，谢谢指出

***洁：**

> 有上课写的中间件Seata，这个应该只是作为了解吧?

讲师回复：

> Seata是比较成熟的分布式事务组件，感兴趣可以去了解下