Mysql分库分表方案

1.为什么要分表：

当一张表的数据达到几千万时，你查询一次所花的时间会变多，如果有联合查询的话，我想有可能会死在那儿了。

**分表的目的就在于此，减小数据库的负担，缩短查询时间。**

mysql中有一种机制是表锁定和行锁定，是为了保证数据的完整性。表锁定表示你们都不能对这张表进行操作，必须等我对表操作完才行。行锁定也一样，别的sql必须等我对这行数据操作完了，才能对这条数据进行操作。

**2. mysql proxy：amoeba**

**做mysql集群,利用amoeba，对于上层JAVA是透明的，看不清的。**

从上层的java程序来讲，不需要知道主服务器和从服务器的来源，即**主从数据库服务器对于上层来讲是透明的**。可以通过amoeba来配置。

**3.大数据量并且访问频繁的表，将其分为若干个表**

比如对于某网站平台的数据库表-公司表，数据量很大，这种能预估出来的大数据量表，我们就事先分出个N个表，这个N是多少，根据实际情况而定。

     某网站现在的数据量至多是5000万条，可以设计每张表容纳的数据量是500万条，也就是拆分成10张表，

那么如何判断某张表的数据是否容量已满呢？可以在程序段对于要新增数据的表，在插入前先做统计表记录数量的操作，当<500万条数据，就直接插入，当已经到达阀值，可以在程序段新创建数据库表（或者已经事先创建好），再执行插入操作。

**4. 利用merge存储引擎来实现分表**

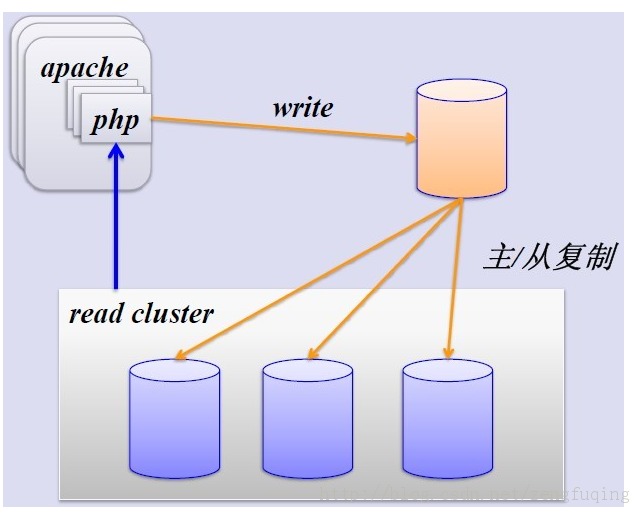
如果要把已有的大数据量表分开比较痛苦，最痛苦的事就是改代码，因为程序里面的sql语句已经写好了。**用merge存储引擎来实现分表, 这种方法比较适合.**

 ------------------- ----------华丽的分割线--------------------------------------

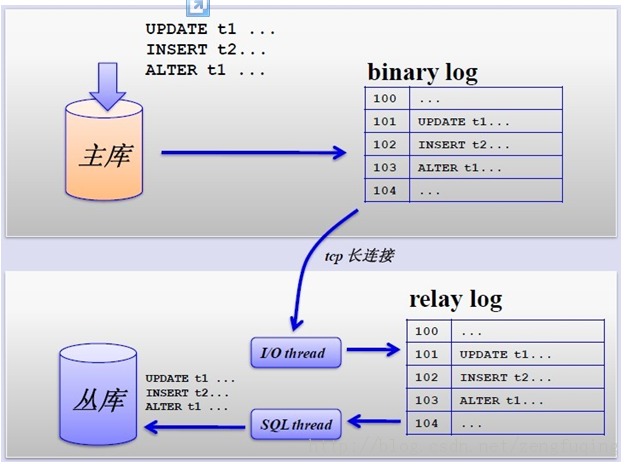
 数据库架构

**1、简单的MySQL主从复制:**

**MySQL的主从复制解决了数据库的读写分离，并很好的提升了读的性能**，其图如下：

[](http://static.oschina.net/uploads/img/201402/13104102_nxBO.jpg)

其主从复制的过程如下图所示：

[](http://static.oschina.net/uploads/img/201402/13104103_7YzJ.jpg)

1、**Master将数据改变记录到二进制日志(binary log)中**，也就是配置文件log-bin指定的文件，这些记录叫做二进制日志事件(binary log events)   
2、Slave**通过I/O线程读取Master中的binary log events并写入到它的中继日志(relay log)**3、Slave重做中继日志中的事件，**把中继日志中的事件信息一条一条的在本地执行一次，完成数据在本地的存储**，从而实现将改变反映到它自己的数据(数据重放)

