



下载APP



08 | 基础篇大串讲：重难点回顾+思考题答疑+知识全景图

2020-08-26 王磊

分布式数据库30讲

[进入课程 >](#)



讲述：王磊


时长 15:58 大小 14.64M



你好，我是王磊，你也可以叫我 Ivan。

这一讲是我们课程的答疑篇，我会集中讨论前 7 讲布置的思考题，以及留言区中大家关注的一些内容。

第 1 讲：分布式数据库的定义

在 [第 1 讲](#) 中，我们通过层层递进式的分析，给这门课程要讨论的“分布式数据库”下了一个定义：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，具有  数据存储能力和高可靠性的关系型数据库。在“内部构成”这一节，我们还着重讨论了几种不属于分布式数据库的解决方案。

在这一讲的思考题部分，我们聊到了 Aurora，我说“Aurora 和这里说的分布式数据库还是有明显差别的”，想看看大家的理解。在留言中，我看到有些同学是持不同观点的，理由是 Aurora 也基于分布式存储的。

那么，为什么我说它不是分布式数据库呢？主要原因就是 Aurora 依然是不支持写入能力的水平扩展。

Aurora 是亚马逊推出的云原生数据库，它采用计算与存储分离的思想，计算能力垂直扩展，存储能力水平扩展。究其原因，它的存储系统是直接架设在自家的分布式存储系统（S3）之上的；而计算节点仍然是单节点，所以是垂直扩展。当然 Aurora 也像 MySQL 一样是支持一写多读的，根据亚马逊的官方说明，可以配置 15 个备节点来分流读操作的压力。由于 Aurora 的元数据会缓存在主节点上的，在发生变更时，主备同步数据有一个小的延迟（小于 100 毫秒），这就造成备节点不能承接写入功能，读也不能保证严格的数据一致性。

我们在定义中强调了海量并发和写多读少，这其实就是要求分布式数据库的写入能力必须是可水平扩展的。

“开心哥”的留言中，提到了 Aurora 是不能支持多写的，准确地抓住了它与 NewSQL 的重要差别。而“南国”同学的留言中还提到了 Aurora 的论文。这篇论文是 2017 年，亚马逊在 SIGMOD 上发表的，论文题目叫做“[Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases](#)”，其中披露了系统架构的设计细节，推荐有兴趣的同学阅读。其实阅读顶会论文是非常不错的学习方法，给“南国”同学点赞，希望大家也尝试一下。

最后，“xy”同学的留言还提到了另外两款同架构的产品，阿里 polarDB，腾讯 CynosDB，说明“xy”同学很关注对系统的横向比较，这也是非常好的学习习惯。我这里再补充一点，华为的 Taurus 也采用了类似 Aurora 的架构。

第 2 讲：数据一致性

🔗第 2 讲中，我们首先明确了强一致性包含数据一致性和事务一致性两个方面，而后展开介绍了数据一致性。我们的讲解方式是先给出一个分析框架，也就是状态和操作双视角，并从状态视角引出了最终一致性这个概念。而后，我们在最终一致性的基础上介绍了 5 种不同强度的一致性模型，其中线性一致性和因果一致性是分布式数据库中普遍应用的。

思考题部分则是“你觉得 Paxos 这个一致性协议和数据一致性又是什么关系呢？”

这个答案嘛，很显然它们是不同的概念。可为什么不同的概念，都叫做**一致性**呢？就像“峰”同学说的，这个问题其实是翻译造成的。数据一致性对应是 Consistency，而一致性协议对应的则是 Consensus，这个单词更多时候被翻译成共识，就是我们常说的共识算法。

我认为，Paxos 本质上是一种复制协议，约定了副本之间的同步策略，就像我们谈到的最终一致性，同样也只是描述了副本之间同步情况。再看看我们具体介绍的 5 个数据一致性模型，它们都在多副本的基础上又约定了读写策略，所以这两点都是一致性模型 (Consistency Model) 必不可少的内容。

