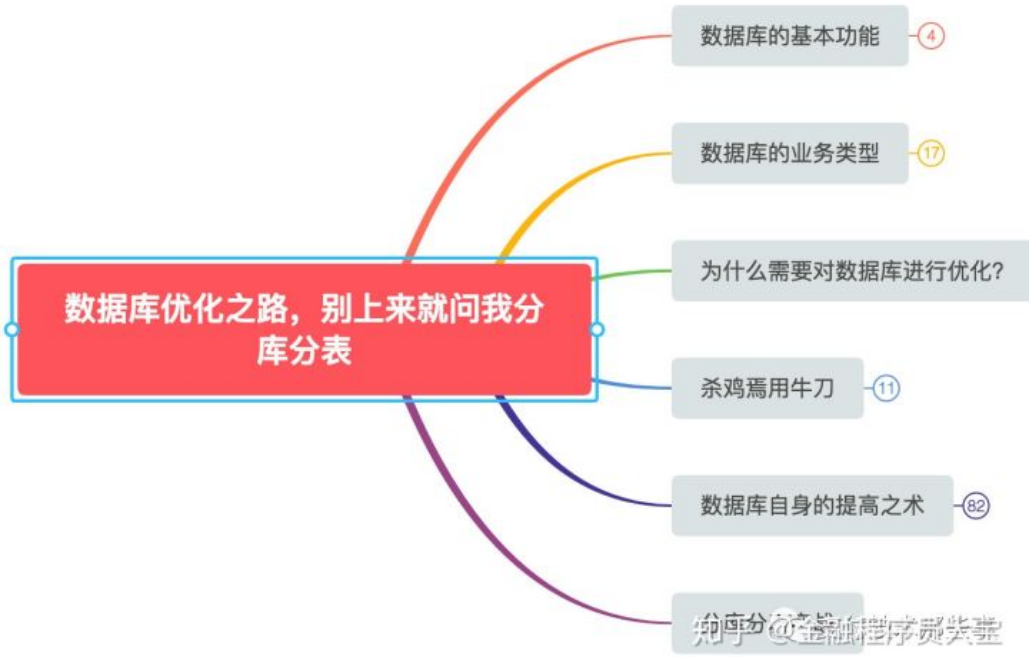
数据库优化之路，别上来就问我分库分表

支付技术那些事
后台开发工程师

24 人赞同了该文章



1. **数据库的基本功能**

不管是关系型还是非关系型数据库，不要被其众多的知识点淹没，回归初心，数据库**基本服务就是提供数据的读和写，剩下的全部围绕这两点展开和优化。**

为什么需要对数据库进行优化?

首先本文讲的是关系型数据库的优化，那不可避免的一个就是，为什么不使用其他的数据库替代比如NoSQL？或者说为什么还保留关系型数据库?

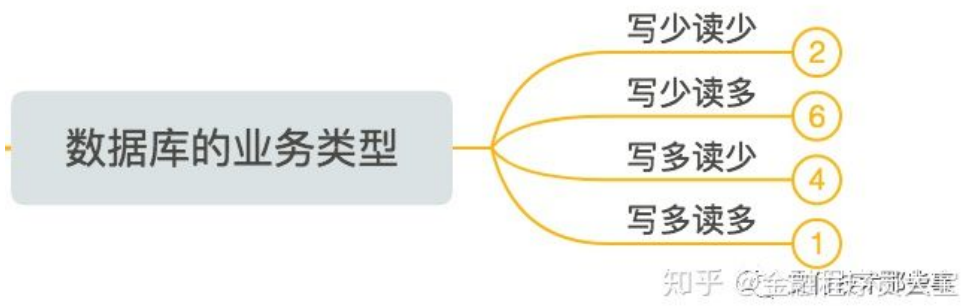
无外乎有一下几点：

1. 技术及时更迭很快，关系型数据库暂时还离不开

2. 即使关系型数据库再不断提升自己的能力，也**架不住互联网海量用户的增长速度。**

所以，优化就是为了解决问题，那么问题是什么，问题就是矛盾：**互联网日益增长的用户对数据库的依赖，和关系型数据库存储和读写的性能提升的矛盾。**

***2.* 数据库业务类型和解决方案总结**



写少读少

单数据库就够了，一般不需要进行特别的优化；

业务梳理清楚，SQL 写的正确一些即可；

写少读多

首先进行垂直拓展，也就是在硬件上进行改进，往往一个SSD就能搞定很多问题；

优化SQL，该建索引建索引；

业务梳理，减少没必要的查询或者不合理的查询；

使用缓存：进程内的缓存、进程外缓存；

进行读写分离设计；

写多读少

往往对写入的能力即存储能力要求高；

进行分库；

进行分表；

分库又分表；

写多读多

首先恭喜你，遇到这种业务，是你技术生涯的一次挑战和经验；

能用上的方式 ALL in 考虑；

~~3. 系统高用平方~~

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

能用**简单的方式解决80%的问题，也比复杂的方案解决95%的方式优先；**

做硬件优化，比如从机械硬盘改成使用固态硬盘，能抗多少就抗多少；

做数据库服务器的调优操作：**增加索引，数据库参数调整；**

程序与数据库表优化；

SQL 合理吗？SQL尽力优化了吗：

业务重构，例如根据业务逻辑对程序逻辑做优化，减少不必要的查询；

缓存用上了吗？引入缓存技术，例如Redis，减少数据库压力

总之，**能先进行简单优化则没必要引入负责的解决方案，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；**

