主从同步介绍

-实现数据自动同步的服务结构

-主服务器:接受客户端请求

-从服务器:自动同步主服务器数据

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

Binlog Dump :(Master has sent all binlog to slave) 将master 本地的bin-log 推送给slave中的IO

Slave:

Slave\_IO:接收 master主机推送的 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

show processlist; 查看主从状态 可以确定主服务器是否完全将binlog日志推送到从服务器

本质:Slave服务器IO线程发送请求给 Master服务器 binlog dump接受请求并将本地的binlog日志推送给Slave IO线程

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 (必须在还原数据之前)

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 若使用也是不同库的互为主主

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\_updates 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 (本质就是:master将请求发给slave就反馈客户端)

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 (本质就是:slave把relay-log中的sql语句写入磁盘后 反馈给客户端)

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 (本质就是:slave把请求写入到 relay-log 后反馈给客户端)

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

设置复制模式

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

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

一、mysql主从同步

1.1 主从同步介绍？

1.2 工作过程？

1.3 配置mysql主从同步

1.3.1 拓扑结构

1.3.2 配置主服务器

[root@host51 ~]# vim /etc/my.cnf

[mysqld]

server\_id=51

log\_bin=master51

:wq

[root@host51 ~]# systemctl restart mysqld

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

[root@host51 ~]# mysql -uroot -p123456

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

mysql> show master status;

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

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

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

| master51.000001 | 441 | | | |

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

1 row in set (0.00 sec)

1.3.3 配置从服务器

[root@52 ~]# vim /etc/my.cnf

[mysqld]

server\_id=52

:wq

[root@52 ~]# systemctl restart mysqld

[root@51 ~]# mysqldump -uroot -p123qqq...A --master-data db5 > /root/db5.sql

[root@51 ~]# scp /root/db5.sql root@192.168.4.52:/opt/

[root@52 ~]# mysql -uroot -p123qqq...A -e 'create database db5'

[root@52 ~]# mysql -uroot -p123qqq...A db5 < /opt/db5.sql

[root@52 ~]# mysql -uroot -p123qqq...A -e 'use db5 ; show tables'

[root@52 ~]# grep master51 /opt/db5.sql

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

[root@52 ~]#

[root@52 ~]# mysql -uroot -p123qqq...A

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=441;

mysql> start slave ;

mysql> show slave status\G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

时间15分钟到 10：43

相关文件

master.info 主库信息

relay-log.info 中继日志信息

主机名-relay-bin.xxxxxx 中继日志

主机名-relay-bin.index 索引文件

1.4 排错

Last\_IO\_Error: IO线程报错信息

Last\_SQL\_Error:SQL线程报错信息

stop slave ;

根据提示解决问题

start slave;

show slave status\G;

终极解决办法：

删除相关文件的4种文件后 重启数据库服务，重新指定主服务器信息。

自己排错时间

休息+排错30分钟 到 11：23

1.5 验证配置

1.5.1 添加授权用户（给客户端连接使用）

[root@host51 ~]# mysql -uroot -p123qqq...A

mysql> create database bbsdb ;

mysql> grant all on bbsdb.\* to yaya108@"%" identified by "123qqq...A";

mysql> grant all on db5.\* to yaya108@"%" identified by "123qqq...A";

1.5.2 客户端使用授权用户连接主服务器51 对数据做访问

[root@mysql50 ~]# mysql -h192.168.4.51 -uyaya108 -p123qqq...A

mysql> show grants;

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

mysql> insert into bbsdb.a values(100);

mysql> select \* from bbsdb.a;

+------+

| id |

+------+

| 100 |

+------+

mysql>

1.5.3 在从服务器本机能够查看到和主服务器一样的数据

[root@mysql52 ~]# mysql -uroot -p123qqq...A -e 'select \* from bbsdb.a'

时间到 11：50

二、主从同步模式

2.1主从同步结构模式

2.1.1 主从同步结构模式 ?

