1、面试题

如何设计可以动态扩容缩容的分库分表方案？

2、面试官心里分析

（1）选择一个数据库中间件，调研、学习、测试

（2）设计你的分库分表的一个方案，你要分成多少个库，每个库分成多少个表，3个库每个库4个表

（3）基于选择好的数据库中间件，以及在测试环境建立好的分库分表的环境，然后测试一下能否正常进行分库分表的读写

（4）完成单库单表到分库分表的迁移，双写方案

（5）线上系统开始基于分库分表对外提供服务

（6）扩容了，扩容成6个库，每个库需要12个表，你怎么来增加更多库和表呢？

这个是你必须面对的一个事儿，就是你已经弄好分库分表方案了，然后一堆库和表都建好了，基于分库分表中间件的代码开发啥的都好了，测试都ok了，数据能均匀分布到各个库和各个表里去，而且接着你还通过双写的方案咔嚓一下上了系统，已经直接基于分库分表方案在搞了。

那么现在问题来了，你现在这些库和表又支撑不住了，要继续扩容咋办？这个可能就是说你的每个库的容量又快满了，或者是你的表数据量又太大了，也可能是你每个库的写并发太高了，你得继续扩容。

这都是玩儿分库分表线上必须经历的事儿

3、面试题剖析

（1）停机扩容

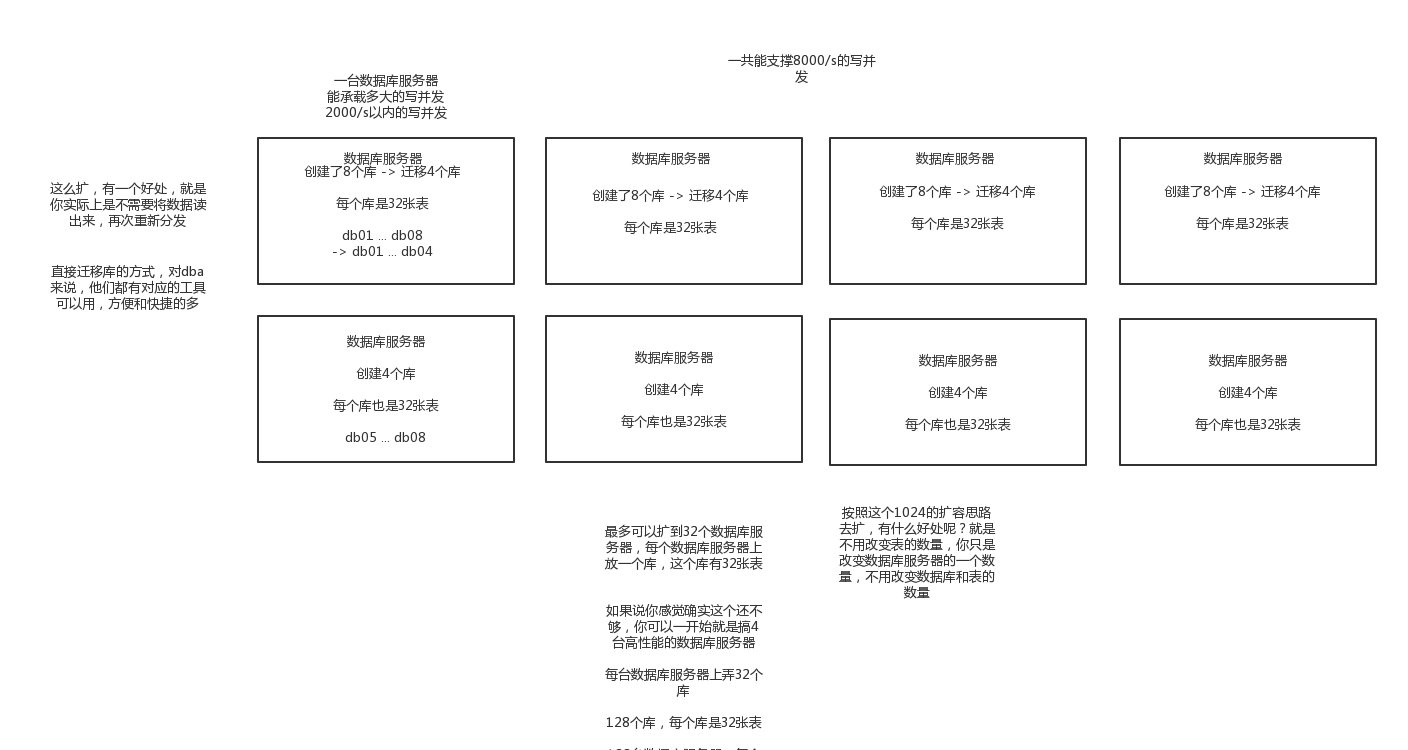
这个方案就跟停机迁移一样，步骤几乎一致，唯一的一点就是那个导数的工具，是把现有库表的数据抽出来慢慢倒入到新的库和表里去。但是最好别这么玩儿，有点不太靠谱，因为既然分库分表就说明数据量实在是太大了，可能多达几亿条，甚至几十亿，你这么玩儿，可能会出问题。

