15 | 高性能数据库集群：分库分表

2018-05-31 李运华

15 | 高性能数据库集群：分库分表

李运华

00:00 / 16:42

上期我讲了“读写分离”，读写分离分散了数据库读写操作的压力，但没有分散存储压力，当数据量达到千万甚至上亿条的时候，单台数据库服务器的存储能力会成为系统的瓶颈，主要体现在这几个方面：

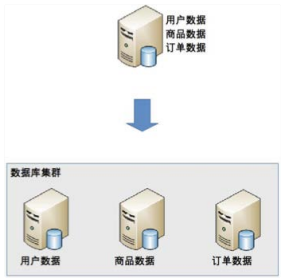
- 数据量太大，读写的性能会下降，即使有索引，索引也会变得很大，性能同样会下降。
- 数据文件会变得很大，数据库备份和恢复需要耗费很长时间。
- 数据文件越大，极端情况下丢失数据的风险越高（例如，机房火灾导致数据库主备机都发生故障）。

基于上述原因，单个数据库服务器存储的数据量不能太大，需要控制在一定的范围内。为了满足业务数据存储的需求，就需要将存储分散到多台数据库服务器上。

今天我来介绍常见的分散存储的方法“分库分表”，其中包括“分库”和“分表”两大类。

业务分库

业务分库指的是按照业务模块将数据分散到不同的数据库服务器。例如，一个简单的电商网站，包括用户、商品、订单三个业务模块，我们可以将用户数据、商品数据、订单数据分开放到三台不同的数据库服务器上，而不是将所有数据都放在一台数据库服务器上。



虽然业务分库能够分散存储和访问压力，但同时也带来了新的问题，接下来我进行详细分析。

1.join操作问题

业务分库后，原本在同一个数据库中的表分散到不同数据库中，导致无法使用SQL的join查询。

例如：“查询购买了化妆品的用户中女性用户的列表”这个功能，虽然订单数据中有用户的ID信息，但是用户的性别数据在用户数据库中，如果在同一个库中，简单的join查询就能完成；但现在数据分散在两个不同的数据库中，无法做join查询，只能采取先从订单数据库中查询购买了化妆品的用户ID列表，然后再到用户数据库中查询这批用户ID中的女性用户列表，这样实现就比简单的join查询要复杂一些。

2.事务问题

原本在同一个数据库中不同的表可以在同一个事务中修改，业务分库后，表分散到不同的数据库中，无法通过事务统一修改。虽然数据库厂商提供了一些分布式事务的解决方案（例如，MySQL的XA），但性能实在太低，与高性能存储的目标是相违背的。

例如，用户下订单的时候需要扣商品库存，如果订单数据和商品数据在同一个数据库中，我们可以使用事务来保证扣减商品库存和生成订单的操作要么都成功要么都失败，但分库后就无法使用数据库事务了，需要业务程序自己来模拟实现事务的功能。例如，先扣商品库存，扣成功后生成订单，如果因为订单数据库异常导致生成订单失败，业务程序又需要将商品库存加上；而如果因为业务程序自己异常导致生成订单失败，则商品库存就无法恢复了，需要人工通过日志等方式来手工修复库存异常。

3.成本问题

业务分库同时也带来了成本的代价，本来1台服务器搞定的事情，现在要3台，如果考虑备份，那就是2台变成了6台。

基于上述原因，对于小公司初创业务，并不建议一开始就这样拆分，主要有几个原因：

- 初创业务存在很大的不确定性，业务不一定能发展起来，业务开始的时候并没有真正的存储和访问压力，业务分库并不能为业务带来价值。
- 业务分库后，表之间的join查询、数据库事务无法简单实现了。
- 业务分库后，因为不同的数据要读写不同的数据库，代码中需要增加根据数据类型映射到不同数据库的逻辑，增加了工作量。而业务初创期间最重要的是快速实现、快速验证，业务分库会拖慢业务节奏。

有的架构师可能会想：如果业务真的发展很快，岂不是很快就又要进行业务分库了？那为何不一开始就设计好呢？

其实这个问题很好回答，按照我前面提到的"架构设计三原则"，简单分析一下。

首先，这里的"如果"事实上发生的概率比较低，做10个业务有1个业务能活下去就很不错了，更何况快速发展，和中彩票的概率差不多。如果我们每个业务上来就按照淘宝、微信的规模去做架构设计，不但会累死自己，还会害死业务。

其次，如果业务真的发展很快，后面进行业务分库也不迟。因为业务发展好，相应的资源投入就会加大，可以投入更多的人和更多的钱，那业务分库带来的代码和业务复杂的问题就可以通过增加人来解决，成本问题也可以通过增加资金来解决。

