MySQL学习笔记（Day005：slow_log/generic_log/audit/存储引擎一）

MySQL学习

一. 慢查询日志进阶

1. 相关参数：

- **slow_query_log**
  - 是否开启慢查询日志
- **slow_query_log_file**
  - 慢查询日志文件名, 在 `my.cnf` 我们已经定义为slow.log，默认是 *机器名-slow.log*
- **long_query_time**
  - 制定慢查询阈值, 单位是秒，且当版本 `>=5.5.X` ，支持毫秒。例如 `0.5` 即为 `500ms`
  - `大于` 该值，不包括值本身。例如该值为2，则执行时间正好 `等于` 2的SQL语句 `不会记录`
- **log_queries_not_using_indexes**
  - 将没有使用索引的SQL记录到慢查询日志
    - 如果一开始因为数据少，查表快，耗时的SQL语句没被记录，当数据量大时，该SQL可能会执行很长时间
    - 需要测试阶段就要发现问题，减小上线后出现问题的概率
- **log_throttle_queries_not_using_indexes**
  - 限制每分钟内，在慢查询日志中，去记录没有使用索引的SQL语句的次数；版本需要 `>=5.6.X`
    - 因为没有使用索引的SQL可能会短时间重复执行，为了避免日志快速增大，限制每分钟的记录次数
- **min_examined_row_limit**
  - 扫描记录少于改值的SQL不记录到慢查询日志
    - 结合去记录没有使用索引的SQL语句的例子，有可能存在某一个表，数据量维持在百行左右，且没有建立索引。这种表即使不建立索引，查询也很快，扫描记录很小，如果确定有这种表，则可以通过此参数设置，将这个SQL不记录到慢查询日志。
- **log_slow_admin_statements**
  - 记录超时的管理操作SQL到慢查询日志，比如ALTER/ANALYZE TABLE
- **log_output**
  - 慢查询日志的格式，[FILE | TABLE | NONE]，默认是FILE；版本 `>=5.5`
  - 如果设置为TABLE，则记录的到 `mysql.slow_log`
- **log_slow_slave_statements**
  - 在从服务器上开启慢查询日志
- **log_timestamps**
  - 写入时区信息。可根据需求记录UTC时间或者服务器本地系统时间

2. 慢查询日志实践

- 设置慢查询记录的相关参数

```
--
-- 终端A
--
-- 注意做实验以前，先把my.cnf中的 slow_query_log = 0，同时将min_examined_row_limit = 100 进行注释
--
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.9-log |
+-----------+
1 row in set (0.01 sec)

mysql> show variables like "slow_query_log";  -- 为了测试，特地在my.cnf中关闭了该选项
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | OFF   |
+----------------+-------+
1 row in set (0.00 sec)

mysql> set global slow_query_log = 1;          -- slow_query_log可以在线打开
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like "slow_query_log";  -- 已经打开
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | ON    |
+----------------+-------+
1 row in set (0.00 sec)

mysql> show variables like "long_query_time";
+-----------------+----------+
| Variable_name   | Value    |
+-----------------+----------+
| long_query_time | 2.000000 |   -- my.cnf 中该值设置为2秒
+-----------------+----------+
1 row in set (0.00 sec)

mysql> show variables like "min_ex%";  -- my.cnf 中已经关闭注释，所以这里为0
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| min_examined_row_limit | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)
```

- 查看慢查询日志

```
#
#终端B
#
[root@localhost mysql_data]# tail -f slow.log
/usr/local/mysql/bin/mysqld, Version: 5.7.9-log (MySQL Community Server (GPL)). started with:
Tcp port: 3306  Unix socket: (null)
Time                 Id Command    Argument  #测试没有任何慢查询日志信息
```

- 进行模拟耗时操作

```
--
-- 终端A
--
mysql> select sleep(4);
+----------+
| sleep(4) |
+----------+
|        0 |
+----------+
1 row in set (4.00 sec)
```

