01 | 什么是分布式数据库？

2020-08-10 王磊

分布式数据库30讲

进入课程 >

讲述：王磊

时长 17:37　大小 16.15M

你好，我是王磊，你也可以叫我 Ivan。

在这门课的第 1 讲，我想和你探讨一个最基本的问题：**什么是分布式数据库？**

回答这个问题，其实就是在给分布式数据库下定义。

分布式数据库和很多技术概念一样，没有权威机构来做这个定义，甚至对于哪些机构是权威机构，我们都很难有共识。

作为技术人员，我们常会提到一个概念，叫“事实标准”。当一个技术产品占据市场的主导位置时，它自然就成了同类产品的事实标准。例如，对于关系型数据库，可以说 Oracle

就是事实标准，因为所有数据库产品发布新版本时，都要拿自己的特性去和 Oracle 比一比。

很遗憾，分布式数据库作为一个新兴的基础软件，还没有一款产品占据“事实标准”的位置。既然没有参照，我们就自己动手，一起来定义分布式数据库这个概念吧。

由表及里、由外到内是人们认识事物的普遍规律，所以我们让也从内外部两个视角来观察。

外部视角：外部特性

外部视角，就是看看分布式数据库具备哪些特性，能解决什么问题。

通常，业务应用系统可以按照交易类型分为联机交易（OLTP）场景和联机分析（OLAP）场景两大类。OLTP 是面向交易的处理过程，单笔交易的数据量小，但是要在很短的时间内给出结果，典型场景包括购物、缴费、转账等；而 OLAP 场景通常是基于大数据集的运算，典型场景包括生成个人年度账单和企业财务报表等。

OLTP 与 OLAP 两种场景有很大的差异，所以很难在一款产品中完全满足两者的需求，因此在单体数据库时代就演化出了两个的不同技术体系，也就是两类不同的关系型数据库。向分布式架构演进后，两者在架构设计上也采用了完全不同的策略，很难在一个框架下说清楚。

所以，为了让你有更好的学习体验，**在这个课程中，我们先专注于讨论 OLTP 场景下的分布式数据库。**

说到这里，我想和你统一一下概念，如果没有特别说明，这个课程中出现的“数据库”都默认为“关系型数据库”，分布式数据库也都是指支持关系模型的分布式数据库。这就是说，NoSQL 不是我们要讨论的核心内容。

你可能会说，NoSQL 也很重要呀，MongoDB 多火呀。

NoSQL 当然很重要，MongoDB 确实也在一些细分场景中取得了成功。但是，从整体看，关系型数据库由于支持 SQL、提供 ACID 事务保障，显然具有更好的通用性，在更广泛的场景中无法被 NoSQL 取代。这一点通过 NoSQL 十余年的发展已经被证明。

事实上，分布式数据库的目标正是融合传统关系型数据库与 NoSQL 数据库的优势，而且已经取得了不错的效果。

定义 1.0 OLTP 关系型数据库

仅用“OLTP 场景”作为定语显然不够精准，我们来进一步看看 OLTP 场景具体的技术特点。

OLTP 场景的通常有三个特点：

写多读少，而且读操作的复杂度较低，一般不涉及大数据集的汇总计算；

低延时，用户对于延时的容忍度较低，通常在 500 毫秒以内，稍微放大一些也就是秒级，超过 5 秒的延时通常是无法接受的；

高并发，并发量随着业务量而增长，没有理论上限。

我们是不是可以有这样一个结论：**分布式数据库是服务于写多读少、低延时、高并发的 OLTP 场景的数据库**。

定义 2.0 + 海量并发

你可能会说这个定义有问题，比如 MySQL 和 Oracle 这样的关系型数据库也是服务于 OLTP 场景的，但它们并不是分布式数据库。

你的感觉没错，确实有问题。

那么，相对于传统关系型数据库，分布式数据库最大的差异是什么呢？答案就是分布式数据库远高于前者的并发处理能力。

传统关系型数据库往往是单机模式，也就是主要负载运行在一台机器上。这样，数据库的并发处理能力与单机的资源配置是线性相关的，所以并发处理能力的上限也就受限于单机配置的上限。这种依靠提升单机资源配置来扩展性能的方式，被称为垂直扩展（Scale Up）。

在一台机器中，随随便便就能多塞进些 CPU 和内存来提升提性能吗？当然没那么容易。所以，物理机单机配置上限的提升是相对缓慢的。

这意味着，在一定时期内，依赖垂直扩展的数据库总会存在性能的天花板。很多银行采购小型机或大型机的原因之一，就是相比 x86 服务器，这些机器能够安装更多的 CPU 和内存，可以把天花板推高一些。

