**Morning：**

1、生产环境如何提高读写性能？

1）分析业务模型：读or写为主

读：主从同步

写：分库分表 / Mysql cluster

2）如何解决读写请求都大？

拆。根据不同维度对数据进行分片（地域、时间..）

按照逻辑拆（读请求使用主从同步，写请求使用分库分表）

3）瞬时高并发

读缓存、写队列(写请求放到消息队列)

2、搭建mysql高可用、负载均衡架构

Keepalived

haproxy

高

可

用

haproxy

mycat

mycat

mycat

mysql（M）<==> mysql（M）

MHA或其他高可用方案

mysql（S） 从 mysql（S）

库

集

mysql（S） 群 mysql（S）

web -> VIP

**相关技术点：**

xtrabackup备份与恢复

Mysql主从同步、半同步复制模式（只对2台master开，以防性能低的从库拖慢整体速度）

MHA（主库高可用）

mycat（分库分表+读写分离+负载均衡+健康检查）

haproxy（mycat负载均衡，后端mycat8066端口转成前端3306，自动健康检查）

Keepalived（haproxy的高可用+haproxy健康检查）

Tips：为防止雪崩，设计服务器架构时，建议多出最大并发量一半的冗余量。

**步骤1：安装mysql-server、xtrabackup、分发my.cnf**

ansible mysql -m shell -a ‘yum clean all’

ansible mysql -m shell -a ‘yum repolist’

ansible mysql -m shell -a ‘yum -y install mysql-community-server

percona-xtrabackup-24’

vim my.cnf

bind-address=0.0.0.0

server-id={{ var\_ip.stdout }}

binlog-format=mixed

log\_bin=/var/log/mysql/mysql-bin

relay-log=/var/log/mysql/relay-log

relay-log-index=/var/log/mysql/relay-log.index

relay-log-info-file=/var/log/mysql/relay-log.info

ansible-playbook my.yml

ansible mysql -m shell -a 'mkdir -p /var/log/mysql'

ansible mysql -m shell -a 'chown -R mysql:mysql /var/log/mysql'

**步骤2：初始化主库**

[root@mysql-0001 ~]# mysqld --user mysql --initialize-insecure

#root无初始密码

[root@mysql-0001 ~]# systemctl start mysqld

**步骤3：设置共性配置**

mysql> set password=”123”;

mysql> update user set Host="%" where User="root";

mysql> flush privileges;

mysql> grant replication slave on \*.\* to repl@"%" identified by '123';

**步骤4：备份&还原备用主库**

[root@mysql-0001 ~]# innobackupex -u root -p 123 -H localhost --no-timestamp --slave-info ./backup

[root@mysql-0001 ~]# scp -r ./backup/ my2:/root/

[root@mysql-0002 ~]# innobackupex --user root --password 123 --apply-log ./backup/

[root@mysql-0002 ~]# innobackupex --user root --password 123 --copy-back ./backup/

[root@mysql-0002 ~]# chown -R mysql:mysql /var/lib/mysql

[root@mysql-0002 ~]# systemctl start mysqld

[root@mysql-0002 ~]# cat /var/lib/mysql/xtrabackup\_info

mysql> change master to master\_host='192.168.1.245',

>master\_user='repl',master\_password='123',

>master\_log\_file='mysql-bin.000003',master\_log\_pos=581;

mysql> start slave;

mysql> stop slave IO\_THREAD;

mysql> show slave status \G;

Read\_Master\_Log\_Pos: 154

Exec\_Master\_Log\_Pos: 154

[root@mysql-0002 ~]# systemctl stop mysqld

**步骤5：rsync同步数据库目录到所有从库**

[root@ecs-proxy ~]# ansible mysql -m shell -a 'echo 1 | passwd --stdin root'

[root@mysql-0002 ~]# for i in my{3..6}; do

> rsync -avz --delete /var/lib/mysql $i:/var/lib/

> done

[root@ecs-proxy ~]# ansible mysql\_slave -m shell -a 'rm -rf /var/lib/mysql/auto.cnf'

[root@ecs-proxy ~]# ansible mysql\_slave -m shell -a 'systemctl start mysqld'

[root@mysql-0001 ~]# mysql -uroot -p123 -e ‘show processlist;’

