14 | 高性能数据库集群：读写分离

2018-05-29 李运华

14 | 高性能数据库集群：读写分离
李运华
00:00 / 09:24

“从0开始学架构”专栏已经更新了13期，从各个方面阐述了架构设计相关的理论和流程，包括架构设计起源、架构设计的目的、常见架构复杂度分析、架构设计原则、架构设计流程等，掌握这些知识是做好架构设计的基础。

在具体的实践过程中，为了更快、更好地设计出优秀的架构，除了掌握这些基础知识外，还需要掌握业界已经成熟的各种架构模式。大部分情况下，我们做架构设计主要都是基于已有的成熟模式，结合业务和团队的具体情况，进行一定的优化或者调整；即使少部分情况我们需要进行较大的创新，前提也是需要对已有的各种架构模式和技术非常熟悉。

接下来，我将逐一介绍最常见的“高性能架构模式”“高可用架构模式”“可扩展架构模式”，这些模式可能你之前大概了解过，但其实每个方案里面都有很多细节，只有深入的理解这些细节才能理解常见的架构模式，进而设计出优秀的架构。

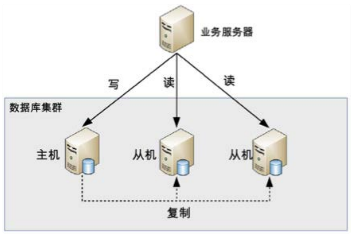
虽然近十年来各种存储技术飞速发展，但关系数据库由于其ACID的特性和功能强大的SQL查询，目前还是各种业务系统中关键和核心的存储系统，很多场景下高性能的设计最核心的部分就是关系数据库的设计。

不管是为了满足业务发展的需要，还是为了提升自己的竞争力，关系数据库厂商（Oracle、DB2、MySQL等）在优化和提升单个数据库服务器的性能方面也做了非常多的技术优化和改进。但业务发展速度和数据增长速度，远远超出数据库厂商的优化速度，尤其是互联网业务兴起之后，海量用户加上海量数据的特点，单个数据库服务器已经难以满足业务需要，必须考虑数据库集群的方式来提升性能。

从今天开始，我会分几期来介绍高性能数据库集群。高性能数据库集群的第一种方式是“读写分离”，其本质是将访问压力分散到集群中的多个节点，但是没有分散存储压力；第二种方式是“分库分表”，既可以分散访问压力，又可以分散存储压力。先来看看“读写分离”，下一期我再介绍“分库分表”。

读写分离原理

读写分离的基本原理是将数据库读写操作分散到不同的节点上，下面是其基本架构图。



读写分离的基本实现是：

- 数据库服务器搭建主从集群，一主一从、一主多从都可以。
- 数据库主机负责读写操作，从机只负责读操作。
- 数据库主机通过复制将数据同步到从机，每台数据库服务器都存储了所有的业务数据。
- 业务服务器将写操作发给数据库主机，将读操作发给数据库从机。

需要注意的是，这里用的是“主从集群”，而不是“主备集群”。“从机”的“从”可以理解为“仆从”，仆从是要帮主人干活的，“从机”是需要提供读数据的功能的；而“备机”一般被认为仅仅提供备份功能，不提供访问功能。所以使用“主从”还是“主备”，是要看场景的，这两个词并不是完全等同的。

读写分离的实现逻辑并不复杂，但有两个细节点将引入设计复杂度：主从复制延迟和分配机制。

复制延迟

以MySQL为例，主从复制延迟可能达到1秒，如果有大量数据同步，延迟1分钟也是有可能的。主从复制延迟会带来一个问题：如果业务服务器将数据写入到数据库主服务器后立刻（1秒内）进行读取，此时读操作访问的是从机，主机还没有将数据复制过来，到从机读取数据是读不到最新数据的，业务上就可能出现问题。例如，用户刚注册完后立刻登录，业务服务器会提示他"你还没有注册"，而用户明明刚才已经注册成功了。