而分布式数据库就不同了，在维持关系型数据库特性不变的基础上，它可以通过水平扩展，也就是增加机器数量的方式，提供远高于单体数据库的并发量。这个并发量几乎不受单机性能限制，我将这个级别的并发量称为“海量并发”。

听到这里你可能还要追问，这个“海量并发”到底是多大呢，有没有一个数字？

很遗憾，据我所知并没有权威数字。虽然理论上是可以找一台世界上最好的机器来测试一下，但考虑到商业因素，这个数字不会有什么实际价值。不过，我可以给出一个经验值，这个“海量并发”的下限大致是 10,000TPS。如果你有相关的经验，也欢迎你在评论区留言，我们一起讨论。

现在，基于这些理解，我们可以再得到一个 2.0 版本的定义：**分布式数据库是服务于写多读少、低延时、海量并发 OLTP 场景的关系型数据库**。

定义 3.0 + 高可靠

这个定义你觉得满意吗？

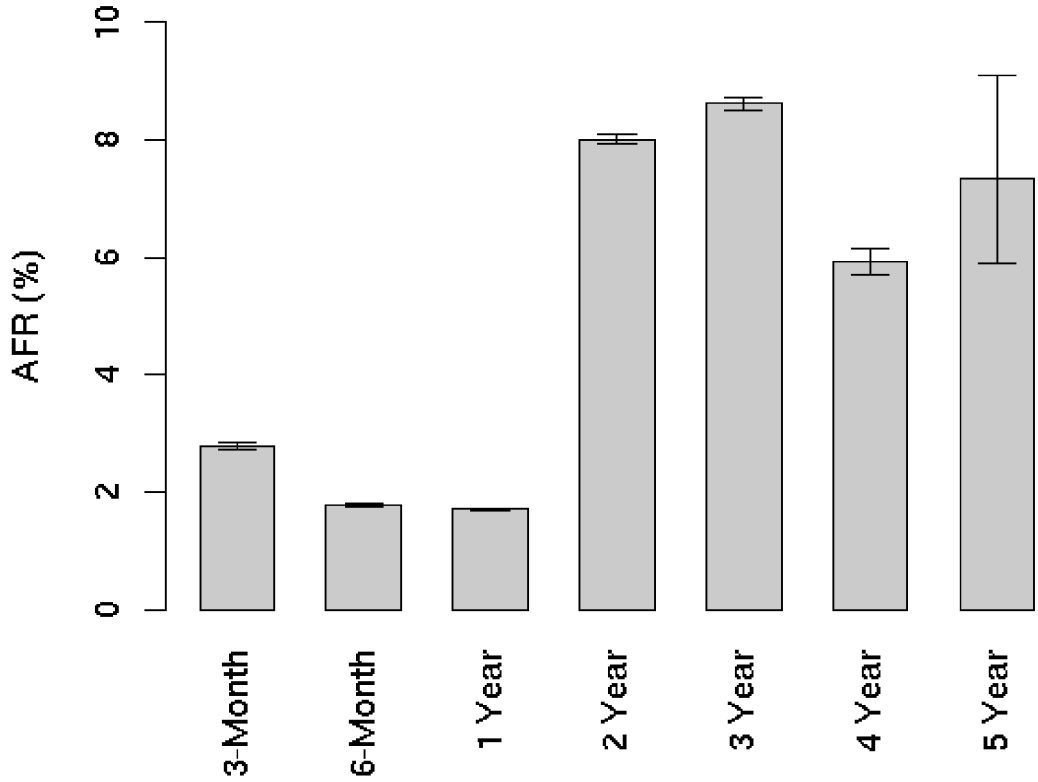
其实，这个 2.0 版本仍然有问题。

是不是没有海量并发需求，就不需要使用分布式数据库了呢？不是的，你还要考虑数据库的高可靠性。

一般来说，可靠性是与硬件设备的故障率有关的。

与银行不同，很多互联网公司和中小企业通常是采用 x86 服务器的。x86 服务器有很多优势，但故障率会相对高一些，坊间流传的年故障率在 5% 左右。

一些更加可靠的数据来自 Google 的论文*Failure Trends in a Large Disk Drive Population*，文中详细探讨了通用设备磁盘的故障情况。它给出的磁盘年度故障率的统计图，如下所示：



