|  |  |  |  |
| --- | --- | --- | --- |
|  | **文 档 编 号** | **产品版本** | **密级** |
|  |  |  |
|  | | **共 页** |

数据库部署文档

**北京瑞金麟网络技术服务有限公司**

**Rkylin Network Technology & Service Co. Ltd.**

**二○一五年十月**

改 版 履 历 表

|  |  |  |  |
| --- | --- | --- | --- |
| **版数** | **发行年月** | **作成者** | **变 更 内 容 概 要** |
| 1.0 | 2015.10.23 | 胡成龙 | 初版 |
|  |  |  |  |
|  |  |  |  |

目录

[1 引言 4](#_Toc433376045)

[1.1 编写目的 4](#_Toc433376046)

[1.2 项目背景 4](#_Toc433376047)

[1.3 目标读者 4](#_Toc433376048)

[1.4 参考文档 4](#_Toc433376049)

[1.5 定义 4](#_Toc433376050)

[2 数据库部署 5](#_Toc433376051)

[2.1 规划 5](#_Toc433376052)

[2.2 安装准备 5](#_Toc433376053)

[2.3 操作步骤 5](#_Toc433376054)

[3 数据库中间件部署 6](#_Toc433376055)

[3.1 规划 6](#_Toc433376056)

[3.2 安装准备 6](#_Toc433376057)

[3.3 操作步骤 6](#_Toc433376058)

# 引言

## 编写目的

本文档主要描述在阿里云ECS上如何部署MySQL+Atlas+MHA高可用架构。

## 项目背景

瑞金麟网络技术服务有限公司目前是中国领先的电子商务一体化服务公司，致力于为品牌商和零售商提供完善的电子商务和互动营销服务。2015年瑞金麟网络技术服务有限公司开始大力发展瑞金麟金融产品中心，力求在电商领域从信息流与资金流形成闭环；在此需求下，需要建立起以虚拟账户系统和资金清结算系统为基础、交易风控系统与运营平台为之保驾护航、代收付系统与智能多渠道路由为能力，最终提供托管记账、网关支付、代收付、BI决策等功能的现代化支付体系。

## 目标读者

需求编写团队、系统设计团队、开发团队、测试团队、运维团队，及所有参与的相关人员。

## 参考文档

无

## 定义

|  |  |  |
| --- | --- | --- |
| 序号 | 定义 | 解释说明 |
| 1 | DB（Data Base） | 数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度的特点、是与应用程序彼此独立的数据集合。 |
| 2 | Atlas | 数据库中间件，实现读写分离，负载均衡，客户端Ip过滤等。 |
| 3 | MHA | MySQL高可用脚本，实现主从复制自动切换 |

# 系统部署

## 系统检查更新软件

1. 版本检查

[root@iZ233yjy5hvZ ~]# cat /etc/redhat-release

CentOS release 6.7 (Final)

1. 查看CPU信息：

cat /proc/cpuinfo

1. 查看内存信息：

cat /proc/meminfo

1. 检测系统是否安装MySQL

rpm -qa | grep -i mysql

1. 移除mysql的包

yum -y remove mysql-libs-\*

1. 删除老版本的数据文件夹和配置文件

rm -rf /var/lib/mysql

rm -rf /etc/my.cnf

1. 检查系统字符集，更新软件

# echo $LANG

如果不是正确字符集，需要修改：

# vi /etc/sysconfig/i18n

en\_US.UTF-8

即时生效：

source /etc/sysconfig/i18n

1. 更新软件：

yum update

## 建数据分区

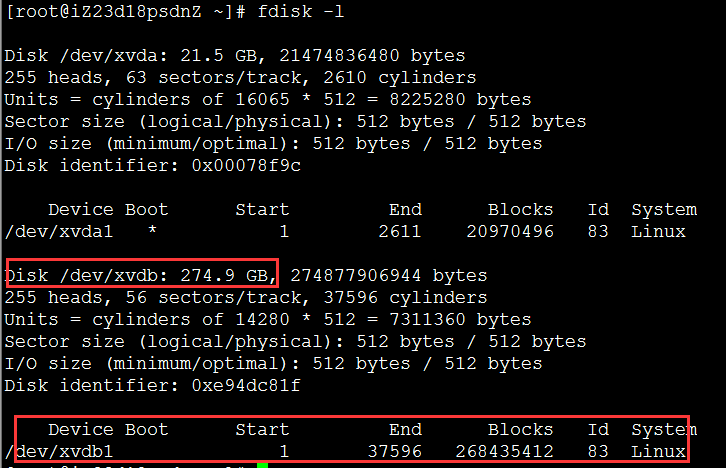
适用系统：Linux（Redhat , CentOS，Debian，Ubuntu）

\* Linux的云服务器数据盘未做分区和格式化，可以根据以下步骤进行分区以及格式化操作。

下面的操作将会把数据盘划分为一个分区来使用。

1. 查看数据盘

可以使用“fdisk -l”命令查看。如下图：



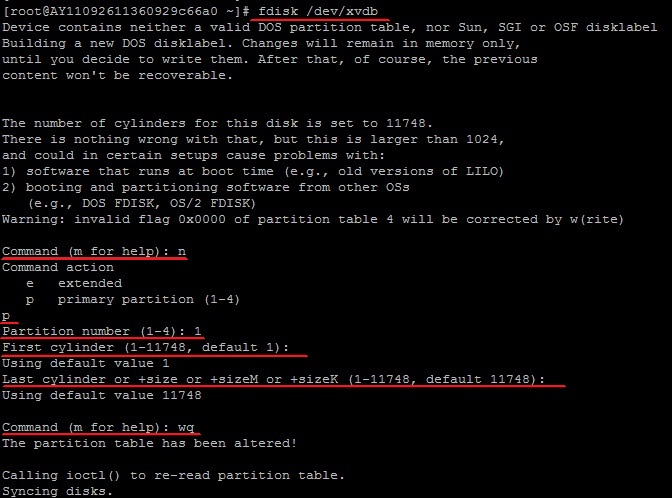
如图显示，已经对/dev/xvdb盘进行了分区，若没有分区信息，需要按照下面方法进行分区。

注：若执行fdisk -l命令，发现没有 /dev/xvdb 标明您的云服务无数据盘，则无法分区了。

1. 对数据盘进行分区

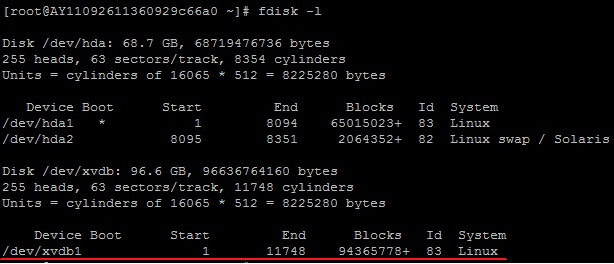
fdisk /dev/xvdb

根据提示，依次输入“n”，“p”，“1”，两次回车，“wq”，分区就开始了，很快就会完成。



1. 查看新的分区

使用“fdisk -l”命令可以看到，新的分区xvdb1已经建立完成了。



1. 格式化新分区

让linux支持XFS文件系统：

yum -y install xfsprogs

mkfs.xfs -f /dev/xvdb1

格式化的时间根据硬盘大小有所不同。

查看文件系统类型：

file -s /dev/xvdb1

1. 添加分区信息

echo '/dev/xvdb1 /mydata xfs defaults 0 0' >> /etc/fstab

cat /etc/fstab

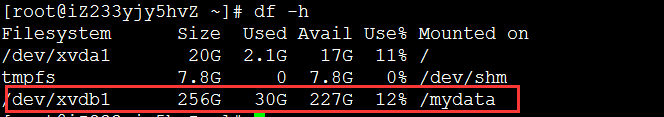
1. 创建目录

mkdir /mydata

1. 挂载新分区

mount –a

df -h



## 建swap分区

Swap空间大小设置标准

|  |  |  |  |
| --- | --- | --- | --- |
| 物理内存大小 | 小于4GB | 大于4GB小于8GB | 大于8GB |
| Swap空间值 | 物理内存\*2 | 物理内存\*1.5 | 物理内存\*1 |

1. 创建分区所在目录

#mkdir /mydata/swap

1. 创建分区

#dd if=/dev/zero of=/mydata/swap/swap bs=1024 count=16777216

count的计算公式: count=SIZE\*1024 (size以MB为单位）

1. 格式化分区

#mkswap -f /mydata/swap/swap

1. 开启分区

#swapon /mydata/swap/swap

1. 查看

#free –m

1. 为了每次开机都会自动启用SWAP，编辑/etc/fstab，在最后加入一行，格式如下：

#echo '/mydata/swap/swap swap swap defaults 0 0' >> /etc/fstab

1. 删除swap分区方法：

首先停止swap分区

#swapoff /mydata/swap/swap

删除swap分区文件

#rm -rf /mydata/swap/swap

删除/etc/fstab文件中以下行

/mydata/swap/swap swap swap defaults 0 0

## 创建系统目录

mkdir -p /mydata/software

mkdir -p /mydata/log

mkdir -p /mydata/scripts

## 配置邮件发送（sendEmail）

1. 拷贝sendEmail到/usr/local/bin

# cp -a sendEmail /usr/local/bin

1. 测试邮件发送

sendEmail -f rkylin2015@yeah.net -s smtp.yeah.net -u "test" -xu rkylin2015@yeah.net -xp 'othnzrexmapxtbdl' -t huchenglong@rongcapital.cn -m "it is a test"

## 安装crond

# yum -y install vixie-cron

# yum install crontabs

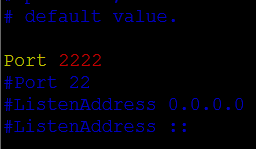
# service crond start //启动服务

查看crontab服务状态：

# service crond status

## 修改sshd端口号

# vim /etc/ssh/sshd\_config



重启sshd服务生效

# service sshd restart

## 防火墙设置（建议最后配置，以免影响软件的安装）

脚本如下：

#!/bin/bash

# by www.jbxue.com

iptab="/sbin/iptables"

#----------------------- 清空所有规则 -----------------------------------#

$iptab -F

$iptab -X

$iptab -Z

$iptab -t nat -F

$iptab -t nat -X

$iptab -t nat -Z

#------------------- 拒绝默认规则之前开放SSH 端口（2222）----------------#

$iptab -A INPUT -p tcp --dport 2222 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 2222 -m state --state RELATED,ESTABLISHED -j ACCEPT

