Day 02 MySQL读写分离 MySQL多实例 MySQL性能调优

一、MySQL读写分离

1.读写分离概述

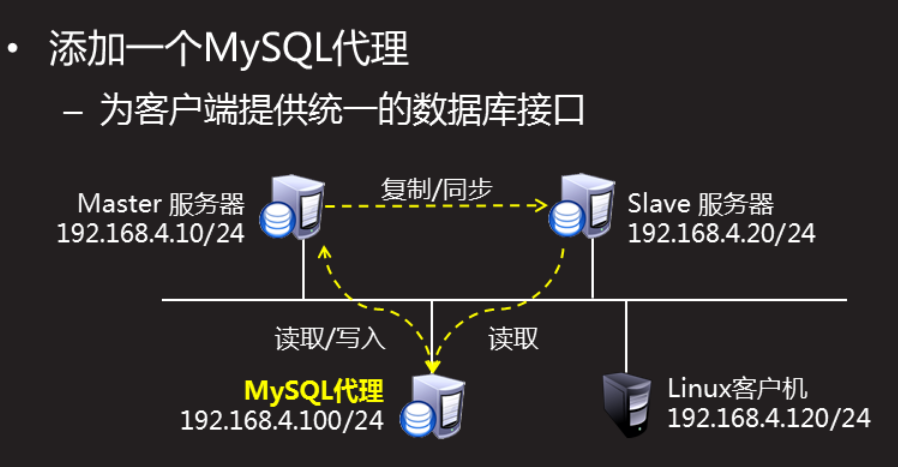
1）MySQL读写分离

·主从复制的应用局限性？

·如何分离MySQL读、写流量？

·在客户端区分，还是在服务端区分？

2）案例拓扑



3）读写分离原理

·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

2.构建读写分离

1）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

2）部署maxscale服务

·MaxScale代理软件

- 由MySQL的兄弟公司MariaDB开发

- 下载地址

https://downloads.marriadb.com/files/MaxScale

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

...

·修改配置/etc/maxscale.cnf

[server1] //定义数据库服务器主机名

type=server

address=192.168.4.10 //master主机ip地址

port=3306

protocol=MySQLBackend

[server2] //定义数据库服务器

type=server

address=192.168.4.20 //slave主机ip地址

port=3306

protocol=MySQLBackend

[MySQL Monitor] //定义要监视的数据库节点

type=monitor

module=mysqlmon

servers=server1,server2 //主、从数据库的主机名

user=scalemon //用户名

passwd=111111 //密码

monitor\_interval=10000

[Read-Write Service] //定义读写分离的数据库节点

type=service

router=readwritesplit

servers=server1,server2 //主从数据库的主机名

user=maxscale //用户名

passwd=111111 //密码

max\_slave\_connections=100%

·在主从数据库服务器创建授权用户

> grant replication slave,replication client on \*.\* to scalemon@'%' identifiedby '111111';

//创建监控用户

> grant select on mysql.\* to maxscale@'%' identified by '111111';

//创建路由用户

> grant all on \*.\* to student@'%' identified by '111111';

//创建访问数据用户

3）启动服务

·主要操作：

- 启动服务、查看端口、停止服务

# maxscale --config=/etc/maxscale.cnf

# netstat -antulp | grep maxscale

# kill -9 19081

4）测试配置

·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

# mysql -h192.168.4.100 -P4006 -ustudent -p111111

> select @@hostname;

//查看当前主机名

二、MySQL多实例

1.多实例概述

1）多实例概述

·什么是多实例

- 在一台物理主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

2.配置多实例

1）配置步骤

1.安装支持多实例服务的软件包

- 解压软件、修改目录名、调整PATH变量

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

# mv mysql-5.7.20-linux-glibc2.12-x86\_64 /usr/local/mysql

# vim /etc/profile

.. ..

