**MySQL中间件---MyCAT**

# MyCAT分布式架构

## 高性能架构演变

### 读写分离：

def

对新功能提高很多，

desc

代表产品：ATLAS

--不足：不考虑业务的一种均衡策略

--极端情况：少量节点被频繁命中，可能热点表，对写操作无提升

### 传统分布式策略:

**代表产品:atlas-sharding**

**优点:读写性能都进行了提升**

**缺点:无法实现垮分片访问,无法垮分片join**

**分片策略较少,几乎没有选择性.局限性较大.**

### 企业级分布式策略

#### 需求

1.数据量要均匀

2.业务访问要均匀

3.高可用策略

4.耦合度低

5.无缝请求所有的业务==单节点SQL功能等同于mysql服务器SQL功能（join等）

6.可选择多种分片策略

### MyCAT

#### def

2013阿里cobar出现严重问题，改良版cobar出现，即mycat。Mycat开源以后，一些Cobar的用户参与了Mycat的开发，最终Mycat发展成为一个由众多软件公司的实力派架构师和资深开发人员维护的社区型开源软件。

#### 功能特性

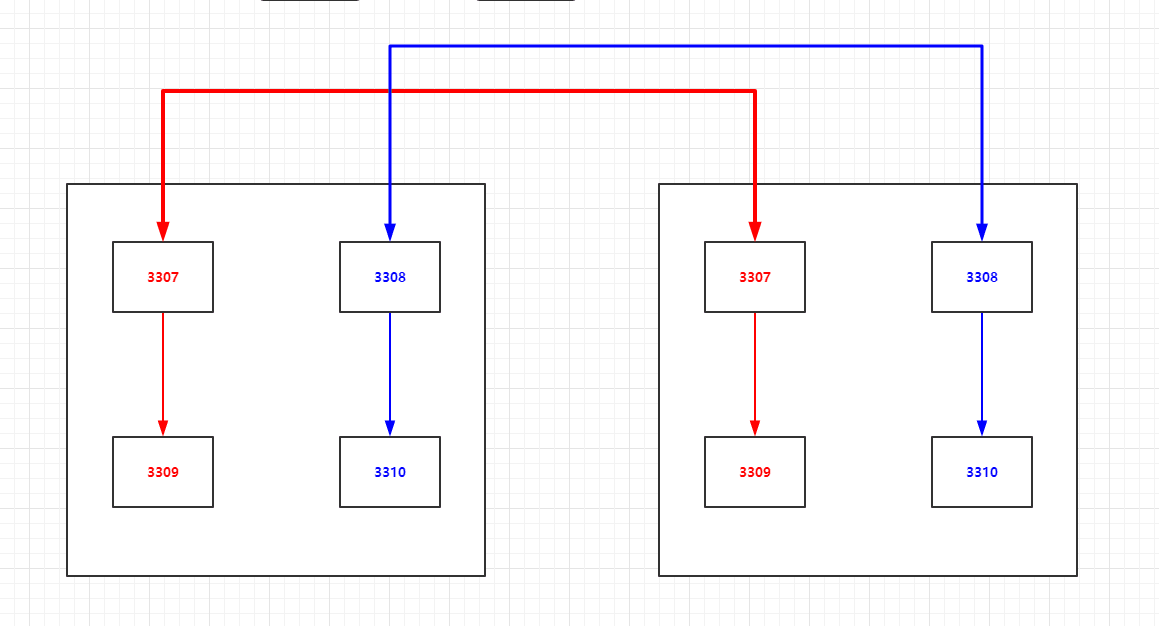
1.支持多种mysql集群，mysql/mogodb/oracle/sqlserver

2.自动故障切换，高可用性

3.支持读写分离，支持Mysql双主多从，以及一主多从的模式

# mycat-主从复制工作原理

## mycat两主两从规划图



### consist

1.两台虚拟主机

db02 10.0.0.52

db03 10.0.0.53

2.db02和db03上的配置

2.1db02和db03两个主从配置：

---mysql服务端口为3307和3309的主从，3308和3310构成的主从，期中3307和3308作为主库

2.2db02和db03上主库配置

---将db02上的prot为3307mysql和db03上的prot为3307mysql配置为互为主从关系

---将db02上的prot为3308mysql和db03上的prot为3308mysql配置为互为主从关系

### 功能

1.db02上3307端口的mysql作为主库，db03上的3307端口mysql作为standby master==即处于待命状态

2.当db02上3307端口的mysql宕机后，mycat会自动将master切换到db03上3307端口的mysql，端口为3309负责读服务

---db02上3307master恢复后，会变为standby master，db03上3307仍然是负责写服务的master，db02上的3307、3309

和db03上的3309负责读服务

3.双主结构可以分布式的双读双写，但是可能会影响数据一致性，所以架构一般是：master负责写、另一个master stanby，两个从库是读节点

## 读写分离

### 配置文件

#schema.xml文件解读：

#cat schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

### 配置简介

1.层次关系

shcema------🡪table--------🡪datanode-------🡪datahost=writehost+readhost

2.参数解释

<头定义/>

<?xml version="1.0"?>

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

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

<TESTDB=datahost（writehost+readhost）+database/>

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

< schema==逻辑库==库定义；

name=逻辑库名字叫TESTDB；

checkSQLschema=true==表示SQL查询时去掉schema（即TESTDB），避免后端数据库报错；

sqlMaxLimit="100"：最大申请了查询结果数为100/>

<table name="t3" dataNode="sh1,sh2" rule="auto-sharding-long" />

</schema>

<table ==逻辑表定义----所有需要拆分的表都需要在这个标签中定义；

dataNode ==数据节点==数据分片==逻辑表所属dataNode,，须和name对应，定义的sh过多的话，可以定义多个<datanode/>一个\*\*dataNode\*\* 标签就是一个独立的数据分片，如下；

