09-CAP定理：三选二，架构师必须学会的取舍

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:

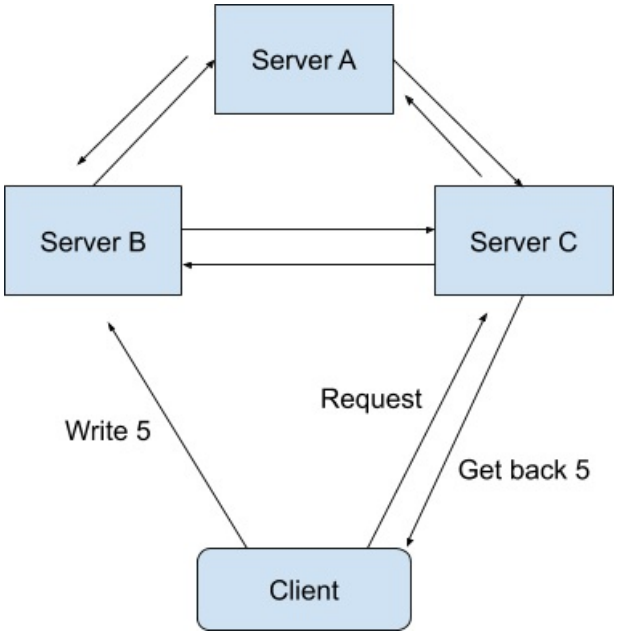
- Consistency
- Availability
- Partition-tolerance

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。



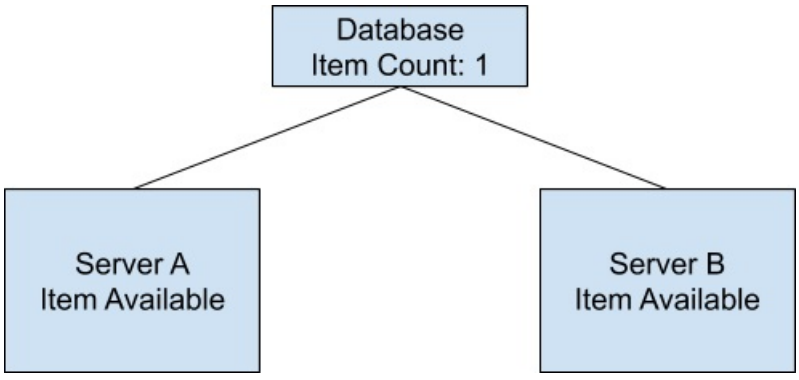
打个比方，现在有两个操作（Operation），操作A和操作B，都需要在同一个分布式系统上完成。

