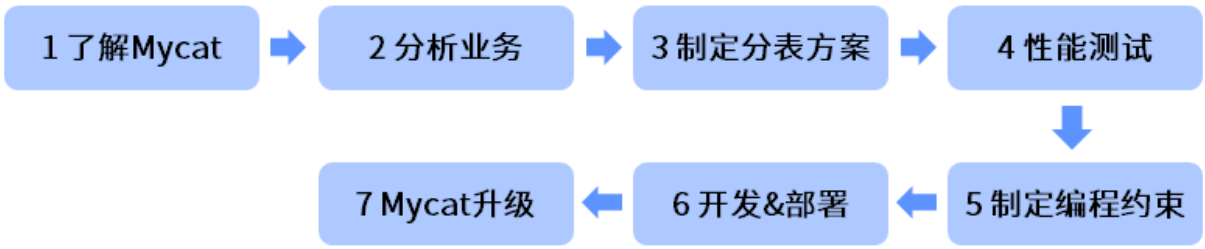
4 Mycat最佳实践

4.1 Mycat实施指南

4.1.1 实施流程



了解Mycat 的能力，包括如下的方面：

- Mycat 的起源和解决的目标；
- Mycat 在数据库中间件方面的独特功能和定位；
- Mycat 的实际案例情况；
- Mycat 的优点和不足；
- Mycat 所提供的监控和测试工具；
- Mycat 社区的动态。

其中，关于分片规则的支持和扩展、多数据库支持、SQL 拦截和注解、跨库Join、读写分离、缓存功能、高 可用性等方面需要比较深入的学习和理解，有助于正确的使用Mycat 来解决当前的业务问题。

接下来是分析当前业务，具体内容包括如下几个方面：

- 数据模型：重点关注数据的增长模式（实时大量增长还是缓慢增长）和规律、数据之间的关联关系；
- 数据访问模式：通过抓取系统中实际执行的SQL，分析其频率、响应时间、对系统性能和功能的影响程度；
- 数据可靠性的要求：系统中不同数据表的可靠性要求，以及操作模式；
- 事务的要求：系统中哪些业务操作是严格事务的，哪些是普通事务或可以无事务的；
- 数据备份和恢复问题：目前的备份模式，对系统的压力等。

数据的模型和访问模式在很大程度上决定了未来数据分片的模式，包括哪些表用全局表、哪些用ER 分片、哪 些用范围分片规则、哪些用一致性Hash 或自定义方式。而数据可靠性的要求，则影响到Mycat 后端是采用普通 的MySQL 主从还是用Gluster 多写模式，事务性要求需要相关的表或者SQL 尽量不会垮分片执行，对于以后制 定本项目的编程约束有重要意义。

分表方案则需要确定如下一些问题：

- 哪些表要分片、什么分片规则、依赖关联关系如何解决；
- 数据迁移和扩容的手段。

建议根据业务分析的结果，确定两套比较合适分表方案，然后进行性能测试，选出最佳的分表方案，性能测 试可以采用Mycat 自带的超级工具，此工具在前面提到过，可以模拟接近真实业务数据的数据，并随机制造大量的数据供测试，是目前开源的最佳数据库性能测试工具。

在最终进入开发之前，架构师还需要给出一个编程约束，需要明确列出不能执行的SQL 语句，这些约束可能包括如下几种：

- 跨越太多节点的查询语句；
- 不能Join 的表和相关的Join SQL；
- 很影响性能的复杂SQL；
- 对比较大的表的SQL 操作提示。

最后在开发阶段，还应该做到如下几点

- 一开始就按照最初的分片设计和数据规模，制造大量的随机数据，进行开发和测试，尽早发现性能问题；
- 对所有的SQL 进行统计分析，找出异常的SQL，包括跨越太多分片的SQL，以及执行缓慢的SQL，对这些SQL 进行分析和优化；
- 时刻关注性能问题。

当项目上线后，通过Mycat Web 对系统进行监控，特别是服务的IO 和网络指标，除此之外，对Mycat 运行过程中的日志也要进行排查，告警信息可能是SQL 错误，可能是Mycat Bug，及时分析处理，并积极反馈给Mycat 社区，寻求帮助。

