mysql高可用架构之MHA，haproxy实现读写分离详解

MySQL高可用架构之MHA

一、运维人员需要掌握的MySQL技术：

1、基本SQL语句

2、基本的管理【库表数据的管理　权限的管理】

3、容灾　　保证数据不丢失。

二、工作中MySQL的问题：

AB不同步怎么办?

1）如果当天及时发现的：先基于当前同步，再做差异还原

2）好几天才发现：重做B

三、数据完整性如何实现?

1）AB同步

AB互为主备+keepalived

php连接VIP，如果master宕机，slave承担访问流量；但如果master回来了，来承当访问流量并且造成数据不一致的情况（MHA解决这种问题，当A回来了进行差异还原使数据一致）。

2）MySQL注入

MySQL注入：登上你MySQL，删掉数据（在数据库层），然后所有的服务器就没数据了

备份策略：每周全备一次，每天增量备份

3）数据量的增长

IO成为瓶颈—分析业务（1、读多写少；2、读多写多；3、读少写多）

以读多写少的情况为例：

读写分离　A负责写　　B负责读

因为读的要求较多，一台B效率低，所以架构演变成ABBB（一主多从），并把B做成集群，以提高读效率。

那么，当A宕机了，由哪个B来负责写入呢?

通过选举机制，从B中选出一个来当A，然后所有的B给新A做备份，并保证数据的完整性!

******

四、MHA实现机制:

监控AB的状态

完整的选举机制（看谁的数据跟master最接近）

让一个B切换到新A

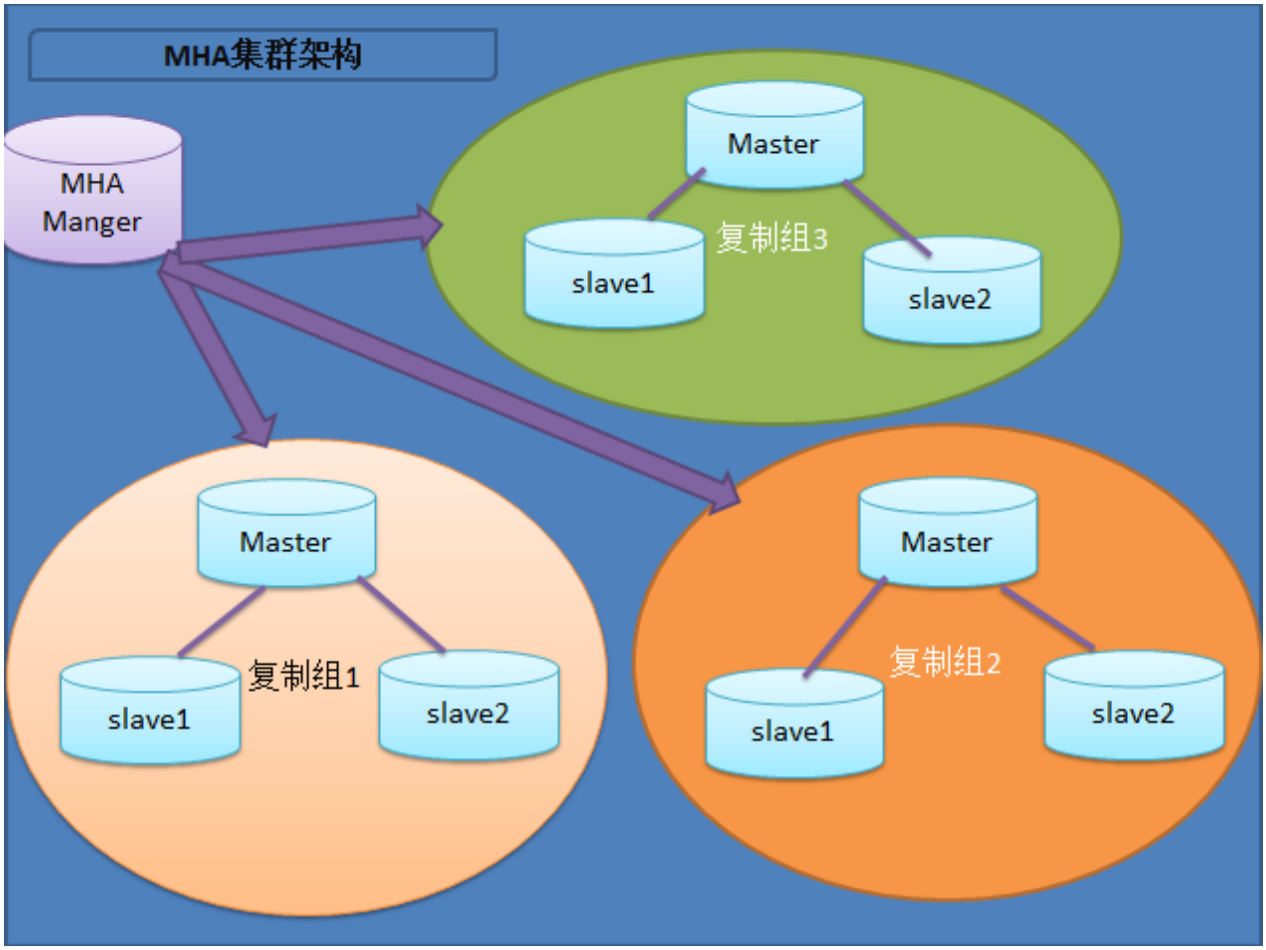
保证数据的完整性（通过差异还原）

MHA（Master High Availability）目前在MySQL高可用方面是一个相对成熟的解决方案，它由日本DeNA公司youshimaton（现就职于Facebook公司）开发，是一套优秀的作为MySQL高可用性环境下故障切换和主从提升的高可用软件。在MySQL故障切换过程中，MHA能做到在0~30秒之内自动完成数据库的故障切换操作，并且在进行故障切换的过程中，MHA能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

该软件由两部分组成：MHA Manager（管理节点）和MHA Node（数据节点）。MHA Manager可以单独部署在一台独立的机器上管理多个master-slave集群，也可以部署在一台slave节点上。MHA Node运行在每台MySQL服务器上，MHA Manager会定时探测集群中的master节点，当master出现故障时，它可以自动将最新数据的slave提升为新的master，然后将所有其他的slave重新指向新的master。整个故障转移过程对应用程序完全透明。

在MHA自动故障切换过程中，MHA试图从宕机的主服务器上保存二进制日志，最大程度的保证数据的不丢失，但这并不总是可行的。例如，如果主服务器硬件故障或无法通过ssh访问，MHA没法保存二进制日志，只进行故障转移而丢失了最新的数据。使用MySQL 5.5的半同步复制，可以大大降低数据丢失的风险。MHA可以与半同步复制结合起来。如果只有一个slave已经收到了最新的二进制日志，MHA可以将最新的二进制日志应用于其他所有的slave服务器上，因此可以保证所有节点的数据一致性。

目前MHA主要支持一主多从的架构，要搭建MHA,要求一个复制集群中必须最少有三台数据库服务器，一主二从，即一台充当master，一台充当备用master，另外一台充当从库，因为至少需要三台服务器，出于机器成本的考虑，淘宝也在该基础上进行了改造，目前淘宝TMHA已经支持一主一从。



假如MySQL MHA架构运行正常，突然在上午10：00一个复制组中的A的MySQL服务 down了！

因为AB复制是异步复制，所以可能有一些数据尚没有被B拉到其relay_log中,即AB数据不一致，MHA是怎样解决这种情况的呢?