但是，主从复制也带来其他一系列性能瓶颈问题：

**1. 写入无法扩展**

**2. 写入无法缓存**

**3. 复制延时**

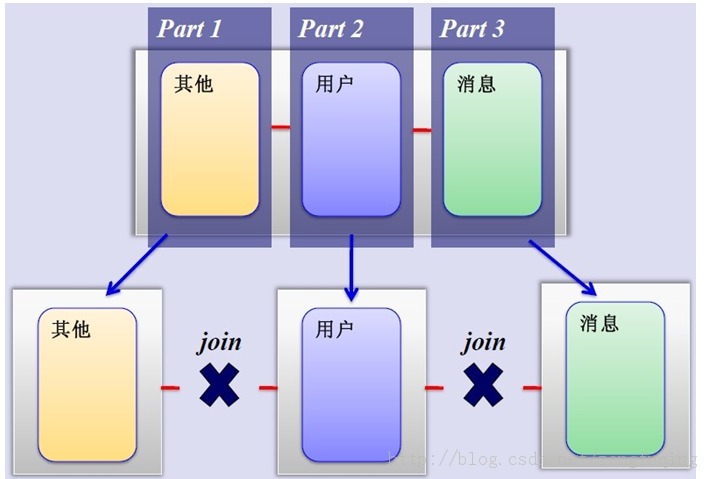
**4. 锁表率上升**

**5. 表变大，缓存率下降**

那问题产生总得解决的，这就产生下面的优化方案，一起来看看。

2、MySQL垂直分区

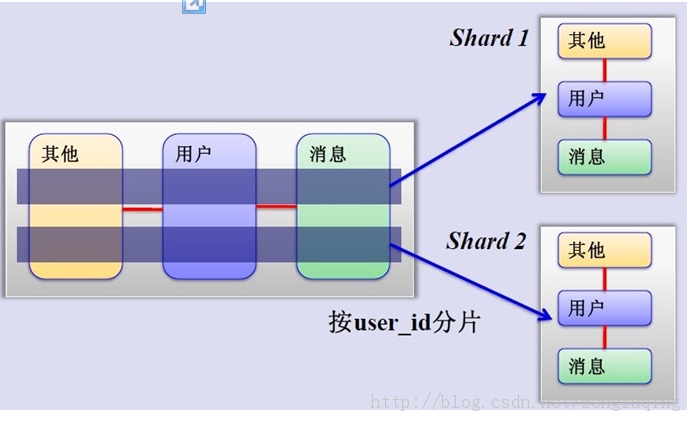
**如果把业务切割得足够独立，那把不同业务的数据放到不同的数据库服务器将是一个不错的方案，**而且万一其中一个业务崩溃了也不会影响其他业务的正常进行，并且也起到了负载分流的作用，大大提升了数据库的吞吐能力。经过垂直分区后的数据库架构图如下：

[](http://static.oschina.net/uploads/img/201402/13104103_CHZw.jpg)

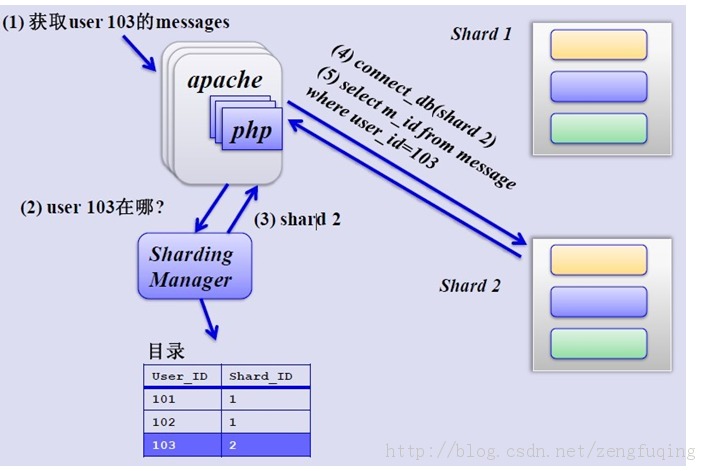
然而，**尽管业务之间已经足够独立了，但是有些业务之间或多或少总会有点联系，**如用户，基本上都会和每个业务相关联，况且这种分区方式，也不能解决单张表数据量暴涨的问题，因此为何不试试水平分割呢？

**3、MySQL水平分片（Sharding）**

这是一个非常好的思路，将用户按一定规则（按id哈希）分组，并把该组用户的数据存储到一个数据库分片中，即一个sharding，这样随着用户数量的增加，只要简单地配置一台服务器即可，原理图如下：

[](http://static.oschina.net/uploads/img/201402/13104104_2qZP.jpg)

如何来确定某个用户所在的shard呢，可以建一张**用户和shard对应的数据表，每次请求先从这张表找用户的shard id，再从对应shard中查询相关数据，如下图所示：**

[](http://static.oschina.net/uploads/img/201402/13104105_7oa0.jpg)

**分表的方法有：**

**单库单表 (自己的项目最经常用的)**  
  
单库单表是最常见的数据库设计，例如，**有一张用户(user)表放在数据库db中，所有的用户都可以在db库中的user表中查到。**

**单库多表**

随着用户数量的增加，user表的数据量会越来越大，当数据量达到一定程度的时候对user表的查询会渐渐的变慢，从而影响整个DB的性能。**如果使用mysql, 还有一个更严重的问题是，当需要添加一列的时候，mysql会锁表，期间所有的读写操作只能等待。**  
可以通过某种方式将user表进行**水平的切分**，产生两个表结构完全一样的user\_0000,user\_0001等表，**user\_0000 + user\_0001 + …的数据刚好是一份完整的数据。** 

