一、安装Percona数据库

1. 离线安装Percona

- 进入RPM安装文件目录，执行下面的脚本

```
yum localinstall *.rpm
```

- 管理MySQL服务

```
systemctl start mysqld
systemctl stop mysqld
systemctl restart mysqld
```

2. 在线安装Percona

- 使用yum命令安装

```
yum install http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm
yum  install  Percona-Server-server-57
```

- 管理MySQL服务

```
service mysql start
service mysql stop
service mysql restart
```

3. 开放防火墙端口

```
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --reload
```

4. 修改MySQL配置文件

```
vi /etc/my.cnf
```

```
[mysqld]
character_set_server = utf8
bind-address = 0.0.0.0
#跳过DNS解析
skip-name-resolve
```

```
service mysql restart
```

5. 禁止开机启动MySQL

```
chkconfig mysqld off
```

6. 初始化MySQL数据库

- 查看MySQL初始密码

  ```
  cat /var/log/mysqld.log | grep "A temporary password"
  ```

- 修改MySQL密码

  ```
  mysql_secure_installation
  ```

- 创建远程管理员账户

  ```
  mysql -u root -p
  ```

  ```
  CREATE USER 'admin'@'%' IDENTIFIED BY 'Abc_123456';
  GRANT all privileges ON *.* TO 'admin'@'%';
  FLUSH PRIVILEGES;
  ```

二、创建PXC集群

1. 删除MariaDB程序包

```
yum -y remove mari*
```

2. 开放防火墙端口

```
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --zone=public --add-port=4444/tcp --permanent
firewall-cmd --zone=public --add-port=4567/tcp --permanent
firewall-cmd --zone=public --add-port=4568/tcp --permanent
```

3. 关闭SELINUX

```
vi /etc/selinux/config
```

把SELINUX属性值设置成disabled

```
reboot
```

4. 离线安装PXC

- 进入RPM文件目录，执行安装命令

  ```
  yum localinstall *.rpm
  ```

- 参考第一章内容，修改MySQL配置文件、创建账户等操作

5. 创建PXC集群

- 停止MySQL服务
- 修改每个PXC节点的/etc/my.cnf文件（在不同节点上，注意调整文件内容）

```
server-id=1  #PXC集群中MySQL实例的唯一ID，不能重复，必须是数字
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so
wsrep_cluster_name=pxc-cluster  #PXC集群的名称
wsrep_cluster_address=gcomm://192.168.99.151,192.168.99.159,192.168.99.215
wsrep_node_name=pxc1  #当前节点的名称
wsrep_node_address=192.168.99.151  #当前节点的IP
wsrep_sst_method=xtrabackup-v2  #同步方法（mysqldump、rsync、xtrabackup）
wsrep_sst_auth= admin:Abc_123456  #同步使用的帐户
pxc_strict_mode=ENFORCING  #同步严厉模式
binlog_format=ROW  #基于ROW复制（安全可靠）
default_storage_engine=InnoDB  #默认引擎
innodb_autoinc_lock_mode=2  #主键自增长不锁表
```

- 主节点的管理命令（第一个启动的PXC节点）

```
systemctl start mysql@bootstrap.service
systemctl stop mysql@bootstrap.service
systemctl restart mysql@bootstrap.service
```

- 非主节点的管理命令（非第一个启动的PXC节点）

```
service start mysql
service stop mysql
service restart mysql
```

- 查看PXC集群状态信息

```
show status like 'wsrep_cluster%' ;
```

- **按照上述配置方法，创建两组PXC集群**

6. PXC节点启动与关闭

- 如果最后关闭的PXC节点是安全退出的，那么下次启动要最先启动这个节点，而且要以主节点启动
- 如果最后关闭的PXC节点不是安全退出的，那么要先修改`/var/lib/mysql/grastate.dat`文件，把其中的`safe_to_bootstrap`属性值设置为1，再安装主节点启动

三、安装MyCat

1. JDK安装与配置

- 安装JDK

```
#搜索JDK版本
yum search jdk
#安装JDK1.8开发版
yum install java-1.8.0-openjdk-devel.x86_64
```

