NSD DBA2 DAY02

1 案例1：实现MySQL读写分离

1.1 问题

- 搭建一主一从结构
- 配置maxscale代理服务器
- 测试分离配置

1.

1.2 方案

使用4台RHEL 7虚拟机，如图-1所示。其中192.168.4.10和192.168.4.20，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192.168.4.100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。linux客户机用于测试配置，可以使用真机代替

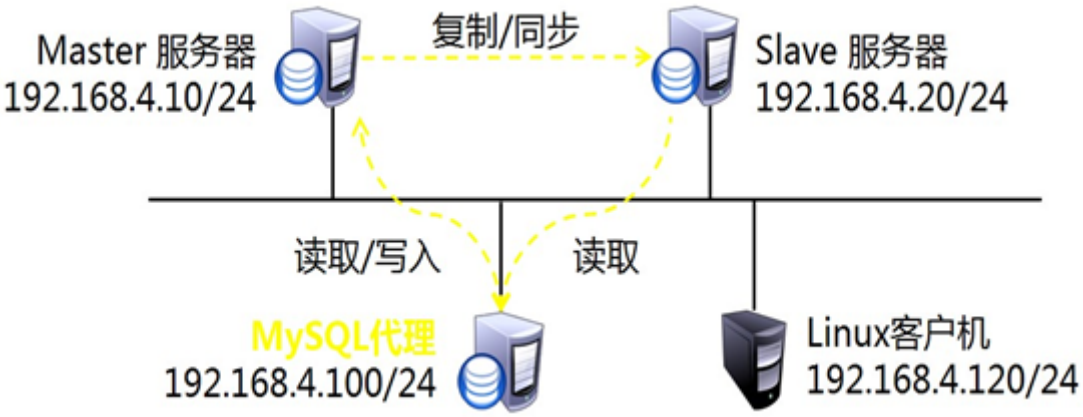

图 - 1

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：搭建主从

1）搭建一主一从结构，主库192.168.4.10上面操作

```
[root@master10 ~]# vim /etc/my.cnf
[mysqld]
server_id=10    //指定服务器ID号
log-bin=master10      //启用binlog日志，并指定文件名前缀
...
```

Top

```
[root@master10 ~]# systemctl restart mysqld       //重启mysqld
```

2）从库192.168.4.20上面操作

```
[mysqld]
server_id=20          //指定服务器ID号，不要与Master的相同
log-bin=slave20       //启动SQL日志，并指定文件名前缀
read_only=1           //只读模式
...
[root@slave20 ~]# systemctl restart  mysqld
```

3）主库授权一个用户并查看master的状态

```
[root@master10 ~]# mysql -u root -p123456
mysql> grant all on *.* to 'replicater'@'%' identified by '123456';
Query OK, 0 rows affected, 1 warning (0.00 sec)
mysql> show master status;
+-----------------+----------+--------------+------------------+-------------------+
| File            | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----------------+----------+--------------+------------------+-------------------+
| master10.000002 |      449 |              |                  |                   |
+-----------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

4）从库通过CHANGE MASTER语句指定MASTER服务器的IP地址、同步用户名/密码、起始日志文件、偏移位置（参考MASTER上的状态输出）

```
[root@slave20 ~]# mysql -u root -p123456
mysql> change master to master_host='192.168.4.10',
    -> master_user='replicater',
    -> master_password='123456',
    -> master_log_file='master10.000002',
    -> master_log_pos=738;
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected (0.01 sec)
```

Top