我在留言中发现有的同学对 Paxos 这样的共识算法认识很深刻，谈了多副本的一致性，讲得很好，但是会忽略了读写策略的作用。“chenchukun”和“tt”同学的留言则抓住了这两个点，点赞。

第 3 讲：事务一致性

🔗第 3 讲谈的事务一致性也是强一致性的组成部分，它具体又细化为 ACID 四个特性，其中的一致性比较宽泛，持久性的实现机制比较稳定，而原子性在分布式架构下面临挑战，最后的隔离性则非常复杂。即使在单体数据库下，工业界也没找到公认的处理隔离性问题的完美方法，很难实现最高级别的可串行化。所以，在分布式架构下，多数产品依然需要在性能与正确性之间进行权衡。

关于原子性和隔离性，我们还有比较多的篇幅展开讨论，所以课程的最后我留了一道关于持久性的思考题，就是预写日志 (WAL) 写成功，但是数据表写失败，要怎么处理？

在留言中，我发现很多同学都对 WAL 有深刻的认识，也都了解基于日志恢复数据的运作原理。其实，我这个问题是想让大家思考，联机写入的那一刻，除了记录 WAL，数据库还干了什么。这也是一个与 WAL 有关的设计，也很有意思。

事实上，对大多数的数据库来说，实时写入数据时，并不是真的将数据写入数据表在磁盘中的对应文件里，因为数据表的组织形式复杂，不像 WAL 那样只是在文件尾部追加，所以 I/O 操作的延迟太长。因此，写入过程往往是这样的，记录 WAL 日志，同时将数据写入内存，两者都成功就返回客户端了。这些内存中的数据，在 Oracle 和 MySQL 中都被称

为脏页，达到一定比例时会批量写入磁盘。而 NewSQL 所采用的 LSM-Tree 存储模型也是大致的思路，只不过在磁盘的数据组织上不同。

写入内存和 WAL 这两个操作构成了一个事务，必须一起成功或失败。

第 4 讲：两种架构风格

🔗第 4 讲我们谈了分布式数据库的两种架构风格 NewSQL 和 PGXC。PGXC 是从代理中间件演化而来，以单体数据库作为数据节点，它的优势是工程实现更稳定。NewSQL 则是以分布式键值系统为基础，引入了很多新技术，这些技术都会在我们的课程中逐步介绍。NewSQL 的代表系统是 Google 的 Spanner，而它的优势就是架构的先进性。

其实关于架构风格的讨论，往往是百家争鸣，各持观点，所以我们的思考题也是一个开放性话题，请大家聊聊自己熟悉的分布式数据库，或者其他分布式系统的架构。

在留言区，“xy”和“赵见跃”同学都提到了 TDSQL，它是不是也属于 PGXC 风格呢？我认为目前腾讯输出的 TDSQL 还不是典型的 PGXC，因为它没有全局时钟，也没有等效的设计去解决全局一致性问题。当然，说它不是，我也是有点纠结的，在 2019 年 TDSQL 的技术演讲中，腾讯的研发人员深入地分析了缺失全局时钟带来的一致性问题的，同时也提及了正在进行的技术尝试。所以，我相信 TDSQL 很快会在新版本中增加类似的特性。

“南国”同学还提出了一个新问题：NewSQL 与 PGXC 的界限似乎很模糊，是不是差别就在存储层面，NewSQL 只能存储，而 PGXC 是完整的数据库呢？我认为这只是一个表象，最关键的差异其实是分片设计，或者说是两种架构对数据组织形式上的根本差别。PGXC 的数据是相对固定的，而 NewSQL 的数据是能够更加灵活移动的，移动意味着解锁了数据与节点的关系，有点像灵魂和躯体的关系。如果灵魂不被限制在一个躯体里，那是不是就可以实现永生。解锁了数据与节点的依赖关系，系统也更加鲁棒。总的来说，我认为能够适应变化，在各种意外情况下，都能生存下来，这是设计分布式系统的核心思想。

