05 | 分布式系统（下）：架构师不得不知的三大指标

2019-04-26 蔡元楠 来自北京

《大规模数据处理实战》

你好，我是蔡元楠。

上一讲中，我们学习了如何用服务等级协议（SLA）来评估我们设计的分布式系统，并了解了几个常见的 SLA 指标。

今天我们继续来探索分布式系统的另外几个重要基础概念。

可扩展性

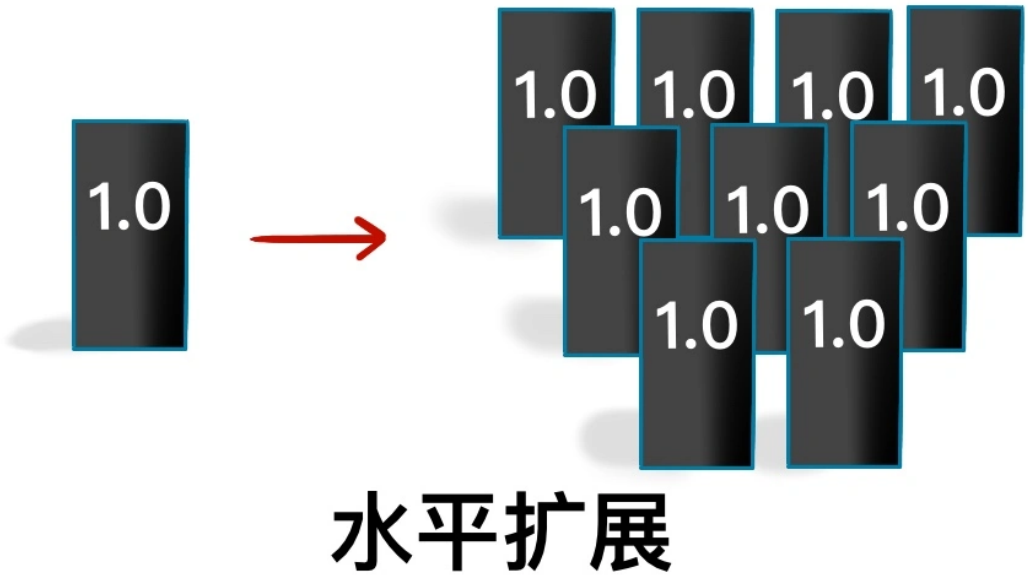
还是从我们为什么需要分布式系统讲起。原因是我们要面对的数据量越来越大，从 GB 到 TB 再到现在的 PB 级，单机无法胜任这样的工作。

工作中也常有这样的场景，随着业务变得越来越复杂，之前设计的系统无法处理日渐增长的负载。这时，我们就需要增加系统的容量。

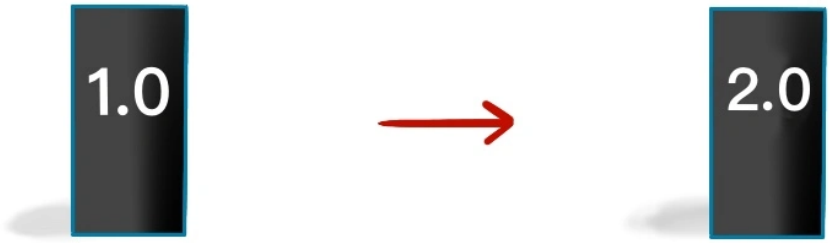
分布式系统的核心就是可扩展性（Scalability）。

最基本而且最流行的增加系统容量的模型有两种: 水平扩展（Horizontal Scaling）和垂直扩展（Vertical Scaling）。

所谓水平扩展，就是指在现有的系统中增加新的机器节点。



垂直扩展就是在不改变系统中机器数量的情况下，“升级”现有机器的性能，比如增加机器的内存。



垂直扩展

举个例子，假设你现在负责一批木材采伐的操作。你有 3 辆卡车，每辆车一次可以运 25 根木材。那么 1 小时最多可以运 3 辆卡车 * 25 根木材 * 1 小时 =75 根木材 / 小时。

如果要使这个系统的负荷量增加一倍，用水平扩展的办法，我们可以将卡车的数量增加到 6 辆；用垂直扩展的办法，我们可以使每辆卡车的运输量增加一倍，或者使每辆卡车的速度增加一倍。

