

## 22 | 想成为架构师，你必须知道CAP理论

2018-06-16 李运华



### 22 | 想成为架构师，你必须知道CAP理论

朗读人：黄洲君 09'30" | 4.35M

CAP 定理 (CAP theorem) 又被称作布鲁尔定理 (Brewer's theorem)，是加州大学伯克利分校的计算机科学家埃里克·布鲁尔 (Eric Brewer) 在 2000 年的 ACM PODC 上提出的一个猜想。2002 年，麻省理工学院的赛斯·吉尔伯特 (Seth Gilbert) 和南希·林奇 (Nancy Lynch) 发表了布鲁尔猜想的证明，使之成为分布式计算领域公认的一个定理。对于设计分布式系统的架构师来说，CAP 是必须掌握的理论。

布鲁尔在提出 CAP 猜想的时候，并没有详细定义 Consistency、Availability、Partition Tolerance 三个单词的明确定义，因此如果初学者去查询 CAP 定义的时候会感到比较困惑，因为不同的资料对 CAP 的详细定义有一些细微的差别，例如：

Consistency: where all nodes see the same data at the same time.

Availability: which guarantees that every request receives a response about whether it succeeded or failed.

Partition tolerance: where the system continues to operate even if any one part of the system is lost or fails.

([https://console.bluemix.net/docs/services/Cloudant/guides/cap\\_theorem.html#cap-](https://console.bluemix.net/docs/services/Cloudant/guides/cap_theorem.html#cap-))

Consistency: Every read receives the most recent write or an error.

Availability: Every request receives a (non-error) response – without guarantee that it contains the most recent write.

Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

([https://en.wikipedia.org/wiki/CAP\\_theorem#cite\\_note-Brewer2012-6](https://en.wikipedia.org/wiki/CAP_theorem#cite_note-Brewer2012-6))

Consistency: all nodes have access to the same data simultaneously.

Availability: a promise that every request receives a response, at minimum whether the request succeeded or failed.

Partition tolerance: the system will continue to work even if some arbitrary node goes offline or can't communicate.

(<https://www.teamsilverback.com/understanding-the-cap-theorem/>)

为了更好地解释 CAP 理论，我挑选了 Robert Greiner (<http://robertgreiner.com/about/>) 的文章作为参考基础。有趣的是，Robert Greiner 对 CAP 的理解也经历了一个过程，他写了两篇文章来阐述 CAP 理论，第一篇被标记为 “outdated”（有一些中文翻译文章正好参考了第一篇），我将对比前后两篇解释的差异点，通过对比帮助你更加深入地理解 CAP 理论。

## CAP 理论

第一版解释：

Any distributed system cannot guaranty C, A, and P simultaneously.

(<http://robertgreiner.com/2014/06/cap-theorem-explained/>)

简单翻译为：对于一个分布式计算系统，不可能同时满足一致性（Consistence）、可用性（Availability）、分区容错性（Partition Tolerance）三个设计约束。

第二版解释：

In a distributed system (a collection of interconnected nodes that share data.), you can only have two out of the following three guarantees across

a write/read pair: Consistency, Availability, and Partition Tolerance - one of them must be sacrificed.

(<http://robertgreiner.com/2014/08/cap-theorem-revisited/>)

简单翻译为：在一个分布式系统（指互相连接并共享数据的节点的集合）中，当涉及读写操作时，只能保证一致性（Consistence）、可用性（Availability）、分区容错性（Partition Tolerance）三者中的两个，另外一个必须被牺牲。

对比两个版本的定义，有几个很关键的差异点：

- 第二版定义了什么才是 CAP 理论探讨的分布式系统，强调了两点：interconnected 和 share data，为何要强调这两点呢？因为分布式系统并不一定会互联和共享数据。最简单的例如 Memcache 的集群，相互之间就没有连接和共享数据，因此 Memcache 集群这类分布式系统就不符合 CAP 理论探讨的对象；而 MySQL 集群就是互联和进行数据复制的，因此是 CAP 理论探讨的对象。
- 第二版强调了 write/read pair，这点其实是和上一个差异点一脉相承的。也就是说，CAP 关注的是对数据的读写操作，而不是分布式系统的所有功能。例如，ZooKeeper 的选举机制就不是 CAP 探讨的对象。

相比来说，第二版的定义更加精确。

虽然第二版的定义和解释更加严谨，但内容相比第一版来说更加难记一些，所以现在大部分技术人员谈论 CAP 理论时，更多还是按照第一版的定义和解释来说的，因为第一版虽然不严谨，但非常简单和容易记住。

第二版除了基本概念，三个基本的设计约束也进行了重新阐述，我来详细分析一下。

## 1. 一致性（Consistency）

第一版解释：

All nodes see the same data at the same time.

