双十一电商项目-数据库读写分离解决方案

1、读写分离背景介绍

2、主从复制

3、Altas 读写分离实现

4、Sharding-jdbc 读写分离实现

5、总结对比

课堂源码会上传到图灵源码中心

什么是读写分离：

我们一般应用访问数据库无非是读取数据、修改数据、插入数据、删除数据。

而我们对数据库一般分为：master（主库也是写库） slave（从库也为读库）

而读写分离的意思就是：所有的写（insert update delete）操作走主库、其他走从库。

目的是什么了？

我们一般应用对数据库而言都是“读多写少”，也就是说对数据库读取数据的压力比较大。

读写分离的主要目的是降低主库的压力。降低主库的读的压力

前提条件：

1、读库 slave 需要跟写库 master 的数据一致

2、写数据必须写到证据库

3、读取数据必须到读库（不一定）

行业用的多的：1 主多从 mysql 集群方案

至少两个库

主从复制：

Master my.cnf 配置：

```
binlog-do-db=tlshop

binlog-ignore-db=mysql

binlog_format=mixed

log-bin=mysql-bin

server-id=1
```

Slave my.cnf 配置

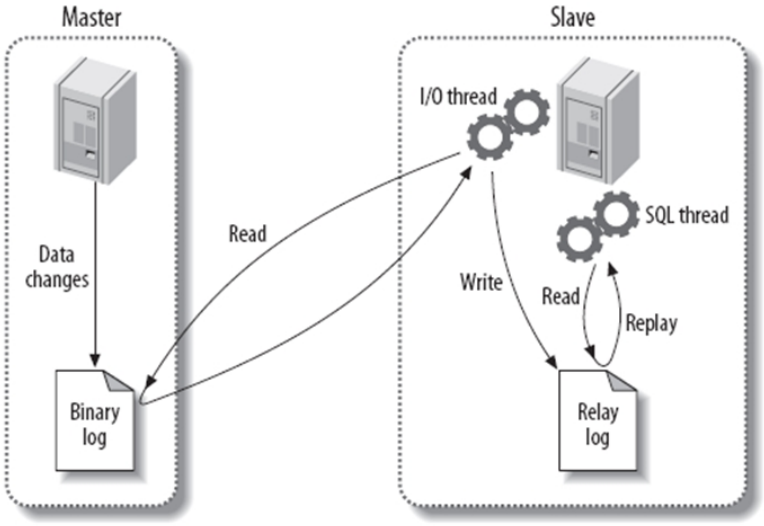
```
replicate-do-db=tlshop

replicate-ignore-db=mysql

server-id=2

#5.6 之后版本不管用

master-host=192.168.0.15

master-port=3306

master-user=root

master-password=123456
```

动态配置 slave 节点信息

change master to master_host='192.168.0.15', master_user='root', master_password='123456', master_log_file='', master_log_pos=;

slave start;//启动

主从复制：



数据异步复制、全同步复制、半同步

读写分离实现：

业界方案：

代理层 proxy：Atlas 开源软件

应用层：Sharding-jdbc

代理层 Atlas：

Atlas 是由 Qihoo 360 公司 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。它在 MySQL 官方推出的 MySQL-Proxy 0.8.2 版本的基础上，修改了大量 bug，添加了很多功能特性。目前该项目在 360 公司内部得到了广泛应用，很多 MySQL 业

务已经接入了 Atlas 平台，每天承载的读写请求数达几十亿条。同时，有超过 50 家公司在生产环境中部署了 Atlas，超过 800 人已加入了我们的开发者交流群，并且这些数字还在不断增加。

