主从同步原理: (本质就是复制主服务器上的sql语句到从服务器上运行 并不是直接复制数据!!)

Master 启用binlog日志

Slave:

Slave\_IO:复制master主机 binlog日志文件里的SQL命令到本机的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

构建思路

Master

1 启用binlog日志(若要指定其他目录下 必须再加一级 文件名!! log\_bin=/XX(目录)/xx(主机名))

2 授权用户

3 查看binlog日志信息

Slave

1 设置server\_id

2 确保与主服务器数据一致

3 指定主库信息

4 启动slave程序

5 查看状态信息

replication salve 复制数据的权限

--master-data 记录当前备份数据对应的日志信息 及日志名 与 偏移量 及当前服务器show master status 展示的信息

指定主服务器信息

change master to master\_host='XX',master\_user='XX',master\_password='XX',master\_log\_file='XX',master\_log\_pos=XX;

show slave status; start slave;

##Master 信息会自动保存到/var/lib/mysql/master.info文件中

##若更改主库信息时 应先执行stop slave;修改后再执行start slave

##若出现IO错误可以查看 /var/lib/mysql/master.info

##若要还原从服务器 删除相关文件后 重启服务

相关文件(/var/lib/mysql):

master.info 主库信息

relay-log.info 中继日志信息

主机名-relay-bin.XXX 中继日志 (只保留最新的2个)

主机名-relay-bin.index 索引文件

主从同步结构类型

基本: 单向复制: 一主 <--- 一从

扩展: 一主多从: 从<-- 主 -->从

链式复制:主<-- 从 <-- 从 一般不单独使用

互为主从: 主<-->主 一般不单独使用 互相写入时 可能存在冲突

一主多从注意事项!!

1 优先设置server\_id

2 !!! 2台从服务器的主机名不能一致 不然会导致mysql默认产生的relay\_log名被该server上的另一个mysql slave占用了

这样只会有一台生效!!! 非常重要!!

3 如果一定重名 则可以修改配置文件 来更改默认生成relay\_log名称

1.在my.cnf中添加

relay\_log=/db/mysql56/logs/relay\_98\_3326

报错现象

Relay log 导致复制启动失败一、报错现象dba:(none) start slave;ERROR 1872 (HY000): Slave failed to initialize relay log in

从报错上看，意思是启动slave时，使用repository中信息初始化relay log结构失败了。为什么失败了？原来是从tjtx135-1-95-relay-bin.index文件中找不到tjtx-96-67-relay-bin.017814文件。到这里，答案就很清楚了，由于我使用的是冷备份文件恢复的实例，在mysql库中的slave\_relay\_log\_info表中依然保留之前relay\_log的信息，所以导致启动slave报错。

问题解决

通过上面的报错以及relay log介绍，很容易知道由于mysql.slave\_relay\_log\_info表中保留了以前的复制信息，导致新从库启动时无法找到对应文件，那么我们清理掉该表中的记录不就可以了。再次提醒，不要手动删该表数据，MySQL已经提供工具给我们了：reset slave：

以后用冷备份恢复实例后，在启动slave前，先进行reset slave清空下以前的旧信息。

配置主从从中 需要在 那台又是又是主又是从的服务器的配置文件中添加

log\_slave\_update 允许级联复制

配置同步复制模式:

异步复制 (默认):主库执行完一次事务后,立即将结果返给客户端,并步关心从库是否已经接受并处理

直接由主库反馈给客户端 不考虑是否同步到从服务器上

全同步复制 :当主库执行玩一次事务,且 所有 (所有!)从库都执行了该事务后才将结果返还给客户端

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之后,等待 至少一个(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

命令行配置(临时设置,重启失效)

1.查看是否允许动态加载模块

show variables like 'have\_dynamic\_loading';

2.命令行加载插件(只有root有这个权限)

install plugin rpl\_semi\_sync\_master SONAME "semisync\_master.so"; //主服务器上执行

install plugin rpl\_semi\_sync\_slave SONAME "semisync\_slave.so"; //从服务器上执行

检查模块是否安装上

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

3启用半同步复制 (半同步复制,默认是关闭的)

set global rpl\_semi\_sync\_master\_enabled=1; //主服务器执行启用

set global rpl\_semi\_sync\_slave\_enabled=1; //从服务器执行启用

检查是否启用

show variables like "rpl\_semi\_sync\_%\_enabled";

