Lvs的集群组成：

1. 前端负载均衡层（nginx）
2. 中间层：提供服务的机器
3. 底端：提供存储空间的存储区域

LVS的工作模式：

轮询 round robin

最小连接LC

加权最小连接WLC

基于局部的最小连接

带复制的基于局部的最小连接

源地址散列

视图：不同的人看到的内容不一样，不同的连接用户看到的数据不一样

存储过程：数据库中的脚本，完成一些重复性的工作

修改密码

[root@localhost ~]# mysqladmin -hlocalhost -uroot -p password " 新密码 "

update mysql.user set password=password("123456") where user="root" and host="localhost"; //这个是肯定对的用这个

mysql-community-server-5.7.17-1.el7.x86\_64.rpm

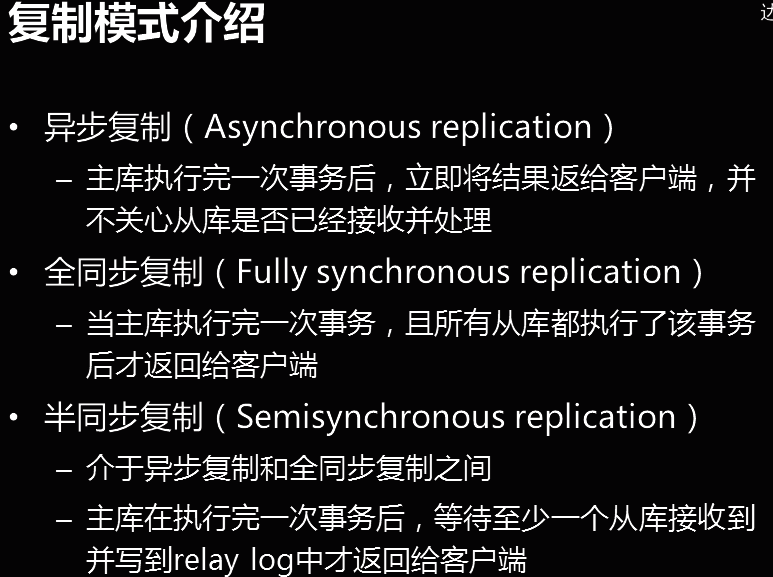
percona-xtrabackup-24.x86\_64 //主从,innobackup备份工具

mysql\_install\_db

mysql --verbose --help| grep innitialize

mysqld --user mysql --innitialize-insecure





半同步的时候不删中继日志，因为他可能是从库生的主库，这时中继日志才是所有的数据，

这句话应该部队，因为中继日志会删除吗？是为了从库发生故障时，保证有一台机器与主库数据一致。

异步复制可以提供最佳的性能，因为主服务器在将更新的数据写入它的二进制日志（Binlog）文件中后，无需等待验证更新数据是否已经复制到从服务器中。

Ifconfig eth0|grep -Po “(?<=inet 192.168.1.)\d+”

Sed ‘s/\(server-id\s\+=\).\*/\1 ${id}/’ my.xnf

Replication client slave

Innobackupex --slave-info --user=root --password=”123456” --host=localhost --no-timestamp /root/basckup //数据库的备份

Innobackupex --apply-log backup/

Innobackupex --copy-back backup/

数据库的备份

**物理备份：物理备份：直接备份数据库**

166 2018-12-11 09:50:55 root tar -zcPf mysql.gz /var/lib/mysql //服务端

128 2018-12-11 09:52:26 root tar -xPf mysql.gz -C /var/lib //客户端

**逻辑备份**

• 备份操作

– **mysqldump** -uroot -p123456 源库名 > 路径 /xxx.sql （-A，-B）

• 恢复操作

– **mysql** -uroot -p123456 目标库名 < 路径 /xxx.sql //恢复命令是**mysql**

**Binlog日志恢复**

**179 2018-12-11 10:25:37 root# mysqlbinlog --stop-position=1721 ecs-acad-bin.000001 | mysql -uroot -p123456**

**为什么之前不可以自定义binlog的位置？？？？因为你自定义的的那个目录你没有给他权限！！！！！！！属组和属主必须改为mysql才能够成功。**

**完全备份**

完全备份的缺点

**1.完全备份会锁表。**

**2.不安全**

**3.mysql的权限非常的重要，机的修改！！！！**

**增量备份**

**差异备份**

1. **首先先备份（上一部的操作）**

innobackupex --user root --password 123456 --incremental /fullback --no-timestamp //后面没有incremental-basedir 就是完全备份