解决主从复制延迟有几种常见的方法：

1.写操作后的读操作指定发给数据库主服务器

例如，注册账号完成后，登录时读取账号的读操作也发给数据库主服务器。这种方式和业务强绑定，对业务的侵入和影响较大，如果哪个新来的程序员不知道这样写代码，就会导致一个bug。

2.读从机失败后再读一次主机

这就是通常所说的"二次读取"，二次读取和业务无绑定，只需要对底层数据库访问的API进行封装即可，实现代价较小，不足之处在于如果有很多二次读取，将大大增加主机的读操作压力。例如，黑客暴力破解账号，会导致大量的二次读取操作，主机可能顶不住读操作的压力从而崩溃。

3.关键业务读写操作全部指向主机，非关键业务采用读写分离

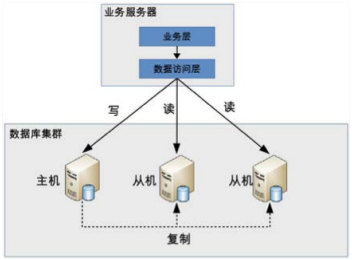
例如，对于一个用户管理系统来说，注册+登录的业务读写操作全部访问主机，用户的介绍、爱好、等级等业务，可以采用读写分离，因为即使用户改了自己的自我介绍，在查询时却看到了自我介绍还是旧的，业务影响与不能登录相比就小很多，还可以忍受。

分配机制

将读写操作区分开来，然后访问不同的数据库服务器，一般有两种方式：程序代码封装和中间件封装。

1.程序代码封装

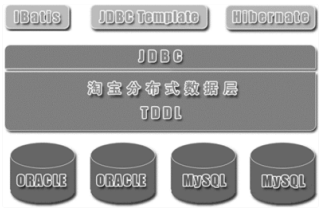
程序代码封装指在代码中抽象一个数据访问层（所以有的文章也称这种方式为"中间层封装"），实现读写操作分离和数据库服务器连接的管理。例如，基于Hibernate进行简单封装，就可以实现读写分离，基本架构是：



程序代码封装的方式具备几个特点：

- 实现简单，而且可以根据业务做较多定制化的功能。
- 每个编程语言都需要自己实现一次，无法通用，如果一个业务包含多个编程语言写的多个子系统，则重复开发的工作量比较大。
- 故障情况下，如果主从发生切换，则可能需要所有系统都修改配置并重启。

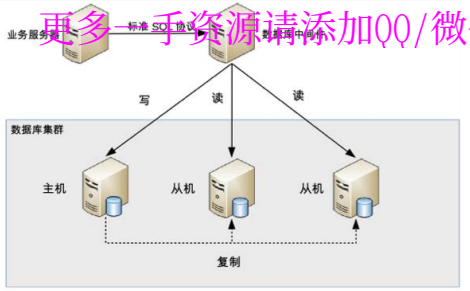
目前开源的实现方案中，淘宝的TDDL（Taobao Distributed Data Layer，外号:头都大了）是比较有名的。它是一个通用数据访问层，所有功能封装在jar包中提供给业务代码调用。其基本原理是一个基于集中式配置的 jdbc datasource实现，具有主备、读写分离、动态数据库配置等功能，基本架构是：



（http://1.im.guokr.com/0Y5YjfjQ8eGOzeskpen2mINIYA_b7DBLbGT0YHyUiLFZAgAAgwEAAFBO.png）

2.中间件封装

中间件封装指的是独立一套系统出来，实现读写操作分离和数据库服务器连接的管理。中间件对业务服务器提供SQL兼容的协议，业务服务器无须自己进行读写分离。对于业务服务器来说，访问中间件和访问数据库没有区别，事实上在业务服务器看来，中间件就是一个数据库服务器。其基本架构是：