你是不是已经发现了，水平扩展的适用范围更广，操作起来更简单，并且会提升系统的可用性（Availability）。

如果你的系统部署在 AWS 或者其他主流的云服务上，你只需要点几个按钮，就可以在现有的机器集群中增加一个新的节点。

但是，无节制地增加机器数量也会带来一些问题，比如机器的管理、调度、通信会变得更加复杂，出错的可能性会更高，更难保证数据的一致性等等。

与之相反，垂直扩展并没有让整个系统变得更加复杂，控制系统的代码也不需要做任何调整，但是它受到的限制比较多。多数情况下，单个机器的性能提升是有限的。而且受制于摩尔定

律，提高机器的性能往往比购买新的机器更加昂贵。

所以在工作中，我们要对这两种模式进行取舍，要具体情况具体分析。

同样地，在大数据的时代，数据增长速度越来越快，数据规模越来越大，对数据存储系统的扩展性要求也越来越高。

传统的关系型数据库因为表与表之间的数据有关联，经常要进行 Join 操作，所有数据要存放在单机系统中，很难支持水平扩展。而 NoSQL 型的数据库天生支持水平扩展，所以这类存储系统的应用越来越广，如 BigTable、MongoDB 和 Redis 等。

一致性

可用性对于任何分布式系统都很重要。一般来说，构成分布式系统的机器节点的可用性要低于系统的可用性。

举个例子，如果我们想要构建一个可用性 99.999% 的分布式系统（每年约 5 分钟的宕机时间），但是我们使用的单台机器节点的可用性是 99.9%（每年约 8 个小时的宕机时间）。那么想要达到我们的目标，最简单的办法就是增加系统中机器节点的数量。这样即使有部分机器宕机了，其他的机器还在持续工作，所以整个系统的可用性就提高了。

这种情况下，我们要思考一个问题：如何保证系统中不同的机器节点在同一时间，接收到和输出的数据是一致的呢？这时就要引入一致性（Consistency）的概念。

回到之前的例子，要保证分布式系统内的机器节点有相同的信息，就需要机器之间定期同步。

然而，发送信息也会有失败的可能，比如信息丢失或者有的节点正好宕机而无法接收。因此，一致性在高可用性的系统里是非常核心的概念。

接下来，我会给你介绍几个在工程中常用的一致性模型，分别是：强一致性（Strong Consistency），弱一致性（Weak Consistency），最终一致性（Eventual Consistency）。

强一致性：系统中的某个数据被成功更新后，后续任何对该数据的读取操作都将得到更新后的值。所以在任意时刻，同一系统所有节点中的数据是一样的。

弱一致性：系统中的某个数据被更新后，后续对该数据的读取操作可能得到更新后的值，也可能是更改前的值。但经过“不一致时间窗口”这段时间后，后续对该数据的读取都是更新后的值。

最终一致性：是弱一致性的特殊形式。存储系统保证，在没有新的更新的条件下，最终所有的访问都是最后更新的值。

上面这三点我描述得比较正式，但其实都不难理解。这里，我进一步给你做个说明。

在强一致性系统中，只要某个数据的值有更新，这个数据的副本都要进行同步，以保证这个更新被传播到所有备份数据库中。在这个同步进程结束之后，才允许服务器来读取这个数据。

所以，强一致性一般会牺牲一部分延迟性，而且对于全局时钟的要求很高。举个例子，Google Cloud 的 Cloud Spanner 就是一款具备强一致性的全球分布式企业级数据库服务。

在最终一致性系统中，我们无需等到数据更新被所有节点同步就可以读取。尽管不同的进程读同一数据可能会读到不同的结果，但是最终所有的更新会被按时间顺序同步到所有节点。所以，最终一致性系统支持异步读取，它的延迟比较小。比如亚马逊云服务的 DynamoDB 就支持最终一致的数据读取。

除了以上三个，分布式系统理论中还有很多别的一致性模型，如顺序一致性（Sequential Consistency），因果一致性（Casual Consistency）等，如果你感兴趣的话，可以自己查资料了解一下。

在实际应用系统中，强一致性是很难实现的，应用最广的是最终一致性。我们一起来看两个例子。

很多人认为银行间转账应该是强一致的。但是你仔细分析一下就会发现，事实并非如此。

举个例子，小王给小张转账 1000 元，小王的账户扣除了 1000，此时小张并不一定立刻就收到 1000 元。这里可能会存在一个不一致的时间窗口：小王的钱扣除了 1000 元，小张还没收

