**MySql主备+灾备（Keepalived）**

**people.net: sotp**

**目 录**

[1 集群网络架构 3](#_Toc464757369)

[2 使用yum安装mysql 4](#_Toc464757370)

[2.1 安装mysql依赖 4](#_Toc464757371)

[2.2 编辑yum源 4](#_Toc464757372)

[2.3 yum安装mysql 4](#_Toc464757373)

[2.4 修改root 用户口令 4](#_Toc464757374)

[2.5 设置开机启动 5](#_Toc464757375)

[2.6 安装mysql周边产品 5](#_Toc464757376)

[2.7 登录mysql 5](#_Toc464757377)

[2.8 关闭防火墙 5](#_Toc464757378)

[3 数据库参数配置 6](#_Toc464757379)

[3.1 设置Mysql远程访问 8](#_Toc464757380)

[4 mysql双主互为备份设置 8](#_Toc464757381)

[4.1 设置auto\_increment、server\_id等 8](#_Toc464757382)

[4.2 重启mysql 9](#_Toc464757383)

[4.3 A主B从设置 9](#_Toc464757384)

[4.3.1 锁定主数据库DDL操作 9](#_Toc464757385)

[4.3.2 停止从服务 9](#_Toc464757386)

[4.3.3 登录主数据库查看master状态 10](#_Toc464757387)

[4.3.4 备库执行change master to系列命令 10](#_Toc464757388)

[4.3.5 重启从服务 10](#_Toc464757389)

[4.3.6 解锁主数据库 10](#_Toc464757390)

[4.4 B主A从设置 10](#_Toc464757391)

[4.5 B主C从设置 11](#_Toc464757392)

[4.6 C主D从设置 11](#_Toc464757393)

[4.7 验证主从同步 11](#_Toc464757394)

[5 Keepalived集群架构 11](#_Toc464757395)

[6 keepalived 安装配置 11](#_Toc464757396)

[6.1 配置keepalived 12](#_Toc464757397)

[6.1.1 主服务器配置 14](#_Toc464757398)

[6.1.2 从服务器配置 16](#_Toc464757399)

[7 Nginx配置 17](#_Toc464757400)

[8 验证集群 18](#_Toc464757401)

[8.1 主从复制验证 18](#_Toc464757402)

[8.2 性能验证 18](#_Toc464757403)

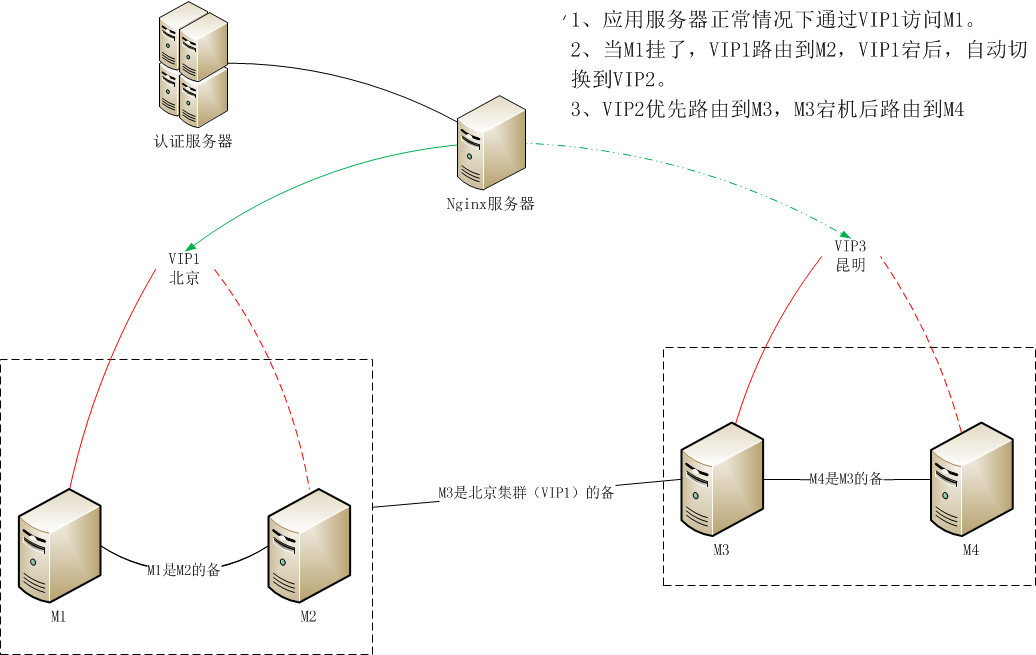
[8.3 稳定性验证 21](#_Toc464757404)

[8.3.1 读操作（选取“加密”接口）测试 22](#_Toc464757405)

[8.3.2 写操作（选取“申请插件”接口）测试 24](#_Toc464757406)

# 集群网络架构

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实际地址（RIP） | HostName | Port | Slave=>Master | 虚拟地址（VIP） | Jdbc Url | 代理连接方式 | 描述 | 备注 |
| 192.168.1.146 | mysql1 | 3306 | mysql2 | 192.168.1.98 权值:mysql1>mysql2 | jdbc:mysql://192.168.1.98/sotp?useUnicode=true&characterEncoding=UTF-8 | jdbc:mysql:// 192.168.1.152:3306/sotp?useUnicode=true&characterEncoding=UTF-8 | 充当实际服务提供者 | 两个VIP切换采用nginx代理转发 |
| 192.168.1.185 | mysql2 | 3306 | mysql1 |
| 192.168.1.208 | mysql3 | 3306 | mysql2 | 192.168.1.99 权值:mysql3>mysql4 | jdbc:mysql://192.168.1.99/sotp?useUnicode=true&characterEncoding=UTF-8 | 充当容灾服务 |
| 192.168.1.216 | mysql4 | 3306 | mysql3 |
| 192.168.1.152 | nginx |  |  |  |  |  | nginx服务器 |  |



# 使用yum安装mysql

分别在192.168.1.146、192.168.1.185、192.168.1.208和192.168.1.216服务器上安装mysql。

## 安装mysql依赖

yum -y install gcc libxml2-dev curl screen libpng12-dev autoconf libpcre3-dev make bzip2 libevent-dev patch libjpeg62-dev libcurl4-openssl-dev libfreetype6-dev g++ libtool libncurses5-dev psmisc lrzsz gcc-c++

## 编辑yum源

**vi  /etc/yum.repos.d/mysql-community.repo**

|  |
| --- |
| **# Enable to use MySQL 5.6**  **[mysql56-community]**  **name=MySQL 5.6 Community Server**  **baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/6/$basearch/**  **enabled=1**  **gpgcheck=0**  **# gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql** |

## yum安装mysql

**yum repolist enabled | grep mysql**

**sudo yum install mysql-community-server**

**sudo service mysqld start**

**sudo service mysqld status**

## 修改root 用户口令

**mysql\_secure\_installation**

根据提示设置口令

## 设置开机启动

chkconfig mysqld on

## 安装mysql周边产品

Installing Additional MySQL Products and Components

**yum --disablerepo=\\* --enablerepo='mysql\*-community\*' list available**

Install any packages of your choice with the following command, replacing ***package-name*** with name of the package (for dnf-enabled systems, replace **yum** in the command with **dnf**):

**sudo yum install *package-name***

eg： yum install -y mysql-community-test.x86\_64

## 登录mysql

mysql -u root -p

忘记密码的处理方式：

1、停止mysql ：service mysqld stop

2、 利用vim命令打开mysql配置文件my.cnf 在mysqld进程配置文件中添加skip-grant-tables,添加完成后,执行wd保存。

3、重启数据库 service mysqld start

4、修改root密码 mysql -p

update mysql.user  set password=password('newpassword') where user='root' ;