可以看到，前三个月会超过 2% 的磁盘损坏率，到第二年这个数字会上升到 8% 左右。

你可能会说，这个数字也不是很高啊。

但你要知道，对金融行业的关键应用系统来说，通常是要求具备 5 个 9 的可靠性（99.999%），也就是说，一年中系统的服务中断时间不能超过 5.26 分钟（365*24*60*（1-99.999%）≈ 5.26 ）。

而且，不只是金融行业，随着人们对互联网的依赖，越来越多的系统都会有这样高的可靠性要求。

根据这两个数字，我们可以设想一下，如果你所在的公司有四、五个关键业务系统，十几台数据库服务器，磁盘数量一定会超过 100 个吧？那么我们保守估计，按照损坏率 2% 来算，一年中就会碰到 2 次磁盘损坏的情况，要达到 5 个 9 的可靠性你就只有 5.26 分钟，能处理完一次磁盘故障吗？这几乎是做不到的，可能你刚冲到机房，时间就用完了。

我猜你会建议用 RAID（独立冗余磁盘阵列）来提高磁盘的可靠性。这确实是一个办法，但也会带来性能上的损耗和存储空间上的损失。分布式数据库的副本机制可以比 RAID 更好

地平衡可靠性、性能和空间利用率三者的关系。副本机制就是将一份数据同时存储在多个机器上，形成多个物理副本。

回到数据库的话题上，可靠性还要更复杂一点，包括两个度量指标，恢复时间目标（Recovery Time Objective, RTO）和恢复点目标（Recovery Point Objective, RPO）。RTO 是指故障恢复所花费的时间，可以等同于可靠性；RPO 则是指恢复服务后丢失数据的数量。

数据库存储着重要数据，而金融行业的数据库更是关系到客户资产安全，不能容忍任何数据丢失。所以，数据库高可靠意味着 RPO 等于 0，RTO 小于 5 分钟。

传统上，银行通过两种方法配合来实现这个目标。

第一种还是采购小型机和大型机，因为它们的稳定性优于 x86 服务器。

第二种是引入专业存储方案，例如 EMC 的 Symmetrix 远程镜像软件（Symmetrix Remote Data Facility, SRDF）。数据库采用主备模式，在高端共享存储上保存数据库文件和日志，使数据库近似于无状态化。主库一旦出现问题，备库启动并加载共享存储的文件，继续提供服务。这样就可以做到 RPO 为零，RTO 也比较小。

但是，这套方案依赖专用的软硬件，不仅价格昂贵，而且技术体系封闭。在去 IOE（IBM 小型机、Oracle 数据库和 EMC 存储设备）的大背景下，我们必须另辟蹊径。分布式数据库则是一个很好的备选方案，它凭借节点之间的互为备份、自动切换的机制，降低了 x86 服务器的单点故障对系统整体的影响，提供了高可靠性保障。

令人兴奋的是，这种单点故障处理机制甚至可以延展到机房层面，通过远距离跨机房部署。如此一来，即使在单机房整体失效的情况下，系统仍然能够正常运行，数据库永不宕机。

至此，我们得出一个 3.0 版本的定义，**分布式数据库是服务于写多读少、低延时、海量并发 OLTP 场景的，高可靠的关系型数据库**。

定义 4.0 + 海量存储

还有没有 4.0 版本呢？

你猜的没错，我们还要补充一些存储能力的变化。

虽然单体数据库依靠外置存储设备可以扩展存储能力，但这种方式本质上不是数据库的能力。现在，借助分布式的横向扩展架构，通过物理机的本地磁盘就可以获得强大的存储能力，这让海量存储成为分布式数据库的标配。

最后，我们终于得到一个 4.0 终极版本的定义，**分布式数据库是服务于写多读少、低延时、海量并发 OLTP 场景的，具备海量数据存储能力和高可靠性的关系型数据库**。

内部视角：内部构成

只通过外部视角来看分布式数据库，已经足够了吗？其实，具有相同的外在特性和功效，未必就是同样的事物。

举个例子，哥白尼刚提出“日心说”来反驳“地心说”的时候，要用到 34 个圆周来解释天体的运动轨迹；而 100 多年后，开普勒只用 7 个椭圆就达到了同样的效果，彻底摧毁了“地心说”。从哥白尼到开普勒，效果近似，简洁程度却大不一样，这背后代表的是巨大的科学进步。

