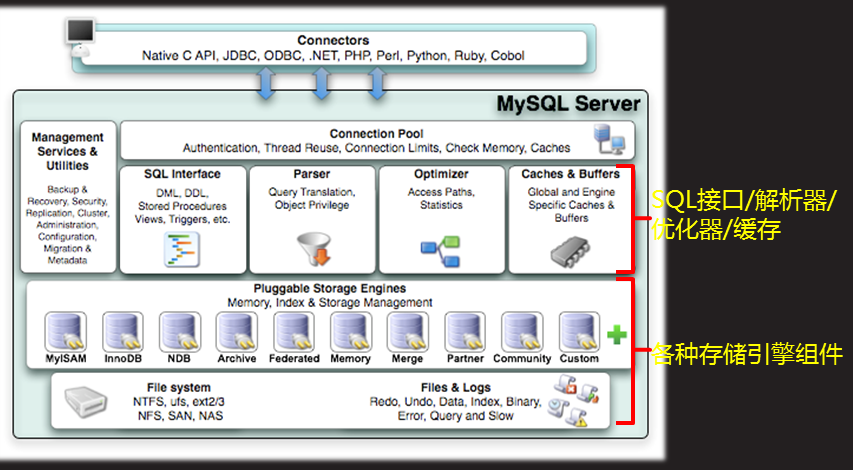
**mysql性能调优**

mysql工作原理

mysql体系结构



工作原理：

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 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：优化执行的sql命令，使其能以最节省系统资源的方式执行

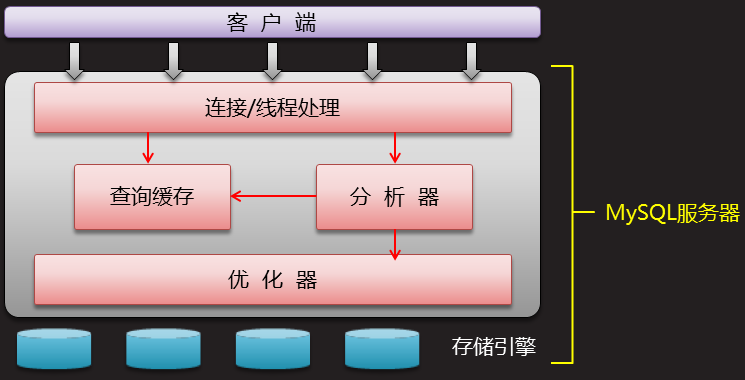
查询缓存：查询缓存的存储空间是从系统的物理内存里划分出来的，用来存储查询过的查询结果

存储引擎：软件自带的功能程序，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

mysql执行流程

mysql执行流程



客户端发起连接请求，连接池检查并建立连接，如果是查询请求，分析语法是否有错误，优化sql命令，去查询缓存里头找，如果有直接返回给客户端，如果没有，会去表里头找，然后存到查询缓存，再返还给客户端，下次直接去查询缓存里找

mysql性能调优

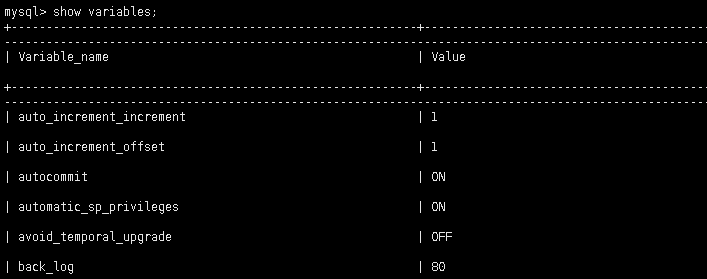
mysql性能调优

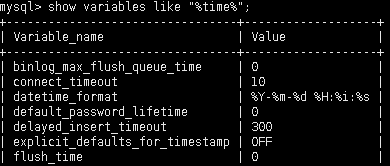
提高mysql系统的性能、响应速度

替换有问题的硬件（CPU/磁盘/内存等）

服务程序的运行参数调整

对sql查询进行优化

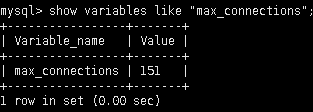
所有变量

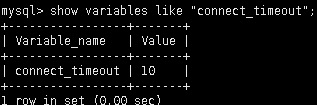


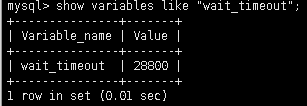
并发及连接控制

连接数、连接超时

|  |  |
| --- | --- |
| 选项 | 含义 |
| max\_connections | 允许的最大并发连接数，用有过的最大连接（show global status like “max\_used\_connections”;）除以并发连接数，大约为0.85为合适 |
| connect\_timeout | 等待连接超时，默认是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） |