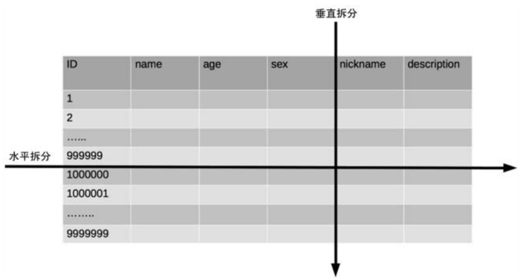
第三，单台数据库服务器的性能其实也没有想象的那么弱，一般来说，单台数据库服务器能够支撑10万用户量量级的业务，初创业务从0发展到10万级用户，并不是想象得那么快。

而对于业界成熟的大公司来说，由于已经有了业务分库的成熟解决方案，并且即使是尝试性的新业务，用户规模也是海量的，这与前面提到的初创业务的小公司有本质区别，因此最好在业务开始设计时就考虑业务分库。例如，在淘宝上做一个新的业务，由于已经有成熟的数据库解决方案，用户量也很大，需要在一开始就设计业务分库甚至接下来介绍的分表方案。

分表

将不同业务数据分散存储到不同的数据库服务器，能够支撑百万甚至千万用户规模的业务，但如果业务继续发展，同一业务的单表数据也会达到单台数据库服务器的处理瓶颈。例如，淘宝的几亿用户数据，如果全部存放在一台数据库服务器的一张表中，肯定是无法满足性能要求的，此时就需要对单表数据进行拆分。

单表数据拆分有两种方式：垂直分表和水平分表。示意图如下：



为了形象地理解垂直拆分和水平拆分的区别，可以想象你手里拿着一把刀，面对一个蛋糕切一刀：

- 从上往下切就是垂直切分，因为刀的运行轨迹与蛋糕是垂直的，这样可以把蛋糕切成高度相等（面积可以相等也可以不相等）的两部分，对应到表的切分就是表记录数相同但包含不同的列。例如，示意图中的垂直切分，会把表切分为两个表，一个表包含ID、name、age、sex列，另外一个表包含ID、nickname、description列。
- 从左往右切就是水平切分，因为刀的运行轨迹与蛋糕是平行的，这样可以把蛋糕切成面积相等（高度可以相等也可以不相等）的两部分，对应到表的切分就是表的列相同但包含不同的行数据。例如，示意图中的水平切分，会把表分为两个表，两个表都包含ID、name、age、sex、nickname、description列，但是一个表包含的是ID从1到999999的行数据，另一个表包含的是ID从1000000到9999999的行数据。

上面这个示例比较简单，只考虑了一次切分的情况，实际架构设计过程中并不局限切分的次数，可以切两次，也可以切很多次，就像切蛋糕一样，可以切很多刀。

单表进行切分后，是否要将切分后的多个表分散在不同的数据库服务器中，可以根据实际的切分效果来确定，并不强制要求单表切分为多表后一定要分散到不同数据库中。原因在于单表切分为多表后，新的表即使在同一个数据库服务器中，也可能带来可观的性能提升，如果性能能够满足业务要求，是可以不拆分到多台数据库服务器的，毕竟我们在上面业务分库的内容看到业务分库也会引入很多复杂性的问题；如果单表拆分为多表后，单台服务器依然无法满足性能要求，那就不得不再次进行业务分库的设计了。

分表能够有效地分散存储压力和带来性能提升，但和分库一样，也会引入各种复杂性。

1.垂直分表

垂直分表适合将表中某些不常用且占了大量空间的列拆分出去。例如，前面示意图中的nickname和description字段，假设我们是一个婚恋网站，用户在筛选其他用户的时候，主要是用age和sex两个字段进行查询，而nickname和description两个字段主要用于展示，一般不会在业务查询中用到。description本身又比较长，因此我们可以将这两个字段独立到另外一张表中，这样在查询age和sex时，就能带来一定的性能提升。

垂直分表引入的复杂性主要体现在表操作的数量要增加。例如，原来只要一次查询就可以获取name、age、sex、nickname、description，现在需要两次查询，一次查询获取name、age、sex，另外一次查询获取nickname、description。

不过相比接下来要讲的水平分表，这个复杂性就是小巫见大巫了。

2.水平分表

水平分表适合表行数特别大的表，有的公司要求单表行数超过5000万就必须进行分表，这个数字可以作为参考，但并不是绝对标准，关键还是要看表的访问性能。对于一些比较复杂

的表，可能超过1000万就要分表了，而对于一些简单的表，即使存储数据超过1亿行，也可以不分表。但不管怎样，当看到表的数据量达到千万级别时，作为架构师就要警觉起来，因为这很可能是架构的性能瓶颈或者隐患。