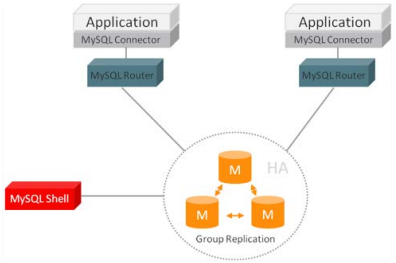


数据库中间件的方式具备的特点是：

- 能够支持多种编程语言，因为数据库中间件对业务服务器提供的是标准SQL接口。
- 数据库中间件要支持完整的SQL语法和数据库服务器的协议（例如，MySQL客户端和服务器的连接协议），实现比较复杂，细节特别多，很容易出现bug，需要较长的时间才能稳定。
- 数据库中间件自己不执行真正的读写操作，但所有的数据库操作请求都要经过中间件，中间件的性能要求也很高。
- 数据库主从切换对业务服务器无感知，数据库中间件可以探测数据库服务器的主从状态。例如，向某个测试表写入一条数据，成功的就是主机，失败的就是从机。

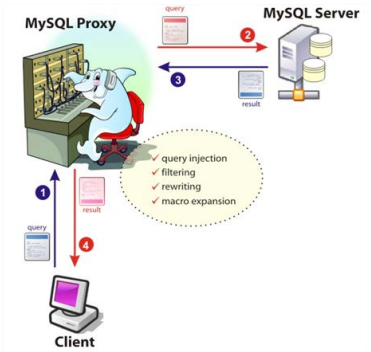
由于数据库中间件的复杂度要比程序代码封装高出一个数量级，一般情况下建议采用程序语言封装的方式，或者使用成熟的开源数据库中间件。如果是大公司，可以投入人力去实现数据库中间件，因为这个系统一旦做好，接入的业务系统越多，节省的程序开发投入就越多，价值也越大。

目前的开源数据库中间件方案中，MySQL官方先是提供了MySQL Proxy，但MySQL Proxy一直没有正式GA，现在MySQL官方推荐MySQL Router。MySQL Router的主要功能有读写分离、故障自动切换、负载均衡、连接池等，其基本架构如下：



（https://dev.mysql.com/doc/mysql-router/2.1/en/images/mysql-router-positioning.png）

奇虎360公司也开源了自己的数据库中间件Atlas，Atlas是基于MySQL Proxy实现的，基本架构如下：



（https://camo.githubusercontent.com/42c01a1245183948ba8c61e5572d3aa9c3e8a08e/687474703a2f2f7777332e73696e61696d672e636e2f6c617267652f366535373037356135356a7731656271353169336668716a32306a69306a6a7767392e6a7067）

以下是官方介绍，更多内容你可以参考这里。

> Atlas是一个位于应用程序与MySQL之间中间件。在后端DB看来，Atlas相当于连接它的客户端，在前端应用看来，Atlas相当于一个DB。Atlas作为服务端与应用程序通信，它实现了MySQL的客户端和服务端协议，同时作为客户端与MySQL通信。它对应用程序屏蔽了DB的细节，同时为了降低MySQL负担，它还维护了连接池。

小结

今天我为你讲了读写分离方式的原理，以及两个设计复杂度：复制延迟和分配机制，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，数据库读写分离一般应用于什么场景？能支撑多大的业务规模？

欢迎你把答案写到留言区，和我、和其他小伙伴一起讨论。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。

2018-05-30

老师，您好
我个人的想法是可以加入缓存，例如注册后登录这种业务，可以在注册后加入数据库，并加入缓存，登录的时候先查缓存再查库表。
例如存入redis中并设置十分钟的过期时间。登录的时候先查redis，再查库表，如果redis中没有，说明就是过期的数据，这时候查从机就肯定存在了，希望能得到老师的点评，谢谢。

作者回复

2018-05-30

赞同，并不是说一有性能问题就上读写分离，而是应该先优化，例如优化慢查询，调整不合理的业务逻辑，引入缓存等，只有确定系统没有优化空间后，才考虑读写分离或者集群

narry

2018-05-29

个人感觉，读写分离适合读压力比写压力大很多的业务类型，最终的瓶颈应该是出现在承担写操作的主机上，最大规模和这台主机的iops等能力相关

wuxu

2018-05-30

读写分离的前提是并发量大，单机已经不能处理该数量的并发请求了，想要解决问题就得做作拆分，于是有了读写分离，主库负责写，从库负责读，降低了同台机器并发请求，当读越来越多时，可扩充从库，写越来越多时，只好拆分业务或分库分表，如：注册功能，单独出来做一个注册的微服务，但还是会到达一个瓶颈，没做过，不知道能支持多少的并发？