配置文件配置(永久生效)

vim /etc/my.cnf

[mysqld]

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

rpl\_semi\_sync\_master\_enabled=1 //以上为主

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so

rpl\_semi\_sync\_slave\_enabled=1 //以上为从

在 主从从模式下 可以在同为主从的服务器下合并写用 ; 分割

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

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

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

mysql 读写分离

主从复制的应用局限性

访问数据库(包含读与写)流量过大 主数据库压力过大

读写分离 可以再客户端(程序实现 直接指定主 从服务器ip) 与 服务器端(mysql中间件 原理与代理相同)实现

grant replication slave,replication client on \*.\* to maxscalemon@"%" identified by '123qqq...A';

授予复制账号replication slave权限，复制才能真正地工作。

授予复制账号replication client权限,不可用于建立复制,但是复制用户可以使用 SHOW MASTER STATUS, SHOW SLAVE STATUS和 SHOW BINARY LOGS来确定复制状态.

赋予读写服务的用户 : 主要作用核查连接数据库的用户是否存在 所以只需要赋予mysql库的查询权限

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

启动服务器

maxscale -f /etc/maxscale.cnf

关闭服务器

kill -9 PID号

查看监控信息

maxadmin -uadmin -pmariadb -P4016

MaxScale> list servers //查看服务器

客户端访问

mysql -uXX -pxx -hxx(装有mysql中间件) -P4006(读写监听端口)

测试数据读写分离

在从服务器上往表里插入记录

通过客户端连接中间件服务器进行查询 也可以查询到信息

maxscale软件

主配置文件: /etc/maxscale.cnf

日志文件 : /var/log/maxscale/

mysql 多实例

临时添加环境变量

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

永久生效 vim /etc/bashrc

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

新建 /etc/my.cnf 主配置文件 (若此台机器装了mysql则会自动生成 那就无法实现多实例 因为mysqld\_multi 只读此文件)

配置文件中必须指定参数

port 端口

datadir 数据库目录

socket socket文件

pid-file pid文件

log-error 错误日志文件

socket 的原意是“插座”，在计算机通信领域，socket 被翻译为“套接字”，它是计算机之间进行通信的一种约定或一种方式。通过 socket 这种约定，一台计算机可以接收其他计算机的数据，也可以向其他计算机发送数据。

如果出现错误

有进程没端口的情况需要 杀死进程之后 把整个数据库目录删除 新建一个新的让mysqld\_mulit重新生成

主配置文件: /etc/maxscale.cnf (每行前面不能有空格!! 会启动不起来!!)

[maxscale]

threads=auto //线程个数

[server1] //服务名

type=server //类型

address=192.168.4.51 //master主机ip

port=3306

protocol=MySQLBackend //协议

[server2] //服务名(不能重名)

type=server

address=192.168.4.52 //slave主机ip

port=3306

protocol=MySQLBackend

[MySQL Monitor] //mysql 监控

type=monitor

module=mysqlmon

servers=server1,server2 //需要监控的服务

user=maxscalemon //登录mysql服务器的用户

passwd=123qqq...A //登录mysql服务器的密码

monitor\_interval=10000

##注释只读服务 (默认打开) 本次需要读写

#[Read-Only Service]

#type=service

#router=readconnroute

#servers=server1

#user=myuser

#passwd=mypwd

#router\_options=slave

[Read-Write Service]

type=service

router=readwritesplit

servers=server1,server2 //定义读写分离服务需要在什么服务器上执行

user=maxscalerouter //连接数据库的用户验证 即核查用户是否存在在mysql库中存在 (mysql -uXX -pXX -hXX

passwd=123qqq...A

max\_slave\_connections=100%

[MaxAdmin Service] //定义管理服务

type=service

router=cli

注释只读监听服务

#[Read-Only Listener]

