### CAP理论

## 1. CAP 的由来

要理解 CAP，首先我们要清楚，为何会有人提出 CAP？他提出 CAP 是为了解决什么问题？

时间回到 1985 年，彼时，后来证明了 CAP 理论的 Lynch 教授此时给当时的 IT 界来了一记惊雷：

她通过不可辩驳的证明告诉业界的工程师们，如果在一个不稳定（消息要么乱序要么丢了）的网络环境里（分布式异步模型），想始终保持数据一致是不可能的。

这是个什么概念呢？就是她打破了那些既想提供超高质量服务，又想提供超高性能服务的技术人员的幻想。

这本质是在告诉大家，在分布式系统里，需要妥协。

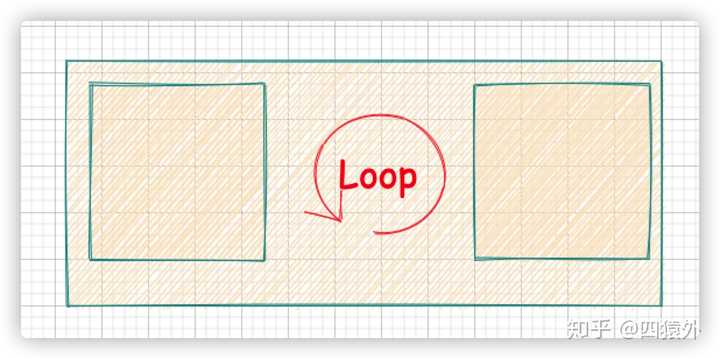
但是，如何妥协？分布式系统里到底应该怎么权衡这种 trade-off？

我们可以想象一下，在 CAP 定理提出之前，没有这些方向性的指引，在设计和实施分布式系统时该有多么混乱。一套分布式系统是由多个模块组成的，这些模块本身可能由不同的开发人员去完成。然而，对于这些人，在公共层面，竟然没有一个原则去指导他们该怎么完成这套功能。

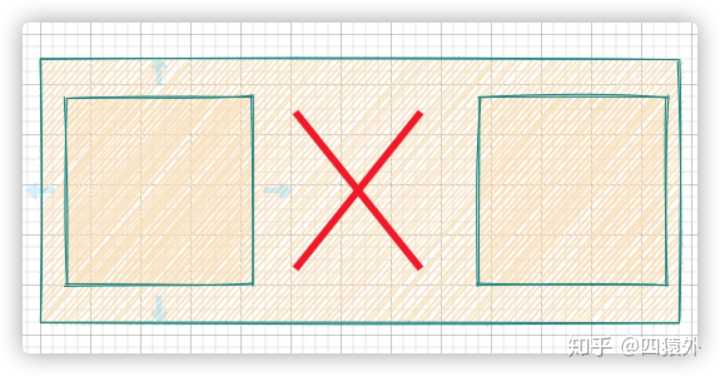
比如，我们在同步两个节点的数据时，如果发生了错误，到底我们应该怎么做呢？如果没有统一的标准和方向，那很可能在一套分布式系统中的不同模块，会出现不同的处理情况。

假设一套系统，由 A、B 两个模块构成。

A 模块的设计理念是：节点间出现了问题，它可能会选择不断的重试，一直等到节点通信恢复。



而 B 的设计理念是：节点间出现了问题，它断开就是了，可能最多就记录下状态，等以后处理。



可是，当 A、B 之间出现了通信怎么办？那会出现 A 往 B 发请求，出问题会不断重试。而 B 往 A 发请求，出问题则直接断开的情况。

当然，在后面我们会说明，CAP 的理念在实际工程中，会允许这种不一致。可是，那种不一致是提前设计好和规划好的，是根据实际数据的重要性和业务需求做的妥协，而不是这种混乱的妥协。

所以，IT 界的人们就一直在摸索，试图找到一些纲领去指导分布式系统的设计，这一找就找了 15 年。

2000 年时，Eric Brewer 教授在 PODC 会议上提出了 CAP 理论，但是由于没有被证明过，所以，当时只能被称为 CAP 猜想。这个猜想引起了巨大的反响，因为 CAP 很符合人们对设计纲领的预期。