export PATH=/ust/local/mysql/bin:$PATH

# source /etc/profile

2）配置文件参数说明

2.修改主配置文件

•主配置文件/etc/my.cnf

- 每个实例要有独立的数据库目录和监听端口号

- 每个实例要有独立的实例名称和独立的sock文件

[mysql\_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

[mysqlX]

port = 3307

datadir = /data3307

socket = /data3307/mysql.sock

pid-file = /data3307/mysqld.pid

log-error = /data3307/mysqld.err

3）管理多实例

3.初始化授权库

- 会提示root用户登录的初始密码

4.启动服务

•启动实例进程

•停止实例进程

[root@localhost bin]# ./mysqld --user=mysql --basedir=软件安装目录 --datadir=数据库目录 --initialize //初始化授权库

[root@stu ~]# mysqld\_multi start 实例编号 //启动实例进程

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

4）客户端访问

5.客户端访问测试

•本机连接

- 使用初始密码连接

- 修改本机登录密码

- 连接实例

# ./mysql -uroot -p初始密码 -S sock文件

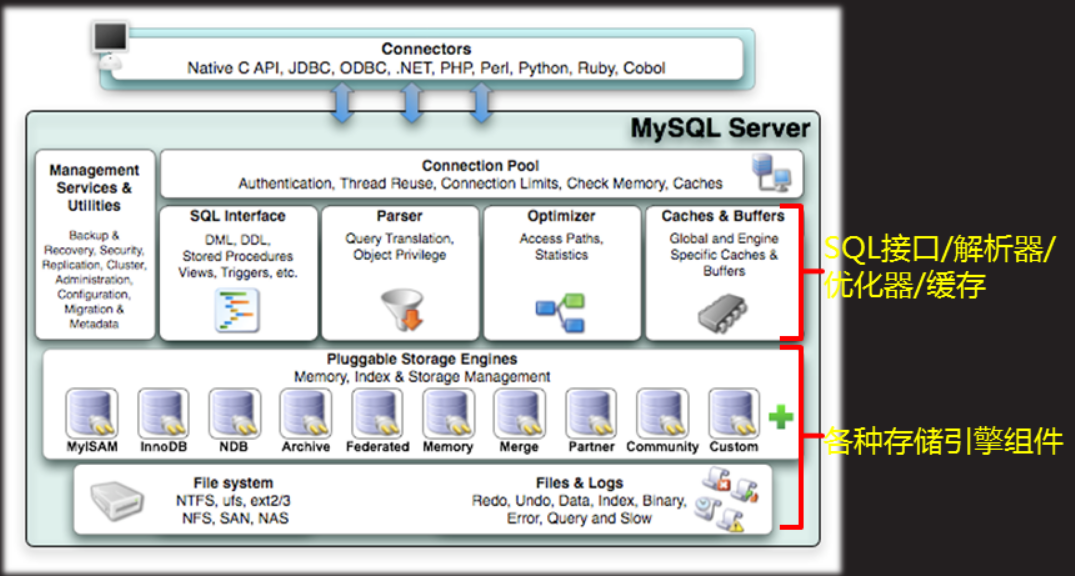
mysql> alter user user() identified by '新密码';