简单翻译为：所有节点在同一时刻都能看到相同的数据。

第二版解释：

A read is guaranteed to return the most recent write for a given client.

简单翻译为：对某个指定的客户端来说，读操作保证能够返回最新的写操作结果。

第一版解释和第二版解释的主要差异点表现在：

- 第一版从节点 node 的角度描述，第二版从客户端 client 的角度描述。

相比来说，第二版更加符合我们观察和评估系统的方式，即站在客户端的角度来观察系统的行为和特征。

- 第一版的关键词是 see，第二版的关键词是 read。

第一版解释中的 see，其实并不确切，因为节点 node 是拥有数据，而不是看到数据，即使要描述也是用 have；第二版从客户端 client 的读写角度来描述一致性，定义更加精确。

- 第一版强调同一时刻拥有相同数据（same time + same data），第二版并没有强调这点。

这就意味着实际上对于节点来说，可能同一时刻拥有不同数据（same time + different data），这和我们通常理解的一致性是有差异的，为何做这样的改动呢？其实在第一版的详细解释中已经提到了，具体内容如下：

A system has consistency if a transaction starts with the system in a consistent state, and ends with the system in a consistent state. In this model, a system can (and does) shift into an inconsistent state during a transaction, but the entire transaction gets rolled back if there is an error during any stage in the process.

参考上述的解释，对于系统执行事务来说，在事务执行过程中，系统其实处于一个不一致的状态，不同的节点的数据并不完全一致，因此第一版的解释 “All nodes see the same data at the same time” 是不严谨的。而第二版强调 client 读操作能够获取最新的写结果就没有问题，因为事务在执行过程中，client 是无法读取到未提交的数据的，只有等到事务提交后，client 才能读取到事务写入的数据，而如果事务失败则会进行回滚，client 也不会读取到事务中间写入的数据。

## 2. 可用性 (Availability)

第一版解释：

Every request gets a response on success/failure.

简单翻译为：每个请求都能得到成功或者失败的响应。

第二版解释：

A non-failing node will return a reasonable response within a reasonable amount of time (no error or timeout).

简单翻译为：非故障的节点在合理的时间内返回合理的响应（不是错误和超时的响应）。

第一版解释和第二版解释主要差异点表现在：

- 第一版是 every request，第二版强调了 A non-failing node。

第一版的 every request 是不严谨的，因为只有非故障节点才能满足可用性要求，如果节点本身就故障了，发给节点的请求不一定能得到一个响应。

- 第一版的 response 分为 success 和 failure，第二版用了两个 reasonable：reasonable response 和 reasonable time，而且特别强调了 no error or timeout。

第一版的 success/failure 的定义太泛了，几乎任何情况，无论是否符合 CAP 理论，我们都可以说请求成功和失败，因为超时也算失败、错误也算失败、异常也算失败、结果不正确也算失败；即使是成功的响应，也不一定是正确的。例如，本来应该返回 100，但实际上返回了 90，这就是成功的响应，但并没有得到正确的结果。相比之下，第二版的解释明确了不能超时、不能出错，结果是合理的，注意没有说“正确”的结果。例如，应该返回 100 但实际上返回了 90，肯定是不正确的结果，但可以是一个合理的结果。

### 3. 分区容忍性 (Partition Tolerance)

第一版解释：

System continues to work despite message loss or partial failure.

简单翻译为：出现消息丢失或者分区错误时系统能够继续运行。

第二版解释：

The system will continue to function when network partitions occur.