rule==规则名字，rule.xml中定义，路径：/application/mycat/conf/rule.xml

/>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataNode name="sh2" dataHost="oldguo2" database= "world" />

<dataNode ==数据节点==数据分片==逻辑表所属dataNode,，须和name对应，定义的sh过多的话，可以定义多个<datanode/>一个\*\*dataNode\*\* 标签就是一个独立的数据分片，如下；

datahost ==分片属于哪个数据库实例，属性值引用所在标签==datanode定义的name属性

database ==分片属性对应的数据库实例的具体库—因为：分片=实例+具体的库

/>

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">

<datahost ==schema.xml最底层标签，定义具体数据库实例、读写分离配置和心跳语句

name ==唯一表示dataHost标签，共上层标签使用

maxCon ==最大并发连接数

minCon ==mycat在启动之后，会在后端节点上自动开启的连接线程

balance ==负载均衡类型，

---取值‘0’==所有r操作发送到当前可用writehost上

---取值‘1’==所有读操作随机发送到readhost

----取值‘2’==所有读操作随机在writehost和readhost上分发

writeType ==负载均衡类型

---取值‘0’ 所有写操作都发送到可用的writeHost上

- --取值‘1’ 所有写操作都随机的发送到readHost。

----取值‘2’ 所有写操作都随机的在writeHost、readhost分上发

dbType ==指定后端连接的数据库类型

~支持二进制mysql协议，JDBC连接的数据库，如mongodb、oracle、spark等

dbDriver ==后端数据库使用的Driver，可选值有native和JDBC

~native: 支持二进制的mysql协议, 故可以使用mysql和maridb

~JDBC : JDBC 4标准的驱动JAR包放到MYCAT\lib目录下……

switchType属性

-1 表示不自动切换

1 默认值，自动切换

2 基于MySQL主从同步的状态决定是否切换 ，心跳语句为 show slave status

/>

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3309" user="root" password="123" />

<writeHost == 指定写实例

host ==标识不同的实例

url ==后端实例连接地址，~native的dbDriver，则为address:port这种形式，当使用JDBC时则可以这么写：jdbc:mysql://localhost:3306/

user ==后端存储实例所需的用户名称

password ==……密码

/>

<readHost == 指定读实例

host ==表示不同的实例

……

/>

</writeHost>

<writeHost host="db3" url="10.0.0.53:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.53:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.53:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.53:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

## 垂直分表

### 配置文件

[root@db02 ~]# cat /application/mycat/conf/schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

<table name="t3" dataNode="sh1,sh2" rule="auto-sharding-long" />

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataNode name="sh2" dataHost="oldguo2" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.53:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.53:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.53:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.53:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

### schema标签的相关属性：

#### dataNode

该属性用于绑定逻辑库到某个具体的database上，如果定义了这个属性，那么这个逻辑库就不能工作在分库分表模式下了。也就是说对这个逻辑库的所有操作会直接作用到绑定的dataNode上，这个schema就可以用作读写分离和主从切换，具体如下配置:

<schema name="USERDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">

<!—这里不能配置任何逻辑表信息-->

</schema>

那么现在USERDB就绑定到dn1所配置的具体database上，可以直接访问这个database。当然该属性只能配置绑定到一个database上，不能绑定多个dn。

#### checkSQLschema

当该值设置为 true 时，如果我们执行语句\*\*select \* from TESTDB.travelrecord;\*\*则MyCat会把语句修改为\*\*select \* from travelrecord;\*\*。即把表示schema的字符去掉，避免发送到后端数据库执行时报\*\*（ERROR 1146 (42S02): Table ‘testdb.travelrecord’ doesn’t exist）。\*\*不过，即使设置该值为 true ，如果语句所带的是并非是schema指定的名字，例如：\*\*select \* from db1.travelrecord;\*\* 那么MyCat并不会删除db1这个字段，如果没有定义该库的话则会报错，所以在提供SQL语句的最好是不带这个字段。

#### sqlMaxLimit

当该值设置为某个数值时。每条执行的SQL语句，如果没有加上limit语句，MyCat也会自动的加上所对应的值。例如设置值为100，执行\*\*select \* from TESTDB.travelrecord;\*\*的效果为和执行\*\*select \* from TESTDB.travelrecord limit 100;\*\*相同。不设置该值的话，MyCat默认会把查询到的信息全部都展示出来，造成过多的输出。所以，在正常使用中，还是建议加上一个值，用于减少过多的数据返回。当然SQL语句中也显式的指定limit的大小，不受该属性的约束。