#----------------------- 拒绝所有的规则 ---------------------------------#

$iptab -P INPUT DROP

$iptab -P OUTPUT DROP

$iptab -P FORWARD DROP

#-------------------------- 打开回环设备 ----------------------------------#

$iptab -A INPUT -p tcp -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

$iptab -A OUTPUT -p tcp -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

$iptab -A INPUT -i lo -j ACCEPT

$iptab -A OUTPUT -o lo -j ACCEPT

#----------------------- 打开FTP服务端口（21）-------------------------------#

$iptab -A INPUT -p tcp --dport 21 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 21 -m state --state RELATED,ESTABLISHED -j ACCEPT

#--------------------------- 开放SMTP 服务端口（25） -------------------------#

$iptab -A INPUT -p tcp --dport 25 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 25 -j ACCEPT

$iptab -A INPUT -p tcp --sport 25 -j ACCEPT

$iptab -A OUTPUT -p tcp --dport 25 -j ACCEPT

#------------------------ 开放DNS解析服务端口（53） ---------------------------#

$iptab -A INPUT -p udp --dport 53 -j ACCEPT

$iptab -A OUTPUT -p udp --sport 53 -j ACCEPT

$iptab -A INPUT -p udp --sport 53 -j ACCEPT

$iptab -A OUTPUT -p udp --dport 53 -j ACCEPT

#------------------- 开放http 服务端口（80）--------------------------------#

$iptab -A INPUT -p tcp --dport 80 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 80 -m state --state RELATED,ESTABLISHED -j ACCEPT

#------------- imp3(143) imp3s(995) pop3(110) pop3s(993) port ----------------#

$iptab -A INPUT -p tcp --dport 110 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 110 -j ACCEPT

$iptab -A INPUT -p tcp --dport 143 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 143 -j ACCEPT

$iptab -A INPUT -p tcp --dport 993 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 993 -m state --state RELATED,ESTABLISHED -j ACCEPT

$iptab -A INPUT -p tcp --dport 995 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 995 -m state --state RELATED,ESTABLISHED -j ACCEPT

#--------------------开放 Mysql 服务端口（3316）----------------------------#

$iptab -A INPUT -p tcp --dport 3316 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 3316 -j ACCEPT

#---------------------开放所有tomcat所依赖端口 ----------------------------#

$iptab -A INPUT -p tcp --dport 8005 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8005 -j ACCEPT

$iptab -A INPUT -p tcp --dport 8009 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8009 -j ACCEPT

$iptab -A INPUT -p tcp --dport 8080 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8080 -m state --state RELATED,ESTABLISHED -j ACCEPT

$iptab -A INPUT -s 222.35.140.70 -p tcp --dport 8080 -j ACCEPT

$iptab -A OUTPUT -d 222.35.140.70 -p tcp --sport 8080 -j ACCEPT

$iptab -A INPUT -p tcp --dport 8081 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8081 -m state --state RELATED,ESTABLISHED -j ACCEPT

#---------------------- 开放http监听端口（8084） --------------------------#

$iptab -A INPUT -p tcp --dport 8084 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8084 -m state --state RELATED,ESTABLISHED -j ACCEPT

$iptab -A INPUT -p tcp --dport 8250 -j ACCEPT

$iptab -A OUTPUT -p tcp --sport 8250 -j ACCEPT

#----------------- 开放SSH服务端口（2222）--------------------------------#

$iptab -A INPUT -p tcp --sport 2222 -j ACCEPT

$iptab -A OUTPUT -p tcp --dport 2222 -m state --state RELATED,ESTABLISHED -j ACCEPT

将配置保存到配置文件：

# service iptables save

保存路径为/etc/sysconfig/iptables

# 数据库部署

## 规划

|  |  |
| --- | --- |
| 角色 | IP地址 |
| Mysql Master1 | 120.26.137.247/10.51.23.196 |
| Mysql Master2 | 120.26.71.71/10.252.124.43 |
| Mysql Slave1 | 121.40.166.38/10.168.6.6 |
| Mysql Slave2 | 120.26.44.153/10.175.204.158 |
| Mysql Slave3 |  |
| MHA(Master) | 120.26.127.218/10.252.82.111 |
| Atlas Proxy(Master) | 120.26.127.218/10.252.82.111 |
| MHA(backup)(不启动) | 121.40.17.62 / 10.251.233.151 |
| Atlas Proxy(backup) | 121.40.17.62 / 10.251.233.151 |

文件系统：

* / ext4
* /mydata XFS

内存：16G

## 安装

1. 解压安装文件

#tar -zxvf mysql-5.6.23.tar.gz

1. 首先安装依赖包，避免在安装过程中出现问题

#yum install -y cmake gcc gcc-c++ gcc-g77 autoconf automake zlib\* fiex\* libxml\* ncurses-devel libmcrypt\* libtool-ltdl-devel\* bison-devel libaio-devel libaio-devel

1. 创建mysql用户及用户组

#groupadd mysql

#useradd mysql -c "start mysqld's account" -d /dev/null -g mysql -s /sbin/nologin

#id mysql

#passwd mysql

1. 创建mysql的安装目录及数据库存放目录

#mkdir -p /mydata/mysql #-p选项允许一次性创建多层次的目录，而不是一次只创建单独的目录。

#mkdir /mydata/mysql/data #存放数据库数据

#mkdir /mydata/mysql/install #存放数据库二进制文件

#mkdir /mydata/mysql/log #存放数据库日志

#mkdir /mydata/mysql/sock #存放数据库套接字

#mkdir /mydata/mysql/binlog #存放二进制日志

1. 编译安装

# cd /mydata/software/mysql-5.6.23

配置

# cmake \

-DCMAKE\_INSTALL\_PREFIX=/mydata/mysql/install \

-DMYSQL\_DATADIR=/mydata/mysql/data \

-DMYSQL\_UNIX\_ADDR=/mydata/mysql/sock/mysql.sock \

-DDEFAULT\_CHARSET=utf8 \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DENABLE\_DOWNLOADS=1 \

-DMYSQL\_TCP\_PORT=3316 \

-DSYSCONFDIR=/mydata/mysql/etc \

编译安装需要20分钟左右

#make&&make install

# chown -R mysql:mysql /mydata/mysql/

# cd /mydata/mysql/install/support-files/

将配置文件拷贝到/etc下

# cp …/my.cnf /etc/my.cnf

# cp mysql.server /etc/rc.d/init.d/mysql

1. 初始化安装

# /mydata/mysql/install/scripts/mysql\_install\_db \

--defaults-file=/etc/my.cnf \

--basedir=/mydata/mysql/install \

--datadir=/mydata/mysql/data \

--user=mysql

1. 配置bin目录文件的软连接

ln -s /mydata/mysql/install/bin/\* /usr/bin/

添加系统环境变量，此步可省略

# vim /etc/profile

在最后添加如下行：

export PATH=$PATH:/mydata/mysql/install/bin

及时生效：# source /etc/profile

1. 使用mysql用户启用mysql数据库

# /mydata/mysql/install/bin/mysqld\_safe --user=mysql &

1. 检测下是否真的开启

#/mydata/mysql/install/bin/mysqladmin version

1. 去掉无用安装安装

#/mydata/mysql/install/bin/mysql\_secure\_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL

SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current

password for the root user. If you've just installed MySQL, and

you haven't set the root password yet, the password will be blank,

so you should just press enter here.

Enter current password for root (enter for none):<–初次运行直接回车

OK, successfully used password, moving on…

Setting the root password ensures that nobody can log into the MySQL

root user without the proper authorisation.

Set root password? [Y/n] <– 是否设置root用户密码，暂不设置

New password: <– 设置root用户的密码

Re-enter new password: <– 再输入一次你设置的密码

Password updated successfully!

Reloading privilege tables..

… Success!

By default, a MySQL installation has an anonymous user, allowing anyone

to log into MySQL without having to have a user account created for

them. This is intended only for testing, and to make the installation

go a bit smoother. You should remove them before moving into a

production environment.

Remove anonymous users? [Y/n] <– 是否删除匿名用户,生产环境建议删除，所以直接回车

… Success!

Normally, root should only be allowed to connect from 'localhost'. This

ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] <–是否禁止root远程登录,根据自己的需求选择Y/n并回车,建议禁止

… Success!

By default, MySQL comes with a database named 'test' that anyone can

access. This is also intended only for testing, and should be removed

before moving into a production environment.

Remove test database and access to it? [Y/n] <– 是否删除test数据库,直接回车

- Dropping test database…

… Success!

- Removing privileges on test database…

… Success!

Reloading the privilege tables will ensure that all changes made so far

will take effect immediately.

Reload privilege tables now? [Y/n] <– 是否重新加载权限表，直接回车

… Success!

Cleaning up…

All done! If you've completed all of the above steps, your MySQL

installation should now be secure.

Thanks for using MySQL!