2.1.2 配置mysql一主多从结构

要求：把数据库服务器53 也配置为51的从服务器

配置从服务器 192.168.4.53 步骤如下： 15分钟时间 到14：35

[root@host53 ~]# vim /etc/my.cnf

[mysqld]

server\_id=53

:wq

[root@host53 ~]# systemctl restart mysqld

[root@host51 ~]# mysqldump -uroot -p123qqq...A --master-data -B db5 bbsdb > /root/bbsdb.sql

[root@host51 ~]# ls /root/\*.sql

/root/bbsdb.sql

[root@host51 ~]# scp /root/bbsdb.sql root@192.168.4.53:/root/

[root@host53 ~]# mysql -uroot -p123456 -e 'create database bbsdb'

[root@host53 ~]# mysql -uroot -p123456 bbsdb < /root/bbsdb.sql

[root@host53 ~]# mysql -uroot -p123456 -e 'use bbsdb ;show tables'

[root@host53 ~]# grep master51 /root/bbsdb.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000003', MASTER\_LOG\_POS=1315;

[root@host53 ~]# mysql -uroot -p123456

mysql> change master to master\_host="192.168.4.51" , master\_user="repluser" , master\_password="123qqq...A" , master\_log\_file="master51.000003" , master\_log\_pos=1315;

mysql> start slave ;

mysql> show slave status\G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

2.1.3 配置mysql主从从结构

2.1.4 配置mysql主主结构(2台数据库服务器互为主从)

要求：把数据库服务器192.168.4.56和192.168.4.57

2台服务器配置为互为主从结构。

2.2主从同步复制模式

2.2.1 主从同步复制模式

介绍？

2.2.2 配置半同步复制模式 （192.168.4.54）

命令行配置

1 查看是否允许动态加载模块

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

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

| Variable\_name | Value |

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

| have\_dynamic\_loading | YES |

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

1 row in set (0.00 sec)

mysql>

2 命令行加载插件

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

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

3 查看模块是否加载

mysql> 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 |

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

2 rows in set (0.00 sec)

4 启用半同步复制

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

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

5 查看是否启用半同步复制

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

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

| Variable\_name | Value |

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

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

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

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

永久配置 （/etc/my.cnf）

]# vim /etc/my.cnf

[mysqld]

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

:wq

[root@host54 ~]# systemctl restart mysqld

]# mysql -uroot -p123qqq...A

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

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

| Variable\_name | Value |

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

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

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

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

2 rows in set (0.00 sec)

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

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

读写分离应用场景:读操作远大于写操作时

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,用于数据复制与监控 replication slave,replication client