简单翻译为：当出现网络分区后，系统能够继续“履行职责”。

第一版解释和第二版解释主要差异点表现在：

- 第一版用的是 work，第二版用的是 function。

work 强调“运行”，只要系统不宕机，我们都可以说系统在 work，返回错误也是 work，拒绝服务也是 work；而 function 强调“发挥作用”“履行职责”，这点和可用性是一脉相承的。也就是说，只有返回 reasonable response 才是 function。相比之下，第二版解释更加明确。

- 第一版描述分区用的是 message loss or partial failure，第二版直接用 network partitions。

对比两版解释，第一版是直接说原因，即 message loss 造成了分区，但 message loss 的定义有点狭隘，因为通常我们说的 message loss（丢包），只是网络故障中的一种；第二版直接说



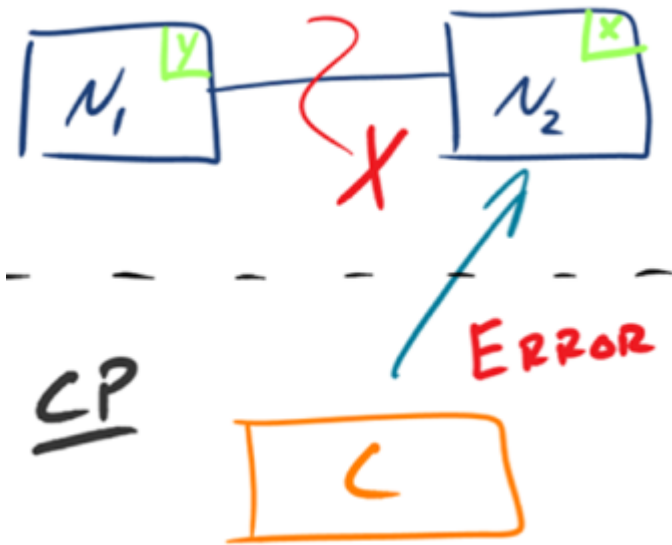
现象，即发生了分区现象，不管是什么原因，可能是丢包，也可能是连接中断，还可能是拥塞，只要导致了网络分区，就通通算在里面。

## CAP 应用

虽然 CAP 理论定义是三个要素中只能取两个，但放到分布式环境下来思考，我们会发现必须选择 P（分区容忍）要素，因为网络本身无法做到 100% 可靠，有可能出故障，所以分区是一个必然的现象。如果我们选择了 CA 而放弃了 P，那么当发生分区现象时，为了保证 C，系统需要禁止写入，当有写入请求时，系统返回 error（例如，当前系统不允许写入），这又和 A 冲突了，因为 A 要求返回 no error 和 no timeout。因此，分布式系统理论上不可能选择 CA 架构，只能选择 CP 或者 AP 架构。

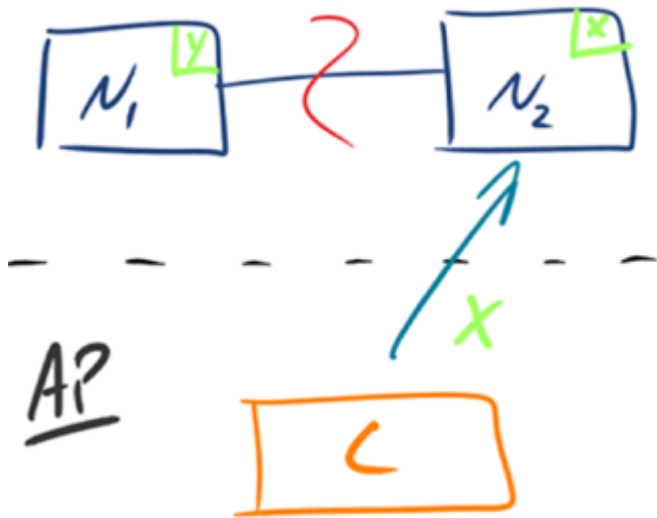
### 1.CP - Consistency/Partition Tolerance

如下图所示，为了保证一致性，当发生分区现象后，N1 节点上的数据已经更新到 y，但由于 N1 和 N2 之间的复制通道中断，数据 y 无法同步到 N2，N2 节点上的数据还是 x。这时客户端 C 访问 N2 时，N2 需要返回 Error，提示客户端 C “系统现在发生了错误”，这种处理方式违背了可用性（Availability）的要求，因此 CAP 三者只能满足 CP。



