1. MySQL读写分离概念：

MYSQL读写分离的原理其实就是让Master数据库处理事务性增、删除、修改、更新操作（CREATE、INSERT、UPDATE、DELETE），而让Slave数据库处理SELECT操作，MYSQL读写分离前提是基于MYSQL主从复制，这样可以保证在Master上修改数据，Slave同步之后，WEB应用可以读取到Slave端的数据。

1.1 读写分离实现方式：

实现MYSQL读写分离可以基于第三方插件，也可以通过开发修改代码实现，具体实现的读写分离的常见方式有如下四种：

- Amoeba读写分离；
- MySQL-Proxy读写分离；
- Mycat读写分离；
- 基于程序读写分离(效率很高，实施难度大，开发改代码)；

Amoeba是阿里08年开源的以MySQL为底层数据存储，并对WEB、APP应用提供MySQL协议接口的proxy。它集中地响应WEB应用的请求，依据用户事先设置的规则，将SQL请求发送到特定的数据库上执行，基于此可以实现负载均衡、读写分离、高可用性等需求。

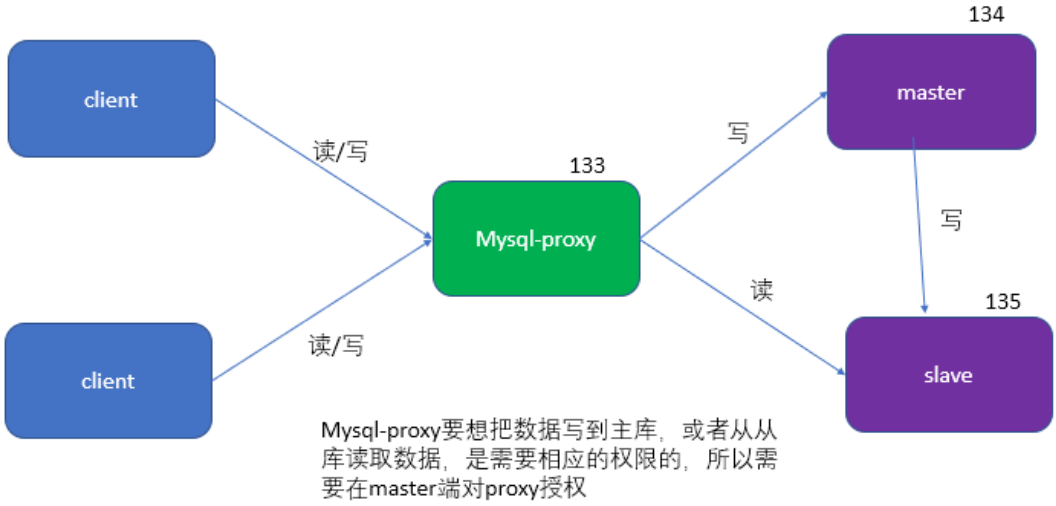
Mysql-Proxy是MySQL官方提供的mysql中间件服务，支持无数客户端连接，同时后端可连接若干台Mysql-Server服务器，MYSQL-Proxy自身基于MySQL协议，连接MYSQL-Proxy的客户端无需修改任何设置， 跟正常连接MYSQL Server没有区别，无需修改程序代码。

Mycat是基于阿里12年开源的cobar开发的一个数据库中间件，在架构体系中是位于数据库和应用层之间的一个组件，并且对于应用层是透明的，它可实现读写分离，分库分表。

2. 基于mysql-proxy实现读写分离：

```
proxy:  192.168.75.133
master: 192.168.75.134
slave:  192.168.75.135
```