```
11.
12.  mysql> show  slave status\G;
13.  *************************** 1. row ***************************
14.                Slave_IO_State: Waiting for master to send event
15.                   Master_Host: 192.168.4.10
16.                   Master_User: replicater
17.                   Master_Port: 3306
18.                 Connect_Retry: 60
19.               Master_Log_File: master10.000002
20.           Read_Master_Log_Pos: 738
21.                Relay_Log_File: slave20-relay-bin.000002
22.                 Relay_Log_Pos: 319
23.         Relay_Master_Log_File: master10.000002
24.              Slave_IO_Running: Yes
25.             Slave_SQL_Running: Yes
26.               Replicate_Do_DB:
27.           Replicate_Ignore_DB:
28.            Replicate_Do_Table:
29.        Replicate_Ignore_Table:
30.       Replicate_Wild_Do_Table:
31.   Replicate_Wild_Ignore_Table:
32.                    Last_Errno: 0
33.                    Last_Error:
34.                  Skip_Counter: 0
35.           Exec_Master_Log_Pos: 738
36.               Relay_Log_Space: 528
37.               Until_Condition: None
38.                Until_Log_File:
39.                 Until_Log_Pos: 0
40.            Master_SSL_Allowed: No
41.            Master_SSL_CA_File:
42.            Master_SSL_CA_Path:
43.               Master_SSL_Cert:
44.             Master_SSL_Cipher:
45.                Master_SSL_Key:
46.         Seconds_Behind_Master: 0
47.  Master_SSL_Verify_Server_Cert: No
48.                  Last_IO_Errno: 0
49.                  Last_IO_Error:
50.                 Last_SQL_Errno: 0
51.                 Last_SQL_Error:
```

Top

```
52.      Replicate_Ignore_Server_Ids:
53.                 Master_Server_Id: 10
54.                      Master_UUID: 95ada2c2-bb24-11e8-abdb-525400131c0f
55.                 Master_Info_File: /var/lib/mysql/master.info
56.                        SQL_Delay: 0
57.              SQL_Remaining_Delay: NULL
58.          Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
59.                 Master_Retry_Count: 86400
60.                        Master_Bind:
61.          Last_IO_Error_Timestamp:
62.         Last_SQL_Error_Timestamp:
63.                   Master_SSL_Crl:
64.               Master_SSL_Crlpath:
65.               Retrieved_Gtid_Set:
66.                Executed_Gtid_Set:
67.                    Auto_Position: 0
68.             Replicate_Rewrite_DB:
69.                     Channel_Name:
70.               Master_TLS_Version:
71. 1 row in set (0.00 sec)
```

5）测试，主库创建aa库

```
01. mysql> create database aa;
02. Query OK, 1 row affected (0.00 sec)
03.
04. mysql> show databases;
05. +--------------------+
06. | Database           |
07. +--------------------+
08. | information_schema |
09. | aa                 |
10. | mysql              |
11. | performance_schema |
12. | sys                |
13. +--------------------+
14. 5 rows in set (0.00 sec)
```

Top

6）从库上面查看，有aa库

```
mysql> show databases;

+--------------------+
| Database           |
+--------------------+
| information_schema |
| aa                 |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

步骤二：实现mysql读写分离

1）配置数据读写分离服务器192.168.4.100

环境准备关闭防火墙和SElinux，保证yum源可以正常使用

```
[root@maxscale ~]# cd mysql/
[root@maxscale mysql]# ls
maxscale-2.1.2-1.rhel.7.x86_64.rpm
[root@maxscale mysql]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86_64.rpm
//安装maxscale
warning: maxscale-2.1.2-1.rhel.7.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 816
Preparing...                          ################################# [100%]
Updating / installing...
   1:maxscale-2.1.2-1                 ################################# [100%]
```

2）配置maxscale

```
[root@maxscale mysql]# vim /etc/maxscale.cnf.template
[maxscale]
threads=auto            //运行的线程的数量

[server1]               //定义数据库服务器
type=server
address=192.168.4.10        //数据库服务器的ip
port=3306
```

Top

```
09.  protocol=MySQLBackend       //后端数据库
10.
11.
12.
13.  [server2]
14.  type=server
15.  address=192.168.4.20
16.  port=3306
17.  protocol=MySQLBackend
18.
19.
20.
21.  [MySQL Monitor]              //定义监控的数据库服务器
22.  type=monitor
23.  module=mysqlmon
24.  servers=server1,server2      //监控的数据库列表，不能写ip
25.  user=scalemon                //监视数据库服务器时连接的用户名scalemon
26.  passwd=123456                //密码123456
27.  monitor_interval=10000       //监视的频率 单位为秒
28.
29.
30.
31.  #[Read-Only Service]         //不定义只读服务器
32.  #type=service
33.  #router=readconnroute
34.  #servers=server1
35.  #user=myuser
36.  #passwd=mypwd
37.  #router_options=slave
38.
39.
40.
41.  [Read-Write Service]         //定义读写分离服务
42.  type=service
43.  router=readwritesplit
44.  servers=server1,server2
45.  user=maxscaled          //用户名 验证连接代理服务时访问数据库服务器的用户是否存在
46.  passwd=123456             //密码
47.  max_slave_connections=100%
48.
49.
```