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

多实例概述

什么是多实例

在一台物理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

数据库多实例安装配置

1 安装

yum -y install libaio

useradd mysql //必须创建

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

socket=/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

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

一、数据读写分离

1.1 数据读写分离 ？

1.2 部署maxscale服务

1.2.1 安装软件

[root@57 ~]# netstat -utnlp | grep :3306

[root@57 ~]# systemctl stop mysqld ; systemctl disable mysqld

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

配置文件

命令

日志文件

服务程序

[root@57 ~]# ls /etc/maxscale.cnf

/etc/maxscale.cnf

[root@57 ~]#

[root@57 ~]# max

maxadmin maxbinlogcheck maxpasswd

maxavrocheck maxkeys maxscale

[root@57 ~]#

[root@57 ~]# ls /var/log/maxscale/

[root@57 ~]#

[root@57 ~]# rpm -qa | grep -i maxscale

maxscale-2.1.2-1.x86\_64

[root@57 ~]#

[root@57 ~]# rpm -ql maxscale

1.2.2 修改主配置文件

[root@57 ~]# cp /etc/maxscale.cnf /root/

[root@57 ~]# vim /etc/maxscale.cnf

1.2.3 配置数据库服务器（ server1 和 server2 ）

maxscalemon 用户授权

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

maxscalerouter用户授权

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

1.2.4 启动服务

]# maxscale -f /etc/maxscale.cnf

]# killall -9 maxscale //停止服务命令

1.2.5 查看服务信息

]# netstat -utnlp | grep :4016

]# netstat -utnlp | grep :4006

]# netstat -utnlp | grep maxscale

]# ls /var/log/maxscale/ 日志文件目录

]# vim /var/log/maxscale/maxscale.log 查看日志内容

休息到 11：20

1.2.6 查看监控信息

[root@host57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale>

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

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

MaxScale> exit

1.2.7 测试配置

1.2.7.1 在主服务器51 ， 添加客户端访问数据的连接用户

host51]# mysql -uroot -p123qqq...A

mysql> create database db7;

mysql> create table db7.a(id int);

mysql> grant select , insert on db7.\* to plj99@"%" identified by "123qqq...A";

host52]# mysql -uroot -p123qqq...A -e 'show grants for plj99@"%"'

1.2.7.2 在客户端连接服务器57 访问数据

client50]# mysql -h192.168.4.57 -P4006 -uplj99 -p123qqq...A

mysql> show grants;

mysql> select \* from db7.a;

mysql> insert into db7.a values(100);

mysql> select \* from db7.a;

时间10分钟 到 14：17

1.2.7.3 测试数据读写分离

在从服务器52本机向表里插入记录

]# mysql -uroot -p123qqq...A

mysql> insert into db7.a values(52);

mysql> select \* from db7.a;

在主服务器51本机查看表记录

mysql> select \* from db7.a;

在客户端50连接服务器57 访问数据

client50]# mysql -h192.168.4.57 -P4006 -uplj99 -p123qqq...A

休息到 15：05

二、mysql多实例

准备主机 克隆1台新的虚拟机配置Ip地址 eth0 192.168.4.58/24

软件 mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

2.1 mysql多实例 介绍

2.2 配置mysql多实例 ， 步骤如下：

1 安装软件

2 创建并编辑主配置文件

]# vim /etc/my.cnf

[mysqld\_multi]

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

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

user = root

[mysqld1]

datadir=/dir1

port=3307

socket=/dir1/mysqld1.sock

log-error=/dir1/mysqld1.err

pid-file=/dir1/mysqld1.pid

[mysqld2]

datadir=/dir2

port=3308

socket=/dir2/mysqld2.sock

log-error=/dir2/mysqld2.err

pid-file=/dir2/mysqld2.pid

[root@host58 ~]#

[root@host58 ~]# mkdir /dir1

[root@host58 ~]# mkdir /dir2

3 启动多实例服务

3.1 启动编号1的实例服务

]# mysqld\_multi start 1

]# ls /dir1

]# netstat -utnlp | grep :3307

]# mysql -uroot -p'zdkk%:q6HCe9' -S /dir1/mysqld1.sock

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

mysql> exit;

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

]# mysqld\_multi --user=root --password=123456 stop 1

]# netstat -utnlp | grep :3307

3.2 启动编号2 的实例服务

]# mysqld\_multi start 2

]# ls /dir2

]# netstat -utnlp | grep :3308

]# mysql -uroot -p'zdkk%:q6HCe9' -S /dir2/mysqld2.sock

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

mysql> exit;

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

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

]# netstat -utnlp | grep :3308

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

数据分片

分库分表: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服务运行后，默认监控的端口 8066

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

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

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

schema.xml 配置数据分片 (重要)!!!