- 配置环境变量

```
#查看JDK安装路径
ls -lrt /etc/alternatives/java
vi /etc/profile
#在文件结尾加上JDK路径，例如export  JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.171-8.b10.el7_5.x86_64/
source  /etc/profile
```

2. 创建数据表

- 在两组PXC集群中分别创建t_user数据表

```
CREATE TABLE t_user(
    id INT UNSIGNED PRIMARY KEY,
    username VARCHAR(200) NOT NULL,
    password VARCHAR(2000) NOT NULL,
    tel CHAR(11) NOT NULL,
    locked TINYINT(1) UNSIGNED NOT NULL DEFAULT 0,
    INDEX idx_username(username) USING BTREE,
    UNIQUE INDEX unq_username(username) USING BTREE
);
```

3. MyCat安装与配置

1. 下载MyCat

   http://dl.mycat.io/1.6.5/Mycat-server-1.6.5-release-20180122220033-linux.tar.gz

2. 上传MyCat压缩包到虚拟机

3. 安装unzip程序包，解压缩MyCat

```
yum install unzip
unzip MyCAT压缩包名称
```

4. 开放防火墙8066和9066端口，关闭SELINUX

5. 修改MyCat的bin目录中所有.sh文件的权限

```
chmod -R 777 ./*.sh
```

6. MyCat启动与关闭

```
#cd MyCat的bin目录
./startup_nowrap.sh #启动MyCat
ps -aux #查看系统进程
kill -9 MyCat进程编号
```

7. 修改server.xml文件，设置MyCat帐户和虚拟逻辑库

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:server SYSTEM "server.dtd">
<mycat:server xmlns:mycat="http://io.mycat/">
    <system>
        <property name="nonePasswordLogin">0</property>
        <property name="useHandshakeV10">1</property>
        <property name="useSqlStat">0</property>
        <property name="useGlobleTableCheck">0</property>
        <property name="sequnceHandlerType">2</property>
        <property name="subqueryRelationshipCheck">false</property>
        <property name="processorBufferPoolType">0</property>
        <property name="handleDistributedTransactions">0</property>
        <property name="useOffHeapForMerge">1</property>
        <property name="memoryPageSize">64k</property>
        <property name="spillsFileBufferSize">1k</property>
```

```
        <property name="useStreamOutput">0</property>
        <property name="systemReserveMemorySize">384m</property>
        <property name="useZKSwitch">false</property>
    </system>
    <!--这里是设置的admin用户和虚拟逻辑库-->
    <user name="admin" defaultAccount="true">
        <property name="password">Abc_123456</property>
        <property name="schemas">test</property>
    </user>
</mycat:server>
```

8. 修改schema.xml文件，设置数据库连接和虚拟数据表

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
    <!--配置数据表-->
    <schema name="test" checkSQLschema="false" sqlMaxLimit="100">
        <table name="t_user" dataNode="dn1,dn2" rule="mod-long" />
    </schema>
    <!--配置分片关系-->
    <dataNode name="dn1" dataHost="cluster1" database="test" />
    <dataNode name="dn2" dataHost="cluster2" database="test" />
    <!--配置连接信息-->
    <dataHost name="cluster1" maxCon="1000" minCon="10" balance="2"
              writeType="1" dbType="mysql" dbDriver="native" switchType="1"
              slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <writeHost host="W1" url="192.168.99.151:3306" user="admin"
                   password="Abc_123456">
            <readHost host="W1R1" url="192.168.99.159:3306" user="admin"
                      password="Abc_123456" />
            <readHost host="W1R2" url="192.168.99.215:3306" user="admin"
                      password="Abc_123456" />
        </writeHost>
        <writeHost host="W2" url="192.168.99.159:3306" user="admin"
                   password="Abc_123456">
            <readHost host="W2R1" url="192.168.99.151:3306" user="admin"
                      password="Abc_123456" />
            <readHost host="W2R2" url="192.168.99.215:3306" user="admin"
```

```
                        password="Abc_123456" />
        </writeHost>
    </dataHost>
    <dataHost name="cluster2" maxCon="1000" minCon="10" balance="2"
              writeType="1" dbType="mysql" dbDriver="native" switchType="1"
              slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <writeHost host="W1" url="192.168.99.121:3306" user="admin"
                   password="Abc_123456">
            <readHost host="W1R1" url="192.168.99.122:3306" user="admin"
                      password="Abc_123456" />
            <readHost host="W1R2" url="192.168.99.123:3306" user="admin"
                      password="Abc_123456" />
        </writeHost>
        <writeHost host="W2" url="192.168.99.122:3306" user="admin"
                   password="Abc_123456">
            <readHost host="W2R1" url="192.168.99.121:3306" user="admin"
                      password="Abc_123456" />
            <readHost host="W2R2" url="192.168.99.123:3306" user="admin"
                      password="Abc_123456" />
        </writeHost>
    </dataHost>
</mycat:schema>
```