Top

```
50.
51.  [MaxAdmin Service]       //定义管理服务
52.  type=service
53.  router=cli
54.
55.
56.
57.  #[Read-Only Listener]       //不定义只读服务使用的端口号
58.  #type=listener
59.  #service=Read-Only Service
60.  #protocol=MySQLClient
61.  #port=4008
62.
63.
64.
65.  [Read-Write Listener]          //定义读写服务使用的端口号
66.  type=listener
67.  service=Read-Write Service
68.  protocol=MySQLClient
69.  port=4006
70.
71.
72.
73.  [MaxAdmin Listener]       //管理服务使用的端口号
74.  type=listener
75.  service=MaxAdmin Service
76.  protocol=maxscaled
77.  socket=default
78.  port=4099    //手动添加，不指定时使用的是默认端口在启动服务以后可以知道默认端口
```

3）根据配置文件的设置，在数据库服务器上添加授权用户（主库执行，从库查看）

```
01.  mysql> grant replication slave,replication client on *.* to  scalemon@'%' identified by "123
02.  Query OK, 0 rows affected, 1 warning (0.00 sec)
03.
04.  mysql> grant select on mysql.* to maxscaled@"%" identified by "123456";
05.  //验证 访问数据时，连接数据库服务器使用的用户，是否在数据库服务器上存在的，连
06.  Query OK, 0 rows affected, 1 warning (0.01 sec)
```

Top

4）查看授权用户

在主库上面查看

```
mysql> select user,host from mysql.user where user in ("scalemon","maxscaled");
+-----------+------+
| user      | host |
+-----------+------+
| maxscaled | %    |
| scalemon  | %    |
+-----------+------+
2 rows in set (0.00 sec)
```

在从库上面查看

```
mysql> select user,host from mysql.user where user in ("scalemon","maxscaled");
+-----------+------+
| user      | host |
+-----------+------+
| maxscaled | %    |
| scalemon  | %    |
+-----------+------+
2 rows in set (0.00 sec)
```

测试授权用户

```
[root@maxscale mysql]# mysql -h 192.168.4.10 -u scalemon -p123456
[root@maxscale mysql]# mysql -h 192.168.4.20 -u scalemon -p123456
[root@maxscale mysql]# mysql -h 192.168.4.10 -u maxscaled -p123456
[root@maxscale mysql]# mysql -h 192.168.4.20 -u maxscaled -p123456
```

5）启动服务

```
[root@maxscale ~]# maxscale -f  /etc/maxscale.cnf
[root@maxscale ~]# ps -C  maxscale       //查看进程
  PID TTY          TIME CMD
17930 ?        00:00:00 maxscale
[root@maxscale ~]# netstat  -antup | grep maxscale      //查看端口
```

Top

```
tcp6       0      0 :::4099                 :::*                    LISTEN      17930/maxscale
tcp6       0      0 :::4006                 :::*                    LISTEN      17930/maxscale
```

6）测试，在本机访问管理端口查看监控状态

maxadmin -P端口 -u用户名 -p密码

```
[root@maxscale ~]# maxadmin -P4099 -uadmin  -pmariadb
MaxScale>
MaxScale> list servers
Servers.
-------------------+-----------------+-------+-------------+---------------
Server             | Address         | Port  | Connections | Status
-------------------+-----------------+-------+-------------+---------------
server1            | 192.168.4.10    |  3306 |           0 | Master, Running
server2            | 192.168.4.20    |  3306 |           0 | Slave, Running
-------------------+-----------------+-------+-------------+---------------
```

