09 | CAP定理：三选二，架构师必须学会的取舍

2019-05-06 蔡元楠 来自北京

《大规模数据处理实战》

你好，我是蔡元楠。

今天我要与你分享的主题是 CAP 定理。

在分布式系统的两讲中，我们一起学习到了两个重要的概念：可用性和一致性。

而今天，我想和你讲解一个与这两个概念相关，并且在设计分布式系统架构时都会讨论到的一个定理——**CAP 定理**（CAP Theorem）。

CAP 定理

CAP 这个概念最初是由埃里克·布鲁尔博士（Dr. Eric Brewer）在 2000 年的 ACM 年度学术研讨会上提出的。

如果你对这次演讲感兴趣的话，可以翻阅他那次名为“Towards Robust Distributed Systems”的演讲 deck。

在两年之后，塞思·吉尔伯特（Seth Gilbert）和麻省理工学院的南希·林奇教授（Nancy Ann Lynch）在他们的论文“Brewer’s conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services”中证明了这一概念。

Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services

Seth Gilbert* Nancy Lynch*

Abstract

When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. In this note, we prove this conjecture in the asynchronous network model, and then discuss solutions to this dilemma in the partially synchronous model.

1 Introduction

At PODC 2000, Brewer[1], in an invited talk [2], made the following conjecture: it is impossible for a web service to provide the following three guarantees:

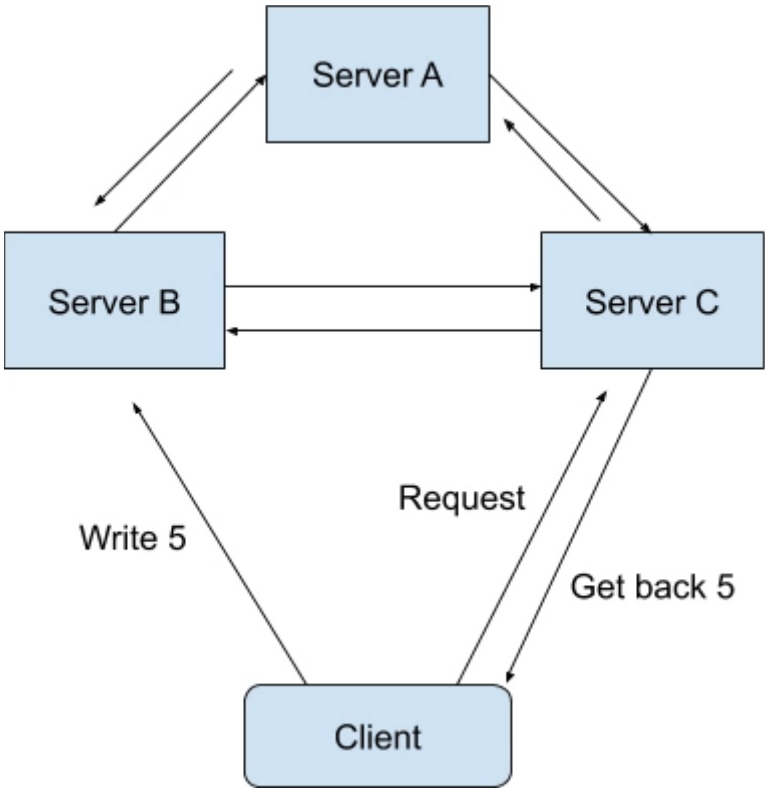
- Consistency
- Availability
- Partition-tolerance

他们在这篇论文中证明了：在任意的分布式系统中，一致性（Consistency），可用性（Availability）和分区容错性（Partition-tolerance）这三种属性最多只能同时存在两个属性。

下面，我来为你解读一下这三种属性在这篇论文里的具体意思。

C 属性：一致性

一致性在这里指的是**线性一致性（Linearizability Consistency）**。在线性一致性的保证下，所有分布式环境下的操作都像是在单机上完成的一样，也就是说图中 Sever A、B、C 的状态一直是一致的。



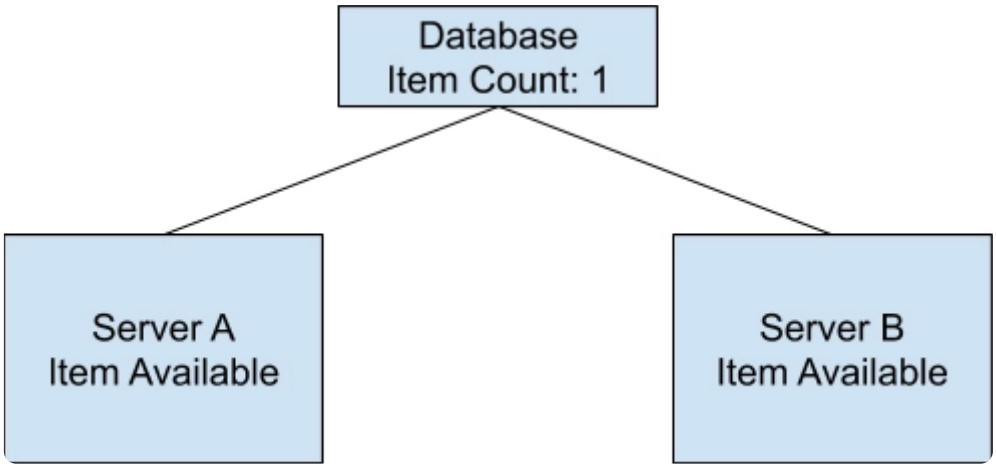
打个比方，现在有两个操作（Operation），操作 A 和操作 B，都需要在同一个分布式系统上完成。