第 5 讲：全局时钟

🔗第 5 讲，我们介绍了全局时钟的不同实现方式，包括物理时钟和逻辑时钟两种方式，物理时钟的难点首先是要做到足够高的精度，其次是在使用时如何处理时钟误差，学术一点的说法叫做时钟的置信区间。逻辑时钟实际上是混合逻辑时钟，还是会引入物理时钟作为参考，但主要通过逻辑控制来保证时钟的单调递增。有同学问是不是可以不用物理时钟，

我要说的是，对于多时间源是不行的，因为这样会造成不相关事件的时钟偏差太大，也就是偏序拼接的全序失真太大。如果是单时间源的混合逻辑时钟，它的好处是不用处理误差，简化了其他模块的设计。而 HLC 这样多时间源的混合逻辑时钟，则依然有时钟误差的问题。

这一讲的思考题是让大家思考一下“时间对于分布式数据库的影响是什么？”我发现大家的留言对这个问题的讨论并不多。其实，时间在很多分布式系统都是存在的，比如 HBase 对于各节点的时钟偏移也是有限制，只不过它的容忍度更高，可以达到几十秒。而在分布式数据库中与时间有关的功能主要体现在事务并发控制，比如 MVCC、读写冲突。既然留言讨论不多，我这里就先不做点评，卖个关子，在第 11 讲、第 12 讲中我们再来详细聊聊。

第 6 讲：数据分片

🔗第 6 讲，我们介绍了分布式数据库中一个非常重要的概念“分片”。分片机制的两个关键点是分片策略和分片调度机制。分片策略包括 Hash 和 Range，调度机制则包括静态和动态两种。分片机制的实现和架构有很大的关系，PGXC 架构基本上都是静态分片，是以 Hash 分片为主，有的产品也同时支持 Range 分片。关于 NewSQL 架构，我们主要介绍了最有代表性的动态 Range 分片。

这一讲的思考题，就是在问分片元数据的存储方案。

分析这个问题，首先要看元数据会不会变更，比如静态分片就不会变更，那么就可以把它复制多份部署在所有工作节点上，如果会变更，那就要考虑变更带来的多副本一致性问题，这里其实是和后面的 07 讲相呼应的。现在读完 07 讲，你自然应该知道，如果是少数节点集中存储元数据，那么可以采用 Paxos 协议保证一致性。如果是 P2P 架构，因为节点规模太大，那就适合采用 Gossip 协议。设计的权衡点主要是在于节点规模大小对传播效率的影响。

“开心哥”和“真名不叫黄金”两位同学都回答其中的一种情况，就是基于 etcd 或 PD（基于 etcd）来存储元数据，而 etcd 是 Raft 协议的开源实现。

第 7 讲：数据复制

🔗第7讲，我们讨论的话题是数据复制，这和分片一样是非常基础和重要的内容。这一讲我们介绍了两个知识点，其中第一个就是分片元数据的存储方案，刚刚我们已经说过了，第二个知识点是数据复制的效率问题。Raft 由于顺序投票的限制，在复制效率上比 Paxos 稍差。但是因为 Raft 有高质量的开源实现项目 etcd，而 Paxos 因为算法复杂没有稳定的开源能实现，所有 TiDB 和 CockroachDB 还是选择了 Raft 协议。同时，TiDB 和 CockroachDB 采用了 Multi Raft 的方式，让多分片并行处理提升性能。两者在 Raft 协议实现上也进行了若干改进。这些改进思路很有普适性，一些独立的 Raft 项目也同样实现了，比如 SOFA-JRaft。

这一讲的思考题，我们讨论的是分布式数据库的存储上限。你一定有点疑惑，既然分布式数据库是一个水平扩展的系统，可以不断地增加节点。那么为什么还有存储上限呢？事实上，不仅分布式数据库，绝大多数分布式存储系统都是有上限的。因为有了这个限制，可以简化系统架构设计，而这个上限当然也是一个很大的数值，能够满足绝大多数业务场景的需求。