#查看主从同步配置情况

**步骤6：my1,my2启动半同步复制模式**

cat >>/etc/my.cnf <<EOF

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

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

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

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

> EOF

systemctl restart mysqld

mysql -uroot -p123 -e 'show variables like "%sync%\_enabled"'

**步骤7：mha：所有主机免密登陆，修改StrictHostKeyChecking no**

[root@ecs-proxy ~]# ansible mysql -m copy -a 'src=.ssh/id\_rsa dest=/root/.ssh/ mode=0600'

[root@xx ~]# cat /etc/ssh/ssh\_config

StrictHostKeyChecking no

**步骤8：安装mha\_node**

[root@ecs-proxy ~]# cat mha\_node.sh

#!/bin/bash

yum -y install perl-\*

cd /root/mha-soft-student/

yum -y install perl-\*

yum -y install mha4mysql-node-0.56-0.el6.noarch.rpm

[root@ecs-proxy ~]# ansible mysql -m script -a 'mha\_node.sh'

**步骤9：在my3部署mha\_manager**

yum -y install gcc pcre-devel pkgconfig autoconf automake

tar -xf mha4mysql-manager-0.56.tar.gz

cd mha4mysql-manager-0.56/

perl Makefile.PL

make

make install

**步骤10：mha配置文件、故障切换脚本**

[root@mysql-0003 ~]# cat /etc/mha.cnf

[server default]

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

manager\_workdir=/var/lib/mha

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

ping\_interval=1

remote\_workdir=/var/lib/mha

repl\_user=repl

repl\_password=123

ssh\_user=root

user=root

password=123

[server1]

candidate\_master=1

hostname=my1

[server2]

candidate\_master=1

hostname=my2

[server3]

hostname=my3

no\_master=1

[server4]

hostname=my4

no\_master=1

[server5]

hostname=my5

no\_master=1

[server6]

hostname=my6

no\_master=1

[root@mysql-0003 ~]# vim /usr/local/bin/master\_ip\_failover

my $vip = '192.168.1.200/24'; #云服务器申请vip

[root@mysql-0003 ~]# chmod 755 /usr/local/bin/mas..\_failover

**步骤11：master配置VIP，使用screen启动mha作业**

[root@mysql-0001 ~]# ifconfig eth0:1 192.168.1.200

[root@mysql-0003 ~]# screen -S mha

[root@mysql-0003 ~]# masterha\_check\_ssh --conf=/etc/mha.cnf

[root@mysql-0003 ~]# masterha\_check\_repl --conf=/etc/mha.cnf

[root@mysql-0003 ~]# masterha\_manager --conf=/etc/mha.cnf --remove\_dead\_master\_conf --ignore\_last\_failover

[root@mysql-0003 ~]# （Ctrl +ad）离线作业

**步骤12：测试mha，故障恢复**

[root@mysql-0001 ~]# systemctl stop mysqld

[root@ecs-proxy ~]# mysql -uroot -p123 -h'192.168.1.200'

mysql> select @@hostname;

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

| @@hostname |

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

| mysql-0002 |

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

mysql> show processlist;

故障恢复：

[root@mysql-0005 ~]# systemctl stop mysqld

[root@mysql-0005 ~]# rsync -avz --delete /var/lib/mysql my1:/var/lib/

[root@mysql-0001 ~]# rm -rf /var/lib/mysql/auto.cnf

[root@mysql-0001 ~]# systemctl start mysqld

[root@mysql-0001 ~]# mysql -uroot -p123 -e 'start slave'

[root@mysql-0003 ~]# cat >>/etc/mha.cnf <<EOF

> [server1]

> candidate\_master=1

> hostname=my1

> EOF

[root@mysql-0003 ~]# screen -r mha

[root@mysql-0003 ~]# masterha\_manager --conf=/etc/mha.cnf --remove\_dead\_master\_conf --ignore\_last\_failover

[root@mysql-0003 ~]# （Ctrl +ad）离线作业

**步骤13：cat1部署mycat**

[root@ecs-proxy ~]# ansible mycat -m shell -a 'yum -y install java-1.8.0-openjdk-devel'

[root@ecs-proxy my]# ansible cat1 -m copy -a 'src="mycat" dest="/usr/local/"'