### table标签

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

Table 标签定义了MyCat中的逻辑表，所有需要拆分的表都需要在这个标签中定义。

#### name属性

定义逻辑表的表名，这个名字就如同我在数据库中执行create table命令指定的名字一样，同个schema标签中定义的名字必须唯一。

#### dataNode属性

定义这个逻辑表所属的dataNode, 该属性的值需要和dataNode标签中name属性的值相互对应。如果需要定义的dn过多可以使用如下的方法减少配置：

<table name="travelrecord" dataNode="multipleDn$0-99,multipleDn2$100-199" rule="auto-sharding-long" ></table>

<dataNode name="multipleDn" dataHost="localhost1" database="db$0-99" ></dataNode>

<dataNode name="multipleDn2" dataHost="localhost1" database=" db$0-99" ></dataNode>

这里需要注意的是database属性所指定的真实database name需要在后面添加一个，例如上面的例子中，我需要在真实的mysql上建立名称为dbs0到dbs99的database。

#### rule属性

该属性用于指定逻辑表要使用的规则名字，规则名字在rule.xml中定义，必须与tableRule标签中name属性属性值一一对应。

#### primaryKey属性

该逻辑表对应真实表的主键，例如：分片的规则是使用非主键进行分片的，那么在使用主键查询的时候，就会发送查询语句到所有配置的DN上，如果使用该属性配置真实表的主键。难么MyCat会缓存主键与具体DN的信息，那么再次使用非主键进行查询的时候就不会进行广播式的查询，就会直接发送语句给具体的DN，但是尽管配置该属性，如果缓存并没有命中的话，还是会发送语句给具体的DN，来获得数据。

#### type属性

该属性定义了逻辑表的类型，目前逻辑表只有“全局表”和”普通表”两种类型。对应的配置：全局表：global。普通表：不指定该值为globla的所有表。

#### autoIncrement属性

MySQL对非自增长主键，使用last\_insert\_id()是不会返回结果的，只会返回0。所以，只有定义了自增长主键的表才可以用last\_insert\_id()返回主键值。mycat目前提供了自增长主键功能，但是如果对应的mysql节点上数据表，没有定义auto\_increment，那么在mycat层调用last\_insert\_id()也是不会返回结果的。由于insert操作的时候没有带入分片键，mycat会先取下这个表对应的全局序列，然后赋值给分片键。这样才能正常的插入到数据库中，最后使用last\_insert\_id()才会返回插入的分片键值。如果要使用这个功能最好配合使用数据库模式的全局序列。使用autoIncrement=“true” 指定这个表有使用自增长主键，这样mycat才会不抛出分片键找不到的异常。使用autoIncrement=“false” 来禁用这个功能，当然你也可以直接删除掉这个属性。默认就是禁用的。

#### needAddLimit属性

指定表是否需要自动的在每个语句后面加上limit限制。由于使用了分库分表，数据量有时会特别巨大。这时候执行查询语句，如果恰巧又忘记了加上数量限制的话。那么查询所有的数据出来，也够等上一小会儿的。所以，mycat就自动的为我们加上LIMIT 100。当然，如果语句中有limit，就不会在次添加了。这个属性默认为true,你也可以设置成false`禁用掉默认行为。

### childTable标签

childTable标签用于定义E-R分片的子表。通过标签上的属性与父表进行关联

<table name="customer" primaryKey="ID" dataNode="dn1,dn2"

rule="sharding-by-intfile">

<childTable name="orders" primaryKey="ID" joinKey="customer\_id"

parentKey="id">

<childTable name="order\_items" joinKey="order\_id"

parentKey="id" />

</childTable>

<childTable name="customer\_addr" primaryKey="ID" joinKey="customer\_id"

parentKey="id" />

</table>

#### name属性

定义子表的表名。

#### joinKey属性

插入子表的时候会使用这个列的值查找父表存储的数据节点。

#### parentKey属性

属性指定的值一般为与父表建立关联关系的列名。程序首先获取joinkey的值，再通过\*\*parentKey\*\*属性指定的列名产生查询语句，通过执行该语句得到父表存储在哪个分片上。从而确定子表存储的位置。

#### primaryKey属性

同table标签所描述的。

#### needAddLimit属性

同table标签所描述的。

### dataNode标签

<dataNode name="dn1" dataHost="lch3307" database="db1" ></dataNode>

dataNode 标签定义了MyCat中的数据节点，也就是我们通常说所的数据分片。一个\*\*dataNode\*\* 标签就是一个独立的数据分片。例子中所表述的意思为：使用名字为lch3307数据库实例上的db1物理数据库，这就组成一个数据分片，最后，我们使用名字dn1标识这个分片。