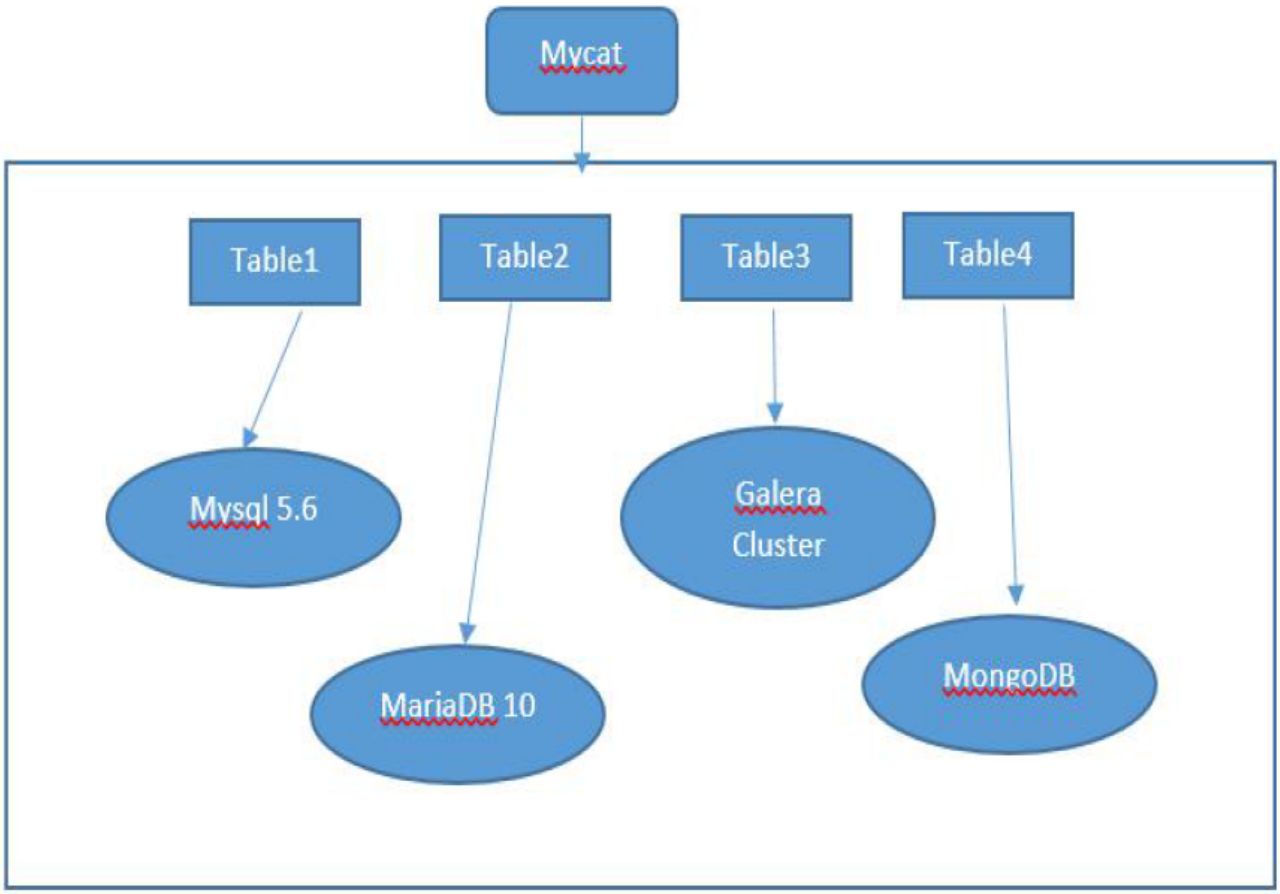
4.1.2 后端存储的选择

Mysql 尽量用比较新的稳定版，当前来说5.6 和5.7 都是比较靠谱的一个选择，因为Mysq 这两个版本做了 大量优化。另外Mysql 的各种变种版本都可以考虑。以下是一些通用准则：

- 对于非严格苛刻交易型的数据表，建议用MariaDB，这个版本目前在开源界很盛行，评价很高，percona 版 本也值得推荐，percona 有很多辅助的运维工具。

- 对于交易型的数据表，可以考虑Mysql 官方稳定版，若交易型的数据表要求可靠性非常高，比如是替代Oracle，也可以选择Galera Cluster 这种高可用的方案，他以一定的写入性能损失带来了数据的高可用和高并发访问。
- 根据数据的可靠性要求，可以采用各种数据同步方案，比如1 主多从，读写分离提升数据表的读的并发能力。
- 部分表可以用NoSQL 方式存储，而前端访问方式不变，Mycat 支持后端MongoDB 和很多NoSQL 系统，以提升查询能力
- 部分表可以采用MySQL 内存表，来提升查询和写入速度，替代部分复杂缓存方案。

下面是一个可能的Mycat 部署方案，不同的表用不同的存储方式，让不同的表根据其访问模式，都达到最佳 状态。



4.1.3 Mycat 目前存在的限制

部分SQL 还不能很好的支持

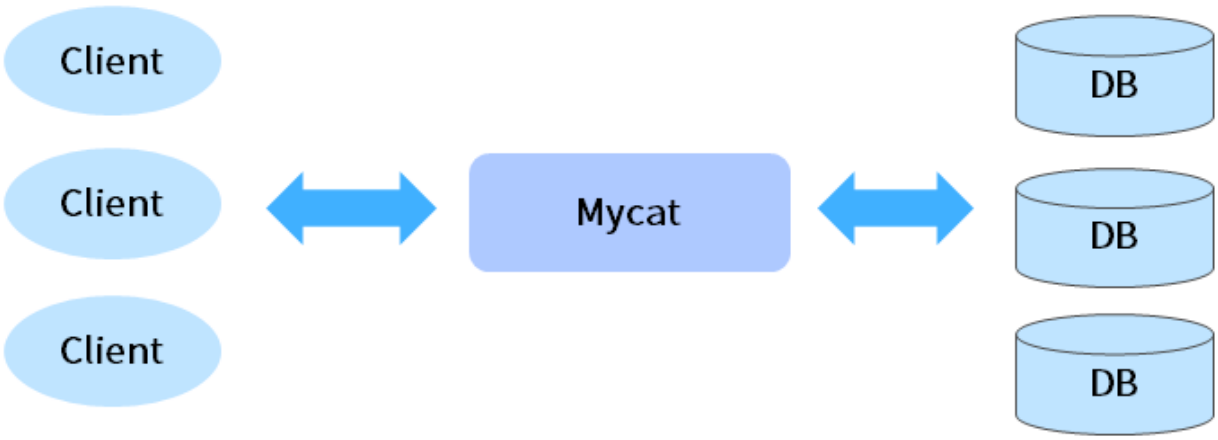
- 除了分片规则相同、ER 分片、全局表、以及SharedJoin，其他表之间的Join 问题目前还没有很好的解决，需要自己编写Catlet 来处理。
- 不支持Insert into 中不包括字段名的SQL

- insert into x select from y 的SQL，若x 与y 不是相同的分片规则，则不被支持，此时会涉及到跨分片转移。
- 跨分片的事务，目前只是弱XA 模式，还没完全实现XA 模式。
- 分片的Table，目前不能执行Lock Table 这样的语句，因为这种语句会随机发到某个节点，也不会全部分 片锁定，经常导致死锁问题，此类问题常常出现在sqldump 导入导出SQL 数据的过程中。
- 目前sql 解析器采用Druid,再某些sql 例如order，group，sum ，count 条件下，如果这类操作会出现 兼容问题，比如： select t.name as name1 from test order by t.name 这条语句select 列的别名与order by 不一致解析器会出现异常，所以在对列加别名时候要注意这类操作 异常，特别是由jpa 等类似的框架生成的语句会有兼容问题。