cat /fullback/xtrabackup\_checkpoints

backup\_type = full-backuped(完全备)

加上数据（insert into a values(200);）

然后在后面加上incremental-basedir

innobackupex --user root --password 123456 --incremental /new1dir --incremental-basedir=/fullback --no-timestamp

在加数据（任意加）

innobackupex --user root --password 123456 --incremental /new2dir --incremental-basedir=/new1dir --no-timestamp

**2恢复增量数据**

systemctl stop mysqld

rm -rf /var/lib/mysql/

mkdir /var/lib/mysql

innobackupex --apply-log --redo-only /fullback

innobackupex --apply-log --redo-only /fullback --incremental-dir /new1dir innobackupex --apply-log --redo-only /fullback --incremental-dir /new2dir

**3.拷贝备份数据到数据库目录下**

innobackupex --copy-back /fullback

1. **修改/var/lib/mysql的所有者和所属组**

ls /var/lib/mysql //查看权限，查看所有者和所属组

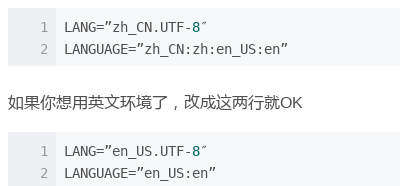
**chown -R mysql:mysql /var/lib/mysql //递归修改权限**

mysql> select \* from db5.a; //查看内容是否恢复

mysql> insert into a values(4000); //写入内容在做一次增量备

#innobackupex --user root --password 123456 --incremental /dir --incremental-basedir=/fullback --no-timestamp

这次的增量备是以/fullbakc为基础 因为恢复数据的时候都恢复到/fullback上了



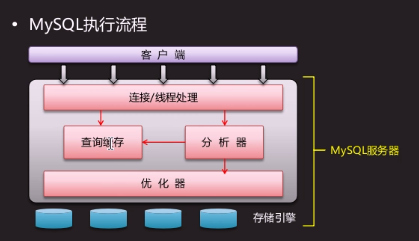
**RX packets** 41797 bytes 17965757 (17.1 MiB)

RX errors 0 dropped 11018 overruns 0 frame 0

**TX packets** 23926 bytes 33193520 (31.6 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0





查询缓存是在数据库的物理内存中分出来的，但是这个功能需要去找

数据库不重复的搜索：

1，可以设index用index查找

2，distinct select distinct(\*) from 表

3，group by 分组

**1.并发**

MariaDB [(none)]> show variables like "%connection%"

**| max\_connections | 151 并发连接数**

MariaDB [(none)]> show variables like "%connection%"