#### name属性

定义数据节点的名字，这个名字需要是唯一的，我们需要在table标签上应用这个名字，来建立表与分片对应的关系。

#### dataHost属性

该属性用于定义该分片属于哪个数据库实例的，属性值是引用dataHost标签上定义的name属性。

#### database属性

该属性用于定义该分片属性哪个具体数据库实例上的具体库，因为这里使用两个纬度来定义分片，就是：实例+具体的库。因为每个库上建立的表和表结构是一样的。所以这样做就可以轻松的对表进行水平拆分。

### dataHost标签

作为Schema.xml中最后的一个标签，该标签在mycat逻辑库中也是作为最底层的标签存在，直接定义了具体的数据库实例、读写分离配置和心跳语句。现在我们就解析下这个标签。

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"

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

<heartbeat>select user()</heartbeat>

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

<writeHost host="hostM1" url="localhost:3306" user="root"

password="123456">

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

<!-- <readHost host="hostS1" url="localhost:3306" user="root" password="123456"

/> -->

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

#### name属性

唯一标识dataHost标签，供上层的标签使用。

#### maxCon属性

指定每个读写实例连接池的最大连接。也就是说，标签内嵌套的writeHost、readHost标签都会使用这个属性的值来实例化出连接池的最大连接数。

#### minCon属性

指定每个读写实例连接池的最小连接，初始化连接池的大小。

#### balance属性

负载均衡类型，目前的取值有3种：

1. balance=“0”, 所有读操作都发送到当前可用的writeHost上。

2. balance=“1”，所有读操作都随机的发送到readHost。

3. balance=“2”，所有读操作都随机的在writeHost、readhost上分发。

#### writeType属性

负载均衡类型，目前的取值有3种：

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

2. writeType=“1”，所有写操作都随机的发送到readHost。

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

#### dbType属性

指定后端连接的数据库类型，目前支持二进制的mysql协议，还有其他使用JDBC连接的数据库。例如：mongodb、oracle、spark等。

#### dbDriver属性

指定连接后端数据库使用的Driver，目前可选的值有native和JDBC。使用native的话，因为这个值执行的是二进制的mysql协议，所以可以使用mysql和maridb。其他类型的数据库则需要使用JDBC驱动来支持。如果使用JDBC的话需要将符合JDBC 4标准的驱动JAR包放到MYCAT\lib目录下，并检查驱动JAR包中包括如下目录结构的文件：META-INF\services\java.sql.Driver。在这个文件内写上具体的Driver类名，例如：com.mysql.jdbc.Driver。

### heartbeat标签

这个标签内指明用于和后端数据库进行心跳检查的语句。例如,MYSQL可以使用select user()，Oracle可以使用select 1 from dual等。这个标签还有一个connectionInitSql属性，主要是当使用Oracla数据库时，需要执行的初始化SQL语句就这个放到这里面来。例如：alter session set nls\_date\_format='yyyy-mm-dd hh24:mi:ss'

### writeHost标签、readHost标签

这两个标签都指定后端数据库的相关配置给mycat，用于实例化后端连接池。唯一不同的是，writeHost指定写实例、readHost指定读实例，组着这些读写实例来满足系统的要求。在一个dataHost内可以定义多个writeHost和readHost。但是，如果writeHost指定的后端数据库宕机，那么这个writeHost绑定的所有readHost都将不可用。另一方面，由于这个writeHost宕机系统会自动的检测到，并切换到备用的writeHost上去。这两个标签的属性相同，这里就一起介绍。

#### host属性

用于标识不同实例，一般writeHost我们使用\*M1，readHost我们用\*S1。

#### url属性

后端实例连接地址，如果是使用native的dbDriver，则一般为address:port这种形式。用JDBC或其他的dbDriver，则需要特殊指定。当使用JDBC时则可以这么写：jdbc:mysql://localhost:3306/。

#### user属性

后端存储实例需要的用户名字

#### password属性

后端存储实例需要的密码

### 以上参考自……

---------------------

作者：冰 河

来源：CSDN

原文：https://blog.csdn.net/l1028386804/article/details/53385637

版权声明：本文为博主原创文章，转载请附上博文链接！

# MyCAT部署

## (1)删除历史环境：

pkill mysqld

rm -rf /data/330\*

mv /etc/my.cnf /etc/my.cnf.bak

## (2)创建相关目录

mkdir /data/33{07..10}/data -p

## (3)初始化数据

mysqld --initialize-insecure --user=mysql --datadir=/data/3307/data --basedir=/usr/local/mysql

mysqld --initialize-insecure --user=mysql --datadir=/data/3308/data --basedir=/usr/local/mysql

mysqld --initialize-insecure --user=mysql --datadir=/data/3309/data --basedir=/usr/local/mysql

mysqld --initialize-insecure --user=mysql --datadir=/data/3310/data --basedir=/usr/local/mysql

