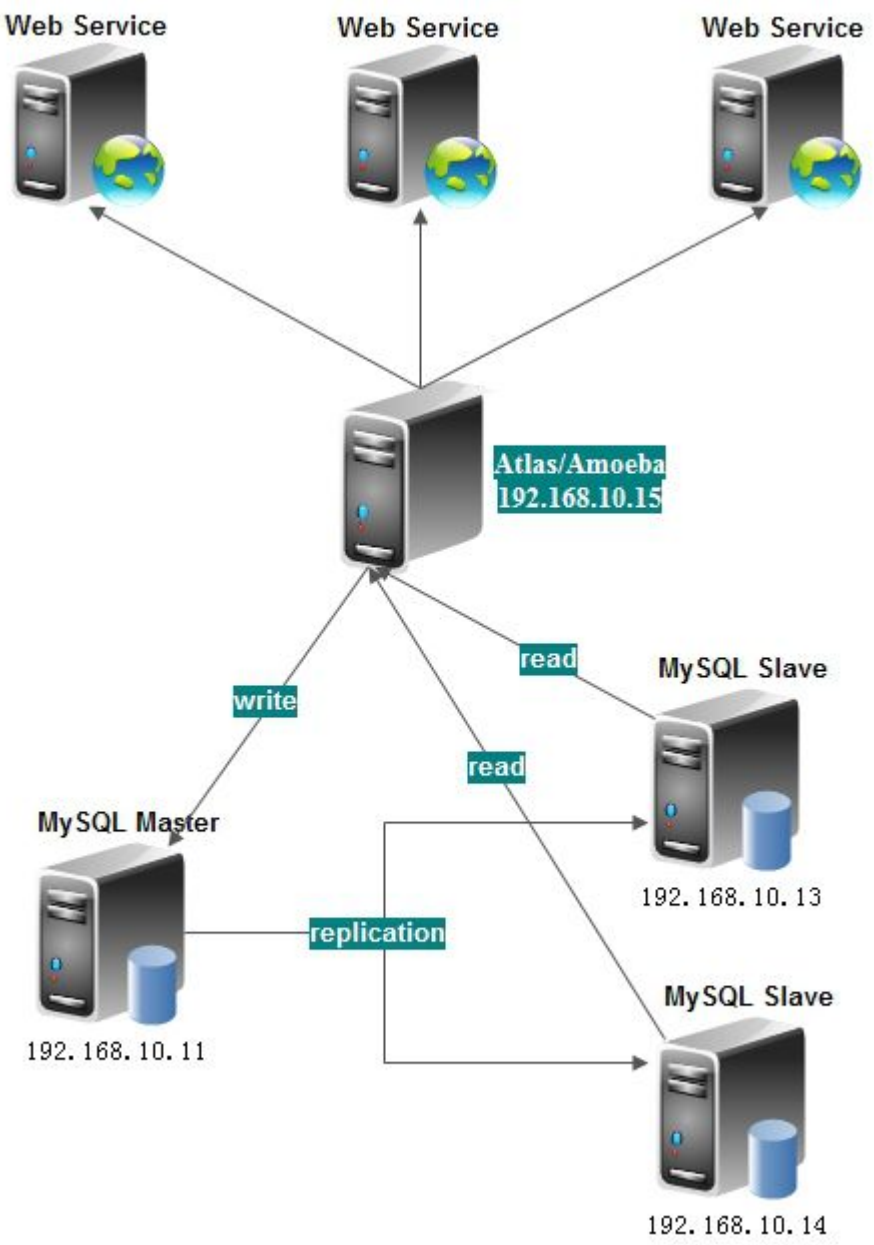
MySQL5.6 读写分离

1 MySQL5.6 读写分离

环境如下：
CentOS6.4_64
MySQL5.6
master：192.168.10.11
slave1：192.168.10.13
slave2：192.168.10.14
读写分离：192.168.10.15