9. 修改rule.xml文件，把mod-long的count值修改成2

```
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
    <property name="count">2</property>
</function>
```

10. 重启MyCat

11. 向t_user表写入数据，感受数据的切分

```
USE test;
#第一条记录被切分到第二个分片
INSERT INTO t_user(id,username,password,tel,locked)
VALUES(1,"A",HEX(AES_ENCRYPT('123456','HelloWorld')));
#第二条记录被切分到第一个分片
INSERT INTO t_user(id,username,password,tel,locked)
VALUES(2,"B",HEX(AES_ENCRYPT('123456','HelloWorld')));
```

4. 配置父子表

1. 在conf目录下创建`customer-hash-int`文件，内容如下：

```
101=0
102=0
103=0
104=1
105=1
106=1
```

2. 在rule.xml文件中加入自定义和

```
<function name="customer-hash-int"
        class="io.mycat.route.function.PartitionByFileMap">
    <property name="mapFile">customer-hash-int.txt</property>
</function>
```

```
<tableRule name="sharding-customer">
    <rule>
        <columns>sharding_id</columns>
        <algorithm>customer-hash-int</algorithm>
    </rule>
</tableRule>
```

3. 修改schema.xml文件，添加父子表定义

```
<table name="t_customer" dataNode="dn1,dn2" rule="sharding-customer">
    <childTable name="t_orders" primaryKey="ID" joinKey="customer_id"
                parentKey="id"/>
</table>
```

4. 在MyCat上执行如下SQL：

```
USE test;
CREATE TABLE t_customer(
    id INT UNSIGNED PRIMARY KEY,
    username VARCHAR(200) NOT NULL,
    sharding_id INT NOT NULL
);
CREATE TABLE t_orders(
    id INT UNSIGNED PRIMARY KEY,
    customer_id INT NOT NULL,
    datetime TIMESTAMP DEFAULT CURRENT_TIMSTAMP
);
```

5. 向t_customer表和t_orders表写入数据，查看字表数据跟随父表切分到同一个分片

5. 创建双机热备的MyCat集群

1. 用两个虚拟机实例，各自部署MyCat
2. 用一个虚拟机实例部署Haproxy
    - 安装Haproxy

      ```
      yum install -y haproxy
      ```

    - 编辑配置文件

      ```
      vi /etc/haproxy/haproxy.cfg
      ```

      ```
      global
          log         127.0.0.1 local2
          chroot      /var/lib/haproxy
          pidfile     /var/run/haproxy.pid
          maxconn     4000
          user        haproxy
          group       haproxy
          daemon
          # turn on stats unix socket
          stats socket /var/lib/haproxy/stats

      defaults
      ```

```
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

listen   admin_stats
    bind    0.0.0.0:4001
    mode  http
    stats uri       /dbs
    stats realm  Global\ statistics
    stats auth    admin:abc123456
listen   proxy-mysql
    bind    0.0.0.0:3306
    mode  tcp
    balance  roundrobin
    option  tcplog       #日志格式
    server   mycat_1  192.168.99.131:3306  check  port  8066  maxconn  2000
    server   mycat_2  192.168.99.132:3306  check  port  8066  maxconn  2000
    option  tcpka        #使用keepalive检测死链
```