在 2002 年后，经过 Seth Gilbert 和 Nancy Lynch 从理论上证明了 CAP 猜想后，CAP 理论正式成为了分布式系统理论的基石之一。

## 2. CAP 到底是什么

CAP 定理表达了一个分布式系统里不可能同时满足以下的三个特性：

## 2.1. C：数据一致性

什么是数据一致性？咋一看真的很让人糊涂，一致性是什么？是指数据能一起变化，是能让数据整齐划一。

那么问题又来了，数据何时会变化？数据怎么才能被称为一起变化？我们现在来回答这些问题，当我们搞清楚了这些问题，那么对数据一致性就会有了清晰的理解。

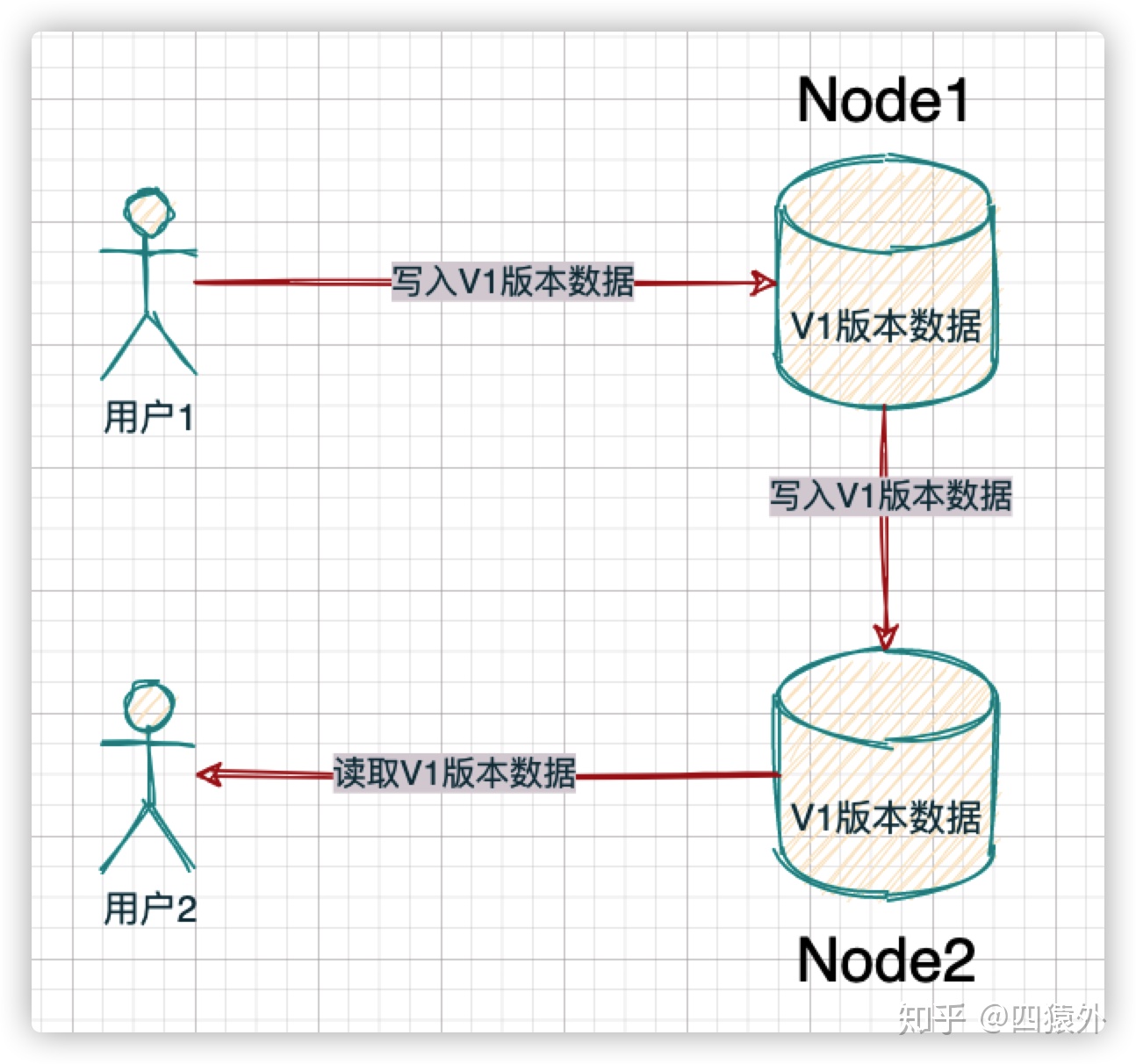
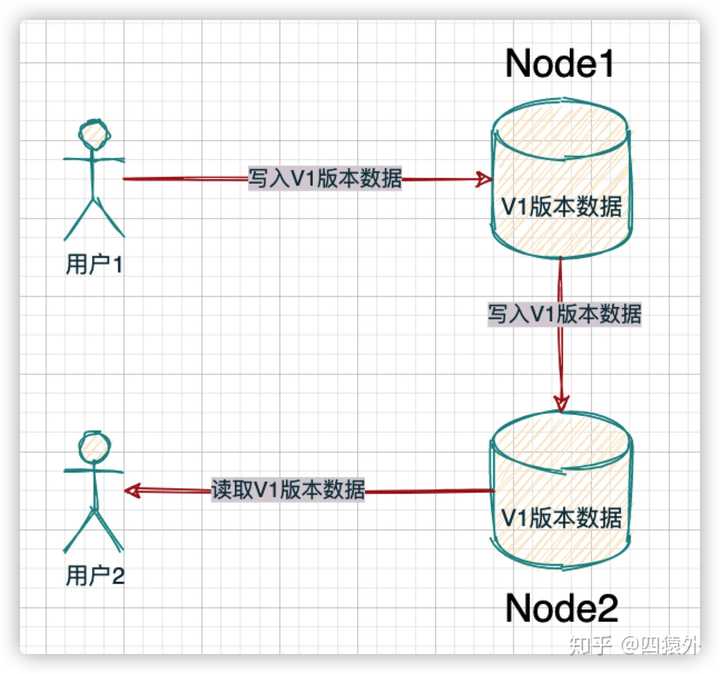
首先第一个问题，数据何时会一起变化？

答案是：仅且仅当包含数据的服务，收到数据更新请求的时候，数据才会发生变化。而数据更新请求则仅包括数据的增、删、改这三种请求，而这三种请求又被统称为写请求。所以，数据只有在写请求的时候才会发生变化。

那我们来回答第二个问题，数据要怎么样才能被称为一起变化了？即谁来判断数据是最终变化了？是服务器对写请求的返回结果吗？告诉写请求成功，数据就一定发生一致性变化了？

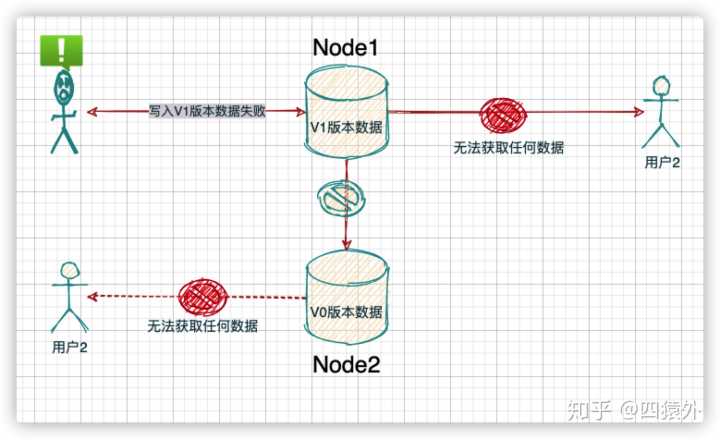
NO，数据发生变化是否一致是需要经过读请求来做检验的。那么读请求判断的依据是什么呢？

假设，我们的分布式存储系统有两个节点，每个节点都包含了一部分需要被变化的数据。如果经过一次写请求后，两个节点都发生了数据变化。然后，读请求把这些变化后的数据都读取到了，我们就把这次数据修改称为数据发生了一致性变化。



但是，这还不是完整的一致性。因为系统不可能永久的正常运行下去。

如果系统内部发生了问题从而导致系统的节点无法发生一致性变化会怎么样呢？当我们这样做的时候，就意味着想看到最新数据的读请求们，很可能会看到旧数据，或者说获取到不同版本的数据。此时，为了保证分布式系统对外的数据一致性，于是选择不返回任何数据。