我们假设操作 A 作用在系统上的时候，所看见的所有系统状态（State）叫作状态 A。而操作 B 作用在系统上的时候，所看见的所有系统状态叫作状态 B。

如果操作 A 是在操作 B 之前发生的，并且操作 A 成功了。那么系统状态 B 必须要比系统状态 A 更加新。

可能光看理论的话你还是会觉得这个概念有点模糊，那下面我就以一个具体例子来说明吧。

假设我们设计了一个分布式的购物系统，在这个系统中，商品的存货状态分别保存在服务器 A 和服务器 B 中。我们把存货状态定义为“有货状态”或者“无货状态”。在最开始的时候，服务器 A 和服务器 B 都会显示商品为有货状态。

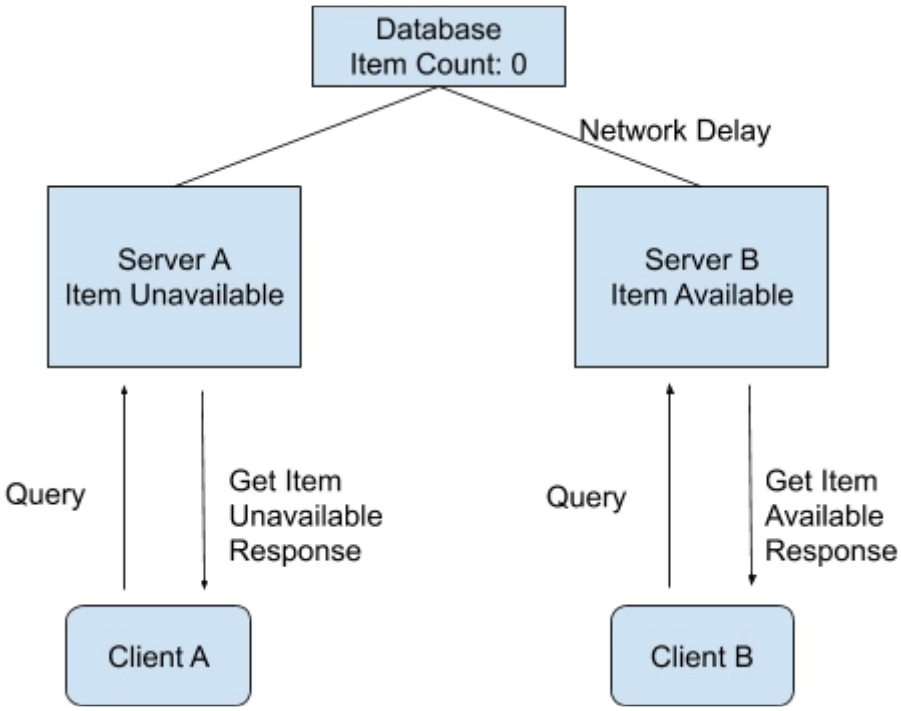


等一段时间过后，商品卖完了，后台就必须将这两台服务器上的商品状态更新为无货状态。

因为是在分布式的环境下，商品状态的更新在服务器 A 上完成了，显示为无货状态。而服务器 B 的状态因为网络延迟的原因更新还未完成，还是显示着有货状态。

这时，恰好有两个用户使用着这个购物系统，先后发送了一个查询操作（Query Operation）到后台服务器中查询商品状态。

我们假设是用户 A 先查询的，这个查询操作 A 被发送到了服务器 A 上面，并且成功返回了商品是无货状态的。用户 B 在随后也对同一商品进行查询，而这个查询操作 B 被发送到了服务器 B 上面，并且成功返回了商品是有货状态的。



我们知道，对于整个系统来说，商品的系统状态应该为无货状态。而操作 A 又是在操作 B 之前发送并且成功完成的，所以如果这个系统有线性一致性这个属性的话，操作 B 所看到的系统状态理论上应该是无货状态。

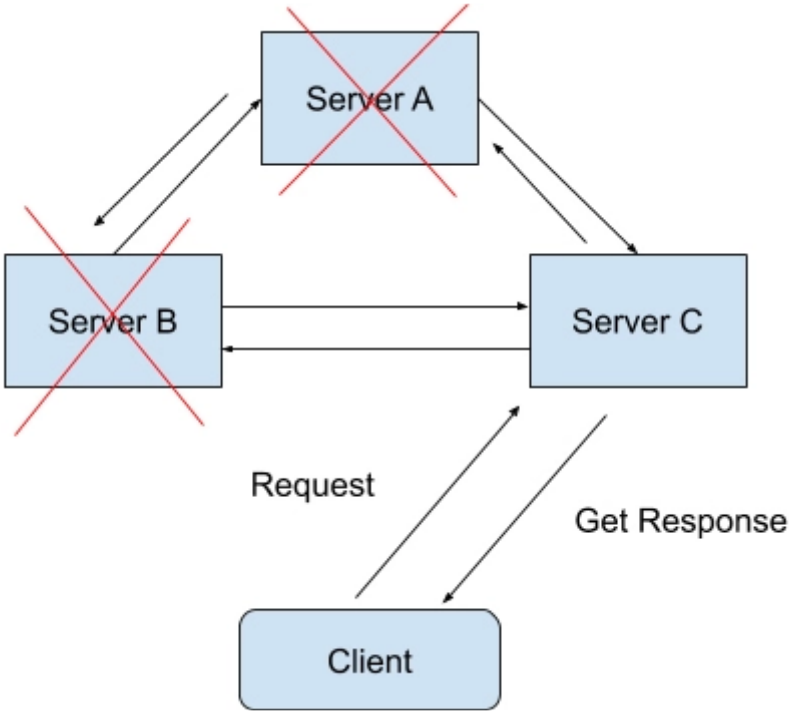
但在我们这个例子中，操作 B 却返回了有货状态。所以我们说，这个分布式的购物系统并不满足论文里所讲到的线性一致性。