1、mha_manager 使用scp命令将A当前binlog拷贝到mha_manager

2、待新A（选举：依据谁的relay_log新）产生后，mha_manager再拿着旧A的binlog和新的relay_log做比对，并进行差异还原以保证新A和旧A数据的一致性

3、mha_manager最后拿着老A的binlog去找复制组中其他B 做差异还原，保证数据的一致性

实验步骤：

1） ssh证书户信任（ssh的密钥验证）

2） ABB架构（）

3） 安装mha_manager 、 mha_node

4） 测试

IP地址规划

Mysql_manager(18.240)一需要是64位的系统

MySQL_A(18.241)

MySQL_B1(18.242)

MySQL_B2(18.243)

1、ssh证书互信脚本---每台机器上都要操作

1.1使用ssh-kegen生成公私钥（每台服务器上）

1.2执行脚本

```
#!/bin/bash
for i in 0 1 2 3
        do
                ssh-copy-id -i /root/.ssh/id_rsa.pub 192.168.18.24$i
        done
```

2、ABB搭建

2.1测试yum仓库的可用性脚本

```
#!/bin/bash
for i in 0 1 2 3
        do
                ssh 192.168.18.24$i     "
                if ( yum install telnet-server -y ) > /dev/null 2>&1;then
                                echo "192.168.18.24$i Yum is TURE"
                        else
                                echo "192.168.18.24$i Yum is FALSE "
                fi
                "
        done
```

2.2安装脚本：

```
#!/bin/bash
```

```
for i in 0 1 2 3
    do
        ssh 192.168.18.24$i     "
        if ( yum install mysql mysql-server perl-* -y ) > /dev/null 2>&1;then
                echo "192.168.18.24$i installation finish"
            else
                echo "192.168.18.24$i installation error "
        fi
        "
    done
```

2.3所有的机器初始化MySQL，设置ABB架构：

MYSQL_A(241)上的操作:

vim /etc/my.cnf

server_id=1　　//设置优先级最高

log_bin=binlog　　//开启binlog日志

log_biin=binlog.index

MYSQL_B1、MYSQL_B2上的操作同mysql_a只是注意server_id的不同!

2.4 设置用户

在msyql_a(241)中：因为前面已经开启二进制日志了，所以其他机器也能学到赋权语句！！故，其他机器就不用再赋权了!

> grant all on *.* to “root”@“%” identified by “123”;　　//给mha_manager用，因为其在failover时需要登陆上来，并且拷贝binlog到所有的slave上去。

> grant replication slave on *.* to “sky”@“%”identified by “123”;　　//复制专用用户

> flush privileges;　　//刷新权限

> show master status\G　　//看一下binlog日志到第几个文件了

在mysql_b1(242)中:

- slave stop;
- change master to master_host="192.168.18.241",master_user="sky",master_passoword="123",master_log_file="binlog.000001";
- slave start;
- show slave status\G　　//查看复制线程状态

在mysql_b2（243）做242上同样的操作!

3、安装配置启动mha_manager、mha_node

3.1安装

manager机器上:

cd /usr/src/mha_soft

[root@localhost mha_soft]# rpm -ivh mha4mysql-node-0.54-0.el6.noarch.rpm　　//安装node

cd dependent

yum -y install localinstall ./*　　//装上依赖

cd ..

rpm -ivh mha4mysql-manager-0.55-0.el6.noarch.rpm　　//安装manager

mha_manager的配置文件:

[root@localhost mha]# cp -r mha /etc/

mha目录下主要有以下两个文件

mha.cnf　mha_start

[root@localhost mha]# vim /etc/mha/mha.cnf　　//mha配置文件

```
[server default]
#mysql admin account and password

user=root
password=123

#mha workdir  and worklog
```

```
manager_workdir=/etc/mha
manager_log=/etc/mha/manager.log

#mysql A/B account and pw

repl_user=sky
repl_password=123

#check_mha_node time
ping_interval=1

#ssh account
ssh_user=root


[server1]
hostname=192.168.18.241
ssh_port=22
master_binlog_dir=/var/lib/mysql
candidate_master=1


[server2]
hostname=192.168.18.242
ssh_port=22
master_binlog_dir=/var/lib/mysql
candidate_master=1


[server3]
hostname=192.168.18.243
ssh_port=22
master_binlog_dir=/var/lib/mysql
candidate_master=1
```

vim /etc/mha/mha_start　　//启动脚本

```
nohup masterha_manager --conf=/etc/mha/mha.cnf > /tmp/mha_manager.log </dev/null 2>&1 &
```

其他节点：

安装mha4mysql-node…rpm包（直接rpm安装即可）

3.2 检测ssh互信有没有问题

[root@localhost src]# masterha_check_ssh --conf=/etc/mha/mha.cnf

…

Tue Jun 13 02:21:31 2017 - [info] All SSH connection tests passed successfully.

