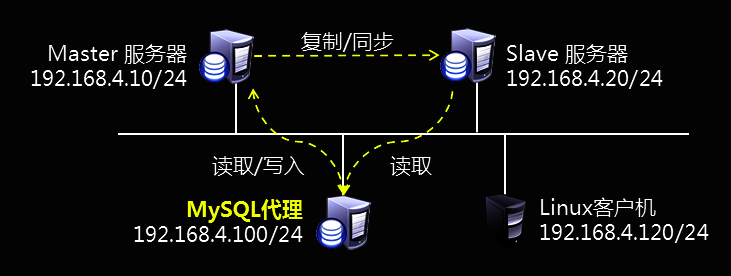
**读写分离，性能调优**

## mysql数据读写分离:

### 把可客户端查询数据的请求分别给不同的数据库服务器处理



# 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

# 构建读写分离

**基本构建思路**

**1. 已搭建好 MySQL 主从复制**

– 基于上一个实验的结果

– 其中 Slave 为只读

**2. 添加一台 MySQL 代理服务器**

– 部署 / 启用 maxscale

**3. 客户端通过代理主机访问 MySQL 数据库**

– 访问代理服务器

# 配置MYSQL数据读写分离结构

**1 配置MYSQL 一主一从 同步结构：要求把51配置为52的从库**

1. **准备代理服务器**

把maxscale-2.1.2-1.rhel.7.x86\_64.rpm软件拷贝到主机53

给53主机其名 proxy53

停止MYSQL数据库服务

1. **配置代理服务器主机53（）**
   1. 装包（提供命令max[两次tab键]）

[root@mysql53 ~]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

* 1. 修改配置文件

[root@mysql53 ~]# cp /etc/maxscale.cnf /root/ //备份配置文件

[root@mysql53 ~]# vim /etc/maxscale.cnf //进入修改配置文件

#服务启动之后线程数量

9 [maxscale] //配置项名称

10 threads=auto //根据CPU的核数指定线程数量

#定义数据库服务器的名称和ip地址

#定义第1台

18 [server1]

19 type=server

20 address=192.168.4.51

21 port=3306

22 protocol=MySQLBackend

#定义第2台

24 [server2]

25 type=server

26 address=192.168.4.52

27 port=3306

28 protocol=MySQLBackend

#定义代理服务器启动之后要监控的数据库服务器，以及连接的用户名和密码

36 [MySQL Monitor]

37 type=monitor

38 module=mysqlmon

39 servers=server1,server2 //服务器列表

40 user=scalemon //定义监控时使用的用户名（自定义）

41 passwd=123asd...A //定义监控时的用户名密码（自定义）

42 monitor\_interval=10000

#不定义只读服务器

53 #[Read-Only Service]

54 #type=service

55 #router=readconnroute

56 #servers=server1

57 #user=myuser

58 #passwd=mypwd

59 #router\_options=slave

#定义读写分离服务

64 [Read-Write Service]

65 type=service

66 router=readwritesplit

67 servers=server1,server2 //数苦苦服务器列表

68 user=maxscale //定义客户端连接数据库时使用的用户名

69 passwd=123asd...A //定义客户端连接数据库时使用的用户名密码

70 max\_slave\_connections=100%

#定义管理服务

76 [MaxAdmin Service]

77 type=service

78 router=cli

#不定义只读服务器

86 #[Read-Only Listener]

87 #type=listener

88 #service=Read-Only Service

89 #protocol=MySQLClient

90 #port=4008

#定义读写分离默认使用的端口号

92 [Read-Write Listener]

93 type=listener

94 service=Read-Write Service

95 protocol=MySQLClient

96 port=4006

#管理服务使用的端口号

98 [MaxAdmin Listener]

99 type=listener

100 service=MaxAdmin Service

101 protocol=maxscaled

102 socket=default

103 port=4016

* 1. 根据配置文件的设置，在数据库服务器上添加对应的授权用户

对应的授权用户scalemon 和 maxscale

在主库52 做用户授权，51主机会自动把授权同步到本机

1. 授权监控用户

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

1. 授权路由用户

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

1. 授权对数据有访问权限的用户

mysql> create database db2;

mysql> create table db2.a(id int);

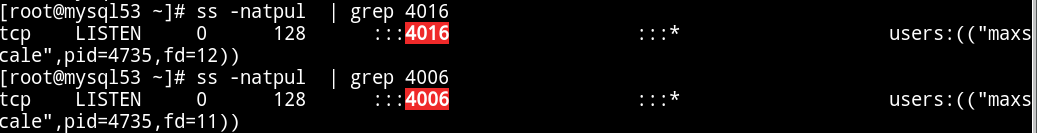
mysql> grant select,insert on db2.\* to webuser@"%" identified by "123asd...A";

