Day05 DBA进阶

MySQL主从同步

主从同步介绍

-实现数据自动同步的**服务结构**

-主服务器 : 接受客户端访问连接

-从服务器 : 自动同步主服务器数据

原则 : 客户端连接的那一台服务器 , 那这台为数据库主服务器

主从同步原理

Master

-启用binlog日志 (必须起binlog日志)

Slave

-Slave\_IO : 复制master主机binlog日志文件里的SQL命令写到本机的relay-log文件里

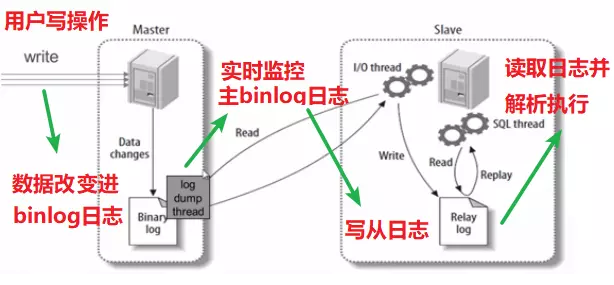
-Slave\_SQL : 执行本机relay-log文件里的SQL语句,实现与master数据一致

-Relay log : 中继日志文件 (~IO线程 ~SQL线程自动生成)

注 :

IO线程 : 负责从主服务器binlog拷贝SQL命令到本机(从) ---> 产生中继日志

SQL线程 : 在中继日志中提取SQL名命令并执行 , 同时产生中继日志



构建主从同步

构建思路

配置主服务器

-启用binlog日志,授权用户,查看binlog日志信息

配置从服务器

-设置server\_id

-确保与主服务器数据一致 必须先有主服务器 , 有库有表情况下才能配置从服务器

-指定主库信息 ---> binlog日志信息

-启动slave程序

-查看状态信息

注 :

授权用户 : 从服务器需要到主服务器拷贝SQL命令 , 需要一个授权用户

配置主服务器

修改配置文件 -启用binlog日志 , 并重启服务器

host51 ~]# vim /etc/my.cnf

server\_id=51 #指定服务器ID号

log\_bin=master51 #启动binlog日志

host51 ~]# systemctl restart mysqld

host51 ~]# ls /var/lib/mysql/master51.\*

/var/lib/mysql/master51.000001 /var/lib/mysql/master51.index #多出两个文件

host51 ~]# mysql -uroot -p123456

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123qqq...A"; #授权 复制从服务器 用户repluser

mysql> show master status; ---> 偏移量 441;(原154)

配置从服务器

51

host51 ~]# mysqldump -uroot -p123456 --master-data db5 > /root/db5.sql

host51 ~]# scp /root/db5.sql root@192.168.4.52:/root/

注 :

mysqldump : 完全备份

/root/db5.sql : db5库数据

--master-data : 记录当前备份数据对应的日志信息

52

修改主配置文件my.cnf

-指定server\_id 不允许与主库server\_id值相同

host52 ~]# vim /etc/my.cnf

server\_id=52 #指定服务器ID

host52 ~]# systemctl restart mysqld

host52 ~]# mysql -uroot -p123qqq...A

mysql> create database db5; #从库可以比主库多,但不可以少

host52 ~]# mysql -uroot -p123qqq...A db5 < /root/db5.sql #从服务器恢复数据

mysql> select count(\*) from db5.a;

mysql> select count(\*) from db5.b;

host52 ~]# grep master 51 /root/db5.sql ---> MASTER\_LOG\_POS=441;

MASTER\_LOG\_FILE='master51.000001'

**指定主服务器信息**

host52 ~]# grep master51 /root/db5.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000001', MASTER\_LOG\_POS=441;

51

mysql> change master to master\_host="192.168.4.51", #主库ip地址

master\_user="repluser", #主库授权用户

master\_password="123qqq...A", #授权用户密码

MASTER\_LOG\_FILE='master51.000001', #主库日志文件

MASTER\_LOG\_POS=441; #日志偏移量

mysql> start slave; #起服务

mysql> show slave status \G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

注 :

1. master信息会自动报错到/var/lib/mysql/master.info文件
2. 若更改主库信息时 , 应先执行stop slave ;修改后 再执行 start slave;

3.查看中继日志文件

host52 ~]# cat /var/lib/mysql/relay-log.info #中继日志文件

7

./host52-relay-bin.000008

365

master51.000003

154

0

0

1

查错:

1. 若出现relay log 结构错误 , 则需先查当前从服务器中继文件内容 , 查看偏移量

, 若出错 , 则需把4种类型文件都删除 , 重新配置

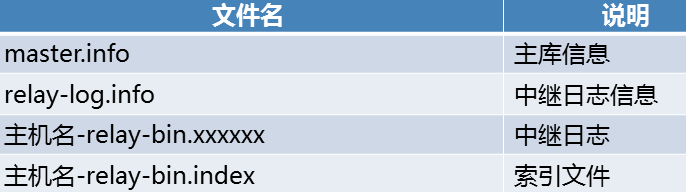
2.mysql> stop slave; #先停服务

Last\_IO\_Error\_Timestamp: #查看这两个报错信息

Last\_SQL\_Error\_Timestamp:

3.I/O线程一直显示connect(连接中) , 授权用户密码错误/从服务器未设置密码等

相关文件 -存放在数据库目录下 /var/lib/mysql



注 : 删除以上文件与新建数据库 , 再重起服务即可**还原**

验证测试 :

51主

mysql> create database db6;

mysql> create table db6.t1 (name char(10));

mysql> grant select,insert on db6.\* to yaya@"%" identified by "123qqq...B";

50 客

host50 ~]# mysql -h192.168.4.51 -uyaya -p123qqq...B

mysql> insert into db6.t1 values ("ben");

主从结构同步

结构类型

基本应用

-单向复制 : 一主 <-- 一从

从

^

扩展应用 : |

-1. 配置一主多从结构 : 从 <-- 主 --> 从

51 主

host51 ~]# mysqldump -uroot -p123456 --master-data **-B** db5 db6 > /root/twodb.sql

host51 ~]# grep master51 /root/twodb.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000001', MASTER\_LOG\_POS=3513;

host51 ~]# scp **db5.sql** root@192.168.4.53:/root

53

host53 ~]# vim /etc/my.cnf

server\_id=53 #指定服务器ID

host53 ~]# systemctl restart mysqld

host53 ~]# mysql -uroot -p123qqq...A db5 < /root/twodb.sql

host53 ~]# grep master51 /root/twodb.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000001', MASTER\_LOG\_POS=3513;

从服务器(53) 指定 主服务器(51)

mysql> change master to master\_host="192.168.4.51",

master\_user="repluser",

master\_password="123qqq...A",

MASTER\_LOG\_FILE='master51.000001',

MASTER\_LOG\_POS=3513;

mysql> start slave;

mysql> show slave status \G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

注 :

[root@host53 ~]# cat /var/lib/mysql/auto.cnf

[auto]

server-uuid=e07ecc67-ee78-11e9-af81-525400cd87fb

1. 配置主-从-从结构 (53(爷) 54(父) 55(子))

54作为 53的从服务器 与 55的主服务器

主 <-- 主 <-- 从 结构

环境准备 (若53已有实验 , 则删除/var/lib/mysql**4个文件**与库)

host53 ~]# cd /var/lib/mysql

host53 mysql]# rm -rf master.info

host53 mysql]# rm -rf relay-log.info

host53 mysql]# rm -rf host53-relay-bin.00000\*

host53 mysql]# rm -rf host53-relay-bin.index

host53 mysql]# systemctl restart mysqld

mysql> drop database db5;

mysql> drop database db6;

配置主服务器53

host53 ~]# vim /etc/my.cnf

server\_id=53 #指定服务器ID号

log\_bin=master53 #启动binlog日志

host53 ~]# systemctl restart mysqld

host53 ~]# ls /var/lib/mysql/master53.\*

/var/lib/mysql/master51.000001 /var/lib/mysql/master51.index

host53 ~]# mysql -uroot -p123qqq...A

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123qqq...A"; #授权 复制从服务器 用户repluser

mysql> show master status; ---> 偏移量 441;(原154)

**mysql> reset master;**

mysql> show master status; ---> 154

mysql> change master to master\_host="192.168.4.53", #指定服务器

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file='master53.000001',

master\_log\_pos=441;

mysql> start slave;

配置主从服务器54

host54 mysql]# vim /etc/my.cnf

log\_bin=master54

server\_id=54

**log\_slave\_updates #允许级联复制**

host54 mysql]# systemctl restart mysqld

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123qqq...A";

#设置主服务器 , 授予 从 服务服务器 repluser权限

replication slave : 用户权限

mysql> change master to master\_host="192.168.4.53",

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file='master53.000001',

master\_log\_pos=441;

mysql> start slave;

注 :