1. 将mysql添加为系统服务并设置开机自启

#chkconfig --add mysql

#chkconfig --level 345 mysql on

1. 改变默认mysql管理员帐号

# mysql -p

mysql> use mysql;

mysql> update user set user="rkylinadmin" where user="root"; //改成不易被猜测的用户名

mysql> UPDATE user SET Password=PASSWORD('Rkylin\_web!1') where USER='rkylinadmin';

mysql> flush privileges;

1. 链接访问授权

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'rkylinadmin'@'%' IDENTIFIED BY 'Rkylin\_web!1' WITH GRANT OPTION;

mysql> FLUSH PRIVILEGES;

1. 数据库文件的安全

保证该目录不能让未经授权的用户访问后把数据库打包拷贝走了，所以要限制对该目录的访问。

#chmod -R go-rwx /mydata/mysql

## 配置

### 文件目录

安装目录：/mydata/mysql/install

日志目录：/mydata/mysql/log（error.log、slow.log）

数据目录：/mydata/mysql/data

二进制日志目录：/mydata/mysql/binlog（binlog、relaylog）

套接字文件：/mydata/mysql/sock/mysql.sock

配置文件：/etc/my.cnf

### 配置文件

/etc/my.cnf

[mysqld]

# These are commonly set, remove the # and set as required.

basedir = /mydata/mysql/install

datadir = /mydata/mysql/data

port = 3316

socket = /mydata/mysql/sock/mysql.sock

default\_storage\_engine = InnoDB

#从上需要设置为不同值

server\_id = 1

tmpdir = /mydata/mysql/tmp

# InnoDB

innodb\_data\_file\_path = ibdata1:1G:autoextend

innodb\_buffer\_pool\_size = 10G

innodb\_file\_per\_table = 1

innodb\_flush\_method = O\_DIRECT

#从上设置为0

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_io\_capacity = 1024

innodb\_max\_dirty\_pages\_pct = 50

# character-set

character-set-server=utf8

collation-server=utf8\_bin

init-connect='SET NAMES utf8'

# name-resolve

skip-host-cache

skip-name-resolve

# Adjust sizes as needed, experiment to find the optimal values.

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

query\_cache\_size = 0

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

# LOG

log\_error = /mydata/mysql/log/error.log

long\_query\_time = 1

slow-query-log

slow\_query\_log\_file = /mydata/mysql/log/slow.log

# BINLOG

event\_scheduler = ON

binlog\_format = mixed

binlog\_cache\_size = 4M

relay\_log = /mydata/mysql/binlog/relaylog

relay\_log\_index = /mydata/mysql/binlog/relaylog.index

log-bin = /mydata/mysql/binlog/binlog

log-bin-index = /mydata/mysql/binlog/binlog.index

log-slave-updates

#skip\_slave\_start

slave-skip-errors=all

expire\_logs\_days = 7

#从上设置为0

sync\_binlog=1

relay\_log\_purge = 0

master\_info\_repository = table

relay\_log\_info\_repository = table

#binlog-do-db = test

binlog-ignore-db = mysql

binlog-ignore-db = information\_schema

binlog-ignore-db = performance\_schema

replicate-ignore-db = mysql

replicate-ignore-db = information\_schema

replicate-ignore-db = performance\_schema

replicate\_wild\_ignore\_table=mysql.%

# Others

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

explicit\_defaults\_for\_timestamp = true

max\_connections = 500

max\_connect\_errors = 100000

max\_allowed\_packet = 32M

open\_files\_limit = 10240

back\_log = 600

table\_open\_cache = 2048

#rpl\_semi\_sync\_master\_enabled = 1

#rpl\_semi\_sync\_slave\_enabled = 1

#plugins根据主从进行配置

#plugin-load = semisync\_master.so

#plugin-load = semisync\_slave.so

[client]

socket = /mydata/mysql/sock/mysql.sock

port = 3316

#default-character-set=utf8

[mysql]

no-auto-rehash

default-character-set=utf8

[mysqldump]

quick

max\_allowed\_packet = 32M

[myisamchk]

key\_buffer\_size = 256M

sort\_buffer\_size = 256M

read\_buffer = 2M

write\_buffer = 2M

### 主从复制

在master1上：

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.252.124.43` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.168.6.6` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.175.204.158` identified by'Slaveadmin!1';

在master2上：

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.51.23.196` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.168.6.6` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.175.204.158` identified by'Slaveadmin!1';

在slave1上：

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.51.23.196` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.252.124.43` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.175.204.158` identified by'Slaveadmin!1';

在slave2上：

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.51.23.196` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.252.124.43` identified by'Slaveadmin!1';

mysql>grant replication slave on \*.\* to `slaveadmin`@`10.168.6.6` identified by'Slaveadmin!1';

在master1上：

mysql>show master status;

根据显示的状态在master2、slave1、slave2上分别执行：

mysql> CHANGE MASTER TO MASTER\_HOST='10.51.23.196',MASTER\_USER='slaveadmin',MASTER\_PASSWORD='Slaveadmin!1',MASTER\_PORT=3316,MASTER\_LOG\_FILE='binlog.000001',MASTER\_LOG\_POS=120;

mysql>start slave;

mysql>show slave status\G;

注意查看：

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

以上这两个参数的值为Yes，即说明配置成功！

配置好了以后主库上会有slave过来的进程：

mysql>show processlist;

### 半同步复制

#### 工作原理

* 从库连接主库时指明时从库是否有半同步能力
* 主库启动半同步复制且至少有一个半同步从库，主库线程执行事务提交之后将会阻塞直到任一半同步从库确认接收到该事务的所有事件或者超时
* 从库确认接收到事务的所有事件之后写入到中继日志并刷新到磁盘
* 如果超时没有任何从库确认事务，主库恢复到异步复制，保障业务的正常使用，直到一台从库追赶上之后，继续切换到半同步模式
* 半同步复制必须主从两端同时启用，如任意一端禁止将使用异步复制

当主库被阻塞时（已经提交的事务等待从从库确认），不会向执行事务的session返回信息；当主库阻塞结束后，返回session执行结果。此时，主库事务已提交，至少一台从库确认接收到事务。如果主备网络故障或者从库崩溃，主库在事务提交后等待10秒（rpl\_semi\_sync\_master\_timeout默认值）后，主从复制将自动降级为异步模式。

半同步复制确实有一定的性能影响，因为需要等待的从库确认。这是对增加数据的完整性的折衷。

半同步复制是通过plugin实现的，master和slave使用不同的plugin。默认情况下没有安装该plugin，插件位于CMAKE\_INSTALL\_PREFIX/lib/plugin/。安装完插件之后，还需要手动设置系统参数以开启半同步复制模式。

#### 启用与关闭

1. 查看插件路径

mysql> show variables like "%plugin\_dir%";

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

| Variable\_name | Value |

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

| plugin\_dir | /mydata/mysql/install/lib/plugin/ |

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

1. 查看半同步复制插件

# ls -l /mydata/mysql/install/lib/plugin/

total 1048

-rwxr-xr-x 1 mysql mysql 16445 Aug 24 14:46 adt\_null.so

-rwxr-xr-x 1 mysql mysql 25467 Aug 24 14:46 auth.so

-rwxr-xr-x 1 mysql mysql 12532 Aug 24 14:46 auth\_socket.so

-rwxr-xr-x 1 mysql mysql 24576 Aug 24 14:46 auth\_test\_plugin.so

-rw-r--r-- 1 mysql mysql 227 Jan 19 2015 daemon\_example.ini

drwxr-xr-x 2 mysql mysql 4096 Aug 24 15:40 debug

-rwxr-xr-x 1 mysql mysql 39179 Aug 24 14:46 libdaemon\_example.so

-rwxr-xr-x 1 mysql mysql 17827 Aug 24 14:46 mypluglib.so

-rwxr-xr-x 1 mysql mysql 18087 Aug 24 14:46 qa\_auth\_client.so

-rwxr-xr-x 1 mysql mysql 23870 Aug 24 14:46 qa\_auth\_interface.so

-rwxr-xr-x 1 mysql mysql 12934 Aug 24 14:46 qa\_auth\_server.so

-rwxr-xr-x 1 mysql mysql 415292 Aug 24 14:46 semisync\_master.so

-rwxr-xr-x 1 mysql mysql 248871 Aug 24 14:46 semisync\_slave.so

-rwxr-xr-x 1 mysql mysql 168672 Aug 24 14:46 validate\_password.so

* semisync\_master.so     #用于Master服务器安装的半同步插件
* semisync\_slave.so      #用于Slave服务器安装的半同步插件

1. master1与master2服务器分别安装半同步插件

* 在master1服务器安装半同步插件

mysql> install plugin rpl\_semi\_sync\_master soname 'semisync\_master.so'; #安装Master半同步插件

mysql> set global rpl\_semi\_sync\_master\_enabled = 1;    #开启Master半同步功能

mysql> set global rpl\_semi\_sync\_master\_timeout = 10000;

* 在master2服务器安装半同步插件

mysql> install plugin rpl\_semi\_sync\_slave soname 'semisync\_slave.so'; #安装Slave半同步插件

mysql> set global rpl\_semi\_sync\_slave\_enabled = 1;      #开启Slave半同步功能

mysql> stop slave io\_thread;start slave io\_thread;      #重启IO线程生效

也可以修改配置文件/etc/my.cnf使半同步永久生效，改完之后记得重启下mysqld服务

* Master1：

# vim /etc/my.cnf

rpl\_semi\_sync\_master\_enabled=1

rpl\_semi\_sync\_master\_timeout=10000

* Master2：

# vim /etc/my.cnf

rpl\_semi\_sync\_slave\_enabled=1

1. 查看半同步开启状态

* 在Master1服务器上查看

mysql> show global status like 'rpl\_semi%';

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

| Variable\_name                              | Value |

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

| Rpl\_semi\_sync\_master\_clients               | 1     | # 有多少slave设置了半同步模式

| Rpl\_semi\_sync\_master\_net\_avg\_wait\_time     | 0     | # 事务提交后，等待备库响应的平均时间

| Rpl\_semi\_sync\_master\_net\_wait\_time         | 0     | # 网络响应等待的总时间

| Rpl\_semi\_sync\_master\_net\_waits             | 0     | # 网络等待总次数

| Rpl\_semi\_sync\_master\_no\_times              | 0     | # 一共有几次从Semi-sync跌回普通状态

| Rpl\_semi\_sync\_master\_no\_tx                 | 0     | # 备库未及时响应的事务次数

| Rpl\_semi\_sync\_master\_status                | ON    | #表示当前是异步模式还是半同步模式，on为半同步

| Rpl\_semi\_sync\_master\_timefunc\_failures     | 0     | # 时间函数未正常工作的次数

| Rpl\_semi\_sync\_master\_tx\_avg\_wait\_time      | 0     | # 开启Semi-sync，事务返回需要等待的平均时间

| Rpl\_semi\_sync\_master\_tx\_wait\_time          | 0     | # 事务等待备库响应的总时间

| Rpl\_semi\_sync\_master\_tx\_waits              | 0     | # 事务等待备库响应的总次数

| Rpl\_semi\_sync\_master\_wait\_pos\_backtraverse | 0     | # 改变当前等待事务记录最小二进制日志的次数

| Rpl\_semi\_sync\_master\_wait\_sessions         | 0     | # 当前有几个线程在等待备库响应

| Rpl\_semi\_sync\_master\_yes\_tx                | 0     | # 备库成功响应的事务次数

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

mysql> show global variables like '%rpl%';

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

| Variable\_name                      | Value |

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

| rpl\_recovery\_rank                  | 0     |

| rpl\_semi\_sync\_master\_enabled       | ON    | # 是否启动半同步复制，默认关闭

| rpl\_semi\_sync\_master\_timeout       | 10000  | # 等待从服务器告诉接受到的超时时间，如果时间到了，还没接受到，自动降级为异步

| rpl\_semi\_sync\_master\_trace\_level   | 32    | # 运行级别

| rpl\_semi\_sync\_master\_wait\_no\_slave | ON    | # 没有slave的时候是否也需要等待，默认为也需要等待