作者回复

2018-05-30

要具体测试，不同业务复杂度不同

彡工鸟

2018-05-30

是否还应该加上一个，当单机写顶不住压力后，就可以做数据库拆分了，例如业务纵向拆分，连同数据库一起，就变成分布式服务，微服务了:)

作者回复

2018-05-30

说法没错，但具体实施的时候要注意，不要一有压力就上读者分离，因为很多时候其实是sql语句或者业务逻辑有问题，因此先优化，只有优化后也无法满足要求的时候才考虑读者分离或者集群

侯海佳

2018-05-29

我认为读写分离适合类似微博这种业务：读多写少

Geek_8242cb

2018-06-03

我们做网银系统，用redis存了一些不太重要的数据，比如数据字典信息，作为缓存。但是不太敢把用户权限，交易数据等重要信息存在缓存里，因为redis并不保证事务，我们担心一旦缓存服务器宕机或者失败会影响银行业务。所以缓存的作用也不是很大，还是把大部分读数据的压力放到了数据库上，您说我们这种担心有必要吗？如果单库后续扛不住压力，是否读写分离比加缓存更好一些？

作者回复

2018-06-04

交易型业务缓存应用不多，缓存一般总在查询类业务上，你们的担心有一定必要

@漆~心endless

2018-05-31

并非所有系统都需要进行读写分离，正如之前讲得架构三原则，其中根据“合适原则”的规定，先确认系统的业务量是否出现了数据库的性能问题。如果是，首先通过优化MySQL语句等，如果还是达不到要求的性能指标，则需进行读写分离。毕竟读写分离会引入一系列不可预知的问题，如数据不同步。

合民

2018-05-31

读写分离，主从架构，顾名思义，写不变，主要解决高性能读的问题，所以适用场景自然是读多写少的情况。比如类似于博客、中小型朋友圈，这种一般写数据库后基本不变，但是很多人会去访问，频繁的读。我认为如果缓存能支撑的话就没必要上读写分离，相对来说缓存更简单。

刘岚乔月

2018-05-29

请问 对于主从出现的数据同步延时问题 在实际生产落地 真的只有把重要的查询指向主吗 还有其他真正的落地方案吗

作者回复

2018-05-30

当然是真的呀，难道我还会骗你不成？��
如果不想用这种方式，用缓存是可以规避这个问题的，但其实这时候的方案就不是读写分离了

刘志刚

2018-05-25

读写分离比较适用于类似消息记录，对于写和读业务的强实时性要求不到苛刻的地步的情况，而且做的时候这种跟业务量还是有比较大关系的，比如，业务量的订单量每年都不超过1千万，整天去做分库分表倒不如好好优化下sql语句，如果订单量每天都超过好几千万，那这个决策难很蛋了！

作者回复

2018-05-30

赞，先优化

姜洋昌

2018-05-29

读写分离适用于单服务器无法满足所有请求的场景，从请求类型的角度对服务器进行拆分，但这样在要求硬件资源能够支撑的同时，对代码实现也有更高的要求。

作者回复

2018-05-30

天下没有免费的午餐��

云学

2018-05-29

相比于前面的几篇高大上文章，这篇更接地气

haydenliu

2018-05-29

老师，个人感觉，不是每个场景都需要读写分离，第一，对于那些及时性要求高的业务，是不合适的，第二，读多写少的业务，也没必要，反而增加了复杂度。不知道这样理解对不服？