这里需要注意一下，CAP 定理是在说在某种状态下的选择，和实际工程的理论是有差别的。上面描述的一致性和 ACID 事务中的一致性是两回事。事务中的一致性包含了实际工程对状态的后续处理。但是 CAP 定理并不涉及到状态的后续处理，对于这些问题，后续出现了 BASE 理论等工程结论去处理，目前，只需要明白 CAP 定理主要描述的是状态。

## 2.2. A：可用性

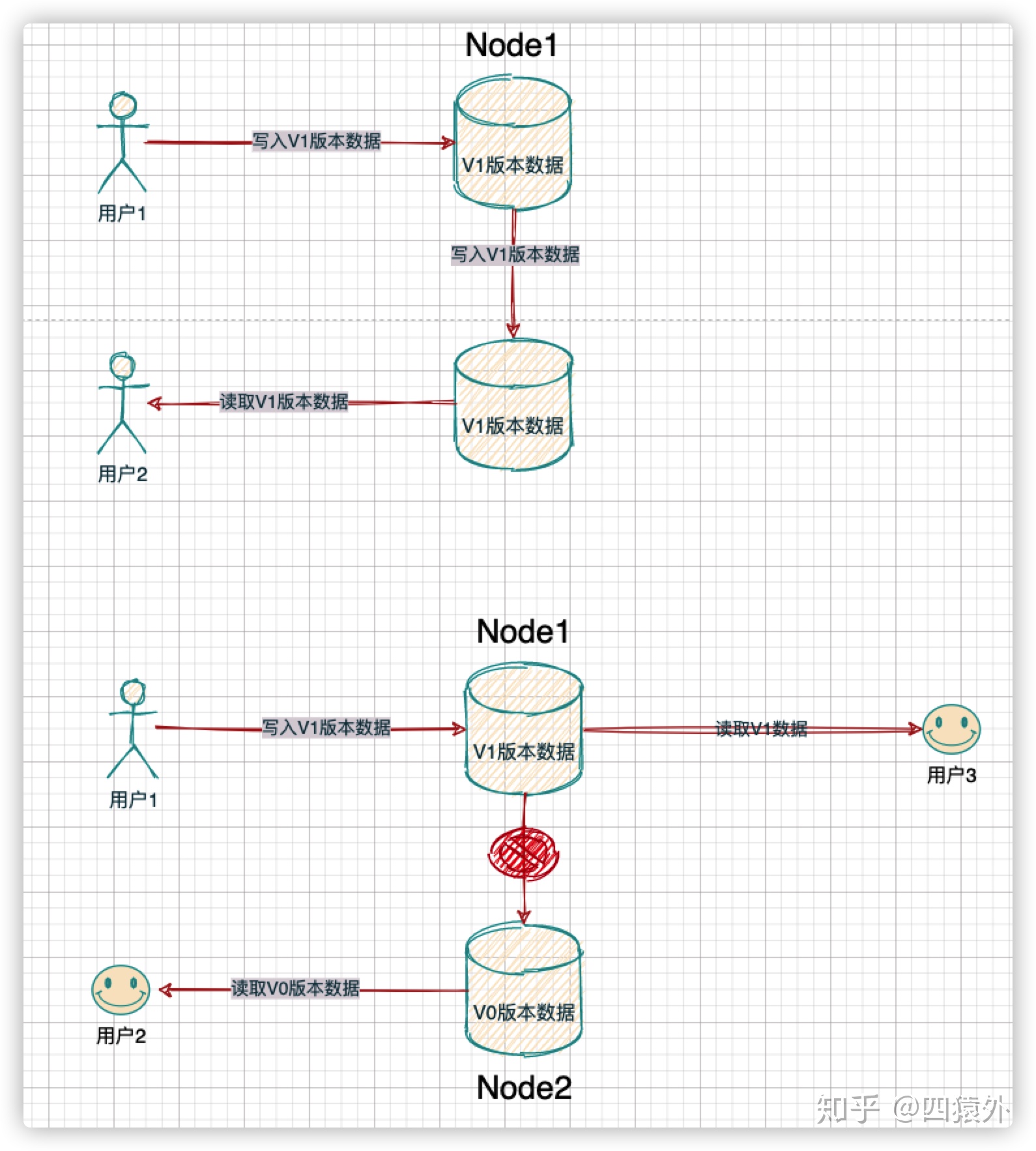
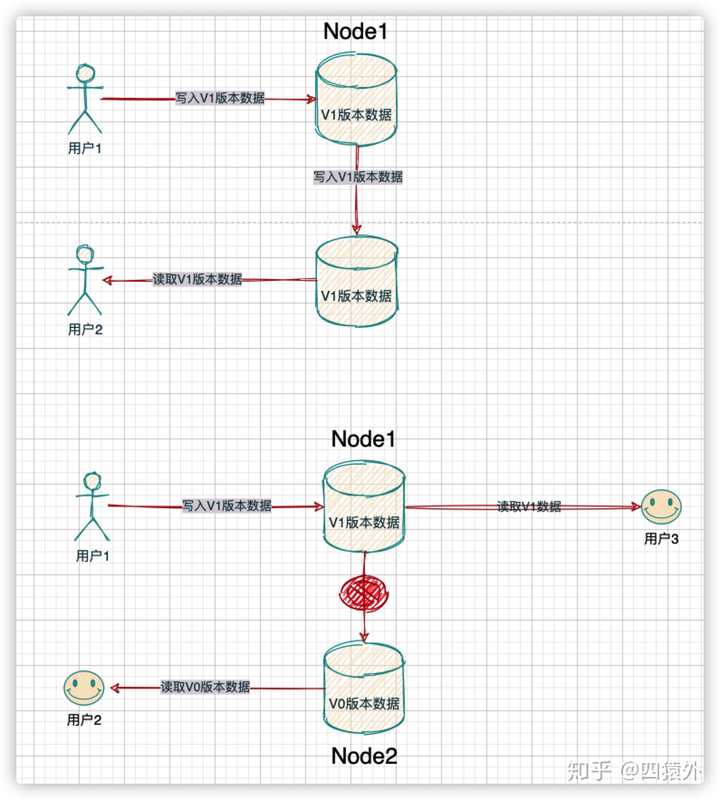
奥维德曾经说过：“行动被人们遗忘，结果却将永存”。

这句话说明了结果的重要性，而可用性在 CAP 里就是对结果的要求。它要求系统内的节点们接收到了无论是写请求还是读请求，都要能处理并给回响应结果。只是它有两点必须满足的条件：