<?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>

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

1 相关概念

2 MyCat软件介绍

3 配置数据分片

3.1 配置数据分片服务器192.168.4.56

3.1.1 安装软件

3.1.2 配置文件说明

3.1.3 修改配置文件

3.1.3.1 配置客户端连接使用的用户及密码

]# vim server.xml (使用默认配置)

TESTDB

3.1.3.2 配置数据分片

]# cp schema.xml /opt/

]# sed -i '56,77d' schema.xml

]# sed -i '39,42d' schema.xml

]# sed -i '16,18d' schema.xml

]# sed -i '45,47d' schema.xml

]# vim schema.xml

3.2 配置数据库服务器192.168.4.53/54/55

3.2.1 用户授权

3.2.2 创建存储数据库

[root@host53 ~]# mysql -uroot -p123qqq...A -e 'create database db1'

[root@host53 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@host54 ~]# mysql -uroot -p123qqq...A -e 'create database db2'

[root@host54 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@host55 ~]# mysql -uroot -p123qqq...A -e 'create database db3'

[root@host55 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

4 启动mycat服务(192.168.4.56)

[root@host56 conf]# /usr/local/mycat/bin/mycat -help

[root@host56 conf]# /usr/local/mycat/bin/mycat status

[root@host56 conf]# /usr/local/mycat/bin/mycat start

[root@host56 conf]# /usr/local/mycat/bin/mycat status

5 查看服务信息

查看进程

[root@host56 conf]# ps -C java

查看端口

[root@host56 conf]# netstat -utnlp | grep :8066

日志文件

[root@host56 conf]# ls /usr/local/mycat/logs/

6 客户端50 连接分片服务器56 访问数据

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

> show databases;

> use TESTDB;

> show tables;

> desc goods;

7 排错时间

自行排错时间

14:40 讲新内容

8 分片规则的使用

枚举法sharding-by-intfile: 字段值必须在列举范围内选择

]# vim schema.xml

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

rule="sharding-by-intfile" />

]# vim rule.xml

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

<rule>

<columns>sharding\_id</columns>

<algorithm>hash-int</algorithm>

</rule>

</tableRule>

<function name="hash-int"

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

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

</function>

:wq

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

10000=0

10010=1

10020=2

:wq

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

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

]# netstart -utnlp | grep ；8066

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

mysql> create table employee ( ID int primary key auto\_increment ,

sharding\_id int , name char(15) , age tinyint ,sex enum("man","woman"));

mysql> insert into employee(sharding\_id , name , age , sex)

values(10000,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10010,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"jim",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"lucy",19,"woman");

mysql> select \* from employee;

求模法mod\_long: 根据字段值与设定的数字求模结果存储数据

]# vim schema.xml

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

:wq

]# vim rule.xml

<tableRule name="mod-long">

<rule>

<columns>id</columns>

<algorithm>mod-long</algorithm>

</rule>

</tableRule>

<function name="mod-long" class="io.mycat.route.function.PartitionByMod">

<!-- how many data nodes -->

<property name="count">3</property>

</function>

:wq

[root@host56 ~]# /usr/local/mycat/bin/mycat stop

[root@host56 ~]# /usr/local/mycat/bin/mycat start

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

Database changed

mysql> create table hotnews(

id xint , title char(50),worker char(10), comment varchar(150));

insert into hotnews(id,title,worker,comment) values(7,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(8,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(9,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(6,"linux","redhat","linux-lb");

type=global 数据不分片存储

]# vim schema.xml

<table name="goods" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />

:wq

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

mysql> create table goods(ID int primary key ,name char(15),class char(9));

mysql> insert into goods(ID ,name , class)values(1,"tom","nsd1906");

mysql> insert into goods(ID ,name , class)values(2,"jim","nsd1906");

mysql> insert into goods(ID ,name , class)values(3,"max","nsd1906");

9 添加新库/新表

9.1 添加新库

]# vim server.xml

<user name="root">

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

<property name="schemas">TESTDB,BBSDB</property>

<user name="user">

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

<property name="schemas">TESTDB,BBSDB</property>

:wq

9.2 添加新表

]# vim schema.xml

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

<table name="company2" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />

<table name="employee2" primaryKey="ID" dataNode="dn1,dn2,dn3"

rule="sharding-by-intfile" />

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

</schema>

:wq

9.3 重启服务

[root@host56 ~]# /usr/local/mycat/bin/mycat stop

[root@host56 ~]# /usr/local/mycat/bin/mycat start

[root@host56 ~]# netstat -utnlp | grep :8066

9.4 测试配置

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> show databases;

