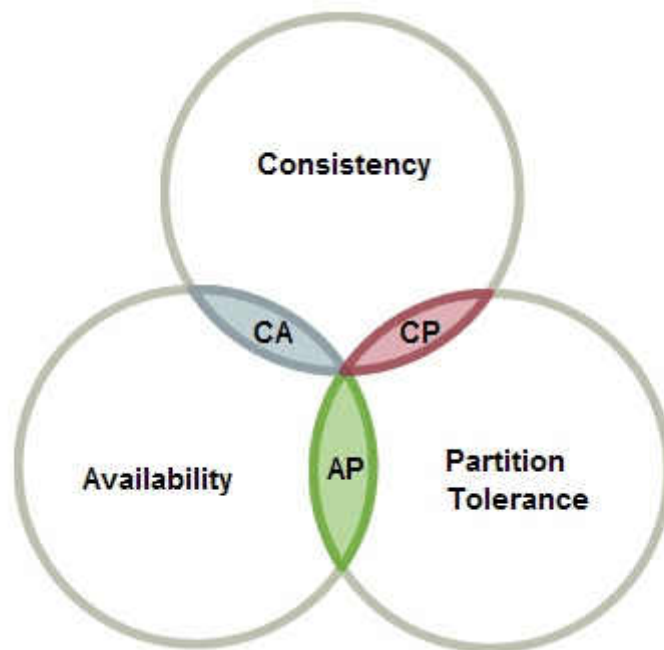


CAP定理



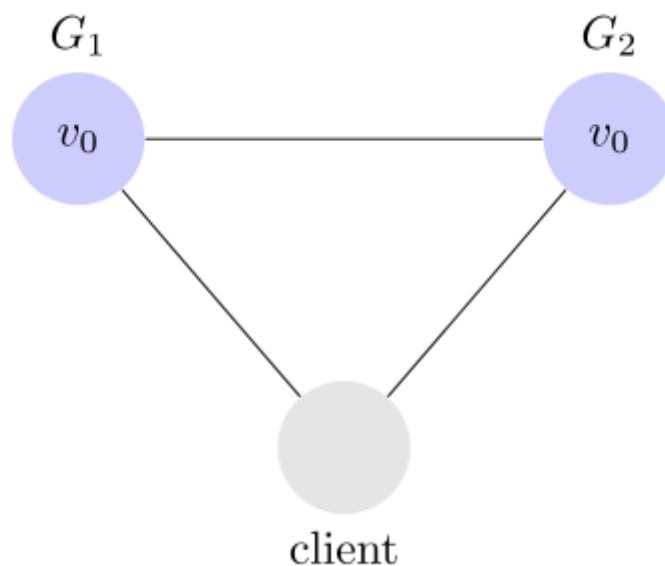
CAP定理是在 1998年加州大学的计算机科学家 Eric Brewer（埃里克.布鲁尔）提出，分布式系统有三个指标

- Consistency 一致性
- Availability 可用性
- Partition tolerance 分区容错性

它们的第一个字母分别是 C、A、P。Eric Brewer 说，这三个指标不可能同时做到。这个结论就叫做 CAP 定理。

分区容错 Partition tolerance

大多数分布式系统都分布在多个子网络。每个子网络就叫做一个区（partition）。分区容错的意思是，区间通信可能失败。比如，一台服务器放在中国，另一台服务器放在美国，这就是两个区，它们之间可能无法通信。



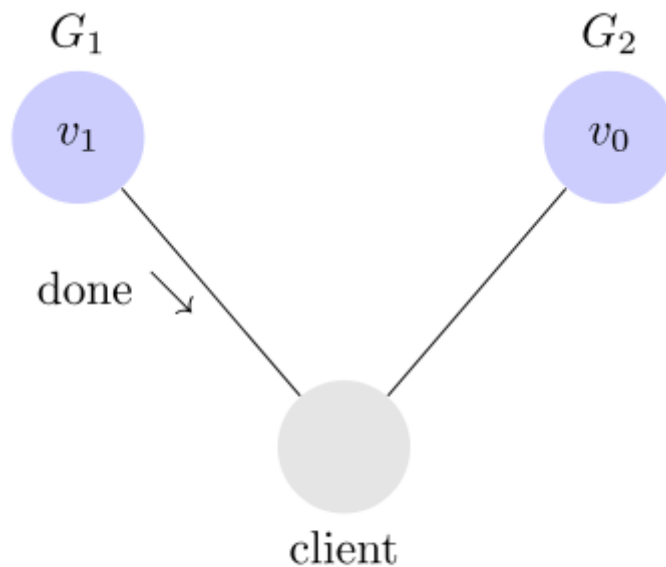
上图中，G1 和 G2 是两台跨区的服务器。G1 向 G2 发送一条消息，G2 可能无法收到。系统设计的时候，必须考虑到这种情况。