条件 1：返回结果必须在合理的时间以内，这个合理的时间是根据业务来定的。业务说必须 100 毫秒内返回，合理的时间就是 100 毫秒，需要 1 秒内返回，那就是 1 秒，如果业务定的 100 毫秒，结果却在 1 秒才返回，那么这个系统就不满足可用性。

条件 2：需要系统内能正常接收请求的所有节点都返回结果。这包含了两重含义：

1. 如果节点不能正常接收请求了，比如宕机了，系统崩溃了，而其他节点依然能正常接收请求，那么，我们说系统依然是可用的，也就是说，部分宕机没事儿，不影响可用性指标。
2. 如果节点能正常接收请求，但是发现节点内部数据有问题，那么也必须返回结果，哪怕返回的结果是有问题的。比如，系统有两个节点，其中有一个节点数据是三天前的，另一个节点是两分钟前的，如果，一个读请求跑到了包含了三天前数据的那个节点上，抱歉，这个节点不能拒绝，必须返回这个三天前的数据，即使它可能不太合理。



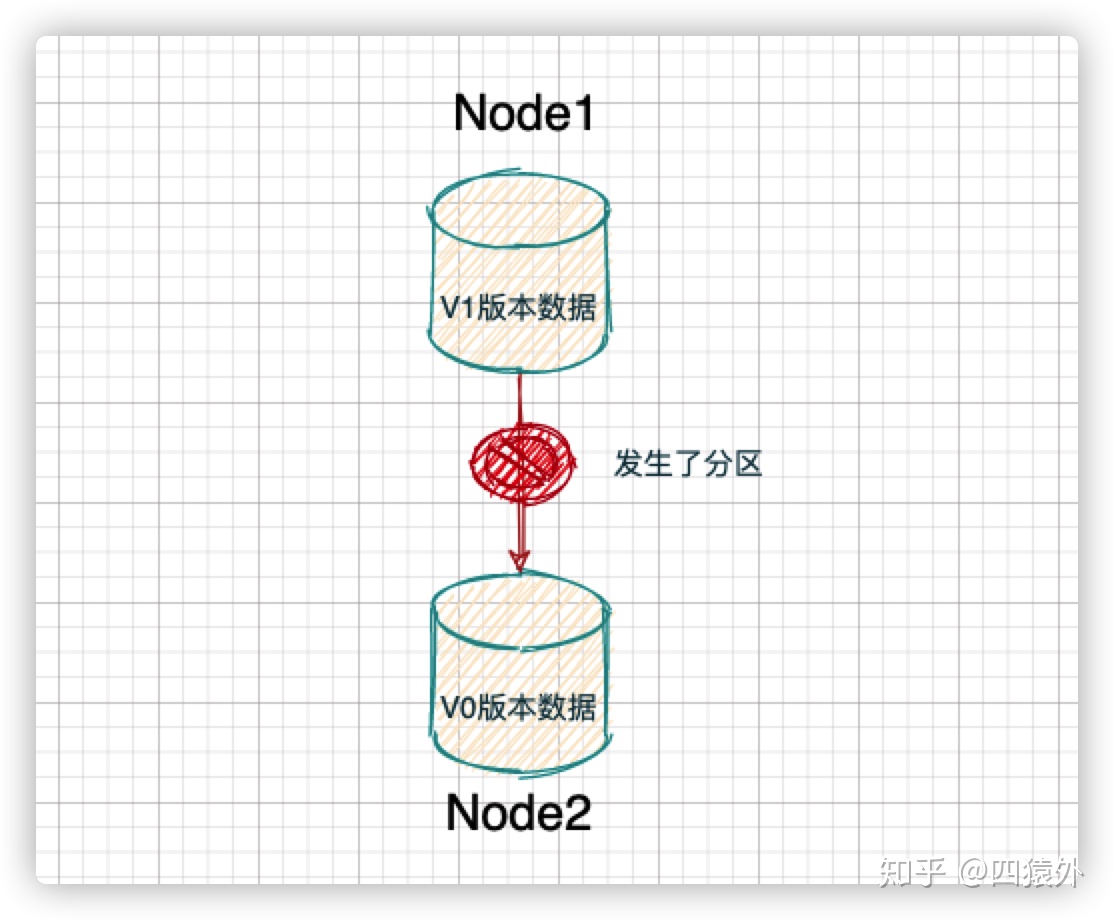
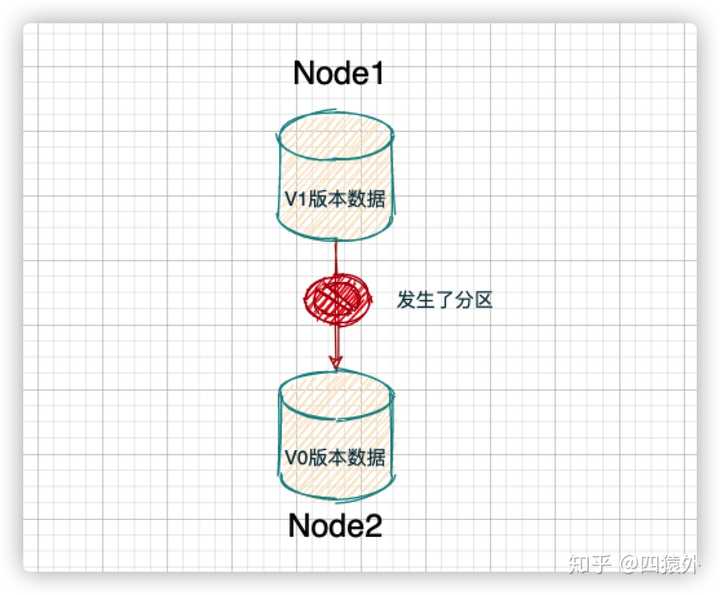
## 2.3. P：分区容忍性

分布式的存储系统会有很多的节点，这些节点都是通过网络进行通信。而网络是不可靠的，当节点和节点之间的通信出现了问题，此时，就称当前的分布式存储系统出现了分区。但是，值得一提的是，分区并不一定是由网络故障引起的，也可能是因为机器故障。