5、注释或删除 skip-grant-tables 然后重启数据库

几个重要目录：

(a) 数据库目录 /var/lib/mysql/

(b) 配置文件 /usr/share/mysql（mysql.server命令及配置文件）

(c) 相关命令 /usr/bin（mysqladmin mysqldump等命令）

(d) 启动脚本 /etc/rc.d/init.d/（启动脚本文件mysql的目录）

## 关闭防火墙

service iptables stop #本次生效

chkconfig iptables off #重启生效

# 数据库参数配置

|  |
| --- |
| [mysqld]  port = 3306  socket = /tmp/mysql.sock  basedir = /usr/local/mysql  datadir = /data/mysql  pid-file = /data/mysql/mysql.pid  user = mysql  bind-address = 0.0.0.0  server-id = 185  sync\_binlog=1  binlog-ignore-db=mysql  replicate-ignore-db=mysql  log-slave-updates  auto\_increment\_offset=1  auto\_increment\_increment=2  init-connect = 'SET NAMES utf8mb4'  character-set-server = utf8mb4  skip-name-resolve  #skip-networking  back\_log = 300  max\_connections = 1000  max\_connect\_errors = 6000  open\_files\_limit = 65535  table\_open\_cache = 1024  max\_allowed\_packet = 4M  binlog\_cache\_size = 1M  max\_heap\_table\_size = 8M  tmp\_table\_size = 128M  read\_buffer\_size = 2M  read\_rnd\_buffer\_size = 8M  sort\_buffer\_size = 8M  join\_buffer\_size = 8M  key\_buffer\_size = 256M  thread\_cache\_size = 64  query\_cache\_type = 1  query\_cache\_size = 64M  query\_cache\_limit = 2M  ft\_min\_word\_len = 4  log\_bin = mysql-bin  binlog\_format = mixed  expire\_logs\_days = 7  log\_error = /data/mysql/mysql-error.log  slow\_query\_log = 1  long\_query\_time = 1  slow\_query\_log\_file = /data/mysql/mysql-slow.log  performance\_schema = 0  explicit\_defaults\_for\_timestamp  #lower\_case\_table\_names = 1  skip-external-locking  default\_storage\_engine = InnoDB  #default-storage-engine = MyISAM  innodb\_file\_per\_table = 1  innodb\_open\_files = 500  innodb\_buffer\_pool\_size = 1024M  innodb\_write\_io\_threads = 4  innodb\_read\_io\_threads = 4  innodb\_thread\_concurrency = 0  innodb\_purge\_threads = 1  innodb\_flush\_log\_at\_trx\_commit = 2  innodb\_log\_buffer\_size = 2M  innodb\_log\_file\_size = 32M  innodb\_log\_files\_in\_group = 3  innodb\_max\_dirty\_pages\_pct = 90  innodb\_lock\_wait\_timeout = 120  bulk\_insert\_buffer\_size = 8M  myisam\_sort\_buffer\_size = 64M  myisam\_max\_sort\_file\_size = 10G  myisam\_repair\_threads = 1  wait\_timeout= 1814400  interactive\_timeout= 1814400 |

## 设置Mysql远程访问

mysql -uroot -p

grant all privileges on \*.\* to 'sotp'@'%' identified by 'sotp123' with grant option;

grant all privileges on \*.\* to 'admin'@'%' identified by 'sotp123' with grant option;

grant all privileges on \*.\* to 'root'@'%' identified by 'sotp123' with grant option;

# mysql双主互为备份设置

## 设置auto\_increment、server\_id等

A服务器 /etc/my.cnf：

|  |
| --- |
| [mysqld]  。。。。。。  # ip地址第4段即第3个点后数字  server\_id=146  log\_bin=mysql\_bin  sync\_binlog=1  log\_bin\_trust\_function\_creators=1  binlog-ignore-db=mysql  replicate-ignore-db=mysql  #log-slave-updates这个参数用来配置从服务器的更新是否写入二进制日志，这个选项默认是不打开的，但是，如果这个从服务器B是服务器A的从服务器，同时还作为服务器C的主服务器，那么就需要开发这个选项，这样它的从服务器C才能获得它的二进制日志进行同步操作。  log-slave-updates  #从1开始每次递增4(集群中有几台机器就设置为几)  auto\_increment\_offset=1  auto\_increment\_increment=4  。。。。。。 |

B服务器 /etc/my.cnf：

|  |
| --- |
| [mysqld]  。。。。。。  # ip地址第4段即第3个点后数字  server\_id=185  log\_bin=mysql\_bin  sync\_binlog=1  log\_bin\_trust\_function\_creators=1  binlog-ignore-db=mysql  replicate-ignore-db=mysql  #log-slave-updates这个参数用来配置从服务器的更新是否写入二进制日志，这个选项默认是不打开的，但是，如果这个从服务器B是服务器A的从服务器，同时还作为服务器C的主服务器，那么就需要开发这个选项，这样它的从服务器C才能获得它的二进制日志进行同步操作。  log-slave-updates  #从1开始每次递增4(集群中有几台机器就设置为几)  auto\_increment\_offset=2  auto\_increment\_increment=4  。。。。。。 |

C服务器 /etc/my.cnf：

|  |
| --- |
| [mysqld]  。。。。。。  # ip地址第4段即第3个点后数字  server\_id=208  log\_bin=mysql\_bin  sync\_binlog=1  log\_bin\_trust\_function\_creators=1  binlog-ignore-db=mysql  replicate-ignore-db=mysql  #log-slave-updates这个参数用来配置从服务器的更新是否写入二进制日志，这个选项默认是不打开的，但是，如果这个从服务器B是服务器A的从服务器，同时还作为服务器C的主服务器，那么就需要开发这个选项，这样它的从服务器C才能获得它的二进制日志进行同步操作。  log-slave-updates  #从1开始每次递增4(集群中有几台机器就设置为几)  auto\_increment\_offset=3  auto\_increment\_increment=4  。。。。。。 |

D服务器 /etc/my.cnf：

|  |
| --- |
| [mysqld]  。。。。。。  # ip地址第4段即第3个点后数字  server\_id=216  log\_bin=mysql\_bin  sync\_binlog=1  log\_bin\_trust\_function\_creators=1  binlog-ignore-db=mysql  replicate-ignore-db=mysql  #log-slave-updates这个参数用来配置从服务器的更新是否写入二进制日志，这个选项默认是不打开的，但是，如果这个从服务器B是服务器A的从服务器，同时还作为服务器C的主服务器，那么就需要开发这个选项，这样它的从服务器C才能获得它的二进制日志进行同步操作。  log-slave-updates  #从1开始每次递增4(集群中有几台机器就设置为几)  auto\_increment\_offset=4  auto\_increment\_increment=4  。。。。。。 |

## 重启mysql

service mysqld stop

service mysqld start

或者

service mysqld restart

注：A、B服务器均需操作

## A主B从设置

### 锁定主数据库DDL操作

B服务器上操作

mysql> flush tables with read lock;

### 停止从服务

A服务器上操作

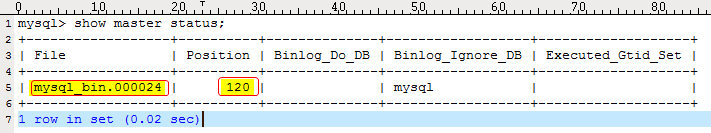
mysql>stop slave;

mysql>reset slave;

### 登录主数据库查看master状态

B服务器上操作

mysql>show master status;



### 备库执行change master to系列命令

A服务器上操作

–该用户要有Repl\_slave\_priv权限，没有可用

grant replication slave on \*.\* to 'sync'@'%' identified by 'sync';

flush privileges;

–根据主服务器show master status出来的File 和 Position结果填写

|  |
| --- |
| change master to master\_host='192.168.xxx.xxx', master\_user='sync', master\_password='sync', master\_log\_file='mysql\_bin.000024', master\_log\_pos=120; |

master\_host 表示指定其主库地址。master\_log\_file表示主库当前日志文件名。master\_log\_pos表示当前主库日志指针位置。

### 重启从服务

A服务器上操作

mysql> start slave;

### 解锁主数据库

B服务器上操作

mysql> unlock tables;

## B主A从设置

将3.3设置中的A换为B，B换为A执行3.3.1到3.3.6步骤。

## B主C从设置

将3.3设置中的A换为B，B换为C执行3.3.1到3.3.6步骤。

## C主D从设置

将3.3设置中的A换为C，B换为D执行3.3.1到3.3.6步骤。

## 验证主从同步

A、B服务器上执行

mysql> show slave status;

+--------------------------+---------------------------+

| Slave\_IO\_Running | Slave\_SQL\_Running |

+--------------------------+---------------------------+

| Yes | Yes |

+--------------------------+----------------------------+

A服务器上创建一个库并创建一张表，B服务器上查看是否存在；

B服务器上创建一个库并创建一张表，A服务器上查看是否存在。

# Keepalived集群架构

keepalived 配置文件分3类：全局配置 VRRPD配置 LVS配置

# 或 ! 打头的是注释

VRRPD配置 分为VRRP同步组合VRRP实例配置

将所有VRRP实例都加入同步组中，任何一个实例出现问题都会导致keepalived进行主备切换。

notify\_master notify\_backup notify\_fault notify\_stop 分别表示keepalived相应状态执行脚本。

state MASTER/BACKUP 表示主机主服务/备用服务。LVA以virtual\_server 作为开始标识，分为两部分：real\_server段和健康检查段。

