**从分布式一致性谈到CAP理论、BASE理论**

转载 2016年04月13日 15:08:05

* 2246

**问题的提出**

在计算机科学领域，**分布式一致性**是一个相当重要且被广泛探索与论证问题，首先来看三种业务场景。

1、火车站售票

假如说我们的终端用户是一位经常坐火车的旅行家，通常他是去车站的售票处购买车票，然后拿着车票去检票口，再坐上火车，开始一段美好的旅行----一切似乎都是那么和谐。想象一下，如果他选择的目的地是杭州，而某一趟开往杭州的火车只剩下最后一张车票，可能在同一时刻，不同售票窗口的另一位乘客也购买了同一张车票。假如说售票系统没有进行一致性的保障，两人都购票成功了。而在检票口检票的时候，其中一位乘客会被告知他的车票无效----当然，现代的中国铁路售票系统已经很少出现这样的问题了。但在这个例子中我们可以看出，终端用户对于系统的需求非常简单：

"请售票给我，如果没有余票了，请在售票的时候就告诉我票是无效的"

这就对购票系统提出了严格的一致性要求----系统的数据（本例中指的就是那趟开往杭州的火车的余票数）无论在哪个售票窗口，每时每刻都必须是准确无误的！

**2、银行转账**

假如我们的终端用户是一位刚毕业的大学生，通常在拿到第一个月工资的时候，都会选择向家里汇款。当他来到银行柜台，完成转账操作后，银行的柜台服务员会友善地提醒他："您的转账将在N个工作日后到账！"。此时这名毕业生有一定的沮丧，会对那名柜台服务员叮嘱："好吧，多久没关系，钱不要少就好了！"----这也成为了几乎所有用户对于现代银行系统最基本的需求

3、网上购物

假如说我们的终端用户是一位网购达人，当他看见一件库存量为5的心仪商品，会迅速地确认购买，写下收货地址，然后下单----然而，在下单的那个瞬间，系统可能会告知该用户："库存量不足！"。此时绝大部分消费者都会抱怨自己动作太慢，使得心爱的商品被其他人抢走了。

但其实有过网购系统开发经验的工程师一定明白，在商品详情页上显示的那个库存量，通常不是该商品的真实库存量，只有在真正下单购买的时候，系统才会检查该商品的真实库存量。但是，谁在意呢？

问题的解读

对于上面三个例子，相信大家一定看出来了，我们的终端用户在使用不同的计算机产品时对于数据一致性的需求是不一样的：

1、有些系统，既要快速地响应用户，同时还要保证系统的数据对于任意客户端都是真实可靠的，就像火车站售票系统

2、有些系统，需要为用户保证绝对可靠的数据安全，虽然在数据一致性上存在延时，但最终务必保证严格的一致性，就像银行的转账系统

3、有些系统，虽然向用户展示了一些可以说是"错误"的数据，但是在整个系统使用过程中，一定会在某一个流程上对系统数据进行准确无误的检查，从而避免用户发生不必要的损失，就像网购系统

分布一致性的提出

在分布式系统中要解决的一个重要问题就是数据的复制。在我们的日常开发经验中，相信很多开发人员都遇到过这样的问题：假设客户端C1将系统中的一个值K由V1更新为V2，但客户端C2无法立即读取到K的最新值，需要在一段时间之后才能读取到。这很正常，因为数据库复制之间存在延时。

分布式系统对于数据的复制需求一般都来自于以下两个原因：

1、为了增加系统的可用性，以防止单点故障引起的系统不可用

2、提高系统的整体性能，通过负载均衡技术，能够让分布在不同地方的数据副本都能够为用户提供服务

数据复制在可用性和性能方面给分布式系统带来的巨大好处是不言而喻的，然而数据复制所带来的一致性挑战，也是每一个系统研发人员不得不面对的。

所谓分布一致性问题，是指在分布式环境中引入数据复制机制之后，不同数据节点之间可能出现的，并无法依靠计算机应用程序自身解决的数据不一致的情况。简单讲，数据一致性就是指在对一个副本数据进行更新的时候，必须确保也能够更新其他的副本，否则不同副本之间的数据将不一致。

那么如何解决这个问题？一种思路是"**既然是由于延时动作引起的问题，那我可以将写入的动作阻塞，直到数据复制完成后，才完成写入动作**"。没错，这似乎能解决问题，而且有一些系统的架构也确实直接使用了这个思路。但这个思路在解决一致性问题的同时，又带来了新的问题：写入的性能。如果你的应用场景有非常多的写请求，那么使用这个思路之后，后续的写请求都将会阻塞在前一个请求的写操作上，导致系统整体性能急剧下降。

总得来说，我们无法找到一种能够满足分布式系统所有系统属性的分布式一致性解决方案。因此，如何既保证数据的一致性，同时又不影响系统运行的性能，是每一个分布式系统都需要重点考虑和权衡的。于是，一致性级别由此诞生：

1、强一致性

这种一致性级别是最符合用户直觉的，它要求系统写入什么，读出来的也会是什么，用户体验好，但实现起来往往对系统的性能影响大

2、弱一致性

这种一致性级别约束了系统在写入成功后，不承诺立即可以读到写入的值，也不久承诺多久之后数据能够达到一致，但会尽可能地保证到某个时间级别（比如秒级别）后，数据能够达到一致状态

