一篇文章搞清楚什么是分布式系统 CAP 定理

本文是对 Gilbert and Lynch's specification and proof of the CAP Theorem 文章的概括版本。大部分内容参照 An Illustrated Proof of the CAP Theorem 文章的。

什么是 CAP 定理

CAP 定理是分布式系统中的基本定理，这个理论表明任何分布式系统最多可以满足以下三个属性中的两个。

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

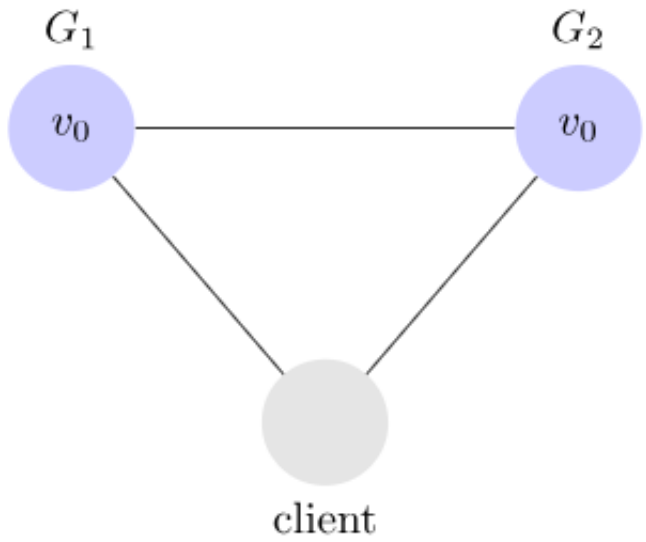
CAP 定理指出分布式系统不可能同时满足一致性，可用性和分区容忍性。听起来很简单，但一致性、可用、 分区容忍意味着什么？

在本文中，我们将介绍一个简单的分布式系统，并解释该系统可用性，一致和分区容错的含义。

什么是分布式系统

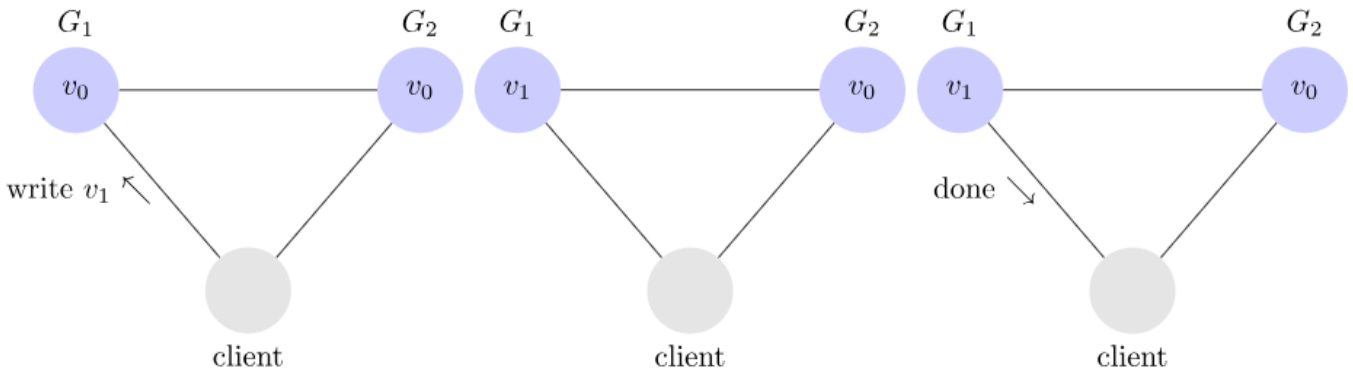
分布式系统（Distributed System）是一组电脑，通过网络相互连接传递讯息与通讯后并协调它们的行为而形成的系统。组件之间彼此进行交互以实现一个共同的目标。把需要进行大量计算的工程数据分割成小块，由多台计算机分别计算，再上传运算结果后，将结果统一合并得出数据结论的科学。

现在让我们考虑一个非常简单的分布式系统。该系统由 ₩(G_1₩) 和 ₩(G_2₩) 两个服务组成。这两个服务都追踪相同的变量 ₩(V₩)，这个变量的初始值为 ₩(v_0₩)。₩(G_1₩) 和 ₩(G_2₩) 彼此之间可以通信，并且能够和外部的客户端进行通信。下图正是我们系统的架构：