# keepalived 安装配置

双主模式允许两台机器均处于工作状态并互相作为备份。

搭建keepalived双方模式的要素：

1、必须有两个虚拟IP, 分别绑定至两个节点上

2、每个节点作为某个虚拟IP的主节点，并同时作为另外一个虚拟IP的备用节点。

3、当某个节点产生故障时，两个虚拟IP自动绑定至正常节点上

yum安装方式： yum install -y keepalived

安装 openssl-devel : yum install openssl-devel -y

[root@people ~]# which openssl

/usr/bin/openssl

[root@people ~]# ldd /usr/bin/openssl

linux-vdso.so.1 => (0x00007fffca76c000)

libssl.so.10 => /usr/lib64/libssl.so.10 (0x00007f0b17081000)

libgssapi\_krb5.so.2 => /lib64/libgssapi\_krb5.so.2 (0x00007f0b16e3d000)

libkrb5.so.3 => /lib64/libkrb5.so.3 (0x00007f0b16b55000)

libcom\_err.so.2 => /lib64/libcom\_err.so.2 (0x00007f0b16951000)

libk5crypto.so.3 => /lib64/libk5crypto.so.3 (0x00007f0b16725000)

libcrypto.so.10 => /usr/lib64/libcrypto.so.10 (0x00007f0b16340000)

libdl.so.2 => /lib64/libdl.so.2 (0x0000003c86800000)

libz.so.1 => /lib64/libz.so.1 (0x0000003c86000000)

libc.so.6 => /lib64/libc.so.6 (0x0000003c86400000)

libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007f0b16134000)

libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f0b15f31000)

libresolv.so.2 => /lib64/libresolv.so.2 (0x0000003c88000000)

libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003c86c00000)

/lib64/ld-linux-x86-64.so.2 (0x0000003c85c00000)

libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f0b15d10000)

[root@people ~]#

[root@people ~]# openssl version

OpenSSL 1.0.1e-fips 11 Feb 2013

[root@people ~]#

配置环境变量：

export JAVA\_HOME=/usr/local/jdk

export MYCAT\_HOME=/usr/local/mycat

export OPENSSL=/usr/bin/openssl

export PATH=$JAVA\_HOME/bin:$MYCAT\_HOME/bin:$OPENSSL/bin:$PATH

## 配置keepalived

进入目录 /etc/keepalived 创建脚本notify.sh 。本脚本分别部署到A、B、C和D服务器。

注意：各个脚本权限 777

|  |
| --- |
| #!/bin/bash  #set -x  # James  # ip addr show  contact='root@localhost'  LOGFILE="/etc/keepalived/keepalived-haproxy-state.log"  start() {  cnt=`ps -C mysqld --no-header|wc -l`  if [ $cnt -eq 0 ];then  ( /sbin/service mysqld start ) 1>> $LOGFILE 2>> $LOGFILE  fi  }  stop() {  ( /sbin/service mysqld stop ) 1>> $LOGFILE 2>> $LOGFILE  }  check() {  cnt=`ps -C mysqld --no-header|wc -l`  if [ $cnt -eq 0 ];then  ( /sbin/service keepalived stop ) 1>> $LOGFILE 2>> $LOGFILE  exit 1  fi  exit 0  }  notify() {  mailsubject="`hostname` to be $1: $vip floating"  mailbody="`date '+%F %H:%M:%S'`: vrrp transition, `hostname` changed to be $1"  echo $mailbody | mail -s "$mailsubject" $contact  echo "[ $1 ]" >> $LOGFILE  date >> $LOGFILE  }  case "$1" in  master)  notify master  start  exit 0  ;;  backup)  notify backup  start  exit 0  ;;  fault)  notify fault  stop  sleep 3  start  exit 0  ;;  check)  #notify check  check  exit 0  ;;  \*)  echo 'Usage: `basename $0` {master|backup|fault|check}'  exit 1  ;;  esac |

### 主服务器配置

A服务器

|  |
| --- |
| ! Configuration File for keepalived  vrrp\_script chk\_alive {  script "/etc/keepalived/notify.sh check"  interval 1  weight 30  }  vrrp\_instance VI\_1 {  state MASTER #备机 上改为BACKUP  interface eth0 #对外提供服务的网络接口  virtual\_router\_id 91 #VRRP组名，两个节点的设置必须一样，以指明各个节点属于同一VRRP组  priority 150 #数值愈大，优先级越高，backup上改为130  advert\_int 1 #同步通知间隔  authentication { #包含验证类型和验证密码。类型主要有PASS、AH两种，通常使用的类型为PASS，据说AH使用时有问题  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_alive  }  virtual\_ipaddress { #vip地址  192.168.1.98/24  }  notify\_master "/etc/keepalived/notify.sh master"  notify\_backup "/etc/keepalived/notify.sh backup"  notify\_fault "/etc/keepalived/notify.sh fault"  notify\_stop "/etc/keepalived/notify.sh stop"  } |

C服务器

|  |
| --- |
| ! Configuration File for keepalived  vrrp\_script chk\_alive {  script "/etc/keepalived/notify.sh check"  interval 1  weight 30  }  vrrp\_instance VI\_1 {  state MASTER #备机 上改为BACKUP  interface eth0 #对外提供服务的网络接口  virtual\_router\_id 91 #VRRP组名，两个节点的设置必须一样，以指明各个节点属于同一VRRP组  priority 150 #数值愈大，优先级越高，backup上改为130  advert\_int 1 #同步通知间隔  authentication { #包含验证类型和验证密码。类型主要有PASS、AH两种，通常使用的类型为PASS，据说AH使用时有问题  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_alive  }  virtual\_ipaddress { #vip地址  192.168.1.99/24  }  notify\_master "/etc/keepalived/notify.sh master"  notify\_backup "/etc/keepalived/notify.sh backup"  notify\_fault "/etc/keepalived/notify.sh fault"  notify\_stop "/etc/keepalived/notify.sh stop"  } |

### 从服务器配置

B服务器

|  |
| --- |
| ! Configuration File for keepalived  vrrp\_script chk\_alive {  script "/etc/keepalived/notify.sh check"  interval 1  weight 30  }  vrrp\_instance VI\_1 {  state BACKUP #备机 上改为BACKUP  interface eth0 #对外提供服务的网络接口  virtual\_router\_id 91 #VRRP组名，两个节点的设置必须一样，以指明各个节点属于同一VRRP组  priority 130 #数值愈大，优先级越高，backup上改为130  advert\_int 1 #同步通知间隔  authentication { #包含验证类型和验证密码。类型主要有PASS、AH两种，通常使用的类型为PASS，据说AH使用时有问题  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_alive  }  virtual\_ipaddress { #vip地址  192.168.1.98/24  }  notify\_master "/etc/keepalived/notify.sh master"  notify\_backup "/etc/keepalived/notify.sh backup"  notify\_fault "/etc/keepalived/notify.sh fault"  notify\_stop "/etc/keepalived/notify.sh stop"  } |

D服务器

|  |
| --- |
| ! Configuration File for keepalived  vrrp\_script chk\_alive {  script "/etc/keepalived/notify.sh check"  interval 1  weight 30  }  vrrp\_instance VI\_1 {  state BACKUP #备机 上改为BACKUP  interface eth0 #对外提供服务的网络接口  virtual\_router\_id 91 #VRRP组名，两个节点的设置必须一样，以指明各个节点属于同一VRRP组  priority 130 #数值愈大，优先级越高，backup上改为130  advert\_int 1 #同步通知间隔  authentication { #包含验证类型和验证密码。类型主要有PASS、AH两种，通常使用的类型为PASS，据说AH使用时有问题  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_alive  }  virtual\_ipaddress { #vip地址  192.168.1.99/24  }  notify\_master "/etc/keepalived/notify.sh master"  notify\_backup "/etc/keepalived/notify.sh backup"  notify\_fault "/etc/keepalived/notify.sh fault"  notify\_stop "/etc/keepalived/notify.sh stop"  } |

# Nginx配置

root 用户登录192.168.1.152服务器 进入nginx配置文件目录，即cd /usr/local/nginx/conf

编辑 nginx.conf

在 stream 下增加以下内容：

|  |
| --- |
| stream {  。。。。。。  #mysql vip setup ## start  upstream db\_3306 {  server 192.168.1.98:3306;  server 192.168.1.100:3306 backup;  }  server {  listen 3306 so\_keepalive=30m::100;  proxy\_connect\_timeout 20s;  proxy\_timeout 300s;  proxy\_pass db\_3306;  }  #mysql vip setup ## end  。。。。。。  } |

# 验证集群

查看进程: ps -ef |egrep "mysql|keepalived "