主要功能：

1.读写分离

2.从库负载均衡

3.IP 过滤

4.自动分表

5.DBA 可平滑上下线 DB

6.自动摘除宕机的 DB

https://github.com/Qihoo360/Atlas

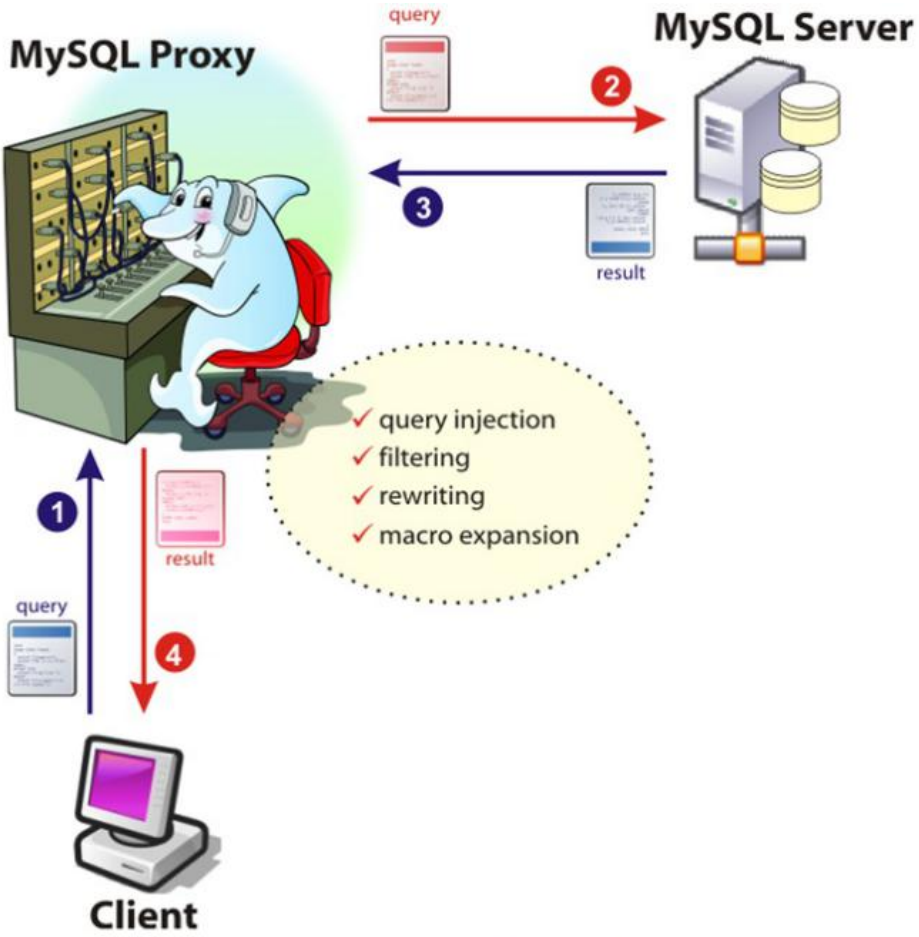


图 1：Atlas 架构图形象表示

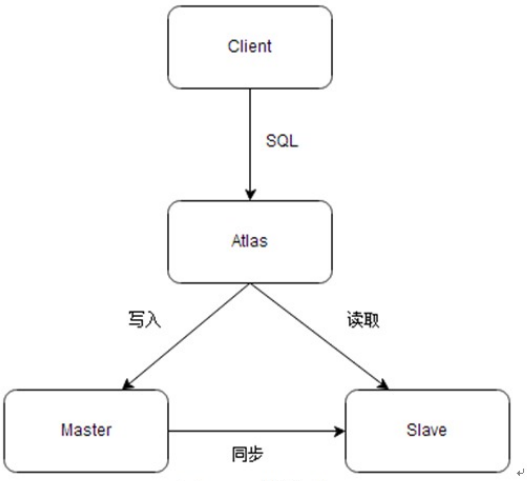


图 2 Atlas 总体架构

Atlas 配置文件：

```
#带#号的为非必需的配置项目

#管理接口的用户名
admin-username = root

#管理接口的密码
admin-password =123456

#Atlas后端连接的MySQL主库的IP和端口，可设置多项，用逗号分隔
proxy-backend-addresses =192.168.0.15:3306

#Atlas后端连接的MySQL从库的IP和端口，@后面的数字代表权重，用来作负载均衡，若省略则默认
proxy-read-only-backend-addresses =192.168.0.16:3306@1

#用户名与其对应的加密过的MySQL密码，密码使用PREFIX/bin目录下的加密程序encrypt加密，下
pwds = root:/iZxz+0GRoA=

#设置Atlas的运行方式，设为true时为守护进程方式，设为false时为前台方式，一般开发调试时
daemon = true

#设置Atlas的运行方式，设为true时Atlas会启动两个进程，一个为monitor，一个为worker，moni
时设为false，线上运行时设为true,true后面不能有空格。
keepalive = true

#工作线程数，对Atlas的性能有很大影响，可根据情况适当设置
event-threads = 8

#日志级别，分为message、warning、critical、error、debug五个级别
log-level = message

#日志存放的路径
log-path = /usr/local/mysql-proxy/log
```