开发框架方面，虽然支持Hibernate，但不建议使用Hibernate，而是建议Mybatis 以及直接JDBC 操作，原因Hibernat 无法控制SQL 的生成，无法做到对查询SQL 的优化，导致大数量下的性能问题。此外，事务方面，建议自己手动控制，查询语句尽量走自动提交事务模式，这样Mycat 的读写分离会被用到，提升性能很明显。

4.2 Mycat高可用方案

4.2.1 Mycat单点方案

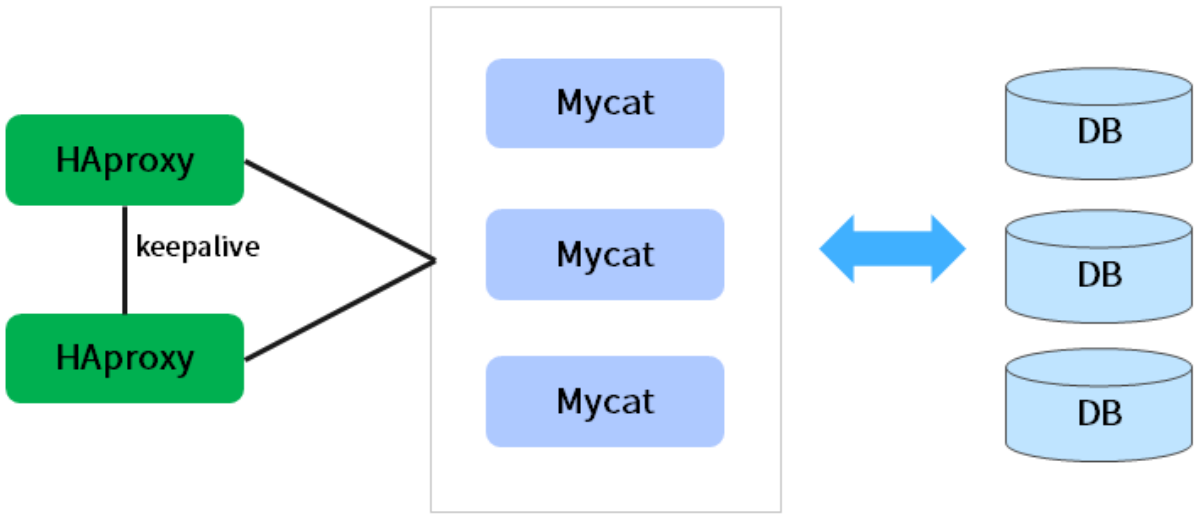


单点的不足：

1. 单点故障，整个系统不可用！
2. 并发处理能力有上限！

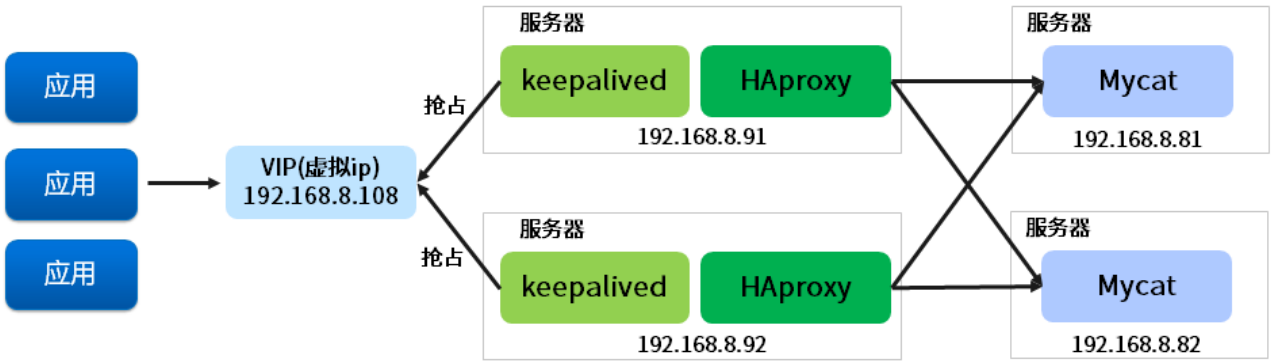
4.2.2 高可用方案

集群负载均衡：



HAproxy + keepalived+Mycat集群+DB主从同步复制 组成Mycat高可用系统

集群部署图示：



集群部署图说明：

1. keepalived 和haproxy 必须装在同一台机器上（如192.168.8.91 机器上，keepalived 和haproxy 都要安装），keepalived 负责为该服务器抢占vip（虚拟ip），抢占到vip 后，对该主机的访问可以通过原来的ip（192.168.8.91）访问，也可以直接通过vip（192.168.8.108）访问。
2. 192.168.8.92 上的keepalived 也会去抢占vip，抢占vip 时有优先级，配置keepalived.conf 中的（priority 150 #数值愈大，优先级越高,192.168.8.92 上改为120，master 和slave 上该值配置不同）决定。但是一般哪台主机上的keepalived 服务先启动就会抢占到vip，即使是slave，只要先启动也能抢到。

3. haproxy 负责将对vip 的请求分发到mycat 上。起到负载均衡的作用，同时haproxy 也能检测到mycat 是否存活，haproxy 只会将请求转发到存活的mycat 上。
4. 如果一台服务器（keepalived+haproxy 服务器）宕机，另外一台上的keepalived 会立刻抢占vip 并接管服务。
5. 如果一台mycat 服务器宕机，haporxy 转发时不会转发到宕机的mycat 上，所以mycat 依然可用。

