1.两阶段提交协议(2PC)

1.1 原理介绍

第一阶段:

TM通知各个RM准备提交它们的事务分支。如果RM判断自己进行的工作可以被提交，那就就对工作内容进行持久化，再给TM肯定答复；要是发生了其他情况，那给TM的都是否定答复。在发送了否定答复并回滚了已经的工作后，RM就可以丢弃这个事务分支信息。

以mysql数据库为例，在第一阶段，事务管理器向所有涉及到的数据库服务器发出prepare"准备提交"请求，数据库收到请求后执行数据修改和日志记录等处理，处理完成后只是把事务的状态改成"可以提交",然后把结果返回给事务管理器。

第二阶段:

TM根据阶段1各个RM prepare的结果，决定是提交还是回滚事务。如果所有的RM都prepare成功，那么TM通知所有的RM进行提交；如果有RM prepare失败的话，则TM通知所有RM回滚自己的事务分支。

以mysql数据库为例，如果第一阶段中所有数据库都prepare成功，那么事务管理器向数据库服务器发出"确认提交"请求，数据库服务器把事务的"可以提交"状态改为"提交完成"状态，然后返回应答。如果在第一阶段内有任何一个数据库的操作发生了错误，或者事务管理器收不到某个数据库的回应，则认为事务失败，回撤所有数据库的事务。数据库服务器收不到第二阶段的确认提交请求，也会把"可以提交"的事务回撤。

1.2 存在的问题

1）同步阻塞问题

两阶段提交方案下全局事务的ACID特性，是依赖于RM的。我们使用mysql来支持XA分布式事务的话，那么最好将事务隔离级别设置为SERIALIZABLE。SERIALIZABLE(串行化)是四个事务隔离级别中最高的一个级别，也是执行效率最低的一个级别。

2）单点故障

由于协调者的重要性，一旦协调者TM发生故障。参与者RM会一直阻塞下去。尤其在第二阶段，协调者发生故障，那么所有的参与者还都处于锁定事务资源的状态中，而无法继续完成事务操作。（如果是协调者挂掉，可以重新选举一个协调者，但是无法解决因为协调者宕机导致的参与者处于阻塞状态的问题）

3）数据不一致

在二阶段提交的阶段二中，当协调者向参与者发送commit请求之后，发生了局部网络异常或者在发送commit请求过程中协调者发生了故障，这回导致只有一部分参与者接受到了commit请求。而在这部分参与者接到commit请求之后就会执行commit操作。但是其他部分未接到commit请求的机器则无法执行事务提交。于是整个分布式系统便出现了数据不一致性的现象。

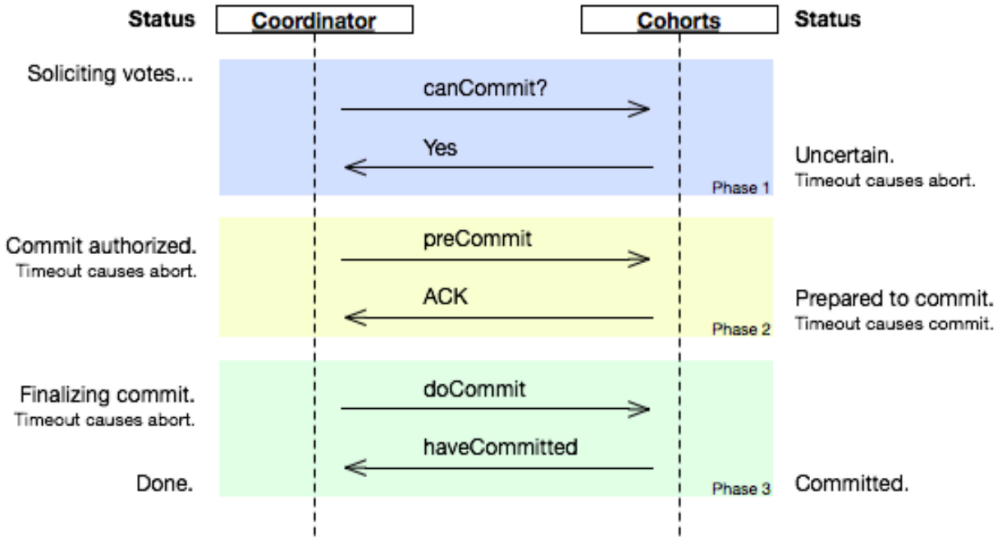
由于二阶段提交存在着诸如同步阻塞、单点问题等缺陷，所以，研究者们在二阶段提交的基础上做了改进，提出了三阶段提交。

2 三阶段提交协议(Three-phase commit)