2.1 工作原理图解：



2.2 配置proxy:

proxy可以选择和mysql部署在同一台服务器，也可以选择单独部署在另一台独立服务器。

```
# 下载mysql-proxy：
wget http://mirrors.163.com/mysql/Downloads/MySQL-Proxy/mysql-proxy-0.8.4-linux-el6-x86-64bit.tar.gz
# 解压：
tar xf mysql-proxy-0.8.4-linux-el6-x86-64bit.tar.gz
mv mysql-proxy-0.8.4-linux-el6-x86-64bit /usr/local/mysql-proxy
# 配置环境变量：
[root@node3 src]# echo "export PATH=/usr/local/mysql-proxy/bin:$PATH" > /etc/profile.d/mysql-proxy.sh
[root@node3 src]# . /etc/profile.d/mysql-proxy.sh
# 启动MYSQL-Proxy中间件：
[root@node3 src]# useradd  -r  mysql-proxy
[root@node3 src]# mysql-proxy --daemon --log-level=debug --user=mysql-proxy --keepalive --log-file=/var/log/mysql-proxy.log --plugins="proxy" --proxy-backend-addresses="192.168.75.134:3306" --proxy-read-only-backend-addresses="192.168.75.135:3306" --proxy-lua-script="/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua" --plugins=admin --admin-username="admin" --admin-password="admin" --admin-lua-script="/usr/local/mysql-proxy/lib/mysql-proxy/lua/admin.lua"
```

```
# 查看端口/日志：
[root@node3 src]# netstat -ntlp |grep 40
tcp        0      0 0.0.0.0:4040            0.0.0.0:*
          LISTEN      1348/mysql-proxy
tcp        0      0 0.0.0.0:4041            0.0.0.0:*
          LISTEN      1348/mysql-proxy
```

2.3 启动的相关参数：

```
# Mysql-Proxy的相关参数详解如下：
--help-all                                  ：获取全部帮助信息；
--proxy-address=host:port                   ：代理服务监听的地址和端口，默认为4040；
--admin-address=host:port                   ：管理模块监听的地址和端口，默认为4041；
--proxy-backend-addresses=host:port    ：后端mysql服务器的地址和端口；
--proxy-read-only-backend-addresses=host:port ：后端只读mysql服务器的地址和端口；
--proxy-lua-script=file_name            ：完成mysql代理功能的Lua脚本；
--daemon                                ：以守护进程模式启动mysql-proxy；
--keepalive                             ：在mysql-proxy崩溃时尝试重启之；
--log-file=/path/to/log_file_name          ：日志文件名称；
--log-level=level                       ：日志级别；
--log-use-syslog                        ：基于syslog记录日志；
--plugins=plugin                        ：在mysql-proxy启动时加载的插件；
--user=user_name                        ：运行mysql-proxy进程的用户；
--defaults-file=/path/to/conf_file_name ：默认使用的配置文件路径，其配置段使用[mysql-proxy]标识；
--proxy-skip-profiling                  ：禁用profile；
--pid-file=/path/to/pid_file_name          ：进程文件名；
```

2.4 启动master/slave;

```
systemctl start mariadb
```

2.5 查看读写分离状态：

基于4041端口MySQL-Proxy查看读写分离状态，登录4041管理端口 ：

```
mysql -h192.168.75.133 -uadmin -padmin -P4041

# 这时可以看到后端数据库信息，只是状态为unknown,表示还没有客户端连接，可以通过4040代理端口通过
查询数据等操作激活。
mysql> select * from backends;
+-------------+---------------------+---------+------+----
--+-------------------+
| backend_ndx | address             | state   | type |
uuid | connected_clients |
+-------------+---------------------+---------+------+----
--+-------------------+
|           1 | 192.168.75.134:3306 | unknown | rw   |
NULL |                 0 |
|           2 | 192.168.75.135:3306 | unknown | ro   |
NULL |                 0 |
+-------------+---------------------+---------+------+----
--+-------------------+
2 rows in set (0.00 sec)
```

2.5 授权proxy:

```
grant all on *.* to "mysql-proxy"@"192.168.75.133"
identified by "123456";
```

2.6 通过代理创建数据：

通过**4040代理端口**插入数据，该sql语句会走master，于是可以激活master状态：

```
mysql -h192.168.75.133 -umysql-proxy -p123456 -P4040 -e
"create database lutixia charset utf8;"
```