7）在客户端访问读写分离服务器（没有mysql命令可以安装）

mysql -h读写分离服务ip -P4006 -u用户名 -p密码

```
[root@slave53 ~]# mysql -h192.168.4.100  -P4006 -ureplicater -p123456
mysql> select  @@hostname;          //查看当前主机名
+-------------+
| @@hostname |
+-------------+
| slave20    |
+-------------+
1 row in set (0.00 sec)
mysql> create table t2(id int(4) );
Query OK, 0 rows affected (0.02 sec)

mysql> insert into aa.t2 values(777);
Query OK, 1 row affected (0.01 sec)
```

在主库上面查看

Top

```
mysql> use aa
```

```
mysql> select * from t2;
+------+
| id   |
+------+
|  777 |
+------+
1 row in set (0.00 sec)
```

从库（主库同步到从库）

```
mysql> use aa
mysql> select * from t2;
+------+
| id   |
+------+
|  777 |
+------+
1 row in set (0.00 sec)
```

2 案例2：配置MySQL多实例

2.1 问题

- 在主机192.168.4.56上，配置第1个MySQL实例
- 实例名称mysql1、端口3307
- 数据库目录/data3307、pid文件mysql1.pid
- 错误日志mysql1.err
- 在主机192.168.4.56上，配置第2个MySQL实例
- 实例名称mysql2、端口3308
- 数据库目录/data3308、pid文件mysql2.pid
- 错误日志mysql2.err

步骤一：配置多实例（192.168.4.56上面操作）

什么是多实例：

在一台物理主机上运行多个数据库服务，可以节约运维成本，提高硬件利用率

1）解压软件、修改目录名

```
[root@mysql ~]# cd mysql/
[root@mysql mysql]# ls
mysql-5.7.20-linux-glibc2.12-x86_64.tar.gz
[root@mysql mysql]# tar -xf mysql-5.7.20-linux-glibc2.12-x86_64.tar.gz
```

Top

```
[root@mysql mysql]# mv  mysql-5.7.20-linux-glibc2.12-x86_64 /usr/local/mysql
```

2）调整PATH变量

```
[root@mysql mysql]# echo  "export  PATH=/usr/local/mysql/bin:$PATH" \
 >> /etc/profile
[root@mysql mysql]# source /etc/profile
[root@mysql mysql]# echo $PATH
/usr/local/mysql/bin:/usr/local/mycat/bin:/usr/local/mycat/bin:/usr/local/sbin:/usr/lo
```

3）编辑主配置文件/etc/my.cnf

每个实例要有独立的数据库目录、监听端口号、实例名称和独立的sock文件

```
[mysqld_multi]        //启用多实例
mysqld = /usr/local/mysql/bin/mysqld_safe        //指定进程文件路径
mysqladmin = /usr/local/mysql/bin/mysqladmin    //指定管理命令路径
user = root        //指定进程用户

[mysqld1]        //实例进程名称
port=3307        //端口号
datadir=/data3307        //数据库目录 ，要手动创建
socket=/data3307/mysqld.sock        //指定sock文件的路径和名称
pid-file=/data3307/mysql1.pid        //进程pid号文件位置
log-error=/data3307/mysql1.err        //错误日志位置

[mysqld2]
port=3308
datadir=/data3308
socket=/data3308/mysqld.sock
pid-file=/data3308/mysql2.pid
log-error=/data3308/mysql2.err
```

4）创建数据库目录

```
[root@mysql mysql]# mkdir -p /data3307
[root@mysql mysql]# mkdir -p /data3308
```

Top

5）创建进程运行的所有者和组 mysql

```
[root@mysql mysql]# useradd mysql
[root@mysql mysql]# chown  mysql:mysql  /data*
```

6）初始化授权库

```
[root@mysql mysql]# mysqld --user=mysql --basedir=/usr/local/mysql
--datadir=/data3307 --initialize
...
2018-09-26T07:07:33.443378Z 1 [Note] A temporary password is generated for root@lo
[root@mysql mysql]# mysqld --user=mysql --basedir=/usr/local/mysql
 --datadir=/data3308 --initialize
...
2018-09-26T07:08:07.770289Z 1 [Note] A temporary password is generated for root@lo
```