* 1. 启动服务并查看服务信息

[root@mysql53 ~]# maxscale -f /etc/maxscale.cnf

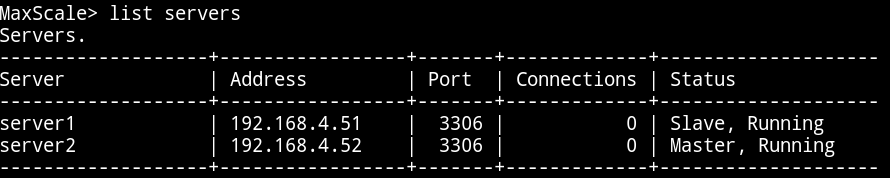
[root@mysql53 ~]# ss -natpul | grep 4016

[root@mysql53 ~]# ss -natpul | grep 4006



* 1. 访问管理服务查看配置信息

[root@mysql53 ~]# maxadmin -uadmin -pmariadb -P4016



* 1. 测试读写分离配置：在客户端连接代理服务器，访问数据，能够实现数据读写分离功能

在客户端50主机上连接53，在51从库上插入数据，分别在52、51和50上查看，会发现51和50上有插入的数据，在50上插入数据，会在51和52上同时有数据

[root@client50 ~]# mysql -uwebuser -h192.168.4.53 -P4006 -p123asd...A

52: mysql> insert into db2.a values(123); //在52上插入数据，会同步到51

50: mysql> select \* from db2.a; //在50上查看

51: mysql> insert into db2.a values(456); //在51上插入数据，不会同步到52

52: mysql> select \* from db2.a; //在52上查看不到在51上插入的数据

50: mysql> select \* from db2.a; //在50上可以查看到在51上插入的数据

50: mysql> insert into db2.a values(888); //在50上插入数据

51: mysql> select \* from db2.a; //在51上能看到插入的数据

52: mysql> select \* from db2.a; //在52上能看到在51上插入的数据

将51上slave服务停止，在50上访问的所有操作均交给52执行，不影响访问和写入，当服务启动时，会自动同步在服务器宕机期间产生的数据；将数据库服务停止，不影响访问和写入

将52数据库服务停止，只是不同步数据到从库，在50上访问不影响访问和写入

## mysql多实例

优点：节约运维成本

提高硬件利用率

### 在主机50上做多实例实验

把软件mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz拷贝到主机50里

### 多实例：在1台服务上运行多个数据库服务器

# 1. 停止其他软件提供的MYSQL服务

[root@client50 ~]# systemctl stop mysqld

# 2. 安装提供多实例服务的MYSQL软件包

[root@client50 ~]# tar -zxvf /root/mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

[root@client50 ~]# mv /root/mysql-5.7.20-linux-glibc2.12-x86\_64/

/usr/local/mysql

[root@client50 ~]# ls /usr/local/mysql/

# 3. 创建主配置文件

[root@client50 ~]# mv /etc/my.cnf{,.bak}

[root@client50 ~]# vim /etc/my.cnf

[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，1表示示例编号

datadir=/datadir3307 //数据库目录，要手动创建

socket=/datadir3307/mysql3307.sock //指定sock文件的路径和名称

port=3307 //端口号

log-error=/datadir3307/mysqld3307.log //错误日志文件

pid-file=/datadir3307/mysqld3307.pid //指定pid号文件位置

[mysqld2]

datadir=/datadir3308

socket=/datadir3308/mysql3308.sock

port=3308

log-error=/datadir3308/mysqld3308.log

pid-file=/datadir3308/mysqld3308.pid

[root@client50 ~]# mkdir /datadir3307

[root@client50 ~]# mkdir /datadir3308

# 4. 启动多实例服务

[root@client50 ~]# /usr/local/mysql/bin/mysqld\_multi start 1

..........

2018-10-23T09:16:25.201603Z 1 [Note] A temporary password is generated for root@localhost: sLcw?E!Hq8Bi-------->初始密码

# 5. 访问多实例服务

[root@client50 ~]# ss -natpul | grep mysqld

[root@client50 ~]# /usr/local/mysql/bin/mysql -uroot -p'sLcw?E!Hq8Bi' -S /datadir3307/mysql3307.sock

**修改登陆密码：**

mysql> alter user user() identified by '123456';

**使用修改后的密码登陆**

[root@client50 ~]# /usr/local/mysql/bin/mysql -uroot -p123456 -S /datadir3307/mysql3307.sock

**建库、建表**

mysql> show databases;

mysql> create database db2;