一般来说，分区容错无法避免，因此可以认为 CAP 的 P 总是成立。CAP 定理告诉我们，剩下的 C 和 A 无法同时做到。

可用性 Availability

Availability 中文叫做"可用性"，意思是只要收到用户的请求，服务器就必须给出回应。

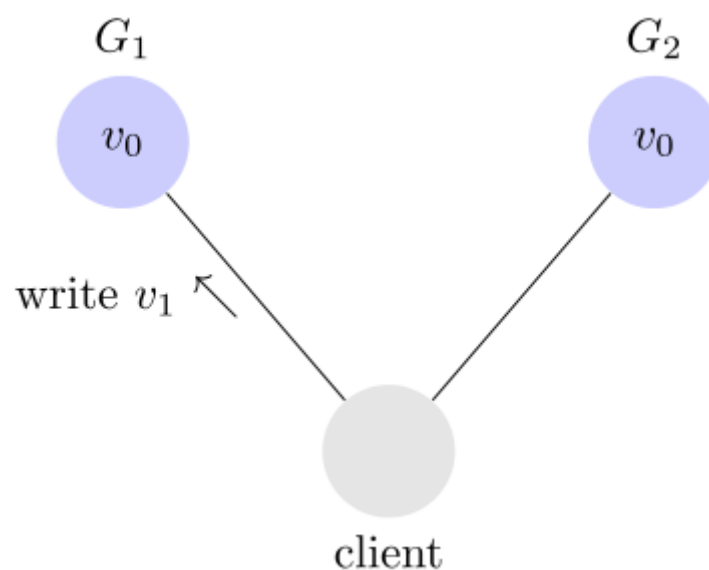
用户可以选择向 G1 或 G2 发起读操作。不管是哪台服务器，只要收到请求，就必须告诉用户，到底是 v0 还是 v1，否则就不满足可用性。