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

| DATABASE |

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

| BBSDB |

| TESTDB |

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

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

MHA介绍

--由日本DeNA公司youshimaton开发

--是一套优秀的实现mysql高可用的解决方案

--数据库的自动故障切换操作能做到0-30秒内完成

--MHA能确保在故障切换过程中最大限度保证数据的一致性,以达到正真意义上的高可用

MHA工作过程:由Manager定时探测集群中的master节点;当master故障时,由Manager自动将拥有最新数据的Slave提升为新的master

##实现此操作需要将从复制模式改成半同步复制 这样至少有一台就拥有最新数据了!!!

MHA Manager(管理节点)

-管理所有数据库服务器

-可以单独部署在一台独立的机器上

-也可以部署在某台数据库服务器上

MHA Node(数据节点)

-存储数据的mysql服务器

-运行在每台mysql服务器上

修复故障服务器

1.启动mysql服务

2与当前主服务器数据一致

3指定主服务器信息 (修改mha主配置文件信息 添加服务器信息)

4启动slave进程

5查看状态信息 (查看mha服务状态信息)

配置MHA集群

1.配置ssh免密登录

--MHA Manager (管理节点)需要与 MHA Node(数据节点) 无秘钥配对 //主库需要与备用库无秘钥连接

--每一个 数据节点之间 也需要无秘钥配对 //当主库故障时 拥有最新数据的某一台备用库升级成主库 也需要与每个备用库无秘钥接连

2.配置MySQL一主多从结构

配置数据节点

rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm //所有数据库服务器都需要安装mha-node软件包

在所有数据服务器上添加监控用户

]# mysql –uroot –p密码

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

//可以只在host51主机执行授权命令，host52和host53 会自动同步授权

[root@host52]# mysql –uroot –p密码

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

//在2台从服务器上添加，数据同步连接用户 若在配置主从的时候 客户端已经设置为 % (任意) 则此步不需要

修改数据库服务运行参数 (主从都需要)

修改主服务器

# vim /etc/my.cnf

[mysqld]

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so" //加载模块

rpl\_semi\_sync\_master\_enabled=1 //启用master模块

rpl\_semi\_sync\_slave\_enabled=1 //启用slave模块

relay\_log\_purge=0 //禁止自动删除中继日志文件

##目的是为了 修改复制模式 使至少有一台从服务器有最新的数据 (半同步复制) 原因是 备用库升级到主库条件就是拥有最新的数据

3.配置管理节点

3.1安装

rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm// 每个都需要装 安装mha-node软件包

tar -zxvf mha4mysql-manager-0.56.tar.gz //解压mha-manager软件包

cd mha4mysql-manager-0.56 //进入源码目录

perl Makefile.PL //配置 并且根据反馈信息安装对应的包

make (编译)&& make install (安装)

3.2编辑主配置文件

mkdir /etc/mha //创建工作目录

cp mha4mysql-manager-0.56/sample/conf/app1.cnf /etc/mha/ //拷贝模板文件

vim /etc/mha/app1.cnf //编辑主配置文件

[server default] //管理服务默认配置

manager\_workdir=/etc/mha //工作目录

manager\_log=/etc/mha/manager.log //日志文件

master\_ip\_failover\_script=/etc/mha/master\_ip\_failover //故障切换脚本

ssh\_user=root //访问ssh服务用户

ssh\_port=22 //ssh服务端口

repl\_user=repluser //主服务器数据同步授权用户

repl\_password=123qqq…A //密码

user=root //监控用户

password=123qqq…A //密码

[server1] //指定第1台数据库服务器

hostname=192.168.4.51 //服务器ip地址

port=3306 //服务端口

candidate\_master=1 //竞选主服务器

....(等)