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

* 在master2服务器上查看

mysql> show global status like 'rpl\_semi%';

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

| Variable\_name              | Value |

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

| Rpl\_semi\_sync\_slave\_status | ON    |         #已经为开启状态

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

mysql> show global variables like '%rpl%';

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

| Variable\_name                   | Value |

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

| rpl\_recovery\_rank               | 0     |

| rpl\_semi\_sync\_slave\_enabled     | ON    |    # 是否启动半同步复制，默认关闭

| rpl\_semi\_sync\_slave\_trace\_level | 32    |    # 运行级别

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

1. 查看Slave线程是否启动

# mysql -e 'show slave status\G;' | grep Running

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

PS：这两项必须为”Yes”,如果是”No”说明启动失败

1. 取消半同步复制的插件

* 在Master1服务器上取消半同步复制插件

mysql > uninstall plugin rpl\_semi\_sync\_master;

mysql > show status like "%semi%"

* 在Master2服务器上取消半同步复制插件

mysql > uninstall plugin rpl\_semi\_sync\_slave;

mysql > show status like "%semi%"

### 修改复制信息表存储引擎

alter table slave\_master\_info engine=innodb;

alter table slave\_relay\_log\_info engine=innodb;

alter table slave\_worker\_info engine=innodb;

### 权限管理

grant select,update,insert,delete,execute,show view on `account`.\* to `account`@`10.252.82.111` identified by 'Account\_capital!1';

grant select,update,insert,delete,execute,show view on `account`.\* to `account`@`10.251.233.151` identified by 'Account\_capital!1';

grant select on mysql.proc to `account`@`10.252.82.111`;

grant select on mysql.proc to `account`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `order`.\* to `order`@`10.252.82.111` identified by 'Order\_capital!1';

grant select,update,insert,delete,execute,show view on `order`.\* to `order`@`10.251.233.151` identified by 'Order\_capital!1';

grant select on mysql.proc to `order`@`10.252.82.111`;

grant select on mysql.proc to `order`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `checkstand`.\* to `checkstand`@`10.252.82.111` identified by 'Checkstand\_capital!1';

grant select,update,insert,delete,execute,show view on `checkstand`.\* to `checkstand`@`10.251.233.151` identified by 'Checkstand\_capital!1';

grant select on mysql.proc to `checkstand`@`10.252.82.111`;

grant select on mysql.proc to `checkstand`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `payment`.\* to `payment`@`10.252.82.111` identified by 'Payment\_capital!1';

grant select,update,insert,delete,execute,show view on `payment`.\* to `payment`@`10.251.233.151` identified by 'Payment\_capital!1';

grant select on mysql.proc to `payment`@`10.252.82.111`;

grant select on mysql.proc to `payment`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `multi\_gate`.\* to `multi\_gate`@`10.252.82.111` identified by 'Multi\_gate\_capital!1';

grant select,update,insert,delete,execute,show view on `multi\_gate`.\* to `multi\_gate`@`10.251.233.151` identified by 'Multi\_gate\_capital!1';

grant select on mysql.proc to `multi\_gate`@`10.252.82.111`;

grant select on mysql.proc to `multi\_gate`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `settle`.\* to `settle`@`10.252.82.111` identified by 'Settle\_capital!1';

grant select,update,insert,delete,execute,show view on `settle`.\* to `settle`@`10.251.233.151` identified by 'Settle\_capital!1';

grant select on mysql.proc to `settle`@`10.252.82.111`;

grant select on mysql.proc to `settle`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `user`.\* to `user`@`10.252.82.111` identified by 'User\_capital!1';

grant select,update,insert,delete,execute,show view on `user`.\* to `user`@`10.251.233.151` identified by 'User\_capital!1';

grant select on mysql.proc to `user`@`10.252.82.111`;

grant select on mysql.proc to `user`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `operate`.\* to `operate`@`10.252.82.111` identified by 'Operate\_capital!1';

grant select,update,insert,delete,execute,show view on `operate`.\* to `operate`@`10.251.233.151` identified by 'Operate\_capital!1';

grant select on mysql.proc to `operate`@`10.252.82.111`;

grant select on mysql.proc to `operate`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `logsystem`.\* to `logsystem`@`10.252.82.111` identified by 'Logsystem\_capital!1';

grant select,update,insert,delete,execute,show view on `logsystem`.\* to `logsystem`@`10.251.233.151` identified by 'Logsystem\_capital!1';

grant select on mysql.proc to `logsystem`@`10.252.82.111`;

grant select on mysql.proc to `logsystem`@`10.251.233.151`;

grant select,update,insert,delete,execute,show view on `monitor`.\* to `monitor`@`10.252.82.111` identified by 'Monitor\_capital!1';

grant select,update,insert,delete,execute,show view on `monitor`.\* to `monitor`@`10.251.233.151` identified by 'Monitor\_capital!1';

grant select,show view on \*.\* to 'monitor'@'10.252.82.111' identified by 'Monitor\_capital!1';

grant select,show view on \*.\* to 'monitor'@'10.251.233.151' identified by 'Monitor\_capital!1';

grant select,show view on \*.\* to 'queryuser'@'%' identified by 'Query!1';

### 安装percona-toolkit

# yum install -y percona-toolkit-2.2.15-1.noarch.rpm

# MHA部署

## 工具说明

MHA软件由两部分组成，Manager工具包和Node工具包，具体的说明如下。

Manager工具包主要包括以下几个工具：

* masterha\_check\_ssh：检查MHA的SSH配置状况
* masterha\_check\_repl：检查MySQL复制状况
* masterha\_manger：启动MHA
* masterha\_check\_status：检测当前MHA运行状态
* masterha\_master\_monitor：检测master是否宕机
* masterha\_master\_switch：控制故障转移（自动或者手动）
* masterha\_conf\_host：添加或删除配置的server信息

Node工具包（这些工具通常由MHA Manager的脚本触发，无需人为操作）主要包括以下几个工具：

* save\_binary\_logs：保存和复制master的二进制日志
* apply\_diff\_relay\_logs ：识别差异的中继日志事件并将其差异的事件应用于其他的slave
* filter\_mysqlbinlog：去除不必要的ROLLBACK事件（MHA已不再使用这个工具）
* purge\_relay\_logs：清除中继日志（不会阻塞SQL线程）

## 安装MHA Node和MHA Manager

其中master对外提供写服务，slave提供相关的读服务，一旦master宕机，将会把备选master提升为新的master，slave指向新的master。

1. 安装依赖包(所有节点上)

# yum install -y cpan perl-DBD-MySQL perl-Config-Tiny perl-Log-Dispatch perl-Parallel-ForkManager perl-Time-HiRes

1. 安装MHA node软件（所有节点上）
2. 解压文件

# tar -zxvf mha4mysql-node-0.56.tar.gz

# cd mha4mysql-node-0.56

1. 编译及安装

# perl Makefile.PL

# make && make install

安装完成后会在/usr/local/bin目录下生成以下脚本文件：

# ll /usr/local/bin

total 44

-r-xr-xr-x 1 root root 16367 Aug 4 17:34 apply\_diff\_relay\_logs

-r-xr-xr-x 1 root root 4807 Aug 4 17:34 filter\_mysqlbinlog

-r-xr-xr-x 1 root root 8261 Aug 4 17:34 purge\_relay\_logs

-r-xr-xr-x 1 root root 7525 Aug 4 17:34 save\_binary\_logs

1. 安装MHA Manager软件（120.26.127.218）
2. 解压文件

# tar -zxvf mha4mysql-manager-0.56.tar.gz

# cd mha4mysql-manager-0.56

注：在MHA Manager的主机也是需要安装MHA Node。

1. 编译及安装

# perl Makefile.PL

# make && make install

安装完成后会在/usr/local/bin目录下面生成以下脚本文件：

# ll /usr/local/bin

total 84

-r-xr-xr-x 1 root root 16367 Aug 4 17:34 apply\_diff\_relay\_logs

-r-xr-xr-x 1 root root 4807 Aug 4 17:34 filter\_mysqlbinlog

-r-xr-xr-x 1 root root 1995 Aug 4 17:48 masterha\_check\_repl

-r-xr-xr-x 1 root root 1779 Aug 4 17:48 masterha\_check\_ssh

-r-xr-xr-x 1 root root 1865 Aug 4 17:48 masterha\_check\_status

-r-xr-xr-x 1 root root 3201 Aug 4 17:48 masterha\_conf\_host

-r-xr-xr-x 1 root root 2517 Aug 4 17:48 masterha\_manager

-r-xr-xr-x 1 root root 2165 Aug 4 17:48 masterha\_master\_monitor

-r-xr-xr-x 1 root root 2373 Aug 4 17:48 masterha\_master\_switch

-r-xr-xr-x 1 root root 5171 Aug 4 17:48 masterha\_secondary\_check

-r-xr-xr-x 1 root root 1739 Aug 4 17:48 masterha\_stop

-r-xr-xr-x 1 root root 8261 Aug 4 17:34 purge\_relay\_logs

-r-xr-xr-x 1 root root 7525 Aug 4 17:34 save\_binary\_logs

1. 复制相关脚本到/usr/local/bin

# cd /mydata/software/mha4mysql-manager-0.56/samples/scripts

# ll

total 32

-rwxr-xr-x 1 root root 3443 Jun 8 2015 master\_ip\_failover #自动切换时vip管理的脚本，不是必须，如果我们使用keepalived的，我们可以自己编写脚本完成对vip的管理，比如监控mysql，如果mysql异常，我们停止keepalived就行，这样vip就会自动漂移

-rwxr-xr-x 1 root root 9186 Jun 8 2015 master\_ip\_online\_change #在线切换时vip的管理，不是必须，同样可以可以自行编写简单的shell完成

-rwxr-xr-x 1 root root 11867 Jun 8 2015 power\_manager #故障发生后关闭主机的脚本，不是必须

-rwxr-xr-x 1 root root 1360 Jun 8 2015 send\_report #因故障切换后发送报警的脚本，不是必须，可自行编写简单的shell完成。

