

数据库管理

NSD DATABASE

DAY06

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	MySQL 主从同步
	10:30 ~ 11:20	MySQL 读写分离
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	MySQL 性能调优
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



MySQL 主从同步



常用选项

常用的其他选项

- 适用于Master服务器

选 项	用 途
binlog_do_db=name	设置Master对哪些库记日志
binlog_ignore_db=name	设置Master对哪些库不记日志

常用的其他选项（续1）

- 适用于Slave服务器

选 项	用 途
log_slave_updates	记录从库更新，允许链式复制（A-B-C）
relay_log=dbsvr2-relay-bin	指定中继日志文件名
replicate_do_db=mysql	仅复制指定库，其他库将被忽略，此选项可设置多条（省略时复制所有库）
replicate_ignore_db=test	不复制哪些库，其他库将被忽略，ignore-db与do-db只需选用其中一种

知识讲解



主从复制的结构

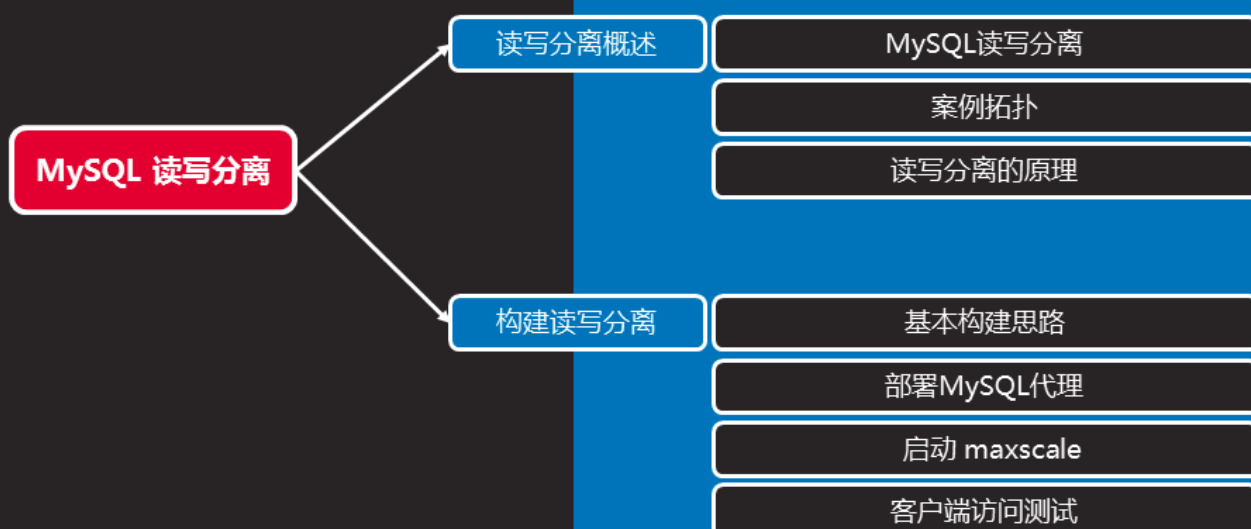
主从复制的结构

知识讲解

- 基本应用
 - 单向复制：主 --> 从
- 扩展应用
 - 链式复制：主 --> 从 --> 从
 - 双向复制：主 <--> 从
 - 放射式复制：从 <-- 主 --> 从
 - ∴
 - 从