强制路由：

注释的方式强制走主库。

Alatas：/*master*/ 业务需要

下单即查

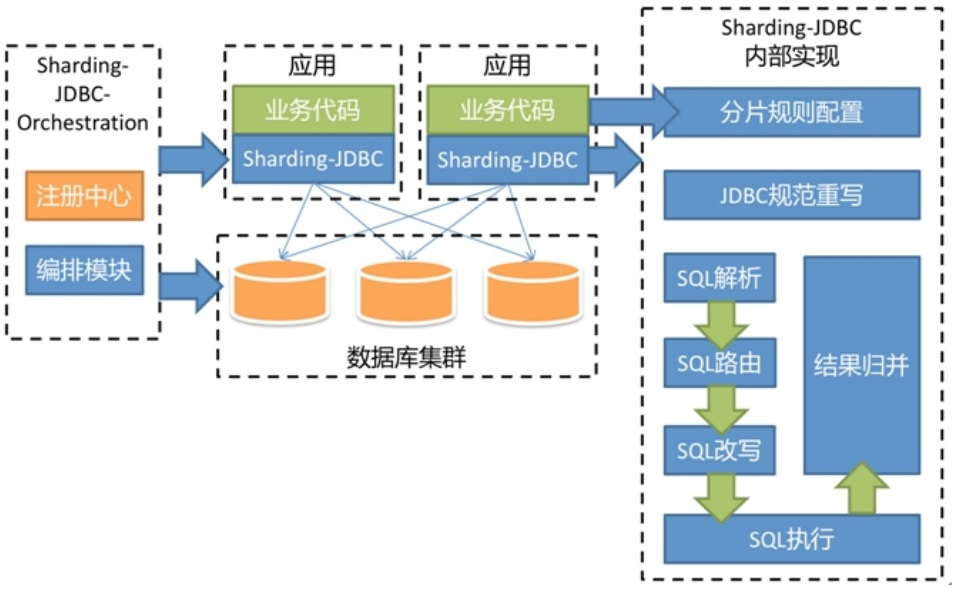
Sharding-jdbc

Sharding-JDBC 是一个开源的分布式数据库中间件，它无需额外部署和依赖，完全兼容 JDBC 和各种 ORM 框架。Sharding-JDBC 作为面向开发的微服务云原生基础类库，完整的实现了分库分表、读写分离和分布式主键功能，并初步实现了柔性事务。

http://shardingjdbc.io/index_zh.html

```
<dependency>
    <groupId>io.shardingjdbc</groupId>
    <artifactId>sharding-jdbc-core-spring-namespace</artifactId>
    <version>${sharding-jdbc.version}</version>
</dependency>

<dependency>
    <groupId>io.shardingjdbc</groupId>
    <artifactId>sharding-jdbc-core</artifactId>
    <version>${sharding-jdbc.version}</version>
</dependency>
```

```
<master-slave:data-source id="dataSource" master-data-source-name="dataSourceMaster" slave-data-source-names="dataSourceSlave,dataSourceSlave,dataSourceSlave" strategy-type="ROUND_ROBIN" />
```



强制路由：

下单即查

不支持注释方式

hint

总结：

Alatas:

1、程序不需要管主从配置的具体细节

2、实现原理是 proxy，所以性能上会下降

3、而且需要维护其高可用

4、减少了程序员技能要求

5、只支持 mysql

Sharding-jdbc：

2、主从配置在程序中，所以增加了程序员的技术要求

3、实现原理是 jdbc 增强，所以支持任何数据库类型 性能比上面那个强

4、 而且不需要维护。

5、Mysql、Oracle、sql server