## 主从复制验证

在146上创建一个数据库，看是否同步到185、208和216

在185上创建一个数据库，看是否同步到146、208和216

## 性能验证

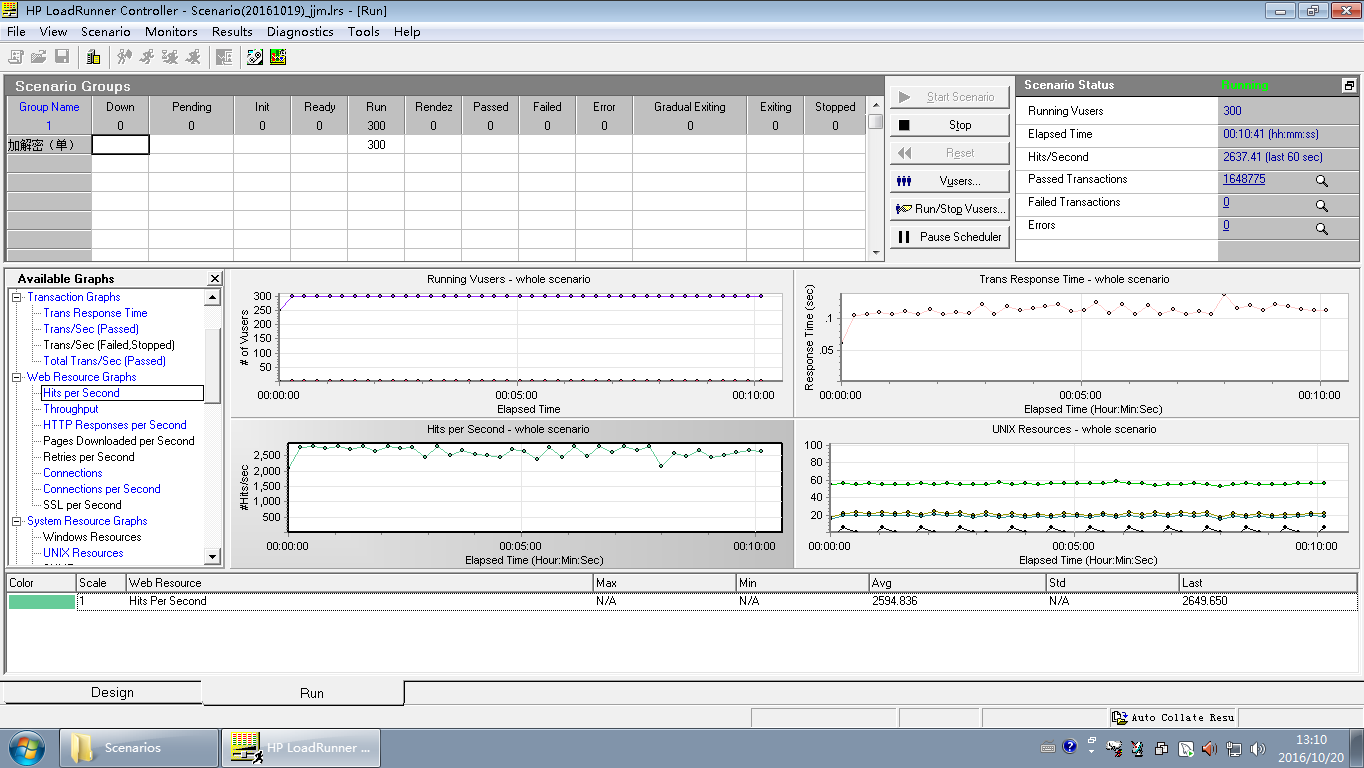
认证系统、密码机和redis不变的情况下测试10 mins左右。

（1）认证系统连接192.168.1.220 数据库，log日志级别error。数据库连接如下：

jdbc:mysql://192.168.1.220:3306/sotp\_jry?useUnicode=true&characterEncoding=UTF-8

LR测试测试”加密”接口性能：

300 Vusers TPS: 2595

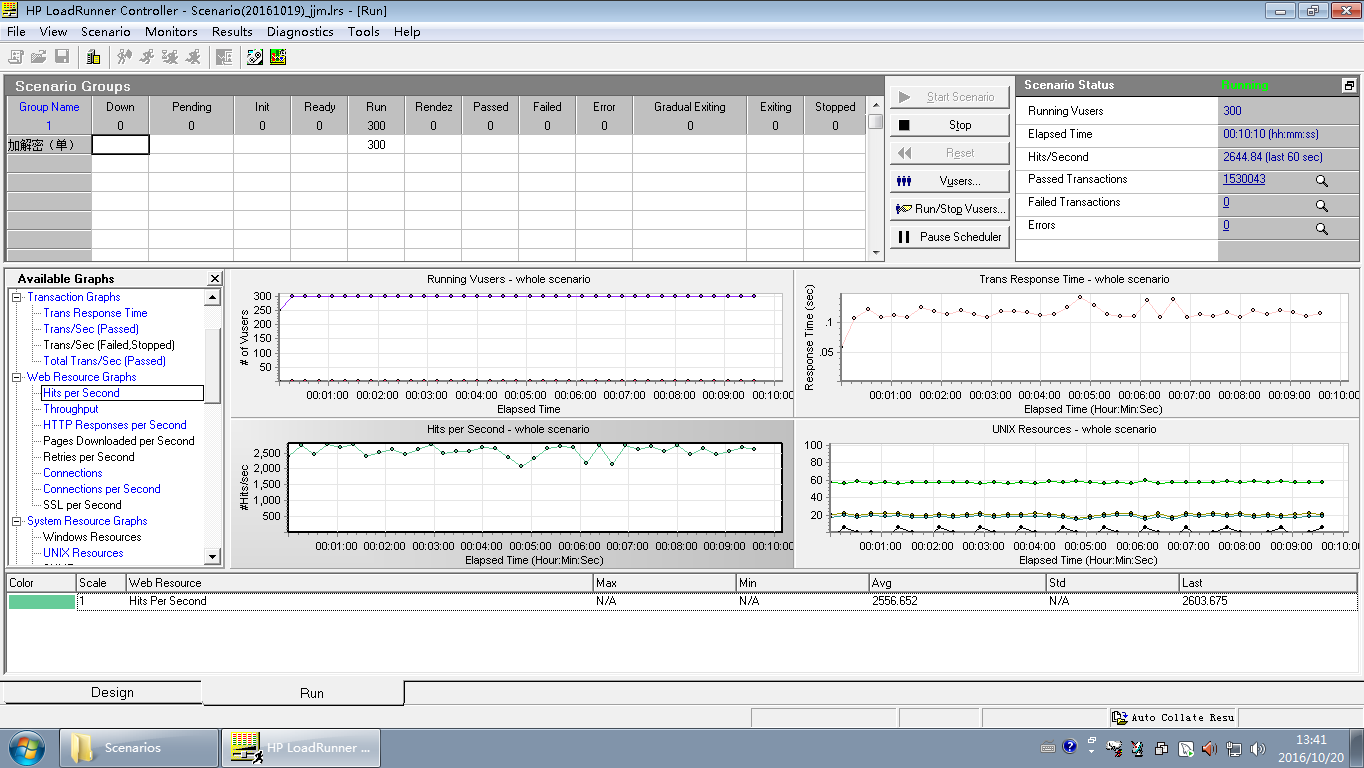


（2）认证系统连接192.168.1.146 数据库（集群中M1），log日志级别error。数据库连接如下：

jdbc.jdbcUrl=jdbc:mysql://192.168.1.146:3306/sotp\_jry?useUnicode=true&characterEncoding=UTF-8