MySQL 读写分离



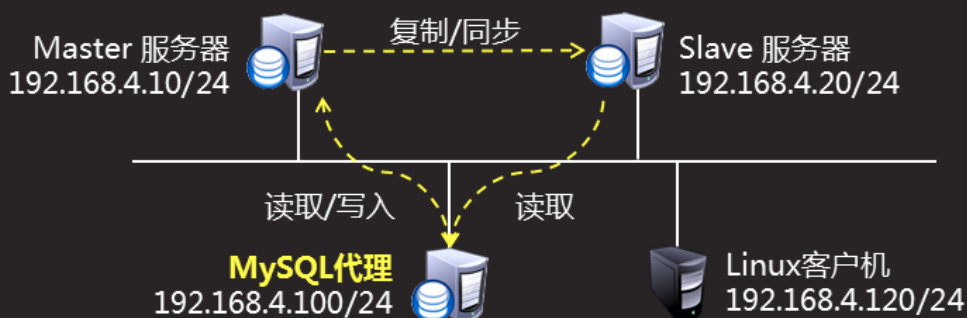
读写分离概述

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

- 添加一个MySQL代理
 - 为客户端提供统一的数据库接口



知识讲解



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

知识讲解



构建读写分离

基本构建思路

知识讲解

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



部署MySQL代理

知识讲解

- 安装 maxscale
 - MaxScale 是 Mysql 的兄弟公司 MariaDB 开发的
 - 下载地址
<https://downloads.mariadb.com/files/MaxScale>
 - 主配置文件 /etc/maxscale.cnf

```
[root@pxysvr dbproxy]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86_64.rpm
```



部署MySQL代理（续1）

知识讲解

- 修改配置文文件

```
[server1] //定义数据库服务器
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
```



部署MySQL代理（续2）

- 修改配置文件

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2 //定义的主、从数据库服务器列表
user=scalemon //用户名
passwd=111111 //密码
monitor_interval=10000
```

```
[Read-Write Service]
type=service
router=readwritesplit
servers=server1,server2 //定义的主、从数据库服务器列表
user=maxscale //用户名
passwd=111111 //密码
max_slave_connections=100%
```

知识讲解



部署MySQL代理（续3）

- 在主、从数据库服务器创建授权用户

```
mysql> grant replication slave, replication client on *.* to
scalemon@'%' identified by "111111" ; //创建监控用户
```

```
mysql> grant select on mysql.* to maxscale@'%' identified by
"111111" ; //创建路由用户
```

```
mysql> grant all on *.* to student@'%' identified by
"111111" ;
//创建访问用户为
```

知识讲解



启动 maxscale

- 主要命令：
 - 启动服务
 - 查看端口
 - 停止服务

```
[root@bogon ~]# maxscale --config=/etc/maxscale.cnf
[root@bogon ~]# netstat -utnlp | grep maxscale
tcp      0      0 192.168.1.110:58960  192.168.1.101:3306
ESTABLISHED 19081/maxscale
tcp      0      0 192.168.1.110:43508  192.168.1.111:3306
ESTABLISHED 19081/maxscale
tcp6     0      0 :::4006              :::*                  LISTEN
19081/maxscale
[root@bogon ~]# kill -9 19081
```

知识讲解



客户端访问测试

- 登录MySQL代理
 - mysql -h 代理的IP地址 -P 端口 -u 用户名 -p 密码
- 测试SQL查询、更新操作
 - 可成功查询表记录
 - 可成功写入数据

知识讲解



客户端访问测试(续1)

- 登录MySQL代理