3.3创建故障切换脚本

cp mha-soft-student/master\_ip\_failover /etc/mha/ //拷贝模板

vim +35 /etc/mha/master\_ip\_failover //打开第35行

my $vip = '192.168.4.100/24'; # Virtual IP //定义VIP地址

3.4 在当前主服务器部署vip地址

ifconfig eth0:1 192.168.4.100 //部署vip地址 (临时)

修改从服务器

# vim /etc/my.cnf

[mysqld]

log-bin=master52 //必须开启!才能完成备用库到主库的转换

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so" //加载模块

rpl\_semi\_sync\_master\_enabled=1 //启用master模块

rpl\_semi\_sync\_slave\_enabled=1 //启用slave模块

relay\_log\_purge=0 //禁止自动删除中继日志文件

5测试配置

masterha\_check\_ssh --conf=/etc/mha/app1.cnf //执行测试命令

masterha\_check\_repl --conf=/etc/mha/app1.cnf //执行测试命令

!!!出现此条报错 说明主服务器的binlog日志存放不在默认路径 !!!

!!!!Failed to save binary log: Binlog not found from /var/lib/mysql,/var/log/mysql! If you got this error at MHA Manager, please set "master\_binlog\_dir=/path/to/binlog\_directory\_of\_the\_master" correctly in the MHA Manager's configuration file and try again.

启动管理服务

masterha\_manager --conf=/etc/mha/app1.cnf --remove\_dead\_master\_conf --ignore\_last\_failover

//执行启动命令 --必须添加--remove\_dead\_master\_conf (删除故障主库的信息 若不删除会导致服务下次无法启动)

修复故障服务器

1.将修复好的服务器作为新的备用服务器与当前主服务器做从属关系

2.与主服务器,其他备用服务器,管理节点服务器做无密码连接

3.将服务器信息添加到mha的配置文件中

4.测试集群环境 masterha\_check\_ssh masterha\_check\_repl

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

一、相关概念（知识回顾）

集群分类：

LB : LVS 、Nginx 、HAproxy

HA : Keepalived (主 备 vip)

HPC: 专业领域

二、MHA介绍

三、环境准备

3.1 安装依赖的perl软件

3.2 配置ssh密钥对认证登陆

3.3 配置一主多从同步结构

1 配置主服务器192.168.4.51

四、创建MHA集

4.1 配置管理主机

4.1.1 安装软件

]# yum -y install perl-DBD-mysql perl-DBI perl-ExtUtils-\* perl-CPAN\*

]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

]# tar -zxvf mha4mysql-manager-0.56.tar.gz

]# cd mha4mysql-manager-0.56/

]# perl Makefile.PL

]# make

]# make install

4.1.2 编写主配置文件

]# mkdir /etc/mha

[root@host57 mha4mysql-manager-0.56]# cp samples/conf/app1.cnf /etc/mha/

]#vim /etc/mha/app1.cnf

4.1.2 创建故障切换脚本

mha-soft-student]# cp master\_ip\_failover /etc/mha/

]# chmod +x /etc/mha/master\_ip\_failover

]# vim +35 /etc/mha/master\_ip\_failover

my $vip = '192.168.4.100/24'; # Virtual IP

my $key = "1";

my $ssh\_start\_vip = "/sbin/ifconfig eth0:$key $vip";

my $ssh\_stop\_vip = "/sbin/ifconfig eth0:$key down";

:wq

4.1.3 在主服务器51 配置VIP地址

]# ifconfig eth0:1 192.168.4.100

]# ifconfig eth0:1

4.2 配置数据节点 192.168.4.51/52/53

4.2.1 安装mha\_node软件 (3台数据库服务器都要安装)

4.2.2 用户授权(3台数据库服务器都要做用户授权)

2.2.3 启用半同步复制模式

五、测试配置 (192.168.4.57)

