# 分布式事务

### 一．业务场景：

1. 当业务量级扩大之后分库
2. 微服务的业务服务化

以上业务场景都会产生分布式数据不一致的问题。既然本地事务无法满足需求，因此分布式事务就要登上舞台。

### 二．什么是分布式事务？

简单理解：保证不同数据库之间数据一致性事务解决方案。

下面我们了解一下：

1. CAP原则和 BASE 理论。

1.1 CAP原则：

* Consistency: 一致性
* Availablity: 可用性
* Partition-tolerance: 分区容错性

CAP 是上面三个单词的缩写。它是分布式系统中的平衡理论。

Ps: 难道不同数据库之间一致性靠它来保证？

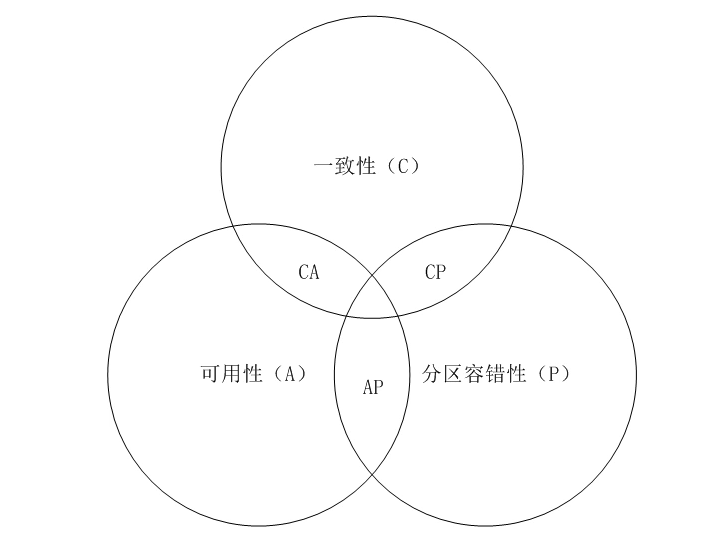
1.2 在分布式系统中

* 一致性：要求所有节点每次读操作都能保证获取到最新数据；
* 可用性：要求无论任何故障产生后都能保证服务仍然可用；
* 分区容错性：要求被分区的节点可以正常对外提供服务。（分区节点是什么？）

事实上，任何系统只可同时满足其中两个，无法三者兼顾。（为什么？）

对分布式系统而言，分区容错性是一个最基本的要求。

* 如果选择了一致性和分区容错性，放弃可用性，那么网络问题会导致系统不可用。
* 如果选择了可用性和分区容错性，放弃一致性，不同节点之间的数据不能及时同步数据不能及时同步数据而导致数据的不一致。



此时，BASE 理论针对一致性和可用性提出了一个方案。

1. BASE 理论

* Basically Availble : 基本可用
* Soft-state: 软状态
* Eventually Consistent: 最终一致性

它是最终一致性的理论支撑。

简单理解：在分布式系统中，允许损失部分可用性，并且不同节点进行数据同步的过程存在延时，但是在经过一段时间的修复后，最终能够达到数据的最终一致性。BASE 强调的就是数据的最终一致性。相对于 ACID 而言，BASE 通过允许损失部分一致性来获得可用性。

**网络分区的解释：**

**一个分布式系统里面，节点组成的网络本来应该是连通的。然而可能因为一些故障，使得有些节点之间不连通了，整个网络就分成了几块区域。数据就散布在了这些不连通的区域中。这就叫分区。**

**当你一个数据项只在一个节点中保存，那么分区出现后，和这个节点不连通的部分就访问不到这个数据了。这时分区就是无法容忍的。**

**提高分区容忍性的办法就是一个数据项复制到多个节点上，那么出现分区之后，这一数据项就可能分布到各个区里。容忍性就提高了。**

**然而，要把数据复制到多个节点，就会带来一致性的问题，就是多个节点上面的数据可能是不一致的。要保证一致，每次写操作就都要等待全部节点写成功，而这等待又会带来可用性的问题。**

**总的来说就是，数据存在的节点越多，分区容忍性越高，但要复制更新的数据就越多，一致性就越难保证。为了保证一致性，更新所有节点数据所需要的时间就越长，可用性就会降低。**

### 三．强一致性解决方案

1. 二阶段提交协议

1.1 业务场景：在分布式系统中，每个数据库只能保证自己的数据可以满足 ACID 保证强一致性，但是它们可能部署在不同的服务器上，只能通过网络进行通信，因此无法准确的知道其他数据库中的事物执行情况。因此，为了解决多个节点之间的协调问题，就需要引入一个协调者负责控制所有节点的操作结果，要么全部成功，要么全部失败。其中，XA 协议是一个分布式事务协议，它有两个角色：事务管理者和资源管理者。

这里我们可以把 事务管理者 理解为协调者，而资源管理者理解为参与者（不同节点？）。

XA 协议通过二阶段提交协议保证强一致性。

（二阶段提交是啥？提交协议又是啥？）

1.2 二阶段协议定义?

顾名思义：它具有两个二阶段：第一个阶段准备、第二个阶段提交。

这里，事务管理者（协调者） 主要负责控制所有节点的操作结果，包括准备流程和提交流程。

已开会为例子：

甲乙丙丁四人要组织一个会议，需要确定会议时间，假设 甲是协调者，乙丙丁是参与者。

投票阶段（第一阶段准备）：

1. 甲发送邮件给乙丙丁，周二 10点开会是否有时间；
2. 甲回复有时间；
3. 已回复有时间；
4. 丙迟迟不回复，此时对于这个活动，甲乙丙均处于阻塞状态，算法无法继续进行；
5. 丙回复有时间（或者没有时间）；

提交阶段（第二阶段提交）：

1. 协调者甲将收集到的结果反馈给乙丙丁（什么时候反馈，以及反馈结果如何，在此列中取决与丙的时间与决定）；
2. 乙收到；
3. 丙收到；
4. 丁收到；

不仅要锁住参与者的所有资源，而且要锁住协调者的资源，开销大。一句话总结就是 2PC 效率很低，分布式事务很难做。

在对事实性要求没有那么高的情况下，可以用基于最大努力交付和消息队列以及消息存储来解决最终一致性。