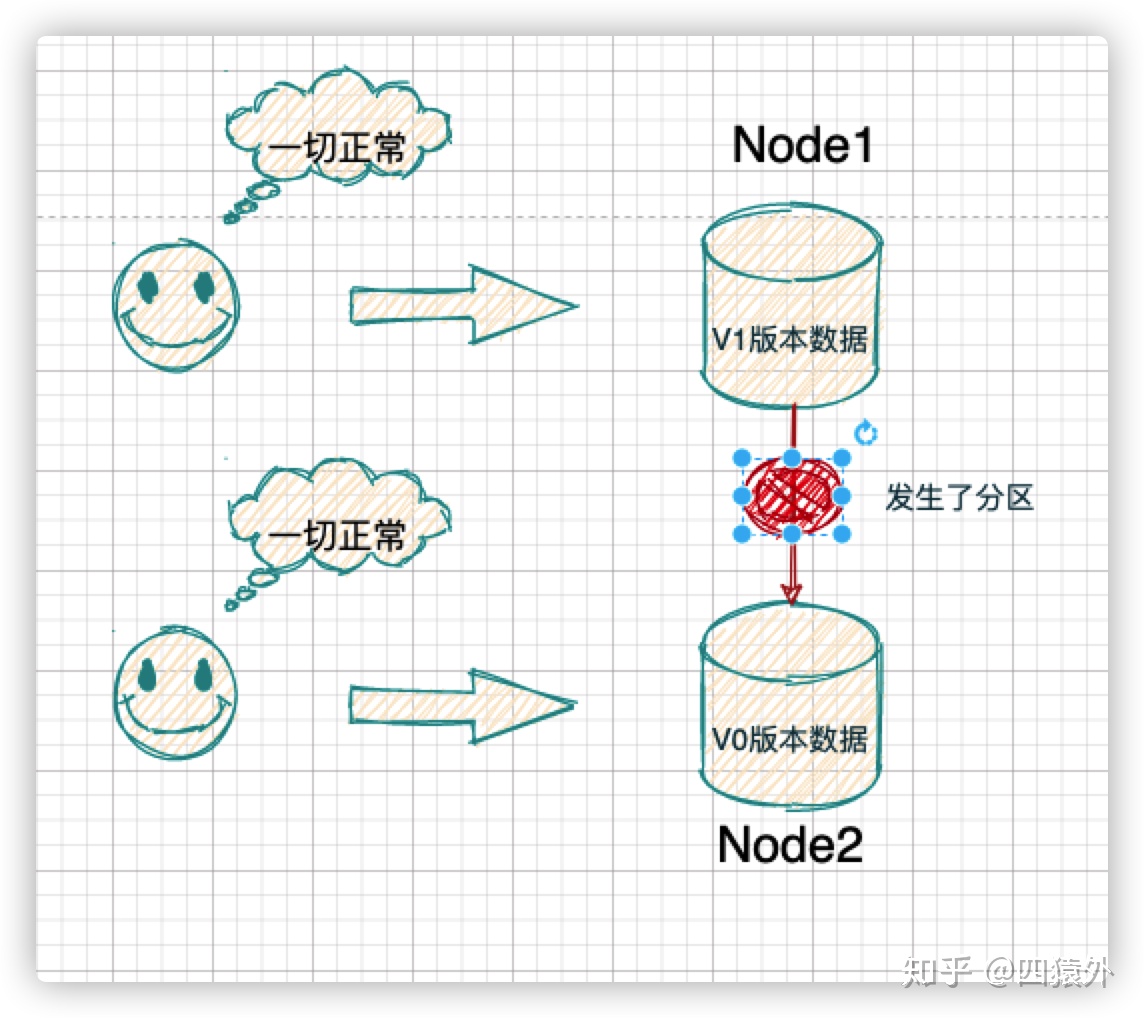
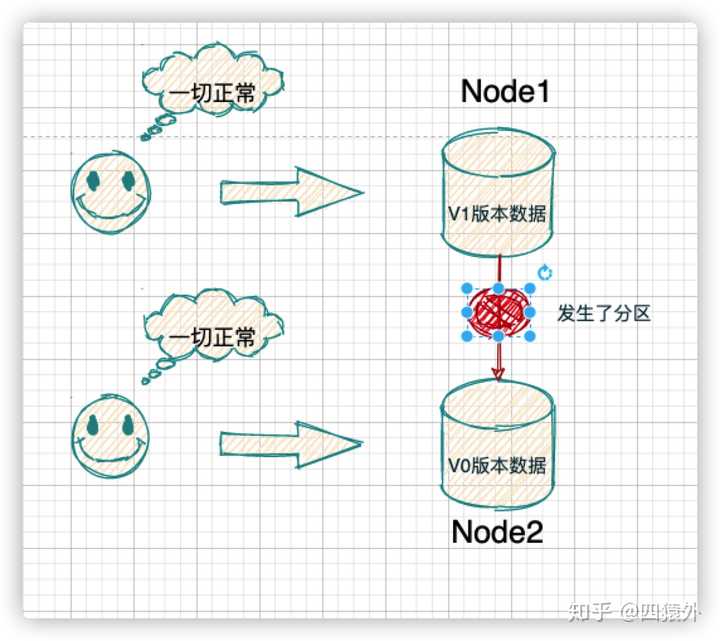
比如，我们的分布式存储系统有 A、B 两个节点。那么，当 A、B 之间由于可能路由器、交换机等底层网络设备出现了故障，A 和 B 通信出现了问题，但是 A、B 依然都在运行，都在对外提供服务。这时候，就说 A 和 B 发生了分区。

还有一种情况也会发生分区，当 A 出现了宕机，A 和 B 节点之间通信也是出现了问题，那么我们也称 A 和 B 发生了分区。

综上，我们可以知道，只要在分布式系统中，节点通信出现了问题，那么就出现了分区。



那么，分区容忍性是指什么？ 它是说，如果出现了分区问题，我们的分布式存储系统还需要继续运行。不能因为出现了分区问题，整个分布式节点全部就熄火了，罢工了，不做事情了。



## 3. CAP 怎么选择

我们上面已经知道了，在设计分布式系统时，架构师们在 C、A、P 这三种特性里，只能选择两种。

但是，这道 CAP 的选择题，就像别人在问你“小明的父亲有三个孩子，老大叫大朗，老二叫二郎，请问老三叫什么”一样。在以分布式存系统为限定条件的 CAP 世界里，P 是早已经确定的答案，P 是必须的。

因为，在分布式系统内，P 是必然的发生的，不选 P，一旦发生分区错误，整个分布式系统就完全无法使用了，这是不符合实际需要的。所以，对于分布式系统，我们只能能考虑当发生分区错误时，如何选择一致性和可用性。