到 1000 元的时候。

另外一个例子，在 12306 网站买票的功能，也不是强一致的。

如果你在 12306 上发现一趟列车还剩余 10 张车票，你发起请求订了一张票，系统给你返回的可能是“正在排队，剩余 10 张票，现在有 15 人在购买”。

这时，你可能就需要去查询未完成订单，因为系统并没有给你及时返回订票成功或失败的结果。如果有人退了一张票，这张票也不会立即返回到票池中。这里明显也存在不一致的时间窗口。

但是，最终 10 张票只会卖给 10 个人，不可能卖给 11 个人，这就是最终一致性所谓的“最终所有数据都会同步”。

讲到这里，你对分布式系统的扩展性和一致性就很清楚了吧？接下来再给你介绍一个重要概念。

持久性

数据持久性（Data Durability）意味着数据一旦被成功存储就可以一直继续使用，即使系统中的节点下线、宕机或数据损坏也是如此。

不同的分布式数据库拥有不同级别的持久性。有些系统支持机器 / 节点级别的持久性，有些做到了集群级别，而有些系统压根没有持久性。

想要提高持久性，数据复制是较为通用的做法。因为把同一份数据存储在不同的节点上，即使有节点无法连接，数据仍然可以被访问。

在分布式数据处理系统中，还有一个持久性概念是消息持久性。什么意思呢？在分布式系统中，节点之间需要经常相互发送消息去同步以保证一致性。对于重要的系统而言，常常不允许任何消息的丢失。

分布式系统中的消息通讯通常由分布式消息服务完成，比如 RabbitMQ、Kafka。这些消息服务能支持（或配置后支持）不同级别的消息送达可靠性。消息持久性包含两个方面：

1. 当消息服务的节点发生了错误，已经发送的消息仍然会在错误解决之后被处理；
2. 如果一个消息队列声明了持久性，那么即使队列在消息发送之后掉线，仍然会在重新上线之后收到这条消息。

小结

在这一讲中，我们探讨了分布式处理系统的三个重要指标：扩展性，一致性和持久性。

结合前边提到的延迟性、可用性以及准确性，我们不难发现，这些设计分布式系统所要考虑的量化指标存在一定程度上的冲突。不可能有一个分布式处理系统在不牺牲某一指标的前提下，让每一个指标都达到最好。

作为优秀的系统架构师，我们一定要学会具体情况具体分析，找到最适合自己系统的指标，适当做出取舍。但是这一点说起来容易做起来难，到底该怎么取舍呢？你可以先思考一下这个问题，下一讲中我会结合 CAP 定理和你进一步讨论。

思考题

对于微信朋友圈的评论功能，你觉得哪种一致性模型更适用？为什么？

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 (33)

JohnT3e
2019-04-26

微信朋友圈评论主要由评论和后续回复组成：
首先，对于评论，评论内容对评论者而言应该要保证读写一致性（read-your-writes consistency），即评论一旦发出，那么对于该评论者无论在手机、网页还是其它城市应该都能看到其之前写的评论。而对于朋友圈可见的其它人来说，只要保证最终一致性（eventual consistency）就可以了（可能有时间要求），不同人的评论读取顺序无需和真实发生的顺序保持一致；
其次，对于评论的后续回复。回复内容对于回复者而言应该要保证读写一致性（read-your-writes consistency），而其它朋友圈可见的人一样，评论和回复内容应该按顺序被读取到，即需要保证一致前缀读（Consisten Prefix Reads）

共 2 条评论 >　　50

3SKarl
2019-04-26

老师，我不太明白弱一致性和最终一致性的区别在哪里，文章对弱一致性提到
“经过不一致时间窗口这段时间后，后续对该数据的读取都是更新后的值”
这句描述的不就是最终一致性么

是不是说弱一致性允许数据始终不一致，而要求最终结果一致的弱一执性叫做最终一致性

作者回复: 你好，这是个很棒的问题。简而言之，弱一致性是个很宽泛的概念，它是区别于强一致性而定义的。广义上讲，任何不是强一致的，而又有某种同步性的分布式系统，我们都可以说它是弱一致的。而最终一致性是弱一致性的一个特例，而且是最常被各种分布式系统用到的一个特例。其他的比如因果一致性、FIFO一致性等都可以看作是弱一致性的特例，不同弱一致性只是对数据不同步的容