原理：让主数据库处理增、改、删操作（INSERT、UPDATE、DELETE），而从数据库处理SELECT 查询操作，从而实现读写分离，如下图所示：



1.1 通过 Atlas 实现读写分离

Atlas 是一个位于应用程序与 MySQL 之间的中间件，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。

a. 下载并安装 Atlas 到 192.168.10.15（也可以安装在 master、slave）上，默认会安装在“/usr/local”目录下

```
[root@localhost src]# wget https://github.com/Qihoo360/Atlas/releases/download/2.0.5/Atlas-2.0.5.el6.x86_64.rpm
[root@localhost src]# rpm -ivh Atlas-2.0.5.el6.x86_64.rpm
```

b. 修改 Atlas 的配置文件

```
[root@localhost src]# cd /usr/local/mysql-proxy/conf
[root@localhost conf]# vim test.cnf
[mysql-proxy]

#带#号的为非必需的配置项目

#管理接口的用户名
admin-username = admin

#管理接口的密码
admin-password = admin

#实现管理接口的 Lua 脚本所在路径
admin-lua-script = /usr/local/mysql-proxy/lib/mysql-proxy/lua/admin.lua

#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 = 192.168.10.11:3306

#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，若省略则默认为 1，可设置多项，用逗号分隔
proxy-read-only-backend-addresses = 192.168.10.13:3306@1,192.168.10.14:3306@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
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，一般开发调试时设为 false，线上运行时设为 true
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，
```

```
monitor 在 worker 意外退出后会自动将其重启，设为 false 时只有 worker，没有 monitor，一般开发调试时设为 false，线上运行时设为 true
keepalive = true

#工作线程数，推荐设置与系统的 CPU 核数相等
event-threads = 4

#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

#SQL 日志的开关，可设置为 OFF、ON、REALTIME，OFF 代表不记录 SQL 日志，ON 代表记录 SQL 日志，REALTIME 代表记录 SQL 日志且实时写入磁盘，默认为 OFF
#sql-log = OFF

#实例名称，用于同一台机器上多个 Atlas 实例间的区分
instance = test

#Atlas 监听的工作接口 IP 和端口
proxy-address = 0.0.0.0:1234

#Atlas 监听的管理接口 IP 和端口
admin-address = 0.0.0.0:2345

#分表设置，此例中 person 为库名，mt 为表名，id 为分表字段，3 为子表数量，可设置多项，以逗号分隔，若不分表则不需要设置该项
#tables = person.mt.id.3

#默认字符集，若不设置该项，则默认字符集为 latin1
charset = utf8

#允许连接 Atlas 的客户端的 IP，可以是精确 IP，也可以是 IP 段，以逗号分隔，若不设置该项则允许所有 IP 连接，否则只允许列表中的 IP 连接
#client-ips = 127.0.0.1, 192.168.1

#Atlas 前面挂接的 LVS 的物理网卡的 IP(注意不是虚 IP)，若有 LVS 且设置了 client-ips 则此项必须设置，否则可以不设置
#lvs-ips = 192.168.1.1
```

其中，“pwds = root:/iZxz+0GRoA=”为数据库的账号及密码，账号为：root，密码为：123456

```
[root@localhost bin]# pwd
/usr/local/mysql-proxy/bin
```

```
[root@localhost bin]# ./encrypt 123456
/iZxz+0GRoA=
```

c. 启动 Atlas

```
[root@localhost bin]# ./mysql-proxyd test start
```

d. 管理平台的使用

```
[root@localhost bin]# mysql -h 192.168.10.15 -P2345 -uadmin -padmin

##通过 select * from help，可以查看如何使用管理平台
mysql> select * from help;


##可以看到 192.168.10.11 具有读写，其他的只能读
mysql> SELECT * FROM backends;
+-------------+--------------------+-------+------+
| backend_ndx | address            | state | type |
+-------------+--------------------+-------+------+

|           1 | 192.168.10.11:3306 | up    | rw   |
|           2 | 192.168.10.13:3306 | up    | ro   |
|           3 | 192.168.10.14:3306 | up    | ro   |
+-------------+--------------------+-------+------+
3 rows in set (0.00 sec)


##动态的移除一台 slave
mysql> REMOVE BACKEND 3;
Empty set (0.00 sec)

mysql> SELECT * FROM backends;
+-------------+--------------------+-------+------+
| backend_ndx | address            | state | type |
+-------------+--------------------+-------+------+
|           1 | 192.168.10.11:3306 | up    | rw   |
|           2 | 192.168.10.13:3306 | up    | ro   |
+-------------+--------------------+-------+------+
2 rows in set (0.00 sec)

##动态的添加一台 slave
mysql> add slave 192.168.10.14:3306;
Empty set (0.00 sec)

mysql> SELECT * FROM backends;
```

```
+-------------+-------------------+-------+------+
| backend_ndx | address           | state | type |
+-------------+-------------------+-------+------+
|           1 | 192.168.10.11:3306 | up    | rw   |
|           2 | 192.168.10.13:3306 | up    | ro   |
|           3 | 192.168.10.14:3306 | up    | ro   |
+-------------+-------------------+-------+------+
3 rows in set (0.00 sec)
```