而根据一致性和可用性的选择不同，开源的分布式系统往往又被分为 CP 系统和 AP 系统。

当一套系统在发生分区故障后，客户端的任何请求都被卡死或者超时，但是，系统的每个节点总是会返回一致的数据，则这套系统就是 CP 系统，经典的比如 Zookeeper。

如果一套系统发生分区故障后，客户端依然可以访问系统，但是获取的数据有的是新的数据，有的还是老数据，那么这套系统就是 AP 系统，经典的比如 Eureka。

说了这么多，其实 CAP 定理本质很简单，它就是一种分布式系统设计的不同理念概括，包括它说的一致性，可用性和分区容错性。这就类似一个大学的校训，是极度概念化的东西。

所以，大白话来形容下 CAP 吧，CAP 就是告诉程序员们当分布式系统出现内部问题了，你要做两种选择：

* 要么迁就外部服务，像外包公司。
* 要么让外部服务迁就你，像银行。

迁就外部服务就是我们不能因为我们自己的问题让外部服务的业务运行受到影响，所以要优先可用性。而让外部服务迁就我们，就要优先一致性。

## 4. 对 CAP 的常见误解

## 误解一：分布式系统因为 CAP 定理放弃了 C 或者 A 中的其中一个

很多人在没有对 CAP 做深入了解的情况下，听到很多人说分布式系统必须在 CAP 三个特性里选择两个，就觉得一套分布式系统肯定要么只有可用性要么只有一致性，不存在完整的可用性和一致性功能。

这种理解是大有问题的。因为，P 这种问题发生的概率非常低，所以：

**当没有出现分区问题的时候，系统就应该有完美的数据一致性和可用性。**

你什么时候见过一个系统，当内部没有问题的时候，会经常让外部请求卡一下的？要么就冷不丁的提供陈旧的老数据？那还能叫系统吗？

## 误解二：C 和 A 之间的选择是针对整个分布式系统的，只能整体考虑 C 和 A 之间的选择

这个理解也是不对的。当分区发生的时候，其实对一致性和可用性的抉择是局部性的，而不是针对整个系统的。

可能是在一些子系统做一些抉择，甚至很可能只需要对某个事件或者数据，做一致性和可用性的抉择而已。

比如，当我们做一套支付系统的时候，会员的财务相关像账户余额，账务流水是必须强一致性的。这时候，你就要考虑选 C。但是，会员的名字，会员的支付设置就不必考虑强一致性，可以选择可用性 A。

一套分布式系统的运行，就像人生一样，就是一次又一次的选择。在不同阶段，不同的时刻有不同的事件发生的时候，又怎么可能会有完全一样的选择呢？

## 误解三：CAP 的三个特性只有是和否两种极端选择，而不是一个范围

这种二元性的理解更是极其误导人。

CAP 理论的三种特性不是 Boolean 类型的，不是一致和不一致，可用和不可用，分区和没分区的这类二选一的选项。而是这三种特性都是范围类型。

拿可用性来说，就像我从银行取钱。当我目的是派发压岁钱的时候，我很可能就想全要新票子，但是，新票子很可能就还得多一个步骤，就是需要拿旧票子去换一些新票，此时，我可以多等会儿，能拿到新票子就好。而当我的目的就是做生活花销的时候，票子是新是旧，我根本不那么关心，快点拿到钱就行。这就是可用性的范围需求之一，对时延性的要求。

再比如，分区容错则由于探测机制的问题，可能还得各节点搞投票去协商分区是否存在，当某一台机器出现了问题，可能不影响业务的话，就会被机器投票认为分区不存在。然后一直等到多数机器出现了问题，才会投票确认出现了分区问题。这就好像新冠疫情，还会分低、中、高风险区呢，不是一出现通信故障就都被逻辑认定为分区问题。

作者：四猿外  
链接：https://www.zhihu.com/question/54105974/answer/1643846752  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 5. CAP 理论的一些疑问

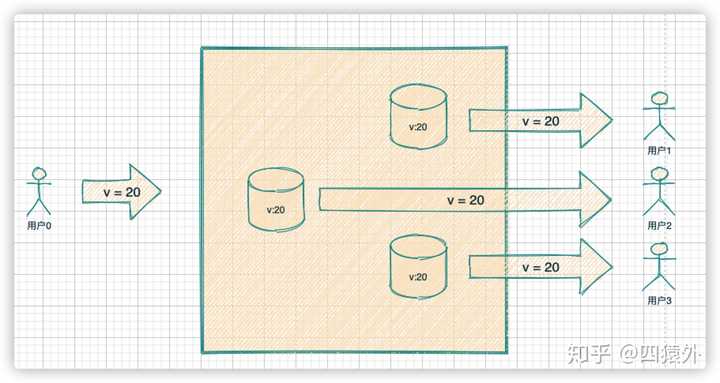
## 疑问一：在遵从 CAP 定理的系统中是否适合任意的写请求

首先，在 CAP 定理中，关于一致性会有多种说法，但是总的来说，都是在描述数据最新版本的可见性。而这些可见性往往代表的是读请求返回的数据的可见性。

那么问题来了，当我们要求读数据的可见性的时候，对写数据有什么要求吗？

比如，我们系统有三个节点，一个客户端给这个系统发了一个写请求，要求系统写入一个值为 20 的数据。那么，如果要满足 CAP 定理中的一致性，就需要在写完 20 这个数据之后，当其他客户端请求读取这个值为 20 的数据之后，无论请求被转发到系统中任何节点都能返回这个值。

这就要求写入这个值为 20 的写请求必须成功写到三个节点上，此时，系统就满足了写一致性的。所以，我们可以说对于读一致性的要求是同时约束了写一致性的。

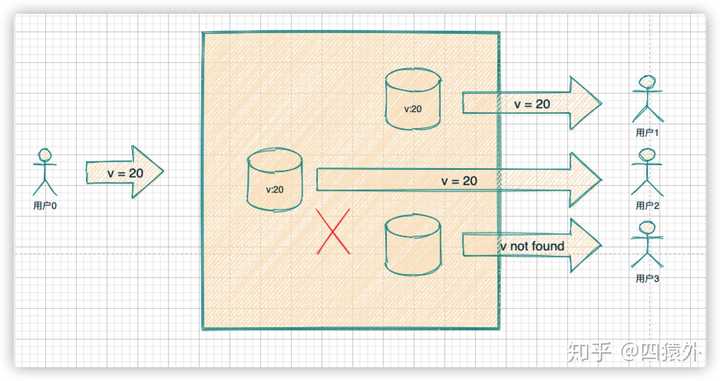


其次，在 CAP 定理中，可用性本身要求对读、写请求都要处理。如果我们以可用性作为标准的时候，在发生分区错误时，由于我们对读请求并没有强行要求返回完全准确的数据，所以，可能在本次读请求之前的最近一次写请求可能是部分失败的。