聊完了一致性，我们一起来看看可用性的含义。

A 属性：可用性

可用性的概念比较简单，在这里指的是**在分布式系统中，任意非故障的服务器都必须对客户的请求产生响应**。

当系统满足可用性的时候，不管出现什么状况（除非所有的服务器全部崩溃），都能返回消息。

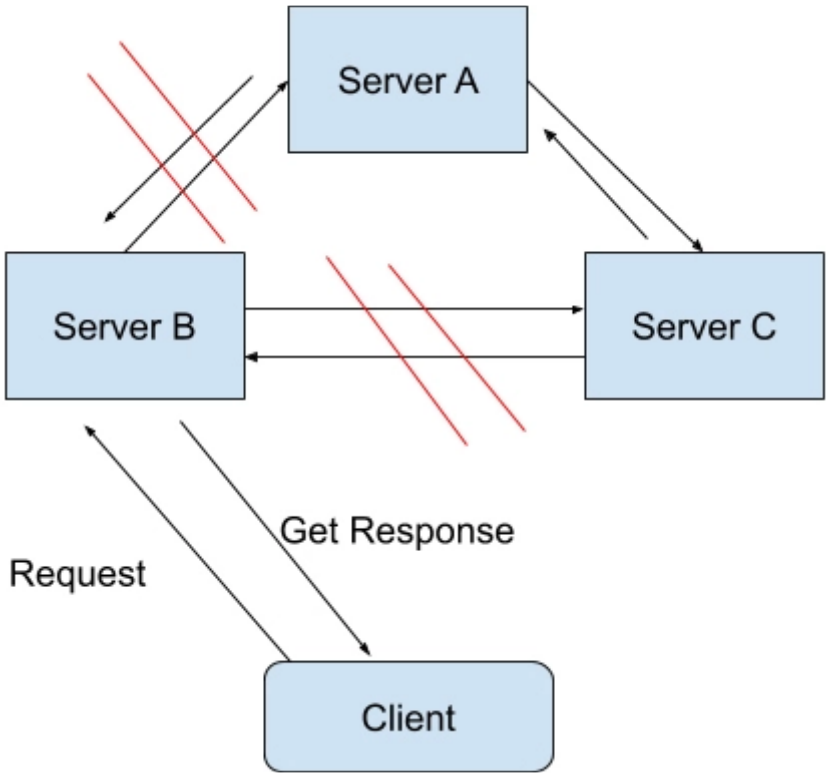


也就是说，当客户端向系统发送请求，只要系统背后的服务器有一台还未崩溃，那么这个未崩溃的服务器必须最终响应客户端。

P 属性：分区容错性

在了解了可用性之后，你还需要了解分区容错性。它分为两个部分，“分区”和“容错”。

在一个分布式系统里，如果出现一些故障，可能会使得部分节点之间无法连通。由于这些故障节点无法联通，造成整个网络就会被分成几块区域，从而使数据分散在这些无法连通的区域中的情况，你可以认为这就是发生了分区错误。



如图所示，如果你要的数据只在 Sever A 中保存，当系统出现分区错误，在不能直接连接 Sever A 时，你是无法获取数据的。我们要“分区容错”，意思是即使出现这样的“错误”，系统也需要能“容忍”。也就是说，就算错误出现，系统也必须能够返回消息。

分区容错性，在这里指的是我们的**系统允许网络丢失从一个节点发送到另一个节点的任意多条消息**。

我们知道，在现代网络通信中，节点出现故障或者网络出现丢包这样的情况是时常会发生的。

如果没有了分区容错性，也就是说系统不允许这些节点间的通讯出现任何错误的话，那我们日常所用到的很多系统就不能再继续工作了。

所以在**大部分情况下，系统设计都会保留 P 属性，而在 C 和 A 中二选一**。

论文中论证了在任意系统中，我们最多可以保留 CAP 属性中的两种，也就是 CP 或者 AP 或者 CA。关于具体的论证过程，如果你感兴趣的话，可以自行翻阅论文查看。

你可能会问，在我们平常所用到的开发架构中，有哪些系统是属于 CP 系统，有哪些是 AP 系统又有哪些是 CA 系统呢？我来给你介绍一下：

CP 系统：Google BigTable, Hbase, MongoDB, Redis, MemCacheDB，这些存储架构都是放弃了高可用性（High Availablity）而选择 CP 属性的。

AP 系统：Amazon Dynamo 系统以及它的衍生存储系统 Apache Cassandra 和 Voldemort 都是属于 AP 系统

CA 系统：Apache Kafka 是一个比较典型的 CA 系统。

我在上面说过，P 属性在现代网络时代中基本上是属于一个必选项，那为什么 Apache Kafka 会放弃 P 选择 CA 属性呢？我来给你解释一下它的架构思想。

放弃了 P 属性的 Kafka Replication

在 Kafka 发布了 0.8 版本之后，Kafka 系统引入了 Replication 的概念。Kafka Relocation 通过将数据复制到不同的节点上，从而增强了数据在系统中的持久性（Durability）和可用性（Availability）。在 Kafka Replication 的系统设计中，所有的数据日志存储是设计在同一个数据中心（Data Center）里面的，也就是说，在同一个数据中心里网络分区出现的可能性是十分之小的。

它的具体架构是这样的，在 Kafka 数据副本（Data Replication）的设计中，先通过 Zookeeper 选举出一个领导者节点（Leader）。这个领导者节点负责维护一组被称作同步数据副本（In-sync-replica）的节点，所有的数据写入都必须在这个领导者节点中记录。