忍程度不同，但是经过一段时间，所有节点的数据都要求要一致。

学习时也没必要抠字眼，重要的是理解它们的区别。这部分知识是为了后边讲CAP理论服务的，实际的工作中不会像考试考概念题一样让你背写这些一致性的定义。

如果你还有疑问，欢迎继续留言。

39

leeon
2019-04-26

朋友圈保证最终一致性即可，消息发布后，先保证"端"是可见的，等待网络请求后会确认最终有无发布成功，成功后最终其他人的timeline会收到

作者回复: 你的想法很好，但是有的场景下最终一致性还不够。试想这个场景，A发布了一张图片，B问他这是哪里，然后C回答B这里是北京。这个例子中，C的评论一定要在B之后，因为他俩有逻辑上的因果关系。所以微信朋友圈的评论要满足这样的因果一致性。因果一致性也是弱一致性的一个特例，感兴趣的话可以去多搜索点资料来学习 ~

共 3 条评论 >

20

hua168
2019-04-30

老师我想问一下，所谓的强一致性，它允许的误差是多少范围?
比如金融股票大厅显示屏数据，应该是强一致性的吧，全球显示误差不会超过0.2s吧?
这样强一致性怎达到？网络传输，路由处理都不止了，更何况还有网络延迟

作者回复: 谢谢你的提问！强一致性并没有误差可言的，强一致性简单地说指的就是如果更新一条数据，那所有用户读取数据的时候必须都看到这条更新了的数据。

在这里我也分享一个自己的面试经历。其实这个问题恰好我当年在面试Bloomberg的时候面到过，是一道系统设计题。问的大概是在设计他们家股票信息系统的时候，数据的更新写入量太大，用户也需要读取最新的股票资讯，该如何设计这套系统。最后和它家的tech lead讨论发现，原来他们的股票系统显示的延迟范围是1分钟左右，因为应用场景上普通股民并不会需要实时关心每秒钟股票价格的动态，更多的是关心大盘走势。而金融巨头在操作股票的时候更多只关心特定的几只股票，所以这些股票的价格通常对于他们来说会更新快一点。

所以说金融股票大厅上的数据应该不会是强一致性的，延时误差也应该没有你想象的那么少。

共 3 条评论 >

12

Flash

2019-05-06

老师，消息队列的持久性第二点不是太能理解，意思是说消息发送者发完消息后，接收者下线了，然后接收者上线仍能收到吗?

作者回复: 谢谢你的提问！没错，这里的持久性指的就是消息队列在接收到发送者发送的消息后，只要没有收到接收者的回应，就会一直尝试发送消息给接收者，直到收到回应为止。所以接收者下线了再上线仍是能收到消息队列发送的消息。当然中间如果超过了消息保留期限或者一定的重发次数也会消息队列也会停止发送。

7

涵

2019-04-27

对于微信朋友圈的评论功能我想最终一致性，保护因果一致性的特征就够了。从需求角度，微信朋友圈是一种社交工具，不具有critical的特性，数据一致的实效性要求不高。每个独立评论的先后顺序也不那么重要。重要的是对于某一条评论的评论必须显示在派生这条评论的原始评论之后，否则用户读起来会很混乱。从最低成本满足需求的角度，因此最终一致性加上因果一致性特征即可。

6

hallo128

2019-04-26

恩~其实我好像没有听懂~介绍了几个概念的定义，但还是不理解它们之间的联系和差别。

4

陈建斌红了..

2019-04-26

ap模型，先保证评论的人自己能看见，再保证被评论的人和别人能看见。

4

挖矿的小戈

2019-04-28

朋友圈采用最终一致性+因果一致性就好，只要保证评论及其派生评论是有序的，以保证其逻辑、语义上面不会出现混乱，场景对延时性要求不高；另外，有注意到手机端的微信和PC端的微信，会出现数据不一致的情况（特别是微信公众号订阅的文章，PC端少些文章）