1.若不开启级联复制 , 在53上写的数据 , 54有55没有 ,

配置从服务器55

host54 mysql]# vim /etc/my.cnf

server\_id=55

host54 mysql]# systemctl restart mysqld

mysql> change master to master\_host="192.168.4.54",

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file="master54.000001", #主服务器文件

master\_log\_pos=441; #主服务器偏移量

mysql> start slave; #起服务

把数据库

10.16

复制模式

异步复制 主--->从模式

-主库执行完一次事务后(即客户端访问) , 立即将结果返给客户端 , 并不关心从库是否已经接受并处理

优点 : 客户得到响应速度快

缺点 : 所有从库不知是否存储数据完毕 , 当主库挂了 , 数据可能会丢失

全同步复制

-当主库执行完一次事务 , 且所有从库都执行了该事务后 , 才将结果返回给客户端

半同步复制

-介于异步复制和全同步复制之间

-主库在执行完一次事务后 , 等待至少有一个从库接收到并写到 relay log(中继日志)中 , 才将结果返回给客户端

修改复制模式

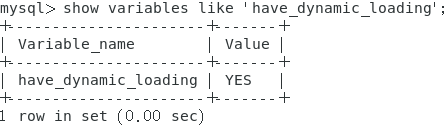
主服务器修改为半同步复制模式

从服务器修改为半同步复制模式

加载模块

1.查看是否允许动态加载模块 -默认允许

mysql> show variables like 'have\_dynamic\_loading'; #查看



1. 命令行启用模块(仅对本次启动有效) (好处:立刻生效,无需重启服务)

-使用数据库管理员 root 用户访问服务

rpl\_semi\_sync\_master : 模块(主库)类型

SONAME : 模块名称格式

semisync\_master.so : 模块名称

主(master)服务器上执行 (51)

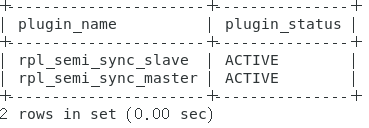
mysql> **install plugin** rpl\_semi\_sync\_master SONAME "semisync\_master.so";

从(slave)服务器上执行 (如果主服务器也做从服务器 ,也要加载slave的)

mysql> **install plugin** rpl\_semi\_sync\_slave SONAME "semisync\_slave.so";

mysql> select plugin\_name,plugin\_status from information\_schema.plugins where plugin\_name like "%semi%";

# 检查安装后 , 是否安装成功



注 :

若还要做从库 , 从服务器模块

rpl\_semi\_sync\_slvae : 模块(从库)类型

plugin\_name : 插件名称

plugin\_status : 插件状态

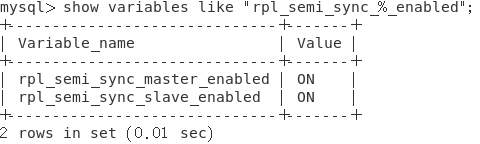
3.启用模块 set global (51)

-半同步复制 , 默认是关闭的 (0)(OFF)

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

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

mysql> show variables like "rpl\_semi\_sync\_%\_enabled"; #查看模块启用状态



4.永久启用 半同步复制

修改配置文件永久有效 (此处是 master和slave两种角色)

54(即做主又做从)

host54 ~]# vim /etc/my.cnf

[mysqld]

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

rpl\_semi\_sync\_slave=semisync\_slave.so" #主从服务器 (模块用分号”;”间隔)

rpl\_semi\_sync\_master\_enabled=1; #主服务器启用模块

rpl\_semi\_sync\_slave\_enabled=1; #从~~启用~~

注 :

so : 模块名

plugin-load #安装模块

rpl\_semi\_sync\_master #模块类型

semisync\_master.so #模块名

Day02

数据读写分离

原理 : 让 主数据库 处理事务性 增 删 改 查 , 而 从数据库处理select操作

如 :

请求 执行 读select 就访问 192.168.4.52 从服务器

192.168.4.50 ---------> 机器

(客户端) 执行 写insert 就访问 192.168.4.51 主数据库服务器

机器 : 接受客户端请求 , 即搭理服务器 (用192.168.4.57做)

读写分离适用环境 : 实质上是一个在资金比较缺乏 , 但又需要保证数据安全的需求下 , 在双机热备方案**上** , 做出的一种折中的扩展方案

使用读写分离方案注意事项:

1.数据库连接池要进行区分 , 那些是**读**连接池 , 那些是**写**~池 , 研发的难度会增加;

2.为了保证高可用 , **读**连接池要能实现故障自动转移;

3.主从的一致性问题需要考虑;

1.配置mysql主从同步

检查 51(主) 52(从) 主从结构

host52 ~]# mysql -uroot -p123qqq...A -e 'show slave status \G' | grep -i "192.168.4.51"