- 启动Haproxy

```
service haproxy start
```

- 访问Haproxy监控画面

  http://192.168.99.131:4001/dbs

3. 用另外一个虚拟机同样按照上述操作安装Haproxy
4. 在某个Haproxy虚拟机实例上部署Keepalived

- 开启防火墙的VRRP协议

```
#开启VRRP
firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --protocol
vrrp -j ACCEPT
#应用设置
firewall-cmd --reload
```

- 安装Keepalived

```
yum install -y keepalived
```

- 编辑配置文件

```
vim /etc/keepalived/keepalived.conf
```

```
vrrp_instance  VI_1 {
    state  MASTER
    interface  ens33
    virtual_router_id  51
    priority  100
    advert_int  1
    authentication {
        auth_type  PASS
        auth_pass  123456
    }
    virtual_ipaddress {
        192.168.99.133
    }
}
```

- 启动Keepalived

```
service keepalived start
```

- ping 192.168.99.133

5. 在另外一个Haproxy虚拟机上，按照上述方法部署Keepalived
6. 使用MySQL客户端连接192.168.99.133，执行增删改查数据

四、Sysbench基准测试

1. 安装Sysbench

- 在线安装

```
curl -s https://packagecloud.io/install/
repositories/akopytov/sysbench/script.rpm.sh | sudo bash
```

```
yum -y install sysbench
```

- 本地安装
  - 下载压缩文件

    https://codeload.github.com/akopytov/sysbench/zip/1.0
  - 安装依赖包

```
yum install -y automake libtool
yum install -y mysql-devel
```

  - 执行安装

```
#cd sysbench
./autogen.sh
./configure
make
make install
sysbench --version
```

2. 执行测试

- 准备测试库

```
sysbench  /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua --mysql-host=192.168.99.131 --mysql-port=3306 --mysql-user=admin --mysql-password=Abc_123456 --oltp-tables-count=10 --oltp-table-size=100000 prepare
```

- 执行测试

```
sysbench  /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua --mysql-host=192.168.99.131 --mysql-port=3306 --mysql-user=admin --mysql-password=Abc_123456 --oltp-test-mode=complex --threads=10 --time=300 --report-interval=10 run >> /home/mysysbench.log
```

- 清理数据

```
sysbench /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua --mysql-host=192.168.99.131 --mysql-port=3306 --mysql-user=admin --mysql-password=Abc_123456 --oltp-tables-count=10 cleanup
```

五、tpcc-mysql 压力测试

1. 准备工作

- 修改my.cnf配置文件

```
vi /etc/my.cnf
```

pxc_strict_mode=DISABLED

- 修改某个Haproxy的配置文件

```
server   mysql_1  192.168.99.151:3306  check  port  3306  weight  1  maxconn  2000
server   mysql_2  192.168.99.159:3306  check  port  3306  weight  1  maxconn  2000
server   mysql_3  192.168.99.215:3306  check  port  3306  weight  1  maxconn  2000
```

- 重新启动Haproxy
- 安装依赖程序包

```
yum install -y gcc
yum install -y mysql-devel
```

2. 安装tpcc-mysql

- 下载压缩包

  https://codeload.github.com/Percona-Lab/tpcc-mysql/zip/master

- 执行安装

```
#cd tpcc的src目录
make
```

- 执行`create_table.sql`和`add_fkey_idx.sql`两个文件
- 执行数据初始化

```
./tpcc_load -h 192.168.99.131 -d tpcc -u admin -p Abc_123456 -w
```

- 执行压力测试

```
./tpcc_start -h 192.168.99.131 -d tpcc -u admin -p Abc_123456 -w 1 -c 5 -r 300 -l
600 ->tpcc-output-log
```

六、导入数据

1. 生成1000万条数据

```
import java.io.FileWriter
import java.io.BufferedWriter

class Test {
    def static void main(String[] args) {
        var writer=new FileWriter("D:/data.txt")
        var buff=new BufferedWriter(writer)
        for(i:1..10000000){
```

```
            buff.write(i+",测试数据\n")
        }
        buff.close
        writer.close
    }
}
```