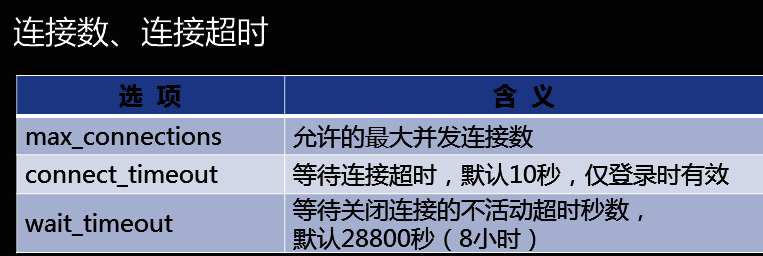
| Max\_used\_connections | 6

//当前有过的最大连接值

当前有过的最大连接值/并发连接数 ～= 0.85

MariaDB [(none)]> show processlist; 显示当前（谁）连接着的信息。

**2.连接控制**

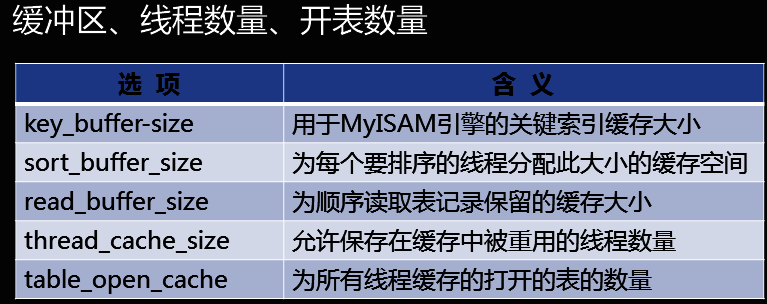


会影响连接的速度

connect\_timeout 10S比较好

wait\_timeout 等待连接后，数据库输入命令超时的时间。例：输入传输需要10分钟才能传完，传完才能执行下一步操作。

**3.缓存参数控制**



MariaDB [(none)]> show variables like '%buffer%'

key\_buffer\_size | 134217728（字节）/1024=k /1024=M

索引缓存的大小 = 128M

内存的速度要比硬盘中快所以这个**索引**大点好，多少还不知道。**可以加快查询速度**

Sort\_buffer\_size system echo $[2097152/1024/1024] 2M

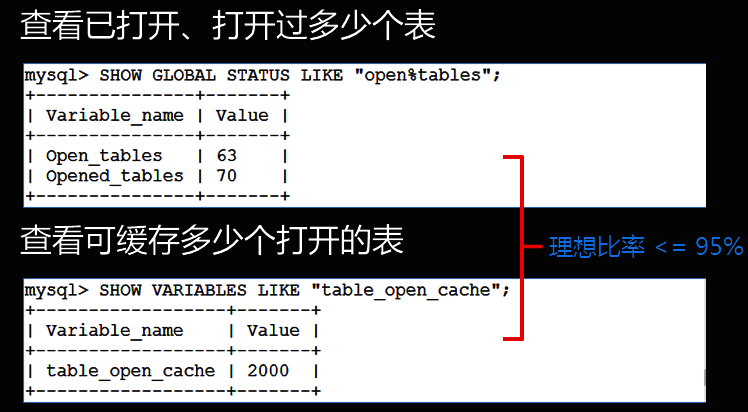
//如果**经常进行排队的查询**，将这个调大。

read\_buffer\_size | 131072  **顺序读取的缓存**

table\_open\_cache

**a表从硬盘中打开，调入到数据库服务内存中编辑，操作完了将内存中a写入硬盘，如果是10，就相当于开了10个表等着，不会关闭，只有超过10的数量，打开后才会关闭。**

thread\_cache\_size **可以重复用的线程，默认开启9个等待连接。按需调整(按并发量来调)**



**查看查询缓存的大小**

MariaDB [(none)]> show variables like 'query\_cache%';

| query\_cache\_type | ON 查询缓存开关，

一般都有缓存服务器，很少用数据库本机开

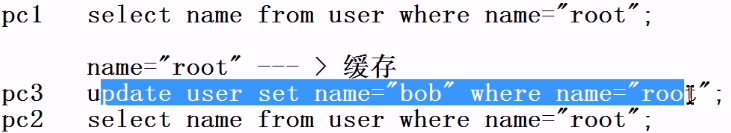
| query\_cache\_limit | 1048576 查询缓存大于1M不让存