## (4)准备配置文件(注意server\_id：51：7,8,9,10，52：17,18,19,20)

vim /data/3307/my.cnf

[mysqld]

basedir=/usr/local/mysql

datadir=/data/3307/data

socket=/data/3307/mysql.sock

port=3307

log-error=/data/3307/mysql.log

log\_bin=/data/3307/mysql-bin

binlog\_format=row

skip-name-resolve

server-id=7

gtid-mode=on

enforce-gtid-consistency=true

log-slave-updates=1

-----------------------------

vim /data/3308/my.cnf

[mysqld]

basedir=/usr/local/mysql

datadir=/data/3308/data

port=3308

socket=/data/3308/mysql.sock

log-error=/data/3308/mysql.log

log\_bin=/data/3308/mysql-bin

binlog\_format=row

skip-name-resolve

server-id=8

gtid-mode=on

enforce-gtid-consistency=true

log-slave-updates=1

------------------

vim /data/3309/my.cnf

[mysqld]

basedir=/usr/local/mysql

datadir=/data/3309/data

socket=/data/3309/mysql.sock

port=3309

log-error=/data/3309/mysql.log

log\_bin=/data/3309/mysql-bin

binlog\_format=row

skip-name-resolve

server-id=9

gtid-mode=on

enforce-gtid-consistency=true

log-slave-updates=1

vim /data/3310/my.cnf

[mysqld]

basedir=/usr/local/mysql

datadir=/data/3310/data

socket=/data/3310/mysql.sock

port=3310

log-error=/data/3310/mysql.log

log\_bin=/data/3310/mysql-bin

binlog\_format=row

skip-name-resolve

server-id=10

gtid-mode=on

enforce-gtid-consistency=true

log-slave-updates=1

## (5）准备启动脚本

vim /etc/systemd/system/mysqld3307.service

[Unit]

Description=MySQL Server

Documentation=man:mysqld(8)

Documentation=http://dev.mysql.com/doc/refman/en/using-systemd.html

After=network.target

After=syslog.target

[Install]

WantedBy=multi-user.target

[Service]

User=mysql

Group=mysql

ExecStart=/usr/local/mysql/bin/mysqld --defaults-file=/data/3307/my.cnf

LimitNOFILE = 5000

## (6）修改权限，启动多实例

chown -R mysql.mysql /data/\*

systemctl start mysqld3307

systemctl start mysqld3308

systemctl start mysqld3309

systemctl start mysqld3310

## (7)节点主从规划：

主从规划（箭头指向谁是主库）

10.0.0.51:3307 <-----> 10.0.0.52:3307

10.0.0.51:3309 ------> 10.0.0.51:3307

10.0.0.52:3309 ------> 10.0.0.52:3307

10.0.0.52:3308 <-----> 10.0.0.51:3308

10.0.0.52:3310 -----> 10.0.0.52:3308

10.0.0.51:3310 -----> 10.0.0.51:3308

# MyCAT多实例部署过程

2.1创建多实例

-参考创建多实例文档

## 3307-🡪3309---db02

2.2 主库创建复制用户，授以replication slave权限

mysql -S /data/3307/mysql.sock

grant replication slave on \*.\* to repl@'10.0.0.%' identified by '123';

2.3 全备主库数据,恢复到从库

mysqldump -S /data/3307/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full.sql

2.4 开启主从复制—deployment

#登录从库

mysql -S /data/3309/mysql.sock

#登录从库，将主库备份数据手动导入到从库中

set sql\_log\_bin=0;

source /tmp/full.sql

#获取记录日志的起始点

vim /tmp/full.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=194;

ymq [(none)]>help change master to

CHANGE MASTER TO

MASTER\_HOST='10.0.0.52',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3307,

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

2.5启动复制线程:

start slave;

2.6确认主从复制成功

查看主从状态:

show slave status \G

#出现以下信息：slave正常

……

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

【出现报错】

mysql> show slave status\G

Last\_IO\_Error: Got fatal error 1236 from master when reading data from binary log: 'Could not find first log file name in binary log index file'

【解决办法】

--主库

mysql> flush logs;

Query OK, 0 rows affected (0.00 sec)

mysql> show master status;

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

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB | Executed\_Gtid\_Set |

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

| mysql-bin.000003 | 194 | | | 2e7ea746-f2bd-11e8-95b3-000c29064a71:1 |

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

1 row in set (0.00 sec)

--从库

stop slave;

reset slave all;

mysql> CHANGE MASTER TO

-> MASTER\_HOST='10.0.0.52',

-> MASTER\_USER='repl',

-> MASTER\_PASSWORD='123',

-> MASTER\_PORT=3307,

-> MASTER\_LOG\_FILE='mysql-bin.000001',

-> MASTER\_LOG\_POS=444,

-> MASTER\_CONNECT\_RETRY=10;

Query OK, 0 rows affected, 2 warnings (0.01 sec)

start slave;

## 3308-🡪3310—db02

mysqldump -S /data/3308/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full8.sql

[root@db02 scripts]# vim /tmp/full8.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=444;

CHANGE MASTER TO

MASTER\_HOST='10.0.0.52',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3308,

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000002', MASTER\_LOG\_POS=194;

【出现的错误】

show slave status\G

Last\_IO\_Error: error connecting to master 'repl@10.0.0.52:3308' - retry-time: 10 retries: 1

【原因】

my.cnf配置修改了没有重启服务

## 3307-🡪3309---db03

#修改配置文件

[root@db03 scripts]# cat /data/33{07..10}/my.cnf|grep 'server-id'

server-id=37

server-id=38

server-id=39

server-id=310

#重启服务

[root@db03 scripts]# systemctl restart mysqld3307.service mysqld3308.service mysqld3308.service mysqld3310.service