同样的例子，我们的分布式系统由三个节点组成，最近一次写请求想把值为 20 的数据写到三个节点上。但是，由于发生了分区问题，有一个节点通信故障，写请求写不过去，因此只有两个节点包含了值为 20 的数据。

此时，写请求会返回给客户端一个结果，可能会告诉客户端写入成功了，也可能告诉客户端写入部分成功。

这时候，当后续的读请求恰巧被发送到有通信故障的那个节点，系统可能只能返回一个空的结果。但是，由于系统处理和返回了读写请求，所以，系统是满足了 CAP 中的可用性的。



## 疑问二：数据分片和数据副本的分布式系统是否都遵守 CAP 定理

我们知道，在一套大规模的分布式系统里，一定是既需要把海量数据做切分，存储到不同的机器上，也需要对这些存储了数据的机器做副本备份的。

那么，如果，一个分布式系统里只有数据分片存储或者只有数据副本存储，他们都会遵守 CAP 定理吗？

答案是当数据分片时，也是要遵守 CAP 定理，但是，是种非常特殊的遵守。

当在一套分布式系统只有分片存储的时候，CAP 理论会表现成什么样？

比如，我们有个分布式系统，由三个节点 a、b、c 组成。其中节点 a 存放了 A 表的数据，b 存放了 B 表的数据，c 存放了 C 表的数据。

如果有一个业务，它的意图是想往 A 表插入一条新数据，在 B 表删除一条已有数据，在 C 表更新一条老数据，这个分布式系统该怎么处理这种业务？

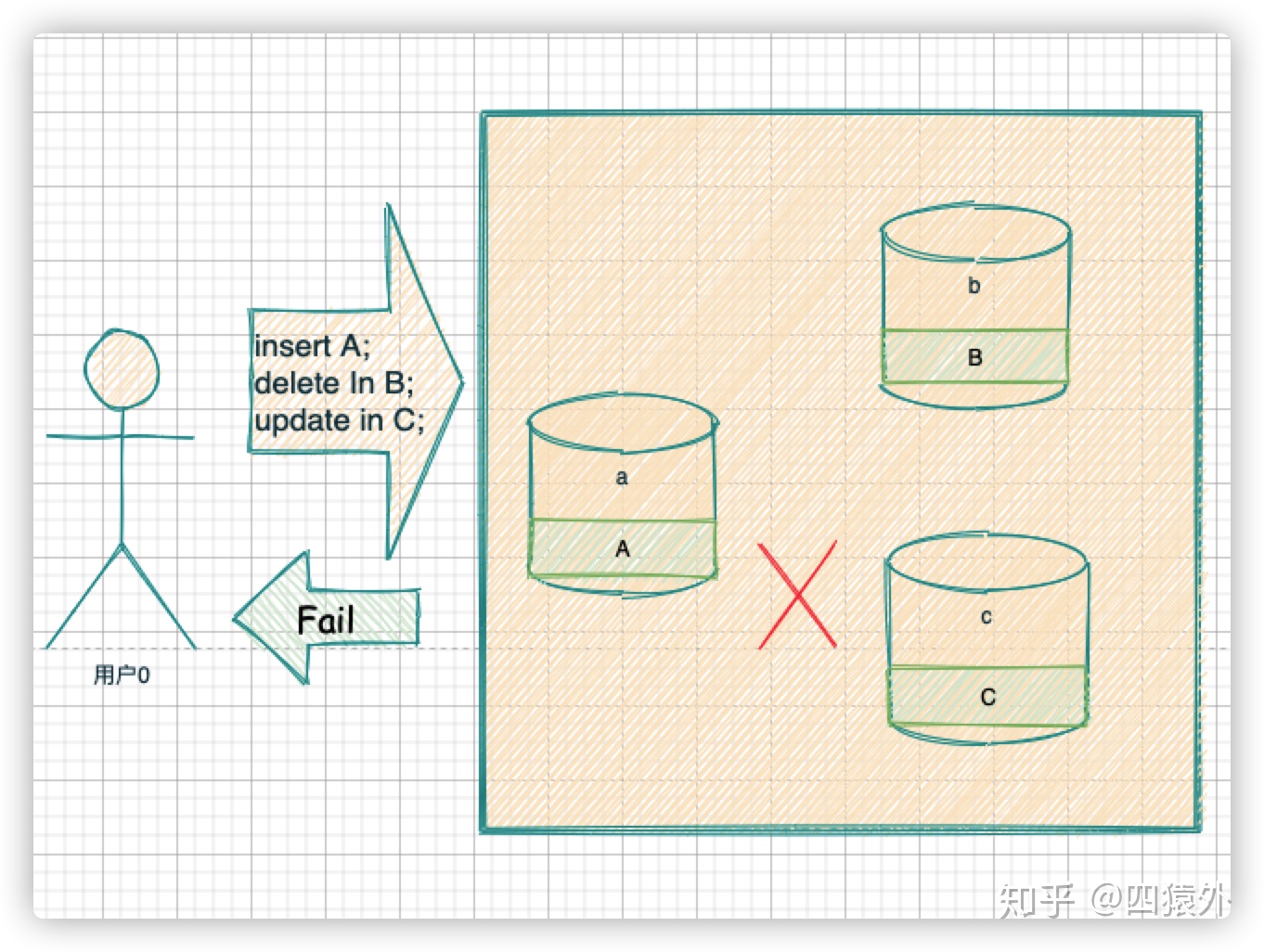
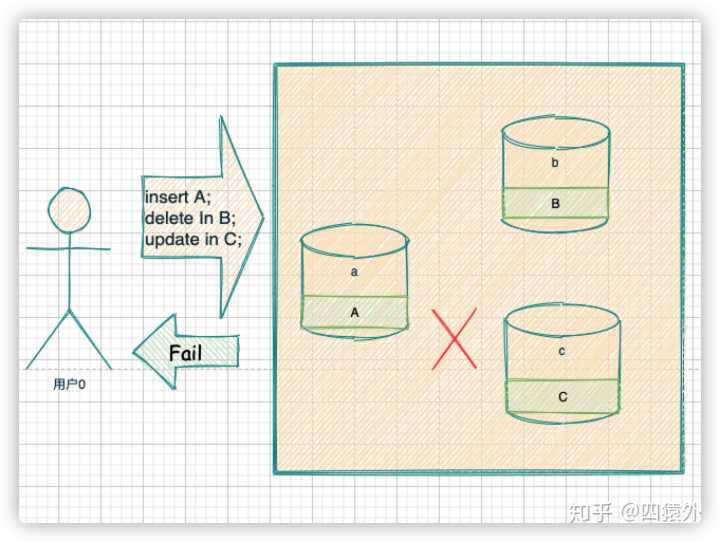
技术上我们对这种一个意图想做多件事的情况往往会包装成一个事务。当我们包装成一个事务以后，我们可能会通过先在 a 节点执行，然后去 b 节点执行，最后去 c 节点执行，等到都成功了，才会返回成功。