e. 测试 Atlas 的读写分离

```
##需要打开 SQL 日志的开关，并设置级别为“REALTIME”
[root@localhost bin]# vim ../conf/test.cnf
sql-log = REALTIME
[root@localhost bin]# ./mysql-proxyd test restart

[root@localhost bin]# mysql -h 192.168.10.15 -P1234 -uroot -p123456
mysql> use baba1;
Database changed
mysql> select * from test;
mysql> select * from test;
mysql> insert into test value(1);
mysql> insert into test value(2);
mysql> insert into test value(3);
mysql> select * from test;
mysql> select * from test;

##SQL 日志文件
[root@localhost log]# tail -f sql_test.log
[02/09/2014 23:23:33] C:192.168.10.15 S:192.168.10.13 OK 1.598 "SELECT DATABASE()"
[02/09/2014 23:23:50] C:192.168.10.15 S:192.168.10.13 OK 1.433 "select * from test"
[02/09/2014 23:23:53] C:192.168.10.15 S:192.168.10.14 OK 1.495 "select * from test"
[02/09/2014 23:23:59] C:192.168.10.15 S:192.168.10.11 OK 4.594 "insert into test value(1)"
[02/09/2014 23:24:01] C:192.168.10.15 S:192.168.10.11 OK 4.922 "insert into test value(2)"
[02/09/2014 23:24:05] C:192.168.10.15 S:192.168.10.11 OK 2.815 "insert into test value(3)"
[02/09/2014 23:24:07] C:192.168.10.15 S:192.168.10.13 OK 1.348 "select * from test"
[02/09/2014 23:24:07] C:192.168.10.15 S:192.168.10.14 OK 1.485 "select * from test"
```

备注：如果 slave 都挂掉了，则读写都会在 master 上执行。

f. 官方资源

文档：https://github.com/Qihoo360/Atlas

下载：https://github.com/Qihoo360/Atlas/releases

1.2 通过 Amoeba 实现读写分离

Amoeba(变形虫)项目,该开源框架于 2008 年开始发布一款 Amoeba for Mysql 软件。这个软件致力于 MySQL 的分布式数据库前端代理层，它主要在应用层访问 MySQL 的 时候充当 SQL 路由功能，专注于分布式数据库代理层(Database Proxy）开发。座落与 Client、DB Server(s)之间,对客户端透明。具有负载均衡、高可用性、SQL 过滤、读写分离、可路由相关的到目标数据库、可并发请求多台数据库合并结果。 通过 Amoeba 你能够完成多数据源的高可用、负载均衡、数据切片的功能，目前 Amoeba 已在很多企业的生产线上面使用。

a. Amoeba 框架是基于 Java 开发的，所以需要安装 JDK

```
[root@localhost local]# pwd
/usr/local
[root@localhost local]# ./jdk-6u35-linux-x64.bin
#添加下面内容到/etc/profile 中
export JAVA_HOME=/usr/local/jdk1.6.0_35
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH

[root@localhost local]# source /etc/profile
```

b. 下载 Amoeba 并解压到 192.168.10.15（也可以安装在 master、slave 里）的“/usr/local/amoeba”目录里

```
[root@localhost amoeba]# pwd
/usr/local/amoeba
[root@localhost amoeba]# wget
http://nchc.dl.sourceforge.net/project/amoeba/Amoeba%20for%20mysql/2.2.x/amoeba-mysql-bin
ary-2.2.0.tar.gz
[root@localhost amoeba]# tar xf amoeba-mysql-binary-2.2.0.tar.gz
```