Master\_Host: 192.168.4.51

52

host52 ~]# mysql -uroot -p123qqq...A -e 'show slave status \G' | grep -i "yes"

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

57

host57 ~]# systemctl stop mysqld.service #停mysql服务

host57 ~]# yum -y install maxscale-2.1.2-1.rhel.7.x86\_64.rpm

#mysql代理服务器包

host57 ~]# ls /etc/maxscale.cnf #模板文件

host57 ~]# ls /var/log/maxscale/ #max服务日志文件存放目录 (默认空)

修改maxscale配置文件

1~7 : 描述信息 2 #帮助文档信息来源

host57 ~]# vim /etc/maxscale.cnf

9 [maxscale] #进程名

10 threads=auto #服务进程 线程数 默认1

18 [server1] #指定服务器 名称为 server1

20 address=192.168.4.51

24 [server2] #指定服务器 名称为 server2

26 address=192.168.4.52

35 [MySQL Monitor] #监视 MySQL 服务

38 servers=server1, server2 #监视server1与server2

39 user=**maxscalemon** #定义**监控用户** 可自定义

40 passwd=123qqq...A

注 : maxsaclemon 用来连接server1与server1的用户

52~58 #[Read-Only Service] +4008段 #全注释

63 [Read-Write Service]

#读写分离服务192.168.4.57访问自己使用的用户名, 密码是否存在server1与2

66 servers=server1, server2 (管理用户)

67 user=**maxscalerouter** #用maxscalerouter 验证 客户端密码是否

68 passwd=123qqq...A 有查询权限即可 存在server1 与 server2

97 [MaxAdmin Listener]

102 port=4016 #指定管理默认端口 4006 服务端口

host57 ~]# sed -i '/#/d' /etc/maxscale.cnf #清空注释行

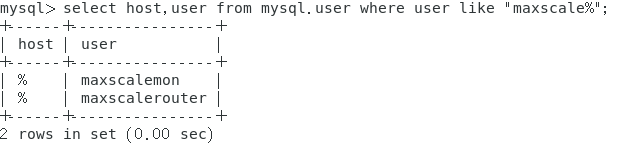
在主/从服务器创建授权用户 51(主库)

mysql> grant replication slave, replication client on \*.\* to maxscalemon@'%' identified by '123qqq...A'; #创建监控用户

mysql> grant select on mysql.\* to maxscalerouter@'%' identified by '123qqq...A'; #创建路由用户

查看创建 52(从库)

mysql> select host,user from mysql.user where user like "maxscale%";



启动服务

host57 ~]# maxscale -f /etc/maxscale.cnf #无显示则成功

注 :

报错查看日志 , host57 ~]# vim /var/log/maxscale/maxscale.log

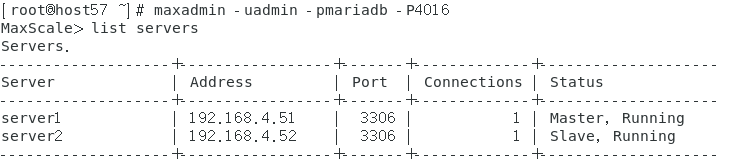
测试配置

host57 ~]# netstat -utnlp | grep maxscale --> 4006 服务端口 4016监控端口

host57 ~]# ps -C maxscale host57 ~]# kill -9 进程号 #**停止服务**

host57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale> list servers ---> #查看服务器列表 (有master与slave即为成功)



51 (主库)

mysql> create database gamedb;

mysql> create table gamedb.a(id int);

mysql> grant select,insert on gamedb.\* to yaya88@"%" identified by "123qqq...A";

mysql> show grants for yaya88@"%";

50 (客户)

host50 ~]# mysql -h192.168.4.57 **-P4006** -uyaya88 -p123qqq...A

报错 :

1.maxscale报错日志信息文件 : /var/log/maxscale/maxscale.log

2.测试配置时无法连接 Connection refused

查看配置文件 vim /etc/maxscale.cnf , 若出错 ,则修改

3.客户端(50) ERROR 1045 (28000): failed to create new session

一般是授权 管理用户 监视用户出错 ,

检查 代理服务器授权 与 主服务器授权 账户与密码是否一致

不一致 : 客户端测试会失败

多实例服务 (不支持mysql-5.7.17.tar)

什么是多实例?

-在一台物理主机上运行多个数据库服务

为什么要使用多实例?