从单库单表迁移到分库分表的时候，数据量并不是很大，单表最大也就两三千万

写个工具，多弄几台机器并行跑，1小时数据就导完了

3个库+12个表，跑了一段时间了，数据量都1亿~2亿了。光是导2亿数据，都要导个几个小时，6点，刚刚导完数据，还要搞后续的修改配置，重启系统，测试验证，10点才可以搞完

1. 优化后的方案



一开始上来就是32个库，每个库32个表，1024张表

我可以告诉各位同学说，这个分法，第一，基本上国内的互联网肯定都是够用了，第二，无论是并发支撑还是数据量支撑都没问题

每个库正常承载的写入并发量是1000，那么32个库就可以承载32 \* 1000 = 32000的写并发，如果每个库承载1500的写并发，32 \* 1500 = 48000的写并发，接近5万/s的写入并发，前面再加一个MQ，削峰，每秒写入MQ 8万条数据，每秒消费5万条数据。

有些除非是国内排名非常靠前的这些公司，他们的最核心的系统的数据库，可能会出现几百台数据库的这么一个规模，128个库，256个库，512个库

1024张表，假设每个表放500万数据，在MySQL里可以放50亿条数据

每秒的5万写并发，总共50亿条数据，对于国内大部分的互联网公司来说，其实一般来说都够了

谈分库分表的扩容，第一次分库分表，就一次性给他分个够，32个库，1024张表，可能对大部分的中小型互联网公司来说，已经可以支撑好几年了

一个实践是利用32 \* 32来分库分表，即分为32个库，每个库里一个表分为32张表。一共就是1024张表。根据某个id先根据32取模路由到库，再根据32取模路由到库里的表。

刚开始的时候，这个库可能就是逻辑库，建在一个数据库上的，就是一个mysql服务器可能建了n个库，比如16个库。后面如果要拆分，就是不断在库和mysql服务器之间做迁移就可以了。然后系统配合改一下配置即可。

比如说最多可以扩展到32个数据库服务器，每个数据库服务器是一个库。如果还是不够？最多可以扩展到1024个数据库服务器，每个数据库服务器上面一个库一个表。因为最多是1024个表么。

这么搞，是不用自己写代码做数据迁移的，都交给dba来搞好了，但是dba确实是需要做一些库表迁移的工作，但是总比你自己写代码，抽数据导数据来的效率高得多了。

哪怕是要减少库的数量，也很简单，其实说白了就是按倍数缩容就可以了，然后修改一下路由规则。

对2 ^ n取模

orderId 模 32 = 库

orderId / 32 模 32 = 表

259 3 8

1189 5 5

352 0 11

4593 17 15

1、设定好几台数据库服务器，每台服务器上几个库，每个库多少个表，推荐是32库 \* 32表，对于大部分公司来说，可能几年都够了

2、路由的规则，orderId 模 32 = 库，orderId / 32 模 32 = 表

3、扩容的时候，申请增加更多的数据库服务器，装好mysql，倍数扩容，4台服务器，扩到8台服务器，16台服务器

4、由dba负责将原先数据库服务器的库，迁移到新的数据库服务器上去，很多工具，库迁移，比较便捷

5、我们这边就是修改一下配置，调整迁移的库所在数据库服务器的地址

6、重新发布系统，上线，原先的路由规则变都不用变，直接可以基于2倍的数据库服务器的资源，继续进行线上系统的提供服务