c. 修改配置 dbServers.xml（数据库池的定义与配置）

```
<amoeba:dbServers xmlns:amoeba="http://amoeba.meidusa.com/">
    <dbServer name="abstractServer" abstractive="true">
        <factoryConfig
class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">
            <!-- MySQL 端口号 -->
            <property name="port">3306</property>
            <!-- MySQL 数据库 -->
            <property name="schema">baba1</property>
            <!-- MySQL 账号 -->
            <property name="user">root</property>
            <!--  MySQL 密码 -->
            <property name="password">123456</property>
```

```
        </factoryConfig>
    </dbServer>
    <dbServer name="master"    parent="abstractServer">
        <factoryConfig>
            <property name="ipAddress">192.168.10.11</property>
        </factoryConfig>
    </dbServer>
    <dbServer name="slave1"    parent="abstractServer">
        <factoryConfig>
            <property name="ipAddress">192.168.10.12</property>
        </factoryConfig>
    </dbServer>
    <dbServer name="slave2"    parent="abstractServer">
        <factoryConfig>
            <property name="ipAddress">192.168.10.13</property>
        </factoryConfig>
    </dbServer>
    <!--   多 slave 的配置方式，名称为：multiPool-->
    <dbServer name="multiPool" virtual="true">
        <poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">
            <!-- Load balancing strategy: 1=ROUNDROBIN , 2=WEIGHTBASED , 3=HA-->
            <property name="loadbalance">1</property>
            <!-- Separated by commas,such as:slave1,slave1,slave2 -->
            <property name="poolNames">slave1,slave2</property>
        </poolConfig>
    </dbServer>
</amoeba:dbServers>
```

d. 修改配置 amoeba.xml

```
<amoeba:configuration xmlns:amoeba="http://amoeba.meidusa.com/">
    <proxy>
        <service name="Amoeba for Mysql"
class="com.meidusa.amoeba.net.ServerableConnectionManager">
            <property name="portipAddress">8066</property>
            <property name="">192.168.10.15</property>
            <property name="authenticator">
                <bean class="com.meidusa.amoeba.mysql.server.MysqlClientAuthenticator">
                    <!--   Amoeba 的账号    -->
                    <property name="user">root</property>
                    <!--   Amoeba 的密码    -->
                    <property name="password">12345678</property>
                </bean>
            </property>
```

```
            </service>
        </proxy>
        <queryRouter class="com.meidusa.amoeba.mysql.parser.MysqlQueryRouter">
            <!--    定义了 Amoeba 缓存的 SQL 语句解析的条数    -->
            <property name="LRUMapSize">1500</property>
            <!--    defaultPool 配置了默认的数据库节点，
            除了 SELECT\UPDATE\INSERT\DELETE 的语句都会在 defaultPool 执行      -->
            <property name="defaultPool">master</property>
            <!--    writePool 配置了数据库的写库    -->
            <property name="writePool">master</property>
            <!--    readPool 配置了数据库读库，通常配为 Slave 或者 Slave 组成的数据库池，
            这里就配置了 multiPool 数据库池    -->
            <property name="readPool">multiPool</property>
            <property name="needParse">true</property>
        </queryRouter>
</amoeba:configuration>
```

e. 启动 amoeba

```
[root@localhost bin]# pwd
/usr/local/amoeba/bin
[root@localhost bin]# ./amoeba start

The stack size specified is too small, Specify at least 160k
Could not create the Java virtual machine.

##启动的时候，如果提示上面的错误，则修改 amoeba 文件，把-Xss 的值改成 256k
[root@localhost bin]# vim amoeba
DEFAULT_OPTS="-server -Xms256m -Xmx256m -Xss256k"
[root@localhost bin]# ./amoeba start
```

f. 测试 amoeba

```
[root@localhost conf]# mysql -uroot -p12345678 -h192.168.10.15 -P8066

测试方法：
1、执行 insert select 是否正常 (正常)
2、关闭 slave1 slave2，是否可以执行 select (不可以)
3、启动 slave1 slave2，关闭 master，是否可以执行 insert (不可以)
```

g. 官方资源

文档：http://docs.hexnova.com/amoeba/

下载：http://sourceforge.net/projects/amoeba/files/

备注：经测试，amoeba2.2.x 支持事务