在**4041管理端口**，再次查看：

```
mysql> select * from backends;
+-------------+---------------------+---------+------+----
--+-------------------+
| backend_ndx | address             | state   | type |
uuid | connected_clients |
+-------------+---------------------+---------+------+----
--+-------------------+
|           1 | 192.168.75.134:3306 | up      | rw   |
NULL |                 0 |
|           2 | 192.168.75.135:3306 | unknown | ro   |
NULL |                 0 |
+-------------+---------------------+---------+------+----
--+-------------------+
2 rows in set (0.00 sec)
```

2.7 通过代理查询数据：

```
# 先在从库选择lutixia数据库（因为主从关系，在主库创建的lutixia会同步至从库），创建表格，并插入数据：
MariaDB [(none)]> use lutixia
Database changed
MariaDB [lutixia]> create table t1( id int, name
varchar(20) );
Query OK, 0 rows affected (0.00 sec)

MariaDB [lutixia]> insert t1 values(1,"xiaoming");
Query OK, 1 row affected (0.00 sec)
```

通过**4040代理端口**查询数据，该sql语句会走slave，于是可以激活slave状态：

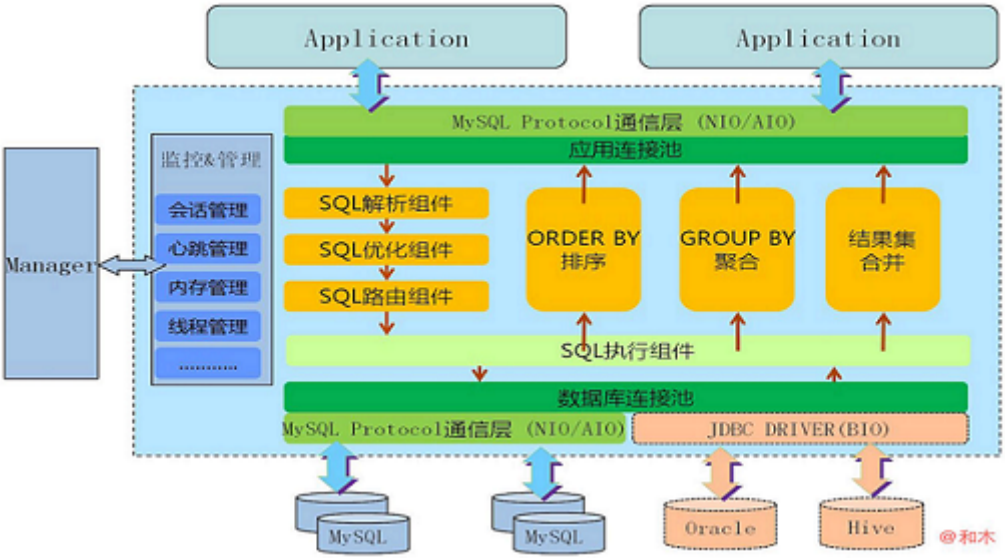
```
# 多执行几次!
[root@node4 ~]# mysql -h192.168.75.133 -umysql-proxy -
p123456 -P4040 -e "select * from lutixia.t1;"
+------+----------+
| id   | name     |
+------+----------+
|    1 | xiaoming |
+------+----------+
```

在**4041管理端口**，再次查看：

```
mysql> select * from backends;
+-------------+---------------------+-------+------+------
+------------------+
| backend_ndx | address             | state | type | uuid
| connected_clients |
+-------------+---------------------+-------+------+------
+------------------+
|           1 | 192.168.75.134:3306 | up    | rw   | NULL
|                 0 |
|           2 | 192.168.75.135:3306 | up    | ro   | NULL
|                 0 |
+-------------+---------------------+-------+------+------
+------------------+
2 rows in set (0.00 sec)
```

3. 基于Mycat实现读写分离：

Mycat基于阿里开源的Cobar产品而研发，一个彻底开源的，面向企业应用开发的大数据库集群，一个可以视为MySQL集群的企业级数据库，用来替代昂贵的Oracle集群， MYCAT并不依托于任何一个商业公司， 永不收费，永不闭源！




```
mycat:  192.168.75.133
master: 192.168.75.134
slave:  192.168.75.135
```