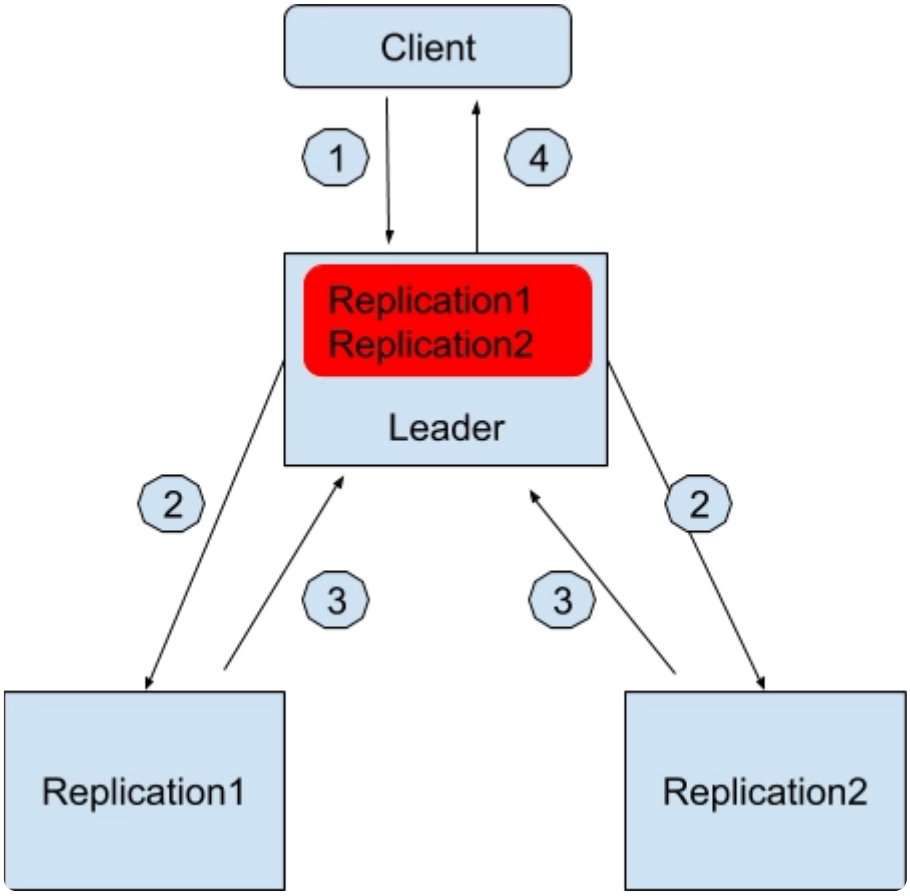
我来举个例子，假设现在数据中心有三台服务器，一台被选为作为领导者节点，另外两台服务器用来保存数据副本，分别是 Replication1 和 Replication2，它们两个节点就是被领导者节点维护的同步数据副本了。领导者节点知道它维护着两个同步数据副本。

如果用户想写入一个数据，假设是“Geekbang”

1. 用户会发请求到领导者节点中想写入“Geekbang”。

2. 领导者节点收到请求后先在本地保存好，然后也同时发消息通知 Replication1 和 Replication2。
3. Replication1 和 Replication2 收到消息后也保存好这条消息并且回复领导者节点写入成功。
4. 领导者节点记录副本 1 和副本 2 都是健康（Healthy）的，并且回复用户写入成功。

红色的部分是领导者节点本地日志，记录着有哪些同步数据副本是健康的。



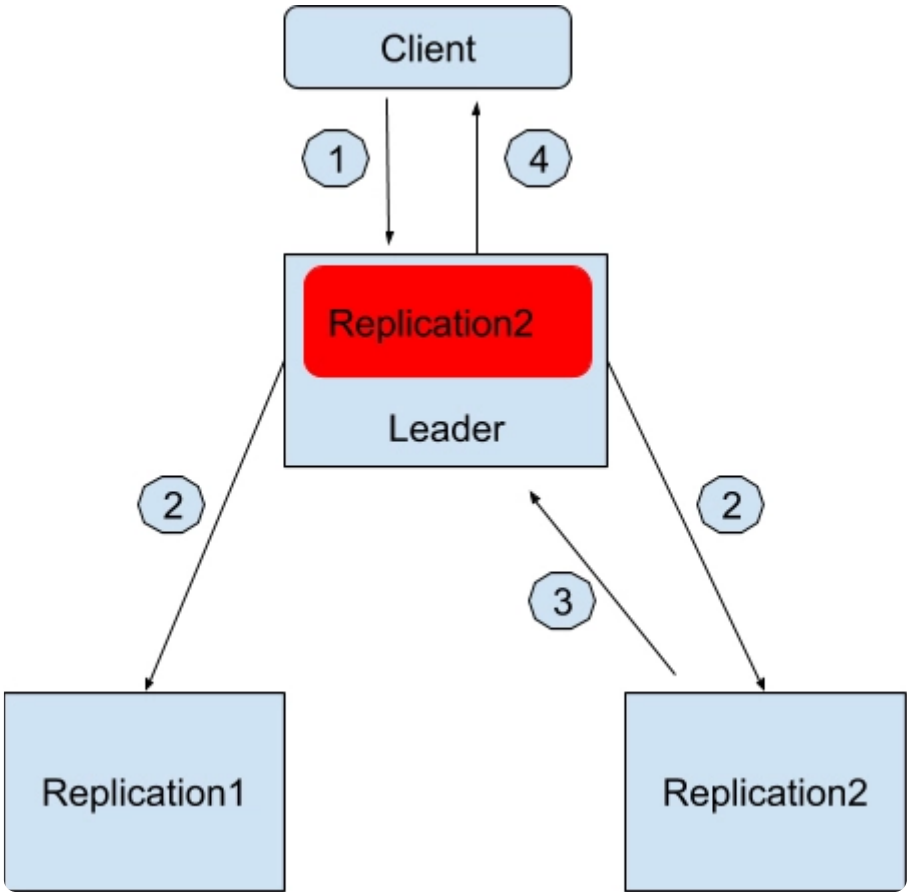
往后用户如果想查询写入的数据，无论是领导者节点还是两个副本都可以返回正确同步的结果。