# cp \* /usr/local/bin/

## 配置SSH免密码登录（key模式）

1. 准备工作

角色：

|  |  |
| --- | --- |
| IP地址 | 主机名 |
| 10.51.23.196 | iZ233yjy5hvZ |
| 10.252.124.43 | iZ23vv1joz4Z |
| 10.168.6.6 | iZ23k0d7o6qZ |
| 10.252.82.111 | iZ23zc5fzqaZ |
| 10.251.233.151 | iZ23d18psdnZ |

Linux下修改主机名的完整步骤如下：

第一步 : # hostname Server1

第二步：修改/etc/sysconfig/network中的hostname

第三步：修改/etc/hosts文件

注：将其他三个服务器的IP及名称补进/etc/hosts。

1. 用ssh-keygen创建公钥
2. 在所有服务器上执行下面的命令：

# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

aa:b9:a9:4d:88:0a:cb:96:a1:fb:5e:e2:b3:b8:ad:bd root@iZ233yjy5hvZ

The key's randomart image is:

+--[ RSA 2048]----+

| |

| |

| |

| |

| S |

| o . . |

|+ = o . |

|=O.= + |

|OBE=\*. |

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

然后，会提示创建.ssh/id\_rsa、id\_rsa.pub的文件，其中第一个为密钥，第二个为公钥。过程中会要求输入密码，为了ssh访问过程无须密码，可以直接回车。使四台机子中的任何两台之间可以免密码登录。

1. 补充说明：

ssh-keygen:生成秘钥

其中：

* -t指定算法
* -f 指定生成秘钥路径
* -N 指定密码

1. 查看钥匙

# ll /root/.ssh

total 12

-rw------- 1 root root 1675 Aug 5 11:15 id\_rsa

-rw-r--r-- 1 root root 399 Aug 5 11:15 id\_rsa.pub

-rw-r--r-- 1 root root 409 Aug 3 17:49 known\_hosts

可以发现 ssh目录下的两枚钥匙。

1. 用 ssh-copy-id 把公钥复制到远程主机上，相互之间都要进行复制。

# ssh-copy-id 其他主机IP

ligh@remote-host‘s password:

Now try logging into the machine, with ―ssh ?remote-host‘‖, and check in:

.ssh/authorized\_keys to make sure we haven‘t added extra keys that you weren‘t expecting.

注: ssh-copy-id 把密钥追加到远程主机的.ssh/authorized\_key上。

1. 验证使用SSH IP地址的方式无密码访问

#ssh 10.51.23.196

#ssh 10.252.124.43

#ssh 10.168.6.6

#ssh 10.252.82.111

#ssh 10.251.233.151

1. 验证使用SSH主机名的方式无密码访问

我们将所有服务器的IP地址解析为主机名

#cat /etc/hosts

127.0.0.1 localhost

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

10.51.23.196 iZ233yjy5hvZ

10.252.124.43 iZ23vv1joz4Z

10.168.6.6 iZ23k0d7o6qZ

10.252.82.111 iZ23zc5fzqaZ

10.251.233.151 iZ23d18psdnZ

使用ssh尝试登陆：

#ssh iZ233yjy5hvZ

#ssh iZ23vv1joz4Z

#ssh iZ23k0d7o6qZ

#ssh iZ23zc5fzqaZ

#ssh iZ23d18psdnZ

1. 补充说明

* 文件和目录的权限千万别设置成chmod 777.这个权限太大了，不安全，数字签名也不支持。
* 生成的rsa/dsa签名的公钥是给对方机器使用的。这个公钥内容是拷贝到其他主机的authorized\_keys文件中。
* linux之间的访问直接ssh机器ip或者主机名。
* 某个机器生成自己的RSA或者DSA的数字签名，将公钥给目标机器，然后目标机器接收后设定相关权限（公钥和authorized\_keys权限），这个目标机就能被生成数字签名的机器无密码访问了。

## 配置MHA

1. 在中间件服务器(10.51.23.196)上创建MHA的工作目录，并且创建相关配置文件（在软件包解压后的目录里面有样例配置文件）。

# mkdir -p /mydata/masterha

# cp …/app1.cnf /etc/

# cat app1.cnf

[server default]

manager\_log=/mydata/log/manager.log

manager\_workdir=/mydata/masterha/app1

master\_binlog\_dir=/mydata/mysql/binlog

master\_ip\_online\_change\_script=/usr/local/bin/master\_ip\_online\_change

user="rkylinadmin"

password="Rkylin\_web!1"

ping\_interval=1

remote\_workdir=/mydata/masterha/binlog

repl\_password="Slaveadmin!1"

repl\_user="slaveadmin"

report\_script=/usr/local/bin/send\_report

ssh\_user=root

#master\_ip\_failover\_script=/usr/local/bin/master\_ip\_failover

#secondary\_check\_script=/usr/local/bin/masterha\_secondary\_check -s B-dev18 -s B-dev10 --user=root --master\_host=B-dev10 --master\_ip=120.26.137.247 --master\_port=3316

[server1]

candidate\_master=1

check\_repl\_delay=0

hostname=10.51.23.196

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

[server2]

candidate\_master=1

check\_repl\_delay=0

hostname=10.252.124.43

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

[server3]

#candidate\_master=1

#check\_repl\_delay=0

hostname=10.168.6.6

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

参数说明：

[server default]

manager\_log=/mydata/log/manager.log //设置manager的工作目录

manager\_workdir=/mydata/masterha/app1 //设置manager的日志

master\_binlog\_dir=/mydata/mysql/binlog //设置master 保存binlog的位置，以便MHA可以找到master的日志，我这里的也就是mysql的数据目录

#master\_ip\_failover\_script= /usr/local/bin/master\_ip\_failover //设置自动failover时候的切换脚本,这里先注释掉后面集成keepalived后再开启

master\_ip\_online\_change\_script= /usr/local/bin/master\_ip\_online\_change //设置手动切换时候的切换脚本

user=root //设置监控用户root

password="Rkylin\_web!1" //设置mysql中root用户的密码

ping\_interval=1 //设置监控主库，发送ping包的时间间隔，默认是3秒，尝试三次没有回应的时候自动进行railover

remote\_workdir=/mydata/masterha/binlog //设置远端mysql在发生切换时binlog的保存位置

repl\_password="Slaveadmin!1" //设置复制用户的密码

repl\_user="slaveadmin" //设置复制环境中的复制用户名

report\_script=/usr/local/send\_report //设置发生切换后发送的报警的脚本

#secondary\_check\_script= /usr/local/bin/masterha\_secondary\_check -s B-dev18 -s B-dev10 --user=root --master\_host=B-dev10 --master\_ip=192.168.3.110 --master\_port=3306 //一旦MHA到B-dev10的监控之间出现问题，MHA Manager将会尝试从B-dev18登录到B-dev10

#shutdown\_script="" //设置故障发生后关闭故障主机脚本（该脚本的主要作用是关闭主机防止发生脑裂,这里没有使用）

ssh\_user=root //设置ssh的登录用户名

[server]

candidate\_master=1 //设置为候选master，如果设置该参数以后，发生主从切换以后将会将此从库提升为主库，即使这个主库不是集群中事件最新的slave

check\_repl\_delay=0 //默认情况下如果一个slave落后master 100M的relay logs的话，MHA将不会选择该slave作为一个新的master，因为对于这个slave的恢复需要花费很长时间，通过设置check\_repl\_delay=0,MHA触发切换在选择一个新的master的时候将会忽略复制延时，这个参数对于设置了candidate\_master=1的主机非常有用，因为这个候选主在切换的过程中一定是新的master

no\_master=1 // 通过设置no\_master=1可以让某一个节点永远不成为新的主节点

## 设置relay log的清除方式（在每个slave节点上）：

1. 脚本说明

MHA在发生切换的过程中，从库的恢复过程中依赖于relay log的相关信息，所以这里要将relay log的自动清除设置为OFF，采用手动清除relay log的方式。在默认情况下，从服务器上的中继日志会在SQL线程执行完毕后被自动删除。但是在MHA环境中，这些中继日志在恢复其他从服务器时可能会被用到，因此需要禁用中继日志的自动删除功能。定期清除中继日志需要考虑到复制延时的问题。在ext3的文件系统下，删除大的文件需要一定的时间，会导致严重的复制延时。为了避免复制延时，需要暂时为中继日志创建硬链接，因为在linux系统中通过硬链接删除大文件速度会很快。（在mysql数据库中，删除大表时，通常也采用建立硬链接的方式）

MHA节点中包含了pure\_relay\_logs命令工具，它可以为中继日志创建硬链接，执行SET GLOBAL relay\_log\_purge=1,等待几秒钟以便SQL线程切换到新的中继日志，再执行SET GLOBAL relay\_log\_purge=0。

pure\_relay\_logs脚本参数如下所示：

* --user mysql 用户名
* --password mysql 密码
* --port 端口号
* --workdir 指定创建relay log的硬链接的位置，默认是/var/tmp，由于系统不同分区创建硬链接文件会失败，故需要执行硬链接具体位置，成功执行脚本后，硬链接的中继日志文件被删除
* --disable\_relay\_log\_purge 默认情况下，如果relay\_log\_purge=1，脚本会什么都不清理，自动退出，通过设定这个参数，当relay\_log\_purge=1的情况下会将relay\_log\_purge设置为0。清理relay log之后，最后将参数设置为OFF

1. 设置定期清理relay脚本（在每个slave节点上）

# vi purge\_relay\_log.sh

#!/bin/bash

user="rkylinadmin"

passwd="Rkylin\_web!1"

port=3316

log\_dir='/mydata/masterha/log'

work\_dir='/mydata/mysql'

purge='/usr/local/bin/purge\_relay\_logs'

if [ ! -d $log\_dir ]

then

mkdir $log\_dir -p

fi

$purge --user=$user --password=$passwd --disable\_relay\_log\_purge --port=$port --workdir=$work\_dir >> $log\_dir/purge\_relay\_logs.log 2>&1

## 添加到crontab定期执行

1. 添加定时任务

#crontab -e

#每周一23点清理relay脚本

0 23 \* \* 1 /mydata/scripts/purge\_relay\_log.sh >/mydata/scripts/purge\_relay\_log.log 2>&1