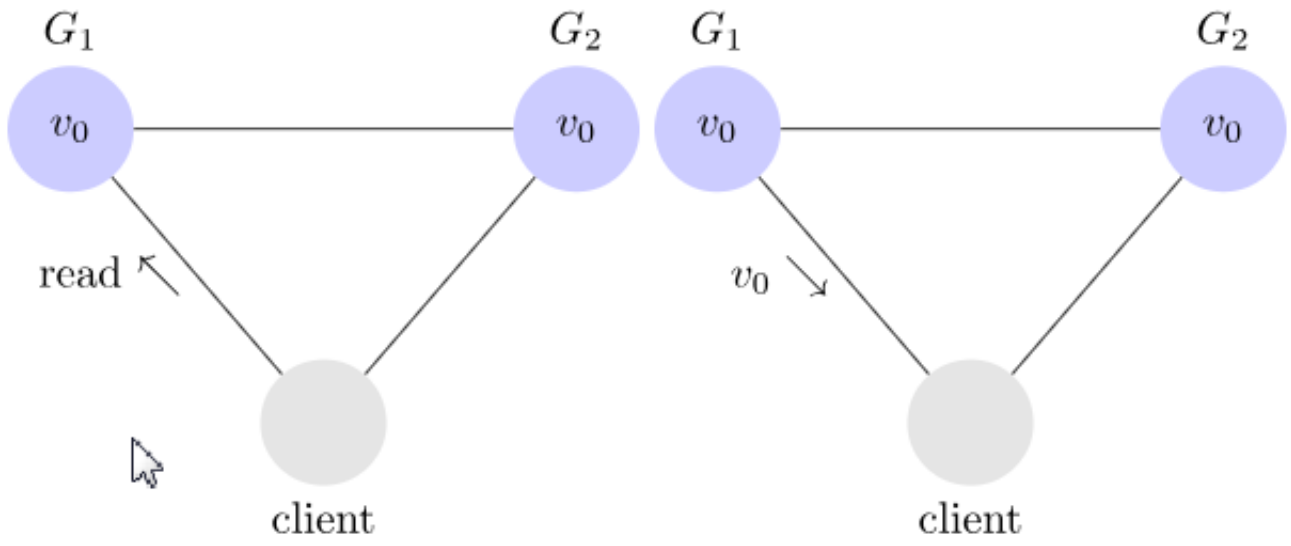
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

客户端可以向任何服务器发出读写请求。当一个服务接收到请求，它会做任何需要的计算，之后对客户端发出响应。比如下面就是一个写请求的例子：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