3.3 测AB

[root@localhost src]# masterha_check_repl --conf=/etc/mha/mha.cnf

…

MySQL Replication Health is OK.

3.4 启动

[root@localhost src]# cat /etc/mha/mha_start　//先看一下启动方式

nohup masterha_manager --conf=/etc/mha/mha.cnf > /tmp/mha_manager.log </dev/null 2>&1 &

运行:

[root@localhost src]# nohup masterha_manager --conf=/etc/mha/mha.cnf > /tmp/mha_manager.log </dev/null 2>&1 &

3.5 测试

现在两个slave的Master_Host同为241，把241干掉后，就会选举新的master了!

先在**mha_manager**上打开日志:

tailf /etc/mha/manager.log

到241上关闭MySQL服务:

service mysqld stop

查看**mha_manager**日志输出:

…　　//此处省略

----- Failover Report -----

mha: MySQL Master failover 192.168.18.241 to 192.168.18.242 succeeded

Master 192.168.18.241 is down!

Check MHA Manager logs at localhost.localdomain:/etc/mha/manager.log for details.

Started automated(non-interactive) failover.

The latest slave 192.168.18.242(192.168.18.242:3306) has all relay logs for recovery.

Selected 192.168.18.242 as a new master.

192.168.18.242: OK: Applying all logs succeeded.

192.168.18.243: This host has the latest relay log events.

Generating relay diff files from the latest slave succeeded.

192.168.18.243: OK: Applying all logs succeeded. Slave started, replicating from 192.168.18.242.

192.168.18.242: Resetting slave info succeeded.

Master failover to 192.168.18.242(192.168.18.242:3306) completed successfully.

看来242变成了master！～

可以去新主上创建个库或到243上查看一下master.info来验证！！

老master恢复后如果想要它再做master，要先将新master的数据同步，之后删除下面两个文件，再重新开启MHA功能，令新master宕掉即可（实验可行，实际操作不推荐）如下：

注意：mha_manager每执行一次failover后，该进程自动退出。如果还想测试failover需要重新开启---开启前要将下面两个文件删掉：

[root@MHA_243 mha]# cd /etc/mha/

[root@MHA_243 mha]# rm -fr mha.failover.complete saved_master_binlog_from_192.168.19.241_3306_20170817154913.binlog

下面演示old_master回来，如何保证old_master同步new_master的新产生的数据：

当old_master服务宕掉后，去mha_monitor上执行：

[root@mha_master mha]# grep -i change /etc/mha/manager.log (-i 是不区分大小写)

Tue Jun 13 02:30:23 2017 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER_HOST='192.168.18.242', MASTER_PORT=3306, MASTER_LOG_FILE='binlog.000001', MASTER_LOG_POS=106, MASTER_USER='sky', MASTER_PASSWORD='xxx';

然后在old_master上执行：

mysql> slave stop;

mysql> change master to master_host='192.168.18.242', master_port=3306, master_log_file='binlog.000001', master_log_pos=106, master_user='sky', master_password='123';(只需要修改密码即可)

mysql> slave start;

4、MHA failover(master故障)后VIP漂移

MHA架构中，master来承担写请求，但是如果发生了failover，那么就应该让new_master来承担写请求，有哪些方式可以实现呢?