7）启动多实例

```
[root@mysql mysql]# mysqld_multi  start 1      //1为实例编号
[root@mysql mysql]# mysqld_multi  start 2
```

8）查看端口

```
[root@mysql mysql]# netstat -utnlp | grep :3307
tcp6       0      0 :::3307                 :::*                    LISTEN      21009/mysqld
[root@mysql mysql]# netstat -utnlp | grep :3308
tcp6       0      0 :::3308                 :::*                    LISTEN      21177/mysqld
[root@mysql mysql]# ps -C mysqld
  PID TTY          TIME CMD
21009 pts/1    00:00:00 mysqld
21177 pts/1    00:00:00 mysqld
```

9）访问多实例

使用初始化密码登录多实例1

Top

```
[root@mysql mysql]# mysql -u root -p'7L?Vi!dGKmgu' -S /data3307/mysqld.sock
mysql> alter user root@"localhost" identified by '123456';        //修改密码
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

使用初始化密码登录多实例2

```
[root@mysql bin]# mysql -u root -p'kC)Bby Up1a-b' -S /data3307/mysqld.sock
mysql> alter user root@"localhost" identified by '123456';        //修改密码
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

10）创建库

```
mysql> create database db1;
Query OK, 1 row affected (0.00 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
```

Top

```
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

11）停止启动的实例服务

mysqld_multi --user=root --password=密码 stop 实例编号

```
[root@mysql mysql]# mysqld_multi --user=root --password=123456 stop 1
[root@mysql mysql]# netstat -utnlp | grep :3307      //查看没有端口
[root@mysql mysql]# mysqld_multi --user=root --password=123456 stop 2
[root@mysql mysql]# netstat -utnlp | grep :3308      //查看没有端口
[root@mysql mysql]# mysql -uroot -p123456 -S /data3307/mysqld.sock
//拒绝连接
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/data3307/my
```

3 案例3：MySQL性能优化

3.1 问题

- 练习相关优化选项
- 启用慢查询日志
- 查看各种系统变量、状态变量

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：mysql性能优化

1）查看服务运行时的参数配置

```
mysql> show variables\G;
......
*************************** 171. row ***************************
Variable_name: innodb_log_file_size
        Value: 50331648
*************************** 172. row ***************************
Variable_name: innodb_log_files_in_group
        Value: 2
```

Top

```
*************************** 173. row ***************************
Variable_name: innodb_log_group_home_dir
        Value: ./
*************************** 174. row ***************************
Variable_name: innodb_log_write_ahead_size
        Value: 8192
*************************** 175. row ***************************
Variable_name: innodb_lru_scan_depth
        Value: 1024
*************************** 176. row ***************************
Variable_name: innodb_max_dirty_pages_pct
        Value: 75.000000
*************************** 177. row ***************************
Variable_name: innodb_max_dirty_pages_pct_lwm
        Value: 0.000000
*************************** 178. row ***************************
Variable_name: innodb_max_purge_lag
        Value: 0
......

mysql> show variables like "%innodb%";
+---------------------------------------+----------------------+
| Variable_name                         | Value                |
+---------------------------------------+----------------------+
| ignore_builtin_innodb                 | OFF                  |
| innodb_adaptive_flushing              | ON                   |
| innodb_adaptive_flushing_lwm          | 10                   |
| innodb_adaptive_hash_index            | ON                   |
| innodb_adaptive_hash_index_parts      | 8                    |
| innodb_adaptive_max_sleep_delay       | 150000               |
......
......
| innodb_undo_log_truncate              | OFF                  |
| innodb_undo_logs                      | 128                  |
| innodb_undo_tablespaces               | 0                    |
| innodb_use_native_aio                 | ON                   |
| innodb_version                        | 5.7.17               |
| innodb_write_io_threads               | 4                    |
+---------------------------------------+----------------------+
134 rows in set (0.01 sec)
```

Top

2）并发连接数量

查看当前已经使用的连接数