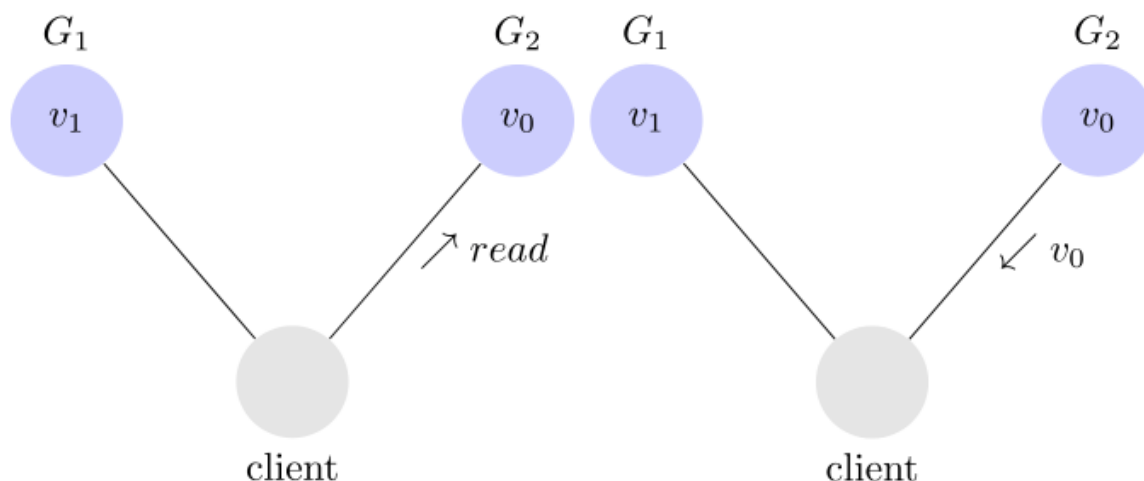
一致性 Consistency

Consistency 中文叫做"一致性"。意思是，写操作之后的读操作，必须返回该值。

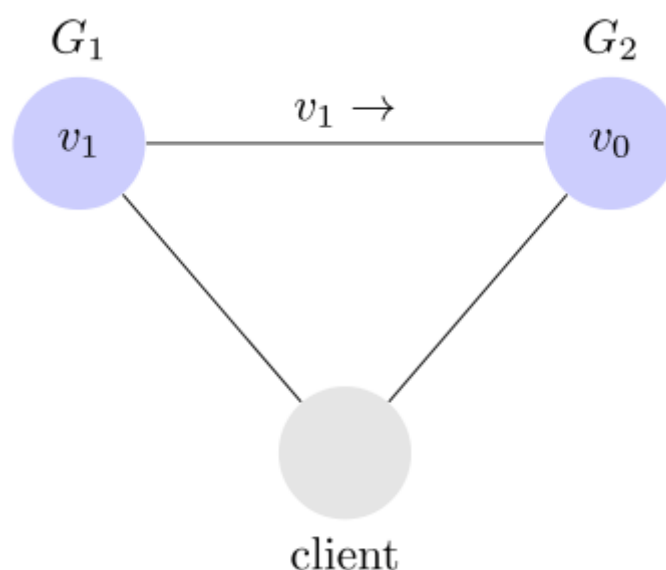
举例来说，某条记录是 v0，用户向 G1 发起一个写操作，将其改为 v1。



问题是，用户有可能向 G2 发起读操作，由于 G2 的值没有发生变化，因此返回的是 v0。G1 和 G2 读操作的结果不一致，这就不满足一致性了。



为了让 G_2 也能变为 v_1 ，就要在 G_1 写操作的时候，让 G_1 向 G_2 发送一条消息，要求 G_2 也改成 v_1 。



一致性和可用性的矛盾

一致性和可用性，为什么不可能同时成立？答案很简单，因为可能通信失败（即出现分区容错）。

如果保证 G_2 的一致性，那么 G_1 必须在写操作时，锁定 G_2 的读操作和写操作。只有数据同步后，才能重新开放读写。锁定期间， G_2 不能读写，没有可用性。

如果保证 G_2 的可用性，那么势必不能锁定 G_2 ，所以一致性不成立。

综上所述， G_2 无法同时做到一致性和可用性。系统设计时只能选择一个目标。如果追求一致性，那么无法保证所有节点的可用性；如果追求所有节点的可用性，那就没法做到一致性。

BASE理论

BASE：全称：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，来自 ebay 的架构师提出。BASE 理论是对 CAP 中一致性和可用性权衡的结果，其来源于对大型互联网分布式实践的总结，是基于 CAP 定理逐步演化而来的。其核心思想是：

既是无法做到强一致性（Strong consistency），但每个应用都可以根据自身的业务特点，采用适当的方式来使系统达到最终一致性（Eventual consistency）。

Basically Available(基本可用)

什么是基本可用呢？假设系统，出现了不可预知的故障，但还是能用，相比较正常的系统而言：

1. 响应时间上的损失：正常情况下的搜索引擎 0.5 秒即返回给用户结果，而**基本可用**的搜索引擎可以在 1 秒作用返回结果。
2. 功能上的损失：在一个电商网站上，正常情况下，用户可以顺利完成每一笔订单，但是到了大促期间，为了保护购物系统的稳定性，部分消费者可能会被引导到一个降级页面。

Soft state (软状态)

什么是软状态呢？相对于原子性而言，要求多个节点的数据副本都是一致的，这是一种“硬状态”。

软状态指的是：允许系统中的数据存在中间状态，并认为该状态不影响系统的整体可用性，即允许系统多个不同节点的数据副本存在数据延时。

Eventually consistent (最终一致性)

系统能够保证在没有其他新的更新操作的情况下，数据最终一定能够达到一致的状态，因此所有客户端对系统的数据访问最终都能够获取到最新的值。