5.1 测试ssh配置

]#masterha\_check\_ssh --conf=/etc/mha/app1.cnf

5.2 测试主从同步配置

]#masterha\_check\_repl --conf=/etc/mha/app1.cnf

时间 到 16:30

5.3 启动管理服务

[root@host57 ~]# masterha\_manager --conf=/etc/mha/app1.cnf --remove\_dead\_master\_conf --ignore\_last\_failover

]# masterha\_check\_status --conf=/etc/mha/app1.cnf

六、测试集群功能

6.1 客户端50 连接vip地址访问数据库服务，存取数据

6.2 测试集群高可用功能

6.3 修复故障的数据库服务器

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

!!拥有自己的存储服务的软件 若装了mysql则无法正常安装运行!!

PXC

--是基于Galera的mysql高可用集群解决方案

PXC特点

--数据强一致性

--没有主从切换操作,无需使用虚拟ip

--支持InnoDB存储引擎 (必须是)

--多线程复制 (线程是共享资源的 进程是独占资源)

--部署使用简单

--支持节点自动加入,无需手动拷贝数据

--Galera Cluster 是Codership公司开发的一套免费开源的高可用方案

--PXC集群主要由两部分组成:Percona Server with XtraDB和Writ Set Replication patches(同步,多主复制插件)

PXC特点

--数据强一致性.无同步延迟

--没有主从切换操作,无需使用虚拟ip

--支持InnoDB存储引擎

--多线程复制

--部署使用简单

--支持节点自动加入,无需手动拷贝数据

相应端口

3306 数据库服务端口

4444 SST端口

4567 集群通信端口

4568 IST端口

SST State Snapshot Transfer 全量同步

IST Increment State Transfer 增量同步

!!拥有自己的存储服务的软件 若装了mysql则无法正常安装运行!!

1.装包 (顺序不变 percona-xtrabackup与libev有依赖 qpress与percona-xtrabackup libev 2者有依赖 )

]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm //安装依赖

]# yum -y install percona-xtrabackup-24-2.4.13-1.el7.x86\_64.rpm //percona-xtrabackup 在线热备程序

]# rpm -ivh qpress-1.1-14.11.x86\_64.rpm //安装依赖 qpress 递归压缩程序

]# tar -xvf Percona-XtraDB-Cluster-5.7.25-31.35-r463-el7-x86\_64-bundle.tar

]# yum -y install Percona-XtraDB-Cluster-\*.rpm //集群服务程序

2.配置

配置文件说明 ( /etc/percona-xtradb-cluster.conf.d)

mysqld.cnf /数据库服务运行参数配置文件

mysqld\_safe.cnf //Percona Server 5.7配置文件

wsrep.cnf //PXC集群配置文件

mysqld.cnf

server-id=1 //server-id 不允许重复

wsrep.cnf

wsrep\_cluster\_address=gcomm:// 集群成员列表 必须将本机的ip写在最后 这样可以同步前面的数据库数据

wsrep\_node\_address=192.168.4.71 //本机ip地址

wsrep\_cluster\_name=pxc-cluster //集群名称 3台必须相同

wsrep\_node\_name=pxcnode70 //本机主机名 即 /etc/hosts 映射的主机名

wsrep\_sst\_auth="sstuser:123qqq...A" //SST数据同步授权用户

mysqld\_safe.cnf

pid-file=/var/run/mysqld/mysqld.pid //pid文件位置及名称

socket=/var/lib/mysql/mysql.sock //socket文件位置及名称

3启服务

systemctl start mysql@bootstrap.service //启动集群 (无法tab需手打)

grep password /var/log/mysqld.log //查看数据库初始root登录密码

mysql -uroot -pXXX

mysql>alter user user() identified by "123456"

mysql> grant reload,lock tables,replication client,process on \*.\* to sstuser@"localhost" identified by "123qqq...A";