水平分表相比垂直分表，会引入更多的复杂性，主要表现在下面几个方面：

- 路由

水平分表后，某条数据具体属于哪个切分后的子表，需要增加路由算法进行计算，这个算法会引入一定的复杂性。

常见的路由算法有：

范围路由：选取有序的数据列（例如，整形、时间戳等）作为路由的条件，不同分段分散到不同的数据库表中。以最常见的用户ID为例，路由算法可以按照1000000的范围大小进行分段，1 ~ 999999放到数据库1的表中，1000000 ~ 1999999放到数据库2的表中，以此类推。

范围路由设计的复杂点主要体现在分段大小的选取上，分段太小会导致切分后子表数量过多，增加维护复杂度；分段太大可能会导致单表依然存在性能问题，一般建议分段大小在100万至2000万之间，具体需要根据业务选取合适的分段大小。

范围路由的优点是可以随着数据的增加平滑地扩充新的表。例如，现在的用户是100万，如果增加到1000万，只需要增加新的表就可以了，原有的数据不需要动。

范围路由的一个比较隐含的缺点是分布不均匀，假如按照1000万来进行分表，有可能某个分段实际存储的数据量只有1000条，而另外一个分段实际存储的数据量有900万条。

Hash路由：选取某个列（或者某几个列组合也可以）的值进行Hash运算，然后根据Hash结果分散到不同的数据库表中。同样以用户ID为例，假如我们一开始就规划了10个数据库表，路由算法可以简单地用user_id % 10的值来表示数据所属的数据库表编号，ID为985的用户放到编号为5的子表中，ID为10086的用户放到编号为6的字表中。

Hash路由设计的复杂点主要体现在初始表数量的选取上，表数量太多维护比较麻烦，表数量太少又可能导致单表性能存在问题。而用了Hash路由后，增加字表数量是非常麻烦的，所有数据都要重分布。

Hash路由的优缺点和范围路由基本相反，Hash路由的优点是表分布比较均匀，缺点是扩充新的表很麻烦，所有数据都要重分布。

配置路由：配置路由就是路由表，用一张独立的表来记录路由信息。同样以用户ID为例，我们新增一张user_router表，这个表包含user_id和table_id两列，根据user_id就可以查询对应的table_id。

配置路由设计简单，使用起来非常灵活，尤其是在扩充表的时候，只需要迁移指定的数据，然后修改路由表就可以了。

配置路由的缺点就是必须多查询一次，会影响整体性能；而且路由表本身如果太大（例如，几亿条数据），性能同样可能成为瓶颈，如果我们再次将路由表分库分表，则又面临一个死循环式的路由算法选择问题。

- join操作

水平分表后，数据分散在多个表中，如果需要与其他表进行join查询，需要在业务代码或者数据库中间件中进行多次join查询，然后将结果合并。

- count()操作

水平分表后，虽然物理上数据分散到多个表中，但某些业务逻辑上还是会将这些表当作一个表来处理。例如，获取记录总数用于分页或者展示，水平分表前用一个count()就能完成的操作，在分表后就没那么简单了。常见的处理方式有下面两种：

count()相加：具体做法是在业务代码或者数据库中间件中对每个表进行count()操作，然后将结果相加。这种方式实现简单，缺点就是性能比较低。例如，水平分表后切分为20张表，则要进行20次count(*)操作，如果串行的话，可能需要几秒钟才能得到结果。

记录数表：具体做法是新建一张表，假如表名为“记录数表”，包含table_name、row_count两个字段，每次插入或者删除子表数据成功后，都更新“记录数表”。

这种方式获取表记录数的性能要大大优于count()相加的方式，因为只需要一次简单查询就可以获取数据。缺点是复杂度增加不少，对子表的操作要同步操作“记录数表”，如果有一个业务逻辑遗漏了，数据就会不一致；且针对“记录数表”的操作和针对子表的操作无法放在同一事务中进行处理，异常的情况下会出现操作子表成功了而操作记录数表失败，同样会导致数据不一致。

此外，记录数表的方式也增加了数据库的写压力，因为每次针对子表的insert和delete操作都要update记录数表，所以对于一些不要求记录数实时保持精确的业务，也可以通过后台定时更新记录数表。定时更新实际上就是“count()相加”和“记录数表”的结合，即定时通过count()相加计算表的记录数，然后更新记录数表中的数据。