1. crond介绍

vixie-cron 软件包是 cron 的主程序；

crontabs 软件包是用来安装、卸装、或列举用来驱动 cron 守护进程的表格的程序。

cron 是 linux 的内置服务，但它不自动起来，可以用以下的方法启动、关闭这个服务：

service crond start //启动服务

service crond stop //关闭服务

service crond restart //重启服务

service crond reload //重新载入配置

service crond status //查看crontab服务状态

在CentOS系统中加入开机自动启动: chkconfig --level 345 crond on

cron 的主配置文件是 /etc/crontab，它包括下面几行：

SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root

HOME=/

# run-parts

01 \* \* \* \* root run-parts /etc/cron.hourly

02 4 \* \* \* root run-parts /etc/cron.daily

22 4 \* \* 0 root run-parts /etc/cron.weekly

42 4 1 \* \* root run-parts /etc/cron.monthly

前四行是用来配置 cron 任务运行环境的变量。

SHELL 变量的值告诉系统要使用哪个 shell 环境（在这个例子里是 bash shell）；

PATH 变量定义用来执行命令的路径。

cron 任务的输出被邮寄给 MAILTO 变量定义的用户名。

如果 MAILTO 变量被定义为空白字符串（MAILTO=""），电子邮件就不会被寄出。

HOME 变量可以用来设置在执行命令或脚本时使用的主目录。

限制对 cron 的使用：

/etc/cron.allow和/etc/cron.deny 文件被用来限制对 cron 的使用。

这两个使用控制文件的格式都是每行一个用户。

两个文件都不允许空格。

如果使用控制文件被修改了，cron 守护进程（crond）不必被重启。

使用控制文件在每次用户添加或删除一项 cron 任务时都会被读取。

无论使用控制文件中的规定如何，root 都总是可以使用 cron。

如果 cron.allow 文件存在，只有其中列出的用户才被允许使用 cron，并且 cron.deny 文件会被忽略。

如果 cron.allow 文件不存在，所有在 cron.deny 中列出的用户都被禁止使用 cron。

crontab 命令：

功能：设置计时器。

语法：crontab[-u <用户名称>][配置文件] 或 crontab [-u <用户名称>][-elr]

解释：cron是一个常驻服务，它提供计时器的功能，让用户在特定的时间得以执行预设的指令或程序。只要用户会编辑计时器的配置文件，就可以使 用计时器的功能。其配置文件格式如下：Minute Hour Day Month DayOFWeek Command

参数：

-e　编辑该用户的计时器设置。

-l　列出该用户的计时器设置。

-r　删除该用户的计时器设置。

-u<用户名称> 　指定要设定计时器的用户名称。

格式:

\* \*　 \*　 \*　 \*　　command

分　时　日　月　周　 命令

第1列表示分钟0～59 每分钟用\*或者 \*/1表示

第2列表示小时0～23（0表示0点）

第3列表示日期1～31

第4列表示月份1～12

第5列标识号星期0～6（0表示星期天）

第6列要运行的命令

## 修改send\_report脚本

#vim /usr/local/bin/send\_report

在最后添加如下内容：

# Do whatever you want here

system("/mydata/scripts/change\_master.sh");

exit 0;

change\_master.sh内容如下：

#!/bin/bash

#mysql -h 10.252.82.111 -P 3315 -u admin -p'admin' -e 'remove master 10.51.23.196;add master 10.252.124.43:3316;save config

MYSQL='mysql -h10.252.82.111 -P3315'

USER='admin'

PASSWORD='admin'

NEW\_MASTER='10.51.23.196:3316'

#NEW\_MASTER='10.252.124.43:3316'

backend\_ndx=`$MYSQL -u $USER -p$PASSWORD -e "select \* from backends;"|grep down|egrep -v backend\_ndx|awk '{print $1}'`

OLD\_MASTER=`$MYSQL -u $USER -p$PASSWORD -e "select \* from backends;"|grep down|egrep -v backend\_ndx|awk '{print $2}'`

echo $backend\_ndx

if [$backend\_bdx = " "]

then

echo "Master is not down"

else

$MYSQL -u $USER -p$PASSWORD -e"remove backend $backend\_ndx;add master $NEW\_MASTER;save config; "

#backend\_ndx\_1=`$MYSQL -u $USER -p$PASSWORD -e "select \* from backends;"|grep ro|grep $NEW\_MASTER|egrep -v backend\_ndx|awk '{print $1}'`

#$MYSQL -u $USER -p$PASSWORD -e"remove backend $backend\_ndx\_1;save config; "

/usr/local/bin/sendEmail -f rkylin2015@yeah.net -s smtp.yeah.net -u "$OLD\_MASTER is down!" -xu rkylin2015@yeah.net -xp 'othnzrexmapxtbdl' -t huchenglong@rongcapital.cn luzhidong@rongcapital.cn kangjian@rongcapital.cn liujingjing@rongcapital.cn -m "$OLD\_MASTER is down! $NEW\_MASTER set to rw!"

fi

## 检查SSH配置（MHA Manger层）

检查MHA Manger到所有MHA Node的SSH连接状态：

# masterha\_check\_ssh --conf=/etc/app1.cnf

Thu Aug 6 11:59:21 2015 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Thu Aug 6 11:59:21 2015 - [info] Reading application default configuration from /mydata/masterha/app1.cnf..

Thu Aug 6 11:59:21 2015 - [info] Reading server configuration from /mydata/masterha/app1.cnf..

Thu Aug 6 11:59:21 2015 - [info] Starting SSH connection tests..

Thu Aug 6 11:59:22 2015 - [debug]

Thu Aug 6 11:59:21 2015 - [debug] Connecting via SSH from root@120.26.137.247(120.26.137.247:22) to root@120.26.71.71(120.26.71.71:22)..

Thu Aug 6 11:59:22 2015 - [debug] ok.

Thu Aug 6 11:59:22 2015 - [debug] Connecting via SSH from root@120.26.137.247(120.26.137.247:22) to root@121.40.166.38(121.40.166.38:22)..

Thu Aug 6 11:59:22 2015 - [debug] ok.

Thu Aug 6 11:59:22 2015 - [debug]

Thu Aug 6 11:59:22 2015 - [debug] Connecting via SSH from root@120.26.71.71(120.26.71.71:22) to root@120.26.137.247(120.26.137.247:22)..

Thu Aug 6 11:59:22 2015 - [debug] ok.

Thu Aug 6 11:59:22 2015 - [debug] Connecting via SSH from root@120.26.71.71(120.26.71.71:22) to root@121.40.166.38(121.40.166.38:22)..

Thu Aug 6 11:59:22 2015 - [debug] ok.

Thu Aug 6 11:59:23 2015 - [debug]

Thu Aug 6 11:59:22 2015 - [debug] Connecting via SSH from root@121.40.166.38(121.40.166.38:22) to root@120.26.137.247(120.26.137.247:22)..

Thu Aug 6 11:59:23 2015 - [debug] ok.

Thu Aug 6 11:59:23 2015 - [debug] Connecting via SSH from root@121.40.166.38(121.40.166.38:22) to root@120.26.71.71(120.26.71.71:22)..

Thu Aug 6 11:59:23 2015 - [debug] ok.

Thu Aug 6 11:59:23 2015 - [info] All SSH connection tests passed successfully.

可以看见各个节点ssh验证都是ok的。

## 检查整个复制环境状况

通过masterha\_check\_repl脚本查看整个集群的状态

#masterha\_check\_repl --conf=/etc/app1.cnf

Thu Aug 6 13:25:12 2015 - [info] Reading default configuration from /etc/masterha\_default.cnf..

Thu Aug 6 13:25:12 2015 - [info] Reading application default configuration from /mydata/masterha/app1.cnf..

Thu Aug 6 13:25:12 2015 - [info] Reading server configuration from /mydata/masterha/app1.cnf..

Thu Aug 6 13:25:12 2015 - [info] MHA::MasterMonitor version 0.56.

Thu Aug 6 13:25:12 2015 - [info] GTID failover mode = 0

Thu Aug 6 13:25:12 2015 - [info] Dead Servers:

Thu Aug 6 13:25:12 2015 - [info] Alive Servers:

Thu Aug 6 13:25:12 2015 - [info] 120.26.137.247(120.26.137.247:3316)

Thu Aug 6 13:25:12 2015 - [info] 120.26.71.71(120.26.71.71:3316)

Thu Aug 6 13:25:12 2015 - [info] 121.40.166.38(121.40.166.38:3316)

Thu Aug 6 13:25:12 2015 - [info] Alive Slaves:

Thu Aug 6 13:25:12 2015 - [info] 120.26.71.71(120.26.71.71:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Thu Aug 6 13:25:12 2015 - [info] Replicating from 120.26.137.247(120.26.137.247:3316)

Thu Aug 6 13:25:12 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Thu Aug 6 13:25:12 2015 - [info] 121.40.166.38(121.40.166.38:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Thu Aug 6 13:25:12 2015 - [info] Replicating from 120.26.137.247(120.26.137.247:3316)

Thu Aug 6 13:25:12 2015 - [info] Current Alive Master: 120.26.137.247(120.26.137.247:3316)

Thu Aug 6 13:25:12 2015 - [info] Checking slave configurations..

Thu Aug 6 13:25:12 2015 - [info] read\_only=1 is not set on slave 120.26.71.71(120.26.71.71:3316).

Thu Aug 6 13:25:12 2015 - [info] read\_only=1 is not set on slave 121.40.166.38(121.40.166.38:3316).

Thu Aug 6 13:25:12 2015 - [info] Checking replication filtering settings..

Thu Aug 6 13:25:12 2015 - [info] binlog\_do\_db= , binlog\_ignore\_db= information\_schema,mysql,performance\_schema

Thu Aug 6 13:25:12 2015 - [info] Replication filtering check ok.

Thu Aug 6 13:25:12 2015 - [info] GTID (with auto-pos) is not supported

Thu Aug 6 13:25:12 2015 - [info] Starting SSH connection tests..

Thu Aug 6 13:25:14 2015 - [info] All SSH connection tests passed successfully.

Thu Aug 6 13:25:14 2015 - [info] Checking MHA Node version..

Thu Aug 6 13:25:14 2015 - [info] Version check ok.

Thu Aug 6 13:25:14 2015 - [info] Checking SSH publickey authentication settings on the current master..

Thu Aug 6 13:25:14 2015 - [info] HealthCheck: SSH to 120.26.137.247 is reachable.

Thu Aug 6 13:25:14 2015 - [info] Master MHA Node version is 0.56.

Thu Aug 6 13:25:14 2015 - [info] Checking recovery script configurations on 120.26.137.247(120.26.137.247:3316)..