3、最终一致性

最终一致性是弱一致性的一个特例，系统会保证在一定时间内，能够达到一个数据一致的状态。这里之所以将最终一致性单独提出来，是因为它是弱一致性中非常推崇的一种一致性模型，也是业界在大型分布式系统的数据一致性上比较推崇的模型

分布式环境的各种问题

分布式系统体系结构从其出现之初就伴随着诸多的难题和挑战：

1、通信异常

从集中式向分布式演变的过程中，必然引入网络因素，由于网络本身的不可靠性，因此也引入了额外的问题。分布式系统需要在各个节点之间进行网络通信，因此每次网络通信都会伴随着网络不可用的风险，网络光纤、路由器或是DNS等硬件设备或是系统不可用都会导致最终分布式系统无法顺利完成一次网络通信。另外，即使分布式系统各个节点之间的网络通信能够正常进行，其延时也会大于单机操作。通常我们认为现代计算机体系结构中，单机内存访问的延时在纳秒数量级（通常是10ns），而正常的一次网络通信的延迟在0.1~1ms左右（相当于内存访问延时的105倍），如此巨大的延时差别，也会影响到消息的收发过程，因此消息丢失和消息延迟变得非常普遍

2、网络分区

**当网络由于发生异常情况，导致分布式系统中部分节点之间的网络延时不断增大，最终导致组成分布式系统的所有节点中，只有部分节点之间能够正常通信，而另一些节点则不能----我们将这个现象称为网络分区**。当网络分区出现时，分布式系统会出现局部小集群，在极端情况下，这些局部小集群会独立完成原本需要整个分布式系统才能完成的功能，包括对数据的事物处理，这就对分布式一致性提出了非常大的挑战

3、三态

上面两点，我们已经了解到在分布式环境下，网络可能会出现各式各样的问题，因此分布式系统的每一次请求与响应，存在特有的三态概念，即**成功、失败、超时**。在传统的单机系统中，应用程序在调用一个函数之后，能够得到一个非常明确的响应：成功或失败。而在分布式系统中，由于网络是不可靠的，虽然在绝大部分情况下，网络通信也能够接受到成功或失败的响应，当时当网络出现异常的情况下，就可能会出现超时现象，通常有以下两种情况：

（1）由于网络原因，该请求并没有被成功地发送到接收方，而是在发送过程中就发生了消息丢失现象

（2）该请求成功地被接收方接收后，进行了处理，但是在将响应反馈给发送方的过程中，发生了消息丢失现象

当出现这样的超时现象时，网络通信的发起方是无法确定当前请求是否被成功处理的

4、节点故障

节点故障则是分布式环境下另一个比较常见的问题，指的是组成分布式系统的服务器节点出现的宕机或"僵死"现象，通常根据经验来说，每个节点都有可能出现故障，并且每天都在发生

分布式事物

随着分布式计算的发展，事物在分布式计算领域也得到了广泛的应用。在单机数据库中，我们很容易能够实现一套满足ACID特性的事物处理系统，但在分布式数据库中，数据分散在各台不同的机器上，如何对这些数据进行分布式的事物处理具有非常大的挑战。

分布式事物是指事物的参与者、支持事物的服务器、资源服务器以及事物管理器分别位于分布式系统的不同节点上，通常一个分布式事物中会涉及对多个数据源或业务系统的操作。

可以设想一个最典型的分布式事物场景：一个跨银行的转账操作涉及调用两个异地的银行服务，其中一个是本地银行提供的取款服务，另一个则是目标银行提供的存款服务，这两个服务本身是无状态并且相互独立的，共同构成了一个完整的分布式事物。如果从本地银行取款成功，但是因为某种原因存款服务失败了，那么就必须回滚到取款之前的状态，否则用户可能会发现自己的钱不翼而飞了。

从这个例子可以看到，一个分布式事务可以看做是多个分布式的操作序列组成的，例如上面例子的取款服务和存款服务，通常可以把这一系列分布式的操作序列称为子事物。因此，分布式事务也可以被定义为一种嵌套型的事物，同时也就具有了ACID事物特性。但由于在分布式事务中，各个子事物的执行是分布式的，因此要实现一种能够保证ACID特性的分布式事物处理系统就显得格外复杂。

CAP理论

一个经典的分布式系统理论。CAP理论告诉我们：**一个分布式系统不可能同时满足一致性（C：Consistency）、可用性（A：Availability）和分区容错性（P：Partition tolerance）这三个基本需求，最多只能同时满足其中两项**。

1、一致性

在分布式环境下，一致性是指数据在多个副本之间能否保持一致的特性。在一致性的需求下，当一个系统在数据一致的状态下执行更新操作后，应该保证系统的数据仍然处于一直的状态。