那假如分区出现了该怎么办呢？例如领导者节点和副本 1 无法通讯了，这个时候流程就变成这样了。

1. 用户会发请求到领导者节点中想写入“Geekbang”。

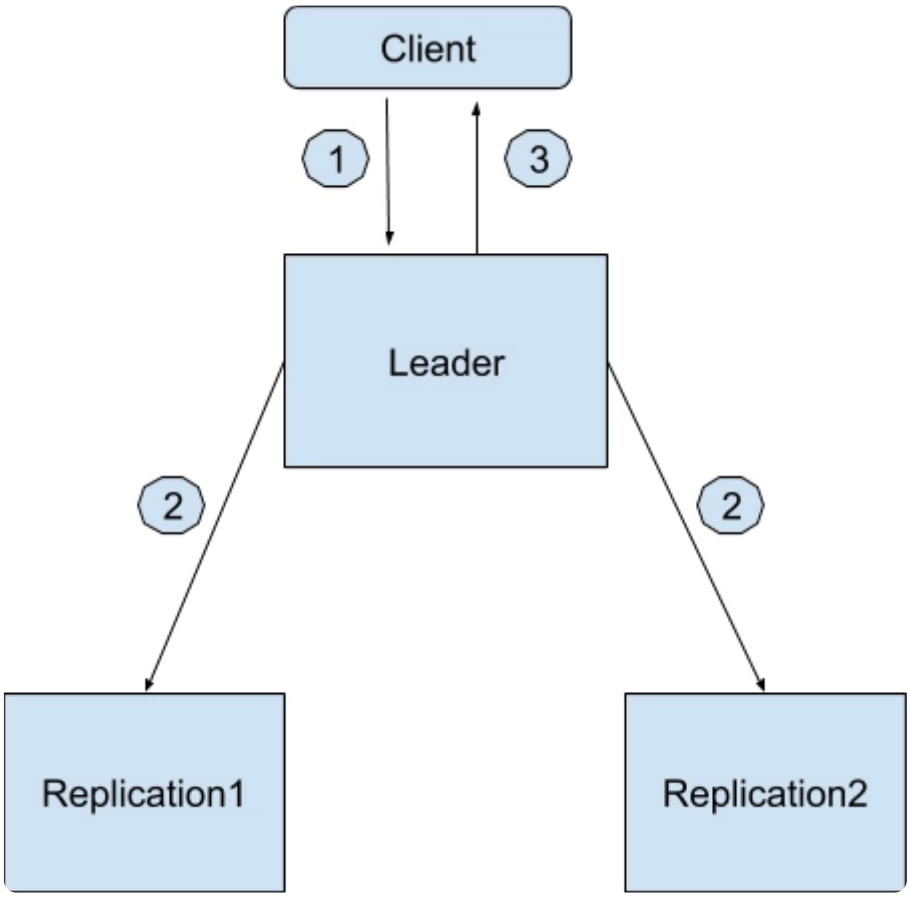
2. 领导者节点收到请求后先在本地保存好，然后也同时发消息通知 Replication1 和 Replication2。
3. 只有 Replication2 收到消息后也保存好这条消息并且回复领导者节点写入成功。
4. 领导者节点记录副本 2 是健康的，并且回复用户写入成功。

同样，红色的部分是领导者节点本地日志，记录着有哪些同步数据副本是健康的。



如果所有副本都无法通讯的时候，Apache Kafka 允许系统只有一个节点工作，也就是领导者节点。这个时候所有的写入都只保存在领导者节点了。过程如下，

1. 用户会发请求到领导者节点中想写入“Geekbang”。
2. 领导者节点收到请求后先在本地保存好，然后也同时发消息通知 Replication1 和 Replication2。
3. 没有任何副本回复领导者节点写入成功，领导者节点记录无副本是健康的，并且回复用户写入成功。



当然，在最坏的情况下，连领导者节点也挂了，Zookeeper 会重新去寻找健康的服务器节点来当选新的领导者节点。

小结

通过今天的学习，我们知道在 CAP 定理中，一致性，可用性和分区容错性这三个属性最多只能选择两种属性保留。CAP 定理在经过了差不多 20 年的讨论与演化之后，大家对这三个属性可能会有着自己的一些定义。

例如在讨论一致性的时候，有的系统宣称自己是拥有 C 属性，也就拥有一致性的，但是这个一致性并不是论文里所讨论到的线性一致性。

在我看来，作为大规模数据处理的架构师，我们应该熟知自己的系统到底应该保留 CAP 中的哪两项属性，同时也需要熟知，自己所应用到的平台架构是保留着哪两项属性。

思考题

如果让你重新设计微博系统中的发微博功能，你会选择 CAP 的哪两个属性呢？为什么呢？

欢迎你把答案写在留言区，与我和其他同学一起讨论。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。

精选留言 (76)

常超 置顶

2019-05-06

关于大家出现很多疑问的Kafka是否有P属性的问题，上网搜了一下，找到一个可以说明Kafka不具备P属性的例子，不知道理解对不对，请老师和大家批评指正。

比如：在以下的场景序列下，会出现数据写入丢失的情况，所以不能说kafka是有P属性

前提：leader和两个slave 1、2
序列：
1.机器leader跟两个slave之间发生partion
这是leader成为唯一服务节点，继续接受写入请求w1,但是w1只保存在了leader机器上，无法复制到slave1和2
2.leader和zookeeper之间发生partition，导致kafka选取slave 1为新leader，新leader接受新写入w1
这时候，上面1的w1写入丢失了。即使之后旧leader复活，w1的写入数据也不会被恢复出来。
参考：https://bit.ly/2VGKtu1