4.2.3 高可用安装部署

haproxy 安装

```
#useradd haproxy
#wget http://haproxy.1wt.eu/download/1.4/src/haproxy-1.4.25.tar.gz

#tar zxvf haproxy-1.4.25.tar.gz

#cd haproxy-1.4.25

#make TARGET=linux26 PREFIX=/usr/local/haproxy ARCH=x86_64

#make install PREFIX=/usr/local/haproxy

#cd /usr/local/haproxy
#chown -R haproxy.haproxy *
```

haproxy.cfg

```
#cd /usr/local/haproxy
#touch haproxy.cfg
#vi/usr/local/haproxy/haproxy.cfg
global
log 127.0.0.1 local0 ##记日志的功能
maxconn 4096
chroot/usr/local/haproxy
user haproxy
group haproxy
daemon
defaults
log global
option dontlognull
```

```
retries 3
option redispatch
maxconn 2000
contimeout 5000
clitimeout 50000
srvtimeout 50000
listen admin_status 172.17.210.103:48800 ##VIP
stats uri/admin-status ##统计页面
stats auth admin:admin
mode http
option httplog
listen allmycat_service 172.17.210.103:8096 ##转发到mycat 的8066 端口，即mycat的服务端口
mode tcp
option tcplog
option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
balance roundrobin
server mycat_91 192.168.8.91:8066 check port 48700 inter 5s rise 2 fall 3
server mycat_92 192.168.8.92:8066 check port 48700 inter 5s rise 2 fall 3
srvtimeout 20000
listen allmycat_admin 172.17.210.103:8097 ##转发到mycat 的9066 端口，即mycat的管理控制台端口
mode tcp
option tcplog
option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
balance roundrobin
server mycat_91 192.168.8.91:9066 check port 48700 inter 5s rise 2 fall 3
server mycat_92 192.168.8.92:9066 check port 48700 inter 5s rise 2 fall 3
srvtimeout 20000
```

haproxy 记录日志

默认haproxy 是不记录日志的，为了记录日志还需要配置syslog 模块，在linux 下是rsyslogd 服务，先安装rsyslog

```
yum -y install rsyslog
```

然后 记录haproxy 日志的配置

```
cd /etc/rsyslog.d/
```

如果没有这个目录，新建

```
cd /etc
mkdir rsyslog.d
```

```
cd /etc/rsyslog.d/
touch haproxy.conf
vi /etc/rsyslog.d/haproxy.conf
```

内容如下：

```
$ModLoad imudp
$UDPServerRun 514
local0.* /var/log/haproxy.log
```

配置rsyslog:

```
vi /etc/rsyslog.conf
```

1、在#### RULES ####上面一行的地方加入以下内容：

```
#Include all config files in /etc/rsyslog.d/

$IncludeConfig /etc/rsyslog.d/*.conf

#### RULES ####
```

2、在local7.* /var/log/boot.log 的下面加入以下内容（增加后的效果如下）：

```
# Save boot messages also to boot.log

local7.* /var/log/boot.log
local0.* /var/log/haproxy.log
```

保存，重启rsyslog 服务

```
service rsyslog restart
```

现在你就可以看到日志（/var/log/haproxy.log）了

配置监听mycat 是否存活

在Mycat server1 Mycat server2 上都需要添加检测端口48700 的脚本，为此需要用到xinetd，xinetd 为linux 系统的基础服务。首先在xinetd 目录下面增加脚本与端口的映射配置文件 1、如果xinetd 没有安装，使用如下命令安装：

```
yum install xinetd -y
```

2、检查/etc/xinetd.conf 的末尾是否有这一句：includedir /etc/xinetd.d 没有就加上

3、检查/etc/xinetd.d 文件夹是否存在，不存在也加上

```
cd /etc
mkdir xinetd.d
```

4、增加/etc/xinetd.d/mycat_status 监听mycat 是否存活的配置,执行以下命令：

```
cd /etc
mkdir xinetd.d
cd /etc/xinetd.d/
touch mycat_status
vim /etc/xinetd.d/mycat_status
```

内容如下：

```
service mycat_status
{
flags = REUSE
socket_type = stream
port = 48700
wait = no
user = root
server =/usr/local/bin/mycat_status
log_on_failure += USERID
disable = no
}
```

5、/usr/local/bin/mycat_status 脚本 内容如下：

```
#!/bin/bash
#/usr/local/bin/mycat_status.sh
```

```
#This script checks if a mycat server is healthy running on localhost. It
will

#return:
#"HTTP/1.x 200 OK\r" (if mycat is running smoothly)

#"HTTP/1.x 503 Internal Server Error\r" (else)

mycat=`/usr/local/mycat/bin/mycatstatus |grep'not running'| wc -l`
if [ "$mycat" = "0" ];
then
/bin/echo-e"HTTP/1.1 200 OK\r\n"
else
/bin/echo-e"HTTP/1.1 503 Service Unavailable\r\n"
fi
```

6、/etc/services 中加入mycat_status 服务

```
cd /etc
vi services
```

在末尾加入以下内容：

```
mycat_status 48700/tcp # mycat_status
```

保存，重启xinetd 服务

```
service xinetd restart
```

7、验证mycat_status 服务是否启动成功

```
netstat -antup|grep 48700
```

如果成功会显示如下内容：

```
[root@localhost log]# netstat -antup|grep 48700
tcp 0 0 :::48700 :::* LISTEN 12609/xinetd
```

启动haproxy

启动haproxy 前必须先启动keepalived，否则启动不了。 启动命令：

```
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/haproxy.cfg
```

启动haproxy 异常情况 如果报以下错误：

```
[root@localhost bin]# /usr/local/haproxy/sbin/haproxy -f
/usr/local/haproxy/haproxy.cfg
[ALERT] 183/115915 (12890) :Starting proxy admin_status: cannot bind
socket
[ALERT] 183/115915 (12890) :Starting proxy allmycat_service: cannot bind
socket
[ALERT] 183/115915 (12890) :Starting proxy allmycat_admin: cannot bind
socket
```

原因为：该机器没有抢占到vip，如果另一台服务启动正常，这个错误可以忽略不管，如果另一台也一样，使 用ping vip 命令看看vip 是否生效，如果没有生效，说明keepalived 没有启动成功，回去检查keepalived 的异 常再说。

为了使用方便可以增加一个启动，停止haproxy 的脚本

```
touch /usr/local/haproxy/sbin/starthaproxy
chmod +x /usr/local/haproxy/sbin/starthaproxy
touch /usr/local/haproxy/sbin/stophaproxy
chmod +x /usr/local/haproxy/sbin/stophaproxy
```

