MySQL分布式中间件

业界主要产品 ：

1. 上夜班数据库 ： MySQL Proxy
2. Amoeba
3. Cobar

地址 ：

cobar：https://github.com/alibaba/cobar

cobarclient：https://github.com/alibaba/cobarclient

4. guzz
5. TDDL

地址 ：https://github.com/alibaba/tb_tddl

参考 ： https://www.guokr.com/blog/475765/

http://blog.csdn.net/wkwanglei/article/details/49738787

mysql-proxy是官方提供的mysql中间件产品可以实现负载平衡，读写分离，failover等，但其不支持大数据量的分库分表且性能较差。下面介绍几款能代替其的mysql开源中间件产品，Atlas，cobar，tddl，让我们看看它们各自有些什么优点和新特性吧。

Atlas

Atlas是由 Qihoo 360, Web平台部基础架构团队开发维护的一个基于MySQL协议的数据中间层项目。它是在mysql-proxy 0.8.2版本的基础上，对其进行了优化，增加了一些新的功能特性。360内部使用Atlas运行的mysql业务，每天承载的读写请求数达几十亿条。

Altas架构：

Atlas是一个位于应用程序与MySQL之间，它实现了MySQL的客户端与服务端协议，作为服务端与应用程序通讯，同时作为客户端与MySQL通讯。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。

Altas的一些新特性：

1.主库宕机不影响读

主库宕机，Atlas自动将宕机的主库摘除，写操作会失败，读操作不受影响。从库宕机，Atlas自动将宕机的从库摘除，对应用没有影响。在mysql官方的proxy中主库宕机，从库亦不可用。

2.通过管理接口，简化管理工作，DB的上下线对应用完全透明，同时可以手动上下线。

3.自己实现读写分离

（1）为了解决读写分离存在写完马上就想读而这时可能存在主从同步延迟的情况，Altas中可以在SQL语句前增加 /*master*/ 就可以将读请求强制发往主库。

（2）如图2中，主库可设置多项，用逗号分隔，从库可设置多项和权重，达到负载均衡。

4.自己实现分表（图3）

（1）需带有分表字段。

（2）支持SELECT、INSERT、UPDATE、DELETE、REPLACE语句。

（3）支持多个子表查询结果的合并和排序。

这里不得不吐槽Atlas的分表功能，不能实现分布式分表，所有的子表必须在同一台DB的同一个database里且所有的子表必须事先建好，Atlas没有自动建表的功能。

5.之前官方主要功能逻辑由使用lua脚本编写，效率低，Atlas用C改写，QPS提高，latency降低。

6.安全方面的提升

（1）通过配置文件中的pwds参数进行连接Atlas的用户的权限控制。

（2）通过client-ips参数对有权限连接Atlas的ip进行过滤。

（3）日志中记录所有通过Altas处理的SQL语句，包括客户端IP、实际执行该语句的DB、执行成功与否、执行所耗费的时间 ，如下面例子（图4）。

7.平滑重启

通过配置文件中设置lvs-ips参数实现平滑重启功能，否则重启Altas的瞬间那些SQL请求都会失败。该参数前面挂接的lvs的物理网卡的ip，注意不是虚ip。平滑重启的条件是至少有两台配置相同的Atlas且挂在lvs之后。

source：https://github.com/Qihoo360/Atlas

alibaba.cobar

Cobar是阿里巴巴（B2B）部门开发的一种关系型数据的分布式处理系统，它可以在分布式的环境下看上去像传统数据库一样为您提

供海量数据服务。那么具体说说我们为什么要用它，或说cobar--能干什么？以下是我们业务运行中会存在的一些问题：
1.随着业务的进行数据库的数据量和访问量的剧增，需要对数据进行水平拆分来降低单库的压力，而且需要高效且相对透明的来屏蔽掉水平拆分的细节。
2.为提高访问的可用性，数据源需要备份。
3.数据源可用性的检测和failover。
4.前台的高并发造成后台数据库连接数过多，降低了性能，怎么解决。
针对以上问题就有了cobar施展自己的空间了，cobar中间件以proxy的形式位于前台应用和实际数据库之间，对前台的开放的接口是mysql通信协议。将前台SQL语句变更并按照数据分布规则转发到合适的后台数据分库，再合并返回结果，模拟单库下的数据库行为。