作者回复

2018-05-29

1. 正确
2. 正好反过来了，读多写少就适合读写分离

null

2018-05-29

re: 写操作后的读操作指定发给数据库主服务器

后端无法知道本次请求是否为写操作之后的读，因此会依赖前端传递一个参数，如 target_db=master / slave，来决定目标数据库。
所以这种方式，需要在前后端代码实现相关逻辑，代码耦合较大。

这种解决方案是否只提供了一种思路，实际开发时很少使用这方案，不知道理解是否正确，谢谢！

作者回复

2018-05-29

是的，对代码逻辑有要求，

richey

2018-06-26

TDDL已经长期不更新了，老师怎么不提一下Mycat

马广乐

2018-05-29

加个缓存能解决写完立即读的场景吗，老师。

作者回复

2018-05-30

可以的，但那是另外一个方案了，很多场景就算用了缓存也要读写分离

侯佳林

2018-05-29

主服务器充当业务的写服务器、从服务器的读服务器，如果从服务器较多，单台写服务器的压力会很大

Tom

2018-05-29

读写分离一方面分离了读和写，另一方面读可通过多个从机再进一步分担读。
1.一主
一开始，读和写都在主机这条船上，随着读写越来越多，读和写相互影响，主机压力越来越大船快沉了。
2.一主一从
于是，读和写商量说分家吧，否则要抱团死了，写是数据源头留在主机，你读就自己找个从机新船吧；
便分成了一主一从，由主向从单向同步数据，有主推和从拉两种同步方式；读和写各自轻松了。
3.一主多从
随着业务开展，读写再次增长，特别是一般系统都是读的增长一般比写快很多，读一个从机也快沉船了，就多加了几个从主，变成了一主多从，数据由主机同步到所有从机，可以很方便地水平扩展。
4.多组
这块已经不算是读写分离了，要分离写了。
当写增长到一定程度，单个主机已经hold不住写了，要分库了。
按业务垂直拆分或水平拆分，拆分后的多库每库作为新的一组，每组是一个主从集群。

还有一种可能的过程是一主机直接到分库成多主机，某个主机读太多时进行读写分离。

能支撑多大规模就不了解了。
查了下mysql官网benchmarks(https://www.mysql.com/why-mysql/benchmarks/)，单机32users并发只读近40w+ qps，读写10w+ qps。
如果按这个数据，以32users算：
一主一从能支撑写10w+qps，读40w+；
一主三从能支撑写10w+qps，读120w+；

没压测过db，是不是太高了？
请指正，谢谢！

作者回复

2018-05-30

这个数据有误导性，要看具体的SQL语句和表结构，实际应用中一般不可能这么高

大光头

2018-05-29

大部分业务场景都适合，因为基本上都是读多写少，又要应付大规模的访问。

作者回复

2018-05-29

事实上大部分业务场景可能并不需要读写分离��

LEON

2018-05-29

您好，请问如果没有数据库中间件，通过客户端程序的方式直接与数据库交互。客户端程序是不是只能指定一个写一个读。不能制定多个读？另外业务到达多大基线的时候需要引入数据库中间件？谢谢老师回复。

作者回复

2018-05-29

1. 客户端程序写代码写的，可以为所欲为
2. 中间件在不影响性能上有优势，而且可以移植到各种数据库，大公司才有实力做

王磊

2018-05-29

华仔，这样理解对吗？
读写分离的原因是同一个表的读写操作往往是阻塞的(事务隔离级别中的可重复读和串行)，因此如果数据库有较多的更新和插入操作，并且影响了读操作，这时应该考虑读写分离。
但是数据库复制时，是否一样会对从机的读操作阻塞？

作者回复

2018-05-29

现在的数据库，写阻塞读是比较少了，你了解一下MVCC之类的机制，主要还是单台服务器撑不住性能要求

鸡蛋🍜 达芬奇

2018-07-17

使用mycat这种不是也很好吗？

作者回复

2018-07-18

可以的

peison

2018-07-09

老师您好，想请教要怎么样做数据库压测？还有您说不同的数据库表设计性能测试结果不同，想问问怎样的设计性能高？怎么样的设计性能低？这方面有哪些资料可以参考吗？请老师指教