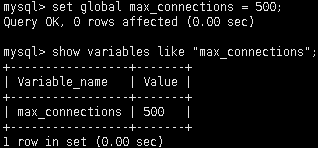


修改变量值：set [global] 变量名=值;

永久生效

[mysqld]

变量名=值



缓存参数控制

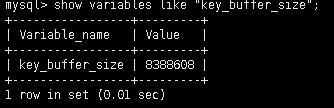
缓冲区、线程数量、开表数量

|  |  |
| --- | --- |
| 选项 | 含义 |
| key\_buffer\_size | 用于myisam引擎的关键索引缓存大小，分配多少字节用于内存存储索引 |
| sort\_buffer-size | 为每个要排序的线程分配此大小的缓存时间 |
| read\_buffer\_size | 为顺序读取表记录保留的缓存大小 |
| thread\_cache\_size | 可以重复使用的线程的数量 |
| table\_open\_cache | 为所有线程缓存的打开的表的数量 |

key\_buffer\_size=8M

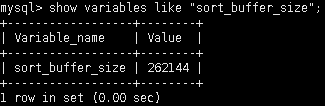
当key\_reads/key\_read\_requests较低时，可适当加大此缓存值





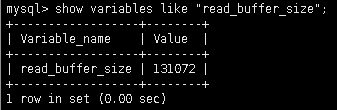
sort\_buffer\_size=256k

增大此值可提高ORDER和GROUP的速度

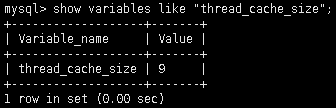


查看表记录读取缓存

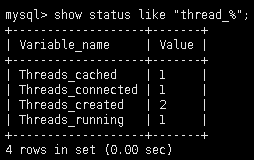
此缓存值影响sql查询的响应速度



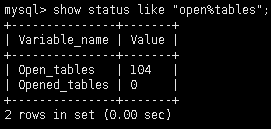
查看可重用线程数



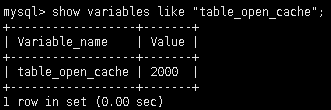
查看当前的线程重用状态



查看已打开、打开过多少个表



查看可缓存多少个打开的表



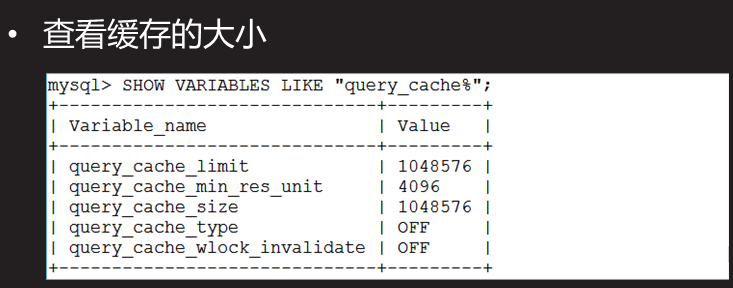
sql查询优化

mysql日志类型

常见日志种类及选项

|  |  |  |
| --- | --- | --- |
| 类型 | 用途 | 配置 |
| 错误日志 | 记录启动/运行/停止过程中的错误信息 | log-error[=name] |
| 查询日志（默认是主机名.log） | 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= |