```
mysql> flush status;
Query OK, 0 rows affected (0.00 sec)
mysql> show global status like "Max_used_connections";
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 3     |
+----------------------+-------+
1 row in set (0.00 sec)
```

查看默认的最大连接数

```
mysql> show variables like "max_connections%";
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 151   |
+-----------------+-------+
1 row in set (0.00 sec)
```

3）连接超时时间

```
mysql> show variables like "%timeout%";
+-----------------------------+----------+
| Variable_name               | Value    |
+-----------------------------+----------+
| connect_timeout             | 10       |
| delayed_insert_timeout      | 300      |
| have_statement_timeout      | YES      |
| innodb_flush_log_at_timeout | 1        |
| innodb_lock_wait_timeout    | 50       |
| innodb_rollback_on_timeout  | OFF      |
| interactive_timeout         | 28800    |
| lock_wait_timeout           | 31536000 |
| net_read_timeout            | 30       |
```

Top

```
| net_write_timeout          | 60       |
| rpl_stop_slave_timeout     | 31536000 |
| slave_net_timeout          | 60       |
| wait_timeout               | 28800    |
+----------------------------+----------+
13 rows in set (0.00 sec)
```

4）允许保存在缓存中被重用的线程数量

```
mysql> show variables like "thread_cache_size";
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| thread_cache_size | 9     |
+-------------------+-------+
1 row in set (0.00 sec)
```

5）用于MyISAM引擎的关键索引缓存大小

```
mysql> show variables like "key_buffer_size";
+-----------------+---------+
| Variable_name   | Value   |
+-----------------+---------+
| key_buffer_size | 8388608 |
+-----------------+---------+
1 row in set (0.00 sec)
```

6）为每个要排序的线程分配此大小的缓存空间

```
mysql> show variables like "sort_buffer_size";
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| sort_buffer_size | 262144 |
+------------------+--------+
1 row in set (0.00 sec)
```

Top

7）为顺序读取表记录保留的缓存大小

```
mysql> show variables like "read_buffer_size";
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| read_buffer_size | 131072 |
+------------------+--------+
1 row in set (0.01 sec)
```

8）为所有线程缓存的打开的表的数量

```
mysql> show variables like "table_open_cache";
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| table_open_cache | 2000  |
+------------------+-------+
1 row in set (0.00 sec)
```

步骤二：SQL查询优化

1）常用日志种类及选项，如图-1所示：

类 型	用 途	配 置
错误日志	记录启动/运行/停止过程中的错误消息	log-error[=name]
查询日志	记录客户端连接和查询操作	general-log 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 slow-query-log-file= long-query-time=

图-1

记录慢查询，图-2所示：

Top

选 项	含 义
slow-query-log	启用慢查询
slow-query-log-file	指定慢查询日志文件
long-query-time	超过时间（默认10秒）
log-queries-not-using-indexes	记录未使用索引的查询

启用慢查询日志

```
[root@master10 ~]# vim /etc/my.cnf
...
slow_query_log=1
slow_query_log_file=mysql-slow.log
long_query_time=5
log_queries_not_using_indexes=1
...
[root@master10 ~]# systemctl restart mysqld
```

2）查看慢查询日志

```
[root@master10 ~]# mysqldumpslow  /var/lib/mysql/mysql-slow.log

Reading mysql slow query log from /var/lib/mysql/mysql-slow.log
Count: 1 Time=0.00s (0s)  Lock=0.00s (0s)  Rows=0.0 (0), 0users@0hosts
```

查看缓存的大小

```
mysql> show variables like "query_cache%";
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | OFF     |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
5 rows in set (0.00 sec)
```

Top

3）查看当前的查询缓存统计

```
mysql> show global status like "qcache%";
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Qcache_free_blocks      | 1       |
| Qcache_free_memory      | 1031832 |
| Qcache_hits             | 0       |
| Qcache_inserts          | 0       |
| Qcache_lowmem_prunes    | 0       |
| Qcache_not_cached       | 40      |
| Qcache_queries_in_cache | 0       |
| Qcache_total_blocks     | 1       |
+-------------------------+---------+
8 rows in set (0.00 sec)
```

Top