mysql 主从搭建

#--------------------ALL---------------------------#

1、配置 yum 源，安装 mysql-5.7.17

[local\_soft]

name=Local Base Soft

baseurl="ftp://192.168.1.254/public"

enabled=1

gpgcheck=0

清理缓存

yum clean all

#--------------------master------------------------#

在 mysql-master 上修改 my.cnf 打开 binlog 并添加 server\_id

bind-address = 0.0.0.0

server-id = 18

log\_bin = mysql-bin

binlog-format = statement

relay-log = relay-log

重启服务 systemctl restart mysqld

初始化master

reset master;

添加同步用户

create user 'repl'@'%' IDENTIFIED BY 'lper';

grant replication client,replication slave on \*.\* to repl@'%';

安装备份工具 xtrabackup

yum install -y percona-xtrabackup-24

备份数据库

slave-info 记录 show master 的信息

innobackupex --slave-info --user="root" --password="toor" \

--host="localhost" --no-timestamp ./backup

#--------------------slave-------------------------#

安装 mysql server 和 xtrabackup

yum install -y mysql-community-server percona-xtrabackup-24

使用 innobackup 恢复备份

innobackupex --apply-log backup

innobackupex --copy-back ./backup

恢复权限

chown -R mysql:mysql /var/lib/mysql

设置 mysql slave 的my.cnf 增加 server\_id 及 binlog 配置

bind-address = 0.0.0.0

server-id = 17

log\_bin = mysql-bin

binlog-format = statement

relay-log = relay-log

启动 mysql 设置主从，binlog 文件及其执行位置在 /var/lib/mysql/xtrabackup\_info 查找

reset slave;

change master to master\_host='192.168.1.18',\

master\_user='repl',master\_password='lper',\

master\_log\_file="mysql-bin.000001", master\_log\_pos=615;

start slave;

检查验证

show slave status\G

#------------------semi sync master----------------#

查看 mysql 插件

show plugins;

安装半同步插件

install plugin rpl\_semi\_sync\_master soname 'semisync\_master.so';

开启半同步

set global rpl\_semi\_sync\_master\_enabled=1;

等待超时时间

设置此参数值（ms）,为了防止半同步复制在没有收到确认的情况下发生堵塞，如果Master在超时

之前没有收到任何确认，将恢复到正常的异步复制，并继续执行没有半同步的复制操作。

set global rpl\_semi\_sync\_master\_timeout=1000;

查看状态

show global variables like '%rpl\_semi%';

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

#------------------semi sync slave-----------------#

查看 mysql 插件

show plugins;

安装半同步插件

install plugin rpl\_semi\_sync\_slave soname 'semisync\_slave.so';

开启半同步

set global rpl\_semi\_sync\_slave\_enabled=1;

查看状态

show global variables like '%rpl\_semi%';

重启 IO 线程

stop slave io\_thread;

start slave io\_thread;

#----------------------my.cnf ---------------------#

plugin-load = "rpl\_semi\_sync\_master=semisync\_master.so"

plugin-load = "rpl\_semi\_sync\_slave=semisync\_slave.so"

rpl\_semi\_sync\_slave\_enabled = 1

rpl\_semi\_sync\_master\_enabled = 1

rpl\_semi\_sync\_master\_timeout = 3000

#----------------------mha node--------------------#

安装 mha node 节点包

yum install gcc pcre-devel pkgconfig autoconf automake perl-ExtUtils-MakeMaker perl-CPAN perl-DBI perl-DBD-MySQL

安装 mha4mysql-node

perl Makefile.PL

make

make install

#--------------------mha manager-------------------#

# mha 官方网站 https://github.com/yoshinorim/mha4mysql-manager/wiki/Downloads

安装 mha node 节点包

yum install -y gcc pcre-devel pkgconfig autoconf automake perl-ExtUtils-MakeMaker perl-CPAN perl-DBI perl-DBD-MySQL

安装 mha4mysql-node

perl Makefile.PL

make

make install

安装 mha manager 节点

安装依赖软件包

yum install -y perl-Config-Tiny perl-Log-Dispatch perl-Parallel-ForkManager perl-Time-HiRes

安装 mha 管理节点

perl Makefile.PL

[Core Features]

- DBI ...loaded. (1.627)

- DBD::mysql ...loaded. (4.023)

- Time::HiRes ...loaded. (1.9725)

- Config::Tiny ...loaded. (2.14)

- Log::Dispatch ...loaded. (2.41)

- Parallel::ForkManager ...loaded. (1.18)

- MHA::NodeConst ...loaded. (0.56)

\*\*\* Module::AutoInstall configuration finished.

Checking if your kit is complete...