// user 是wsrep.cnf中定义的用户和密码 若不授权 后面的其他的服务器都无法启动

[root@pxcnode72 ~]# systemctl start mysql //启动其他节点数据库服务

[root@pxcnode72 ~]# netstat -utnlp | grep :3306

tcp6 0 0 :::3306 :::\* LISTEN 12794/mysqld

[root@pxcnode72 ~]# netstat -utnlp | grep :4567

tcp 0 0 0.0.0.0:4567 0.0.0.0:\* LISTEN 12794/mysqld

show status like "%wsrep%" //显示集群信息

建表时,必须要有主键字段 (硬性规定) (若增加了自增长选项 增长幅度为当前集群内服务器的总数 )

PXC 表存储引擎必须是innodb

存储引擎概述

--mysql服务软件自带的功能程序,处理表的处理器

--不同的存储引擎有不同的功能和数据的存储方式

--MySQL 5.0/5.1 --> MyISAM

--MySQL 5.5./5.6 --> InnoDB

做外键 存储引擎必须是innodb

使用innobackupex 存储引擎是innodb

事务(Transactions) 事务回滚 事务的特点

事务:一次访问从开始到结束的过程;事务是一个最小的不可再分的工作单元

数据库应用中完成单一逻辑功能的操作集合，是一个既具有原子性又具有一致性的功能，我们要求事务不违反任何数据库的一致性约束，也就是说，如果事务启动时数据是一致的，那么当这个事务成功结束的时候数据库也应该是一致的

mysql组成的8大组件

客户端(connectors) 与其他编程语言中的sql 语句进行交互，如php、java等。

连接池(connection pool) 管理缓冲用户连接，线程处理等需要缓存的需求

管理工具(Management Services) 系统管理和控制工具

SQL接口(SQL Interface) 接受用户的SQL命令，并且返回用户需要查询的结果。比如select from就是调用SQL Interface

分析器(Parser) SQL命令传递到解析器的时候会被解析器验证和解析。

优化器(Optimizer) SQL语句在查询之前会使用查询优化器对查询进行优化(产生多种执行计划,最终数据库会选择最优化的方案去执行,尽快返会结果) 他使用的是“选取-投影-联接”策略进行查询。

查询缓存(Caches & Buffers)(从真机的物理内存划分出来) 如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。

存储引擎(Pluggable Storage Engines) Mysql的存储引擎是插件式的。它根据MySql AB公司提供的文件访问层的一个抽象接口来定制一种文件访问机制（这种访问机制就叫存储引擎）。

文件系统(File system)

Myisam 自动生成表文件 a.frm a.MYD a.MYI

Innnodb 自动生成的表文件 a.frm a.ibd

MEMORY 自动生成的表文件 a.frm (存在内存中 所以只有frm表结构)

指定表存储的存储引擎

create table a.b(id int) engine=innodb;

修改表使用的存储引擎

alter table XXX engine=XX;

MyiSam

主要特点

---支持表级锁

---不支持事务,事务回滚,外键

表文件

---表名.frm //表结构 (desc)

---表名.MYI //索引

---表名.MYD //数据

Innodb

主要特点

---支持行级锁定

---支持事务,事务回滚,外键 (因为有事务日志文件 所以可以进行事务回滚)

表文件

---表名.frm 表结构

---表名.ibd (index(索引)+data(数据))

事务日志文件

---ibdata1

---ib\_logfile0

Mysql锁机制 (避免并发访问冲突问题 类似2个客户端同时对一条数据进行不同修改)

锁粒度

---表级锁:对整张表加锁

---行级锁:仅对被访问的行分别加锁

(被锁的对象无法让其他人访问 必须等锁被释放了之后才可以) 类似mysqldump导致锁表 innobackup只锁行

锁类型

---读锁(共享锁):支持并发读

---写锁(互斥锁,排它锁):是独占锁,上锁期间其他线程不能读表或者写表

查看当前锁的状态

show status like "table\_lock%"