应用介绍：
1.通过Cobar提供一个名为test的数据库，其中包含t1,t2两张表。后台有3个MySQL实例(ip:port)为其提供服务，分别为：A,B,C。
2.期望t1表的数据放置在实例A中，t2表的数据水平拆成四份并在实例B和C中各自放两份。t2表的数据要具备HA功能，即B或者C实例其中一个出现故障，不影响使用且可提供完整的数据服务。
cabar优点总结：
1.数据和访问从集中式改变为分布：
（1）Cobar支持将一张表水平拆分成多份分别放入不同的库来实现表的水平拆分
（2）Cobar也支持将不同的表放入不同的库
（3） 多数情况下，用户会将以上两种方式混合使用
注意！：Cobar不支持将一张表，例如test表拆分成test_1,test_2, test_3.....放在同一个库中，必须将拆分后的表分别放入不同的库来实现分布式。
2.解决连接数过大的问题。
3.对业务代码侵入性少。
4.提供数据节点的failover,HA：
(1)Cobar的主备切换有两种触发方式，一种是用户手动触发，一种是Cobar的心跳语句检测到异常后自动触发。那么，当心跳检测到主机异常，切换到备机，如果主机恢复了，需要用户手动切回主机工作，Cobar不会在主机恢复时自动切换回主机，除非备机的心跳也返回异常。
(2)Cobar只检查MySQL主备异常，不关心主备之间的数据同步，因此用户需要在使用Cobar之前在MySQL主备上配置双向同步。
cobar缺点：
开源版本中数据库只支持mysql，并且不支持读写分离。
source：http://code.alibabatech.com/wiki/display/cobar/Home

TDDL
淘宝根据自己的业务特点开发了TDDL（Taobao Distributed Data Layer 外号:头都大了 ©_Ob）框架，主要解决了分库分表对应用的透明化以及异构数据库之间的数据复制，它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。
TDDL所处的位置（tddl通用数据访问层，部署在客户端的jar包，用于将用户的SQL路由到指定的数据库中）：
淘宝很早就对数据进行过分库的处理， 上层系统连接多个数据库，中间有一个叫做DBRoute的路由来对数据进行统一访问。DBRoute对数据进行多库的操作、数据的整合，让上层系统像操作一个数据库一样操作多个库。但是随着数据量的增长，对于库表的分法有了更高的要求，例如，你的商品数据到了百亿级别的时候，任何一个库都无法存放了，于是分成2个、4个、8个、16个、32个……直到1024个、2048个。好，分成这么多，数据能够存放了，那怎么查询它？这时候，数据查询的中间件就要能够承担这个重任了，它对上层来说，必须像查询一个数据库一样来查询数据，还要像查询一个数据库一样快（每条查询在几毫秒内完成），TDDL就承担了这样一个工作。在外面有些系统也用DAL（数据访问层） 这个概念来命名这个中间件。
下图展示了一个简单的分库分表数据查询策略：
主要优点：
1.数据库主备和动态切换
2.带权重的读写分离
3.单线程读重试
4.集中式数据源信息管理和动态变更
5.剥离的稳定jboss数据源
6.支持mysql和oracle数据库
7.基于jdbc规范，很容易扩展支持实现jdbc规范的数据源
8.无server,client-jar形式存在，应用直连数据库

9.读写次数,并发度流程控制，动态变更

10.可分析的日志打印,日志流控，动态变更

TDDL必须要依赖diamond配置中心（diamond是淘宝内部使用的一个管理持久配置的系统，目前淘宝内部绝大多数系统的配置，由diamond来进行统一管理，同时diamond也已开源）。

TDDL动态数据源使用示例说明：http://rdc.taobao.com/team/jm/archives/1645

diamond简介和快速使用：http://jm.taobao.org/tag/diamond%E4%B8%93%E9%A2%98/

TDDL源码：https://github.com/alibaba/tb_tddl

TDDL复杂度相对较高。当前公布的文档较少，只开源动态数据源，分表分库部分还未开源，还需要依赖diamond，不推荐使用。

终其所有，我们研究中间件的目的是使数据库实现性能的提高。具体使用哪种还要经过深入的研究，严谨的测试才可决定。