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。

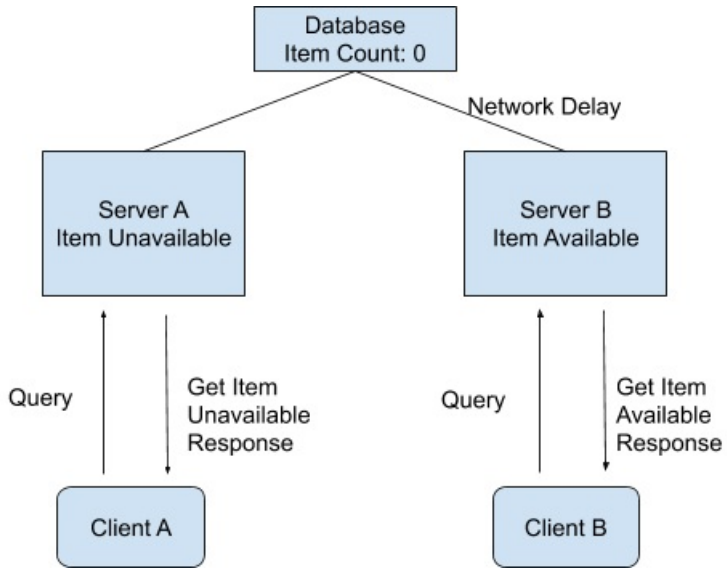


等一段时间过后，商品卖完了，后台就必须将这两台服务器上的商品状态更新为无货状态。

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。

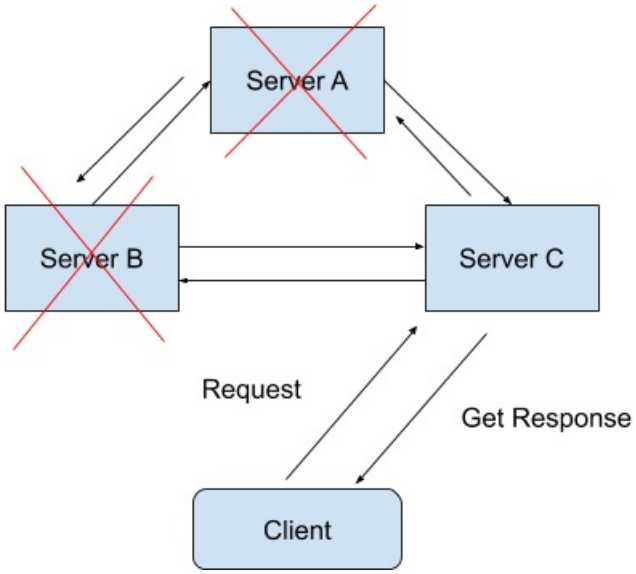
但在我们这个例子中，操作B却返回了有货状态。所以我们说，这个分布式的购物系统并不满足论文里所讲到的线性一致性。

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。

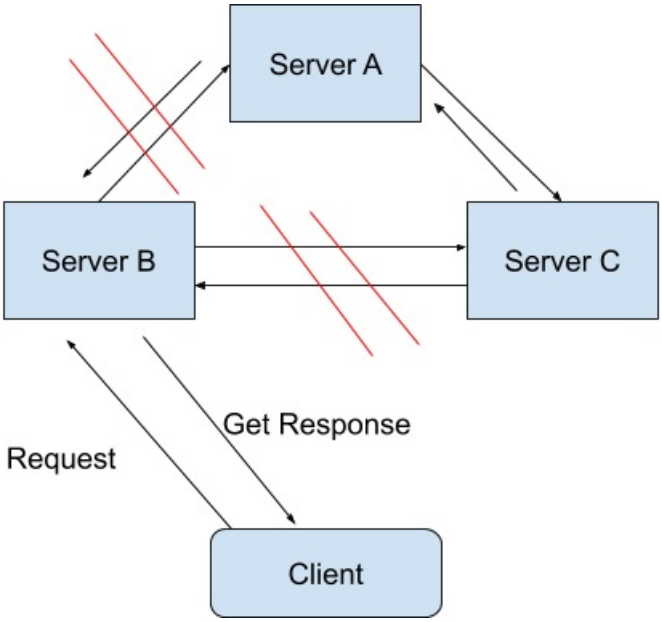


也就是说，当客户端向系统发送请求，只要系统背后的服务器有一台还未崩溃，那么这个未崩溃的服务器必须最终响应客户端。

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，整个网络就会被分成几块区域。而数据分散在这些无法连通的区域中的情况，就被叫做“分区”。



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，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”，意思是即使出现分区这样的“错误”，系统也需要能“容忍”。也就是说，就算分区出现，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。

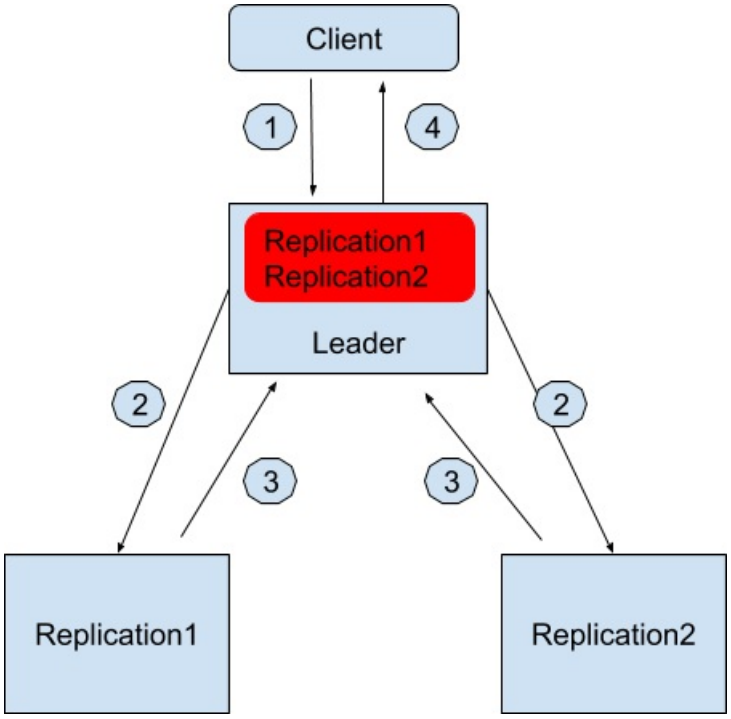
它的具体架构是这样的，在Kafka数据副本（Data Replication）的设计中，先通过Zookeeper选举出一个领导者节点（Leader）。这个领导者节点负责维护一组被称作同步数据副本（In-sync-replica）的节点，所有的数据写入都必须在这个领导者节点中记录。