[root@db03 tmp]# vim full37.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000002', MASTER\_LOG\_POS=444;

CHANGE MASTER TO

MASTER\_HOST='10.0.0.53',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3307,

MASTER\_LOG\_FILE='mysql-bin.000002',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

## 3308-🡪3310---db03

#修改配置文件

[root@db03 scripts]# cat /data/33{07..10}/my.cnf|grep 'server-id'

server-id=37

server-id=38

server-id=39

server-id=310

#重启服务

[root@db03 scripts]# systemctl restart mysqld3307.service mysqld3308.service mysqld3308.service mysqld3310.service

2.3 全备主库数据,恢复到从库

mysqldump -S /data/3308/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full38.sql

#登录从库，将主库备份数据手动导入到从库中

stop slave;

set sql\_log\_bin=0;

source /tmp/full38.sql;

#找到master文件中的file和pos

[root@db03 tmp]# vim full37.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000004', MASTER\_LOG\_POS=154;

CHANGE MASTER TO

MASTER\_HOST='10.0.0.53',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3308,

MASTER\_LOG\_FILE='mysql-bin.000004',

MASTER\_LOG\_POS=154,

MASTER\_CONNECT\_RETRY=10;

## 3307-🡪3307---db03—》db02(master)

2.2 主库创建复制用户，授以replication slave权限

mysql -S /data/3307/mysql.sock

grant replication slave on \*.\* to repl@'10.0.0.%' identified by '123';

2.3 全备主库数据,恢复到从库

[root@db02 ~]# mysqldump -S /data/3307/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full77.sql

2.4 开启主从复制—deployment

#登录从库

mysql -S /data/3309/mysql.sock

#登录从库，将主库备份数据手动导入到从库中

stop slave;

set sql\_log\_bin=0;

source /tmp/full77.sql;

#获取记录日志的起始点

vim /tmp/full77.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=444;

ymq [(none)]>help change master to

CHANGE MASTER TO

MASTER\_HOST='10.0.0.52',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3307,

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

## 3307-🡪3307---db02—》db03(master)

2.2 主库创建复制用户，授以replication slave权限

mysql -S /data/3307/mysql.sock

grant replication slave on \*.\* to repl@'10.0.0.%' identified by '123';

2.3 全备主库数据,恢复到从库

[root@db02 ~]# mysqldump -S /data/3307/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full37.sql

2.4 开启主从复制—deployment

#登录从库

mysql -S /data/3307/mysql.sock

#登录从库，将主库备份数据手动导入到从库中

stop slave;

set sql\_log\_bin=0;

source /tmp/full287.sql;

#获取记录日志的起始点

vim /tmp/full37.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000003', MASTER\_LOG\_POS=194;

#配置主从TCP/IP连接参数

ymq [(none)]>help change master to

CHANGE MASTER TO

MASTER\_HOST='10.0.0.53',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3307,

MASTER\_LOG\_FILE='mysql-bin.000003',

MASTER\_LOG\_POS=194,

MASTER\_CONNECT\_RETRY=10;

## 3308-🡪3308---db03—》db02(master)

2.2 主库创建复制用户，授以replication slave权限

mysql -S /data/3308/mysql.sock

grant replication slave on \*.\* to repl@'10.0.0.%' identified by '123';

2.3 全备主库数据,恢复到从库

[root@db02 ~]# mysqldump -S /data/3308/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full287.sql

2.4 开启主从复制—deployment

#登录从库

mysql -S /data/3307/mysql.sock

#登录从库，将主库备份数据手动导入到从库中

stop slave;

set sql\_log\_bin=0;

source /tmp/full287.sql;

#获取记录日志的起始点

vim /tmp/full287.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=444;

#配置主从TCP/IP连接参数

ymq [(none)]>help change master to

CHANGE MASTER TO

MASTER\_HOST='10.0.0.52',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3308,

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

start slave;

show slave status\G

## 3308-🡪3308---db02—》db03(master)

2.2 主库创建复制用户，授以replication slave权限

mysql -S /data/3308/mysql.sock

grant replication slave on \*.\* to repl@'10.0.0.%' identified by '123';

2.3 全备主库数据,恢复到从库

[root@db02 ~]# mysqldump -S /data/3308/mysql.sock -A -R --triggers --master-data=2 --single-transaction >/tmp/full387.sql

2.4 开启主从复制—deployment

#登录从库

mysql -S /data/3308/mysql.sock

#登录从库，将主库备份数据手动导入到从库中

stop slave;

set sql\_log\_bin=0;

source /tmp/full387.sql;

#获取记录日志的起始点

vim /tmp/full387.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000004', MASTER\_LOG\_POS=444;