### 2.AP - Availability/Partition Tolerance

如下图所示，为了保证可用性，当发生分区现象后，N1 节点上的数据已经更新到 y，但由于 N1 和 N2 之间的复制通道中断，数据 y 无法同步到 N2，N2 节点上的数据还是 x。这时客户端 C 访问 N2 时，N2 将当前自己拥有的数据 x 返回给客户端 C 了，而实际上当前最新的数据已经是 y 了，这就不满足一致性（Consistency）的要求了，因此 CAP 三者只能满足 AP。注意：这里 N2 节点返回 x，虽然不是一个“正确”的结果，但是一个“合理”的结果，因为 x 是旧的数据，并不是一个错乱的值，只是不是最新的数据而已。



### 小结

今天我为你讲了 CAP 理论，通过对比两个不同版本的 CAP 理论解释，详细地分析了 CAP 理论的准确定义，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，基于 Paxos 算法构建的分布式系统，属于 CAP 架构中的哪一种？谈谈你的分析和理解。

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

**极客时间**  
重拾极客精神 · 提升技术认知

# 从0开始学架构

—— 资深技术专家的  
实战架构心法 ——

**李运华** 资深技术专家



版权归极客邦科技所有，未经许可不得转载

精选留言



鹅米豆发

👍 31

Paxos算法本身能提供的是，可靠的最终一致性保证。如有足够的隔离性措施，中间状态的无法被客户端读取，则可以达到强一致性，这种属于CP架构。其它情况，就是AP架构。

CAP定理存在不少坑点，理解起来很是令人费解。

1、适用场景。分布式系统有很多类型，有异构的，比如节点之间是上下游依赖的关系，有同构的，比如分区/分片型的、副本型的（主从、多主）。CAP定理的适用场景是副本型的这种。

2、一致性的概念，从强到弱，线性一致性、顺序一致性、因果一致性、单调一致性、最终一致性，CAP中的一致性应该是指顺序一致性。

3、CAP中的一致性，与ACID中的一致性的区别。事务中的一致性，是指满足完整性约束条件，CAP中的一致性，是指读写一致性。

4、CAP中的可用性，与我们常说的高可用的区别。比如HBase、MongoDB属于CP架构，Cassandra、CouchDB属于AP系统，能说后者比前者更高可用么？应该不是。CAP中的可用性，是指在某一次读操作中，即便发现不一致，也要返回响应，即在合理时间内返回合理响应。我们常说的高可用，是指部分实例挂了，能自动摘除，并由其它实例继续提供服务，关键是冗余。

5、哪些情况属于网络分区。网络故障造成的分区，属于。节点应用出现问题导致超时，属于。节点宕机或硬件故障，不属于。

2018-06-16



Yole

👍 12

应该再补充哪些系统上ca，哪些是cp，哪些是ap，他们为什么这么设计，都有什么好处。

2018-06-16

#### 作者回复

你可以自己尝试去分析一下，有疑问评论即可

2018-06-19



鹅米豆发

👍 7

前面对于一致性的描述有些问题。修正一下。

1、Paxos算法本身是满足线性一致性的。线性一致性，也是实际系统能够达到的最强一致性。

2、Paxos及其各种变体，在实际工程领域的实现，大多是做了一定程度的取舍，并不完全是线性一致性的。

3、比如，Zookeeper和Etcd，都是对于写操作（比如选举），满足线性一致性，对于读操作未必满足线性一致性。即可以选择线性一致性读取，也可以选择非线性一致性读取。这里的非线性一致性，就是顺序一致性。

4、cap中的一致性，是指线性一致性，而不是顺序一致性。

2018-06-20

#### 作者回复

感谢，根据Raft的论文描述，工程上目前还没有完全实现paxos算法的系统

2018-06-21



卡莫拉内西

👍 3



paxos, zk的zab协议的理论基础, 保证的是最终一致性, 满足的是cp

2018-06-17

### 作者回复

zk官方资料说zab不是paxos, 而且zk的读操作没有满足CAP的C要求

2018-06-19



Leon Wong

👍 2

老师你好, 有个问题想请教:

最近正在研究 zookeeper, 通读完本篇课程, 心中存疑, 还望解答。

zookeeper 并不保证所有的 client 都能读到最新的数据, 相较于线性一致性而言, zookeeper 采用的是顺序一致性 (我理解一致性程度更弱)。

那么对于这种情况, zookeeper 与最终一致性方案相比, 结合本篇文章的解释, 其本质上依然不能保证所有的 client 读到最新的数据, 那是否可以理解为 zookeeper 就是 AP 系统?