- 最终产生慢查询日志

```
#
#终端B
#
[root@localhost mysql_data]# tail -f slow.log
/usr/local/mysql/bin/mysqld, Version: 5.7.9-log (MySQL Community Server (GPL)). started with:
Tcp port: 3306  Unix socket: (null)
Time                 Id Command    Argument  #测试没有任何慢查询日志信息
# Time: 2015-11-21T07:18:10.741663+08:00
# User@Host: root[root] @ localhost []  Id:     2
# Query_time: 4.000333  Lock_time: 0.000000 Rows_sent: 1  Rows_examined: 0
                                                          #这个就是min_examined_row_limit
                                                          #设置的意义。如my.cnf中设置该值为100
                                                          #则这条语句因为Rows_examined < 100,而不会被记录
SET timestamp=1448061490;
select sleep(4);
```

> **注意**
> 如果在终端A中 `set global min_examined_row_limit = 100;`，然后执行 `select sleep(5);` ，会发现该记录仍然被记录到慢查询日志中。原因是因为 `set global min_examined_row_limit` 设置的是全局变量，此次会话不生效。
>
> **但是我们上面 `set global slow_query_log = 1;` 却是在线生效的，这点有所不通**

- mysqldumpslow

```
[root@localhost mysql_data]# mysqldumpslow  slow.log

Reading mysql slow query log from slow.log
Count: 2  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=0.0 (0), 0users@0hosts
  Time: N-N-21T07:N:N.N+N:N
  # User@Host: root[root] @ localhost []  Id:     N
  # Query_time: N.N  Lock_time: N.N Rows_sent: N  Rows_examined: N
  SET timestamp=N;
  select sleep(N)

Count: 1  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=0.0 (0), 0users@0hosts
  # Time: N-N-21T07:N:N.N+N:N
  # User@Host: root[root] @ localhost []  Id:     N
  # Query_time: N.N  Lock_time: N.N Rows_sent: N  Rows_examined: N
  SET timestamp=N;
  select sleep(N)

##########################################################################

[root@localhost mysql_data]# mysqldumpslow  --help
Usage: mysqldumpslow [ OPTS... ] [ LOGS... ]

Parse and summarize the MySQL slow query log. Options are

  --verbose    verbose
  --debug      debug
  --help       write this text to standard output

  -v           verbose
  -d           debug
  -s ORDER     what to sort by (al, at, ar, c, l, r, t), 'at' is default #根据以下某个信息来排序
                al: average lock time
                ar: average rows sent
                at: average query time
                 c: count
                 l: lock time
                 r: rows sent
                 t: query time
  -r           reverse the sort order (largest last instead of first)  # 逆序输出
  -t NUM       just show the top n queries      # TOP(n)参数
  -a           don't abstract all numbers to N and strings to 'S'
  -n NUM       abstract numbers with at least n digits within names
  -g PATTERN   grep: only consider stmts that include this string
  -h HOSTNAME  hostname of db server for *-slow.log filename (can be wildcard),
               default is '*', i.e. match all
  -i NAME      name of server instance (if using mysql.server startup script)
  -l           don't subtract lock time from total time
```

如果在线上操作，不需要 `mysqldumpslow` 去扫整个 `slow.log` ， 可以去 `tail -n 10000 slow.log > last_10000_slow.log` (*10000这个数字根据实际情况进行调整*),然后进行 `mysqldumpslow last_10000_slow.log`