#配置主从TCP/IP连接参数

ymq [(none)]>help change master to

CHANGE MASTER TO

MASTER\_HOST='10.0.0.53',

MASTER\_USER='repl',

MASTER\_PASSWORD='123',

MASTER\_PORT=3308,

MASTER\_LOG\_FILE='mysql-bin.000004',

MASTER\_LOG\_POS=444,

MASTER\_CONNECT\_RETRY=10;

start slave;

show slave status\G

## 检测主从状态

mysql -S /data/3307/mysql.sock -e "show slave status\G"|grep Yes

mysql -S /data/3308/mysql.sock -e "show slave status\G"|grep Yes

mysql -S /data/3309/mysql.sock -e "show slave status\G"|grep Yes

mysql -S /data/3310/mysql.sock -e "show slave status\G"|grep Yes

# mycat 下载安装

## 2.linux版本mycat下载地址

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

## 3、MyCAT安装

### 3.1 预先安装Java运行环境

yum install -y java

### 3.2下载mycat

下载Mycat-server-xxxxx.linux.tar.gz

http://dl.mycat.io/

### 3.3 解压文件

tar xf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz

### 3.4 软件目录结构

ls

bin catlet conf lib logs version.txt

#### 3.5 启动和连接

配置环境变量

vim /etc/profile

export PATH=/usr/local/mysql/bin:/application/mycat/bin:$PATH

source /etc/profile

启动

mycat start

连接mycat：

mysql -uroot -p123456 -h 127.0.0.1 -P8066

## 4、配置文件介绍

wrapper.log ---->mycat启动日志

mycat.log ---->mycat详细工作日志

schema.xml ----->主配置文件（读写分离、高可用、分表、节点控制）

server.xml ----->mycat软件本身相关的配置

rule.xml ----->分片规则配置文件（分片规则列表、使用方法）

## 5.应用前环境准备

### 5.1用户创建及数据库导入:

db01:

mysql -S /data/3307/mysql.sock

grant all on \*.\* to root@'10.0.0.%' identified by '123';

source /root/world.sql

mysql -S /data/3308/mysql.sock

grant all on \*.\* to root@'10.0.0.%' identified by '123';

source /root/world.sql

### 5.2 配置文件处理

#### #原本的主配置文件有过多不必要信息

cd /application/mycat/conf

mv schema.xml schema.xml.bak

#### #配置

vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

</schema>

<dataNode name="sh1" dataHost="ymq1" database= "world" />

<dataHost name="ymq1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

#### #重启

[root@db02 conf]# mycat restart

Stopping Mycat-server...

Stopped Mycat-server.

Starting Mycat-server...

#### #测试

#登录

mysql -uroot -p -h 127.0.0.1 -P8066

#读写测试

mysql> show variables like 'server\_id';

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

| Variable\_name | Value |

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

| server\_id | 9 |

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

1 row in set (0.04 sec)

mysql>

mysql> begin;

Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'server\_id';

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

| Variable\_name | Value |

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

| server\_id | 7 |

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

1 row in set (0.02 sec)

mysql>

# 配置读写分离及高可用

## 读写分离配置

[root@db01 conf]# mv schema.xml schema.xml.rw

[root@db01 conf]# vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

真正的 writehost：负责写操作的writehost

standby writeHost ：和readhost一样，只提供读服务

## 读写分离测试

mysql -uroot -p -h 127.0.0.1 -P8066

show variables like 'server\_id';

show variables like 'server\_id';

show variables like 'server\_id';

begin;

show variables like 'server\_id';

# 垂直分表配置

#重命名

mv schema.xml schema.xml.ha

#vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

<table name="city" dataNode="sh1"/>

<table name="country" dataNode="sh2"/>

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataNode name="sh2" dataHost="oldguo2" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

## 处理后端数据:

db02:

mysql -S /data/3307/mysql.sock

use world;

create table city\_new like city;

insert into city\_new select \* from city;

drop table city;

drop table countrylanguage;

drop table country;

mysql -S /data/3308/mysql.sock

use world;

create table country\_new like country;

insert into country\_new select \* from country;

drop table city;

drop table countrylanguage;

drop table country;

# MyCAT核心特性——分片（水平拆分）

分片：对一个"bigtable"，比如说t3表

(1)行数非常多，800w

(2)访问非常频繁

分片的目的：

（1）将大数据量进行分布存储

（2）提供均衡的访问路由

分片策略：

范围

取模

枚举

哈希

时间

全局表

ER分片

## deployment

mv schema.xml schema.xml.1

#编辑schema.xml文件

#vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

<table name="t3" dataNode="sh1,sh2" rule="auto-sharding-long" />

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataNode name="sh2" dataHost="oldguo2" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

## rule.xml

[root@db02 conf]# mv autopartition-long.txt autopartition-long.txt.bak

[root@db02 conf]# vim autopartition-long.txt

0-10=0

11-20=1

## 创建测试表---db02

mysql -S /data/3307/mysql.sock

create database world charset utf8;

use world

create table t3 (id int not null primary key auto\_increment,name varchar(20) not null);

mysql -S /data/3308/mysql.sock

drop database world;

create database world charset utf8;

use world

create table t3 (id int not null primary key auto\_increment,name varchar(20) not null);