能看的出来，上面的几种方式都是比较**适合量级单库能控制住以及业务属于写少读多的场景。**

鸡刀都用上了，性能还是提升不上去，就需要放大招了。当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，当然，要有预估性；

4. 数据库自身的提高之术

4.1 读写分离

4.1.1 应用场景

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。

有以下几个特点：

1.单机数据库不能支持业务的读写规模

2.前提是写的规模不能高于单机数据库支持的规模，写不是瓶颈；

3.适用单机并发无法支撑，并且读的请求更多的情形；

读规模可以横向扩展，但也不是无限的，因为数据库复制工具也需要从数据复制数据到从库，复制工具复制频率根据业务决定。

在单机数据库情况下，表上加索引一般对查询有优化作用却影响写入速度，读写分离后可以单独对读库进行优化，写库上减少索引，对读写的能力都有提升，且读的提升更多一些。

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。

4.1.2 基本原理

就做了一件事：**将数据库读写操作分散到不同的节点上**

简单的实现，**比如将面向用户的业务读写都用主，面向客户和运营的业务可以读写分离**

4.2.3 基本实现

数据库服务器搭建主从集群，一主一从、一主多从都可以。

主机负责读写操作，从机只负责读操作。

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。

4.1.4 如何进行读写分离

读写请求如何分配

将读写操作区分开来，然后访问不同的数据库服务器

谁去做，怎么做?

客户端模式

属于读写请求操作是业务代码耦合的一部分

在代码中抽象一个数据访问层，实现读写操作分离和数据库服务器连接的管理

典型开源实现方案

淘宝的 TDDL，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

服务端模式

原理：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

该套系统对业务服务器提供 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；

典型开源实现方案

MySQL 官方先是提供了 MySQL Proxy

360 公司也开源了自己的数据库中间件 Atlas

读写分离带来的致命问题

主从复制延迟：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。如果业务服务器将数据写入到数据库主服务器后立刻进行读取，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（写入）后立刻登录（查询），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。

解决方案

1. 写操作后的读操作指定发给数据库主服务器；

2. “二次读取”：读从机失败后再读一次主机；

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；

4.3 分库分表

分库分表：**将访问和存储分散到多个节点上**

适用场景

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；

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；

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。

分库，分库分表最简单的一种

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

分库使用时机：

1、业务不复杂，但整体数据量已影响了数据库的性能。

2、业务复杂，需要分模块由不同开发团队负责开发，这个时候使用分库可以减少团队间交流。

分表，分库分表的最复杂的一种

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。

表数据拆分有两种方式

分库分表引入带来的复杂性问题

垂直分表

垂直分表带来的数据冗余，以及查询次数的增加

例如用户信息表，有一下几个字段，姓名、年龄、性别、昵称、描述，现在将变化较大的的数据（昵称和描述），单独拆分出一个表中。

这样原来只要一次查询就可以获取 name、age、sex、nickname、des，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、desc。

水平分表

表多大才需要考虑分表，确定分表的阈值：

1.水平分表适合表行数特别大的表

2.有的公司要求单表行数超过 5000 万就必须进行分表

3.对于一些比较复杂的表，可能超过 1000 万就要分表了

4.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

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。

如何进行水平分表（如何进行数据路由）

以下三种：

1. **按照范围**

2. **Hash 路由**

3. **配置路由**

分库分表不能支持的操作

分库分表的实践由于涉及的东西较多，下篇继续分享！

发布于 2019-04-09

数据库　分布式数据库　数据库性能

文章被以下专栏收录

支付技术那些事
支付产品、技术、投资理财渠道分享

进入专栏

推荐阅读

Oracle的数据 导入与导出-数据库（27）

Alan　发表于从Java...

数据库知识初步入门

1.关系数据库原理数据库有很多种类型，使用最广泛的为关系型数据库，如下图所示： 需要掌握的知识点：1、多张表的结构 2、各表之间的关系1.表的结构：2.各表之间的关系2.数据库管理系统-管...

优梨

数据库分片（Sharding）技术 - 文章写作框架

Introduction 导言本文中分片的概念与数据库分区的概念一致？本文中的分片会放到不同的物理节点上。 下面是本文目录，帮助您接下来的阅读What is Sharding? 什么是分片？分片（Sharding）是...

一路走好　发表于数据库Da...

4 条评论 切换为时间排序

写下你的评论...

yjj 10 个月前

赞

潇潇 9 个月前

写得很棒

赞

lalalal 9 个月前

写的真好

赞

Legendary 8 个月前

good

赞