一致性问题

在前面介绍的分布式系统 CAP 理论中，我们了解了一致性。在这里我们进一步讨论下 简单来说，一致性可以分为以下几类:

- 强一致性
- 弱一致性
- 最终一致性

强一致性

这种一致性级别是符合用户级别的，它要求系统写入什么，就立马读出来什么。用户体验好，但是强一致性的实现难度大，成本高。 绝大多数的系统都在一定成都上弱化了一致性。 比如：银行转账按道理来说是一个强一致的系统，但是整个过程中，还是有时间差，也就是在很短的时间内，存在不一致的时间窗口。我们分析下张三给李四通过银行转账的一个流程， 张三给李四转账 1000 元，系统首先会从张三的账户扣除 1000 元，此时小张可能并没收到 1000 元，此时有个系统处理时间，也就是有个不一致的时间窗口。 当然了由于转账整个操作 是在一个数据库事务中，由于事务的 ACID 特性，所以不用担心钱转出去了对方没收到这种情况。

弱一致性

若一致性，其实很好理解，就是系统对一致性要求不高，即使系统不一致，也能很好的提供服务，不影响用户的体验。 最典型的就是 web 服务，web 服务由于面对不同的客户端，客户端更新策略也是由用户发起的， 所以在同样的时间内，不同的用户看到的内容不一样。 可能只会在某些特定的时间内，系统才表现出一致性的特点。

最终一致性

最终一致性，也是弱一致性的一种特殊情况，就是系统在某一个终态下是一致的。

对于一个分布式存储系统来说，对一致性往往有很高的要求。所以如何在分布式条件下，保持系统的强一致性，是一个较大的挑战。 etcd 显然是一个强一致的键值数据库。 再一次操作生效之后， 整个系统都会对外表现出强一致的特性。 那么分布式存储系统中的一致性是怎么产生的，强一致性系统为什么难以实现，都有哪些挑战呢？

在分布式存储系统中，通常会通过维护多个副本的方式进行容错，以提高系统的可用性。 正是由于多副本的维护，所以产生了多副本之间的一致性问题。 还记得我们前面提到的分布式系统八大缪论吗？ 如果你还记得，那么你知道：

- 节点之间的网络通讯是不可靠的， 包括任意的延迟和内容故障。
- 节点处理可能是错误的，甚至节点自身会出现宕机。
- 同步调用会让系统变得不可扩展。

由于以上种种问题，在多个节点之间维护副本数据的一致性，变的颇具挑战。不管是学术界或是工业界，都比较关心这样的一致性问题。所以这些年也有很多的一致性算法被提出来，用来解决分布式系统中的 一致性问题。