抑或, 根据本篇的解释, zookeeper 采用顺序一致性, 能保证『指定』 (而非所有) 的 client 读到最新的数据, 即可以称之为 CP 系统; 而 AP 系统甚至可能不能保证任意一个 client 能读到最新数据。因此 zookeeper 属于 CP 系统的范畴?

请问老师, 两个思路, 哪个正确?

2018-06-16

### 作者回复

如果严格按照CAP理论来说, C约束并没有限定“指定”的client.

2018-06-19



Leon Wong

👍 2

老师你好, 有个问题想请教:

最近正在研究 zookeeper, 通读完本篇课程, 心中存疑, 还望解答。

zookeeper 并不保证所有的 client 都能读到最新的数据, 相较于线性一致性而言, zookeeper 采用的是顺序一致性 (我理解一致性程度更弱)。

那么对于这种情况, zookeeper 与最终一致性方案相比, 结合本篇文章的解释, 其本质上依然不能保证所有的 client 读到最新的数据, 那是否可以理解为 zookeeper 就是 AP 系统?

抑或, 根据本篇的解释, zookeeper 采用顺序一致性, 能保证『指定』 (而非所有) 的 client 读到最新的数据, 即可以称之为 CP 系统; 而 AP 系统甚至可能不能保证任意一个 client 能读到最新数据。因此 zookeeper 属于 CP 系统的范畴?

请问老师, 两个思路, 哪个正确?

2018-06-16

### 作者回复

如果严格按照CAP理论来说，C约束并没有约束“指定”的client

2018-06-19



星火燎原

👍 2

paxos算法目的是在分布式环境下对主节点一致性的选举，所以属于pc

2018-06-16



轩辕十四

👍 1

网络分区类似于脑裂。

个人对CAP的类比，不知是否合适:

P要求数据有冗余，

C要求数据同步，会花时间，

A要求返回及时，不需要等。

不可能三角形说的是:

要备份要同步，就得等;

要备份不想等，就会不同步;

要同步还不想等，就别备份

2018-07-06

### 作者回复

P要求分布式和数据同步，C要求数据完全一致，A要求返回及时

2018-07-09



zj

👍 1

ZK出现分区，不能再履行职责了吧，因此ZK不满足P。老师这样理解对吗

2018-06-19

### 作者回复

zk多数节点正常就可以正常运行，分区中的少数节点会进入leader选举状态，这个状态不能处理读写操作，因此不符合A，如果不考虑实时一致性，zk基本满足CP的要求

2018-06-21



王磊

👍 1

关于提到的问题，我首先需要考虑的是，Paxos构建的集群是不是互联和有数据共享的，从而确定它是不是CAP所讨论的？

关于本文，我也建议把主流的集群环境，如spark,集群，kafka集群按照CAP理论，是满足了哪两个要素，和为什么这么取舍做下分析。

2018-06-18

### 作者回复

paxos是复制状态机的实现算法，“状态”就是共享数据。

对于开源方案，你可以自己尝试去分析一下，目前大部分集群方案都是ap方案

2018-06-19



追梦

👍 1

求老师后面讲解分布式事务原理🙏

2018-06-17

作者回复

2PC, 3PC都是标准协议，我讲和网上的资料也是一样的😊

我想特别讲的一般是要么网上的资料不清楚甚至有误，或者没有抓住重点的

2018-06-19



大P

👍 1

ZK应该是CP，因为如果节点挂了则节点会自动down下线，而不会把错误或超时的信息给到客户端。且ZK必须保证3台及以上的节点才会提供服务，本质是不是保证C而不是A？