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。

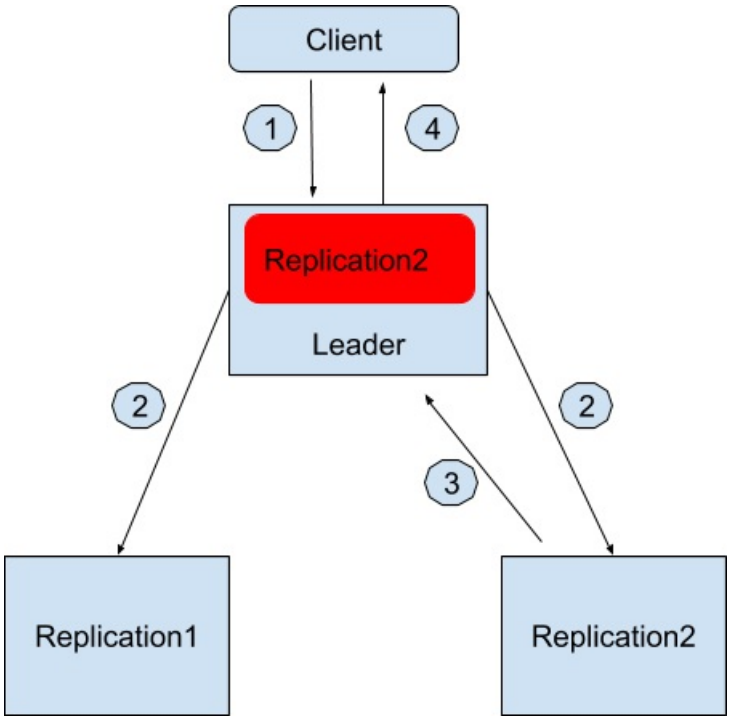


往后用户如果想查询写入的数据，无论是领导者节点还是两个副本都可以返回正确同步的结果。

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。

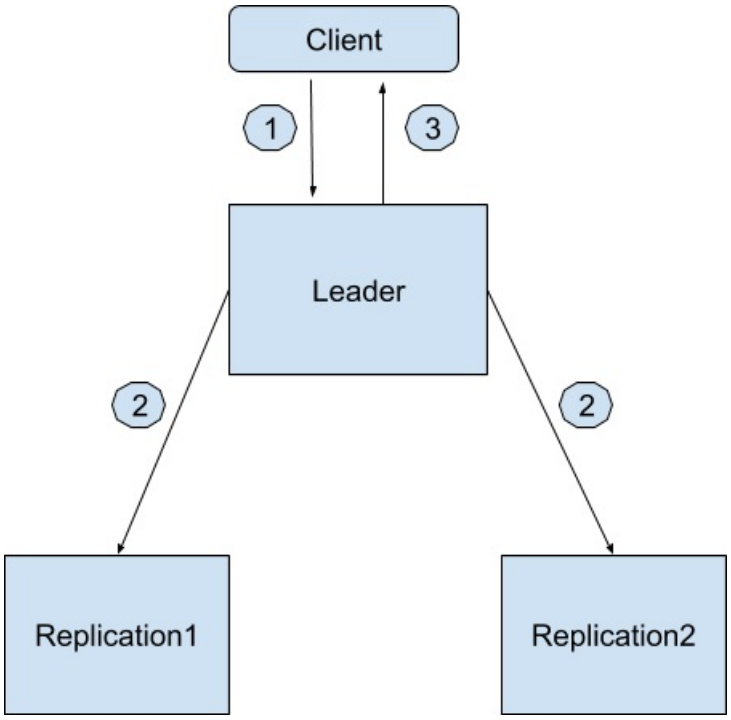
1. 用户会发请求到领导者节点中想写入“Geekbang”。
2. 领导者节点收到请求后先在本地保存好，然后也同时发消息通知Replication1和Replication2。
3. 只有Replication2收到消息后也保存好这条消息并且回复领导者节点写入成功。
4. 领导者节点记录副本2是健康的，并且回复用户写入成功。

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：

- Codelife 2019-05-06 14:59:40
  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 [3赞]

- 王聪 Claire 2019-05-06 09:26:07
  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。 [3赞]

  作者回复2019-05-07 07:24:47
  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。

- 桃园悠然在 2019-05-06 11:28:33
  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！ [2赞]

- 桃园悠然在 2019-05-06 11:24:42
  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！ [2赞]