LR测试测试”加密”接口性能：

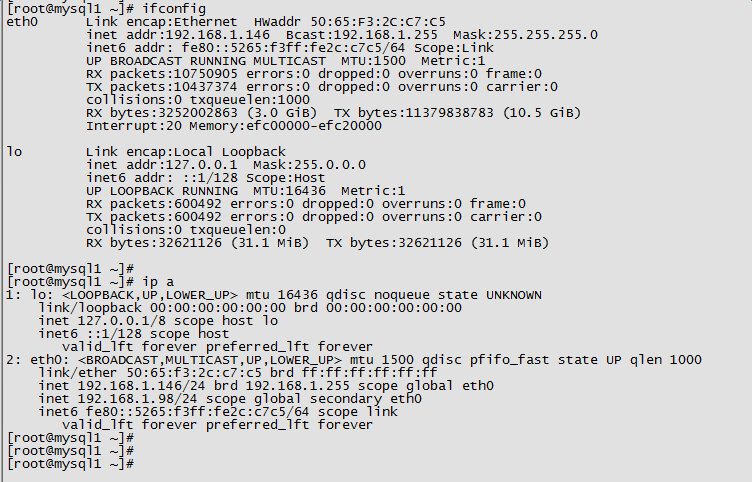
300 Vusers TPS: 2557 与（1）性能差不多



（3）认证系统连接192.168.1.98（98为vip，优先漂移到192.168.1.146），log日志级别error。数据库连接如下：

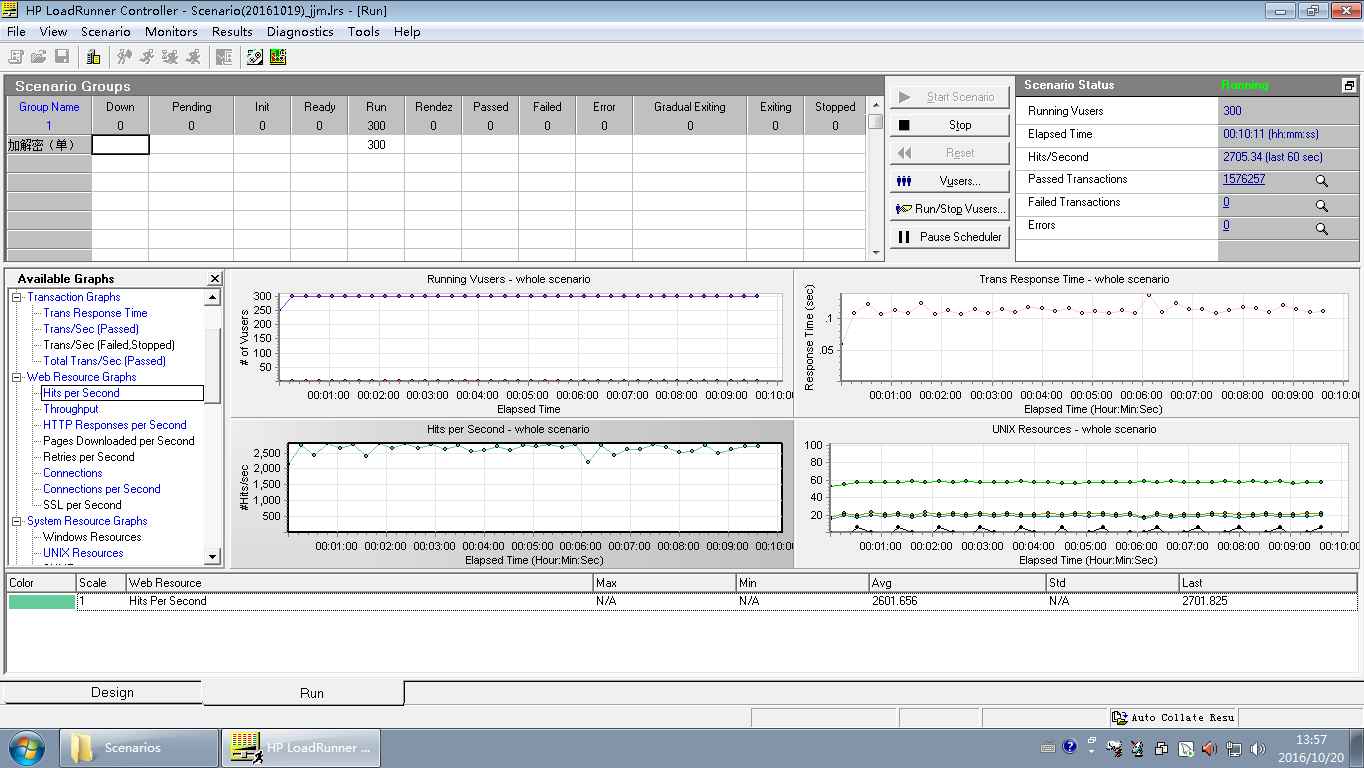
jdbc.jdbcUrl=jdbc:mysql://192.168.1.98:3306/sotp\_jry?useUnicode=true&characterEncoding=UTF-8

vip 漂移情况：



LR测试测试”加密”接口性能：

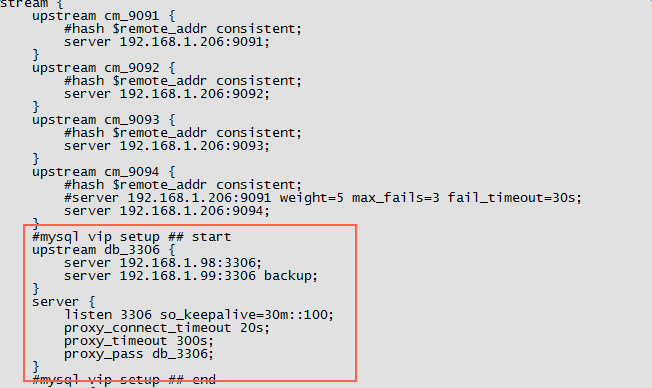
300 Vusers TPS: 2602 与（1）性能差不多



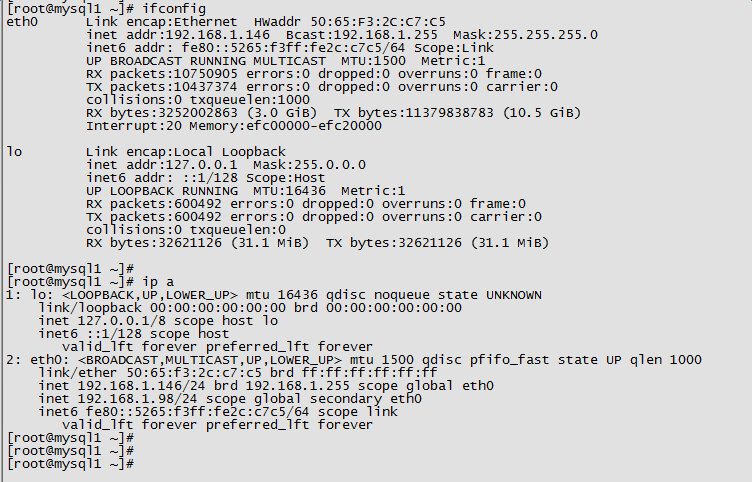
（4）认证系统连接192.168.1.152（152为nginx服务器，优先路由到vip 192.168.1.98），log日志级别error。数据库连接如下：

jdbc.jdbcUrl=jdbc:mysql://192.168.1.152:3306/sotp\_jry?useUnicode=true&characterEncoding=UTF-8