3

Codelife

2019-04-26

以前曾经考虑过微信朋友圈这个一致性问题，最初认为是顺序一致性，仔细一想，如果是顺序一致性，成本太高，也没必要，对于微信朋友圈的评论来说，只有互为好友的人才能看到，评论也是互为好友的人，如果A发了朋友圈，所有人对A的评论，只有A才能完全看到，同时，B对A如果进行了评论，C对B的评论进行了评论，那么，对于A,B,C来说，C的评论肯定在B之后，只要保证互相能看到的评论一致即可，所以应该是因果一致性，至于如何的，暂时还没考虑

3

明翼

2019-04-26

笔记总结：扩展性包括水平和垂直，水平扩展是增加水平类似的机器打群架方式，垂直扩展是现有的每个人提升实力，提升整体实力；一致性，集群上的数据为了保证高可用性显然要存在不同机器上存多份，这就存在不同机器数量同步问题，要求同步造成才可以进行读写的就是最终一致性，中间准许有个不一致时间窗的就是弱一致性，时间窗更长但是最终会统一的就是最终一致性，比如发微博场景，还讲了数据持久性，消息的持久性，一般类似kafka之类消息总线系统可以配置各种持久性要求，比如要求所有主和副本都同步，要求只要发给主即可，有的发送不需要确认。

关于问题：朋友圈属于最终一致性，信息推送晚一会不是特别重要，所以最终一致性满足要求

作者回复: 很棒的总结。

对于思考题的答案，建议你再读读别的读者得留言，有的留言思考很深入～

3

Trane

2019-04-26

弱一致性，场景可以容忍一定时间的延迟。

2

西北偏北
2019-06-13

为了提高系统的吞吐量，我们需要水平扩展。
水平扩展后，消息，数据在集群中的传输，需要可靠的一致性保证。由于是分布式，无法保证像单机一样的强一致，只能保证最终一致性

分布式集群中，可能涉及通过通信沟通，去保证最终一致性，但通信可能会丢失。要么通过算法去保证，即便通信丢失，集群也能协商出一个一致的结果，比如paxos和raft算法。要么通过可靠的消息中间件来保证消息的可靠传达和消费。



1

roaming
2019-05-09

看老师评论里说需要最终一致性和因果一致性，感觉这个因果一致性和Java内存模型里的happens-before好像

> 作者回复: 谢谢你的留言！是的呢，在第九讲里面所讲到的线性一致性（Linearizability）其实也和Java Memory Model有关。如果在Java中修改的变量是volatile的话就会有这种线性一致性。



1

Geek_b04b12
2019-05-02

这一节课，怎么让我想起了阿里李云华 < 从零到架构师 > 中提到的CAP理论，这大规模数据和架构师之间的知识看来是想通的着呀

> 作者回复: 谢谢你的留言！有同感，其实当你学识更广之后会发现计算机中很多方向都是相辅相成的，并不会说某一块知识是相对独立起来的。



1

欢喜
2019-04-26

因果一致性。比如说A评论了，然后B评论了A的评论，只要保证A的评论在B对于A的评论之前就行了。



1

孙稚昊

2019-04-26

我觉得微信朋友圈只要最终一致就够了，没有对更新延迟那么高的要求

1

有铭

2019-04-26

我想请教老师，有些业务，不用关系数据库的话，如果处理分析需求(OLAP)?就我所知连google这样的公司某些时候还是必须用join来完成一些数据分析。比如下面这类需求，分布式系统里，用户表和订单表在不同的库里。请查出所有20-30岁用户的订单，并按照一定的规则进行数据统计。而20-30岁的用户数量可能非常庞大，十几万之多，这类的需求。非关系型的数据库，或者分布式系统如何做统计？统计现在才是大部分公司实施分布式数据处理的困难点，相反事务倒不是那么重要了，互联网企业的需求对强一致性要求没那么高

1

aof

2019-04-26

消息队列声明持久性的意思就是，比如kafka即可以作为消息队列，同时又可以将消息持久化到broker节点上?

> 作者回复: 谢谢你的提问！Broker在Kafka的系统里可以看作是不同消息队列的容器。而消息队列中消息的持久性指的是一般broker都会将消息replicate到不同的节点上，这时候消息才能够被consumer接收。而如果consumer没有acknowledge接收到消息的话，消息队列会一直尝试发送给consumer。

1

非同凡想

2020-12-15

微信朋友圈采用最终一致性，好友谁先看到都可以；
评论列表需要因果一致性，B评论A的朋友圈，C评论B的留言，存在因果关系。