下面是读请求的例子：

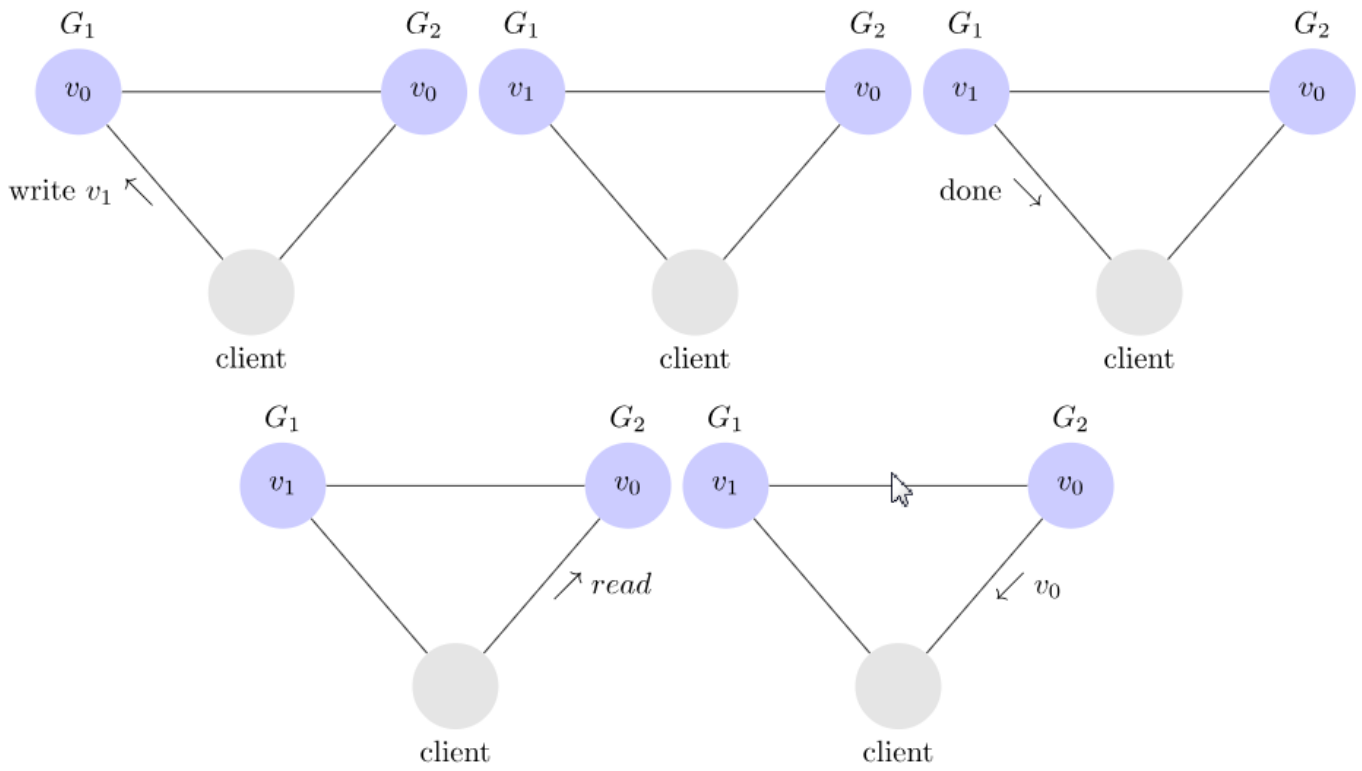


如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

现在我们有了分布式系统的基本概念，接下来的文章将进一步介绍分布式系统的可用性、一致性以及分区容错性。

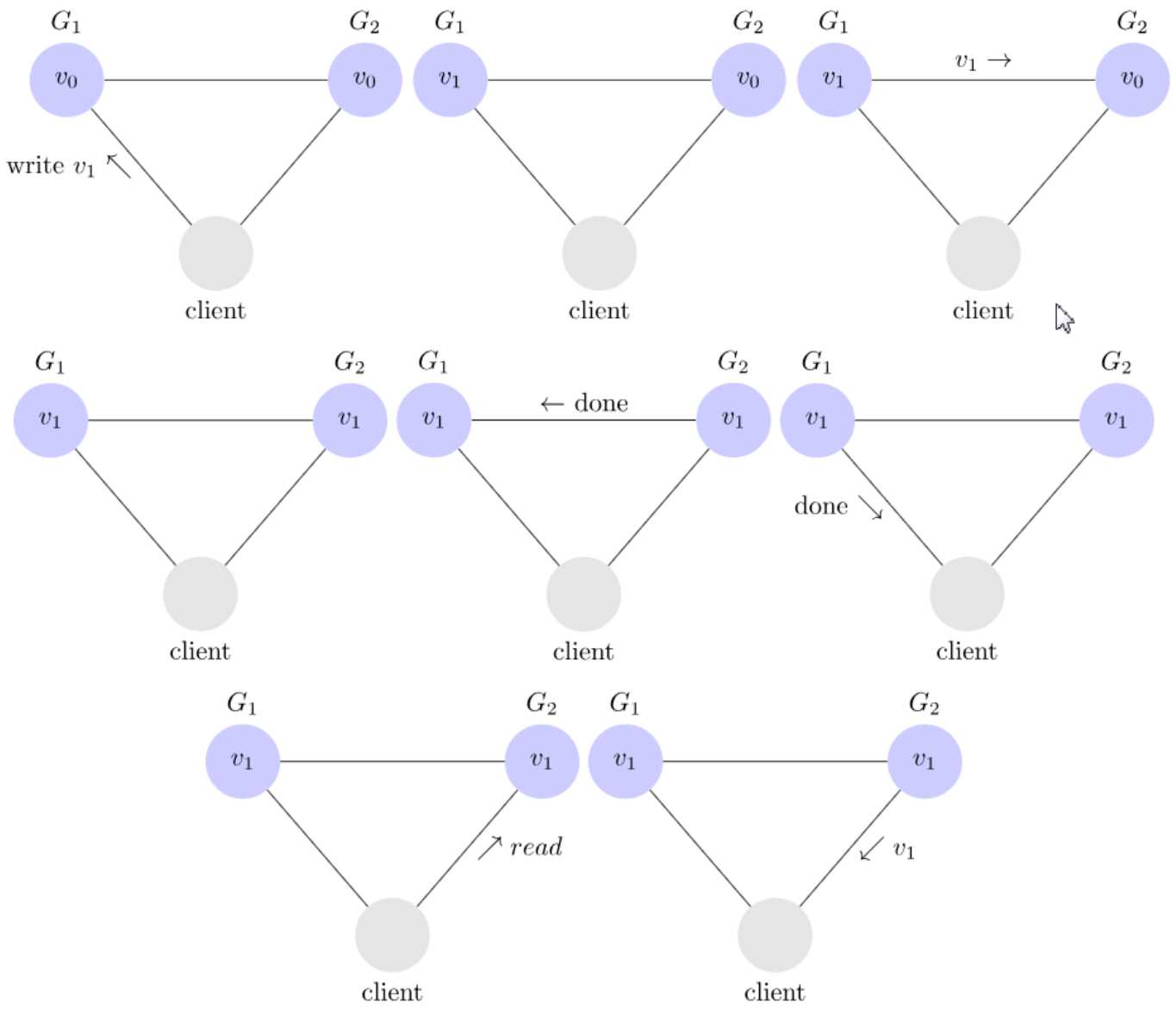
一致性

Gilbert 和 Lynch 对一致性的描述为：any read operation that begins after a write operation completes must return that value, or the result of a later write
operation