- order by操作

水平分表后，数据分散到多个子表中，排序操作无法在数据库中完成，只能由业务代码或者数据库中间件分别查询每个子表中的数据，然后汇总进行排序。

实现方法

和数据库读写分离类似，分库分表具体的实现方式也是“程序代码封装”和“中间件封装”，但实现会更复杂。读写分离实现时只要识别SQL操作是读操作还是写操作，通过简单的判断SELECT、UPDATE、INSERT、DELETE几个关键字就可以做到，而分库分表的实现除了要判断操作类型外，还要判断SQL中具体需要操作的表、操作函数（例如count函数）、order by、group by操作等，然后再根据不同的操作进行不同的处理。例如order by操作，需要先从多个库查询到各个库的数据，然后再重新order by才能得到最终的结果。

小结

今天我为你讲了高性能数据库集群的分库分表架构，包括业务分库产生的问题和分表的两种方式及其带来的复杂度，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，你认为什么时候引入分库分表是合适的？是数据库性能不够的时候就开始分库分表么？

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！）

海盗

2018-05-31

分库，分表带来的问题都知道，能不能介绍些好的方便的改进方案？

明日之春

2018-05-31

应该是这些操作依次尝试
1.做硬件优化，例如从机械硬盘改成使用固态硬盘，当然固态硬盘不适合服务器使用，只是举个例子
2.先做数据库服务器的调优操作，例如增加索引，oracle有很多的参数调整；
3.引入缓存技术，例如Redis，减少数据库压力
4.程序与数据库表优化，重构，例如根据业务逻辑对程序逻辑做优化，减少不必要的查询；
5.在这些操作都不能大幅度优化性能的情况下，不能满足将来的发展，再考虑分库分表，也要有预估性

作者回复

2018-06-01

赞，写的很完善

公号-Java大后端

2018-05-31

分库分表，可以理解为是一种空间换时间的思路，同时分流了存储压力与读写压力。

数据库性能不够时，首先应该想到是否可以通过改善硬件条件等垂直扩容手段；其次可引入读写分离、缓存/NoSQL、全文检索等手段；然后，单库单表的访问仍然存在性能瓶颈，可考虑分库分表，并且分库分表可以按照业务进行垂直拆分，接着进行水平拆分。

我的问题是:当线上已经进行了分库分表的系统，需要进一步水平扩容时，有什么好的设计方案？

作者回复

2018-06-01

没有太好的方案，要么一开始的分表方案就是按照id范围来设计的，要么就需要数据迁移

李元霸

2018-06-01

老师，使用tidb方案来代替分库分表是否推荐？

kylexy_0817

2018-05-31

其实在同一个库里做数据拆分，无需分表吧？就例如MySQL，只需要对表做分区就可以了。分表的目的主要是为后面把表分到不同的数据库实例作准备？
还有个疑问，文中提到，用户表的description等字段内容不是经常查到，所以做垂直划分可以提高查询性能，意思是，如果表中有内容较长的字段，查询的时候不查出来（不使用select *），也会有性能问题？
最后，我觉得当单实例的访问量，已达到机器的60%承载能力，就要考虑分库，而具体如何拆分，则要分析访问量主要集中在哪些表。至于分表，主要依据还是单表的数据量和查询性能。

作者回复

2018-06-01

关系数据库是行存储，即使不用那一列，也会从存储读取到内存

haydenliu

2018-05-31

老师，后面能不能专门讲讲如何分布式一致性的实现？比如最简单的一个例子，下单，扣款，如何保证账户余额的实时一致性？

作者回复

2018-06-01

库存和余额是强一致性，业务要求高的情况下不会用分布式并发操作，除非牺牲一定的用户体验，例如商品查询的时候显示有货，但实际下单的时候却显示没货

oddrock

2018-05-31

分库分表的实施前提：
1. 数据库存在性能问题，且用加索引、慢查询优化、缓存和读写分离都无法彻底解决问题的时候
2. 复杂查询对应的单表数据量级一般超过千万以上，或简单查询的单表数据量级一般超过5000万以上
3. 对一致性要求不是特别高，只要求最终一致性。
4. 用hadoop等大数据技术因为业务需求、技术成本、时间成本无法解决该问题

其他：
1. 业务对数据的操作主要集中在某些字段上，比较适合垂直分表
2. 业务对数据的操作在整个表层面较均匀分布，适合水平分表

齐帜

2018-06-06

华哥，能否介绍下分库分表的最佳实践呢？或者分享下成功的实际案例

作者回复

2018-06-07

要结合业务来具体设计，不存在统一的最佳实践。例如，订单表可以按订单号分表，也可以按照买家id或者买家id分表。还可以按照时间分表，都可以，没有哪个是肯定最优的，和业务场景有关

万里晴空

2018-06-01

那单表的话就会出现千万级数据，这样两三年单表很大了，会不会出现性能问题。而且我们有一个功能是用到这张表进行统计，数据量大的话，在抽取数据也有可能出现卡死问题

作者回复

2018-06-01

单表千万级不一定有问题，关键看性能要求，有的历史表几千万数据，但几乎没有访问

鲁伊李

2018-05-31

分库分表的痛点大家都知道，可否介绍下解决方案...