-节约运维成本

-提高硬件利用率

安装软件 58

host58 ~]# yum -y install libaio #安装依赖包 (安装mysql包时不会安装此依赖包)

host58 ~]# useradd mysql

host58 ~]# tar -zxvf mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz(源码包)

host58 ~]# mv mysql-5.7.20-linux-glibc2.12-x86\_64 /usr/local/mysql

host58 ~]# echo $PATH

host58 ~]# PATH=/usr/local/mysql/bin/:$PATH

host58 ~]# vim /etc/bashrc

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

配置文件 50

client50 ~]# ls -l /var/lib/mysql/mysql.sock

**s**rwxrwxrwx 1 mysql mysql 0 10月 21 15:37 /var/lib/mysql/mysql.sock

s : 套接字文件---> 程序接口文件(服务启动时会自动创建)

进程名称 mysql sock文件用来传递数据信息 (本机访问自己的传递文件)

主配置文件 /etc/my.cnf

每个实例要有独立 : 数据库目录/端口/socket文件/pid文件/错误日志文件

host58 ~]# vim /etc/my.cnf (全手敲)

[mysqld\_multi] #启动多实例

mysqld = /usr/local/mysql/bin/mysqld\_safe #指定进程文件路径

mysqladmin = /usr/local/mysql/bin/mysqladmin #指定管理命令路径

user = root #指定进程用户

[mysqld1] #实例进程名称 , 1表示实例编号

datadir=/dir1 #数据库目录 , 要手动创建

port=3307 #端口号

pid-file=/dir1/mysqld1.pid #进程pid号文件位置 mysqld1.pid 可自定义

log-error=/dir1/mysqld1.err #错误日志位置

socket=/dir1/mysqld1.sock #指定sock文件的路径和名称

[mysqld2]

datadir=/dir2

port=3308

pid-file=/dir2/mysqld2.pid

log-error=/dir2/mysqld2.err

socket=/dir2/mysqld2.sock

启动服务

Mysqld1

host58 ~]# mysqld\_multi start 实例编号 #启动服务

host58 ~]# mysqld\_multi start 1 #最后一行显示管理员初始密码

客户端访问

host58 ~]# mysql -uroot -p'R>\*(Kak!%1Zo' -S /dir1/mysqld1.sock

mysql> alter user root@"localhost" identified by "123456"; #修改初始密码

host58 ~]# mysql -uroot -p123456 -S /dir1/mysqld1.sock

**停止服务**

]# mysqld\_multi --user=root --password=密码 stop 实例编码

Mysqld2

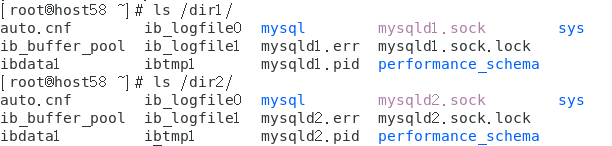
host58 ~]# mysqld\_multi start 2

客户端访问

host58 ~]# mysql -uroot -p'wyZEAg<ts4ge' -S /dir2/mysqld2.sock

mysql> alter user root@"localhost" identified by "123456";

host58 ~]# mysql -uroot -p123456 -S /dir2/mysqld2.sock



若无法起服务 :

配置文件出问题 : 1.修改配置文件 2.把新生成的/dir1或/dir2删除 3.重新启服务

host58 ~]# mysqld\_multi --user=root --password=123456 stop 1 #关服务

host58 ~]# vim /etc/my.cnf

host58 ~]# mysqld\_multi start 1 #起服务

host58 ~]# mysql -uroot -p123qqq...A -S /dir1/mysqld.sock

Mysql> grant all on \*.\* to yaya88@”%” identified by “123456”;

Host50 ~]# mysql -h192.168.4.58 -uyaya88 -P3307 -p123456

===============================================

10.17 Day03

分库/分表

1.什么是分库分表

-将存放在一台数据库服务器中的 数据 , 按照 特定方式 进行拆分 , 分散存放到多台数据库服务器中 , 以达到 分散单台服务器负载 的效果.

2.水平分割(横向切分) ----->用user\_id取余把数据分配到某个数据库服务器

-按照表中 指定字段 的分片规则 , 将表记录**按行**切分 , 分散存储到多个数据库中.

1. 垂直分割(垂直分割)

-将单个数据库 , 多个表按 业务类型 分类 , 分散存储到不同的数据库.