（中文意思是在写操作完成后开始的任何读操作都必须返回该值，或者后续写操作的结果）。也就是在一致的系统中，一旦客户端将值写入任何服务器并获得响应，那么后续的读客户端将从分布式系统中任何的服务器中读取到这个值。下面系统就不满足这个特点：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

客户端更新 ₩(G_1₩) 服务器上的 ₩(v₩) 为 ₩(v_1₩)，₩(G_1₩)
服务器对此做出了响应。但是客户端从 ₩(G_2₩) 获取 ₩(v₩) 的值得到的结果确是
₩(v_0₩)。下面系统就是一致性的系统：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

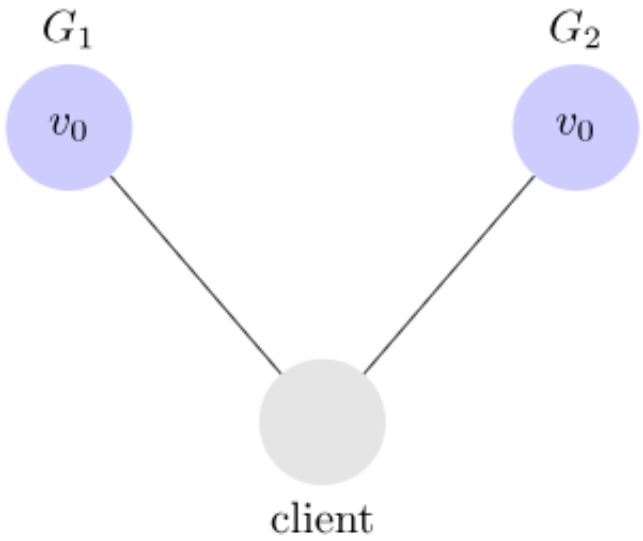
在这个系统中，₩(G_1₩) 服务器在响应客户端之前将 ₩(v₩) 的值复制到 ₩(G_2₩) 服务器上，这时候客户端从 ₩(G_2₩) 获取 ₩(v₩) 的值得到的结果是 ₩(v_1₩)。

