**复制模式**

复制模式介绍:

• 异步复制

–主库执行玩一次事务后,立即将结果返给客户端,不关心从库是否已经接受并处理

• 全同步复制

–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客户端

• 半同步复制(用的最多)

–介于异步复制和全复制之间

–主库在执行完一次事务后,等待至少一个从库接受到并写到relay log中才将结果返回给客户端

**模式配置**

1. **查看是否允许动态加载模块**

–默认允许

> show variables like ‘have\_dynamic\_loading’;

1. **命令行加载插件**

–

主服务器上执行:

> install plugin **rpl\_semi\_sync\_master**

-> SONAME "**semisync\_master.so**";

从服务器上执行:

> install plugin **rpl\_semi\_sync\_slave**

-> SONAME "**semisync\_slave.so**";

即有主又有从:

-- 都执行 --

查看:

#状态记录在**基础表**information\_schema.plugins里

> select plugin\_name,plugin\_status from information\_schema.plugins

-> where plugin\_name like '%semi%';

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

| plugin\_name | plugin\_status |

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

| rpl\_semi\_sync\_master | ACTIVE |

| rpl\_semi\_sync\_slave | ACTIVE |

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

1. **启用半同步复制**

–半同步复制,默认是关闭的

命令行**临时**设置:

> set global rpl\_semi\_sync\_master\_enabled=1; #主服务器上执行

> set global rpl\_semi\_sync\_slave\_enabled=1; #从服务器上执行

> show variables like "rpl\_semi\_sync\_%\_enabled" #查看

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

| Variable\_name | Value |

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

| rpl\_semi\_sync\_master\_enabled | ON |

| rpl\_semi\_sync\_slave\_enabled | ON |

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

**永久**设置:

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

rpl\_semi\_sync\_master\_enabled=1

rpl\_semi\_sync\_slave\_enabled=1

重启mysql后查看验证

###############################################################################

**数据读写分离**

什么是数据读写分离:

*把客户端访问数据的读请求(select)和写请求(insert,updata,delete)分*

*别分配给不同的数据库服务器处理.*

如何实现:

1.程序实现(开发)

2.配置服务实现 (在服务器上安装软件提供服务)

中间件: mysql-proxy mycat maxscale

为什么要配置数据读写分离

#减轻主的压力 (从可以替主负责读请求,因为读不会改变数据)

---------(insert)-------> 主

客 -------> 代 --|

---------(select)-------> 从

#######################################################################

**配置数据读写分离**

1. **配置环境:一主一从**

主服务器: 192.168.4.51

从服务器: 192.168.4.52

1. **配置代理服务器**

代理服务器: 192.168.4.57

中间软件: maxscale

**Maxscale分离软件**

*Maxscale是C语言实现的智能代理,提供读写分离,负载均衡和高可用.*

工作方式: 线程

主配置文件: /etc/maxscale.cnf

日志目录: /var/log/maxscale/

进程:独享

线程:共享(多站多得,少站少得)

1. **安装软件包**

]$scp /linux-soft/03/mysql/maxscale-2.1.2-1.rhel.7.x86\_64.rpm [root@192.168.4.57:/root/](mailto:root@192.168.4.57:/root/)

无任何依赖

]#rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

**2.主配文件配置格式:**

9 [maxscale] #定义线程个数

10 threads=auto #默认为1个线程,改为atuo,根据cpu性能自动分配线程数

18 [server1] #指定数据库服务器

19 type=server

20 address=192.168.4.51 #master主机ip地址

21 port=3306

22 protocol=MySQLBackend

24 [server2] #指定数据库服务器

25 type=server

26 address=192.168.4.52 #slave主机ip地址

27 port=3306

28 protocol=MySQLBackend

35 [MySQL Monitor] #定义要监控的数据库节点

36 type=monitor

37 module=mysqlmon

38 servers=server1,server2 #监视的主,从的主机名(监控需要连接主,从mysql)

39 user=maxscalemon #监视时,连接主,从所需的用户名

40 passwd=123qqq...A #用户密码

41 monitor\_interval=10000 #监视间隔时间

**#注释掉**

52 **#**[Read-Only Service]

53 **#**type=service

54 **#**router=readconnroute

55 **#**servers=server1

56 **#**user=myuser

57 **#**passwd=mypwd

58 **#**router\_options=slave

63 [Read-Write Service] #定义读写分离的数据库节点

64 type=service

65 router=readwritesplit

66 servers=server1,server2 #读写在server1和..2之间实现

67 user=maxscalerouter #路由用户:代理查询客访问时使用的用户

在1,2的表有没有时,所用的访问1,2表的用户

68 passwd=123qqq...A #路由用户密码

69 max\_slave\_connections=100%

75 [MaxAdmin Service] #定义管理服务

76 type=service

77 router=cli

**#注释掉**

85 #[Read-Only Listener]