启动脚本starthap 内容如下：

```
#!/bin/sh
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/haproxy.cfg &
```

停止脚本stophap 内容如下

```
#!/bin/sh
ps -ef | grep sbin/haproxy | grep -v grep |awk '{print $2}'|xargs kill -s
9
```

启动后可以通过http://192.168.8.108:48800/admin-status (用户名密码都是admin，haproxy.cfg 中配置的)

openssl 安装

openssl 必须安装，否则安装keepalived 时无法编译，keepalived 依赖openssl。

```
tar zxvf openssl-1.0.1g.tar.gz
./config--prefix=/usr/local/openssl
./config-t
make depend
make
make test
make install
ln -s /usr/local/openssl /usr/local/ssl
```

openssl 配置

```
vi /etc/ld.so.conf
```

在/etc/ld.so.conf 文件的最后面，添加如下内容：

```
/usr/local/openssl/lib
```

配置环境变量：

```
vi /etc/profile
```

内容如下：

```
export OPENSSL=/usr/local/openssl/bin
export PATH=$PATH:$OPENSSL
```

执行以下语句是环境变量生效：

```
source /etc/profile
```

安装openssl-devel：

```
yum install openssl-devel -y #如无法yum 下载安装，请修改yum 配置文件
```

测试:

```
ldd /usr/local/openssl/bin/openssl

    linux-vdso.so.1 => (0x00007fff996b9000)
    libdl.so.2 =>/lib64/libdl.so.2 (0x00000030efc00000)
    libc.so.6 =>/lib64/libc.so.6 (0x00000030f0000000)
    /lib64/ld-linux-x86-64.so.2 (0x00000030ef800000)

which openssl
    /usr/bin/openssl
openssl version
    OpenSSL 1.0.0-fips 29 Mar 2010
```

keepalived 安装

本文在192.168.8.91、192.168.8.92 两台机器进行keepalived 安装 安装

```
tar zxvf keepalived-1.2.13.tar.gz
cd keepalived-1.2.13
./configure--prefix=/usr/local/keepalived
make
make install
cp /usr/local/keepalived/sbin/keepalived /usr/sbin/
cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/
cp /usr/local/keepalived/etc/rc.d/init.d/keepalived/etc/init.d/
mkdir /etc/keepalived
cd /etc/keepalived/
cp /usr/local/keepalived/etc/keepalived/keepalived.conf/etc/keepalived
mkdir-p/usr/local/keepalived/var/log
```

keepalived 配置

创建检查haproxy 是否存活的脚本：

```
mkdir /etc/keepalived/scripts
cd /etc/keepalived/scripts
```

配置 keepalived.conf:

```
vi /etc/keepalived/keepalived.conf
```

Master(192.168.8.91):

```
! Configuration Fileforkeepalived
vrrp_script chk_http_port {
script"/etc/keepalived/scripts/check_haproxy.sh"
interval 2
weight 2
}
vrrp_instance VI_1 {
state MASTER #192.168.8.92 上为BACKUP
interface eth0 #对外提供服务的网络接口
virtual_router_id 51 #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,192.168.8.92 上改为120
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_http_port #调用脚本check_haproxy.sh 检查haproxy 是否存活
}
virtual_ipaddress { #vip 地址，这个ip 必须与我们在lvs 客户端设定的vip 相一致
192.168.8.108 dev eth0 scope global
}
notify_master/etc/keepalived/scripts/haproxy_master.sh
notify_backup/etc/keepalived/scripts/haproxy_backup.sh
notify_fault /etc/keepalived/scripts/haproxy_fault.sh
notify_stop /etc/keepalived/scripts/haproxy_stop.sh
}
```

slave(192.168.8.92)：

```
! Configuration Fileforkeepalived
vrrp_script chk_http_port {
script"/etc/keepalived/scripts/check_haproxy.sh"
interval 2
weight 2
}
vrrp_instance VI_1 {
state BACKUP
interface eth0 #对外提供服务的网络接口
```

```
virtual_router_id 51 #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 120 #数值愈大，优先级越高
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_http_port #调用脚本check_haproxy.sh 检查haproxy 是否存活
}
virtual_ipaddress { #vip 地址，这个ip 必须与我们在lvs 客户端设定的vip 相一致
192.168.8.108 dev eth0 scope global
}
notify_master /etc/keepalived/scripts/haproxy_master.sh
notify_backup /etc/keepalived/scripts/haproxy_backup.sh
notify_fault /etc/keepalived/scripts/haproxy_fault.sh
notify_stop /etc/keepalived/scripts/haproxy_stop.sh
}
```

注意：

1. virtual_router_id 51 这个代表一个集群组，如果同一个网段还有另一组集群，请使用不同的组编号区分。 如换成52、53 等。
2. interface eth1 和192.168.8.108 dev eth1 scope global 中的eth1 指的是网卡，如果是多网卡，可能 会有eth0，eth1，eth2...，可以使用ifconfig 命令查看，确保eth0 是本机存在的网卡地址。有些服务器如果只有一个网卡，但被人为把eth0 改成eth1 了，你再写eth0 就找不到了的。

check_haproxy.sh

```
vi /etc/keepalived/scripts/check_haproxy.sh
```

脚本含义：如果没有haproxy 进程存在，就启动haproxy，停止keepalived。
check_haproxy.sh

```
#!/bin/bash
STARTHAPROXY="/usr/local/haproxy/sbin/haproxy -f
/usr/local/haproxy/haproxy.cfg"
STOPKEEPALIVED="/etc/init.d/keepalived stop"
```

```
LOGFILE="/usr/local/keepalived/var/log/keepalived-haproxy-state.log"
echo "[check_haproxy status]" >> $LOGFILE
A=`ps-C haproxy --no-header |wc-l`
echo "[check_haproxy status]" >> $LOGFILE
date >> $LOGFILE
if [ $A -eq 0 ];then
echo $STARTHAPROXY >> $LOGFILE
$STARTHAPROXY >> $LOGFILE 2>&1
sleep5
fi
if [ `ps -C haproxy --no-header |wc-l` -eq0 ];then
exit 0
else
exit 1
fi
```

haproxy_master.sh(master 和slave 一样)