> 作者回复: 常超您好，又看到了您的留言！其实归根结底P属性还是说当Network Partition发生后，无论后面网络分区是否会恢复，分离出来的子系统都可以正常运行。您所说这几个例子确实是讲到了在Kafka Replication中，有的节点在分区后就无法再使用了，所以设计的时候并没有考虑P属性。另外还多加一句，并不是Kafka没有P属性，而是Intra-cluster Kafka Replication没有P属性。
>
> 谢谢你的参考资料，这对我和其他读者们来说都非常有帮助！


27

王燊 ون شن 置顶

2019-05-10

我的理解：A是机器挂掉仍可用，P是网络挂掉仍可用。如果我的理解正确，那老师您说Kafka不支持P的解释，即当Kafka leader挂掉整个系统不可用，其实是不是表明Kafka是不支持A，而不是不支持P的？

作者回复: 谢谢你的留言！你的理解非常正确，A就是指集群中即便挂掉几个机器但是集群对外还是正常运行的，P就是指即便机器间无法通讯了但是集群对外还是正常运行。

我对于Kafka Replication pick CA这种设计的理解是，设计师只考虑了在Replication的集群里对CA的保证而放弃了对P的保证。所以当发生Network Partition的时候，系统有可能可以工作，有可能不能继续工作。例如说C的保证是因为Kafka Replication要求了领导者节点的数据一定要同步数据副本节点上，否则不会返回这个数据；A的保证是因为无论有多少数据副本节点挂了，只要领导者节点不挂，这个Replication集群都可以返回数据，但是当领导者挂了，这整个Replication集群就不能再用了，而没了这个集群也就没有CAP属性可言了。希望这能帮助你理解，如果有不明确的地方也欢迎你继续留言提问，一起学习进步。

共 2 条评论 >

14

常超 置顶

2019-05-06

老师您好，文中说kafka放弃了P属性。
但是从后面的解释来看，即使出现分区（领导者节点和副本1无法通讯），整个系统也能正常工作，这种行为难道不是保持了P属性吗。您能否举一个P属性被放弃的例子？

作者回复: 常超您好，感谢您的提问！

关于这个问题我是这么看的。当我们在分布式环境中讨论CAP属性的时候，P属性可以说是当任意节点断开后，系统还是可以正常的运行。对于整个Kafka系统来说，P当然是必须要保留的。可当你只从Kafka Replication来看的时候，如果一个cluster里面领导者挂掉了，单单就这个cluster来说有再多的副本存在也是无法运行了，所以就Kafka Replication来说，它没有保留P属性。

而在Kafka Replication的设计中为什么说P被放弃了呢，引用Kafka的作者之一Jun Rao在设计Kafka Replication的说法，是因为“All distributed systems must make trade-offs between guaranteeing consistency, availability, and partition tolerance. Our goal was to support replication in a Kafka cluster within a single datacenter, where network partitioning is rare, so our design focuses on maintaining highly available and strongly consistent replicas.”，这一点是在Linkedin的Engineering官方文档上publish的。而在2013年的Apachecon上，Kafka Replication的技术演讲上也明确说明了“Kafka Replication: Pi

ck CA”。

不知道我的解释能否让您更好地理解，也欢迎您继续留言提问，我们一起学习进步！

共 3 条评论 > 14

锦 置顶

2019-05-06

AP，发微博保证最终一致性就可以。
疑问一，mongodb采用CP，那么它在可用性方面有做什么事情吗？
疑问二，kafka这种通过选举leader的方式不就是分区容错性吗？为什么说放弃了P呢？

作者回复: 谢谢你的答案。

关于疑问一：MongoDB的设计默认是希望读写有strong consistency的。当然MongoDB也有自身的Replica Set来保证可用性：https://docs.mongodb.com/manual/replication/#replication-in-mongodb
关于疑问二：你说的没有错，作为多个data clusters来说，Kafka系统是不可能放弃P的，不然一旦leader挂掉系统就没有任何结果可以返回了。但是我在这一讲中所讲述的Intra-cluster Kafka Replication Design是对于一个cluster来说。就像我回答其他有同样疑问的读者一样，之所以我在文中说Kafka Replication选择了CA，是因为Kafka的作者之一Jun Rao在设计Kafka Replication的时候，明确说明了“All distributed systems must make trade-offs between guaranteeing consistency, availability, and partition tolerance. Our goal was to support replication in a Kafka cluster within a single datacenter, where network partitioning is rare, so our design focuses on maintaining highly available and strongly consistent replicas.”，这一点是在Linkedin的官方文档上publish的。而在2013年的Apachecon上，Kafka Replication的技术演讲上也明确说明了“Kafka Replication: Pick CA”。

学习之后提出疑问是一个很好的习惯，也希望后面继续看到你的留言！

共 2 条评论 > 2

mccrms

2019-06-03

文中提到的CA示例其实是有很强的误导性的。

首先我们要明确如下的事实:
1)对于一个分布式系统而言，节点故障和网络故障属于常态
2)如果出现网络故障，会造成节点分区
3)分布式系统在存在节点分区的情况下，C和A是冲突的

通过上面的事实我们可以推断出，如果想设计出一个CA系统，必须保证网络不出现分区才有可能，怎样保证网络不出现分区呢，一是单台机器，二是像例子中所述，将所有节点放在同个数据中心中可以假定网络出现分区的概率很低。这也是为什么例子中的情况可以假装称为CA。

还有一些需要辅助理解CAP的知识如下:
1)对于分布式系统而言，最简单可以分为两种，一是所有节点通过都可以通过某种策略对外提供服务，是对等的，二是所有节点通过一个master对外提供服务，甚至是一个单点的master
2)探讨系统CAP的前提应该是系统在能提供服务的情况下的CAP，如果存在master单点，但是有很多从属worker的话，这时的可用性探讨需要划分为worker故障和master故障来看。对于文中的例子master如果挂掉系统不可用则表名A其实也是勉强的