```
[[root@bogon ~]# mysql -h192.168.4.100 -P4006 -ustudent -p111111
MySQL [(none)]> select @@hostname; //查看当前访问的主机名
+-----+
| @@hostname |
+-----+
| slave111   |
+-----+
```

知识讲解



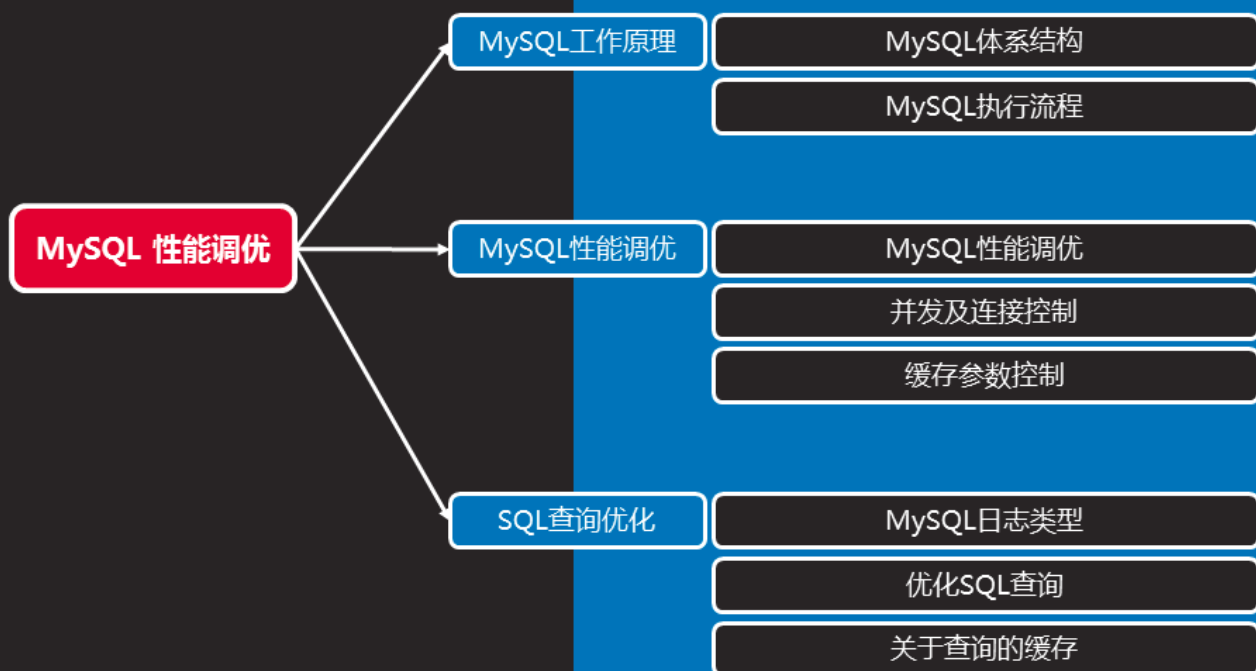
案例1：实现MySQL读写分离

1. 参考AB复制试验搭建好主从复制
2. 添加一台新的虚拟机，安装maxscale
3. 配置/启动maxscale服务
4. 测试代理读写及分离效果

课堂练习



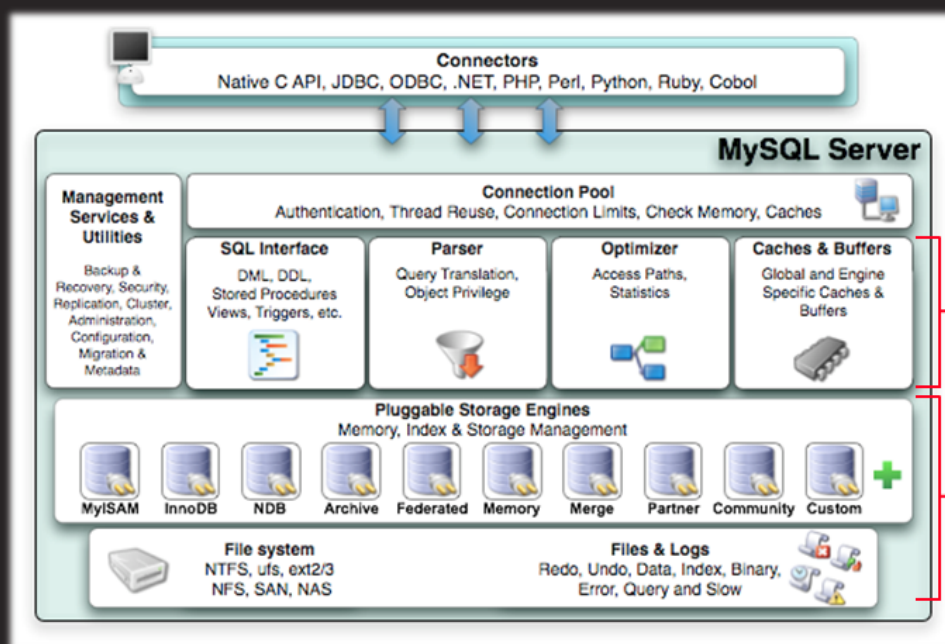
MySQL 性能调优



MySQL工作原理

MySQL体系结构

知识讲解



SQL接口/解析器/
优化器/缓存

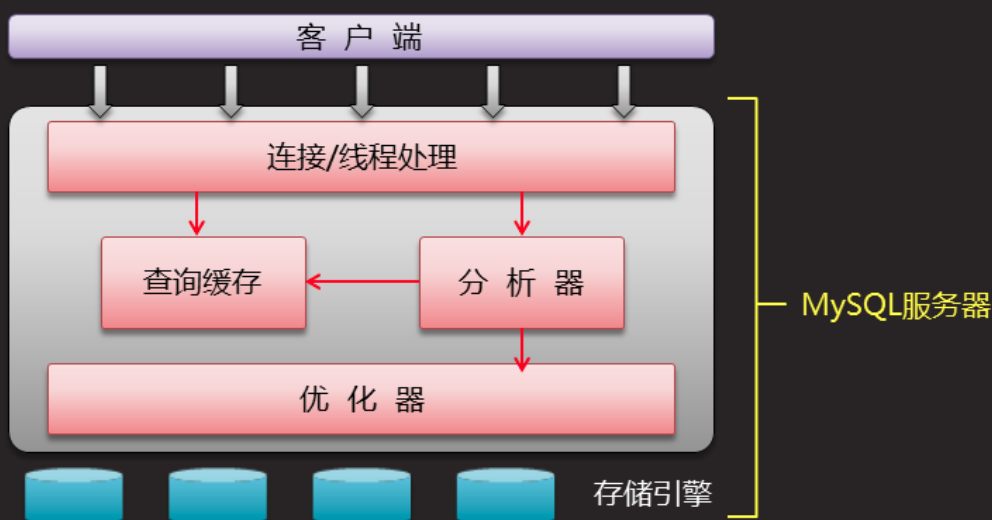
各种存储引擎组件



MySQL执行流程

- MySQL执行流程

知识讲解



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

并发及连接控制

知识讲解

- 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	等待建立连接的超时秒数，默认10秒，只在登录时有效
wait_timeout	等待关闭连接的不活动超时秒数，默认28800秒（8小时）



并发及连接控制（续1）

知识讲解

- 查看当前已使用的连接数