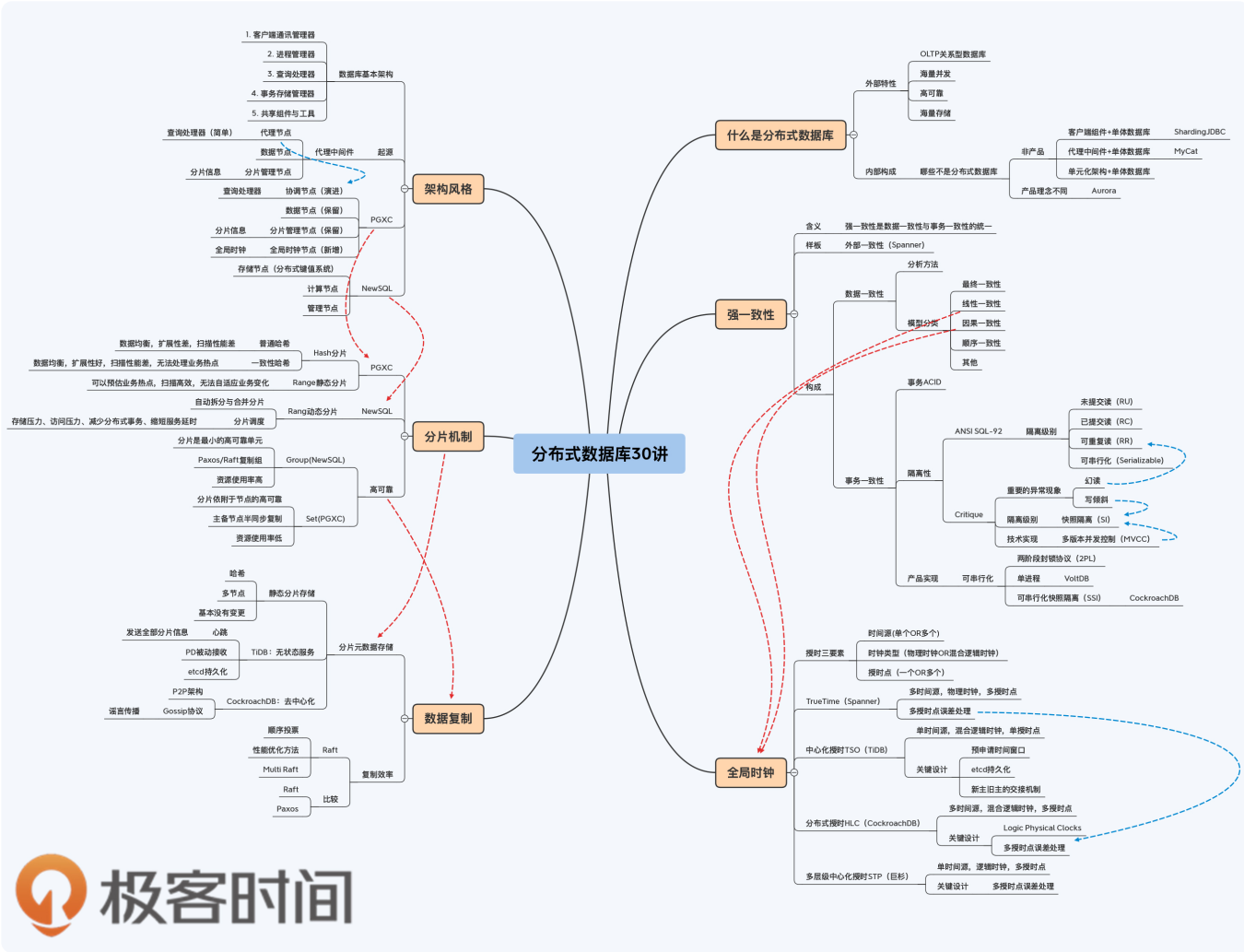
以 CockroachDB 为例，它的存储容量大致是 4EB，而这个限制是由元数据的存储方式决定的。