作者回复

2018-06-01

没有很好的，要么中间件要么代码中间层，业务代码处理总是很麻烦的

Snway

2018-05-31

我们是在设计初就考虑进去了，预估单表数据容量，再考虑未来三年的数据增长，不过现在反观这种做法，感觉有点过度设计，如当初一张用户表，分了127张，2个库，然而实际数据容量根本没这么多，顶多千万级别，不仅带来存储资源浪费，也给编码带来不少复杂度！所以我还是觉得得遵循演化原则，业务真正发展起来再考虑分库分表

作者回复

2018-06-01

我们也这样做过，后来受不了了又缩表������

ASCE1885

2018-05-31

应该要对数据库做监控，在下面两种情况出现时，就要给出告警，架构师就要把分库分表提上日程：1）对数据库表做监控，根据具体业务场景，当表行数超过某个阈值时；2）对数据库查询等性能做监控，同样当时间超过某个阈值时。

sea

2018-05-31

请问配置路由具体如何实现呢？

at三月

2018-06-26

想问下文中提到的单台数据库10万用户量级是指在线用户数10万相当的业务数据量级吗？（应该不是单表10万数据吧？）另外单库数据量级有官方文档说明吗？谢谢老师

成功

2018-06-07

作为架构预判，到了数据模型的阀值就要启动分库分表

作者回复

2018-06-07

先优化，通常数据库刚表现出压力的时候，大部分原因不是因为业务真的发展到数据库撑不住了，而是很多慢查询导致的

三木子

2018-05-31

分库分表会不会一直分下去？像腾讯阿里的一些应用？如果存在，是不是也会有瓶颈？

作者回复

2018-06-01

业务一直发展变化的话，是有可能的

zj

2018-05-31

有哪些开源的分库分表中间件呢

作者回复

2018-06-01

可以搜索

Jonah

2018-05-31

其实hash还有一个问题，就是当新增一张表时，怎么处理，比如说我原来是10张表，现在要增加到11张表，可以用一致性hash解决，但也会有数据迁移问题

Tom

2018-05-31

我的理解：预测压测，提前分表分库，预留实现的时间，出现性能时再实现会影响业务。

不吃番茄的西红柿

2018-07-17

咳咳，华仔，我再问一下，如果就算用用户id来进行分表，当后台需要统计整体的比如一个商户的后台订单情况，这时候可能该平台拥有的用户订单数据来自多个表？这时候怎么统一去查询这块数据呢？或者优化查询？

作者回复

2018-07-17

数据冗余，多个不同纬度的订单表

不吃番茄的西红柿

2018-07-16

华仔，如果这时候分了表之后如果用户a的比如订单数据出现在 a和b表中，那么这时候就会跨裤 这时候性能问题怎么处理呢

作者回复

2018-07-17

跨库不一定有性能问题呀，如果有，那就尽量不要跨库，改成按用户id分库，而不是按订单id分库；如果两种方式都必须，可以考虑数据冗余，即设计两个订单表，一个以用户id分库，一个以订单id分库

Marx

2018-07-15

能不能详细讲讲水平分表有哪些场景，应该考虑哪些因素？

Kongk0ng

2018-07-10

如果使用hash进行分表的话，为什么大多方案推荐2的n次方作为表的总数，除了收缩容易还有什么好处吗？谢谢

作者回复

2018-07-11

这个是Hash函数实现的一个技巧，当计算Hash值的时候，普通做法是取余操作，例如h%len，但如果len=2的N次方，通过位操作性能更高，计算方式为h & (len-1)