设置查询的缓存



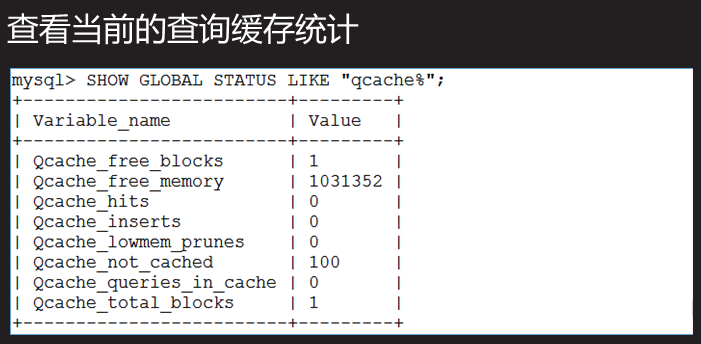
如果查询结果大于这个数，不允许往查询缓存中放，默认是1M

查询缓存的存储最小单位为4K

查询缓存空间的大小，默认为1M

是否开启查询缓存，默认为OFF，可以设置0、1、2，0代表关，1代表只要不超过限制，都可以保存，2代表查询的时候必须加关键字sql\_in\_cache否则不往查询缓存里保存

查询缓存写锁有效（对myisam存储引擎）当对myisam存储引擎的表，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



qcache\_hits：只要客户端的查询结果是在查询缓存中找到的，就会记录一遍，值自动加1

qcache\_inserts：只要接受到一次查询请求，值就+1

qcache\_lowmem\_prunes：查询缓存如果满了，会自动清理查询缓存，清理一次值+1

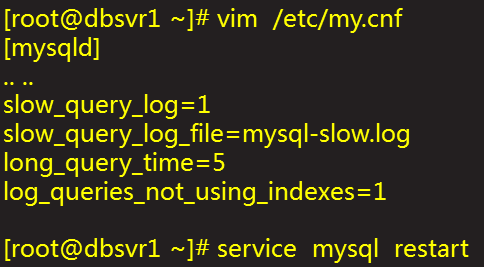
qcache\_not\_cached：不往查询缓存里放的次数，越大意味着limit值设置小了

优化sql查询（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）

记录慢查询

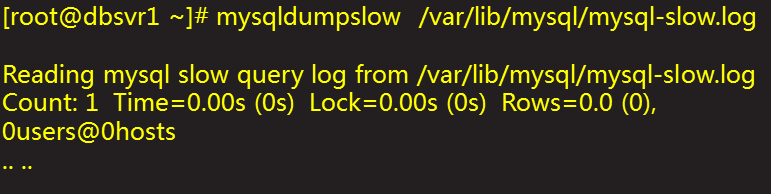
|  |  |
| --- | --- |
| 选项 | 含义 |
| slow-query-log | 启用慢查询 |
| slow-quiery-log-file | 指定慢查询日志文件 |
| long-query-time | 超过时间（默认10秒） |
| log-queries-not-using-indexes | 记录未使用索引的查询 |

调整服务器配置

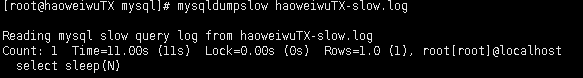




查看慢查询日志（使用mysqldumpslow工具）







mysql性能调优

|  |  |
| --- | --- |
| 手段 | 具体操作 |
| 升级硬件 | CPU、内存、硬盘 |
| 加大网络带宽 | 付费加大带宽 |
| 调整mysql服务运行参数 | 并发连接数、连接超时时间、重复使用的线程数..... |
| 调整与查询相关的参数 | 查询缓存、索引查询..... |
| 启用慢查询日志 | slow-query-log |
| 网站架构不合理 | 调整网络架构 |