| query\_cache\_min\_res\_unit | 4096(4k) 查询缓存单元

大了，浪费存储空间浪费，速度快

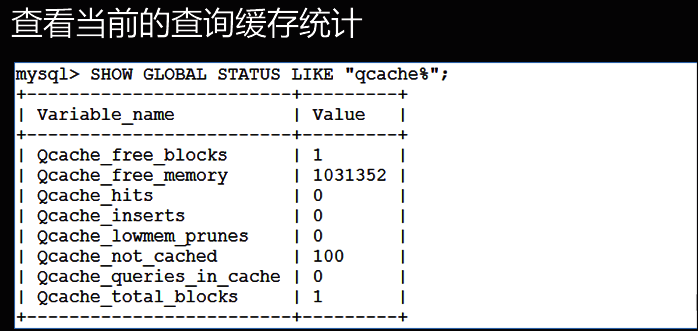
小了，不浪费存储空间，速度慢

| query\_cache\_wlock\_invalidate | OFF 查询缓存写锁，



本来pc2、pc3的时候不影响，应为茶的时候从查询缓存中查询的，

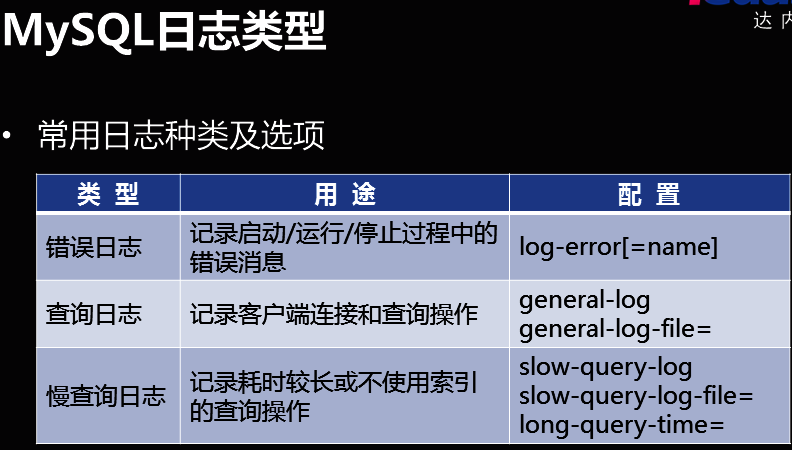
但是，例如：存钱 我需要看到我的变化，所以开启这个功能，改完了之后重新从表里查找数据，放入查询缓存中。 反正每次查询都会放入查询缓存中的记住了！！！！



| Qcache\_hits 9 **查询缓存的命中率**

| Qcache\_inserts 100 接受到查询的次数

9/100 低 命中率低，效率低 91次没找到 调查询缓存



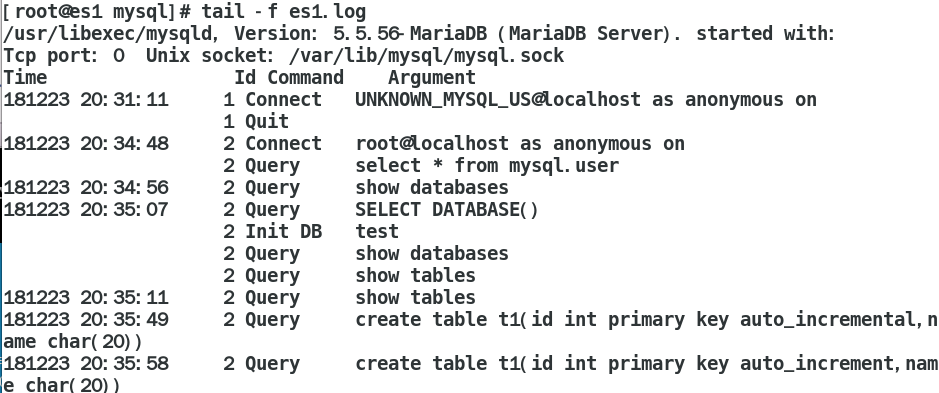


**优化程序员写的sql命令**

Binlog **记录查询之外的命令**

错误日志默认启动

查询日志 **默认不启动 在/etc/my.cnf 里的binlog位置添加，没有=默认在/var/lib/mysql下多了一个 主机名.log 执行的所有命令都记**