nginx代理转发配置

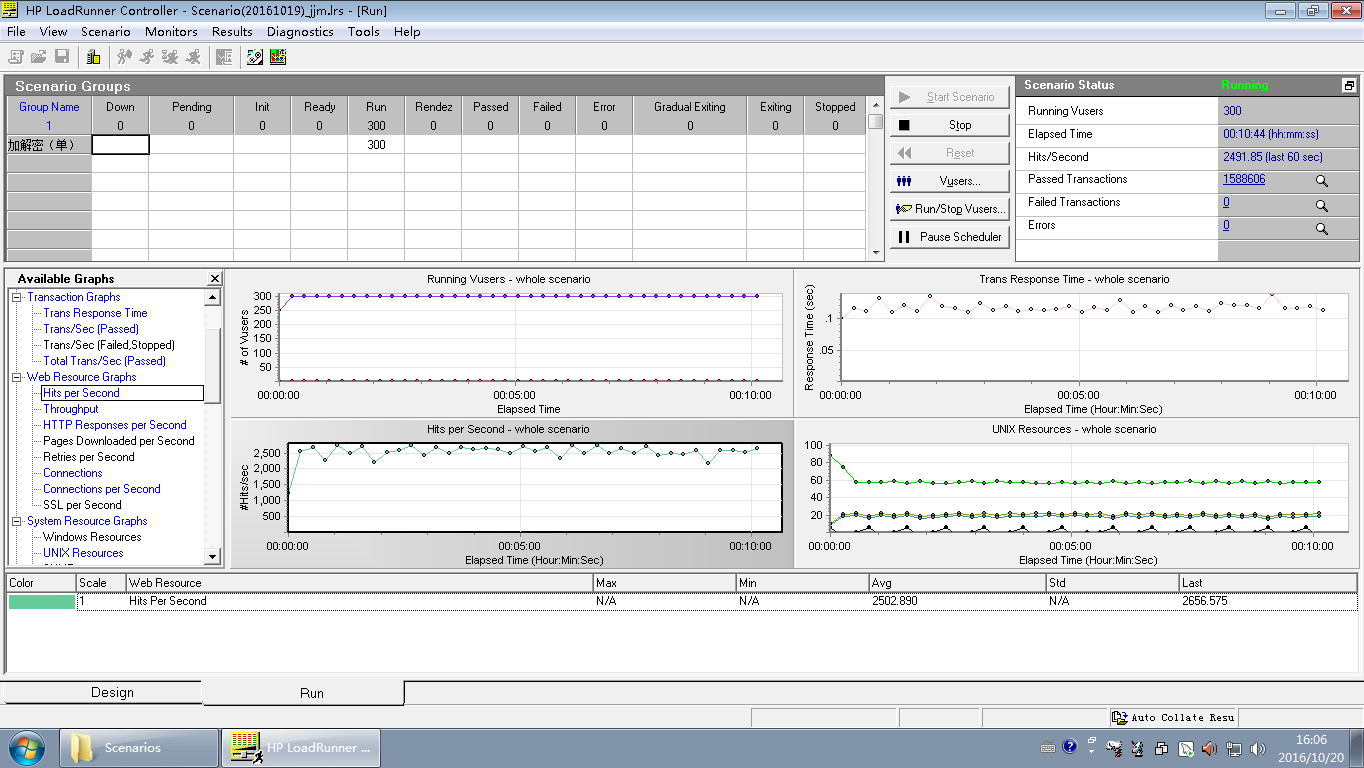


vip 漂移情况：



LR测试测试”加密”接口性能：

300 Vusers TPS: 2503 与（1）性能相差不大



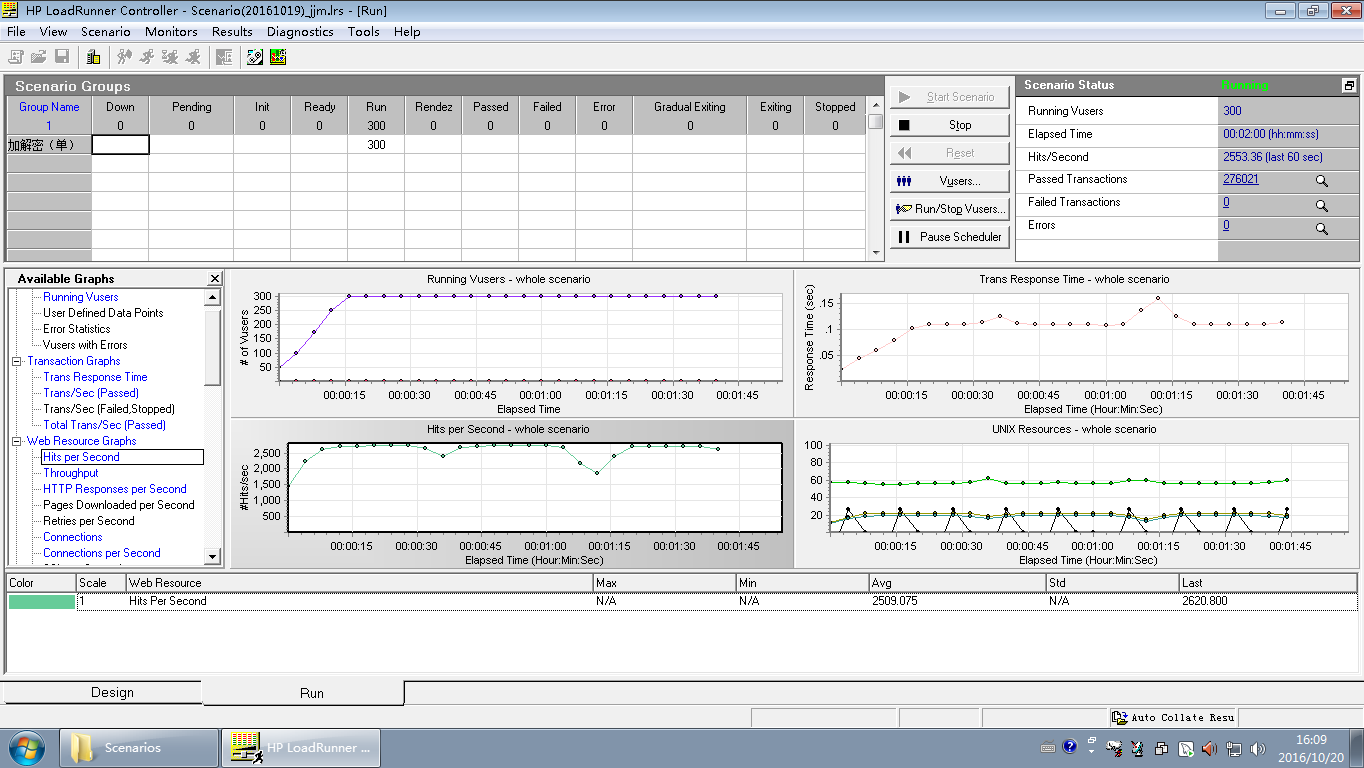
## 稳定性验证

认证系统、密码机和redis不变的情况下测试10 mins左右。认证系统连接192.168.1.152（152为nginx服务器，优先路由到vip 192.168.1.98），log日志级别error。数据库连接如下：

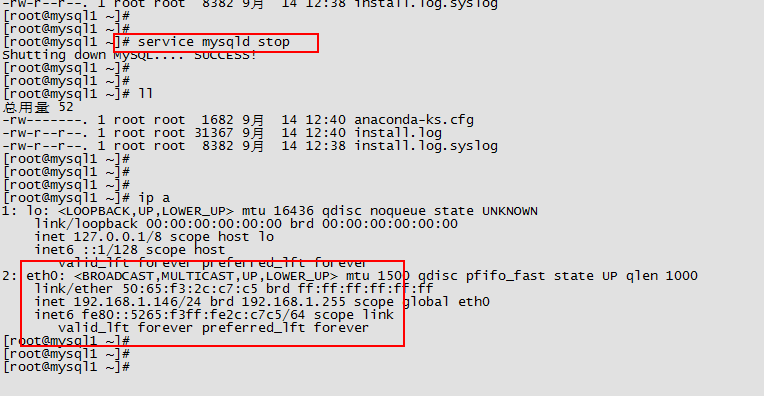
jdbc.jdbcUrl=jdbc:mysql://192.168.1.152:3306/sotp\_jry?useUnicode=true&characterEncoding=UTF-8

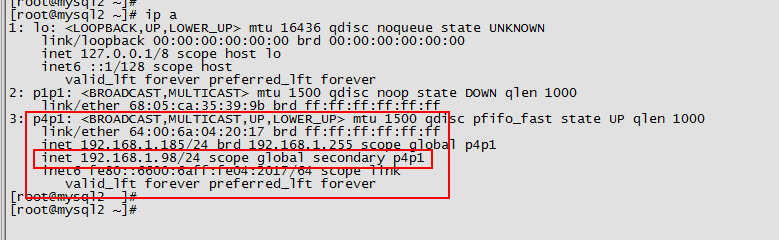
### 读操作（选取“加密”接口）测试

初始LR情况：TPS 2509



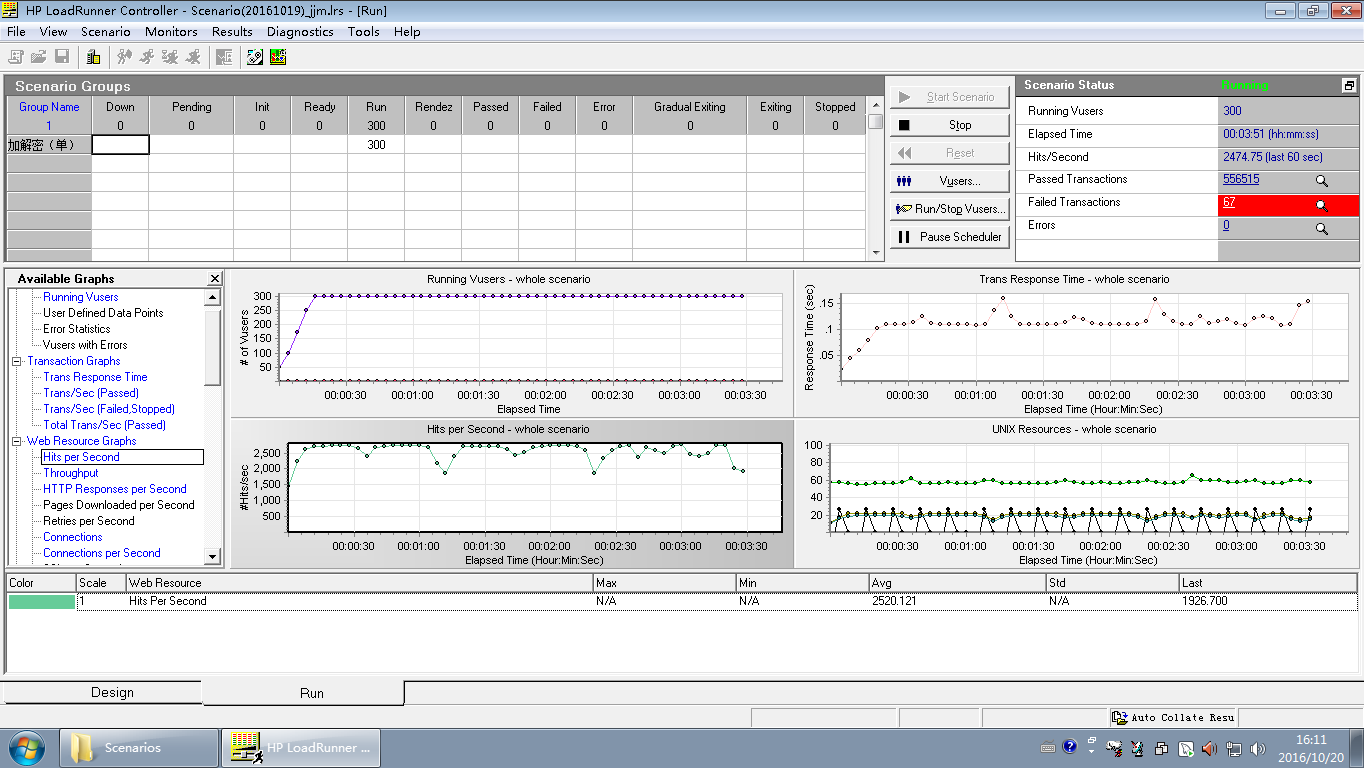
（1）停止146 mysql，nginx负载vip1不变，即192.168.1.98，vip漂移到192.168.1.185，查看LR测试情况：