Thu Aug 6 13:25:14 2015 - [info] Executing command: save\_binary\_logs --command=test --start\_pos=4 --binlog\_dir=/mydata/mysql/binlog --output\_file=/mydata/masterha/binlog/save\_binary\_logs\_test --manager\_version=0.56 --start\_file=mysql-bin.000005

Thu Aug 6 13:25:14 2015 - [info] Connecting to root@120.26.137.247(120.26.137.247:22)..

Creating /mydata/masterha/binlog if not exists.. ok.

Checking output directory is accessible or not..

ok.

Binlog found at /mydata/mysql/binlog, up to mysql-bin.000005

Thu Aug 6 13:25:14 2015 - [info] Binlog setting check done.

Thu Aug 6 13:25:14 2015 - [info] Checking SSH publickey authentication and checking recovery script configurations on all alive slave servers..

Thu Aug 6 13:25:14 2015 - [info] Executing command : apply\_diff\_relay\_logs --command=test --slave\_user='rkylinadmin' --slave\_host=120.26.71.71 --slave\_ip=120.26.71.71 --slave\_port=3316 --workdir=/mydata/masterha/binlog --target\_version=5.6.23-log --manager\_version=0.56 --relay\_log\_info=/mydata/mysql/data/relay-log.info --relay\_dir=/mydata/mysql/data/ --slave\_pass=xxx

Thu Aug 6 13:25:14 2015 - [info] Connecting to root@120.26.71.71(120.26.71.71:22)..

Checking slave recovery environment settings..

Opening /mydata/mysql/data/relay-log.info ... ok.

Relay log found at /mydata/mysql/binlog, up to mysql-relay-bin.000002

Temporary relay log file is /mydata/mysql/binlog/mysql-relay-bin.000002

Testing mysql connection and privileges..Warning: Using a password on the command line interface can be insecure.

done.

Testing mysqlbinlog output.. done.

Cleaning up test file(s).. done.

Thu Aug 6 13:25:15 2015 - [info] Executing command : apply\_diff\_relay\_logs --command=test --slave\_user='rkylinadmin' --slave\_host=121.40.166.38 --slave\_ip=121.40.166.38 --slave\_port=3316 --workdir=/mydata/masterha/binlog --target\_version=5.6.23-log --manager\_version=0.56 --relay\_log\_info=/mydata/mysql/data/relay-log.info --relay\_dir=/mydata/mysql/data/ --slave\_pass=xxx

Thu Aug 6 13:25:15 2015 - [info] Connecting to root@121.40.166.38(121.40.166.38:22)..

Checking slave recovery environment settings..

Opening /mydata/mysql/data/relay-log.info ... ok.

Relay log found at /mydata/mysql/binlog, up to mysql-relay-bin.000002

Temporary relay log file is /mydata/mysql/binlog/mysql-relay-bin.000002

Testing mysql connection and privileges..Warning: Using a password on the command line interface can be insecure.

done.

Testing mysqlbinlog output.. done.

Cleaning up test file(s).. done.

Thu Aug 6 13:25:15 2015 - [info] Slaves settings check done.

Thu Aug 6 13:25:15 2015 - [info]

120.26.137.247(120.26.137.247:3316) (current master)

+--120.26.71.71(120.26.71.71:3316)

+--121.40.166.38(121.40.166.38:3316)

Thu Aug 6 13:25:15 2015 - [info] Checking replication health on 120.26.71.71..

Thu Aug 6 13:25:15 2015 - [info] ok.

Thu Aug 6 13:25:15 2015 - [info] Checking replication health on 121.40.166.38..

Thu Aug 6 13:25:15 2015 - [info] ok.

Thu Aug 6 13:25:15 2015 - [warning] master\_ip\_failover\_script is not defined.

Thu Aug 6 13:25:15 2015 - [warning] shutdown\_script is not defined.

Thu Aug 6 13:25:15 2015 - [info] Got exit code 0 (Not master dead).

MySQL Replication Health is OK.

没有报错信息，只有两个警告而已，复制也显示正常了。

## 检查MHA Manager的状态

通过master\_check\_status脚本查看Manager的状态：

# masterha\_check\_status --conf=/etc/app1.cnf

app1 is stopped(2:NOT\_RUNNING).

注：如果正常，会显示”PING\_OK”，否则会显示”NOT\_RUNNING”，这代表MHA监控没有开启。

## 开启MHA Manager监控

#nohup masterha\_manager --conf=/etc/app1.cnf --ignore\_last\_failover > /mydata/masterha/app1/manager.log 2>&1 &

启动参数介绍：

 –remove\_dead\_master\_conf：该参数代表当发生主从切换后，老的主库的ip将会从配置文件中移除。

 –manger\_log：日志存放位置

 –ignore\_last\_failover：在缺省情况下，如果MHA检测到连续发生宕机，且两次宕机间隔不足8小时的话，则不会进行Failover，之所以这样限制是为了避免ping-pong效应。该参数代表忽略上次MHA触发切换产生的文件，默认情况下，MHA发生切换后会在日志目录，也就是上面设置的/data产生app1.failover.complete文件，下次再次切换的时候如果发现该目录下存在该文件将不允许触发切换，除非在第一次切换后收到删除该文件，为了方便，这里设置为–ignore\_last\_failover。

注意：这里虽然程序已在后台运行，但是我们退出shell的时候一定要用exit方式退出，不可以直接关闭shell窗口。（Xshell4为例，已验证）

## 关闭MHA Manage监控

关闭很简单，使用masterha\_stop命令完成。

# masterha\_stop --conf=/etc/app1.cnf

## 查看启动日志

# tail -f /mydata/log/manager.log

## 运维

安装目录：/usr/local/bin

工作目录：/mydata/masterha/app1

日志：/mydata/log/manager.log

启动：/etc/init.d/masterha\_manager\_start

#!/bin/bash

nohup /usr/local/bin/masterha\_manager --conf=/etc/app1.cnf --ignore\_last\_failover > /dev/null 2>&1 &

关闭：/etc/init.d/masterha\_manager\_stop

#!/bin/bash

/usr/local/bin/masterha\_stop --conf=/etc/app1.cnf

# Atlas部署

## 安装

#rpm -i Atlas-2.2.1.el6.x86\_64.rpm

安装的目录是/usr/local/mysql-proxy

cp /usr/local/mysql-proxy/bin/mysql-proxyd /etc/init.d/mysql-proxyd

# vim /etc/init.d/mysql-proxyd

第8行改为 confile="/etc/$1.cnf"

## 配置

1. 将配置文件test.cnf放在/etc下

配置范例及说明如下：

[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

#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=10.51.23.196:3316

#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=10.168.6.6:3316

#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，下行的user1和user2为示例，将其替换为你的MySQL的用户名和加密密码！

pwds=rkylinadmin:qCyFeuAFF15So2ktcXUZdg==,account:cpgZeixBKW16h5WoxJ+Tqg==,order:wooiZH9oVZiDE9OHJrBUyQ==,checkstand:W7kgAb/RvmOmNvLCWgic/xryz+hZmUy9,payment:GE9UzQV3ILd6h5WoxJ+Tqg==,multi\_gate:ZxyNynXYjEW0kxtImA/O+Rryz+hZmUy9,settle:TV2oWXbdm3ZSo2ktcXUZdg==,user:uoX7y5Kf5TCm5Q11URZpxQ==,operate:a7V0DN26c9V6h5WoxJ+Tqg==,logsystem:BpTkydWi/b0MJwnB7eHRlQ==,queryuser:twBAbMgH1v0=

#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,true后面不能有空格。

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,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

log-level=debug

#日志存放的路径

log-path=/mydata/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=REALTIME

#慢日志输出设置。当设置了该参数时，则日志只输出执行时间超过sql-log-slow（单位：ms)的日志记录。不设置该参数则输出全部日志。

#sql-log-slow = 10

#关闭不活跃的客户端连接设置。当设置了该参数时，Atlas会主动关闭经过'wait-timeout'时间后一直未活跃的连接。单位：秒

wait-timeout=28800

#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=0.0.0.0:3316

#Atlas监听的管理接口IP和端口

admin-address=0.0.0.0:3315

#连接池的最小空闲连接数，可根据业务请求量大小适当调大或调小

#min-idle-connections=10

#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，设置该项后客户端不再需要执行SET NAMES语句

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=10.252.124.43,10.168.6.6,10.51.23.196,10.252.82.111,10.251.233.151

#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

## 配置详解

1. pwds

(1)该参数用来实现多用户的权限控制功能。Atlas 2.x版本中必须配置该项参数，配置方法见《Atlas安装》文档中说明。

(2)密码使用PREFIX/bin/encrypt程序加密，避免有人打开配置文件就能看到所有用户的密码，只是简单地防护一下，并不能防止专门的恶意破解。

(3)在Atlas的内存中，多个用户共用连接池，池内同时有多个用户名及使用其密码认证过的连接，在A用户连接Atlas时，从池中取出的连接有可能是B用户认证过的，这就需要使用A用户的密码将此连接change为A用户的认证连接，否则直接给A用户使用该连接就会引发权限问题。

(4)如果将各用户的连接池独立，则无法与min-idle-connections机制良好配合，会造成连接数一直上涨的问题。（此项说明只针对Atlas 1.x版本。在Atlas 2.x版本中当连接池中没有可用连接时，Atlas支持自建连接，已经不需要min-idle-connections机制。）

(5)官方版本中多用户也是共用连接池，但没有实现change机制，因此经常发生A用户看到B用户的库和表的现象。

2. client-ips

(1)该参数用来实现IP过滤功能。

(2)在传统的开发模式中，应用程序直接连接DB，因此DB会对部署应用的机器(比如web服务器)的IP作访问授权。

(3)在引入中间层后，因为连接DB的是Atlas，所以DB改为对部署Atlas的机器的IP作访问授权，如果任意一台客户端都可以连接Atlas，就会带来潜在的风险。

(4)client-ips参数用来控制连接Atlas的客户端的IP，可以是精确IP，也可以是IP段，以逗号分隔写在一行上即可，如client-ips=192.168.1.2, 192.168.2，这就代表192.168.1.2这个IP和192.168.2.\*这个C段的IP可以连接Atlas，其他IP均不能连接。