1、 改变master的IP：在web上修改PHP页面的代码（所有写请求修改成new_master的IP）

2、 使用虚拟IP（VIP），将VIP漂移给new_master

显然，第二种方案要更加容易实现、高效。

实现起来，大家可能会首当其冲的想到keepalived，但是在这里不适用，因为我们不好判断哪一个salve会在failover后变成master（在keepalived中，VIP根据物理路由器的优先级来确定，万一漂到一台slave上那可如何是好！）。不过我们可以通过脚本的方式来实现将VIP绑定到new_master上。

脚本思路如下：

脚本（/etc/mha/check_mysql）运行在manager上，它来管理VIP

判断谁是主，确保它有VIP，并继续判断，如果slave有VIP，则收回。

脚本名称：master_vip_drift.sh

脚本需要根据前面设置的数据库密码对应修改，此处设置的为用户root，密码123

```
#!/bin/bash
VIP=192.168.18.245
NETMASK=255.255.255.0
MUSER=root
MPW=123
MYSQL_IP="192.168.18.241 192.168.18.242 192.168.18.243"
NIC=eth0
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~main program~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#check_master_mysql
check_master() {          #检测谁是master
for IP in $MYSQL_IP
do
      if ssh $IP "lsof -i :3306" &>/dev/null;then
      ssh $IP "mysql -uroot -p123 -e 'show slave status \G'|grep -w 'Slave_IO_Running'"
&>/dev/null
       if [ $? -eq 1 ];then
        MY_master=$IP
    echo "$MY_master"
       fi
     fi
done
}

check_master_alive() {    #检测master的存活状态
for IP in $MYSQL_ip
do
      if ssh $IP "ip add show eth0"|grep inet|grep "$VIP" &>/dev/null;then
    ssh $IP "lsof -i:3306" &>/dev/null
      if [ $? -ne 0 ];then
    ssh $IP "ifconfig $NIC:245 down "
      fi
      fi
done
}


VIP () {       #给master赋予VIP，不是master收回VIP
for IP in $MYSQL_IP
     do
          ssh $IP "ip add show eth0"|grep inet|grep "$VIP" &>/dev/null
            if [ $? -eq 0 ] && [ $MY_master != "$IP" ];then       #如果有VIP但是不是master，收
回VIP
                ssh $IP "ifconfig $NIC:245 down"
            elif [ $? -eq 1 ] && [ $MY_master == "$IP" ];then      #如果没有VIP却是master，赋予
其VIP
                ssh $IP "ifconfig $NIC:245 $VIP netmask $NETMASK up"
            fi
done
}
```

```
while true
   do
      check_master
      check_master_alive
      VIP
      sleep 1
done
```

当前实验环境：

241 mysql_a

243 mysql_b2

242 mysql_b1 新 master

245 VIP

240 mha_monitor

1. 重新运行masterha_manager:

[root@mha_master mha]# nohup masterha_manager --conf=/etc/mha/mha.cnf > /tmp/mha_manager.log </dev/null 2>&1 &

2. 重新运行脚本master_vip_drift.sh

[root@mha_master mha]# ./master_vip_drift.sh &

3. 去242上检查VIP有没有绑定上

```
[root@mysql_b1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:55:C2:6D
          inet addr:192.168.18.242  Bcast:192.168.18.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe55:c26d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:47612 errors:46292 dropped:0 overruns:0 frame:0
          TX packets:44817 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9006078 (8.5 MiB)  TX bytes:9206565 (8.7 MiB)
          Interrupt:19 Base address:0x2024

eth0:245  Link encap:Ethernet  HWaddr 00:0C:29:55:C2:6D
          inet addr:192.168.18.245  Bcast:192.168.18.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:19 Base address:0x2024
```

4. 模拟242的MySQL服务宕掉

```
[root@mysql_b1 ~]# /etc/init.d/mysqld stop
停止 mysqld:                                               [确定]
[root@mysql_b1 ~]#
```

5. 看看monitor上的日志，找出谁是新master

```
root@mha_master opt]# tailf /etc/mha/manager.log

Started automated(non-interactive) failover.
The latest slave 192.168.18.241(192.168.18.241:3306) has all relay logs fo
Selected 192.168.18.241 as a new master.
```

6、验证241上有没有VIP，242上的VIP有没有被收回

```
[root@mysql_a ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
    link/ether 00:0c:29:67:4e:93 brd ff:ff:ff:ff:ff:ff
    inet 192.168.18.241/24 brd 192.168.18.255 scope global eth0
    inet 192.168.18.245/24 brd 192.168.18.255 scope global secondary eth0:245
    inet6 fe80::20c:29ff:fe67:4e93/64 scope link
       valid_lft forever preferred_lft forever
```