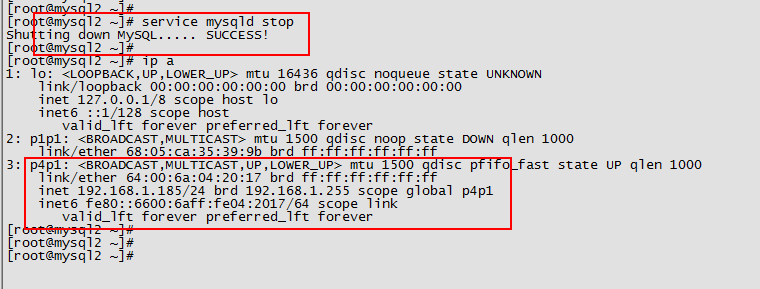


可以查看到vip 已经从146 漂移到了185上。

TPS 无变化 出现少量错误67个

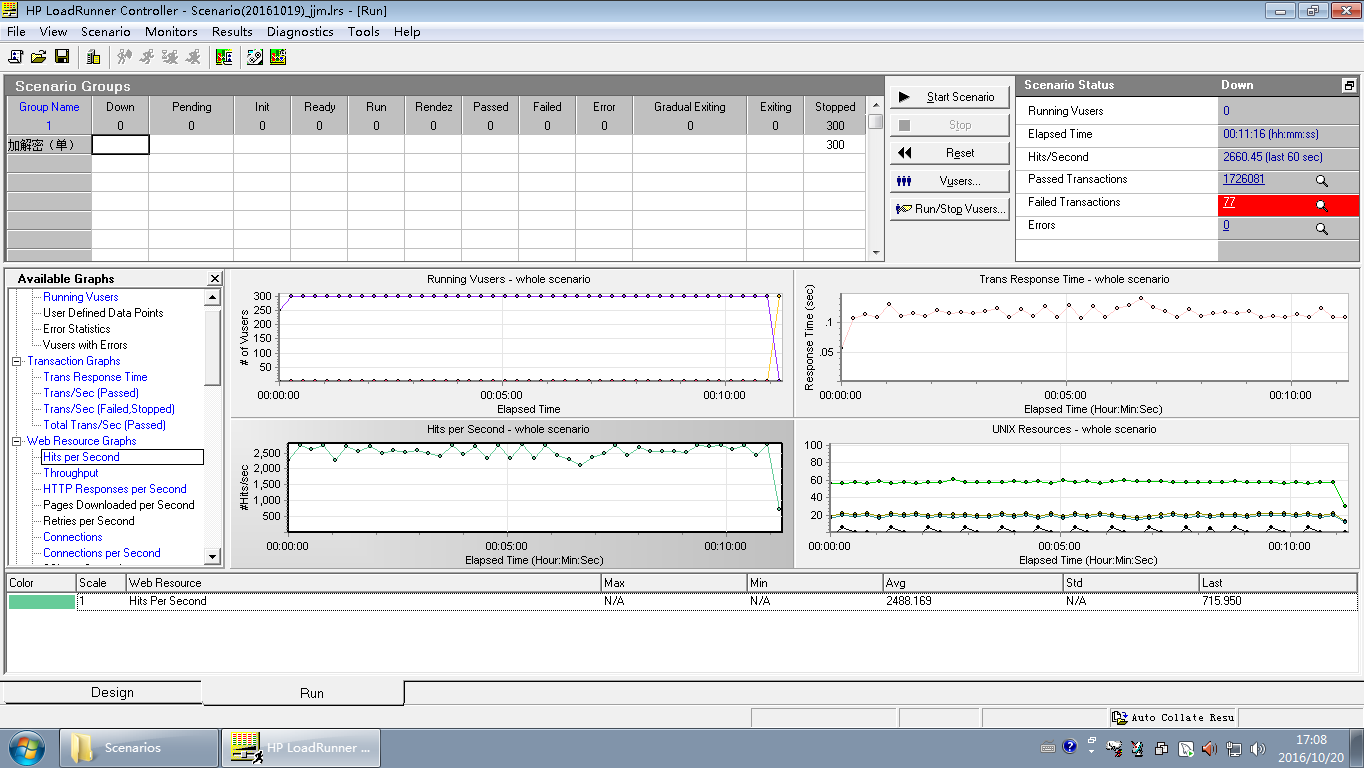


（2）停止185 mysql，nginx负载到vip2，即192.168.1.99，vip漂移到192.168.1.208，查看LR测试情况：



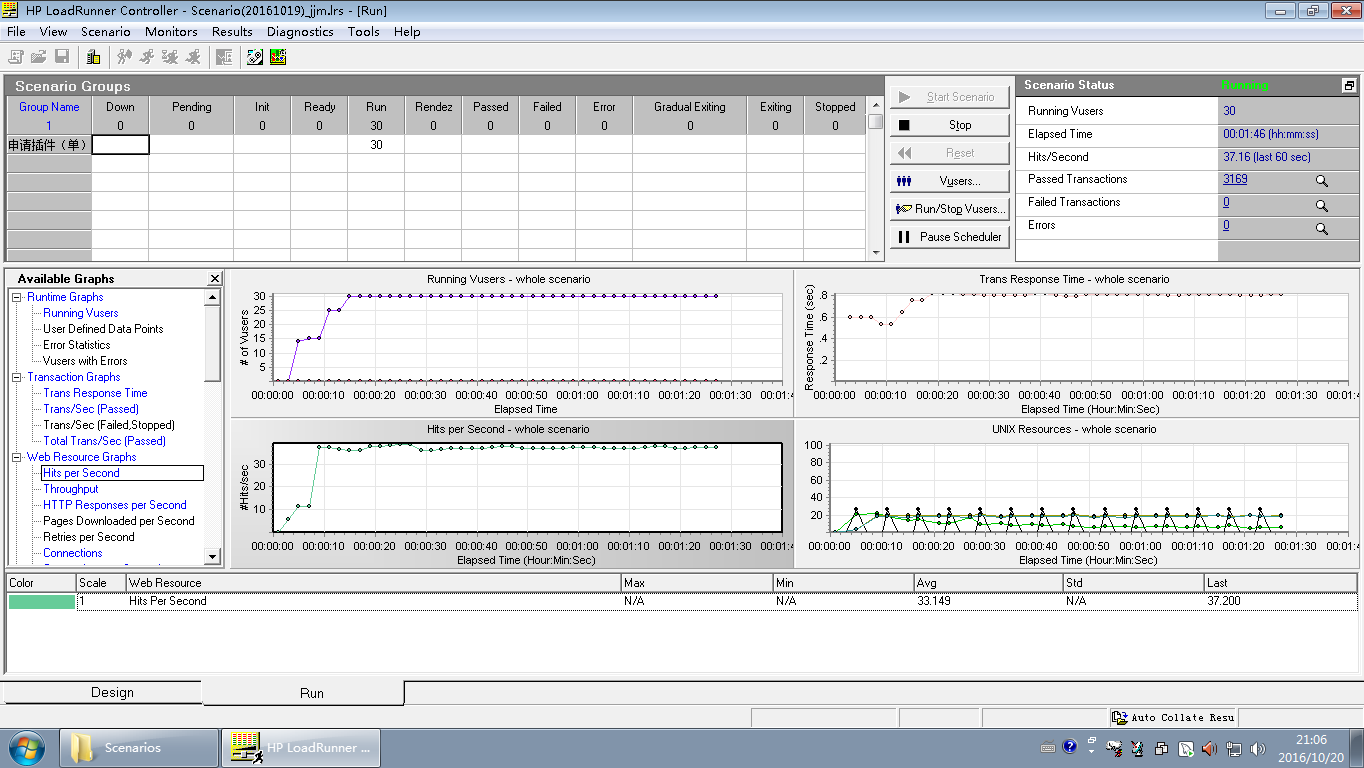
146和185上已经没有vip了。

TPS 无变化 出现少量错误，新增错误数共10个。

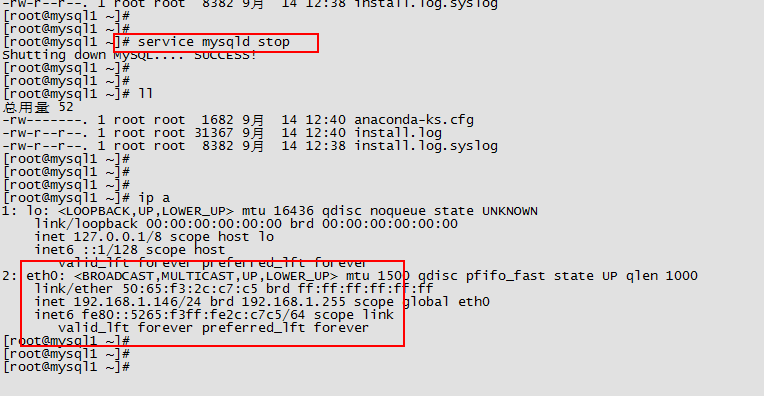


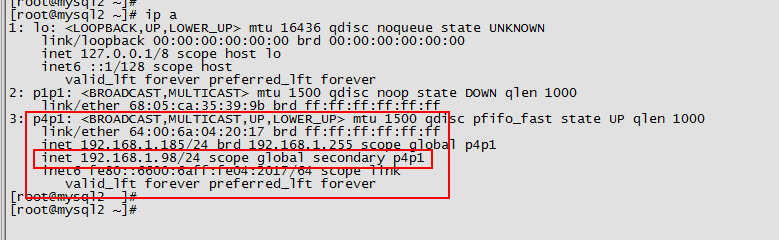
### 写操作（选取“申请插件”接口）测试

初始LR情况：30 Vusers TPS 33