2018-06-17

作者回复

是的，zk不是严格意义上的实时满足C，但也可以算CP系统

2018-06-21



blackncccc

👍 0

cap定理中的p既然是必然要满足的一个而且很容易满足，c和a选择一个就可以满足p了，那为啥不直接说是ca定理呢，另外zk没有数据共享，套用这个定理不太合适呢

2018-08-07

作者回复

ca就不满足p了，简单来说满足ca就不允许分区。

zk的leader和master是共享数据的哦，你看看cap理论解读第二版，节点互联和节点复制数据，zk都满足

2018-08-08



孙振超

👍 0

知识获取来源分为几个层次：论文是第一手，书籍是第二层次，博客资料就是第三层的了，之前学习cap都是在网上找的博客，虽然也找了多篇博客相互佐证进行融合。但今天看了这篇文章才发现对cap的理解还是有些肤浅，真的是无声处惊雷，平凡处见真章，这种求学态度会贯穿在生活工作中的方方面面的，受教了。

paxos协议是为了解决数据一致性而设计的算法，主要是通过投票选举的方式决定出主节点，之后就主节点的数据为准，因而属于pc模式。

另外，作者提到是以 Robert Greiner的文章进行cap的解读，网上介绍cap的文章很多，选择他的原因能否透漏一下？

2018-07-29

作者回复

因为他的解读到位，他的解读契合CAP的精髓

2018-07-30



fiseasky

0

还是半懂不懂，我举例请老师帮我分析一下。

1 我以前做的一个健康系统，就是一张表里面存了用户的健康数据，有一个类型字段来标识数据类型，比如血压，血糖，心跳，体温等。然后搭建多台服务器来分析用户的健康状况，比如服务器A 负责分析血压，血糖。服务器B 负责分析心跳数据。服务器 C 负责分析体温数据。以此类推。请问一下这个属于分布式系统吗？若是，是 AP还是CP。

2 现在的ERP系统存储数据分了好几个数据库来存，比如与业务相关的存在 MY SQL,基础数据放在 SQL SERVER,日志写在MONGO DB中，还用了消息队列REDIS。请问一下这个是否也属于分布式，为什么？谢谢

2018-07-22

#### 作者回复

1. 我理解不算，你是统一存储，分开运算，不存在数据复制的场景，这个架构就是计算高可用中任务分解的架构

2. 不属于CAP定义的分布式系统，CAP要求节点间针对同一份数据进行复制和备份

2018-07-22



tim

0

请问一下作者，在CP的选型中。假如是查询一条并不存在的数据，文中还说得通。

但如果出现更新数据不及时，由于n1和n2 出现分区错误，那么n2如何知道自己不是最新的数据并返回error呢？

假如就是简单的mysql主从，从库并没有断连主库，只是数据在请求来是还没有更新到最新。那么从库又从哪里得知这件事儿的呢？

2018-07-22

#### 作者回复

CAP理论是忽略延时的，这就是说理论做了一个假设，只要网络通数据就会一致，这也是实际应用CAP时容易踩的坑，后面一章会讲

2018-07-22



sensitivemix

0

求华仔介绍分布式的开源案例，让我们可以在实践中学习~

2018-07-22

#### 作者回复

各种技术大会会有很多案例分享呢，可以按照专栏的思想去具体分析一个试试

2018-07-22



tim

0

有个问题，既然无法保证p，那么我们选择它又有什么用呢。

况且有些p是无法通过服务器来处理的，比如连接到服务器时，网络超时，服务器未接到任何

信息。总不能指望浏览器来重试保证吧。

对于p的理解，我觉得还是难以理解。有没有更加深入的文章

2018-07-21

#### 作者回复

P是最容易保证的，中文叫“分区容忍”，意思就是你搭建一个分布式集群，然后断开网络，就是P了

2018-07-22



天真有邪

0

zab是mulit paxos的一种实现，都属于一阶段操作， mulit paxos实现了日志的乱序提交，允许日志空洞，而zab和raft日志需要顺序提交

2018-07-10



大光头

0

应该是一个AP系统，因为当发生分区时，并不能保证拿到最新结果，而只能保证拿到一个结果

2018-07-05

#### 作者回复

paxos是cp，因为分区发生后，少数派的分区会拒绝服务

2018-07-05