***monitor进行一次failover后自动关闭，还需手动运行

***需手动删除/etc/mha/下的失败切换文件failover file和binlog，否则failover不成功!

5、分析failover日志一了解MHA工作流程

5.1 tailf /etc/mha/manager.log

5.2 到master上关闭MySQL服务

5.3 查看日志并从头往下捋一遍

6、MySQL MHA读写分离后将slave做成集群

用的是HAProxy这款支持七层和四层分发的将slave做成集群，来承担读请求

实现读写分离：

通过修改PHP页面中绑定的IP地址，master承担**写**请求用**VIP**，slave集群承担**读**请求用**haproxyIP**。

实验环境：

241 mysql_a //master

242 mysql_b1

243 mysql_b2

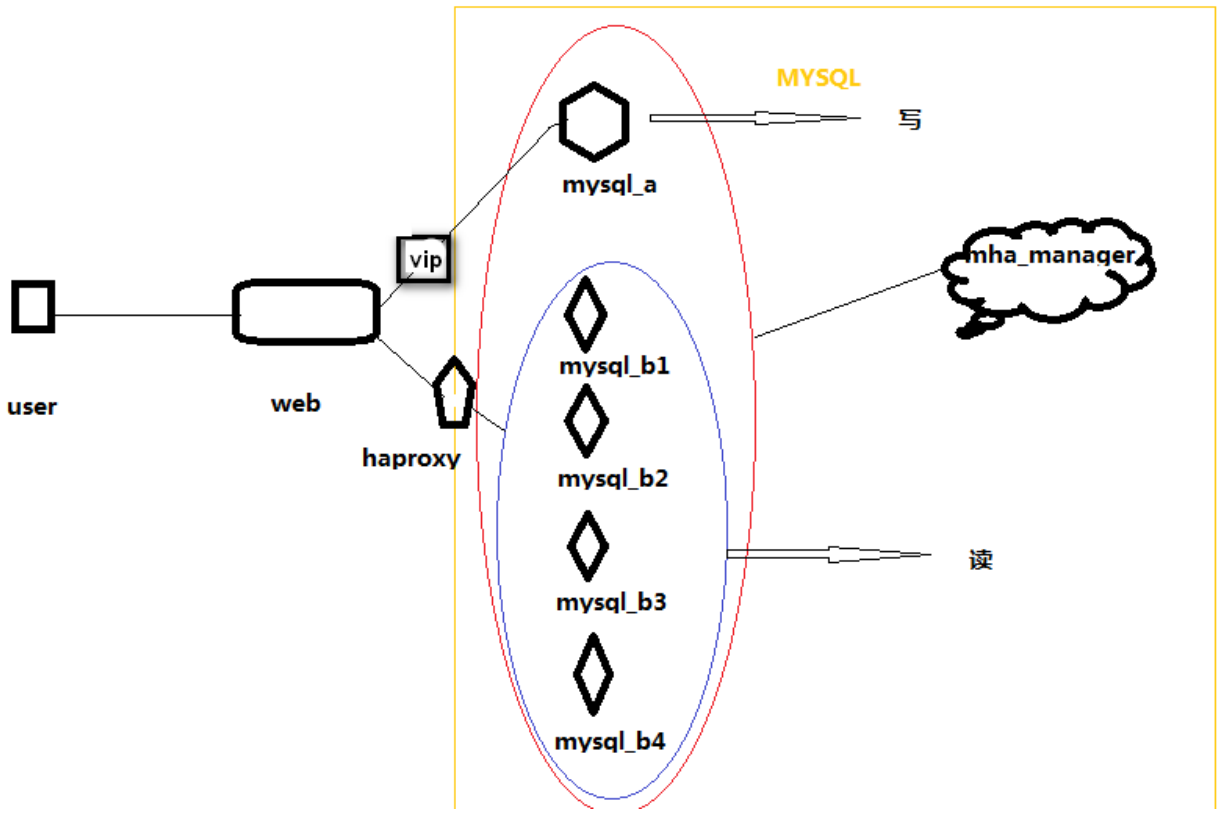
245 VIP(master)