# ./mysql -uroot -p新密码 -S sock文件

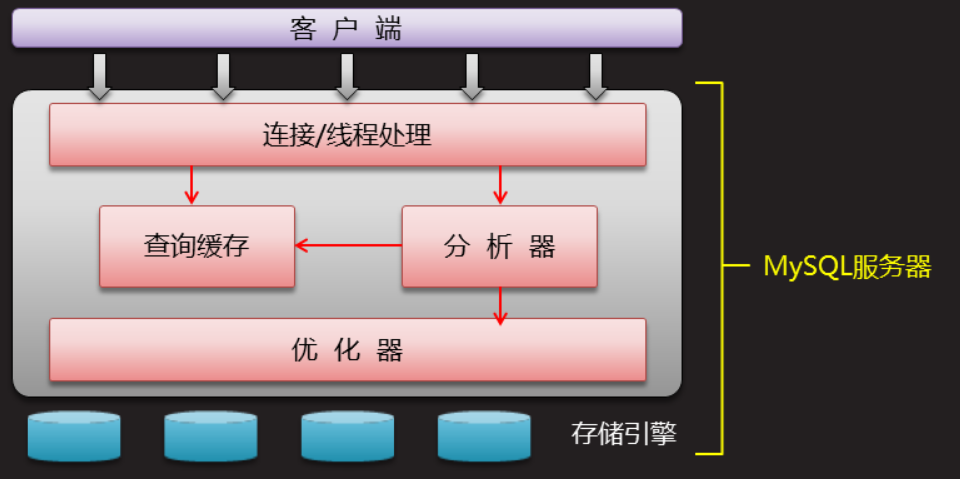
三、MySQL性能调优

1.MySQL工作原理

1）MySQL体系结构



2）MySQL执行流程



2.MySQL性能调优

·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）

- 服务的程序的运行参数调整

- 对SQL查询进行优化

·并发及连接控制

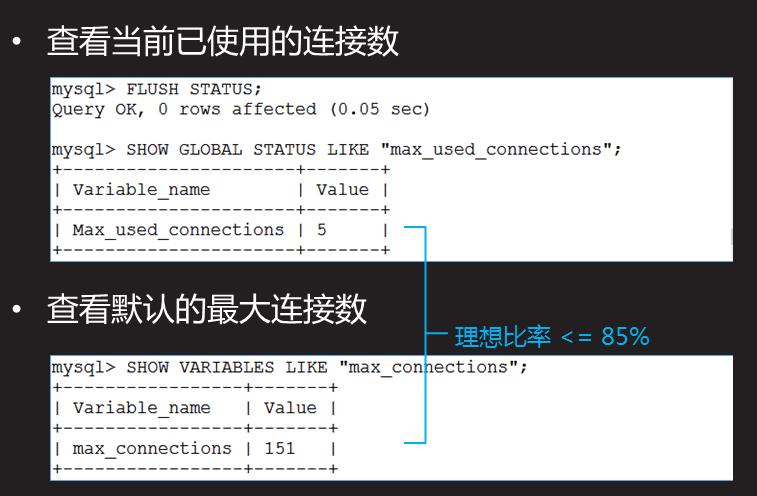
- 连接数、连接超时

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



·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

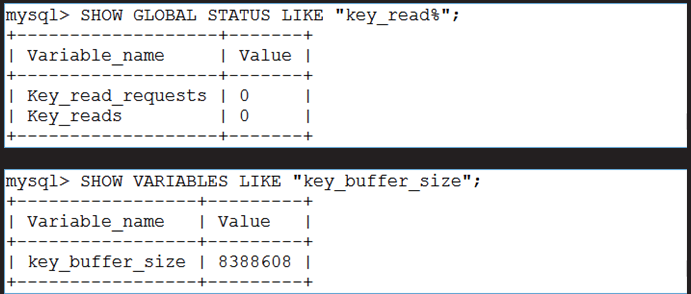
thread\_cache\_size 允许保存在缓存中被重用的线程数量

table\_open\_cache 为所有线程缓存的打开的表的数量

·key\_buffer\_size=8M

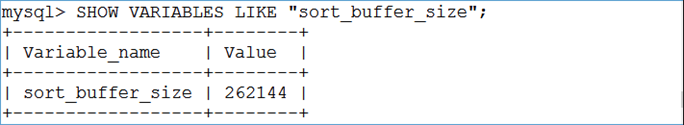
- 当Key\_reads / Key\_read\_requests 较低时

- 可适当加大此缓存值



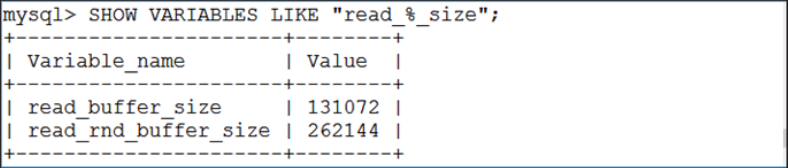
·sort\_buffer\_size=256K

- 增大此值可提高ORDER和GROUP的速度

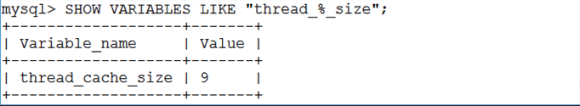