2. 执行文件切分

- 上传data.txt文件到linux
- 执行文件切分

```
split -l 1000000 -d data.txt
```

3. 准备数据库

- 每个PXC分片只开启一个节点
- 修改PXC节点文件，然后重启PXC服务

```
innodb_flush_log_at_trx_commit = 0
innodb_flush_method = O_DIRECT
innodb_buffer_pool_size = 200M
```

- 创建t_test数据表

```
CREATE TABLE t_test(
    id INT UNSIGNED PRIMARY KEY,
    name VARCHAR(200) NOT NULL
);
```

- 配置MyCat

```
<table name="t_test" dataNode="dn1,dn2" rule="mod-long" />
```

```
<dataHost name="cluster1" maxCon="1000" minCon="10" balance="0" writeType="1"
          dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <writeHost host="W1" url="192.168.99.151:3306" user="admin"
               password="Abc_123456"/>
</dataHost>
<dataHost name="cluster2" maxCon="1000" minCon="10" balance="0" writeType="1"
          dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <writeHost host="W1" url="192.168.99.121:3306" user="admin"
               password="Abc_123456"/>
</dataHost>
```

4. 执行Java程序，多线程导入数据

```
import org.eclipse.xtend.lib.annotations.Accessors
import java.io.File
import java.sql.DriverManager

class Task implements Runnable{
    @Accessors
    File file;

    override run() {
        var url="jdbc:mysql://192.168.99.131:8066/test"
        var username="admin"
        var password="Abc_123456"
        var con=DriverManager.getConnection(url,username,password)
        var sql='''
            load data local intfile '/home/data/«file.name»' ignore into table t_test
            character set 'utf8'
            fields terminated by ',' optionally enclosed by '\"'
            lines terminated by '\n' (id,name);
        '''
        var pst=con.prepareStatement(sql);
        pst.execute
        con.close
        LoadData.updateNum();
    }
```

```
}
```

```
import com.mysql.jdbc.Driver
import java.sql.DriverManager
import java.util.concurrent.LinkedBlockingQueue
import java.util.concurrent.ThreadPoolExecutor
import java.util.concurrent.TimeUnit
import java.io.File

class LoadData {
    var static int num=0;
    var static int end=0;
    var static pool=new ThreadPoolExecutor(1,5,60,TimeUnit.SECONDS,new
LinkedBlockingQueue(200))
    def static void main(String[] args) {
        DriverManager.registerDriver(new Driver)
        var folder=new File("/home/data")
        var files=folder.listFiles
        end=files.length //线程池结束条件
        files.forEach[one|
            var task=new Task();
            task.file=one;
            pool.execute(task)
        ]
    }
    synchronized def static updateNum(){
        num++;
        if(num==end){
            pool.shutdown();
            println("执行结束")
        }
    }
}
```

七、大数据归档

1. 安装TokuDB

- 安装jemlloc

```
yum install -y jemalloc
```

- 编辑配置文件

```
vi /etc/my.cnf
```

```
……
[mysqld_safe]
malloc-lib=/usr/lib64/libjemalloc.so.1
……
```

- 重启MySQL
- 开启Linux大页内存

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

- 安装TokuDB

```
yum install -y Percona-Server-tokudb-57.x86_64
ps-admin --enable -uroot -p
service mysql restart
ps-admin --enable -uroot -p
```

- 查看安装结果

```
show engines ;
```

2. 配置Replication集群

- 在两个TokuDB数据库上创建用户

```
CREATE USER 'backup'@'%' IDENTIFIED BY 'Abc_123456' ;
```

```
GRANT super, reload, replication slave ON *.* TO 'backup'@'%' ;
```

```
FLUSH  PRIVILEGES ;
```

- 修改两个TokuDB的配置文件，如下：

```
[mysqld]
server_id = 101
log_bin = mysql_bin
relay_log = relay_bin
……
```

```
[mysqld]
server_id = 102
log_bin = mysql_bin
relay_log = relay_bin
```