可用性（Availability）

Gilbert 和 Lynch 对可用性的描述为：every request received by a non-failing node in the system must result in a
response
（中文意思：系统中非故障节点收到的每个请求都必须产生响应）。也就是说在可用系统中，客户端向服务器发送请求并且该服务器未崩溃，则该服务器必须最终响应客户端。

分区容错性（Partition Tolerance）

Gilbert 和 Lynch 对可用性的描述为：the network will be allowed to lose arbitrarily many messages sent from one node to
another（中文意思：允许网络丢失从一个节点发送到另一个节点的任意多个消息）。这意味着 ₩(G_1₩) 和 ₩(G_2₩) 之间的通信消息可以被丢掉，如果他们之间所有的消息都被丢弃，那么我们的系统看起来像下面一样：



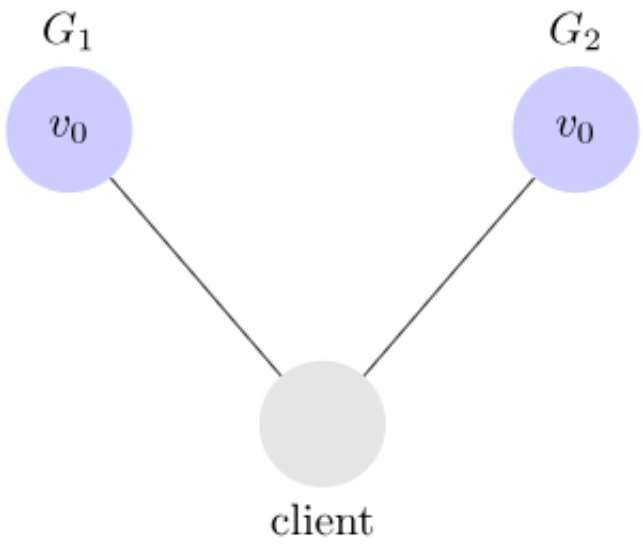
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

在分布式环境下，网络分区是一个必然的事实。所以我们的系统必须满足分区容错性，这样我们的系统才能够正常运行。

CAP 证明

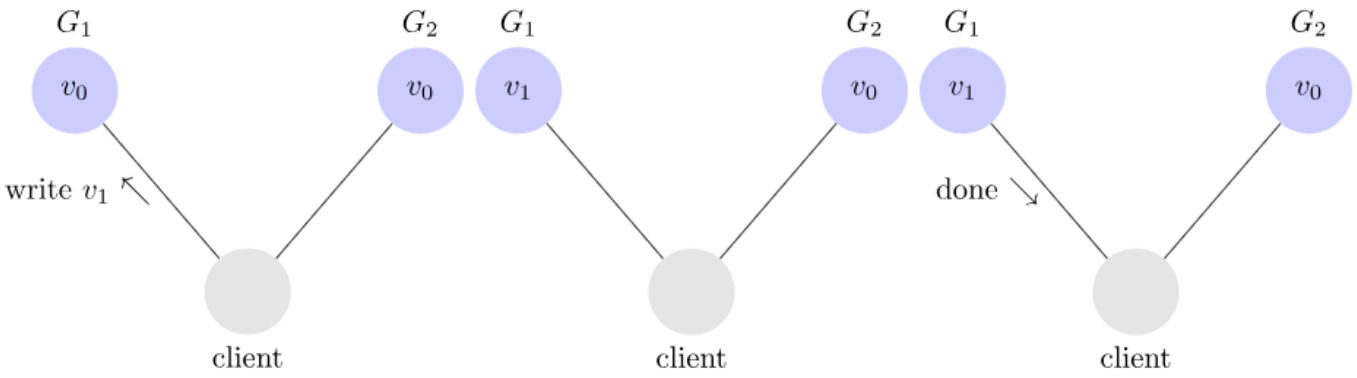
到这里我们已经明白了分布式系统的可用性、一致性以及分区容错性的含义，现在我们来证明为什么分布式系统不能同时满足这三者。