/etc/keepalived/scripts/haproxy_master.sh

```
#!/bin/bash
STARTHAPROXY=`/usr/local/haproxy/sbin/haproxy-
f/usr/local/haproxy/haproxy.cfg`
STOPHAPROXY=`ps-ef |grep sbin/haproxy| grep -vgrep|awk'{print
$2}'|xargskill-s 9`
LOGFILE="/usr/local/keepalived/var/log/keepalived-haproxy-state.log"
echo "[master]" >> $LOGFILE
date >> $LOGFILE
echo "Being master...." >> $LOGFILE 2>&1
echo "stop haproxy...." >> $LOGFILE 2>&1
$STOPHAPROXY >> $LOGFILE 2>&1
echo "start haproxy...." >> $LOGFILE 2>&1
$STARTHAPROXY >> $LOGFILE 2>&1
echo "haproxy stared ..." >> $LOGFILE
```

haproxy_backup.sh(master 和slave 一样)

/etc/keepalived/scripts/haproxy_backup.sh

```
#!/bin/bash
STARTHAPROXY=`/usr/local/haproxy/sbin/haproxy-
f/usr/local/haproxy/haproxy.cfg`
STOPHAPROXY=`ps-ef |grep sbin/haproxy| grep -vgrep|awk'{print
$2}'|xargskill-s 9`
LOGFILE="/usr/local/keepalived/var/log/keepalived-haproxy-state.log"
echo "[backup]" >> $LOGFILE
date >> $LOGFILE
echo "Being backup...." >> $LOGFILE 2>&1
echo "stop haproxy...." >> $LOGFILE 2>&1
$STOPHAPROXY >> $LOGFILE 2>&1
echo "start haproxy...." >> $LOGFILE 2>&1
$STARTHAPROXY >> $LOGFILE 2>&1
echo "haproxy stared ..." >> $LOGFILE
```

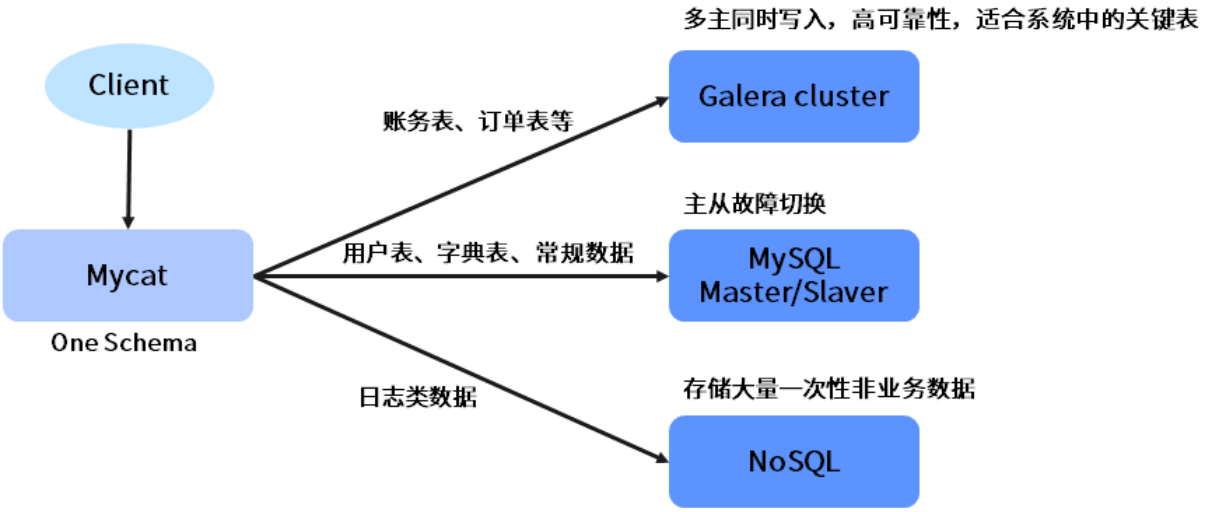
haproxy_fault.sh(master 和slave 一样) /etc/keepalived/scripts/haproxy_fault.sh

```
#!/bin/bash
LOGFILE=/usr/local/keepalived/var/log/keepalived-haproxy-state.log
echo "[fault]" >> $LOGFILE
date >> $LOGFILE
haproxy_stop.sh(master 和slave 一样)
/etc/keepalived/scripts/haproxy_stop.sh
#!/bin/bash
LOGFILE=/usr/local/keepalived/var/log/keepalived-haproxy-state.log
echo "[stop]" >> $LOGFILE
date >> $LOGFILE
```

启用服务

```
service keepalived start
```

4.3 Mycat多类数据源完整解决方案



4.4 Mycat配置Zookeeper化

Mycat1.5开始支持配置zookeeper化，通过zookeeper管理Mycat-Server 。1.6 对zk 模块进行了重构，同时支持zk 的watch 机制，会将所有zk 上的变动同步到本地配置。

4.4.1 配置zk化步骤

1、配置myid.properties

```
loadZk=true      是否启用zk
zkURL=127.0.0.1:2181     zk地址支持多个
clusterId=mycat-cluster-1      mycat集群名称，即一组相同的mycat为一个集群，一个集群名称配置唯一
myid=mycat_fz_01     集群内部mycat 节点的id，请保持集群内唯一
```

2、在一个mycat实例上，在conf 下的zkconf 下配置常用的schemal rule server 等xml 文件

3、在该mycat实例上执行bin 目录下的init_zk_data.bat 或者init_zk_data.sh,会自动将zkconf 下的所有配置文件上传到zk。