```
mysql> FLUSH STATUS;
Query OK, 0 rows affected (0.05 sec)

mysql> SHOW GLOBAL STATUS LIKE "max_used_connections";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Max_used_connections | 5 |
+-----+-----+
```

- 查看默认的最大连接数

```
mysql> SHOW VARIABLES LIKE "max_connections";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
+-----+-----+
```

理想比率 <= 85%



缓存参数控制

- 缓冲区、线程数量、开表数量

知识讲解

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
read_rnd_buffer_size	为按排序结果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



缓存参数控制（续1）

- key_buffer_size=8M
 - 当 Key_reads / Key_read_requests 较低时
 - 可适当加大此缓存值

知识讲解

```
mysql> SHOW GLOBAL STATUS LIKE "key_read%";
```

Variable_name	Value
Key_read_requests	0
Key_reads	0

```
mysql> SHOW VARIABLES LIKE "key_buffer_size";
```

Variable_name	Value
key_buffer_size	8388608



缓存参数控制（续2）

- sort_buffer_size=256K
 - 增大此值可提高ORDER和GROUP的速度

知识讲解

```
mysql> SHOW VARIABLES LIKE "sort_buffer_size";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| sort_buffer_size | 262144 |
+-----+-----+
```



缓存参数控制（续3）

- 查看表记录读取缓存
 - 此缓存值影响SQL查询的响应速度

知识讲解

```
mysql> SHOW VARIABLES LIKE "read_%_size";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| read_buffer_size | 131072 |
| read_rnd_buffer_size | 262144 |
+-----+-----+
```



缓存参数控制（续4）

知识讲解

- 查看可重用线程数

```
mysql> SHOW VARIABLES LIKE "thread_%_size";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| thread_cache_size | 9     |
+-----+-----+
```

- 查看当前的线程重用状态