- 慢查询日志存入表

```
--
-- 在my.cnf 中增加 log_output = TABLE，打开slow_query_log选项，然后重启数据库实例
--
mysql> show variables like "log_output%";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_output    | TABLE |
+---------------+-------+
1 row in set (0.00 sec)

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

mysql> select * from mysql.slow_log;
+----------------------------+----------------------------+----------------+----------------+-----------+---------------+----+----------------+-----------+-----------+-----------------+-----------+
| start_time                 | user_host                  | query_time     | lock_time      | rows_sent | rows_examined | db | last_insert_id | insert_id | server_id | sql_text        | thread_id |
+----------------------------+----------------------------+----------------+----------------+-----------+---------------+----+----------------+-----------+-----------+-----------------+-----------+
| 2015-11-20 19:50:28.574677 | root[root] @ localhost [] | 00:00:04.000306 | 00:00:00.000000 |         1 |             0 |    |              0 |         0 |        11 | select sleep(4) |         3 |
+----------------------------+----------------------------+----------------+----------------+-----------+---------------+----+----------------+-----------+-----------+-----------------+-----------+
1 row in set (0.00 sec)

mysql> show create table mysql.slow_log;
--
-- 表结构输出省略
-- 关键一句如下：
--
ENGINE=CSV DEFAULT CHARSET=utf8 COMMENT='Slow log'  -- ENGINE=CSV 这里使用的是CSV的引擎,性能较差

-- 建议将slow_log表的存储引擎改成MyISAM
mysql> alter table mysql.slow_log engine = myisam;
ERROR 1580 (HY000): You cannot 'ALTER' a log table if logging is enabled  '-- 提示我正在记录日志中，不能转换

mysql> set global slow_query_log = 0;    -- 先停止记录日志
Query OK, 0 rows affected (0.01 sec)

mysql> alter table mysql.slow_log engine = myisam;   -- 然后转换表的引擎
Query OK, 2 rows affected (5.05 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> set global slow_query_log = 1;     -- 再开启记录日志
Query OK, 0 rows affected (0.00 sec)

mysql> show create table mysql.slow_log;
--
-- 表结构输出省略
-- 关键一句如下：
--
ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Slow log'  -- ENGINE 变成了MyISAM
```

> 使用 `TABLE` 的优势在于方便查询，但是记住当在备份的时候，不要备份慢查询日志的表，避免备份过大。
> 使用 `FILE` 也可以，需要定时清除该文件，避免单文件过大。

二. 通用日志(generic_log)与审计

1. 通用日志作用

- 当需要查找某条特定SQL语句，且该SQL语句执行较快，无法记录到slow_log中时，可以开启通用日志 `generic_log` ,进行全面记录， 可用于审计 `Audit`
- 通用日志会记录所有操作，性能下降明显。所以如果需要审计，需要 `Audit Plugin`

2. 审计插件

- MariaDB Audit 插件
  - MySQL社区版本目前没有提供Audit的功能，企业版本提供了该功能。MariaDB 提供了开源的Audit插件，且MySQL也能使用。
- 插件下载
  - server_audit-1.2.0.tar.gz 上述链接如果失效，可以进入官方页面注册，然后下载
  - 官方注册下载插件

3. Audit Plugin安装

- **`MySQL5.7.9` 审计插件安装失败，提示如下：**

> `ERROR 1126 (HY000): Can't open shared library '/usr/lib64/mysql/plugin/server_audit.so' (errno: 13 /usr/lib64/mysql/plugin/server_audit.so: undefined symbol: _my_thread_var)`