## test

测试：

重启mycat

mycat restart

#登录mycat进行测试

mysql -uroot -p123456 -h 127.0.0.1 -P 8066

use TESTDB

insert into t3(id,name) values(1,'a');

insert into t3(id,name) values(2,'b');

insert into t3(id,name) values(3,'c');

insert into t3(id,name) values(4,'d');

insert into t3(id,name) values(11,'aa');

insert into t3(id,name) values(12,'bb');

insert into t3(id,name) values(13,'cc');

insert into t3(id,name) values(14,'dd');

# 枚举分片

mv schema.xml schema.xml.3

#修改主配置文件

vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">

<table name="t5" dataNode="dn1,dn2" rule="sharding-by-intfile" />

</schema>

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

<dataNode name="dn2" dataHost="localhost2" database= "test" />

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

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="localhost2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.53:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.53:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

#修改规则文件

vim rule.xml

<tableRule name="sharding-by-intfile">

<rule> <columns>sheng</columns>

<algorithm>hash-int</algorithm>

</rule>

</tableRule>

<function name="hash-int" class="org.opencloudb.route.function.PartitionByFileMap">

<property name="mapFile">partition-hash-int.txt</property>

<property name="type">1</property>

<property name="defaultNode">0</property>

</function>

#配置\*hash-int文件

[root@db01 conf]# cat partition-hash-int.txt

上海=0

北京=1

defaultNode=0 #分片的列sheng除了北京和上海，还有其他省，默认放入defaultNode中

#创建测试库

mysql -S /data/3307/mysql.sock

create database test charset utf8;

use test

create table t5 (id int not null primary key auto\_increment,sheng varchar(20) not null);

mysql -S /data/3308/mysql.sock

create database test charset utf8;

use test

create table t5 (id int not null primary key auto\_increment,sheng varchar(20) not null);

#重启mycat

mycat restart

#登录mycat中间件，进行测试

mysql -uroot -p123456 -h10.0.0.52 -P8066

use TESTDB

insert into t5(id,sheng) values(1,'北京');

insert into t5(id,sheng) values(2,'上海');

insert into t5(id,sheng) values(3,'北京');

insert into t5(id,sheng) values(4,'上海');

insert into t5(id,sheng) values(5,'天津');

# 案例一：范围分片

mv schema.xml schema.xml.55

vim schema.xml

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">

<table name="city\_new" dataNode="sh1,sh2" rule="auto-sharding-long" />

<table name="country\_new" type="global" dataNode="sh1,sh2" />

</schema>

<dataNode name="sh1" dataHost="oldguo1" database= "world" />

<dataNode name="sh2" dataHost="oldguo2" database= "world" />

<dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />

</writeHost>

</dataHost>

<dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType

="1">

<heartbeat>select user()</heartbeat>

<writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">

<readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />

</writeHost>

<writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">

<readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />

</writeHost>

</dataHost>

</mycat:schema>

vim autopartition-long.txt

0-2000=0

2001-4079=1

准备数据:

mysqldump -S /data/3307/mysql.sock world city\_new>/tmp/city\_new.sql

mysqldump -S /data/3308/mysql.sock world country\_new >/tmp/coutry\_new.sql

mysql -S /data/3307/mysql.sock

use world

truncate table city\_new;

mysql -S /data/3308/mysql.sock

use world

truncate table country\_new;

#建表:

mysql -S /data/3308/mysql.sock

use world

CREATE TABLE `city\_new` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`Name` char(35) NOT NULL DEFAULT '',

`CountryCode` char(3) NOT NULL DEFAULT '',

`District` char(20) NOT NULL DEFAULT '',

`Population` int(11) NOT NULL DEFAULT '0',

PRIMARY KEY (`ID`),

KEY `CountryCode` (`CountryCode`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

mysql -S /data/3307/mysql.sock

use world

CREATE TABLE `country\_new` (

`Code` char(3) NOT NULL DEFAULT '',

`Name` char(52) NOT NULL DEFAULT '',

`Continent` enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South America') NOT NULL DEFAULT 'Asia',

`Region` char(26) NOT NULL DEFAULT '',

`SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',

`IndepYear` smallint(6) DEFAULT NULL,

`Population` int(11) NOT NULL DEFAULT '0',

`LifeExpectancy` float(3,1) DEFAULT NULL,

`GNP` float(10,2) DEFAULT NULL,

`GNPOld` float(10,2) DEFAULT NULL,

`LocalName` char(45) NOT NULL DEFAULT '',

`GovernmentForm` char(45) NOT NULL DEFAULT '',

`HeadOfState` char(60) DEFAULT NULL,

`Capital` int(11) DEFAULT NULL,

`Code2` char(2) NOT NULL DEFAULT '',

PRIMARY KEY (`Code`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

# 用户管理

[root@db02 conf]# vim server.xml

……

<user name="root" defaultAccount="true">

<property name="password">123456</property>

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

<!-- 表级 DML 权限设置 -->

<!--

<privileges check="false">

<schema name="PRODUCT" dml="0110" >

<table name="tb01" dml="0000"></table>

<table name="tb02" dml="1111"></table>

</schema>

</privileges>

-->

</user>

<user name="user">

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

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

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

</user>

……