三阶段提交（3PC），是二阶段提交（2PC）的改进版本。与两阶段提交不同的是：

1）引入超时机制。同时在协调者和参与者中都引入超时机制。

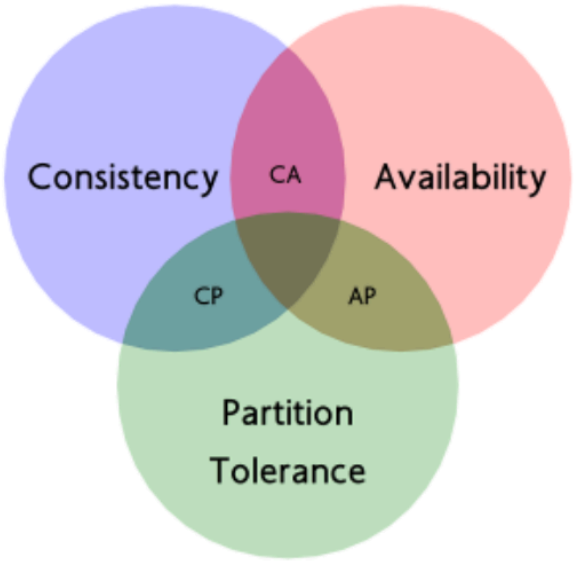
2）在第一阶段和第二阶段中插入一个准备阶段。保证了在最后提交阶段之前各参与节点的状态是一致的。也就是说，除了引入超时机制之外，3PC把2PC的准备阶段再次一分为二，这样三阶段提交就有CanCommit、PreCommit、DoCommit三个阶段。



3 柔性事务

3.1 分布式系统理论-CAP

加州大学伯克利分校的Eric Brewer教授提出CAP猜想，设计一个大规模的分布式系统时会遇到三个特性：一致性（consistency）、可用性（Availability）、分区容错（partition-tolerance），而一个分布式系统最多只能满足其中的2项。



1.）一致性

一致性指“all nodes see the same data at the same time”，即更新操作成功并返回客户端完成后，所有节点在同一时间的数据完全一致，不能存在中间状态。如果时刻保证客户端看到的数据都是一致的，那么称之为强一致性。如果允许存在中间状态，只要求经过一段时间后，数据最终是一致的，则称之为最终一

致性。此外，如果允许存在部分数据不一致，那么就称之为弱一致性。

例如对于电商系统用户下单操作，库存减少、用户资金账户扣减等操作必须在用户下单操作完成后必须是一致的。不能出现类似于库存已经减少，而用户资金账户尚未扣减，积分也未增加的情况。

2） 可用性

可用性是指系统提供的服务必须一直处于可用的状态，对于用户的每一个操作请求总是能够在有限的时间内返回结果。

“有限的时间内”是指，对于用户的一个操作请求，系统必须能够在指定的时间内返回对应的处理结果，如果超过了这个时间范围，那么系统就被认为是不可用的。试想，如果一个下单操作，为了保证分布式事务的一致性，需要10分钟才能处理完。“返回结果”是可用性的另一个非常重要的指标，它要求系统在完成对用户请求的处理后，返回一个正常的响应结果，不论这个结果是成功还是失败。

3） 分区容错性

分布式系统在遇到任何网络分区故障的时候，仍然需要能够保证对外提供满足一致性和可用性的服务，除非是整个网络环境都发生了故障。

总结：

一个分布式系统无法同时满足一致性、可用性、分区容错性三个特点，我们就需要抛弃一个，对于一个分布式系统而言，分区容错性是一个最基本的要求。因为分布式系统中各模块必然被部署到不同的节点，而各模块调用时的网络问题又是一个必定会出现的异常情况，因此分区容错性也就成为了一个分布式系统必然需要面对和解决的问题。因此架构设计需要把精力花在如何根据业务特点在C（一致性）和A（可用性）之间寻求平衡。

XA 两阶段提交协议的分布式事务方案，强调的就是一致性；由于可用性较低，实际应用的并不多。而基于BASE理论的柔性事务，强调的是可用性，目前大行其道。

3.1 BASE理论

eBay的架构师Dan Pritchett源于对大规模分布式系统的实践总结，提出BASE理论。BASE理论是对CAP理论的延伸，核心思想是即使无法做到强一致性（Strong Consistency，CAP的一致性就是强一致性），但应用可以采用适合的方式达到最终一致性（Eventual Consitency）。BASE是Basically Available（基本可用）、Soft state（软状态）和Eventually consistent（最终一致性）三个短语的缩写。

1.）基本可用（Basically Available）

指分布式系统在出现不可预知故障的时候，允许损失部分可用性。

2） 软状态（ Soft State）

指允许系统中的数据存在中间状态，并认为该中间状态的存在不会影响系统的整体可用性。

3） 最终一致（ Eventual Consistency）

强调的是所有的数据更新操作，在经过一段时间的同步之后，最终都能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。

BASE理论面向的是大型高可用可扩展的分布式系统，和传统的事物ACID特性是相反的。它完全不同于ACID的强一致性模型，而是通过牺牲强一致性来获得可用性，并允许数据在一段时间内是不一致的，但最终达到一致状态。但同时，在实际的分布式场景中，不同业务单元和组件对数据一致性的要求是不同的，因此在具体的分布式系统架构设计过程中，ACID特性和BASE理论往往又会结合在一起。