(5)如果该参数不设置，则任意IP均可连接Atlas。

(6)如果设置了client-ips参数，且Atlas前面挂有LVS，则必须设置lvs-ips参数，否则可以不设置lvs-ips。

3. lvs-ips

(1)该参数是Atlas前面挂接的LVS的物理网卡的IP，不是虚IP。

(2)该参数用来实现平滑重启功能，否则在重启Atlas的瞬间的那些SQL请求都会失败，如果认为应用可以接受瞬间的报错则可以直接重启。

(3)平滑重启的条件：至少有两台配置相同的Atlas，且挂接在LVS之后。

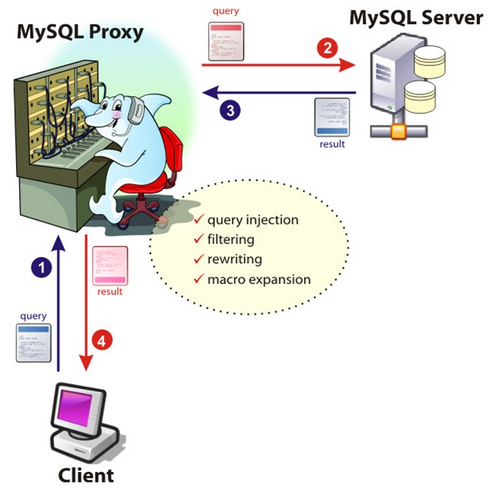
(4)平滑重启的原理：LVS周期性地对后端Atlas的存活检测有两种方式，一是直接去探测端口是否可连接，二是执行一个脚本，这个脚本会去尝试连接Atlas，通过脚本的返回值来决定每个后端是否可用。Atlas有两种运行状态，通常为online，可通过发信号将其置为offline。Atlas检测到来请求的IP是LVS的网卡IP时，如果处于online状态，就向LVS的检测脚本返回online，如果处于offline状态，就向脚本返回offline。比如我现在因为某种原因需要重启一台Atlas，但直接重启势必导致瞬间的SQL请求全部失败，对前端应用造成影响。因此我先发下线信号将Atlas置为offline状态，当LVS的检测脚本发现返回值是offline时，便将这台Atlas摘除，从此时开始便没有新的请求导向这台Atlas。等到已经打向这台Atlas的SQL请求处理完毕后(这是一个很短的时间)，就可以安然重启Atlas而不必担心对前端造成影响了。

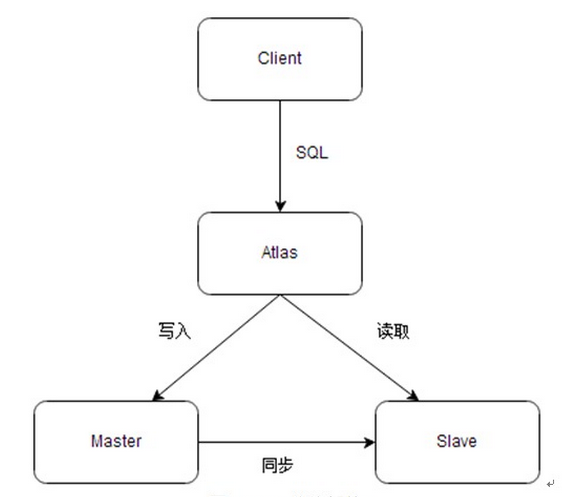
(5)对于大多数应用来说，可能并没有那么敏感，因此不使用平滑重启功能而直接重启也是可以接受的。但在设置了client-ips参数的情况下，必须设置lvs-ips参数，否则LVS的物理网卡的IP会被Atlas拒绝，如果未设置client-ips参数，说明不开启IP过滤功能，则lvs-ips也可以不设置。

## Atlas架构

1.Atlas的整体架构

Atlas是一个位于应用程序与MySQL之间中间件。在后端DB看来，Atlas相当于连接它的客户端，在前端应用看来，Atlas相当于一个DB。Atlas作为服务端与应用程序通讯，它实现了MySQL的客户端和服务端协议，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。Atlas的整体架构，可参考下面这两幅图：





2.Atlas的线程模型

Atlas启动后会创建多个线程，其中一个为主线程，其余为工作线程。主线程负责监听所有的客户端连接请求，工作线程只监听主线程的命令请求。

如下图所示，主线程接收到客户端的连接请求，将该请求的相关信息封装为一个名为CON的结构，再把该结构推入一个异步队列。然后通过round-robin方式选择一个工作线程，向其发送一个字节的数据包以激活它。工作线程在收到主线程的激活指令后，从异步队列中取出CON结构，开始处理客户端的请求。

xchmx

3.Atlas配合LVS使用的架构

下图是一个可以参考的整体架构，LVS前端做负载均衡，两个Atlas做HA,防止单点故障。LVS周期性地对后端Atlas的存活检测有两种方式，一是直接去探测端口是否可连接，二是执行一个脚本，这个脚本会去尝试连接Atlas，通过脚本的返回值来决定每个后端是否可用。Atlas有两种运行状态，通常为online，可通过发信号将其置为offline。Atlas检测到来请求的IP是LVS的网卡IP时，如果处于online状态，就向LVS的检测脚本返回online，如果处于offline状态，就向脚本返回offline。比如我现在因为某种原因需要重启一台Atlas，但直接重启势必导致瞬间的SQL请求全部失败，对前端应用造成影响。因此我先发下线信号将Atlas置为offline状态，当LVS的检测脚本发现返回值是offline时，便将这台Atlas摘除，从此时开始便没有新的请求导向这台Atlas。等到已经打向这台Atlas的SQL请求处理完毕后(这是一个很短的时间)，就可以安全重启Atlas而不必担心对前端造成影响了。

## Atlas FAQ

1.Atlas是否支持多字符集？

对多字符集的支持是我们对原版MySQL-Proxy的第一项改进，符合国情是必须的。并且支持客户端在连接时指定默认字符集。

2.Atlas是否支持事务操作？

支持，且处于事务状态的客户端中途退出时，Atlas会销毁该客户端使用的连接，让后台的mysql回滚事务，保证了事务的完整性。

3.自动读写分离挺好，但有时候我写完马上就想读，万一主从同步延迟怎么办?

SQL语句前增加 /\*master\*/ 就可以将读请求强制发往主库。在mysql命令行测试该功能时，需要加-c选项，以防mysql客户端过滤掉注释信息。

4.主库宕机，读操作受影响么？

在Atlas中读操作不受影响，Atlas会将读请求转发到其他还存活的从库上。但此时写请求将会失败，因为主库宕机了。

5.检测后端DB状态会阻塞正常请求么？

不会, atlas中检测线程是异步进行检测的，即使有db宕机，也不会阻塞主流程。在Atlas中没有什么异常会让主流程阻塞。

6.想下线一台DB, 又不想停掉mysql server, 怎么办？

可以通过管理接口手动上下线后端db, Atlas会优先考虑管理员的意愿。

7.想给集群中增加一台DB, 不想影响线上正常访问可以吗？

通过管理接口可以轻松实现。

8.相比官方mysql-proxy, Atlas还有哪些改进？

A: 这实在是个难以回答的问题，性能，稳定性，可靠性，易维护性，我们做过几十项的改进，下面会尽量列一些较大的改动

9.Atlas支持mysql的prepare特性吗？

目前Atlas部分支持prepare功能，支持java,python,PHP(PDO方式）。

10.Altas支持多个主库的运行模式吗？

目前还未对于Atlas后面挂接多个主库的情形进行测试过，不建议这样使用。建议使用一主一从或一主多从的模式。

11.在使用Atlas的过程中，发现了Atlas存在的bug或者对Atlas有新的功能需求，如何反馈给开发者？

对于用户反馈的bug，我们非常重视。欢迎用户将bug的复现的环境、步骤和运行截图发邮件至zhuchao[AT]360.cn。同时如果用户在实际的应用场景中，对Atlas有新的功能需求，也可以向我们发邮件，我们将及时回复。另外有热心网友建了QQ群326544838，开发者也已经加入，方便讨论。

12.java程序连接Atlas出现乱码问题

把jdbc连接中的amp;删除掉，例如：将

jdbc:mysql://10.10.10.37:3306/user\_db?useUnicode=true&amp;characterEncoding=utf-8&amp;autoReconnect=true

修改为：

jdbc:mysql://10.10.10.37:3306/user\_db?useUnicode=true&characterEncoding=utf-8&autoReconnect=true

13.监控主从同步之间的延迟?

Atlas不负责MySQL的主从同步操作，需要DBA自己管理。但热心的网友已经通过脚本实现了通过调用Atlas提供的接口，来监控主从之间的同步，并做上下线从库的操作。有需要的同学参看：

https://github.com/chenzhe07/Atlas\_auto\_setline

14.java程序连接Atlas发现不能读写分离，所有的请求都发向主库，这是为什么？

检查一下java框架，是不是默认将autocommit设置为0了，很多java框架将语句都封装在一个事务中，而Atlas会将事务请求都发向主库。

15.Atlas有sql语句黑名单过滤机制吗？

有的，Atlas会屏蔽不带where条件的delete和update操作，以及sleep函数。

## 运维

安装目录：/usr/local/mysql-proxy

日志：/mydata/log/test.log

SQL日志：/mydata/log/sql\_test.log

启动：/etc/init.d/mysql-proxyd test start

关闭：/etc/init.d/mysql-proxyd test stop

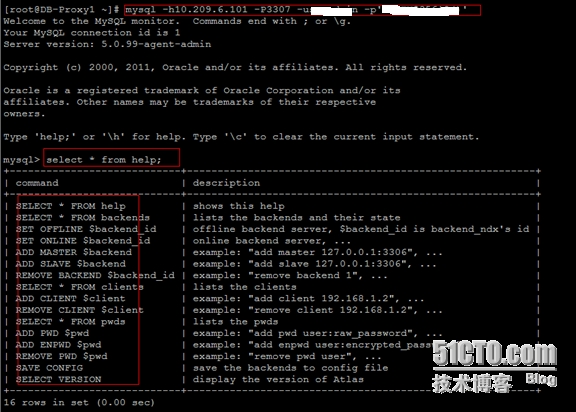
重启：/etc/init.d/mysql-proxyd test restart

登录控制台：

mysql -uadmin -padmin -h 10.252.82.111 -P3315

mysql -uadmin -padmin -h 10.251.233.151 -P3315

mysql> select \* from help;

[](http://s3.51cto.com/wyfs02/M01/58/72/wKioL1Sw8S_gXZOuAALgQK7R39c195.jpg)

连接数据库：# mysql -urkylinadmin -p’Rkylinadmin\_web!1’ -h 10.252.82.111 -P3316