我们用反证法证明，假设现实中确实存在满足这三个条件的分布式系统，那么当系统之间的网络发生分区的时，它看起来像下面的情况：



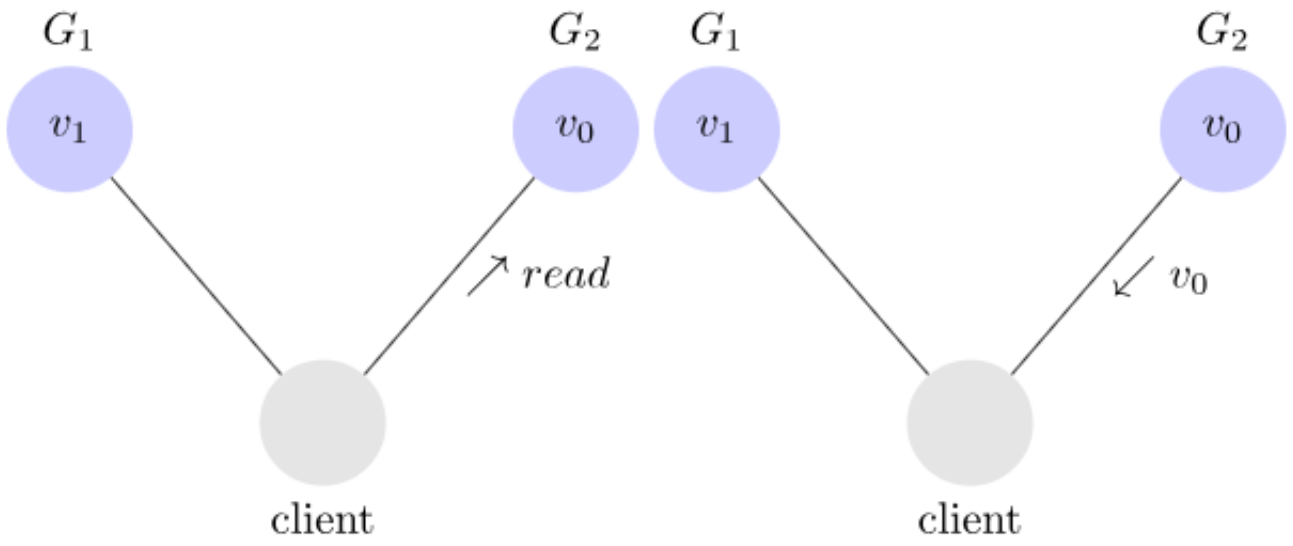
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

现在客户端 ₩(C_1₩) 更新 ₩(G_1₩) 服务器上的 ₩(v₩) 为
₩(v_1₩)，因为我们的系统是可用的，所以 ₩(G_1₩)
服务器会做出响应，但是因为网络发生了分区，₩(G_1₩) 无法将数据复制到 ₩(G_2₩)。



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

写完数据之后，另外一个客户端 ₩(C_2₩) 向 ₩(G_2₩) 服务器发出读取 ₩(v₩) 的请求，
但是因为网络分区的存在，₩(G_2₩) 服务器上 ₩(v₩) 还是更新之前的值，所以客户端
₩(C_2₩) 得到的结果为 ₩(v_0₩)。



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

这种情况下 ₩(C_2₩) 并没有获取到 ₩(C_1₩) 写入的值，也就不满足数据一致性。由此可以得出分布式系统不能同时满足可用性、一致性以及分区容错性。

CP 还是 AP

首先既然是分布式系统，那么网络分区是一定会存在的，所以分布式系统必须满足 P，否则就不是一个正真的分布式系统。所以我们必须在 A 和 C 之间做出选择。

如果分布式系统不要求强的可用性，也就是容许系统停机或者长时间无响应的话，这种情况我们就可以考虑舍弃 A。我们常见的 Zookeeper 就是满足 CP 的。

如果我们的系统可用性要求非常高，那么我们可以牺牲一致性来满足。这里说的牺牲一致性并不是说系统一直处于不一致的状态，要是这样的话这系统就没啥用了。我们说的牺牲一致性一般都是说牺牲强一致性，而保证最终一致性。也就是说系统短暂是不一致性的，过段时间能保证一致，也就是最终一致性。

所以，对于一个分布式系统来说，P 是一个基本要求，CAP 三者中，只能根据系统要求在 C 和 A 两者之间做权衡，并且要想尽办法提升 P。

关于最终一致性可以参见本博客的 BASE理论。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）