3.1 安装mycat:

```
# 下载mycat:
wget http://dl.mycat.io/1.6.7.1/Mycat-server-1.6.7.1-
release-20190627191042-linux.tar.gz
# 解压:
tar xf Mycat-server-1.6.7.1-release-20190627191042-
linux.tar.gz  -C /usr/local/
# 安装java-jdk:
yum install java-1.8.0-openjdk -y
# 配置mycat环境变量:
echo "export PATH=/usr/local/mycat/bin:$PATH" >
/etc/profile.d/mycat.sh
. /etc/profile.d/mycat.sh
```

3.2 授权mycat:

```
# 在主库执行授权信息，从库会自动同步:
grant all on *.* to "mycat-proxy"@"192.168.75.133"
identified by "123456";
```

3.3 配置mycat:

```
# 配置server.xml

# 默认管理用户，可读可写:
<user name="mycat" defaultAccount="true">
                <property
name="password">123456</property>
                <property
name="schemas">lutixiadb</property>
...
</user>
# 只读用户:
 <user name="user">
                <property name="password">user</property>
                <property
name="schemas">lutixiadb</property>
                <property name="readOnly">true</property>
        </user>

# 配置schema.xml
```

```
# 设置逻辑库以及数据库节点
<schema name="lutixiadb" checkSQLschema="false"
sqlMaxLimit="100" dataNode="dn1">
</schema>
# 配置数据库节点对应的后端真实的数据库：
<dataNode name="dn1" dataHost="localhost1"
database="students" />
# 配置读写库以及均衡：
<dataHost name="localhost1" maxCon="1000" minCon="10"
balance="1"
                          writeType="0" dbType="mysql"
dbDriver="native" switchType="1"  slaveThreshold="100">
                <heartbeat>select user()</heartbeat>
                <!-- can have multi write hosts -->
                <writeHost host="hostM1"
url="192.168.75.134:3306" user="mycat-proxy"
                                      password="123456">
                        <!-- can have multi read hosts -->
                        <readHost host="hostS1"
url="192.168.75.135:3306" user="mycat-proxy"
password="123456" />
                </writeHost>
</dataHost>
```

3.3 启动mycat:

```
mycat  start
```

3.4 连接测试：

```
mysql -umycat -p123456 -P8066 -h127.0.0.1
MySQL [(none)]> show databases;
+-----------+
| DATABASE  |
+-----------+
| lutixiadb |
+-----------+
1 row in set (0.00 sec)

# 可以在后端主库创建一个表，继续查询表测试：
```

```
MySQL [(none)]> use lutixiadb;
Reading table information for completion of table and
column names
You can turn off this feature to get a quicker startup
with -A

Database changed
MySQL [lutixiadb]> show tables;
+--------------------+
| Tables_in_students |
+--------------------+
| t1                 |
+--------------------+
1 row in set (0.01 sec)

# 在从库插入数据，继续查询：
MySQL [lutixiadb]> select * from t1;
+------+----------+
| id   | name     |
+------+----------+
|    1 | xiaowang |
+------+----------+
1 rows in set (0.00 sec)
```

在主库查不到数据，通过代理可以查到，即可验证读写分离成功。

3.5 报错解决：

```
MySQL [lutixiadb]> show tables;
ERROR 1184 (HY000): Invalid DataSource:0
```

可能是后端节点出现了问题，也有可能是代理端无法连上后端节点导致：
可以先在代理端直接用授权用户名和密码登录后端数据库测试连接问题：

```
[root@node3 conf]# mysql -umycat-proxy -h192.168.75.134 -p123456
ERROR 1129 (HY000): Host 'node3' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'
# 可以看到因为多次错误，代理端服务器被锁定了，所以也会出现上面的报错：
```

在后端主库执行如下命令：

```
mysqladmin flush-hosts
```

再次测试，一般问题就能解决。