246 mha_monitor

247 HAProxy

248 client/httpd（lamp）

大致结构图如下：



结构图说明：

a.MHA对abb架构实行监控，发现a宕之后选出新a（实现VIP漂移，仅a有VIP）

b.haproxy对slave集群实现负载均衡，承担读请求

c.web集群请求后台数据，写请求传给a(VIP)，读请求传给haproxy，haproxy再对读请求实现分发达到负载均衡

6.1 安装HAProxy

yum install gcc -y

tar xf haproxy-1.5.3.tar

make TARGET=linux26 PREFIX=/usr/local/haproxy

make install PREFIX=/usr/local/haproxy

cp /usr/src/haproxy/haproxy-1.5.3/examples/haproxy.cfg /usr/local/haproxy/

cp /usr/src/haproxy/haproxy-1.5.3/examples/haproxy.init /etc/init.d/haproxy

chmod 755 /etc/init.d/haproxy

ln -s /usr/local/haproxy/sbin/* /usr/sbin/

mkdir /etc/haproxy

mkdir /usr/share/haproxy

ln -s /usr/local/haproxy/haproxy.cfg /etc/haproxy/

cd ..

6.2 设置配置文件

6.2.1 拷贝配置文件

[root@HAProxy_247 haproxy]# cp haproxy.cfg /usr/local/haproxy/

cp: overwrite `/usr/local/haproxy/haproxy.cfg'? y

6.2.2 编辑配置文件

[root@HAProxy_247 haproxy]# vim /usr/local/haproxy/haproxy.cfg

注意修改RS的IP：

```
# this config needs haproxy-1.1.28 or haproxy-1.2.1

global
    log 127.0.0.1    local0
    log 127.0.0.1    local1 notice
    #log loghost    local0 info
    maxconn 4096
    chroot /usr/share/haproxy
    uid 99
    gid 99
    daemon
    #debug
    #quiet

defaults
    log    global
    mode    http
    #option    httplog
```

```
    option    dontlognull
    retries    3
    option redispatch
    maxconn    2000
    contimeout    5000
    clitimeout    50000
    srvtimeout    50000


listen    MySQL 0.0.0.0:3306
    mode tcp
    maxconn    200
    balance roundrobin
    option mysql-check user root
    server mysql_1 192.168.18.242:3306  inter 1s rise 2 fall 2
    server mysql_2 192.168.18.243:3306  inter 1s rise 2 fall 2


listen  admin_status
        mode  http
        bind 0.0.0.0:8899
        option httplog
        log global
        stats enable
        stats refresh 10s
        stats hide-version
        stats realm Haproxy\ Statistics
        stats uri  /admin-status
        stats auth  admin:123456
        stats admin if TRUE
```

6.3 启动HAProxy服务

[root@HAProxy_247 haproxy]# service haproxy start

```
[root@HAProxy_247 haproxy]# lsof -i:3306
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
haproxy 3953 nobody    4u  IPv4  52319      0t0  TCP *:mysql (LISTEN)
```

6.4 访问HAProxy的网页

http://localhost:8899/admin-status

用户名密码查看配置文件舍子

6.5 客户端测试---密码123

```
[root@client_248 桌面]# mysql -uroot -p123 -h 192.168.18.247
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1431
Server version: 5.1.73-log Source distribution

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| DB1                |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.01 sec)
```

6.6 回到HAProxy管理界面，可以看到那台slave响应的请求

MySQL

		Queue			Session rate			Sessions						Bytes		Denied		
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req
	Frontend				0	1	-	1	1	200	3			160	427	0	0	0
☐	mysql_1	0	0	-	0	1		1	1	-	2	2	5s	42	140		0	
☐	mysql_2	0	0	-	0	1		0	1	-	1	1	4m29s	118	287		0	
	Backend	0	0		0	1		1	1	20	3	3	5s	160	427	0	0	