·查看表记录读取缓存

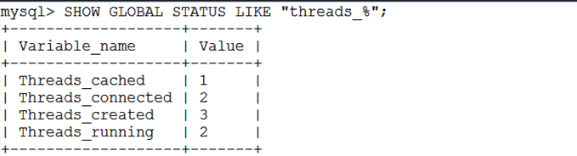
- 此缓存值影响SQL查询的响应速度

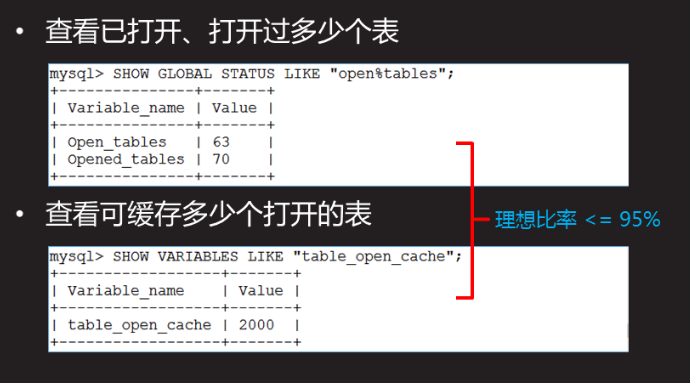


·查看可重用线程数



·查看当前的线程重用状态

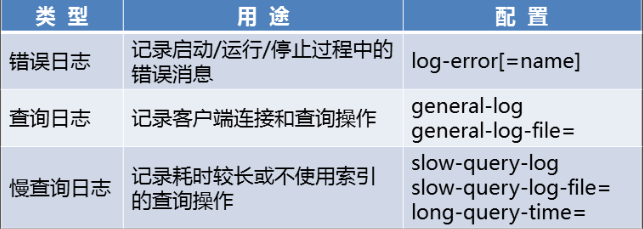




3.SQL查询优化

1）MySQL日志类型

·常用日志种类及选项



2）优化SQL查询

·记录慢查询



·调整服务配置

# vim /etc/my.cnf

[mysqld]

.. ..

slow\_query\_log=1

slow\_query\_log\_file=mysql-slow.log

long\_query\_time=5

log\_queries\_not\_using\_indexes=1

# service mysql restart

·调整服务配置

·查看慢查询日志

- 使用musqldumpslow工具

# mysqldumpslow /var/lib/mysql/mysql-slow.log

Reading mysql slow query log from /var/lib/mysql/mysql-slow.log

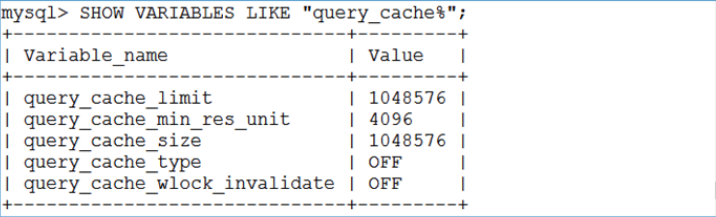
Count:1 Time=0.00s(0s) Lock=0.00s(0s) Rows=0.0(0),

0users@0hosts

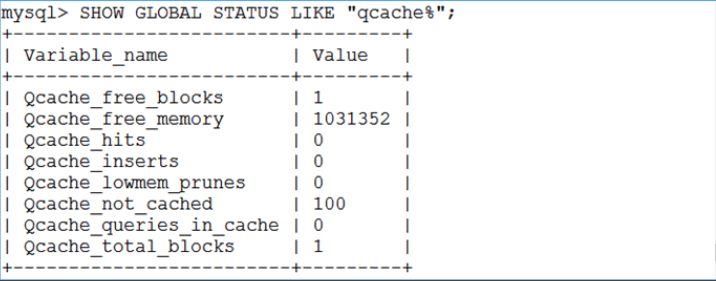
.. ..