如银行 , 分为用户表/贷款表/基金表

MyCAT介绍

Mycat是基于java的分布式数据库系统**中间件** , 为高并发环境的分布式存储提供解决方案

-适合数据大量写入的存储需求

-支持MySQL / Oracle / Sqlserver / Mongpdb等做数据分片

-提供数据读写分离服务

-提供数据分片服务

-基于阿里巴巴Cobar进行研发的开源软件

高可用集群方式有:

**Keepalived+Mycat+Mysql**, **Keepalived+LVS+Mycat+Mysql**, **Keepalived+Haproxy+Mycat+Mysql**

中间件 : 架设在客户端与服务器之间的功能软件 , 实现数据分离 , 数据分片

分片规则

Mycat支持提供10种分片规则

1. 枚举法 sharding-by-intfile
2. 固定分片 rule 1
3. 范围约定 auto-sharding-long
4. 求模法 mod-long
5. 日期列分区法 sharding-by-date
6. 通配取模 sharding-by-pattern
7. ASCII码求模通配 sharding-by-prefixpattern
8. 编程指定 sharding-by-substring
9. 字符串拆分hash解析sharding-by-stringhash
10. 一致性hash sharding-by-murmur

工作过程

当mycat收到一个SQL命令时

1. 解析SQL命令涉及到的表
2. 然后看 对表的设置 , 如果有分片规则 , 则获取SQL命令里分片字段的值 , 并匹配分片函数 , 获得分片列表
3. 然后将SQL命令发往对应的分片服务器去执行
4. 最后收集和处理所有分片结果数据 , 并返回到客户端

环境部署

数据分片拓扑结构

三种角色 : 客户端 ---> 分片服务器 ---> 数据库服务器

部署MyCAT服务

1.安装JDK

host56 ~]# rpm -qa | grep -i jdk

host56 ~]# yum -y install java-1.8.0-openjdk

rpm -e --nodeps 包名 #卸载

2.安装mycat服务软件包

host56 ~]# tar -zxvf Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz

(免安装 ,解压即可使用)

host56 ~]# mv mycat/ /usr/local/

host56 ~]# ls /usr/local/mycat/

-bin #mycat命令 -catlet #扩展功能 -conf #配置文件

-lib #mycat使用的jar包

-logs #mycat启动日志和运行日志

-wrapper.log #mycat服务启动日志 (启动失败时产生的报错信息)

-mycat.log #记录SQL脚本执行后的报错内容

-version.txt #帮助文档

host56 mycat]# ls conf/\*.txt

分片规则 ,对应函数 ,服务软件

host56 mycat]# ls conf/\*.xml (修改3个文件)

-server.xml #设置连接账号及逻辑库

-schema.xml #配置数据分片

-rule.xml #分片规则

-其他文件 #函数调用文件

host56 mycat]# ls conf/\*.properties

host56 mycat]# ls lib/ #不允许修改此处文件(java文件) , 只可调用

3.创建连接用户

-定义客户端连接mycat服务的用户 (用户名/密码/逻辑库名)

host56 mycat]# vim /usr/local/mycat/conf/server.xml (此处使用默认 , 不修改)

80 <user name="root"> #连接mycat服务用户名

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

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

95 <user name="user">

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

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

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

99 </user>

Name为user 密码为user时,只有查询权限

/usr/local/mycat/conf/server.xml -定义客户端连接mycat服务的用户

xml文件注释符 : <!-- -->

4)配置数据分片

定义分片的表 schema.xml

host56 mycat]# vim /usr/local/mycat/conf/schema.xml

host56 mycat]# sed -i '56,77d' /usr/local/mycat/conf/schema.xml 剩56

host56 mycat]# sed -i '39,42d' /usr/local/mycat/conf/schema.xml 剩52

host56 mycat]# sed -i '16,18d' /usr/local/mycat/conf/schema.xml 剩49

host56 mycat]# sed -i '45,47d' /usr/local/mycat/conf/schema.xml 剩46

host56 mycat]# sed -i '43d' /usr/local/mycat/conf/schema.xml

host56 mycat]# wc -l conf/schema.xml

44 conf/schema.xml (文件内显示45行)

<mycat ...>

<schema ...> #定义分片信息

<table>......</table> #定义表 (在TESTDB库)

<table....../>

</schema>

<dataNode ....../> #指定 数据节点 名

<dataHost ....../> #数据库服务器主机名

<heartbeat>select user()</heartbeat>

<writeHost ......> #读写分离配置 name不可一样

</writeHost>

</dataHost>

</mycat>

注 :

-name #逻辑库名 或 逻辑表名 -rule #定义分片规则

-type=global #数据不分片存储

建议 : host56 mycat]# cp /usr/local/mycat/conf/schema.xml /root/

若改错 , 服务起不来 或 服务能起来,配置无作用

5)定义数据节点

<dataNode ....../> #指定数据节点 名

-name #数据节点名

-datahost #数据库服务器主机名 (可自定义 , 与真实主机无效 , 只起标识作用)