作者回复

2018-07-11

高性能mysql，可以看看

peison

2018-07-09

想请教老师，怎么做数据库的压测呢？还有您说不同的数据库表设计性能不同，那怎么样的设计性能高，怎么样的设计性能低？这方面有那些技术资料可以参考吗？请老师指教

作者回复

2018-07-11

可以看看《高性能mysql》

Cola

2018-07-05

读写分离一般用于读操作频繁，读压力比写压力大一个量级的情况下。另外，读写压力较大也可以用各种缓存手段解决问题。

reed

2018-06-30

请教一下，数据库搭建主从的时候，是应该从库配置好，还是主库配置好？我们现在从库都是8核16G，而主库是4核8G，说是为了数据同步更快

作者回复

2018-07-02

同步速度主要受网络延迟影响，和硬件关系不大，而且线上配置最好一样，否则出问题不好排查

张国胜

2018-06-29

其实有时候也大概知道一些概念，遇到性能问题可以做什么，可具体情况下，应该做什么，是新手不太能确定的。比如不知道怎样才是合理的数据库优化顺序。

^_^

2018-06-28

请问老师，写库同步数据到从库具体是怎么实现的

Seven4X

2018-06-23

读写分离，无它，一言以蔽之，一台服务器撑不住了

Seven4X

2018-06-23

读写分离，无它，一言以蔽之，一台服务器撑不住了

孙晓明

2018-06-20

想问的其他同学都问了🍜

放宕的压抑

2018-06-16

现在的数据库，写阻塞读是比较少了，你了解一下MVCC之类的机制，主要还是单台服务器撑不住性能要求

老师您好，我有两个问题。
1.照您的说法，那么划分是否应当是"读写-读写"比"读-写"更加合适呢？
2.个人感觉读写分离并不能减轻写对读的阻塞(虽然老师您说，现在读写阻塞很少了，但是在读写阻塞还比较多的年代而言呢？)，因为从库还是要不停地写，但因为是复制，所以认为一定是可以最终一致的，可以"为所欲为"，避开读的高峰期实现写？但是是否也有这样的场景存在7×24小时读请求都非常高，这种情况怎么办？继续增加机器？

不知道我理解得对不对，望老师指正。
(吐槽一下，第一次提交留言，因为我朋友圈里没几个人是做技术的，所以没想着分享，然后我就取消了。然后我就发了第二遍了🍜)

作者回复

2018-06-19

通常情况下从库的复制写比主库的业务写要简单一些，性能要高

孙振超

2018-06-15

虽然作者已经收到了众多的赞许，但好话不嫌多，在这里还是要向作者表示感谢，在每一篇的文章中都有收获。

对于本篇内容中何时使用读写分离，主要还是看系统所需要承担的容量，对于安装在配置好的物理机上mysql数据库单机qps可以到达数十万，因而对于qps峰值万级以下的是不用考虑读写分离的，对于十万级别的要求高可用的业务可以考虑用读写分离，只是在实际工作中通常会选择加一层缓存，毕竟缓存服务器的成本要比db服务器成本会底不少。

作者回复

2018-06-15

单机mysql 10万QPS可能有误导，我们只是在测试mysql存储和查询k-v数据的时候达到这个量级，正常的业务表结构和查询语句远比这个复杂，远远达不到10万，一般就几千。

Geek_59a17f

2018-06-11

高性能数据库集群的第一种方式是“读写分离”，其本质是将访问压力分散到集群中的多个节点，但是没有分散存储压力；第二种方式是“分库分表”，既可以分散访问压力，又可以分散存储压力。先来看看读写分离，下一期我们介绍分库分表。

gesanri

2018-06-10

我们系统也用到了读写分离，不过我们读写的分配就是访问的数据源不同，读是一个数据源，写是另一个数据源，不知道和文章中说的分配原则是什么区别，就是文章中说的第一种hibernate实现的吗？