#type=listener

#service=Read-Only Service

#protocol=MySQLClient

#port=4008

定义管理服务监听端口

[MaxAdmin Listener]

type=listener

service=MaxAdmin Service

protocol=maxscaled

socket=default

port=4016 //添加此行 端口号随意

数据库多实例安装配置

1 安装

yum -y install libaio

useradd mysql

tar xf XX(数据包)

mv mysql-5.7 (生成文件) /usr/local/mysql(方便管理)

PATH=/usr/local/mysql/bin/:$PATH (临时加入PATH变量)

2配置文件

vim /etc/my.cnf

##新建主配置文件 (若此台机器装了mysql则会自动生成 那就无法实现多实例 因为mysqld\_multi 只读此文件)

[mysqld\_mulit]

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

##指定进程文件路径 (/usr/local/mysql 生成文件修改后的名字与路径 ) 本质就是找 /bin 下找 mysqld\_safe

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

##指定管理命令的路径 本质就是找 /bin 下找 mysqldadmin

user=root //指定进程用户

[mysqldX] //实例进程名称 X表示实例编号

port=3307 [不能重复]

datadir=/dir1 //数据库目录,需要手动创建

socker=/dir1/XXX.sock //指定sock文件的路径和名称

pid-file=/dir1/XX.pid //指定pid号文件位置

log-error=/dir1/xx.err //指定错误日志位置

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

数据分片

将存放在一台数据库服务器中的数据,按照特点的方式进行拆分,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效果

水平分割:横向切分: 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.

垂直分割:纵向切分:将单个数据库的多个表按业务类型分类,分散存储到不同的数据库

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,Mongodb等

--提供数据读写分离

--提供数据分片服务

--基于阿里巴巴Cobar进行研发的开源软件

mycat支持提供的10种分片规则

1 枚举法

2 固定分片

3 范围约定

4 求模法

5 日期列分区法

6 通配取模

7 ASCII码求模通配

8 编程指定

9 字符串拆分hash解析

10 一致性hash

server.xml 建立连接用户

schema.xml 配置数据分片 (重要)!!!

删除所有备注 与 (readhost)以下这行 !!! 备注删不删都可以!

<readHost host="hostS1" url="localhost:3316" user="root" password="123456" />

由于并不考虑读写分离 所以需要删除此行

rule.xml 分片规则

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

<rule>

<columns>sharding</columns> //定义表中字段名称 若不一致无法插入数据

<algorithm>hash-int</algorithm> //匹配文档下function 对面的文件名

</rule>

</tableRule>

<function name="hash-int"

class="io.mycat.route.function.PartitionByFileMap">

<property name="mapFile">partition-hash-int.txt</property> //partition-hash-int.txt 定义新增数据的存入对应的数据库

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

<?xml version="1.0"?>

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

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

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100"> //定义逻辑库名 schema name="TESTDB"

<table name="travelrecord" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" /> //以下是逻辑库中逻辑表

<table name="company" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" /> //type="global" 数据不分片存储 存储到dataNode中所有的数据库节点

........ (等等)

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

// table name="employee" 此表的数据 按rule="sharding-by-intfile" 分片规则存储 对应查询 rule.xml

<table name="customer" primaryKey="ID" dataNode="dn1,dn2,dn3"

rule="sharding-by-intfile">

<childTable name="orders" primaryKey="ID" joinKey="customer\_id" // 以下为table name="customer" 的子表

parentKey="id">

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

parentKey="id" />

</childTable>

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

parentKey="id" />

</table>

</schema>

<dataNode name="dn1" dataHost="mysql53" database="db1" /> //定义上面节点数据库对应的真实数据库中的库

<dataNode name="dn2" dataHost="mysql54" database="db2" /> //database="db1" 需要在真实库中创建

<dataNode name="dn3" dataHost="mysql55" database="db3" />

<dataHost name="mysql53" maxCon="1000" minCon="10" balance="0" //分别对每个逻辑数据库进行具体配置

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM1" url="192.168.4.53:3306" user="admin" //在真实库中授权对应的用户与密码

password="123qqq...A">

</writeHost>

</dataHost>

<dataHost name="mysql54" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM2" url="192.168.4.54:3306" user="admin"

password="123qqq...A">

</writeHost>

</dataHost>

........(等等)

</writeHost>

</dataHost>

</mycat:schema>