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

李运华

00:00 / 09:30

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-)

> 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)

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Greiner（http://robertgreiner.com/about/）的文章作为参考基础。有趣的是，Robert Greiner对CAP的理解也经历了一个过程，他写了两篇文章来阐述CAP理论，第一篇被标记为"outdated"（有一些中文翻译文章正好参考了第一篇），我将对比前后两篇解释的差异点，通过对比帮助你更加深入地理解CAP理论。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

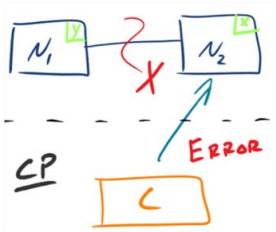
对比两版解释，第一版是直接说原因，即message loss造成了分区，但message loss的定义有点狭隘，因为通常我们说的message loss（丢包），只是网络故障中的一种；第二版直接说现象，即发生了分区现象，不管是什么原因，可能是丢包，也可能是连接中断，还可能是拥塞，只要导致了网络分区，就通通算在里面。

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。

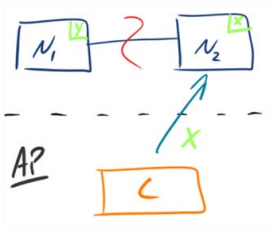
1.CP - Consistency/Partition Tolerance

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！）

鹅米豆发

2018-06-16

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、CounchDB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。

Yole

2018-06-16

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。

作者回复

2018-06-19

你可以自己尝试去分析一下，有疑问评论即可

鹅米豆发

2018-06-20

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。

作者回复

2018-06-21

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

卡莫拉内西

2018-06-17

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

作者回复

2018-06-19

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

星火燎原

2018-06-16

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

zj

2018-06-19

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

作者回复

2018-06-21

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

王磊

2018-06-18

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。

作者回复

2018-06-19

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

Leon Wong

2018-06-16

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？

作者回复

2018-06-19

如果严格按照CAP理论来说，C约束并没有限定"指定"的client.

Leon Wong

2018-06-16

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？

作者回复

2018-06-19

如果严格按照CAP理论来说，C约束并没有约束"指定"的client

天真有邪

2018-07-10

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

轩辕十四

2018-07-06

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

作者回复

2018-07-09

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

大光头

2018-07-05

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

作者回复

2018-07-05

paxos是cp，因为分区发生后，少数派的分区会拒绝服务

王虹凯

2018-07-03

cap如果从关注读写（好像也是定义关注这块）出发的话，paxos应该是ap。读写：因为半数提交，导致数据不一致；不过能保证数据原子提交成功与否！
不知道这样的理解是否正确！

任鉴非

2018-07-02

属于cp架构

刚子

2018-06-28

你好，大神，原文中"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？分布式系统并不一定会互联和共享数据"

其中"分布式系统并不一定会互联和共享数据"结合上下文理解起来是一定的意思

作者回复

2018-06-28

mc集群的节点就不互联，负载均衡的集群节点也不互联

小江

2018-06-26

有很多开源体统 副本更新策略是先更新主再异步更新副本，这种都不满足C？

jacy

2018-06-26

对客户端来说，正常运行时是ap,选举恢复的过程中为p？

朱永昌

2018-06-22

老师您好，请问文中的"网络分区"是什么意思，能否用通俗的需要解释下？谢谢？！

作者回复

2018-06-23

假设本来5台机器网络都是通的，现在由于交换机故障，其中3台联通形成小团体A，另外两台联通形成小团体B，但是A和B不联通

安小依

2018-06-22

李老师，那两张图片是用什么软件画的��

作者回复

2018-06-22

网上引用的，不知道具体什么软件画的

万历十五年

2018-06-20

最后CAP应用中的例一，为什么违背了A就是CP呢？我觉得此例，既违背了A，也违背了C（因为它的读没能返回最新的号）。
分布式系统，如果设计的好的话，至多只能达到CAP中的两个，如果设计的不好，可能两个都不能达到。
请问华仔以上的两个理解是否正确

作者回复

2018-06-21

AP方案中的节点可以返回旧的值，没有违反A

CAP确实是说最多满足两个，有可能满足一个，全部不满足还是比较难的����

王维

2018-06-20

我认为：
不同的系统，不同的应用场景，对于CAP的应用是不同的，有的要求CP，有的要求AP, 例如淘宝在双十一的时候，为了更好的减轻负荷，在下单完成后并没有立即更新用户的订单库，而是采用的延迟更新的方式，这里满足的就是AP。

请问华仔我这个分析对吗

作者回复

2018-06-21

是的，牺牲了一定的用户体验来保证最核心的体验

何国平

2018-06-20

一般更倾向于cp，然后通过冗余来实现a

追梦

2018-06-19

求老师后面讲解分布式事务的一个开源实现��

dbo

2018-06-18

https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed
对于CAP的理解，作者另有解释，并非三者只能取其二，不是1和0的关系，而是1到100的关系，这篇文章解释的挺全的。

作者回复

2018-06-19

下一篇就讲了

追梦

2018-06-17

求老师后面讲解分布式事务原理��

作者回复

2018-06-19

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

大P

2018-06-17

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?

作者回复

2018-06-21

是的，zk不是严格意义上的实时满足C，但也可以算CP系统

空档滑行

2018-06-17

Paxos算法是为了在分布式系统中对某一项决议达成一致的算法，采用的是多数派达成共识。
如果基于paxos实现的分布式系统如果在发生分区时，并不能保证一致性。如果系统是分布式数据库这种需要事务的，则需要结合2PC等强一致性协议。所以我的理解是基于paxos协议实现的分布式系统是满足AP的

作者回复

2018-06-19

分区时，少数派的分区无法完成操作，多数派的分区可以继续保证一致性，因此paxos算法实现的系统是cp