- 常超 2019-05-06 08:13:04
  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？ [2赞]

- 常超 2019-05-06 14:41:31
  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slave 1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
  [1赞]

- kris37 2019-05-06 13:37:49
  kafka之所以说是ca系统，是因为他跟本就不保证p，因为发生了p kafka 的被独立的分区部分无法保证c也无法保证a [1赞]

- 笑地 2019-05-06 12:26:14
  感觉这几讲都很大数据关系没那么大，更多是讲架构方面的，内容能否向大数据倾斜一点 [1赞]

- 锦 2019-05-06 09:22:19
  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？ [1赞]

- ykkk88 2019-05-06 09:06:33
  kafka replication和放弃p有什么关系么 [1赞]

- wmg 2019-05-06 08:25:19
  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？ [1赞]

- 刘万里 2019-05-07 07:39:21
  老师，您有apache beam相关的书推荐吗？谢谢

- coyang 2019-05-07 07:32:07
  发微博选择ap，c不是特别重要，只要最后能够达到一致就可以了。分布式文件系统GlusterFS感觉和kafka很相似，也是放弃了p。

- Ming 2019-05-06 21:20:21
  CAP的理论是20年前的产物，很粗糙，不同的vendor对其的理解都不一样。因此我以为讨论这个的意义不

大（留个wiki链接完全足够），作为工程师往往关注的是“什么情况下会挂，什么情况下会数据不一致”。

我感觉，这个专栏除了从开篇的若干章，之后的内容质量下滑，讲师也少有对顾客的交互。我真希望平台能提供按阶段收费，而不是一锤子买卖。

- miwucc 2019-05-06 19:59:01
kafka的例子觉得不同意。

作者回复2019-05-07 07:36:41
谢谢你的留言，有质疑是一件好的事情！

首先我不确定你是觉得例子中的哪一块内容你觉得不同意，如果你的疑问是Kafka这个系统是否放弃了P，我的回答是并没有放弃P。作为一个整体系统来说，在分布式的环境下，系统的设计是不会放弃P的，也就是大多数系统在CAP里面都会选择AP或者CP。

而如果你的疑问是Kafka Replication Design的思想里有没有放弃P，这一点我觉得我是认同Kafka作者Jun Rao的观点的，所以我也特地在标题中写上“放弃了P属性的Kafka Replication”，而不是“放弃了P属性的Kafka系统”以免引起误会。

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: Pick CA”。

如果还有疑问的地方，也欢迎你继续留言提问，一起学习进步！

- 利利 2019-05-06 19:50:27
1. kafka作为CA系统的理解：
kafka的设计是只需要保证单个数据中心的broker之间能够数据复制就好，出现网络分区的情况比较少，因此他主攻高可用和强一致性
2. 文中，老师提到kafka的Leader选举是由Zookeeper选举的，这儿有点不严谨
原因：kafka会由Zookeeper选举一台broker作为controller，之后由controller来维护partition的leader、flower，而leader宕机之后进行新leader的选举也是由controller负责的
3. 微博发微博的功能，满足AP就行，C的话只需要是最终一致性就好，就跟微信朋友圈一样

- 利利 2019-05-06 19:49:28
1. kafka作为CA系统的理解：
kafka的设计是只需要保证单个数据中心的broker之间能够数据复制就好，出现网络分区的情况比较少，因此他主攻高可用和强一致性
2. 文中，老师提到kafka的Leader选举是由Zookeeper选举的，这儿有点不严谨
原因：kafka会由Zookeeper选举一台broker作为controller，之后由controller来维护partition的leader、flower，而leader宕机之后进行新leader的选举也是由controller负责的
3. 微博发微博的功能，满足AP就行，C的话只需要是最终一致性就好，就跟微信朋友圈一样

- coder 2019-05-06 19:48:08
Kafka集群leader维护了一个in-sync replica (ISR) set，表示一个和leader保持同步的follower集合，如果follower长时间未向leader同步数据，时间超过一个设置的参数，则该follower就会被踢出ISR，而后该follo

wer回重启自己并向leader同步数据，通过这种方法避免长时间不能同步数据的问题

- 孙稚昊 2019-05-06 19:03:41
  hbase 的所有记录都是append only 的，印象中是AP，反而是consistency 有点妥协？

- 陈建斌红了.. 2019-05-06 12:33:54
  redis是ap 吧，redis集群挂一台，key会rehash，保证可用。redis做分不式锁，主节点挂了，假如主还没来得及同步数据，数据丢了，锁会被申请多次。保证你能申请，这是可用性，不限制你申请多次，这是没有一致性。