因此，讲完分布式数据库的外部特性之后，我们还要从内部视角来进行观察。

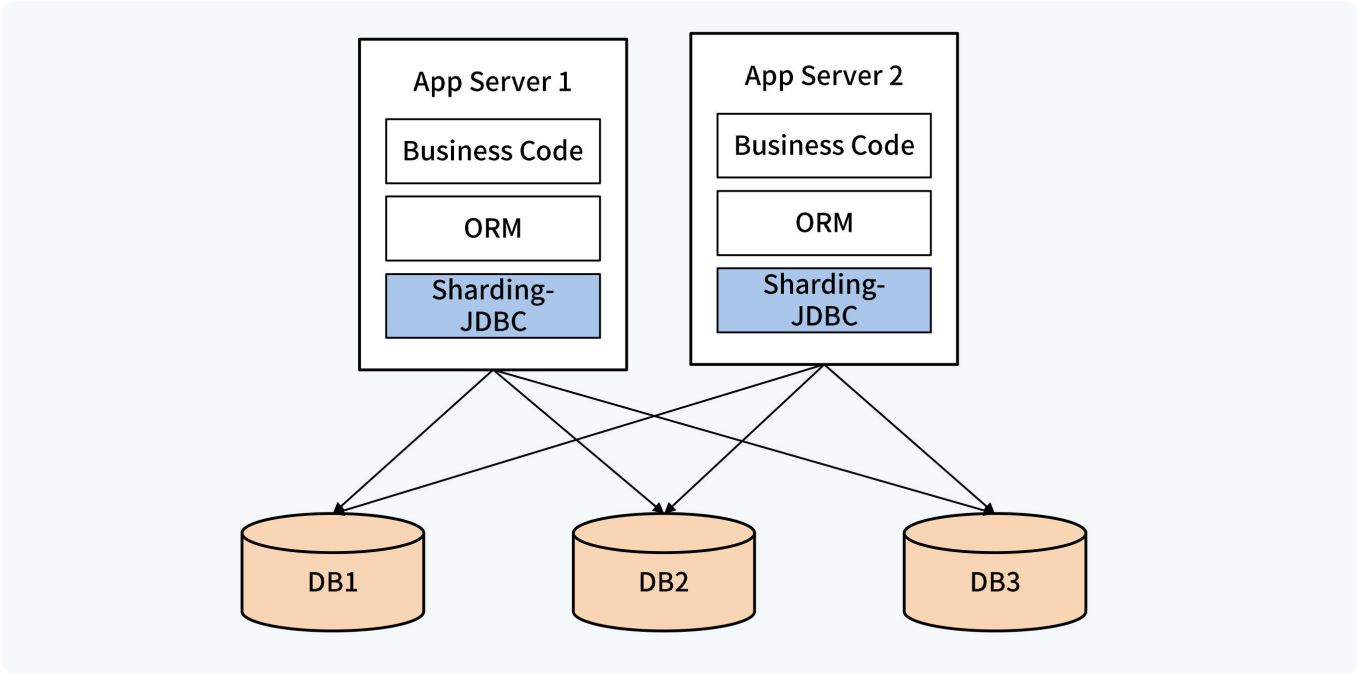
事实上，为了应对海量存储和海量并发，很多解决方案在效果上跟我们 4.0 版本的定义很相似。但是，它们向用户暴露了太多的内部复杂性。在我看来，对用户约束太多、使用过程太复杂、不够内聚的方案，不能称为成熟的产品。同时，业界的主流观点并不认为它们是分布式数据库，所以我们这门课也就不重点讨论了。

为了让你看清其中的差别，我将这些方案简单地分类介绍一下。

1. **客户端组件 + 单体数据库**

通过独立的逻辑层建立数据分片和路由规则，实现单体数据库的初步管理，使应用能够对接多个单体数据库，实现并发、存储能力的扩展。其作为应用系统的一部分，对业务侵入较为深。