-database #数据库名

定义数据库服务器IP地址及端口

-<datahost 选项=值,..>......</datahost> #服务器主机名

-name #主机名(与datahost对应的主机名)

-host #主机名(与ip地址对应的主机名) (只起标识作用)

-url #数据库服务器IP地址及端口号

-user #数据库服务器授权用户

-password #授权用户密码

配置数据库服务器

在数据库服务器添加授权用户和创建存储数据库

53 54 55创建用户plj并给所有权限 , 密码123qqq...A (与schema.xml设置相同)

mysql> grant all on \*.\* to plj@"%" identified by "123qqq...A";

启动服务

host56 ~]# cd /usr/local/mycat/bin/

host56 bin]# **./mycat** --help

Usage: ./mycat { console | start | stop | restart | status | dump }

host56 bin]# ./mycat status

Mycat-server is not running. #服务未运行

host56 bin]# ./mycat start #起服务

host56 bin]# netstat -tnulp | grep :**8066** #

tcp6 0 0 :::8066 :::\* LISTEN 23412/java

客户端连接

客户端client50连接56主机 ,

client50 ~]# mysql -h192.168.4.56 -uroot -P8066 -p123456

mysql> show databases; ---> TESTDB 虚拟库 / 逻辑库

mysql> use TESTDB; ---> 在schema.xml中table定义出来的表 , 实际是虚拟表

===============================================

分片规则

1. **枚举法** -字段值必须在列举范围内选择 **sharding-by-intfile**

分片规则对应的 算法 , 要去 rule.xml 查

创表需要有分片规则要求的字段 , 参考schema.xml文件 , 否则创表失败

(此处使用枚举法)

host56 ~]# vim /usr/local/mycat/conf/**rule.xml**

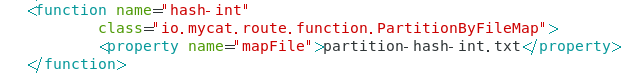
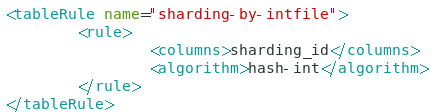
搜索 /sharding-by-intfile

Sharing\_id #数据分片字段名 调用算法的要求

Hash-int #使用的函数名

class #调用

partition-hash-int.txt #函数配置文件(地图文件) / 枚举值列表文件



host56 ~]# vim /usr/local/mycat/conf/partition-hash-int.txt

10000=0

10010=1

10020=2

注 :

0 #对应数据节点(第1台主机) dn1

1 #代表第2台主机 dn2

所以 , 添加第三台 dn3 ---> 10020=2

重启服务

host56 ~]# /usr/local/mycat/bin/mycat stop

host56 ~]# netstat -tnulp | grep :8066 -------> 无显示

host56 ~]# /usr/local/mycat/bin/mycat start

host56 ~]# netstat -tnulp | grep :8066 ---> 有8066端口

50(客户端)

mysql> use TESTDB;

mysql> create table employee (ID int primary key auto\_increment , sharding\_id int , name char(10) , sex enum (“m”,”w”)); #创建员工表

检测 : 查表53 54 55(数据库服务器)

53 use db**1** --> show tables --> desc employee --> 显示50创建的表

54 use db**2** --> show tables --> desc employee --> 显示50创建的表

55 use db**3** --> show tables --> desc employee --> 显示50创建的表

报错 :

2019-12-12 14-46-47 的屏幕截图

**Invalid DaraSource**:0 #无效的数据节点 , 0等于第一台服务器(即53) ,没建库/用户

50(客户机) 向所有字段赋值也要写字段名

mysql> insert into employee (sharding\_id , name ,sex) values (10030 , "bob" , "m"); #报错

mysql> insert into employee (sharding\_id , name ,sex) values (10020 , "johnA" , "m"); #只有dn3 主机能看到 , 即55主机

mysql> insert into employee (sharding\_id , name ,sex) values (10010 , "johnC" , "m"); #只有dn2 主机能看到 , 即54主机

===============================================

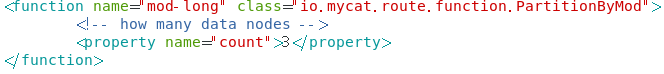
2.**求模法** (取余) (3台主机 , 求3的余数%3 ) 0为dn1/ 1为dn2/ 2为dn3