作者回复

2018-06-10

是的，程序代码实现的

紫clz

2018-06-07

请问下，数据库读写分离对比主库+缓存的优势在哪里？
因为考虑到缓存最大的问题就是数据时效性，但其实这个问题从库也有，而且从库有多久的延迟完全无法把握。更不说缓存的响应时间比从库快很多。
似乎选主从最大的优势就是：简单，不需要怎么开发。业务有一定的读压力时使用。
希望解惑

作者回复

2018-06-07

如果时效性不是关键，一般都先上缓存，缓存是缓存sql查询的结果，数据库备机只是备份了表数据，实际的sql查询执行时间少不了

王维

2018-06-06

想请问下华仔，我们系统用的数据库是ms sqlserver，如果要使用数据库中间件，那么有什么好的可以推荐的呢？貌似ms sqlserver的数据库中间件没有my sql的多。

作者回复

2018-06-07

目测没有几个成熟的面向sqlserver的开源中间件，估计要买或者自己写中间层

LONGER

2018-06-05

目前还在用单机一直在扛着，目前数据量在百万万，在不停的优化，建立冗余等方式，还在保持着一个较快的查询速度，因为业务查询的关系，多表之间的关联，聚合，很难避免，一直想引用缓存，但是查询的条件太多，很动态，就不知道如何设计缓存，类似于京东筛选物品，多品类，多维度筛选，不知道大牛有何高见

作者回复

2018-06-06

按照2-8原则，选出占访问量80%的前20%的请求条件缓存，因为大部分人的查询不会每次都非常多条件，以手机为例，查询苹果加华为的可能占很大一部分

米斯特.杜

2018-06-05

之前在一家小快递公司，日单量50万左右。最早是Oracle Rac，用了几年后扛不住了，上了读写分离。上线后效果还蛮好的，支撑一百万单量问题不大。用的是直接在代码中指向不同数据源的方式，偶尔会出现数据源指错的情况。也出现过一个事务指向了两个库。当然最坑的还是同步延迟，最长有过延迟十几个小时的情况。复杂的加购解决了原有问题，又带来了新问题。

作者回复

2018-06-06

十几个小时还是太夸张了，应该做好监控及时处理

fenghao

2018-06-05

我想问无论主从、主备都需要时间，秒级，如果主挂了，那么这段时间的数据就丢失了。有好的办法吗？要求备写完返回又耗费性能

作者回复

2018-06-06

没有好的办法��

W_T

2018-06-04

从数据库读写的角度分类，一共有四类：写多读多，写少读少，写多读少，写少读多。
写少读少的情况，不需要分离。
写多的情况，单个库写入会造成单点压力，分库写入，那其实就是分库分表了。
所以我认为，读写分离的设计适用于写少读多的情况。
至于业务量，读可以不断水平扩展，主要还是受写的限制。

不吃番茄的西红柿

2018-06-03

对于读写分离其实还少了一些吧，如果用户量特别大了，可能单存的读写分离已经不足以支持了.....也可能比如用hash取余设定它能访问的数据库，把数据表拆开吧

作者回复

2018-06-04

下一篇就讲了

fiseasky

2018-06-03

公司现在的系统时采用读写分离的，是中间层程序封装的api，第一套分两类:1.读主库,2.读从库.然后客户端程序通过传递SQL或存储过程和参数的值调用。
第二套只提供一个api,通过传递一个布尔值来判断是主主库还是从库，这套是供自动调度工具来调用。这两套api都有一个共同点，就程序猿必须手动指定是走主库还是从库。现在出现的问题是大量的SQL应该走从库，结果很多菜鸟都走了主库，导致现在的主库压力很大。听了你的课程后觉得走主库还是从库不应该由程序猿自己指定，而是由中间层来判断。具体如何做呢，请老师指点一下。客户端有时传递一些复杂的SQL,比如，先做更新然后再查。