86 #type=listener

87 #service=Read-Only Service

88 #protocol=MySQLClient

89 #port=4008

91 [Read-Write Listener] #定义读写分离服务端口号(客户端访问的端口)

92 type=listener

93 service=Read-Write Service

94 protocol=MySQLClient

95 port=4006 #客户端连接mysqld端口

97 [MaxAdmin Listener] #定义管理服务端口号

98 type=listener

99 service=MaxAdmin Service

100 protocol=maxscaled

101 socket=default

102 port=4016 #进入监控界面的端口

**3.主从服务器授权主配置文件里的两个用户**

监控用户:

> grant **replication slave,replication client** on \*.\* to maxscalemon@'%'

->identified by '123qqq...A';

路由用户:

> grant select on mysql.\* to maxscalerouter@'%' identified by '123qqq...A'

**4.启动maxscale服务**

]# maxscale -f /etc/maxscale.cnf #主配置文件

]# ss -utnlp | grep maxscale

*#有两个端口: 4006 & 4016*

若起不来,则查看报错日志: /var/log/maxscale/

关闭maxscale的方法:

]#killall -9 maxscale

**5.验证**

默认:

用户名: admin

密码: mariadb

监控端口: 4016(配置文件)

[root@h57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale> #进入界面

MaxScale> list servers #查看后端服务器

Server | Address | Port | Connections | Status

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

server1 | 192.168.4.51 | 3306 | 0 | Master, Running

server2 | 192.168.4.52 | 3306 | 0 | Slave, Running

1. **客户端连接mysql**

51服务器上授权一用户供客户端连接: jom 123qqq...A

50作为客户端访问:

]# mysql -h192.168.4.57 -ujom -p123qqq...A -P4006 #**4006端口**

####################################################

总结:

]#rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

> grant **replication slave,replication client** on \*.\* to maxscalemon@'%'

->identified by '123qqq...A';

> grant select on mysql.\* to maxscalerouter@'%' identified by '123qqq...A';

启动maxscale

]# maxscale -f /etc/maxscale.cnf #主配置文件

]# maxadmin -uadmin -pmariadb -P4016

MaxScale> list servers

客户端连接mysql:

]# mysql -h192.168.4.57 -ujom -p123qqq...A -P4006 #**4006端口**

一共四个用户:

51需授权: 1.代理连接用户

2.

3.客户端访问用户jom

57代理默认: u: admin p: mariadb

##################################################################

Mysql多实例

* 什么是多实例: *一台物理主机上运行多个数据库服务*

为什么要多实例: *节约运维成本,提高硬件效率*

实例: *一台主机上多个mysql,每一个都叫一个实例*

1.环境准备(50主机)

-- 关闭mysql,

-- 关闭开机自启mysql,

-- 移走mysql的主配置文件(冲突问题,配置文件名都为my.cnf)

2.安装

]#scp /linux-soft/03/mysql/mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

[root@192.168.4.50:/root/](mailto:root@192.168.4.50:/root/)

]#tar -zxvf mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

当前目录多出: 与压缩包同名的目录

移动该目录,一般习惯程序移动到: /usr/local/ 下 并改名为: mysql

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

3.配置

自己创建my.cnf

-- 每个实例要有独立的: 数据库目录,端口,socket文件,pid文件,错误日志

socket:自己访问自己时,不需要通过网卡,通过此文件来传递数据

]#vim /etc/my.cnf

[mysqld\_multi]

mysqld=/usr/local/mysql/bin/mysqld\_safe

mysqladmin=/usr/local/mysql/bin/mysqladmin

user=root

[mysqld**1**] #1号

datadir=/dir1 #数据目录位置,需手动创建

port=3307 #指定端口号

log-error=/dir1/mysqld1.err #指定报错日志文件位置

pid-file=/dir1/mysqld1.pid #指定pid文件位置

socket=/dir1/mysqld1.sock #指定socket文件位置

#socket:自己访问自己时,不需要通过网卡,通过此文件来传递数据

[mysqld**2**] #2号

datadir=/dir2

port=3308

log-error=/dir2/mysqld2.err

pid-file=/dir2/mysqld2.pid

socket=/dir2/mysqld2.sock

1. 启动服务

启动1号,命令:

]# /usr/local/mysql/bin/mysqld\_multi start 1

#执行成功后数据库目录下会生成sock文件,pid文件,报错日志<-???

#执行成功后会显示root临时密码:

..............

2019-08-12T09:20:52.334639Z 1 [Note] A temporary password is

generated for root@localhost: **u3LL+(xT%T8%**

关闭服务

]#/mysqld\_multi --user=root --password=123456 stop 2

进入1号mysql

]#/usr/local/mysql/bin/mysql -uroot -p'u3LL+(xT%T8%'

**-S** /dir1/mysqld1.sock #通过指定sock文件号来区别连接哪个号的mysql

修改root密码

见数据库1(>alter user root@.........)

开启2号mysql

查看端口(两个)