但是，发生了分区以后怎么办？当在 a、b 节点都成功了，到 c 发现发生了通信故障？

此时，根据 CAP 定理，你有两个选择，要么就直接返回一个部分成功的结果给客户端，要么直接卡死等客户端超时或者返回失败给客户端。当返回部分成功的时候，这就是选择了可用性（A），当卡死或者返回失败给客户端的时候，就是选择了一致性（C）。

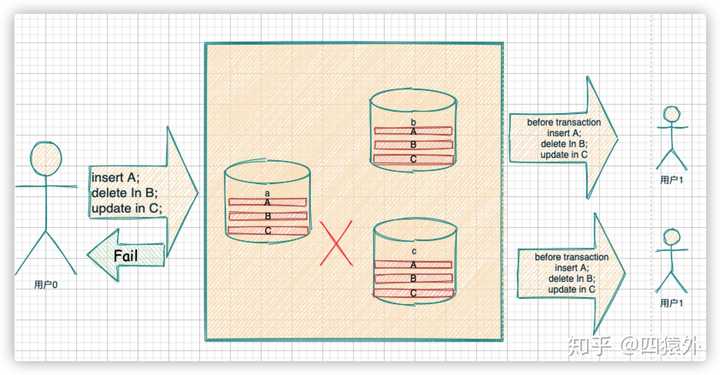
可是，我们将请求包装成了事务，而事务是要求要么都成功，要么都失败……为了遵守这种要求，对于分布式只有分片的情况，迫于客观条件，只能选择C。所以分片的分布式系统，往往都是 CP 的系统。

可选择，但是无法选择是分布式系统只有分片数据存储的情况时，遵守 CAP 定理的特殊表现。

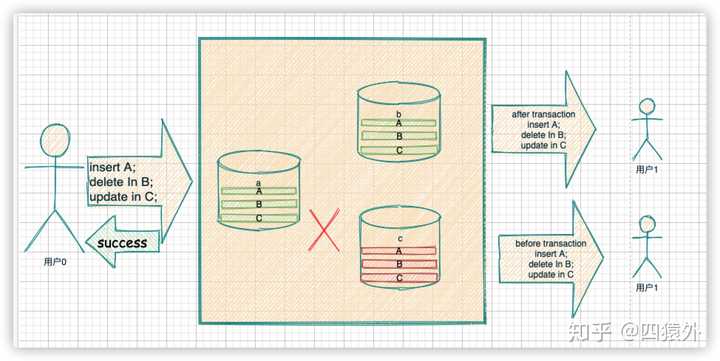


而当分布式系统是多个节点，每个节点存储了完整的一套数据，别的节点只是完整数据的备份的时候，即使事务只在一台机器上成功，当发生分区故障的时候，我们也是可以有充分的余地选择是**单机事务的回退** or **就此认为写成功的**。

单机事务的回退，就可以对外表现为选择了一致性。



就此认为写成功，则可以认为选择了可用性。



## 疑问三：为何有时候区分一个系统是 AP 还是 CP 是如此之难

因为，就像我们前面讲过的，由于 AP 或者 CP 的选择，可能仅局限为整套系统的局部，甚至某些特殊的数据上，而我们又是用这种局部的特性去描述了整套系统，所以就导致了区分的困难。而这本身其实也日渐成为了 CAP 的一个大问题，从而被人诟病。

## 6. CAP 的不足

1. CAP 定理本身是没有考虑网络延迟的问题的，它认为一致性是立即生效的，但是，要保持一致性，是需要时间成本的，这就导致往往分布式系统多选择 AP 方式
2. 由于时代的演变，CAP 定理在针对所有分布式系统的时候，出现了一些力不从心的情况，导致很多时候它自己会把以前很严谨的数学定义改成了比较松弛的业务定义，类似于我们看到，CAP 定理把一致性、可用性、分区容错都变成了一个范围属性，而这和 CAP 定理本身这种数学定理般的称呼是有冲突的，出现了不符合数学严谨定义的问题。
3. 在实践中以及后来 CAP 定理的提出者也承认，一致性和可用性并不仅仅是二选一的问题，只是一些重要性的区别，当强调一致性的时候，并不表示可用性是完全不可用的状态。比如，Zookeeper 只是在 master 出现问题的时候，才可能出现几十秒的不可用状态，而别的时候，都会以各种方式保证系统的可用性。而强调可用性的时候，也往往会采用一些技术手段，去保证数据最终是一致的。CAP 定理并没有给出这些情况的具体描述。
4. CAP 理论从工程角度来看只是一种状态的描述，它告诉大家当有错的时候，分布式系统可能处在什么状态。但是，状态是可能变化的。状态间如何转换，如何修补，如何恢复是没有提供方向的。

## 7. 引申出来的 BASE

正因为 CAP 以上的种种不足，epay 的架构师 Dan Pritchett 根据他自身在大规模分布式系统的实践经验，总结出了 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），但应用可以采用适合的方式达到最终一致性（Eventual Consitency）。

BASE 理论是实践工程的理论，它弥补了CAP 理论过于抽象的问题，也同时解决了 AP 系统的总体工程实践思想，是分布式系统的核心理论之一，我们将在下一篇文章里，详细的讲解此套理论。