- **MySQL5.6.27 审计插件安装成功，步骤如下：**

```
# 找到plugin位置
[root@localhost ~]> cat /etc/my.cnf  | grep plugin_dir
plugin_dir=/usr/local/mysql/lib/plugin

# 解压plugin
[root@localhost ~]> tar zxvf server_audit-1.2.0.tar.gz
server_audit-1.2.0/
server_audit-1.2.0/linux-32_debug/
server_audit-1.2.0/linux-32_debug/server_audit.so
server_audit-1.2.0/linux-32/
server_audit-1.2.0/linux-32/server_audit.so
server_audit-1.2.0/linux-64_debug/
server_audit-1.2.0/linux-64_debug/server_audit.so
server_audit-1.2.0/windows-32/
server_audit-1.2.0/windows-32/server_audit.dll
server_audit-1.2.0/windows-64_debug/
server_audit-1.2.0/windows-64_debug/server_audit.dll
server_audit-1.2.0/linux-64/
server_audit-1.2.0/linux-64/server_audit.so
server_audit-1.2.0/windows-64/
server_audit-1.2.0/windows-64/server_audit.dll
server_audit-1.2.0/windows-32_debug/
server_audit-1.2.0/windows-32_debug/server_audit.dll

# 移动插件到对应的插件目录
[root@localhost ~]> mv server_audit-1.2.0/linux-64/server_audit.so /usr/local/mysql/lib/plugin
[root@localhost ~]> cd /usr/local/mysql/lib/plugin
```

```
--
-- 相关安装步骤
--
mysql> select version();
+------------+
| version()  |
+------------+
| 5.6.27-log |
+------------+
1 row in set (0.00 sec)

mysql> INSTALL PLUGIN server_audit SONAME 'server_audit.so';  -- 安装插件，该步骤在5.7.9中失败
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like "%server_audit%";    -- 查看和server_audit相关的参数
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| server_audit_events           |                       |
| server_audit_excl_users       |                       |
| server_audit_file_path        | server_audit.log      |
| server_audit_file_rotate_now  | OFF                   |
| server_audit_file_rotate_size | 1000000               |
| server_audit_file_rotations   | 9                     |
| server_audit_incl_users       |                       |
| server_audit_logging          | OFF                   |
| server_audit_mode             | 1                     |
| server_audit_output_type      | file                  |
| server_audit_syslog_facility  | LOG_USER              |
| server_audit_syslog_ident     | mysql-server_auditing |
| server_audit_syslog_info      |                       |
| server_audit_syslog_priority  | LOG_INFO              |
+-------------------------------+-----------------------+
14 rows in set (0.00 sec)

mysql> set global server_audit_logging = 1;   -- 打开审计功能
Query OK, 0 rows affected (0.00 sec)

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

mysql> show status like '%audit%';
+-----------------------------+------------------+
| Variable_name               | Value            |
+-----------------------------+------------------+
| server_audit_active         | ON               |
| server_audit_current_log    | server_audit.log |
| server_audit_last_error     |                  |
| server_audit_writes_failed  | 0                |
+-----------------------------+------------------+
4 rows in set (0.00 sec)


#
#查看审计日志
#
[root@MyServer mysql_data]> tail -f server_audit.log
20151120 22:40:54,MyServer,root,localhost,2,9,QUERY,,'set global server_audit_logging = 1',0
20151120 22:41:16,MyServer,root,localhost,2,10,QUERY,,'show variables like "server_audit_logging"',0
20151120 22:41:53,MyServer,root,localhost,1,5,QUERY,,'show status like \'%audit%\'',0
```

> 以上仅为基本功能操作，详细的细粒度控制请参考官方文档

三. 存储引擎(一)

1.Mysql上支持的存储引擎

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)
```

2. 存储引擎的概念

> 用来处理数据库的相关CRUD操作

每个数据库都有存储引擎，只是MySQL比较强调存储引擎的概念。

3. MySQL存储引擎

- 官方存储引擎
  - MyISAM
  - `InnoDB` – 推荐；其他引擎已经体停止维护和开发
  - Memory
  - Federated
  - CSV
  - Archive
- 第三方存储引擎
  - TokuDB – 开源，适合插入密集型
  - InfoBright – 商业，开源版本有数据量限制。属于列存储，面向OLAP场景
  - Spider

> 第三方存储引擎在特定场合下比较适合，除此之外，都应该使用InnoDB

3. 存储引擎之MyISAM

- MySQL5.1版本之前的默认存储引擎
- 堆表数据结构
- 表锁设计
- 支持数据静态压缩
- 不支持事物
- 数据容易丢失
- 索引容易损坏
- 唯一优点
  - 数据文件可以直接拷贝到另一台服务器使用

> 现在MySQL中还有用MyISAM的表，主要是历史原因。数据库文件以 `MY` 开头的基本都是MyISAM的表