MarksGui

2018-07-04

当业务分拆到三四个服务里后，同时又需要同时连接这三张表进行查询。这时只能通过调用不同的接口来处理数据吗？

作者回复

2018-07-04

最好通过接口处理数据，否则代码拆分数据不拆分，会导致非常多的问题

张国胜

2018-06-29

数据库优化顺序:
1. 慢查询
2. 根据业务特点，对表设计进行优化
3. 业务逻辑问题
4. 多用短查询
5. 部分表加缓存
这些做完再考虑分库分表吧？

银太@巨益科技

2018-06-27

对于count的情况通常是有条件的，计数作用不大，这种有什么更好的方案吗？

作者回复

2018-06-28

定期更新，但要牺牲用户体验

天平

2018-06-25

请教下，toc的业务分库分表了，公司内部系统tob的业务怎么解决查询多个库中表数据问题？

孙振超

2018-06-18

引入分库分表的主要考量是数据量和系统容量要求，如果tps要求即将超过了单数据库实例所能承担的上限，或者数据库链接数满足不了上游应用服务器的要求，又或者数据库主从同步延迟超过了大多数业务要求的时间，就需要考虑引入分库分表了。在进行分表的拆分后，如果数据库的链接数或者容量还不满足，需要再次拆库，这里的拆库和文章里面的里面提到的拆库稍有不同，文章中的拆库是按照业务进行分拆，这里提到的拆库是将同一个业务的数据表分拆到多个数据库实例，比如将100张用户表从一个数据库实例分拆到10个数据库实例上，这样总容量可以提升近10倍，并且这样分拆后业务的整体可用性也有所提升，因为之前所有的数据都在一个实例上，如果这个实例出故障，业务就不可用了，分拆到10个实例，即使有一个实例出现了故障，还有90%的业务可用，当然成本也提升了10倍了。

作者回复

2018-06-19

你说的情况其实就是分表，只是最开始的时候所有分表都在一个数据库实例上而已

食指可爱多

2018-06-15

想请教老师一个设计问题，以电商平台交易系统为例，订单数据量非常大的时候也可以考虑水平分库分表。
我的思考是针对消费者端订单表按用户ID哈希规则分表，这样所有对用户订单的查询条件全都带上用户ID，达到了数据分片的效果。
但这时商家端需要对订单做管理，就很尴尬了。于是我想到，可以将订单数据做同步到另一个数据源，表结构一致只是按照商家ID进行哈希规则分表，所有商家端查询走此数据源，条件全部带上商家ID，也可以做到数据分片的效果。
接下来问题又来了，系统还有一个平台的视角，这时貌似不好沿着这个思路继续了，恳请老师提点提点。

作者回复

2018-06-15

淘宝的单元化改造就面临你说的问题，最后他们选择了买家纬度拆分，卖家纬度不拆分，详细可以搜网上的公开资料。

Interesting

2018-06-15

所以在分库分表前，一般的业务用优化sql就搞定了。不用分库分表这把牛刀。老师对sql优化有什么指导建议麽？

作者回复

2018-06-15

高性能mysql，把explain的结果能完全看懂就很厉害了

Silence

2018-06-10

想知道业界有没有什么中间件可以方便解决这类问题

小飞哥 超级會員

2018-06-09

横向拆分后，分页查询会遇到查的结果不是相要的，如。我要查带库存的商品数据，先去查商品然后查库存，查完过滤掉无库存的商品，在组合起来，发现只有几条数据。达不到分页效果！这应该怎么处理？

作者回复

2018-06-10

先组合查后分页

王维

2018-06-09

华仔你好，看了这期文章感觉很受益，感受最深的就是架构的演进要随着业务的发展而发展，是业务成就了技术，而不是技术成就了业务。有个问题想问下，分表分库会带来事务性的问题，那么，如果我在业务层采用消息服务框架，例如NserviceBus，可不可以避免这个分表分库的事务性问题呢？

作者回复

2018-06-10

没看明白，按道理消息服务框架不是为了解决事务问题的

食指可爱多

2018-06-08

我的理解分库分表是在数据库容量到了瓶颈（通常此时也会联动出现性能问题）时才需要引入，若数据量不够大，此时出现性能问题应从当前数据库、应用层优化开始。

刘志刚

2018-06-06

我们公司业务增长比较平稳，已经经历了几个过程，
1.先是最基础的主备，Oracle 扛了3年到15年，中间优化了几次硬件和数据库上的配置
2.到16年左右开始扛不住了，数据量在3千万，但是订单表列太多，导致性能开始不理想，这个时候做了一把分区，性能勉强接受，继续扛着
3.到17年之后做到18年开始做去O项目换MySQL了，顺带着分表和分区了，目前按业务，把一些非核心业务分出去其他库了，订单的库还没分，按照历史年表和热表来做的，做滚表实现的，热表数据差不多在1千万以内，现在性能还不错，历史数据用搜索聚合的，查询性能还不错！
4.到后面如果业务再持续增长的话，估计就要拆订单库了