4.其他mycat实例，启动时如果loadzk=true 会自动从zk 下载配置文件覆盖本地配置。

4.4.2 zk 协调后端mysql 切换

server.xml 中配置

```
<property name="useZKSwitch">true</property>
```

则mycat 在进行切换时会自动通过zk协调，保证同一个集群下的mycat 都切换到一致的状态

4.5 Mycat 性能测试指南

Mycat 自身提供了一套基准性能测试工具，这套工具可以用于性能测试、疲劳测试等，包括分片表插入性能 测试、分片表查询性能测试、更新性能测试、全局表插入性能测试等基准测试工具。

这里需要说明的一点是，分片表的性能测试不同于普通单表，因为它的数据是分布在几个Datahost 上的，因 此插入和查询，都必需要特定的工具，才能做到多个节点同时负载请求，通过观察每个主机的负载，能够确定是 否你的测试是合理和正确的。

大量测试表明，当带宽不是问题而且带宽没有占满，比如千兆网网络连接的Mycat 和MySQL 服务器，以及 测试客户端，（通常个人电脑到服务器的连接为100M），分片表的性能取决于后端部署MySQL 的物理机的个 数，比如每个MySQL 的性能是5 万Tps，则3 台理论上是15 万，而Mycat 能达到80-95%之间，即12 万以上。

关于带宽问题，是一个比较棘手的问题，通常需要监控交换机、MySQL 服务器、Mycat 服务器、以获取测试 过程中的端口流量信息，才能确定是否带宽存在问题，另外，很多企业里，千兆交换机采用了百兆的普通网线的 情况时有发生，防不胜防，所以，在不能控制的网络环境里，测试最大性能的目标通常无法实现。

另外，很多人测试的时候，并不知道MySQL 直连的性能，因此无法正确比较Mycat 的性能，所以，建议性 能测试过程里，首先直连MySQL 进行性能测试，可以同时直连多个MYSQL 服务器，然后把测试结果累计，作 为直连的性能指标，然后改为连接Mycat 进行测试，这样的对比才是有价值的，当插件过大的时候，需要先排除 是否存在MySQL 冷热不均的现象，然后考虑Mycat 性能调优。

测试工具获取

测试工具在单独的包中，解压到任意机器中执行使用，跟MyCAT Server 没有关联关系，此测试工具很强 大，可以测试任意表，和任意数据库，测试工具下载：https://github.com/MyCATApache/Mycat-download 目录下的testtool.tar.gz 中。

解压后，在bin 目录里运行文中的测试脚本:

标准插入性能测试脚本

标准插入性能测试脚本test_stand_insert_perf.sh 支持任意表的定制化业务数据的随机生成功能了，在sql 模板文件中用${int(1-100)}这种变量，测试程序会随机生成符合要求的值并插入数据库。

```
./test_stand_insert_perf.sh jdbc:mysql://localhost:8066/TESTDB test test
10 file=mydata-create.sql
```

其中mydata-create.sql 的内容如下：

```
total=10000000
sql=insert into my_table1 (….) values ('${date(yyyyMMddHHmmssSSS-[2014-
2015]y)}-${int(0-
9999)}ok${int(1111-9999)}xxx ','${char([0-9]2:2)}
OPP_${enum(BJ,SH,WU,GZ)}_1',10,${int(10-999)},${int(10-
99)},100,3,15,'${date(yyyyMMddHHmmssSSS-[2014-2015]y}${char([a-f,0-9]8:8)}
',${phone(139-
189)},2,${date(yyyyMMddHH-[2014-2015]y},${date(HHmmssSSS)},${int(100-
1000)},'${enum(0000,0001,0002)}')
```

目前支持的有以下类型变量：

- Int:${int(..)} 可以是,${int(10-999)}或者,${int(10,999)}前者表示从10 到999 的值，后者表示10 或者999
- Date:日期如${date(yyyyMMddHHmmssSSS-[2014-2015]y)}表示从2014 到2015 年的时间，前面是输出格式，符 合Java 标准
- Char:字符串${char([0-9]2:2)}表示从0 到9 的字符，长度为2 位（ 2:2 ），}${char([a-f,0-9]8:8)}表示从a 到f 以及0 到9 的字符串随机组成，定常为8 位。
- Enmu:枚举，表示从指定范围内获取一个值，${enum(0000,0001,0002)}，里面可以是任意字符串或数字等内容。

标准查询性能测试脚本

标准查询性能测试脚本test_stand_select_perf 也支持sqlTemplate 的变量方式，查询任意指定的sql

```
./test_stand_select_perf.sh jdbc:mysql://localhost:8066/TESTDB test test
10 100000 file=mysql-select.sql
```

其中oppcall-select.sql 的内容类似下面:

```
sql=select * from mytravelrecord where id = ${int(1-1000000)}
```

表明查询id 为1 到1000000 之间的随机SQL。

注意：Windows 下file=xxx.slq 需要加引号：

```
test_stand_insert_perf.bat jdbc:mysql://localhost:8066/TESTDB test test 50
"file=oppcall.sql"
```

测试说明

首先参考MyCAT 性能调优，确保整个系统达到最优配置。

性能测试，建议先小规模压力预热10-20 分钟，这是众所周知的Java 的特性，越跑越快。

测试的硬件和网络条件：

- 建议至少3 台服务器；
- MyCAT Server 一台；
- Mysql 一台；
- 带宽应该是至少100M，建议千兆；
- 压力程序在另一台，压力程序的机器也可以由性能差的机器来代替。

有条件的话，分片库在不同的MYSQL 实例上，如20 个分片，每个MYSQL 实例7 个分片，而且最好有多台 MYSQL 物理机。

分片表的录入性能测试-T01

测试案例：分片表的并发录入性能测试，测试DEMO 中的travelrecord 表，此表的基准DDL 语句：

```
create table travelrecord (
    id bigint not null primary key,
    user_id varchar(100),
    traveldate DATE,
    fee decimal,
    days int);
```

此表的标准分片方式为基于ID 范围的自动分片策略。Schema.xml 中配置如下：

```
<table name="travelrecord" dataNode="dn1,dn2,dn3" rule="auto-sharding-
long" />
```