Looks good

make

make install

mha 是依靠 ssh 远程配置管理 mysql 服务器的，所以要求管理节点机器到所有 mysql

机器能做到 ssh 免密码登录

/etc/ssh/ssh\_config 配置不校验 host key，不输人 yes

StrictHostKeyChecking no

cd /root/.ssh

ssh-keygen -t rsa -b 2048 -N '' -f id\_rsa

for i in mysql{15..18};do

ssh-copy-id -i id\_rsa.pub ${i}

done

把私钥 id\_rsa 拷贝给所有 mysql 主机

for i in mysql{15..18};do

scp id\_rsa ${i}:.ssh/id\_rsa

done

mha 切换 vip 是靠脚本实现，vim 编辑脚本 master\_ip\_failover 设置 vip

（line:35）

my $vip = '192.168.1.10/24'; # Virtual IP

cp master\_ip\_failover /usr/local/bin/

chmod 755 /usr/local/bin/master\_ip\_failover

添加 默认配置文件 /etc/masterha\_default.cnf 和 /etc/mha.cnf 配置文件

touch /etc/masterha\_default.cnf

cat /etc/mha.cnf

[server default]

manager\_log=/var/log/mha.log

manager\_workdir=/var/lib/mha

user=root

password=toor

repl\_user=repl

repl\_password=lper

ssh\_user=root

ping\_interval=1

remote\_workdir=/var/lib/mha

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

[server18]

candidate\_master=1

hostname=mysql18

[server17]

candidate\_master=1

hostname=mysql17

[server16]

hostname=mysql16

no\_master=1

[server15]

hostname=mysql15

no\_master=1

在当前的 master 上手工绑定 vip 执行检查测试

检查 ssh 免密码登录

masterha\_check\_ssh --conf=/etc/mha.cnf

检查 mysql 主从配置

masterha\_check\_repl --conf=/etc/mha.cnf

排除所有错误，添加 root 用户远程登录权限

create user 'root'@'%' IDENTIFIED BY 'toor';

grant ALL ON \*.\* to root@'%';

添加参数 relay\_log\_purge=0

启动 mha

masterha\_manager --conf=/etc/mha.cnf --ignore\_last\_failover

验证测试

#--------------------- mycat --------------------#

创建一个用于查询的用户

create user 'read'@'%' IDENTIFIED BY 'daer';

grant select on \*.\* to 'read'@'%';

在机器上安装 java-1.8.0-openjdk-devel

拷贝 mycat 到 /usr/local/

配置 /usr/local/mycat/conf/server.xml

82: <property name="schemas">mydb</property>

97: <property name="schemas">mydb</property>

配置 /usr/local/mycat/conf/schema.xml

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="mydb" checkSQLschema="false" sqlMaxLimit="100"

dataNode="dn1">

</schema>

<dataNode dataHost="localhost1" database="mydb" name="dn1"/>

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="3"

writeType="0" dbType="mysql" dbDriver="native"

switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostMaster" url="192.168.1.10:3306"

user="root"

password="toor">

<!-- can have multi read hosts -->

<readHost host="hostS2" url="mysql15:3306" user="read"

password="daer" />

<readHost host="hostS2" url="mysql16:3306" user="read"

password="daer" />

<readHost host="hostS2" url="mysql17:3306" user="read"

password="daer" />

<readHost host="hostS2" url="mysql18:3306" user="read"

password="daer" />

</writeHost>

</dataHost>

</mycat:schema>

启动 mycat ,验证测试

/usr/local/mycat/bin/mycat start

配置文件注意事项:

conf/server.xml 可以不修改，但要注意

<property name="schemas">TESTDB</property>

虚拟库名称，要和后面对应

schemas是这个用户下的逻辑数据库可以有多个逻辑数据库可以用“,”逗号隔开 用户名和密码是连接 mycat 的用户名和密码，与 mysql 实例的用户名密码无关 mycat默认的普通连接端口是8066，管理连接端口是9066 schema：逻辑数据库 dataNode：节点

dataHost：节点对应的读库写库的地址和连接

balance指的负载均衡类型，目前的取值有4种：

balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的writeHost上。

balance="1"，全部的readHost与stand by writeHost参与select语句的负载均衡

balance="2"，所有读操作都随机的在writeHost、readhost上分发。

balance="3"，所有读请求随机的分发到wiriterHost对应的readhost执行，writerHost不负担读压力

switchType指的是切换的模式，目前的取值也有4种：

switchType='-1' 表示不自动切换

switchType='1' 默认值，表示自动切换

switchType='2' 基于MySQL主从同步的状态决定是否切换,心跳语句为 show slavestatus

switchType='3' 基于MySQL galary cluster的切换机制（适合集群）（1.4.1），心跳语句为 show status like 'wsrep%'

WriteType参数设置：

writeType=“0”, 所有写操作都发送到可用的writeHost上。

writeType=“1”，所有写操作都随机的发送到readHost。

writeType=“2”，所有写操作都随机的在writeHost、readhost分上发。

配置完成以后连接 mycat 查询

mysql -uroot -p123456 -h192.168.4.20 -P 8066 -e 'select @@hostname;'