总结一句话，分库分表要在业务需要之前一点，看数据量和业务特性！

作者回复

2018-06-06

这样做没错，如果你们12年就开始分库分表，可能当时的方案也不一定适应后来的业务变化

飞

2018-06-06

如何实现对业务透明的分表，同时也能支持数据库join，group by等操作。

作者回复

2018-06-06

只有中间件和中间层能做到，例如TDDL

天下第一帅

2018-06-05

期望能看到实战，这些理论都大差不差

作者回复

2018-06-05

实战就是要基于理论分析和选择的

sprinty

2018-06-04

老师您好，看了复杂度的分析启发很大，希望可以多讲讲实现方案。比如，分表的情况下做order by，需要综合所有数据做二次order by，第二次的时候数据量不是也很大吗？可能还没有索引，会不会更慢？

作者回复

2018-06-06

如果有limit，第二次会小一些，如果都是全量order by，性能怎么做都高不到哪里去，分库分表还会更慢

卡卢比

2018-06-03

分库分表的原理和出现的问题网上一大堆文章，但是介绍具体解决方案和方案优缺点的却很少，希望大大能详细些。

作者回复

2018-06-04

没有太好的方案，文中提到的基本都是常用的

itperson

2018-06-03

应该有一个具体规则 根据服务器配置不同 分库分表的具体条件也会有一定的差异 首先可以做技术储备 在刚开始不必分局分表 当业务迅速发展时提前预见

Geek_8242cb

2018-06-03

单表数据量很大，并且读写都很多的话，应该做数据清理啊。将流水类的数据清理到专门的读数据库或者大数据平台做查询，保证单表数据量不会太大。

fiseasky

2018-06-03

按照ID范围来分表，比如一千万条数据一张表，为什么数据会不均匀呢？

作者回复

2018-06-04

看ID怎么生成的，递增ID容易被人遍历

忠厚

2018-06-03

好的架构师需要不断的方案积累，老师，能不能谈谈未来5-10年内架构师的前景

作者回复

2018-06-04

话题太大，不好展开，对于业务架构来说，简单来说就是“云”+“人工智能”

万里晴空

2018-06-02

历史拆分出来，和分区一个道理，我想知道为什么拆分历史表会好一点？拆分后关于历史数据和业务后期有一种补录功能，这样就会两张表切换了。很想知道原因，谢谢啦

作者回复

2018-06-04

拆分表后表容量小了，索性和数据也会小，性能会高一些

万里晴空

2018-06-01

针对历史数据，是单表进行分区好呢，还是表再分成历史表好呢（属于水平拆分吧），这两个有什么区别？对性能要求很高

作者回复

2018-06-02

历史表建议拆分到独立的表

三月沙@wecatch

2018-06-01

还是要先简单再复杂，遵循演化原则，根据团队能力和精力，在合适时机再分库分表，早分早复杂，晚分晚痛苦。能不能再介绍一些对分库分表的这些问题的替换方案，比如采用类似tidb的新式数据库

作者回复

2018-06-01

不成熟或者未经大规模验证的一般可以尝试，但别太冒进

万里晴空

2018-06-01

非常感谢每次给我回复。再请教一个问题，业务上全省有800万学生（第一次全部刷到系统的学生表中，其中有一个字段决定学期），每个学期学生操作后进行归档（历史数据产生），到了下一个学期，要求这些学生全部激活（即又批量刷800万，其实这些数据消息队列已经实时
推送）现在有两种方案：1用一张历史学生表和当前学生表，也就是你讲的水平拆分
另外一种方案是用一张表来存放，每个学期都在这里面，这个方案会用到分区。请问一下哪种会好一点，用分区可以用一张表搞定，用水平拆分用两张表，业务上就会调整查历史数据用一张，查当前用另外一张。两者之间都会存在每个学期批量刷800万数据，有什么好的思路可以提供不？非常感谢��

作者回复

2018-06-01

每个学期才刷一次，而且才800万数据，也不是天天刷，完全没有什么问题啊，怎么简单怎么来

侯操宇

2018-06-01

配置路由是否可以通过mc或者redis来存储路由信息实现以提升性能？避免数据库中路由表过大对性能的影响

作者回复

2018-06-01

是可以，但其实数据库的热点表一样会放在内存中，访问同样很快，因此不是很有必要引入mc或者redis来提升性能

何磊

2018-06-01

我当前使用的是mycat来进行分表，他会对数据进行聚合，但是性能可能会受到一些影响，不过对于单表上亿来说，性能还是高处不少。
对于什么时候引入分表，还是看业务，对于不重要的数据也可以定期归档。

作者回复

2018-06-01

先优化，包括优化业务，优化慢查询，优化硬件，只有这些优化手段都不够后才考虑分库分表大杀器

万里晴空

2018-05-31

讲了读写分离和分库分表，主要解决数据量问题，那为啥不考虑分区呢？设计每张表的时候都带上一个区域范围，然后根据这个字段进行分区？请问分区有什么影响么？目前项目中考虑分区来实现数据量大的问题，不知道有什么后果