在 CockroachDB 中存储分片元数据的数据结构叫做 Meta ranges，它是一个两层索引结构，第一层 Meta1 存储了第二层 Meta2 的地址，第二层 Meta2 则指向了具体分片。每个节点会保存 Meta1 的定位，而且 Meta1 是不会分拆的，这样就更好的稳定性。Meta1 和 Meta2 的长度都是 18 位，所以 CockroachDB 中最多只能有 2^{36} 个分片。CockroachDB 默认分片初始大小是 64M，那么可以算出一个总存储量是 4EB， $2^{36} \times 64M$ 。从这个意义上说，CockroachDB 的最大存储容量是 4EB。当然，如果分片增大整体容量还会增加，但第 6 讲我们介绍过分片过大是有副作用的，所以不能无限制增加，系统的容量还是有上限的。

小结

最后，要特别感谢“Monday”同学，他建议我们增加一张分布式数据库的全景图，让知识的组织更加系统。我觉得这是个好主意，和编辑商量了一下，最后决定在每个答疑篇都会增量补充这个全景图，在最后的第 30 讲大家就能看到完整的全景图了。这样安排还有一个好处，就是帮助大家阶段性地复习前面课程。

分布式数据全景图 1/4



如果你对今天的内容有任何疑问，欢迎在评论区留言和我一起讨论。要是你身边的朋友也对分布式数据库这个话题感兴趣，你也可以把今天这一讲分享给他，我们一起讨论。

学习资料

Alexandre Verbitski et al.: *Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases*

提建议

更多课程推荐

程序员的数学基础课

在实战中重新理解数学

黄申

LinkedIn 资深数据科学家



涨价倒计时 🕒

今日秒杀 **¥79**, 9月11日涨价至 **¥129**

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 数据复制：为什么有时候Paxos不是最佳选择？

下一篇 09 | 原子性：2PC还是原子性协议的王者吗？

精选留言 (4)

写留言



托尼斯威特

2020-08-27

谢谢老师的总结。

上手分布式数据库之前，我想请问几个基本的问题，

需要ORM框架吗？MyBatis Hilbernite 还是别的什么？

...

展开 ∨

作者回复: 你好，首先要说你提的问题很好，我猜这也是很多同学的疑问。事实上，分布式数据库在功能上没有太多神秘的地方，我们在开篇词中提到过，分布式数据库就是分布式架构实现的关系型数据库。所以说这些数据库的典型特性，专栏中介绍的分布式数据库几乎都可以支持。

**哈德韦**

2020-08-29

老师好，看AWS的最新文档，似乎Aurora也支持多写了：<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-multi-master.html>。是不是说现在的Aurora更加是一个分布式数据库了呢？

**扩散性百万咸面包**

2020-08-26

问一下，文章中说数据库普遍写入数据都是WAL+内存写。那这种情况下，B-Tree和LSM tree还会有那么大的性能差异吗？B-Tree普遍要经过几次搜索，可能还有回表。而LSM Tree只要往有序的文件中写入数据，保证有序即可？这是两者差异的主要原因吗？

展开 ∨

**扩散性百万咸面包**

2020-08-26

PGXC 的数据是相对固定的，而 NewSQL 的数据是能够更加灵活移动的，移动意味着解锁了数据与节点的关系，有点像灵魂和躯体的关系。如果灵魂不被限制在一个躯体里，那是不是就可以实现永生。解锁了数据与节点的依赖关系，系统也更加鲁棒。

对文章中的这一点，PGXC和NewSQL的区别表示疑问。PGXC如果加入动态调度的组件，是否也可以实现Range 动态调度呢？据我的理解普遍PGXC和NewSQL的最大区别...

展开 ∨

作者回复: PGXC当然也有演进的机会，增加动态调度，不过那已经不是我们现在所说的PGXC架构了。

SQL无法水平扩展？这个怎么理解呢？