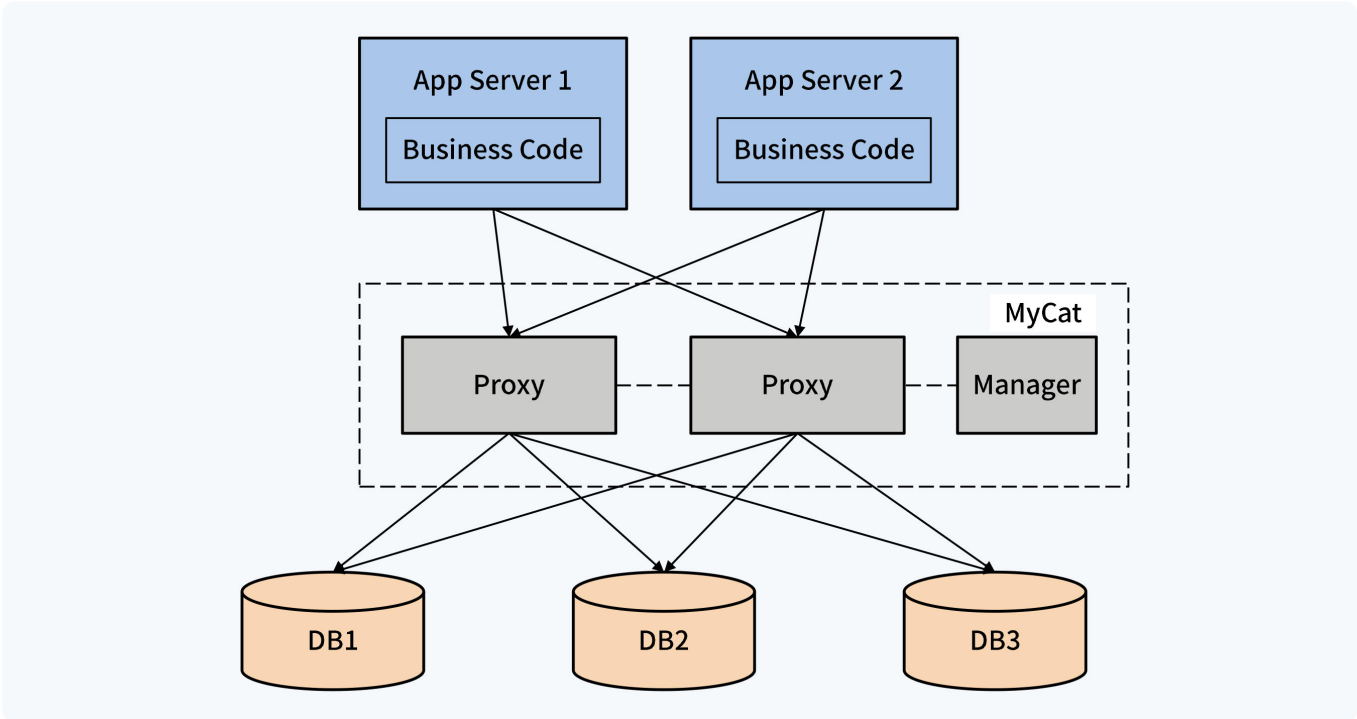
这种客户端组件的典型产品是 Sharding-JDBC。



2. **代理中间件 + 单体数据库**

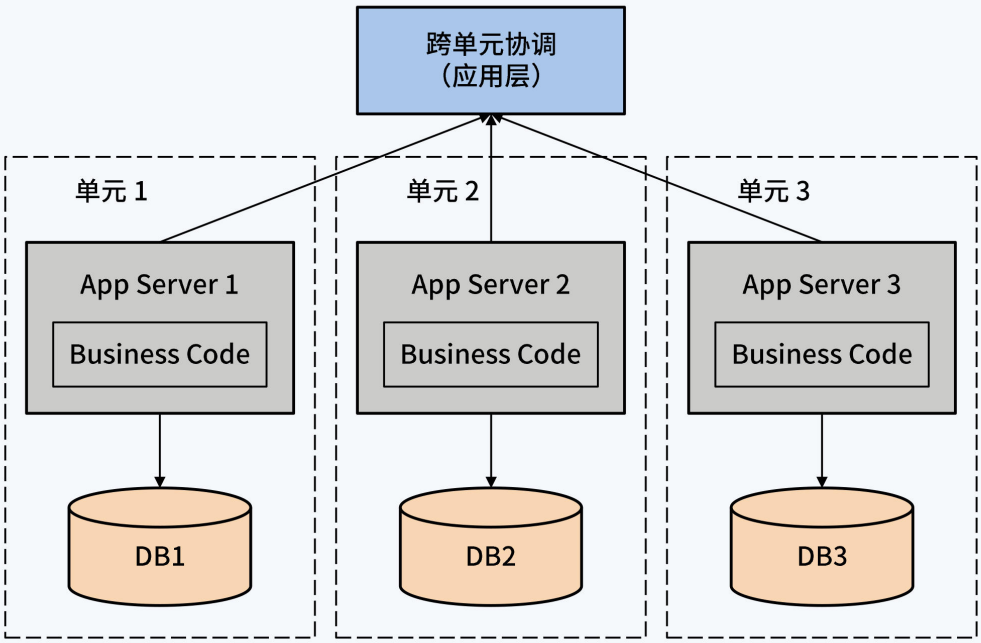
以独立中间件的方式，管理数据规则和路由规则，以独立进程存在，与业务应用层和单体数据库相隔离，减少了对应用的影响。随着代理中间件的发展，还会衍生出部分分布式事务处理能力。