**mysql中间件：mysql-proxy mycat maxscale**

**mysql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**

读写分离概述

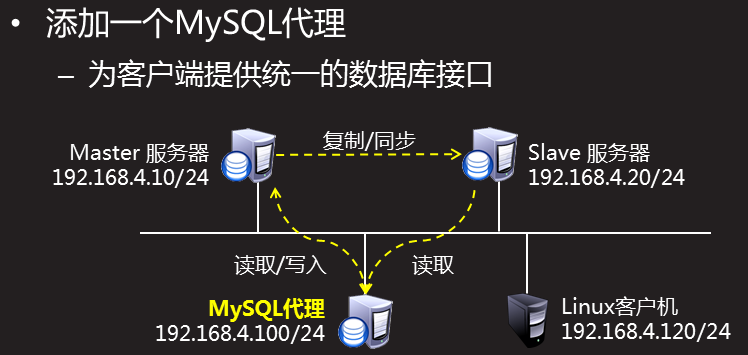
mysql读写分离

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，还是在服务器端区分？

案例拓扑



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的一致性

有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

构建思路

1. 已经建好mysql主从复制

基于上一个实验结果

其中slave为只读

1. 添加一台mysql代理服务器

部署/启用maxscale

1. 客户端通过代理主机访问Mysql数据库

访问代理服务器

部署maxscale服务（只有在写的时候才切换到主库，查在从库）

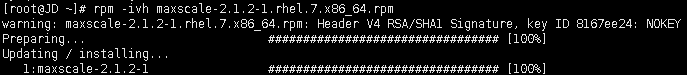
maxscale代理工具

由mysql的兄弟公司mariadb开发

下载地址

<https://downloads.mariadb.com/files/MaxScale>

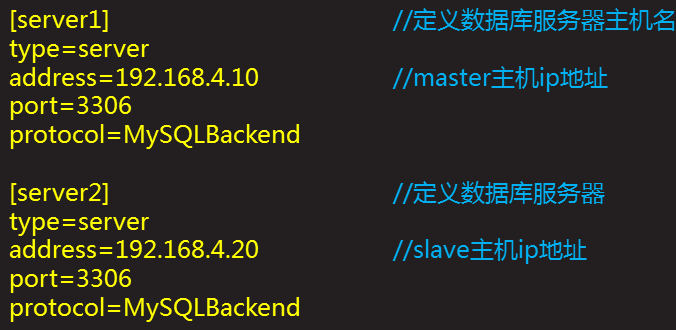
rpm -ivh maxscale.rpm







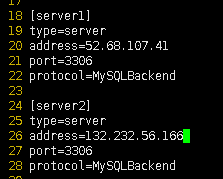
修改配置/etc/maxscale.cnf



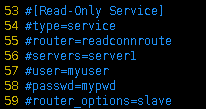
修改配置文件





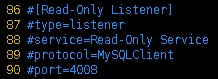
两个数据库的ip地址

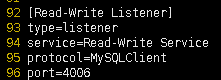
servers是监视的服务器，中间用逗号分隔，用户密码要有相应的权限（执行查看服务器运行状况的权限和查看主从命令的权限），监视频率，默认是毫秒，每隔多少秒查看一下状态

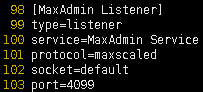


用户名和密码用来验证接受到客户端的访问请求时，验证账户在数据库服务器上是否存在

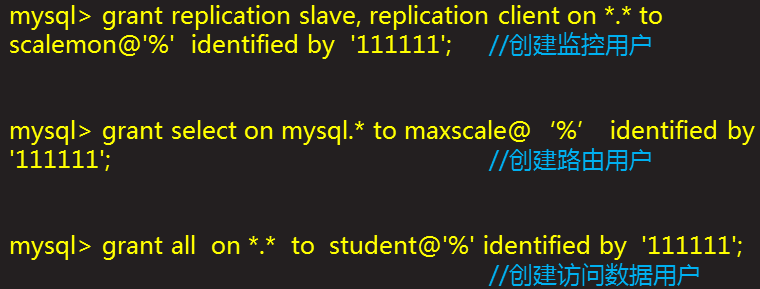


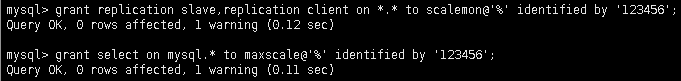




默认是4008，可以指定

在主、从数据库服务器创建授权用户





启动服务

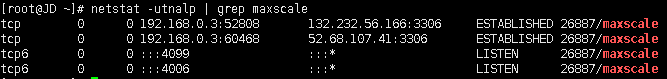
主要操作：启动服务、查看端口、停止服务

maxscale --config=/etc/maxscale.cnf

netstat -utnalp | grep maxscale

kill -9 端口号

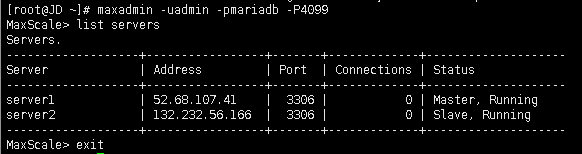




测试配置

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

