一　mysql读写分离

１，基础

－－读写分离的原理

多台mysql服务器

分别提供读写服务，均衡流量

通过主从复制保持数据一致性

由mysql代理面向客户端

收到sql写请求时，交给服务器a处理

收到sql读请求时，交给服务器b处理

具体区分策略由服务设置

读写分离：数据的读，写分离

优点

1.增加冗余

2.增加了机器的处理能力

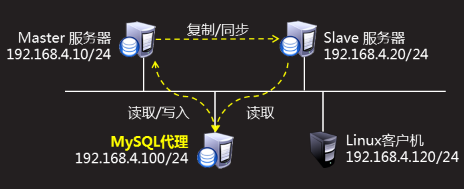
3.对于读操作为主的应用，使用读写分离是最好的场景，因为可以确保写的服务器压力更小，而读又可以接受点时间上的延迟。

缺点

　　单点故障

　　容易形成瓶颈

拓扑图：



２，实战

－－构建思路

部署mysql一主一从　同步结构

　　已经搭建好mysql主从复制

基于上一个实验的结果

其中slave为只读

添加一台mysql代理服务器（读写分离服务器）

部署／启用maxscale

client测试

１）mysql中间件

Mysqlcat

Maxscale

Mysql-proxy

－－配置

１）mysql一主一从同步结构

２）mysql代理服务器配置

使用软件－－＞maxscale

配置文件:/etc/maxscale.cnf

[root@myweb-57 jluo]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

[root@myweb-57 jluo]# vim /etc/maxscale.cnf

[maxscale] #进程名

threads=auto 　　　#线程数量，默认１

[server1]

type=server

address=192.168.4.51 #主库ip

port=3306

protocol=MySQLBackend

[server2]

type=server

address=192.168.4.52 #从库ip

port=3306

protocol=MySQLBackend

[MySQL Monitor] #监视mysql服务

type=monitor

module=mysqlmon

servers=server1, server2

user=scalemon #maxscale登录mysql服务用户名，用于监视mysql 是否running,

passwd=123456　　 #主从结构是否正常，谁是主库，谁是slave

monitor\_interval=10000 # 每隔10s检查一次

#detect\_stale\_master=true # 当所有的slave都不可用时，select查询请求会转发到master。

#[Read-Only Service]　　　　不需要指明只读

#type=service

#router=readconnroute

#servers=server1

#user=scalemon

#passwd=mypwd

#router\_options=slave

[Read-Write Service]　　　#定义读写分离服务

type=service

router=readwritesplit

servers=server1, server2

user=pliadmin　　　　#maxscale连接mysql服务检查客户端

passwd=123456　　　#使用的用户名和密码用的用户名,密码

max\_slave\_connections=100% # 所有的slave提供select查询服务

#max\_slave\_replication\_lag = 5 # slave超时5秒，就把请求转发到其他slave

[MaxAdmin Service]　　　#定义管理服务

type=service

router=cli　　　　　　　#命令行

#[Read-Only Listener]

#type=listener

#service=Read-Only Service

#protocol=MySQLClient

#port=4008

[Read-Write Listener]

type=listener

service=Read-Write Service　　　#与上面的[Read-Write Service]

protocol=MySQLClient

port=4006　　　　　　　　　　　#maxscale服务监听的端口号

[MaxAdmin Listener]

type=listener

service=MaxAdmin Service　　#与上面的[MaxAdmin Service]一样

protocol=maxscaled

socket=default

port=4026 #自定义端口号，不写使用默认端口

#user=maxscale #用户名默认admin

#passwd=Mhu87p2D　　#密码默认mariadb

51,52添加授权

51，52主机都要添加权限

mysql> grant select,insert on db1.\* to rose@'%' identified by '123456';　#授予客户端相应访问权限

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

-> identified by '123456'; #授权给maxscale

mysql> grant select on mysql.\* to pliadmin@'%' identified by '123456'; #授权给maxscal

提示：授权地址必须写’%’，否则会自动添加上ipv6地址，就会出错

replication slave：常用于建立复制时所需要用到的用户权限，也就是slave server必须被master server授权具有该权限的用户，才能通过该用户复制master server 的binlog日志sql语句。

并且"SHOW SLAVE HOSTS"这条命令和REPLICATION SLAVE权限有关，否则执行时会报错：

ERROR 1227 (42000): Access denied; you need (at least one of) the REPLICATION SLAVE privilege(s) for this operation

replication client ：不可用于建立复制，有该权限时，只是多了可以使用如"SHOW SLAVE STATUS"、"SHOW MASTER STATUS"等命令。