这种中间件的典型产品是 MyCat。



3. **单元化架构 + 单体数据库**

单元化架构是对业务应用系统的彻底重构，应用系统被拆分成若干实例，配置独立的单体数据库，让每个实例管理一定范围的数据。例如对于银行贷款系统，可以为每个支行搭建独立的应用实例，管理支行各自的用户，当出现跨支行业务时，由应用层代码通过分布式事务组件保证事务的 ACID 特性。



根据不同的分布式事务模型，应用系统要配合改造，复杂性也相应增加。例如 TCC 模型下，应用必须能够提供幂等操作。

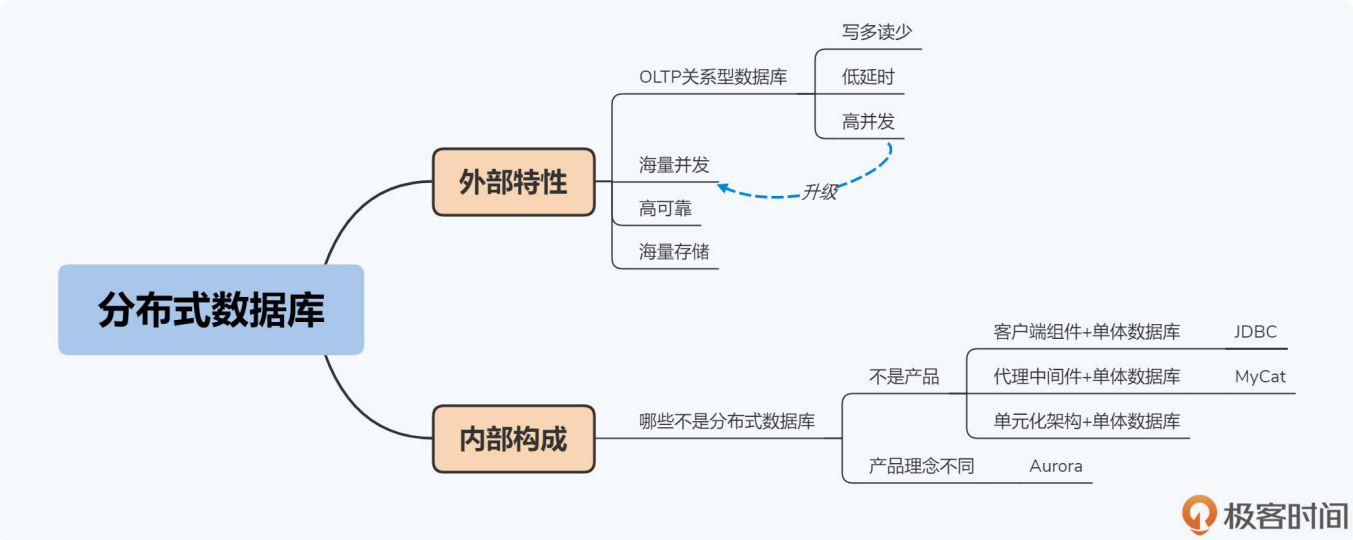
在分布式数据库出现前，一些头部互联网公司使用过这种架构风格，该方案的应用系统的改造量最大，实施难度也最高。

看过这三种方案，我相信你能够明白，它们共同的特点是单体数据库仍然能够被应用系统感知到。相反，分布式数据库则是**将技术细节收敛到产品内部，以一个整体面对业务应用。**

我猜，看到这里你一定很想知道，分布式数据库的内部架构到底长什么样呢？它跟这三种方案有什么区别呢？回答这个复杂的问题，就是我们这门课的使命了。这里你也可以先记下自己的答案，等学完这门课以后再回过头来做个对比，也是对自己学习效果的一种检验。

小结

好了，以上就是今天的主要内容了，我们来小结一下。



我们通过逐层递进的方式，勾勒出分布式数据库的六个外部特性，分别是**写多读少、低延时、海量并发、海量存储、高可靠性、关系型数据库。**

同时，也存在一些与分布式数据库能力近似的解决方案，这些方案的不足之处是都需要对应用系统进行一定的改造，对应用的侵入程度更深；其优势则在于可以最大程度利用单体数据库的稳定可靠，毕竟这些特性已经历经无数次的考验。

最后，我想就分布式数据库的名称做一些延伸。

“分布式数据库”在字面上可以分解为“分布式”和“数据库”两部分，代表了它是跨学科的产物，它的理论基础来自两个领域。这同时也呼应了产品发展的两条不同路径，一些产品是从分布式存储系统出发，进而增加关系型数据库的能力；另外一些产品是从单体数据库出发，增加分布式技术元素。而随着分布式数据库的走向工业应用，在外部需求的驱动下，这两种发展思路又呈现出进一步融合的趋势。