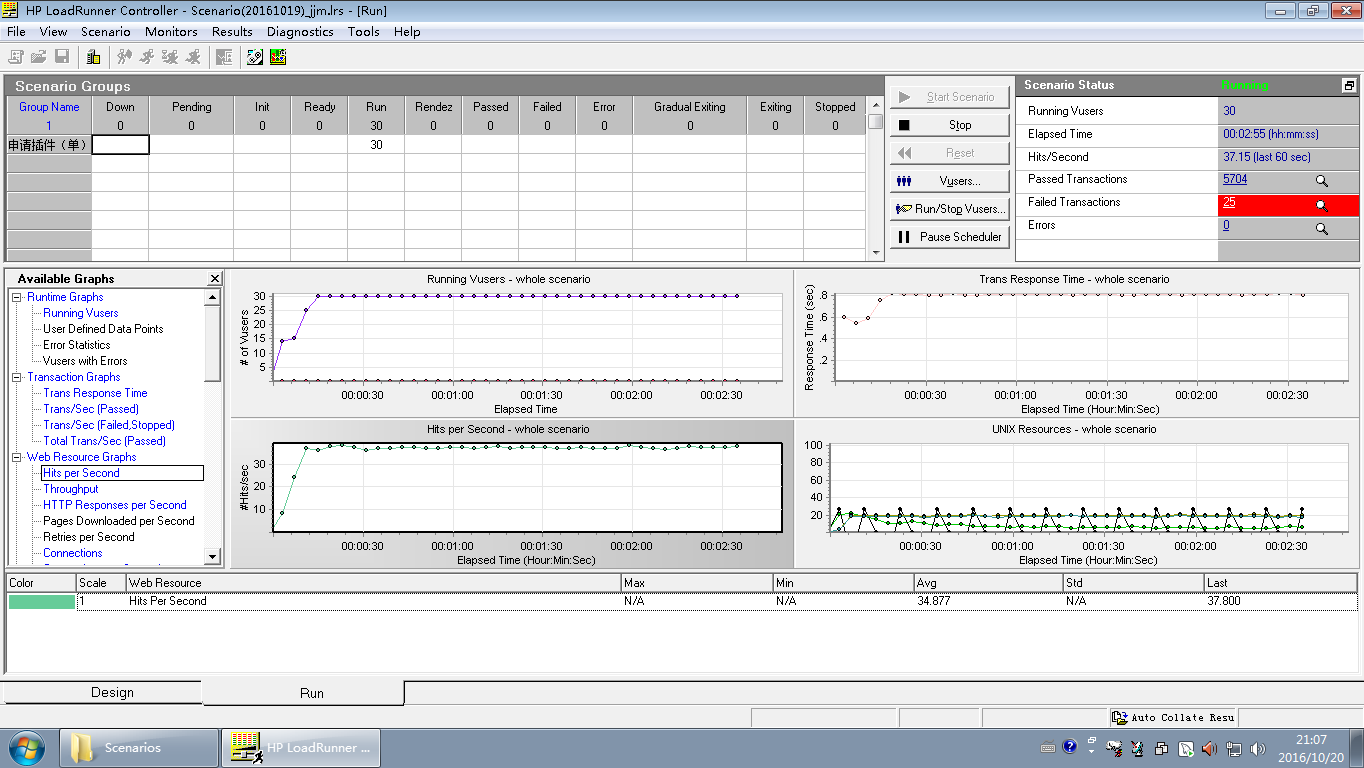
（1）停止146 mysql，nginx负载vip1不变，即192.168.1.98，vip漂移到192.168.1.185，查看LR测试情况：



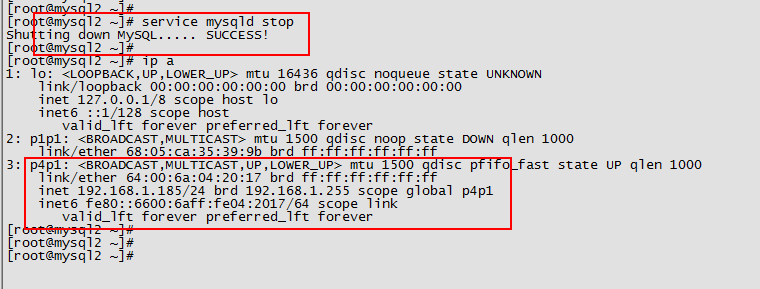


可以查看到vip 已经从146 漂移到了185上。

TPS 无变化 出现少量错误，错误数25个

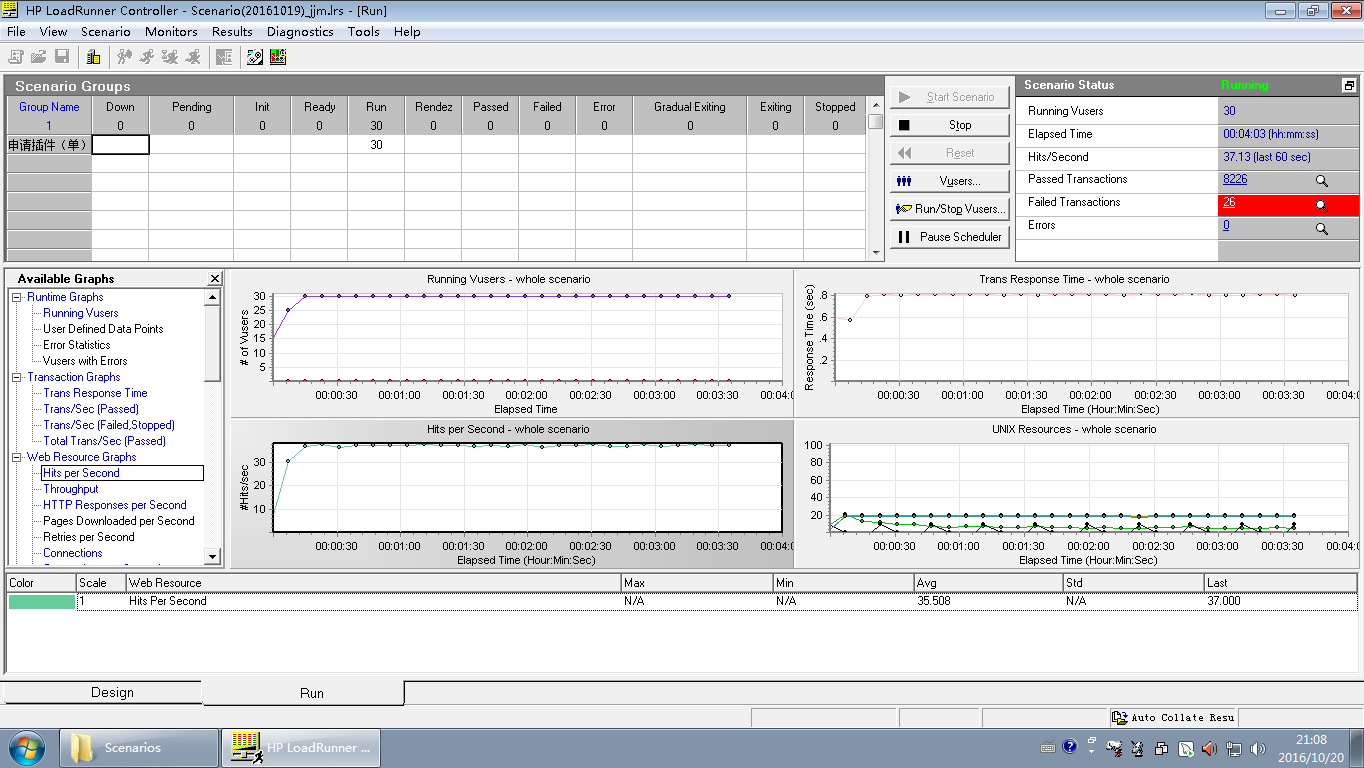


（2）停止185 mysql，nginx负载到vip2，即192.168.1.99，vip漂移到192.168.1.208，查看LR测试情况：



146和185上已经没有vip了。

TPS 无变化 出现少量错误，新增错误数共1个。



（3）重新启动146 和 185 mysql

查看LR测试情况：

TPS 无变化 无新增错误。