作者回复

2018-06-01

你说的分区是分表的一种具体实现方式，分表既可以按照主键id范围划分，也可以按照某列的值划分，例如按省份分表

空档滑行

2018-05-31

1，分库分表的需求肯定是和优化需求一起出现的，当数据量大到通过常规的手段无法提高性能的时候需要考虑分库分表
2，业务预估，业务快速发展过程中，通过预估大概可以判断到什么时间点左右会出现分库分表的需求，提前做好准备。

王磊

2018-05-31

在分表和分库之间，是否还可以通过schema(如默认是public)来分散数据，好处是表名(分表不能同名)可以相同，可以通过schema.table的方式出现在一条sql内进行join操作(分库不能join)。

王磊

2018-05-31

我理解-对于列式数据库，则没有必要对表进行垂直切分了。

作者回复

2018-06-01

那是nosql的内容了，后面会涉及

YMF_WX1981

2018-05-31

作为一个前端肯定没实操过，认为在业务类型和种类发展使得当前顶层设计不管从计算上或扩展上均不能实现业务，那就可分库分表了。

Johnny.Z

2018-05-31

横切的表做order by貌似得到的结果不准确

作者回复

2018-06-01

多表数据聚合后再二次order by就可以了

鹅米豆发

2018-05-31

第二个问题，数据库性能不够的时候就要分库分表吗？性能不够，特别是IO引起的性能问题，首先应该考虑缓存如Memcache、Redis，内存数据库如VoltDB、MemSQL、CounchDB，各种读时模式的数据库如ES、Mongo，各种分布式数据库HBase、TiDB。
第一个问题，关系型数据库有什么优势？研发成本最可控，运维体系最完善，业界产品最成熟，一致性、可靠性、安全性有极好的支持。其中，研发成本这块对小公司，运维体系这块对大公司，都有很大的吸引力。

作者回复

2018-06-01

考虑缓存前还需要考虑优化sql，大部分数据库性能不行的首要原因可能就是慢查询

大光头

2018-05-31

一般不应该轻易分库分表，因为引入复杂度更大，首先应该应该从sql语句出发，优化sql写法，或者是否可以降低耗时比较大sql的频次

合民

2018-05-31

一般公司的系统都会有监控系统对系统各指标进行监控，所以我的思路是结合监控系统和业务指标做判断。这样有明确的数据做参考不仅判断更准确，而且能够提前预估，提前做应对决策。首先根据监控指标判断需要重构的业务点，然后分析性能点瓶颈是TPS还是QPS，如果TPS高再看业务点适合于分表还是分库。此时明确了业务点和需要解决性能点，这里完全可以复用老师架构设计的方法论了。

武坤

2018-05-31

先优化单机性能，如果还是不够，再上集群。这个问题是不是前面提到过...

宇天飞

2018-05-31

业务类型，当前体量，同产品规格预期，项目性能、容量要求

Will

2018-05-31

根据业务、性能、资源的实际情况，综合评估是否需要进行分库分表。
读来本篇主要看到了分库分表的复杂度，不过也确实存在这些问题。

每天都在找小黄车

2018-05-31

当sql语句已经最大优化，但业务查询速度超过5秒，我觉得有必要分库分表了，互联网用户的体验对网页反应的速度越来越挑剔。

LangK

2018-05-31

分表这件事是个无休止的体力活，技术条件允许的话还是要在性能达到瓶颈时直接进行分库操作。使用微服务那一套进行架构重组，这是必然的系统演进结果。

路晓明

2018-05-31

分页操作，估计也得使用中间件比较方便吧

张玮(大圣)

2018-05-31

一般每个公司都有一套数据库规范，比如单表3000万时需要考虑分库分表，另外，成熟公司开始就需要预估数据规模。

另外，分不分库表和分多少，除了需要经验判断，还需要依赖系统的一些指标来定，如数据量增长频率、TPS 、QPS 指标。

sunny

2018-05-31

看当前体量。

熊能

2018-05-31

等业务量达到千万个级别，数据库存储成为主要性能瓶颈的时候

贾盼龙

2018-05-31

分库分表进行查询排序的时候，如果结果集很大(十万级)，也都放在内存中进行排序吗？有没有开源的实现方案可以参考的？

ChamPly

2018-05-31

我觉得应该是在预估的时候或者是压测的时候发现某个流程处理时常异常的时候就要考虑了。再问李老师一个问题，现在常用的mysql中间件有哪些推荐的呢？

郭涛

2018-05-31

应该是在达到性能瓶颈前就要预估吧。