[root@mycat-0001 ~]# vim /usr/local/mycat/conf/server.xml

<user name="root">

<property name="password">rw</property>

<property name="schemas">v\_mydb</property>

</user>

<user name="user">

<property name="password">ro</property>

<property name="schemas">v\_mydb</property>

<property name="readOnly">true</property>

</user>

[root@mycat-0001 ~]# cat /usr/local/mycat/conf/schema.xml

<?xml version="1.0"?>

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

<mycat:schema xmlns:mycat="http://org.opencloudb/">

<schema name="v\_mydb" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">

</schema>

#schema：逻辑数据库，name要和server.xml对应

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

#dataNode：定义节点

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="3" writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

#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。

<heartbeat>select user()</heartbeat>

<writeHost host="M1" url="192.168.1.200:3306" user="root" password="123" >

<readHost host="S3" url="192.168.1.153:3306" user="rrrr" password="123" />

#数据库要有对应的rrrr授权select用户

<readHost host="S4" url="192.168.1.181:3306" user="rrrr" password="123" />

<readHost host="S5" url="192.168.1.205:3306" user="rrrr" password="123" />

<readHost host="S6" url="192.168.1.108:3306" user="rrrr" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

[root@mycat-0001 ~]# /usr/local/mycat/bin/mycat start

mysql> select @@hostname; #验证读的负载均衡

**步骤14：同步mycat目录到cat2、cat3**

[root@mycat-0001 ~]# /usr/local/mycat/bin/mycat stop

[root@mycat-0001 ~]# rsync -avz --delete /usr/local/mycat cat2:/usr/local/

[root@mycat-0001 ~]# rsync -avz --delete /usr/local/mycat cat3:/usr/local/

**步骤15：配置haproxy、并启动**

[root@haproxy ~]# cat /etc/haproxy/haproxy.cfg

#---------------------------------------------------------------------

# main frontend which proxys to the backends

#---------------------------------------------------------------------

listen mycat\_3306 \*:3306

mode tcp

option tcpka

balance leastconn

server mycat\_01 192.168.1.29:8066 check inter 3000 rise 1 maxconn 1000 fall 3

server mycat\_02 192.168.1.101:8066 check inter 3000 rise 1 maxconn 1000 fall 3

server mycat\_03 192.168.1.196:8066 check inter 3000 rise 1 maxconn 1000 fall 3

#使用最小连接算法，保证后端服务器最大限度的稳定。

#如果使用轮询，部分sql计算量大可能会导致单台服务器繁忙；

[root@ecs-proxy ~]# ansible haproxy -m service -a 'name=haproxy state=started enabled=yes'

**步骤16：配置Keepalived，完成haproxy健康检查，双机双工模式**

[root@haproxy-0001 ~]# cat /etc/keepalived/keepalived.conf

global\_defs {

router\_id ha1

}

vrrp\_script chk\_haproxy {

script "killall -0 haproxy"

interval 2

weight 0

}

vrrp\_instance mycat\_1 {

state BACKUP

interface eth0

virtual\_router\_id 50

priority 200

! nopreempt

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.1.150/24 brd 192.168.1.255 dev eth0 label eth0:1

}

track\_script {

chk\_haproxy

}

}

vrrp\_instance mycat\_2 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 100 priority 200

nopreempt ! nopreempt

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 2222

}

virtual\_ipaddress {

192.168.1.151/24 brd 192.168.1.255 dev eth0 label eth0:2

}

track\_script {

chk\_haproxy

}

}

#程序员使用VIP时，需要在DNS配置双A记录

#监控服务健康性脚本：

方法1：cat /var/run/haproxy.pid 、 ps -ef | grep 5672或ls /proc/5672

方法2：pid=$(</var/run/haproxy.pid) ；kill -0 ${pid} 给进程发送伪信号

方法3：killall -0 haproxy

[root@haproxy-000x ~]# ss -ant #验证

State Recv-Q Send-Q Local Address:Port Peer Address:Port

ESTAB 0 0 192.168.1.150:3306 192.168.1.252:45934

1. 生产环境如何判断负载高、怎么解决？

首先考虑CPU核数，不同核数可承受的负载不同；

正常情况下，系统负载一般不超过CPU核数的2倍。

解决办法：增加负载均衡集群(web)；若是数据库服务器(根据读写情况采取措施)；

查看负载脚本：

ps -eo %mem,%cpu,ucomm | awk '{M[$NF]+=$1;P[$NF]+=$2}END{for(c in M){print c,M[c],P[c]}}' | sort -nrk 2,3 | column -t