在5.6.6版本以后，也可以使用"SHOW BINARY LOGS"

52主机

mysql> select user,host from mysql.user;

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

| user | host |

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

| pliadmin | % |

| scalemon | % |

57主机

[root@myweb-57 jluo]# maxscale -f /etc/maxscale.cnf #开启服务

[root@myweb-57 jluo]# ss -tunlp | grep maxscale

tcp LISTEN 0 128 :::4026 :::\* users:(("maxscale",pid=4498,fd=12))

tcp LISTEN 0 128 :::4006 :::\* users:(("maxscale",pid=4498,fd=11))

[root@myweb-57 jluo]# killall -9 maxscale　　#停止服务

提示：注意57主机上不能运行mysql数据库服务

３）客户端验证

在maxscale本机连接管理端口

Maxadmin -uadmin -pmariadb -P端口

访问maxscale代理

Mysql -h服务器地址　-p端口　-u用户名　-p密码

在57本机访问管理服务查看监控信息

[root@myweb-57 jluo]# maxadmin -uadmin -pmariadb -P4026

MaxScale> list servers

Servers.

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

Server | Address | Port | Connections | Status

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

server1 | 192.168.4.51 | 3306 | 0 | Master, Running

server2 | 192.168.4.52 | 3306 | 0 | Slave, Running

在客户端主机连接57对数据进行访问，测试能否实现读写分离

测试一：

51主机

mysql> insert into db1.t1 values(11);

Client主机

[root@mysql-50 ~]# mysql -h192.168.4.57 -P4006 -urose -p123456

mysql> select \* from db1.t1;

+------+

| id |

+------+

| 11 |

测试二：

52主机

mysql> insert into db1.t1 values(52);　　#主机51没有这条数据

Client主机

mysql> select \* from db1.t1;

+------+

| id |

+------+

| 11 |

| 52 |

测试三：

Client主机：

mysql> insert into db1.t1 values(50);

mysql> select \* from db1.t1;

+------+

| id |

+------+

| 11 |

| 52 |

| 50 |

综上三个测试可知，实现了读写分离

二　mysql多实例

多实例：在一台物理机主机上运行多个数据库服务

优点：

节约运维成本

提高硬件利用率

Mysql5.7.17无多实例功能，使用5.7.20版本

１，配置步骤

安装支持多实例服务的软件包包

修改主配置文件

初始化授权库

启动服务

客户端测试

命令：

[root@mysql-50 mysql]# ./bin/mysqld --user=mysql --basedir=软件安装目录 --datadir=数据库目录 --initialize　 #初始化授权库，新建的目录或者目录为空时可以不用执行这条命令

[root@mysql-50 mysql]# ./bin/mysqld\_multi start 　实例编号　#启动实例进程

[root@mysql-50 mysql]# ./bin/mysqld\_multi --user=root --password=密码 stop 实例编号　　　#停止实例进程

[root@mysql-50 mysql]# ./bin/mysql -uroot -p'密码'　-S sock文件　#连接数据库

Mysql>alter user user() identified by ‘新密码’;　　#无密码权限

２，具体配置参数

[root@mysql-50 mysql]# vim /etc/my.cnf

[mysqld\_multi]

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

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

user=root

[mysqld1]　　　　#实例进程名称mysqld，实例编号1

port=3307

datadir=/data3307

socket=/data3307/mysql3307.socket

pid-file=/data3307/mysql3307.pid

log-error=/data3307/mysql3307.err

[mysqld2] #实例进程名称mysqld，实例编号2

port=3308

datadir=/data3308

socket=/data3308/mysql3308.socket

pid-file=/data3308/mysql3308.pid

log-error=/data3308/mysql3308.err

[root@mysql-50 mysql]# ./bin/mysqld\_multi start　1　启动实例进程

2018-11-24T09:36:30.136144Z 1 [Note] A temporary password is generated for root@localhost: 8lVbziaU20?I

[root@mysql-50 mysql]# ./bin/mysql -uroot -p'8lVbziaU20?I' -S /data3307/mysql3307.socket

mysql> alter user root@localhost identified by '123456';　　#更改密码

[root@mysql-50 mysql]# ./bin/mysqld\_multi start　2　启动实例进程

2018-11-24T09:43:17.988469Z 1 [Note] A temporary password is generated for root@localhost: (T2JsddBodgO