3.关于查询的缓存

·查看缓存的大小



·查看当前的查询缓存统计



========================================================================================================

一、MySQL性能优化

1.MySQL工作原理

1.1 MySQL服务的工作过程

管理工具：MySQL服务软件安装后提供的命令

连接池：检查本机是否有资源处理当前的连接请求（空闲的线程和内存）

sql接口：把sql命令传递给mysql服务的进程处理

分析器：检查执行的sql命令是否有语法错误

优化器：优化sql命令，以最节省系统资源的方式执行

查询缓存：存储查询过的查询结果，存储空间是从系统的物理内存中划分出来的

存储引擎：软件自带的功能程序，是用来处理表的处理器

文件系统：数据库服务器存储数据的磁盘

1.2 MySQL服务处理查询请求的过程

·查询缓存 -> 表 -> 回复

2.MySQL性能调优（软调优）

·MySQL优化：数据库服务器处理客户的连接请求慢，可能是由哪些原因导致。

- 网络带宽

- 硬件配置（服务器资源使用情况：CPU,内存,存储）

- 软件版本

- 参数设置

> show variables; #查看所有变量

> show variables like "%关键字%";

> show variables like "%time%";

命令行修改

set [global] 变量名=值;

永久修改

vim /etc/my.cnf

[mysqld]

变量名=值

常用的参数有哪些？

并发连接数

set global max\_connections = 500;

show variables like "max\_connections";

有过的最大连接数量/并发连接数=0.85比较合理

Max\_used\_connections

> show globle status like "%Max\_used\_connections%";

> flush status;

> show variables like "%thread%";

thread\_cache\_size = 9

可以重复使用的线程的数量

table\_open\_cache 2000

为所有线程缓存打开表的数量

key\_buffer\_size 索引缓存大小

index primary key foreign key

字段值排序 存储在a.MYI a.ibd /var/lib/mysql/

> select name from user.a where name="bob";

查看数据库服务查询缓存的设置

> show variables like "%cache%";

> show variables like "query\_cache";

query\_cache\_type = 0 | 1 | 2

select sql\_in\_cache 字段名列表 from 表

query\_cache\_wlock\_invalidate off;

当对myisam存储引擎的表做查询的时候，若此时有客户端对表执行写操作，MySQL服务不会从缓存里查找数据返回给客户端，而是等写操作完成后，重新从表里查找数据返回给客户端。

脏读保护，只对myisam存储引擎的表有效

幻读

查询缓存统计信息

Qcache\_hits 100

Qcache\_inserts 0

程序员编写的访问数据库服务数据的sql命令复杂，导致处理速度慢。

在数据库服务器上启用慢查询日志，记录超过指定时间显示查询结果的sql命令。

binlog日志 错误日志 查询日志 慢查询日志

查询日志：记录所有的sql命令

慢查询日志：只记录超过超时时间显示查询结果的sql命令

启用慢查询日志：

[mysqld]

slow-query-log

log\_queries\_not\_using\_indexes=1

mySQL配置详解

MySQL帮助手册

二、MySQL数据读写分离

·把客户端访问数据库服务时的查询请求和写数据的请求给不同的数据库服务器处理

1.

读 写

select insert/delete/update

slave master

部署MySQL数据读写分离架构

1）配置MySQL主从同步

·要求：把20配置为10的从库

2）使用maxscale工具

·装包

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

·改配置，并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

# vim /etc/maxscale.cnf

[maxscale]

threads=auto

[server1]

type=server

address=192.168.4.10

port=3306

protocol=MySQLBackend

[server2]

type=server

address=192.168.4.20

port=3306

protocol=MySQLBackend

[MySQL Monitor]

type=monitor

module=mysqlmon

servers=server1,server2

user=myuser

passwd=mypwd

monitor\_interval=10000

不定义只读服务器（注释处理）

[Read-Write Service]

typoe=service

router=readwritesplite

servers=server1,server2

user=max

passwd=123456

max\_slave\_connections=100%

[MaxAdmin Service]

type=service

router=cli

只读服务端口配置（注释掉）

指定读写分离服务监听端口

[Read-Write Listener]

type=listener

service=Read-Write Service

protocol=MySQLClient

port=4006

[MaxAdmin Listener]

type=listener

service=MaxAdmin Service

protocol=maxscaled

socket=default

port=4099

用户授权

scalemon

> grant replication slave,replication client on \*.\* to scalemon@'%' identified by "123456";

maxscale

> grant replication select client on mysql.\* to maxscale@'%' identified by "123456";

·起服务

# maxscale -f /etc/maxscale.cnf

·查看服务信息

# ps -C maxscale

# ss -antulp | grep maxscale

·测试配置

1.在主机100上连接管理服务查看监控信息

2.在客户端连接100主机，访问数据时能否实现数据读写分离