**多库多表**  
  
         随着数据量增加也许单台DB的存储空间不够，随着查询量的增加，单台数据库服务器已经没办法支撑。这个时候**可以再对数据库进行水平区分。**  
 **分库分表规则**  
         设计表的时候需要确定此表按照什么样的规则进行分库分表。

例如，当有新用户时，程序得确定将此用户信息添加到哪个表中；同理，当登录的时候我们得通过用户的账号找到数据库中对应的记录，所有的这些都需要按照某一规则进行。

**通过分库分表规则查找到对应的表和库的过程**。如分库分表的规则是user\_id mod 4的方式，当用户新注册了一个账号，账号id的123,我们可以通过id mod 4的方式确定此账号应该保存到User\_0003表中。当用户123登录的时候，我们通过123 mod 4后确定记录在User\_0003中。

**分库分表产生的问题，及注意事项**  
1.   **分库分表维度的问题** 假如用户购买了商品,需要将交易记录保存取来，如果按照**用户的维度分表**，则每个用户的交易记录都保存在同一表中，所以**很快很方便的查找到某用户的购买情况**，但是某商品被购买的情况则很有可能分布在多张表中，查找起来比较麻烦。

反之，按照商品维度分表，可以很方便的查找到此商品的购买情况，但要查找到买人的交易记录比较麻烦。   
  
所以常见的解决方式有：   
  
     a.通过扫表的方式解决，此方法基本不可能，效率太低了。   
  
**b.记录两份数据，一份按照用户纬度分表，一份按照商品维度分表。 （查用户就看用户表，查商品就看商品表）  
  
     c.通过搜索引擎解决，但如果实时性要求很高，又得关系到实时搜索。**  
**2.   联合查询的问题**  
  
 联合查询基本不可能，因为关联的表有可能不在同一数据库中。   
  
**3.   避免跨库事务 （和4类似）**  
避免在一个事务中修改db0中的表的时候同时修改db1中的表，一个是操作起来更复杂，效率也会有一定影响。   
  
**4.   尽量把同一组数据放到同一DB服务器上**  
  
例如将卖家a的商品和交易信息都放到db0中，当db1挂了的时候，卖家a相关的东西可以正常使用。也就是说避免数据库中的数据依赖另一数据库中的数据。 

**分库分表策略：  
一主多备**  
  
 在实际的应用中，绝大部分情况都是读远大于写。**Mysql提供了读写分离的机制，所有的写操作都必须对应到Master，读操作可以在 Master和Slave机器上进行，Slave与Master的结构完全一样，一个Master可以有多个Slave,甚至Slave下还可以挂 Slave,通过此方式可以有效的提高DB集群的 QPS.**

所有的写操作都是先在Master上操作，然后同步更新到Slave上，所以**从Master同步到Slave机器有一定的延迟，当系统很繁忙的时候，延迟问题会更加严重，Slave机器数量的增加也会使这个问题更加严重。**  
  
  **此外，可以看出Master是集群的瓶颈，当写操作过多，会严重影响到Master的稳定性，如果Master挂掉，整个集群都将不能正常工作。**  
  
所以，**1. 当读压力很大的时候，可以考虑添加Slave机器的分式解决，但是当Slave机器达到一定的数量就得考虑分库了。**

**2. 当写压力很大的时候，就必须得进行分库操作。**  
  
---------------------------------------------   
  
MySQL使用为什么要分库分表   
**可以用说用到MySQL的地方,只要数据量一大, 马上就会遇到一个问题,要分库分表.**  
这里引用一个问题为什么要分库分表呢?MySQL处理不了大的表吗?   
其实是可以处理的大表的.我所经历的项目中单表物理上文件大小在80G多,单表记录数在5亿以上,而且这个表   
属于一个非常核心常用的表:朋友关系表.   
  
但这种方式可以说不是一个最佳方式. 因为面临文件系统如Ext3文件系统对大于大文件处理上也有许多问题.   
 这个层面可以用xfs文件系统进行替换.但MySQL单表太大后有一个问题是不好解决:

表结构调整相关的操作基本不在可能.所以大项在使用中都会面监着分库分表的应用.   
  
从Innodb本身来讲数据文件的Btree上只有两个锁, 叶子节点锁和子节点锁,可以想而知道,当发生页拆分或是**添加新叶时都会造成表里不能写入数据.   
所以分库分表还就是一个比较好的选择了.**  
  
那么分库分表多少合适呢?   
 经测试在单表1000万条记录以下,写入读取性能是比较好的. 这样在留点buffer,那么单表全是数据字型的保持在 800万条记录以下, 有字符型的单表保持在500万以下.   
 **如果按 100库100表来规划,如用户业务:   
500万\*100\*100 = 50000000万 = 5000亿记录.**

心里有一个数了,按业务做规划还是比较容易的.