[root@mysql-50 mysql]# ./bin/mysql -uroot -p'(T2JsddBodgO' -S /data3308/mysql3308.socket

mysql> alter user user() identified by '123456';　　#更改密码

三　mysql性能优化

调优思路

升级硬件

加大网络带宽

服务程序的运行参数调整

对sql查询进行优化

[root@room9pc01 ~]# uptime

09:22:08 up 1:18, 4 users, load average: 1.88, 2.14, 2.22

[root@room9pc01 ~]# free -h

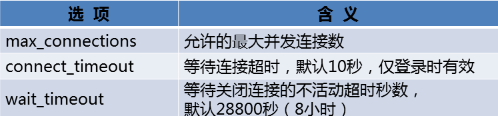
total used free shared buff/cache available

Mem: 15G 3.5G 9G 438M 1.9G 11G

Swap: 0B 0B 0B

[root@room9pc01 ~]# iostat　　　　#查看读写速度

－－Mysql参数优化



@max\_connections 　　　　#最大并发参数

mysql> show global status like "Max\_used\_connections"; #查看全局变量状况

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

| Variable\_name | Value |

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

| Max\_used\_connections | 1 |

mysql> show variables like 'max\_connections';

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

| Variable\_name | Value |

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

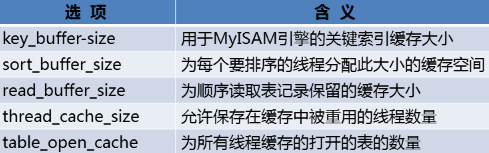
| max\_connections | 151 |

Max\_used\_connections/max\_connections <=0.85\*100<=85 理想值

＠connect\_timeout　　　　#等待客户端连接超时时间，单位s，仅登录时有效

@wait\_timeout #等待关闭连接的不活动超时秒数，默认28800秒（8小时）

缓存参数：



＠key\_buffer\_size #用于myisam引擎的关键索引缓存大小，单位：字节　byte

| key\_buffer\_size | 8388608　　　#单位：字节　byte　　默认8m

@sort\_buffer\_size #为每个要排序的线程分配此大小的缓存空间

sort\_buffer\_size | 262144 　　#单位：字节　　默认0.25m

@thread\_cache\_size　　#允许保存在缓存中被重用的线程数量

thread\_cache\_size | 9 #看内存大小分配

@read\_buffer\_size　　#为顺序读取表记录保留的缓存大小

read\_buffer\_size | 131072

@table\_open\_cache　　#为所有线程缓存打开的表的数量

table\_open\_cache | 2000

mysql> flush status;　　　　#刷新状态

mysql> show processlist;　　　#查看当前数据库的程序列表

－－查询缓存优化

查询缓存：存储曾经查找过的结果集

存储空间是从系统的物理内存划分出来的



| query\_cache\_limit | 1048576 |

| query\_cache\_min\_res\_unit | 4096 |

| query\_cache\_size | 1048576 |

| query\_cache\_type | OFF |

| query\_cache\_wlock\_invalidate | OFF

＠ query\_cache\_limit 　　　　　　查询缓存大小，单位：字节

@query\_cache\_type #

0:off

1:没有超出查询缓存限制，都可以

2:需要明确指定大小，且小于查询缓存，才可以

＠query\_cache\_wlock\_invalidate　#对myisam有效，对innodb无效

mysql> show global status like "qcache%";　　#查看缓存状态

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

| Variable\_name | Value |

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

| Qcache\_free\_blocks | 1 |

| Qcache\_free\_memory | 1031832 |

| Qcache\_hits | 0 |　　　　　　　#在查询缓存找到一次，自动加1

| Qcache\_inserts | 0 |　　　　　　 #收到一次查询请求，自动加1

| Qcache\_lowmem\_prunes | 0 |

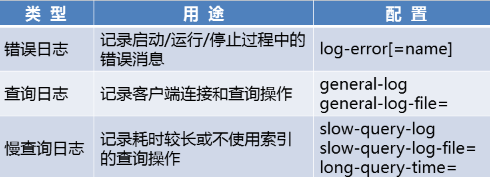
| Qcache\_not\_cached | 0 |

| Qcache\_queries\_in\_cache | 0 |

| Qcache\_total\_blocks | 1 |

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

－－mysql日志类型



错误日志

查询日志：记录客户端连接和查询，增，删，改所有sql操作

General-log #默认以主机名命名，存放数据库目录下

General-log-file=

慢查询日志：记录耗时较长或不使用索引的查询操作

Slow-query-log #启用慢查询

Slow-query-log-file=　　　　　　#指定慢查询日志文件

#Long-query-time=10 #sql查询超过10s，则记录，不写默认10s

Log-queries-not-using-indexes #记录未使用索引的查询

查看慢查询日志工具－mysqldumpslow

Mysqldumpslow 　慢查询文件