```
mysql> SHOW GLOBAL STATUS LIKE "threads_%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 1     |
| Threads_connected | 2     |
| Threads_created | 3     |
| Threads_running | 2     |
+-----+-----+
```



缓存参数控制（续5）

知识讲解

- 查看已打开、打开过多少个表

```
mysql> SHOW GLOBAL STATUS LIKE "open%tables";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Open_tables   | 63    |
| Opened_tables | 70    |
+-----+-----+
```

- 查看可缓存多少个打开的表

```
mysql> SHOW VARIABLES LIKE "table_open_cache";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| table_open_cache | 2000  |
+-----+-----+
```

理想比率 <= 95%



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=
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=

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	超过指定秒数（默认10秒）的查询才被记录
log-queries-not-using-indexes	记录未使用索引的查询

知识讲解



优化SQL查询（续1）

- 调整服务配置

```
[root@dbsvr1 ~]# 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

[root@dbsvr1 ~]# service mysql restart
```

知识讲解



优化SQL查询（续2）

- 查看慢查询日志
 - 使用 mysqldumpslow 工具

```
[root@dbsvr1 ~]# 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
```

```
..
```

```
[root@dbsvr1 ~]#
```

知识讲解



关于查询的缓存

- 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 |  
+-----+-----+
```

知识讲解



关于查询的缓存（续1）

- 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	1031352
Qcache_hits	0
Qcache_inserts	0
Qcache_lowmem_prunes	0
Qcache_not_cached	100
Qcache_queries_in_cache	0
Qcache_total_blocks	1

知识讲解



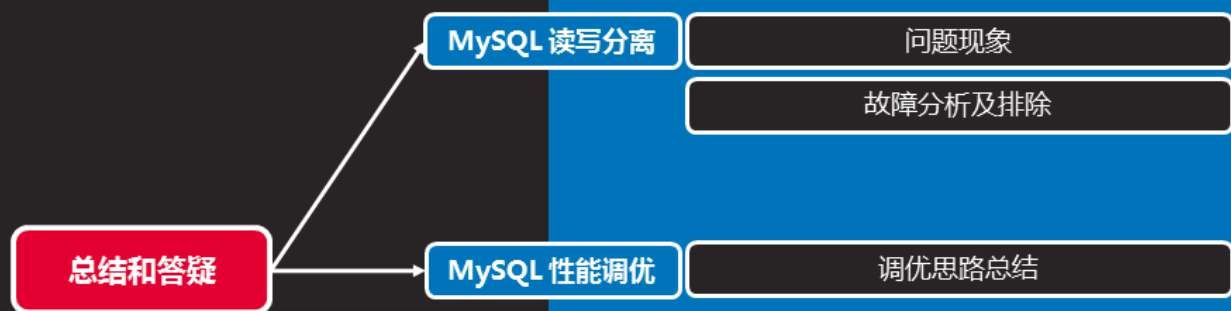
案例2：MySQL性能优化

- 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

课堂练习



总结和答疑



MySQL读写分离

问题现象

知识讲解

- 客户端连接mysql代理服务失败
 - 报错： `ERROR 2003 (HY000): Can't connect to MySQL server`

```
[root@room9pc00 ~]# mysql -h172.40.50.132 -ujerry -p123
Warning: Using a password on the command line interface can
be insecure.
ERROR 2003 (HY000): Can't connect to MySQL server on
'172.40.50.132' (111)
[root@room9pc00 ~]#
```



故障分析及排除

知识讲解

- 原因分析
 - 连接使用的用户名或密码错误
 - mysql-proxy服务没有启动
- 解决办法
 - 查看授权用户是否存在
 - 查看mysql-proxy是否运行

```
mysql> select user,host from mysql.user; //查看授权用户是否存在
ps aux | grep "mysql-proxy" //查看mysql-proxy是否运行
```



MySQL性能调优

调优思路总结

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、索引缓存.....
启用慢查询日志	slow-query-log
网络架构不合理	调整网络架构



