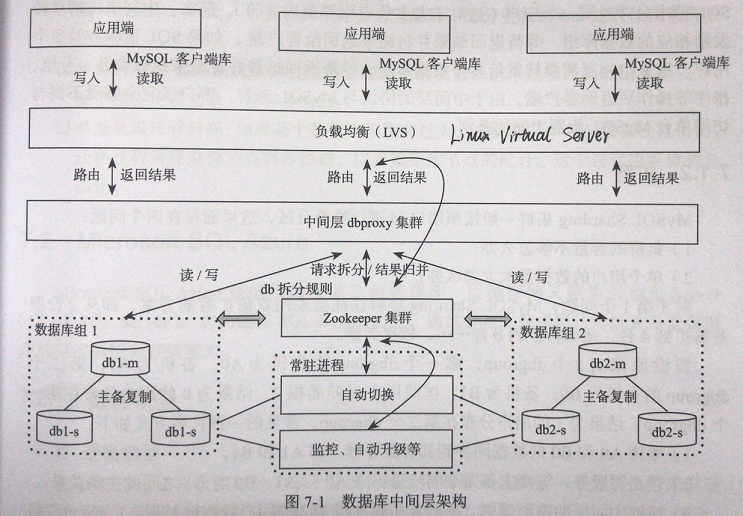
# mysql sharding

## 架构

最简单也是最常见的做法就是在应用层按照规则将数据拆分，分布到多个数据库节点上，需要引入一个中间层对应用屏蔽后端数据库拆分的细节。



在mysql sharding架构中，有中间层dbproxy集群、数据库组、元数据服务器、常驻进程这些部分。

1. dbproxy中间层

中间层解析客户端的sql请求并转发到后端的数据库。它要负责解析sql语句、执行sql路由、结果归并、排序以及分组等，中间层由多个无状态的dbproxy进程组成。可以在客户端和中间层之间引入LVS对客户端请求进行负载均衡，但是LVS本身有一层通信开销，所以常见的做法是直接在客户端配置中间层服务器的列表，让客户端自己处理请求负载均衡以及中间层故障等问题。

1. dbgroup数据库组

每个dbgroup由N台数据库机器组成，其中一台为master，剩下的为slave。master负责所有的写事务及强一致读事务，并将操作以binlog的形式复制到slave。

1. 元数据服务器

元数据服务器主要负责dbgroup拆分规则并用于dbgroup选主。dbproxy通过元数据服务器获取拆分规则从而确定SQL语句的执行计划。dbgroup的选主也需要通过元数据服务器进行。元数据本身也需要多个副本实现HA（High Available），常见的方式是采用Zookeeper。

1. 常驻进程

部署在**每台数据库服务器**上的常驻进程，用于实现监控、单点切换、安装和卸载程序等。dbgroup中的数据库需要进行准备切换，这些控制逻辑需要与数据库读写事务处理逻辑隔离开。

如果数据库安装用户ID哈希分区，那么同一个用户的数据分布在一个dbgroup上。如果一个sql请求只涉及到一个用户，那么中间层将请求转发给对应的一个dbgroup，等待返回的结果转发给客户端就可以了；如果sql涉及到多个用户，那么中间层需要转发给多个dbgroup，等待返回的结果执行合并、分组、排序等操作，然后再返回给客户端。由于中间层的协议与mysql兼容，客户端完全感受不到与访问单台mysql机器之间的差别。

## 扩容

mysql sharding集群通常采用双倍扩容的方案。假设有2个dbgroup，第一个dbgroup有主机a0和a1，第二个dbgroup有主机b0和b1。所以当前的状况是按照用户id哈希对2取模，按照哈希结果选择对应的dbgroup。

现在对这个集群进行扩容：

1. 等待a0和b0的数据分别同步a1和b1；
2. **停止写服务**，解除a0与a1，b0与b1之间的主备关系；
3. 修改中间层的映射规则，将哈希对4取模结果是2的用户数据映射到a1，结果是3的映射到b1；
4. 开启写服务；
5. 分别给a0、a1、b0和b1增加一台备份机。

这样下来，原来2个dbgroup变成了4个dbgroup。可以看到扩容的过程中集群需要停一会服务。

对于单个用户数据量过大的情况，可以在应用层定期统计出大用户，并且这些用户的数据按照数据量拆分到多个dbgroup。定期维护这些信息对应用层来说是一个很大的代价。

## 问题

引入数据库中间层将后端分库分表对应用透明化，在大型互联网公司内部很常见。但是也面临一些问题：

1. 数据库复制：mysql主备之间只支持异步的复制，而且主库压力较大时可能产生很大的延迟，因此主备切换时可能会丢失最后一部分更新事务。
2. 扩容问题：增加新机器涉及到数据的重新划分，过程比较复杂，容易出错。
3. 动态数据迁移问题：某个dbgroup压力过大时，需要将其中部分数据迁移出去，迁移过程需要总控节点整体协调，很难做到自动化。