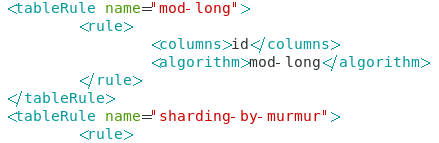
-根据字段值与设定的数字求模结果存储数据

host56 ~]# vim /usr/local/mycat/conf/rule.xml #使用默认,不需修改

末行搜索 /mod-long

<property name="count">3</property> #设定的数字3 , 即取3的余数



要有id字段

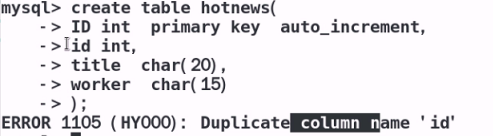
host56 ~]# vim /usr/local/mycat/conf/schema.xml

末行搜索 /hotnews

删除primaryKey="ID" autoIncrement="true"

求模法 带自增长不适合分片服务器 , 添加即报错

不删除则报错(id重复) :



重启服务

host56 ~]# /usr/local/mycat/bin/mycat stop

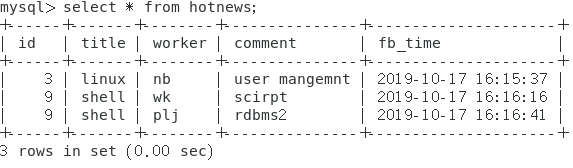
host56 ~]# netstat -tnulp | grep :8066 -------> 无显示

host56 ~]# /usr/local/mycat/bin/mycat start

host56 ~]# netstat -tnulp | grep :8066 ---> 有8066端口

50 (客户机) 建表

mysql> create table hotnews (id int , title char(20) , worker char(15) , comment varchar(15) , fb\_time timestamp);



检测 : 查表53 54 55(数据库服务器)

mysql> desc hotnews;

向所有字段赋值也要写字段名

mysql> insert into hotnews (id,title,worker,comment,fb\_time) values (3,"linux","nb","user mangemnt",now()); #%3余0 ,只有53看到

mysql> insert into hotnews (id,title,worker,comment,fb\_time) values (9,"shell","wk","scirpt",now()); # %3余0 ,只有53看到

mysql> insert into hotnews (id,title,worker,comment,fb\_time) values (11,"mysql","plj","rdbms2",now()); # %3余2 ,只有55看到

mysql> insert into hotnews (id,title,worker,comment,fb\_time) values (4,"ope","dmy","nginx",now()); # %3余1 ,只有54看到

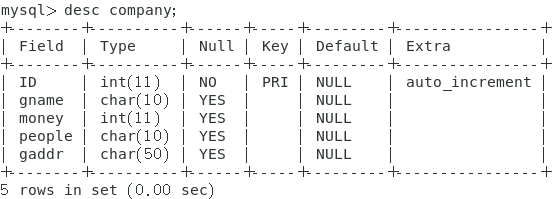
===============================================

3.**全局**(不分片) type=global

host56 conf]# vim schema.xml

搜索//type

mysql> create table company(ID int primary key auto\_increment,gname char(10),money int,people char(10),gaddr char(50));



mysql> insert into company (gname,money,people,gaddr) values ("tedu",10000000,"hsy","beijing");

mysql> insert into company (gname,money,people,gaddr) values ("taren",10000000,"hsy","beijing");

mysql> insert into company (gname,money,people,gaddr) values ("albaba",10000000,"hsy","beijing");

mysql> insert into company (gname,money,people,gaddr) values ("jd",10000000,"hsy","dzs");

===============================================

添加新库/表

1.服务器配置 id为求模法

host56 conf]# vim server.xml #添加新库BBSDB

82 <property name="schemas">TESTDB,BBSDB</property> #添加BBSDB库

host56 conf]# vim schema.xml #添加以下两行

<schema name="BBSDB" checkSQLschema="false" sqlMaxLimit="100">

<table name="hotnews2" dataNode="dn1,dn2,dn3" rule="mod-long" />

<table name="employee2" dataNode="dn1,dn2,dn3" rule="sharding-by-intfile" />

</schema>

重启服务

host56 ~]# /usr/local/mycat/bin/mycat stop

host56 ~]# netstat -tnulp | grep :8066 -------> 无显示

host56 ~]# /usr/local/mycat/bin/mycat start

host56 ~]# netstat -tnulp | grep :8066 ---> 有8066端口

50(客户端)

mysql> insert into hotnews2 (id,title) values (7,"a"),("8","b"),("23","c"),("109","d"),(6,”e”);

1 54 2 55 2 55 1 54 0 53 <--- select \* from hotnews;