思考题

在准备分布式数据库这门课的过程中，有的朋友建议我讲讲 Aurora，但其实 Aurora 和这里说的分布式数据库还是有明显差别的，所以没有纳入正式课程。你了解 Aurora 或者它的同类产品吗？你觉得它和我所说的分布式数据库之间的差异是什么？那导致这种差异的原因又是什么呢？

欢迎你在评论区留言和我一起讨论，我会在答疑篇回复这个问题，如果感兴趣的同学很多，说不定会有加餐哦。

最后，如果你身边也有朋友对如何定义分布式数据库这个概念有困惑，欢迎你把今天这一讲分享给他，我们一起讨论。

提建议

上一篇 开篇词 | 为什么要学习分布式数据库？

下一篇 02 | 强一致性：那么多数据一致性模型，究竟有啥不一样？

精选留言 (18)

写留言

趁早

2020-08-12

我也不太能理解老师说的分布式数据库服务于写多读少的应用，我觉得不管是写多还是读多都可以应用分布式，关键是单体已经承担不了这么多请求了（不论读写），所以其实高并发就够了，不应该吧写多读少加入到分布式数据库的定义里面

展开

作者回复: 你好，之所以强调写多读少，因为写操作的负载只能是单体数据库的主节点上，是无法转移的；而读操作，如果对一致性要求不高可以转移到备节点，甚至在某些条件下还能保证一致性。就是说单体数据库可以通过一主多备解决读负载大的问题，而无需引入分布式数据库

5

开心哥

2020-08-13

aurora也用到一点投票机制，6个副本，半数以上就确认写入成功。也算沾点边。但没有

分片，不能多写，肯定不算分布式。

展开

作者回复: 你好，说的很好。不能多写，这点很关键，适用场景会有很大区别，所以这是一个重要标准。但是，因为Aurora是基于共享存储的，所以说它是分布式也不是没道理。我们定标准的目的，只是为了让学习的思路更清晰。

2

James

2020-08-11

可以更新快点吗？不够看的

展开

作者回复: 呵呵，别急哈，一周更三篇。

2

峰

2020-08-10

云dbms再分布式基础上更关注于计算存储分离后可独立扩展，甚至动态扩缩容，self-driven搞起来，更好卖了哈哈哈。当然这引发了不少问题，aurora提出了log is database的思想，降低写压力，snowflake通过建立中间分布式换存层，降低网络瓶颈等等。

展开

作者回复: 总结的很好，log is database是aurora类数据库的设计思想。

2

Monday

2020-08-15

建议老师画一张分布式数据库的全景图，后续的章节都可以在这张图上添砖加瓦，不然单节学完就是学完了而已，没有成体系，谢谢

展开

作者回复: 这个建议很好，我和编辑商量下，看看放在哪个部分

1

xy

2020-08-14

AWS aurora，阿里polarDB，腾讯CynosDB，架构上是比较类似的，计算存储分离。所有计算节点都访问存储节点上的同一份数据，也可以说是分布式架构的。建议老师加个餐讲讲

作者回复: 你好，你说的很对，三款产品都是类似的架构，还有华为的Taurus。这个架构的局限是写入不能横向扩展，但对于中小型应用也够了。

1

南国

2020-08-13

仅仅是简单看过Aurora的论文，算是有一点点了解。对于老师的问题，首先我觉得Aurora是提出一种新的单体架构以减少网络IO和同步阻塞，逻辑上可以看做一个庞大的单体数据库，用分布式来支持容错和高吞吐量。至于差异方面，我个人其实并没有看出差异。。理由如下：Aurora分片的方式是将数据库的总容量划分为固定大小的数据段，在每一段内存储数据，每一个段是一组机器（六个），个人认为算是支持分片；写多读少、低延时这...