- 重新启动两个TokuDB节点
- 分别在两个TokuDB上执行下面4句SQL

```
#关闭同步服务
stop slave;
#设置同步的Master节点
change master to
master_host="192.168.99.155",master_port=3306,master_user="backup",
master_password="Abc_123456";
#启动同步服务
start slave;
#查看同步状态
show slave status;
```

```
#关闭同步服务
stop slave;
#设置同步的Master节点
change master to
master_host="192.168.99.102",master_port=3306,master_user="backup",
master_password="Abc_123456";
#启动同步服务
start slave;
#查看同步状态
show slave status;
```

3. 创建归档表

```
CREATE TABLE t_purchase (
    id INT UNSIGNED PRIMARY KEY,
    purchase_price DECIMAL(10,2) NOT NULL,
    purchase_num INT UNSIGNED NOT NULL,
    purchase_sum DECIMAL (10,2) NOT NULL,
    purchase_buyer INT UNSIGNED NOT NULL,
    purchase_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    company_id INT UNSIGNED NOT NULL,
    goods_id INT UNSIGNED NOT NULL,
    KEY idx_company_id(company_id),
    KEY idx_goods_id(goods_id)
)engine=TokuDB;
```

4. 配置Haproxy+Keepalived双机热备

- 在两个节点上安装Haproxy

```
yum install -y haproxy
```

- 修改配置文件

```
vi /etc/haproxy/haproxy.cfg
```

```
global
```

```
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

listen   admin_stats
    bind    0.0.0.0:4001
    mode  http
    stats uri       /dbs
    stats realm  Global\ statistics
    stats auth    admin:abc123456
listen   proxy-mysql
    bind    0.0.0.0:4002
    mode  tcp
    balance  roundrobin
    option  tcplog       #日志格式
    server   backup_1  192.168.99.102:3306  check  port  3306  maxconn  2000
    server   backup_2  192.168.99.155:3306  check  port  3306  maxconn  2000
    option  tcpka        #使用keepalive检测死链
```

- 重启Haproxy
- 开启防火墙的VRRP协议

```
firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --protocol vrrp -
j ACCEPT
```

```
firewall-cmd --reload
```

- 在两个节点上安装Keepalived

```
yum install -y keepalived
```

- 编辑Keepalived配置文件

```
vim /etc/keepalived/keepalived.conf
```

```
vrrp_instance  VI_1 {
    state  MASTER
    interface  ens33
    virtual_router_id  51
    priority  100
    advert_int  1
    authentication {
        auth_type  PASS
        auth_pass  123456
    }
    virtual_ipaddress {
        192.168.99.211
    }
}
```

- 重启Keepalived

5. 准备归档数据

- 在两个PXC分片上创建进货表

```
CREATE TABLE t_purchase (
    id INT UNSIGNED PRIMARY KEY,
    purchase_price DECIMAL(10,2) NOT NULL,
    purchase_num INT UNSIGNED NOT NULL,
    purchase_sum DECIMAL (10,2) NOT NULL,
    purchase_buyer INT UNSIGNED NOT NULL,
    purchase_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    company_id INT UNSIGNED NOT NULL,
    goods_id INT UNSIGNED NOT NULL,
    KEY idx_company_id(company_id),
    KEY idx_goods_id(goods_id)
)
```

- 配置MyCat的schema.xml文件，并重启MyCat

```
<table name="t_purchase" dataNode="dn1,dn2" rule="mod-long" />
```

6. 执行数据归档

- 安装pt-archiver

```
yum install percona-toolkit
pt-archiver --version
pt-archiver --help
```

- 执行数据归档

```
pt-archiver --source
h=192.168.99.102,P=8066,u=admin,p=Abc_123456,D=test,t=t_purchase --dest
h=192.168.99.102,P=3306,u=admin,p=Abc_123456,D=test,t=t_purchase --no-check-
charset --where 'purchase_date<"2018-09"' --progress 5000 --bulk-delete --bulk-
insert --limit=10000 --statistics
```