对于一个将数据副本分布在不同分布式节点上的系统来说，如果对第一个节点的数据进行了更新操作并且更新成功后，却没有使得第二个节点上的数据得到相应的更新，于是在对第二个节点的数据进行读取操作时，获取的依然是老数据（或称为脏数据），这就是典型的分布式数据不一致的情况。在分布式系统中，如果能够做到针对一个数据项的更新操作执行成功后，所有的用户都可以读取到其最新的值，那么这样的系统就被认为具有强一致性

2、可用性

可用性是指系统提供的服务必须一直处于可用的状态，对于用户的每一个操作请求总是能够在有限的时间内返回结果。这里的重点是"有限时间内"和"返回结果"。

"有限时间内"是指，对于用户的一个操作请求，系统必须能够在指定的时间内返回对应的处理结果，如果超过了这个时间范围，那么系统就被认为是不可用的。另外，"有限的时间内"是指系统设计之初就设计好的运行指标，通常不同系统之间有很大的不同，无论如何，对于用户请求，系统必须存在一个合理的响应时间，否则用户便会对系统感到失望。

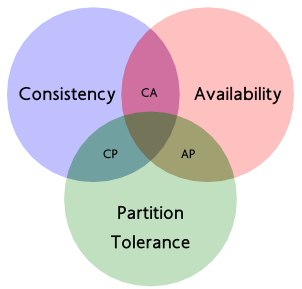
"返回结果"是可用性的另一个非常重要的指标，它要求系统在完成对用户请求的处理后，返回一个正常的响应结果。正常的响应结果通常能够明确地反映出队请求的处理结果，即成功或失败，而不是一个让用户感到困惑的返回结果。

3、分区容错性

分区容错性约束了一个分布式系统具有如下特性：**分布式系统在遇到任何网络分区故障的时候，仍然需要能够保证对外提供满足一致性和可用性的服务，除非是整个网络环境都发生了故障**。

网络分区是指在分布式系统中，不同的节点分布在不同的子网络（机房或异地网络）中，由于一些特殊的原因导致这些子网络出现网络不连通的状况，但各个子网络的内部网络是正常的，从而导致整个系统的网络环境被切分成了若干个孤立的区域。需要注意的是，组成一个分布式系统的每个节点的加入与退出都可以看作是一个特殊的网络分区。

既然一个分布式系统无法同时满足一致性、可用性、分区容错性三个特点，所以我们就需要抛弃一样：



用一张表格说明一下：

|  |  |
| --- | --- |
| **选    择** | **说    明** |
| CA | 放弃分区容错性，加强一致性和可用性，其实就是传统的单机数据库的选择 |
| AP | 放弃一致性（这里说的一致性是强一致性），追求分区容错性和可用性，这是很多分布式系统设计时的选择，例如很多NoSQL系统就是如此 |
| CP | 放弃可用性，追求一致性和分区容错性，基本不会选择，网络问题会直接让整个系统不可用 |

需要明确的一点是，对于一个分布式系统而言，分区容错性是一个最基本的要求。因为既然是一个分布式系统，那么分布式系统中的组件必然需要被部署到不同的节点，否则也就无所谓分布式系统了，因此必然出现子网络。而对于分布式系统而言，网络问题又是一个必定会出现的异常情况，因此分区容错性也就成为了一个分布式系统必然需要面对和解决的问题。因此系统架构师往往需要把精力花在如何根据业务特点在C（一致性）和A（可用性）之间寻求平衡。

BASE理论

BASE是Basically Available（基本可用）、Soft state（软状态）和Eventually consistent（最终一致性）三个短语的缩写。BASE理论是对CAP中一致性和可用性权衡的结果，其来源于对大规模互联网系统分布式实践的总结，是基于CAP定理逐步演化而来的。BASE理论的核心思想是：**即使无法做到强一致性，但每个应用都可以根据自身业务特点，采用适当的方式来使系统达到最终一致性**。接下来看一下BASE中的三要素：

1、基本可用

基本可用是指分布式系统在出现不可预知故障的时候，允许损失部分可用性----注意，这绝不等价于系统不可用。比如：

（1）响应时间上的损失。正常情况下，一个在线搜索引擎需要在0.5秒之内返回给用户相应的查询结果，但由于出现故障，查询结果的响应时间增加了1~2秒

（2）系统功能上的损失：正常情况下，在一个电子商务网站上进行购物的时候，消费者几乎能够顺利完成每一笔订单，但是在一些节日大促购物高峰的时候，由于消费者的购物行为激增，为了保护购物系统的稳定性，部分消费者可能会被引导到一个降级页面

2、软状态

软状态指允许系统中的数据存在中间状态，并认为该中间状态的存在不会影响系统的整体可用性，即允许系统在不同节点的数据副本之间进行数据同步的过程存在延时

3、最终一致性

最终一致性强调的是所有的数据副本，在经过一段时间的同步之后，最终都能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。

总的来说，BASE理论面向的是大型高可用可扩展的分布式系统，和传统的事物ACID特性是相反的，**它完全不同于ACID的强一致性模型，而是通过牺牲强一致性来获得可用性，并允许数据在一段时间内是不一致的，但最终达到一致状态**。但同时，在实际的分布式场景中，不同业务单元和组件对数据一致性的要求是不同的，因此在具体的分布式系统架构设计过程中，ACID特性和BASE理论往往又会结合在一起。