CAP Theorem is like the old joke about software projects: you can have it on TIME, in BUDGET, or CORRECT. Pick any two😀

CAP三者互相制衡，应该是看侧重哪两个，而不是选了哪两个，不是两个100分剩下的一个0分，本质上都要兼顾的。

共 2 条评论 >　　43

Codelife
2019-05-06

严格来说,CAP理论是针对分区副本来定义的，之所以说kafka放弃P，只支持CA，是因为，kafka原理中当出现单个broke宕机，将要出现分区的时候，直接将该broke从集群中剔除，确保整个集群不会出现P现象

> 作者回复: 谢谢你的留言！你的理解非常正确，就是因为这样Kafka Replication的设计中不需要考虑到P的存在，让整个Replication Design成为CA系统。

共 2 条评论 >　　18

coldpark
2019-08-11

有一个形象的比喻不知道恰当不恰当，一个系统相当于一个团队，有C属性说明这个团队每次都能保质保量完成任务，A属性说明这个团队每次都能及时完成任务，P属性相当于这个团队内部偶尔会犯一些小错误。犯错是很常见的，所以一般都具有P属性。
CP类型的团队对外的形象相当于：我的团队不是完美的，但我的产品绝对不会出问题，只要你给我足够的时间让我们把问题排查清楚。

AP类型的团队给人的感觉就是：人非圣贤，孰能无过，我的队员会犯错，我的团队也有估计不足的时候，但是客户的需求我们总会最快响应。
CA类型的团队有个强人领袖（leader节点）：任何事务无论大小都过问一遍，一旦发现手下有人犯错，立马剔除出团队，如果自己犯错，让出领袖地位，整个团队一定要保证最快最好完成任务。

作者回复: 是有一定道理

8

Zoe
2019-05-28

查了查文档，对于Kafka没有考虑P因素，是因为相同的partition （leader和replications）都在同一个data center里，大概是说不需要去考虑P。

但对于同时满足CA还请老师再解释一下。文档里说，可以config availability over durability，也可以反之。
如果选择availability的话，设置acks=all，可以保证isr里登记在册的broker都接收到信息，但isr里也可能只有leader一个。所以当leader挂了之后到再选出新leader期间就有数据丢失的可能性。
如果选择durability的话可以设置min_insync_replicas，但如果小于这个threshold的时候，不是得等着某个broker恢复，这样不就牺牲availabilty了吗?

所以，到底C和A是如何都满足的?

共 1 条评论>

3

桃园悠然在
2019-05-06

另外，增加一个评论：蔡老师的文章头图每张都跟内容强相关（不知是否自己P图或者照相的），而且右上角有文章序号很用心，点赞!

作者回复: 谢谢你的支持，哈哈!

3

王聪 Claire
2019-05-06

kafka 的replication机制，即使出现分区这样的错误，系统也能够通过领导者节点返回消息。

怎么算放弃了P呢？谢谢。

作者回复: 谢谢你的提问！我的理解是如果仅仅就一个Kafka Replication cluster来说，如果领导者挂了我们就不会再从这个cluster拿到内容了，所以在Intra-cluster Replication这个设计点上，他们是不考虑P的。

之所以我在文中说Kafka Replication选择了CA，是因为Kafka的作者之一Jun Rao在设计Kafka Replication的时候，明确说明了“All distributed systems must make trade-offs between guaranteeing consistency, availability, and partition tolerance. Our goal was to support replication in a Kafka cluster within a single datacenter, where network partitioning is rare, so our design focuses on maintaining highly available and strongly consistent replicas.”，这一点是在Linkedin的官方文档上publish的。而在2013年的Apachecon上，Kafka Replication的技术演讲上也明确说明了“Kafka Replication: Pick CA”。

所以你会发现，他们的设计思路是仅仅就Intra-cluster的data replication出发的。当然，如果从整个Kafka系统来说，是不可能放弃P的。

3

Geek_guys
2019-09-26

其实为啥要纠结这些文字概念呢，个人感觉老师在文章里讲的这些例子很容易误导听众和读者，有些根本就没有讲清楚，比如redis你说是CP，大家知道redis的高可用有主从模式，哨兵模式，cluster模式，如果脱离这些架构或者业务场景讲CAP，未免太有点纸上谈兵，不敢苟同。CAP不是一套方法论，而是一个结果，任何分布式系统的设计一定是在特定环境下解决某个或者某些问题，CAP只是这些设计后的一个结果而已。

2

和你一起搬砖的胡大爷
2019-05-14

kafka的例子其实复杂，不大适合举p的例子，kafka也不一定要全sync之后写入才确认，isr也是可以配的。我的理解这里损失p是在isr不等于replica，分区后leader不见了，一个没同步的replica变成leader，一会leader回来了，两边数据就对不上了。如果isr设置成和replica一致，一旦分区了，有replica无法写入了，kafka那个log就没法用了. 但是如果isr能按生存节点收缩，是不是这个问题也解决了。

2

J.M.Liu
2019-05-12

老师，我好像懂了。kafka replication不保证p，指的是当网络出现分区后，和主有一台副本比如replicatin1和leader失联了，那replication1就会被踢出去，相当它于宕机了。在这里，分区导致replication1不可用了，所以说不保证p。
是这样理解吗？

作者回复: 谢谢你的再次留言！是的，理解正确！既然replication1对于集群来说现在以及以后都不可用了，也就相对于集群没有了这个replication1，那也就不存在说网络分区后replication1还是否在集群正常运行的问题了。

2

J.M.Liu
2019-05-11

不太理解ca系统，允许有机子挂掉，却不允许网络分区。机子都挂掉了，到这台机子的网络可不就是连不通了吗？ 允许机子化掉却不允许机子和集群失联，这是个悖论呀。怎么理解这个呢老师