多查询几次，可以看到轮询效果

第二台 mycat

安装 java-1.8.0-openjdk-devel

拷贝 /usr/local/mycat 到本机相同目录，启动服务即可

#--------------- haproxy keepalived--------------#

yum 安装 haproxy

修改 /etc/haproxy/haproxy.cfg

listen mycat\_3306 \*:3306

mode tcp # mysql 得使用 tcp 协议

option tcpka # 使用长连接

balance leastconn # 最小连接调度算法

server mycat\_01 192.168.1.13:8066 check inter 3000 rise 1 maxconn 1000 fall 3

server mycat\_02 192.168.1.14:8066 check inter 3000 rise 1 maxconn 1000 fall 3

启动服务

可以在服务器上使用 ss -atn|grep "ESTAB.\*8066" 查看后端和哪台服务建立连接了

为防止 haproxy 单点故障，配置两台 haproxy 使用 keepalived 实现高可用

第二台 haproxy 配置同第一台

keepalived 配置

yum 安装 keepalived

修改配置文件 keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id mycat

}

vrrp\_script chk\_haproxy {

script "killall -0 haproxy" # cheaper than pidof

interval 2 # check every 2 seconds

}

vrrp\_instance Mycat {

state BACKUP

interface eth0

track\_interface {

eth0

}

virtual\_router\_id 150

priority 200

! nopreempt

advert\_int 2

authentication {

auth\_type PASS

auth\_pass test\_mycat

}

virtual\_ipaddress {

192.168.1.100/24 brd 192.168.1.255 dev eth0 label eth0:1

}

track\_script {

chk\_haproxy weight=0 # +2 if process is present

}

}

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

| keepalived | | +-----+ | | +--------+ +--------+ |

|-------------| | |mycat| | ==> | |mysql(M)|<==>|mysql(M)| |

| +-------+ | | +-----+ | | +--------+ +--------+ |

| |haproxy|=>| ==> | | | MHA或其他多主高可用方案 |

| +-------+ | | +-----+ | |-~-~-~-~-~-~~~~-~-~-~-~-~-|

client --> vip | |高| | | |mycat| | ==> | +--------+ +--------+ |

| |可| | | +-----+ | | |mysql(S)| 从 |mysql(S)| |

| |用| | | | | +--------+ 库 +--------+ |

| +-------+ | | +-----+ | | +--------+ 集 +--------+ |

| |haproxy|=>| ==> | |mycat| | ==> | |mysql(S)| 群 |mysql(S)| |

| +-------+ | | +-----+ | | +--------+ +--------+ |

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

#----------------------redis-----------------------#

源码安装 redis

安装编译工具

yum install gcc make automake pkgconfig

添加用户

adduser -s /sbin/nologin -d /var/lib/redis redis

编译安装 redis

make MALLOC=libc

make PREFIX=/usr/local/redis install

mkdir -p /usr/local/redis/conf

cp \*.conf /usr/local/redis/conf/

配置 redis 1主2从 redis.conf

bind 0.0.0.0

port 6379

dir /var/lib/redis

daemonize yes

启动 redis

./bin/redis-server conf/redis.conf

设置主从，查看状态

redis-cli -h redis02 -p 6379

redis02:6379> slaveof redis01 6379

OK

[root@redis01 ~]# redis-cli -h redis03 -p 6379

redis03:6379> slaveof redis01 6379

OK

查看状态

redis-cli -h redis01 -p 6379 info replication

配置 redis 哨兵 sentinel.conf

bind 0.0.0.0

protected-mode no

daemonize yes

port 26379

dir /tmp

sentinel monitor mymaster redis01 6379 1

sentinel down-after-milliseconds mymaster 3000

sentinel parallel-syncs mymaster 1

sentinel failover-timeout mymaster 5000

sentinel client-reconfig-script mymaster /usr/local/bin/reconfig.sh

查看哨兵状态

redis-cli -h redis01 -p 26379 info sentinel

reconfig.sh

#!/bin/bash

# args=(<master-name> <role> <state> <from-ip> <from-port> <to-ip> <to-port>)

# mymaster leader start old.ip old.port new.ip new.port

logger -p local0.info -t redis "${@:-NULL}"

vip="192.168.1.100/32"

read oldip newip <<<"$4 $6"

if $(ip -o a s |grep -q ${oldip:-0.0.0.0});then

/sbin/ifconfig eth0:1 down &>/dev/null

elif $(ip -o a s|grep -q ${newip:-0.0.0.0});then

/sbin/ifconfig eth0:1 ${vip}

/sbin/arping -q -c 3 -A ${vip%/\*} -I eth0

fi

reconfig 2

#!/bin/bash

# mymaster leader start 192.168.1.13 6379 192.168.1.12 6379

VIP="192.168.1.10/24"

local\_ip=$(ip -o addr show dev eth0 label eth0|awk '{print

gensub("/.\*","","",$4)}')

if [[ "${local\_ip}" == "$4" ]];then

/usr/sbin/ifconfig eth0:1 down

elif [[ "${local\_ip}" == "$6" ]];then

/usr/sbin/ifconfig eth0:1 "${VIP}"

fi