**查看数据库目录文件**

[root@client50 ~]# ls /datadir3307/

# 6. 停止多实例服务

[root@client50 ~]# /usr/local/mysql/bin/mysqld\_multi --user=root

--password=123456 stop 1

[root@client50 ~]# ss -natpul | grep mysqld

## mysql性能调优（理论）

# 提高MYSQL系统的性能、响应速度

— 替换有问题的硬件（CPU/磁盘/内存等）

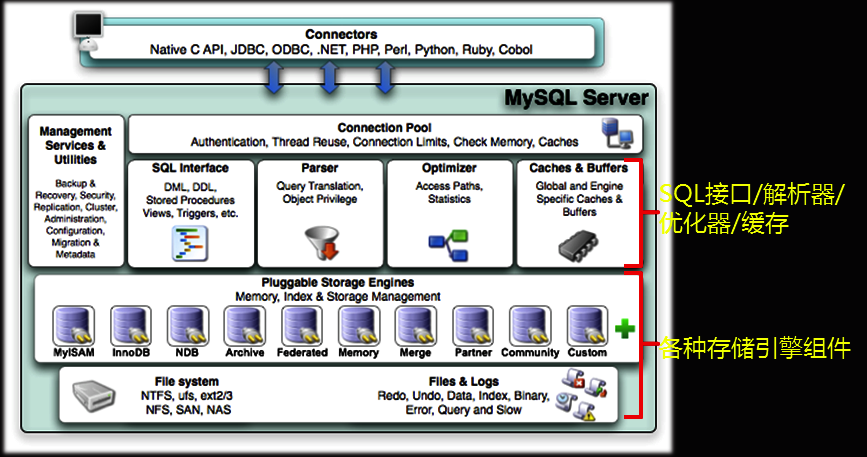
— 服务程序的运行参数调整

— 对SQL查询进行优化

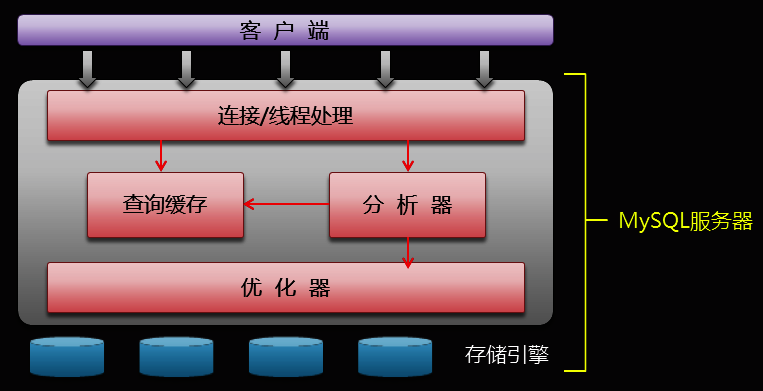
**【搭建监控服务器：监控服务的硬件使用率**

CPU 内存 存储】

### MYSQL体系结构



### MYSQL 执行流程



cpu查看方法 uptime

### 数据库服务运行参数：

**并发及连接控制**

|  |  |
| --- | --- |
| 选项 | 含义 |
| max\_connections | 允许的最大并发连接数 |
| connect\_timeout | 等待连接超时，默认10秒，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） |

**查看全局变量信息**

mysql> show variables;

**查看连接信息**

mysql> show variables like "%connect%";

**查看当前已使用的连接数（最大连接数）**

mysql> show variables like "max\_connections";

mysql> set global max\_connections = 300;（设置最大连接数）

**查看当前连接状态（最大连接数）**

mysql> show global status like "%connect%";

mysql> show global status like "Max\_used\_connections";

**查看当前连接信息**

mysql> show processlist;

**查看等待连接超时时间**

mysql> show variables like "connect\_timeout";

**查看等待关闭连接的不活动超时秒数**

mysql> show variables like "wait\_timeout";

### 缓存参数控制

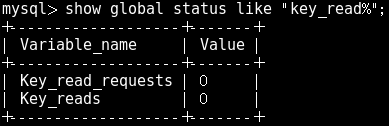
**缓冲区、线程数量、开表数量**

|  |  |
| --- | --- |
| 选项 | 含义 |
| key\_buffer-size | 用于 MyISAM 引擎的关键索引缓存大小 |
| sort\_buffer\_size | 为每个要排序的线程分配此大小的缓存空间 |
| read\_buffer\_size | 为顺序读取表记录保留的缓存大小 |
| thread\_cache\_size | 允许保存在缓存中被重用的线程数量 |
| table\_open\_cache | 为所有线程缓存的打开的表的数量 |

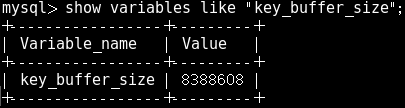
### key\_buffer\_size=8M

**当Key\_reads / Key\_read\_request较低时可适当加大此缓存值**

mysql>show global status like “key\_read%”;



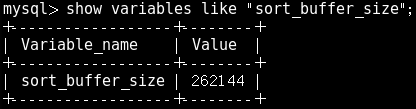
mysql>show variables like “key\_buffer\_size”;