作者回复: 谢谢你的留言！其实选择CA的系统是少之又少，你的问题里面提到了一个关键点，当机子挂掉的时候或者当网络分区出现的时候，系统会直接将这个机子从集群中剔除掉了，也就是说无论以后这个机子处于什么状态都不会再出现在这个集群中，在集群中剩下的机子数据一直都是同步的，所以就有了有CA却不存在P的说法了。不知道这个解释你觉得怎么样，同时也欢迎你不理解的话继续提问。

2

桃园悠然在
2019-05-06

我理解kafka作为一个消息系统平台，是不允许消息丢失的，所以通过replication机制冗余数据保证。但是P属性更像是一个缺点（容忍了消息丢失），它的好处该怎么理解呢？谢谢老师！

作者回复: 谢谢你的留言！P属性的好处是在分布式环境下，无论网络发不发生分区，整个系统都可以照常运行。不管分区后用户读到的是stale data，还是up-to-date data。像常超同学的留言中举了几个例子，如果系统设计上没有了P属性，很多时候network partition发生了， 那剩下的一些节点可能就无法再用了，最坏的情况下可能整个系统都无法运行下去。

2

wmg

2019-05-06

老师，如果kafka支持必须同步节点写成功，那是不是就是一个cp系统，如果支持非健康节点选举为leader，是不是就是ap系统?

作者回复: 谢谢你的提问！对于整个Kafka系统来说，你的理解是正确的！

2

Flash

2019-05-15

谢谢老师，是不了解微博的机制，请问老师，发微博应该要用什么CAP属性?
看到留言有人说，发微博保证最终一致性就好，不知道这个跟CAP的线性一致性是不是一个意思?

作者回复: 谢谢你的留言！最终一致性和线性一致性并不是一个意思。

线性一致性是属于强一致性，如果用在微博上就是说如果一个用户发了一条新的微博，那这个用户的所有好友在刷新这个的用户微博的时候都一定要看到这条新的微博。

如果是最终一致性的话，指的是如果一个用户发了一条新的微博，那这个用户的好友刷新这个用户微博的时候，有可能会看到这条新的微博，也有可能看不到。但是不断地刷新之后，经过一段时间最终所有的用户总会看到这条新的微博。

所以说发微博一般都会用到AP属性。

1

LaimanYeung

2019-05-12

A：集群某个或某几节点挂掉时，客户端仍然可以访问服务端.
C：客户访问集群中任一服务端，返回的状态都是一致的.
P：集群内部机器通讯出现问题导致服务A的数据无法同步到其他节点时，客户端访问服务A，服务A扔能返回未同步到其他节点的数据.

作者回复: 谢谢你的总结！非常棒！

共 2 条评论>

2

hua168
2019-05-10

Availability和Partition Tolerance并不是由是否允许数据丢失来判断的。

我来举个例子说明吧。假设现在有一个由两个节点组成的集群，这两个节点都可以独立运作也可以与对方通讯，而客户端是与这个集群通讯的。作为一个集群来说，Availability指的是不管这两个节点哪一个节点坏掉不能运作了，对于客户端来说还是可以继续与这个集群进行通讯的。Partition Tolerance指得是，本来这两个节点可能是需要互相通讯来同步数据的，而因为Network Partition使得它们之间不能通讯了，但是对于客户端来说还是可以继续与这个集群进行通讯的。

按照您这样说，我疑惑的是：
1. 节点是所有节点，包括主节点吧?
2. 两个节点之间，如果有数据分别写入的话，那不就是数据不一致吗?
　如果node1写入A数据，node2被写入B数据，怎同步？那数据不是乱套了？读也出现不一的情况
比如是金钱类的，不能是负数，我请求node1花完了，我刷新一下又访问到node2发现还是原来，我又花完了，那我不是可以花双倍？一同步那不是出错？同步不了，因为扣双倍，已小于0

3. 如果是CA的话，P出问题，不是出现不一致吗，那么我怎么保证C?

作者回复: 谢谢你的提问!
1. 是的，节点是指所有的节点。
2. 如果按照你所说的架构，两个节点都允许写入数据的话，中间肯定会有一个master node去同步这些写入数据。而我所理解在你说的例子中可以随机读取任意一个node的数据这种架构，你可以参考dynamo quorum的做法，要保持同步的话不可能只读一个node的数据，而是要读取N个nodes的数据看看是否一致，如果一致了才返回结果给客户。
3. 如果是按照专栏中所讲的Kafka Replication的例子，因为出现network partition出现而导致replication节点无法同步数据了，那这个replication节点就会被剔除出集群，剩下的所有节点数据还是同步的。

 　  1

涵
2019-05-08

老师好，通过您对kafka replication的讲解，以及同学们分析的其他系统，感觉一个分布式平台在设计时需要选择保障CP或AP，被拿掉的A或C也不是完全不考虑，会通过一些其他的辅助系统来支持保障，所以对于平台来说绝大多数时间是可以做到CAP的。不知道这样理解是否正确?

作者回复: 谢谢你的提问！现在在很多系统的设计中都是可以通过调节系统配置来达到CP或者AP的功能，但是不可能同时满足C、A、P这三个属性。

像很多同学们都会提到的，如果一套系统里有了主从复制集群这种高可用的架构，那是不是这个系统就一定是AP系统了？其实也不一定的。举个例子，如果系统已经搭建了主从复制的高可用架构，但是系统想达到CP，而这个时候主节点和从节点的数据更新可能并不一致，但是主节点因为某些原因无法连通或者是暂时崩溃了，系统可以选择对客户端的请求不返回任何结果，因为数据不一致。那这个时候系统其实就没有达到A属性了。


1