默认是3 个分片，分片ID 范围定义在autopartition-long.txt 中，建议修改为以下或更大的数值范围分片， 每个分片500 万数据

```
#range start-end ,data node index

0-2000000=0
2000001-4000000=1
4000001-6000000=2
```

根据自己的情况，可以每个分片放更多的数据，进行对比性能测试，当分片index 增加时，注意dataNode 也增加（ dataNode=“dn1,dn2,dn3” ）。

测试的输入参数如下[jdbcurl] [user] [password] [threadpoolsize] [recordrange]：

- Jdbcurl:连接mycat 的地址，格式为jdbc:mysql://localhost:8066/TESTDB
- User 连接Mycat 的用户名
- Password:密码
- Threadpoolsize:并发线程请求，可以在50-2000 左右调整，看看哪种情况下的性能最好
- Recordrang:插入的分片系列以及对应的ID 范围，minId-maxId 然后逗号分开，对应多组分片的ID 范围，如0-200000,200001-400000,400001-600000，跟分片配置保持一致。

测试过程：

每次测试，建议先执行重建表的操作，以保证测试环境的一致性： 连接mycat 8066 端口，在命令行执行下面的操作：

```
drop table travelrecord;
create table travelrecord (
    id bigint not null primary key,
    user_id varchar(100),
    traveldate DATE,
    fee decimal,
    days int);
```

先预测试： 执行命令：

```
test_stand_insert_perf jdbc:mysql://localhost:8066/TESTDB test test 100
“0-100M,100M1-200M,200M1-400M”
```

MyCAT 温馨提示：并发线程数表明同时至少有多少个Mysql 连接会被打开，当SQL 不跨分片的时候，并发线程数 =MYSQL 连接数，在Mycat conf/schema.xml 中，将minCon 设置为>=并发连接数，这种情况下重启MYCAT，会 初始建立minCon 个连接，并发测试结果更好，另外，也可以验证是否当前内存设置，以及MYSQL 是否支持开启这么多连接，若无法支持，则 logs/mycat.log 日志中会有告警错误信息，建议测试过程中tail –f logs/mycat.log 观察有无错误信息。另外，开启单独的Mycat 管理窗口，mysql –utest –ptest –P9066 然后运行show @@datasource 可以看到后端连接的使用情况。Show @@threadpool 可以看线程和SQL 任务积压的情况。 也可以同时启动多个测试程序,在不同的机器上，并发进行测试，每个测试程序写入一个分片的数据范围，对于1 个亿的数据插入测试来说，可能效果更好，毕竟单机并发线程 50 个左右已经差不多极限： test_stand_insert_perf jdbc:mysql://localhost:8066/TESTDB test test 100 “0-100M” est_stand_insert_perf jdbc:mysql://localhost:8066/TESTDB test test 100 100M1-200M”

全局表的查询性能测试T02：

全局表自动在多个节点上同步插入，因此其插入性能有所降低，这里的插入表为goods 表，执行的命令类似 T01 的测试。温馨提示：全局表是同时往多个分片上写数据，因此所需并发 MYSQL 数连接为普通表的3 倍，最 好的模式是全局表分别在多个mysql 实例上。 建表语句：

```
drop table goods;
create table goods(
    id int not null primary key,
    name varchar(200),
    good_type tinyint,
    good_img_url varchar(200),
    good_created date,
    good_desc varchar(500),
    price double);
```

```
test_globaltable_insert_perf.bat jdbc:mysql://localhost:8066/TESTDB test
test 100 1000000
```

本机笔记本，4G 内存，数据库与Mycat 以及测试程序都在一起，跑出来每秒1000 多的插入速度。

分片表的查询性能测试T03：

此测试可以在T01 的集群上运行，先生成大量travelrecord 记录，然后进行并发随机查询，此测试是在分片 库上，基于分片的主键ID 进行随机查询，返回单条记录，多线程并发随机执行N 此记录查询，每次查询的记录主 键ID 是随机选择，在maxID(参数)范围之内。

测试工具test_stand_select_perf 的参数如下

[jdbcurl] [user] [password] [threadpoolsize] [executetimes] [maxId]

- Executetimes：每个线程总共执行多少次随机查询，建议1000 次以上
- maxId：travelrecord 表的最大ID，可以执行select max(id) from travelrecord 来获取。

Example:

```
test_stand_select_perf.bat jdbc:mysql://localhost:8066/TESTDB test test
100 10000 50000
```

分片表的汇聚性能测试T04：

此测试可以在T01 的集群上运行，先生成大量travelrecord 记录，然后进行并发随机查询，此测试执行分片 库上的聚合、排序、分页的性能，SQL 如下：

```
select sum(fee) total_fee, days,count(id),max(fee),min(fee) from
travelrecord group by days order by days desc limit ?
```

测试工具test_stand_merge_sel_perf 的参数如下

[jdbcurl] [user] [password] [threadpoolsize] [executetimes] [limit]

- Executetimes：每个线程总共执行多少次随机查询，建议1000 次以上
- limit：分页返回的记录个数，必须大于30

Example:

```
test_stand_merge_sel_perf.bat jdbc:mysql://localhost:8066/TESTDB test test
10 100 100
```

分片表的更新性能测试T05：

此测试可以在T01 的集群上运行，先生成大量travelrecord 记录，然后进行并发更新操作。

```
update travelrecord set user =? ,traveldate=?,fee=?,days=? where id=?
```

测试工具test_stand_update_perf 的参数如下

[jdbcurl] [user] [password] [threadpoolsize] [record]

- record：总共修改多少条记录，>5000

Example:

```
test_stand_update_perf.bat jdbc:mysql://localhost:8066/TESTDB test test 10
10000
```

4.6 监控与管理

4.6.1 管理命令与监控

请参考《Mycat权威指南》 高级进阶篇 第8章 “管理命令与监控”。

4.6.2 Mycat-Web

请参考《Mycat权威指南》 高级进阶篇 第10章 “Mycat-Web” 和 https://github.com/MyCATApache/Mycat-Web

4.6.3 Mycat-mini-monitor

使用参考：https://github.com/MyCATApache/Mycat-mini-monitor