### sort\_buffer\_size=256K

增大此值可提高ORDER和GROUP的速度

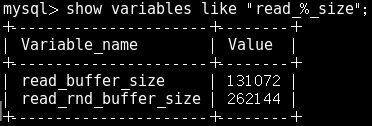
mysql>show variables like “sort\_buffer\_size”;



### 查看表记录读取缓存

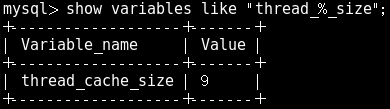
此缓存值影响SQL查询的响应速度

mysql>show variables like “read\_%\_size”;



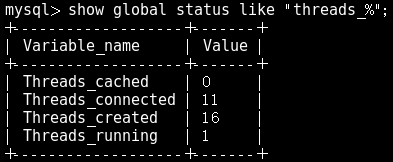
### 查看可重用线程数

mysql>show variables like “thread\_%\_size”;



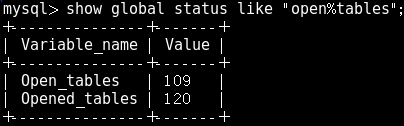
### 查看当前的线程重用状态

msyql>show global status like “threads\_%”;



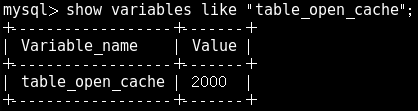
### 查看已打开、打开过多少个表

mysql>show global status like “open%tables”;



### 查看可缓存多少个打开的表

mysql>show variabels like “table\_open\_cache”;



# SQL查询优化

## MySQL日志类型

### 常用日志种类及选项

|  |  |  |
| --- | --- | --- |
| **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=指定名称和位置 |
| 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=指定名称和位置  long-query-time=指定超时时间（默认10秒） |

在配置文件中加入general-log选项，记录所有的sql命令

[root@mysql51 ~]# vim /etc/my.cnf

[mysqld]

general-log

[root@mysql51 ~]# tail -f /var/lib/mysql/mysql51.log //动态查看日志

## 优化SQL查询

### 记录慢查询

|  |  |
| --- | --- |
| **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 |

### 查看慢查询日志

**使用mysqldumpslow工具（统计慢查询日志记录的个数）**

**修改配置文件启用慢查询**

[root@mysql51 ~]# vim /etc/my.cnf

[mysqld]

slow-query-log

log-queries-not-using-indexes

[root@client50 ~]# mysqldumpslow /var/lib/mysql/mysql50-slow.log > /tmp/sql.txt //统计慢查询日志记录的sql命令保存到文本文件中

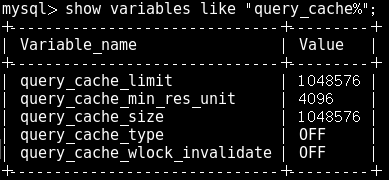
**动态查询**

[root@mysql51 ~]# tail -f /var/lib/mysql/mysql51-slow.log

## 关于缓存的查询

### 查看缓存的大小

msyql>show variables like “query\_cache%”;



query\_cache\_limit //查询结果超过设定值不允许存入查询缓存

query\_cache\_min\_res\_unit //查询缓存的最小存储单元

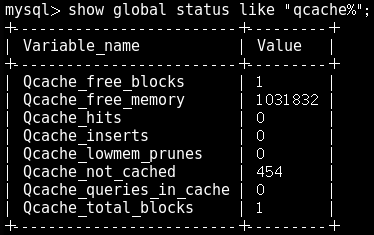
query\_cache\_size //查询缓存的大小

query\_cache\_type //查询缓存类型

query\_cache\_wlock\_invalidate //查询缓存写锁 《针对Myisam存储引擎有效》

查询缓存的最小存储单元值小可最大化利用存储空间，但会导致查询速度变慢；此值大可提高查询速度，但会导致浪费存储空间

### 查看当前的缓存统计



Qcache\_hits //查询缓存命中率

查询缓存命中率过低说明查询缓存空间过小，可适当调大查询缓存空间大小

Qcache\_inserts //查询请求总数

**优化NYSQL数据库服务的方法**：

数据库服务运行时，查看硬件使用情况（CPU、内存、磁盘）

网络接口流量

数据库服务运行时，参数配置：

show variables; //数据库服务启动时系统配置的值

show global status; //当前系统使用的值

并发连接数量 超时时间 重复使用的线程数量

多个线程同事打开表的数量

索引缓存

处理不查询方式MYSQL存储使用缓存设置

select \* from a;

select name,age from a order by age desc;

查询缓存配置

select variables like “query\_cache\_%”;

查看查询缓存统计信息

select variables like “qcache\_%”;

优化程序员编写的SQL命令

优化网络架构：结构中不要有数据传输瓶颈**；**不要有单点故障