慢查询日志 默认不启动

slow-query-log 没有使用索引的命令记录下来

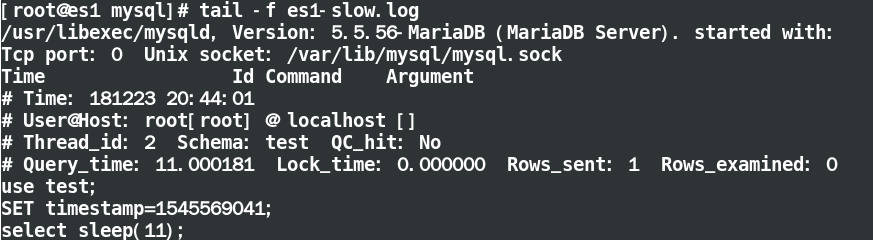
long-quert-time 默认10s 查询超过10s会记录

[mysqld]

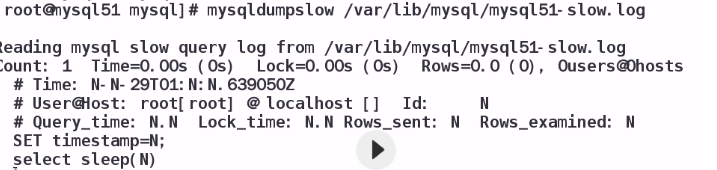
slow-query-log

log-queries-not-using-indexes 记录为使用索引的语句 主机名-slow.log

测试用：select sleep(11) 查询为空过10s查询



**统计慢查询里的sql命令**





mysql的从库是一定要有与主库相同的表的，因为从库从主库binlog日志拿回来在relay-log中，但是relay-log是只存放语句的，没有表就会报错，sql线程就会挂。

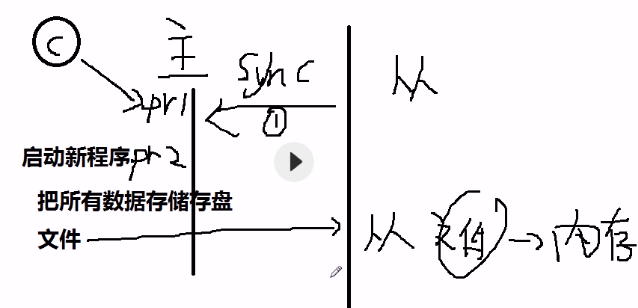


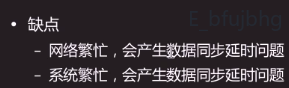
Redis就没有这个限制，没有字段，或者自己多字段，都没有影响，会同步（没有表结构的限制），多的字段被主库传过来的存储存盘文件覆盖在**内存**中。

Redis是memcache的升级版，他们都是存在内存中的，但是redis会定时存在硬盘中，memcache不会！！！



**redis的主从同步**





set v1 10

SLAVEOF 192.168.4.51 6351

info replication

SLAVEOF no one 恢复为主库

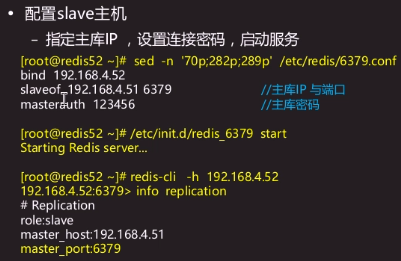
config set masterauth 123456 配置文件添加主机密码

在配置文件中添加主库永久生效，和mysql不同。Mysql没有看过在配置里改的

**主库设置密码** /etc/init.d/... 

redis-cli -h 192.168.4.51 -p 6351连接数据库 ----->auth 123456 认证(就是输入密码)才能看key \*

**从库配置认证**



**端口不一定哦，记得改**

**如果配置文件里设了密码，启动的脚本 /etc/init.d/redis\_6379就启动不了**

**Redis-cli -h 192.168.1.4.53 -p 6353 -a 123456 shutdown 手动关闭服务**

哨兵模式：redis-sentinel redis的另一种**模式**

自己创建/etc/sentinel文件 源码包目录里有模版