作者回复

2018-06-04

默认读走从库，写走主库，特殊情况才由程序员制定，可以代码指定，可以配置指定，这样就不会出现大量sql都走主库了

如风

2018-05-31

写是对于主库，读是从库；读写分离是随着业务的发展，而演变，现有的数据库支撑不了了，特别大量的数据及查询会拖死数据库，这个时候就得考虑读写分离和多个分组主从库，服务也做拆分，从而演变成微服务，保证核心业务不受影响，其他非核心业务挂了也不受影响，这就是服务降级，防止血崩

今夕是何年

2018-05-30

读写分离应用于读多写少的场景

空档滑行

2018-05-30

读操作远大于写操作并且并发量很大的情况需要读写分离。因为分离后可以针对读库做单独的优化。
关于读写分离后的复制延迟问题，我感觉无论是二次读取还是区分关键业务，都会出现写库压力大的问题。还是要结合第一点的业务代码修改。
还有一个问题最近一直在考虑，读写分离后的dal层设计是在查询接口实现时读写库各实现一套还是通过传参来区分更好呢？请教一下老师

何国平

2018-05-30

还有事务是确定次序，而现在很多开发把这个都放到事务一块儿去了导致分离很难

YMF_WX1981

2018-05-30

读写分离如果是为了性能，大可通过硬件扩容解决呀。我是想，既然读写分离了，是否意味数据和对数据操作也分离了，更容易记录和统计，也更安全？适合敏感的金融类业务。

作者回复

2018-05-30

读写分离没有分离数据，只是分离了读写操作，和金融业务关系不大

hello2018

2018-05-30

做了集群，是否可以不需要主备主从读写分离?

作者回复

2018-05-30

集群更强大更复杂

coder_java

2018-05-29

华仔好，看了大家的回答基本上回答的都不错。
关于集群的问题能否做个总结，redis里有集群，es有集群，kafka等等都有集群，他们有什么共同之处么？或者说区别都在于哪？都是主读写，从读么？

作者回复

2018-05-30

这个留作课后作业给你，自己的总结印象最深刻

沧海一粟

2018-05-29

读写分离一般适用于读多写少的业务场景，支持规模应该在千万级别。问一下老师，可否使用缓存来替代数据库读写分离，做读写分离延迟是比较大的问题

作者回复

2018-05-30

可以的，如果缓存能顶住，其实一般不用做读写分离，例如论坛的帖子浏览场景

任锋

2018-05-29

读写分离 主要解决了 读的压力，所以能满足多大的规模 瓶颈在于主库的瓶颈吧，比如连接数超了，系统也达不到高可用？不知道这样理解对不对.....

作者回复

2018-05-30

不一定是读的压力，我们假设机器的能力是100分，写占了70分，读占了30分，虽然写是性能主要消耗方，但读写分离一样可以减轻数据库压力

cqc

2018-05-29

适用场景的话大家都提到了：应该是读多写少的情况。另外对于数据复制/同步的三种方案，我归纳了一下，第1和3是从业务角度去解决，第2是从技术角度去解决，这让我想起一句话：有些技术不好解决的问题也许从业务方面处理会更有效一些，不知道理解得对不对？不过从这三个方案来看，方案三确实是目前最优的。

作者回复

2018-05-29

非常正确，有时候业务上调整1分，技术复杂度能减少10分，但方案3不能说是最优的，对业务有侵入

nola den

2018-05-29

我想在独立一台主机无法承受访问压力时，才需要使用读写分离。目的是分配读操作的压力到从机上。但主机还是承受所有写操作压力，当主机无法应对大量写操作的时候，整个存储系统也就崩掉了。因为从机可以扩展，主机只能一台。

Snway

2018-05-29

我们都是读写分离，单表数据量大就分库分表，有专门的数据库中间件，实现比较方便。当然，对于写后立即读的操作都是指定读主库的！在cqrs架构中还专门区分了读写模型。

作者回复

2018-05-29

有中间件真的可以为所欲为��

yushing

2018-05-29

请问为什么黑客破解账号，会导致大量的二次读取操作呢？大部分帐号在从机中也有吧，怎么还会二次读取呢？

作者回复

2018-05-29

黑客暴力破解的时候是尝试性的，很多账号或者信息不存在