展开

作者回复: 你好，首先你把Aurora的特点说得很清楚了。我们这里说有分别，是因为Aurora的特点是Share storage，计算节点垂直扩展，存储节点水平扩展，写入性能收单机资源的影响。

1

赵见跃

2020-08-11

老师好，MyCat现在发展的怎么样，性能上不知道，是否改善了。谢谢。
再请教一个问题，阿里的PolarDB是分布式数据库吧？它采用的是哪种方案呢？靠谱不？谢谢。

作者回复: 你好，首先我对MyCat的最新发展了解的不多。个人认为随着分布式数据库的发展，MyCat这类中间件的市场会越来越小。当然，它的使用场景也可能转向对异构数据库的支持，就像Presto那样。
PolarDB和Aurora的架构类似，计算与存储分离，计算节点垂直扩展，存储节点水平扩展。这意味着它的写入能力是有上限的，但是因为简化了日志的存储，和其他一些优化，单节点能力比普通的MySQL好强很多。

1 1

kylexy_0817
2020-08-11

一直没接触过分布式关系型数据库，感觉就是把客户端或中间件的方案直接作为数据库服务端的特性组件，把分库分表做得更为自动化

展开

作者回复: 分布式数据库和分库分表的最大区别在于，分布式数据库的使用体验是非常接近关系型数据库的，不需要应用进行额外的控制，这就降低了代码的开发难度。而分库分表方案在分布式事务和跨节点查询等方面，通常支持的都不好。

1

万丰路甲一号
2020-08-11

希望老师能够讲讲交易场景下，交易代码配合分布式数据库而做出的交易补偿或者数据回放等等

作者回复: 你好，如果需要交易代码配合做出补偿和回放，这很可能意味着它不是分布式数据库。在分布式数据库成熟前，确实有不少应用代码配合单体数据库的方式。这类应用代码也会被抽离出来形成独立的框架，如果你感兴趣，可以研究下阿里的SOFA。

1

赵赟
2020-08-17

从老师的措辞上来看感觉老师水平很高，相对于其它老师学术范更浓一点。

作者回复: 谢谢:)

FF
2020-08-15

可以从实际场景说下Aurora和分布式存储的应用的差别吗

作者回复: 你好，Aurora还是关系型数据库，而分布式存储系统范围较广，比如HBase这样的分布式键值系统。两者在功能上有很大差异。


南国
2020-08-13

想问问老师，根据文中的定义，BigTable就算是属于特殊的（代理中间件 + 单体数据库（分布式文件系统））吗？它毕竟是靠Chubby来作为一个中间层的，不过数据的获取是直接与文件系统中交互完成的。我一直以为它也算是分布式数据库。

展开

作者回复: 你好，BigTable是分布式键值系统，并不属于分布式数据库。因为这里所说的分布式数据库是分布式架构实现的关系型数据库。当然它的底层依赖一个分布式文件系统，所以看上去也是分为两层，但它的职能和数据库差别很大，推荐你看下04讲，关注其中的PGXC风格分布式数据库，有问题可以接着留言，我们继续聊。


扩散性百万咸面包
2020-08-12

不是很懂为什么说Aurora不是分布式数据库。Aurora通过计算和存储服务分离，使得亚马逊的云存储服务更加高效和易于使用。算是Newsql中比较成功的一个流派了。

作者回复: 你好，这里没有纳入Aurora，是因为它架构不支持对写入负载的横向扩展。当然，对很多小规模的应用来说也足够了，所以这不影响它取得商业成功。


赵见跃
2020-08-11

老师您好，基于OLAP使用场景的分布式关系型数据库，都有哪些呀？谢谢，

作者回复: 这个还是挺多的，最典型的是MPP架构数据库，比如Greenplum和华为的GaussDB 200，它们的内核都使用了PostgreSQL。此外，还有Vertica。OLAP不再强调事务的支持，如果弱化了对数据更新的要求，很多大数据生态的产品就都可以纳入进来，比如Clickhouse，Hive on spark，甚至Kylin都可以算是广义的OLAP分布式数据库


1


qinsi
2020-08-11

请问OLTP中“写多读少”中的“多”和“少”是指请求的数量还是请求的大小，或是什么其他的指标呢？都说互联网应用的数据请求通常是“读多写少”，所以才会有一主多从读写分离、全量数据缓存等解决“读”的问题的扩容手段。如果说的是同一个指标的话，是不是意味着分布式数据库不适合互联网应用？

展开

作者回复: 你好，首先，OLTP的写多读少，是指请求数量。互联网确实可以通过通过一主多满足“读多写少”的场景，但前提是对读对一致性要求较低。而在金融场景中，很多读操作依然是无法在备库运行的，就是一致性不满足要求。所以，我觉得对互联网也不能一概而论，还是要区分场景。有关一致性的话题，我在02/03会具体讲解，你可以先看看，我们再讨论。

_____Harvey凝枫◆...

2020-08-10

没主动了解过Aurora，一直把它当成常见的分布式数据库看待的，他很特殊么？

作者回复: □

嗯，AWS的拳头产品，云原生数据库，还是很有特点的。

gtp

2020-08-10

期待已久

展开

作者回复: 谢谢，希望不辜负你的期待