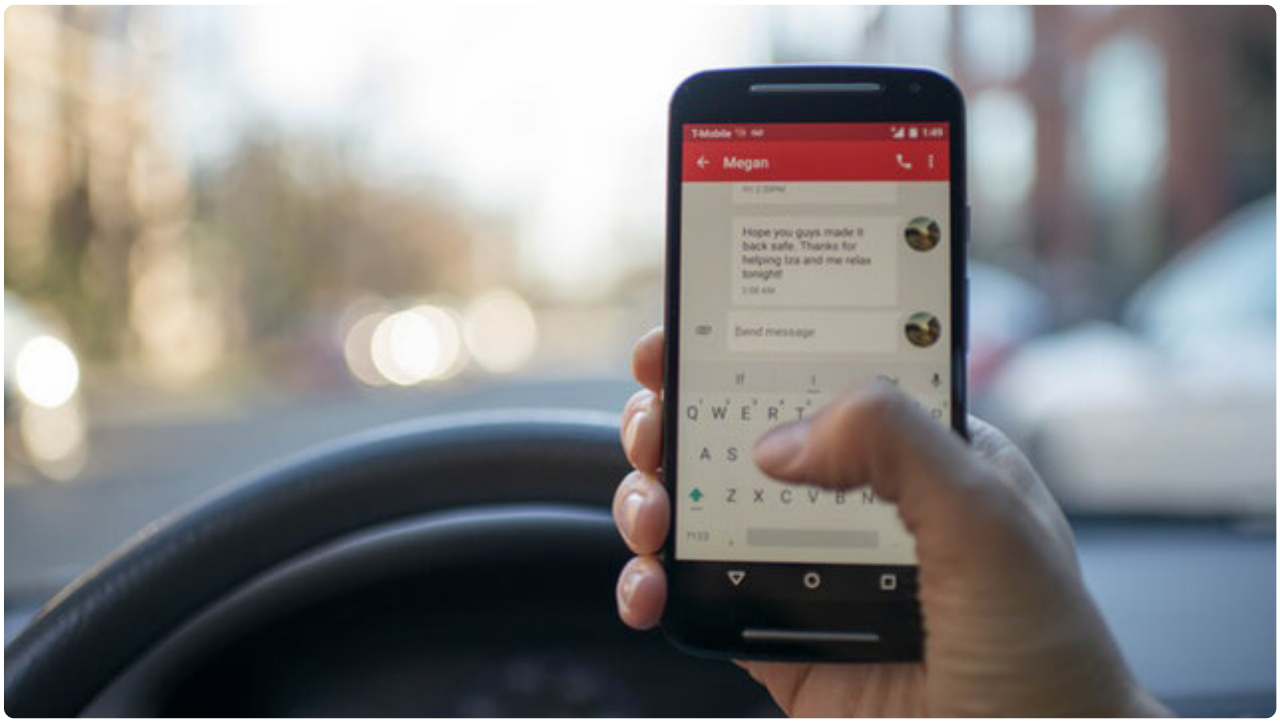
15 认识日志系统，掌握系统运行过程

更新时间：2020-04-03 14:33:06



> 知识犹如人体的血液一样宝贵。——高士其

不论你参与的是大项目还是小项目，一定会在项目的早期把日志系统添加进去，这是非常重要且必要的，因为你将来排查问题、优化系统、对账报表等等都依赖于你的日志。虽说日志是工程不可或缺的组成部分，但是，想要做的很好（易于理解、维护、分析等等）并不容易。这一节里，我们一起来学习 MySQL 的日志系统，看看它能对你产生怎样的启发。

1 日志系统初探

在具体的看 MySQL 日志系统之前，我们先来简单的了解下 MySQL 提供了哪些日志（这里，你可以去思考下为什么 MySQL 要提供这种日志），以及对日志系统做的整体优化。

1.1 日志类型

关于分类，从不同的角度看，可以有不同的分类。例如，从存储类型的角度来看，可以分为两类：逻辑日志（存储 SQL 修改语句）和物理日志（存储数据被修改的值）。但是，更加常见的是按照功能分成以下四类：

- 错误日志：记录 MySQL 服务实例启动、运行、停止等等的过程信息
- 查询日志：也叫做普通（general）查询日志，记录 MySQL 实例运行的所有 SQL 语句和命令
- 慢查询日志：记录执行时间较长（有参数可以指定）的 SQL 语句，或者没有使用索引的 SQL 语句
- 二进制日志：记录 MySQL 实例执行的所有更新语句，不包含 SELECT 和 SHOW 语句

那么，在接下来的内容中，我就会按照功能分类依次讲解这四类日志（可以想一想，为什么 MySQL 要把日志分成很多类）。

1.2 日志缓存

说到缓存（Cache）这个词，大家一定都不会陌生，为了提升系统的 IO、计算性能，我们通常会把将来可能会使用到的数据放到缓存系统中，例如：Redis、MemCache 等等。MySQL 的日志处理过程也同样使用了缓存机制。

MySQL 系统在运行过程中产生的日志起初会放在服务器的内存（日志数据结构）中，当内存中的日志超过了指定的阈值，便会将日志刷写到数据表（本质还是文件）或文件中。这种优化缓冲的做法非常常见，例如 HBase 的 HLog（HBase 的日志组件）也是同样的做法记录日志。

2 错误日志

之前已经谈到日志就是文件，那么，这些文件存储在哪里呢？实际上，这些文件存放在数据库的根目录下，且各种类型的日志都存放在这个目录下。不过，可能你安装完 MySQL 之后就忘了安装在哪里了（特别是 Mac 用户，通过 brew 安装）。好吧，那我们就带着这个问题去看一看错误日志。

相比于其他类型的日志，MySQL 的错误日志比较特殊，你不能将它关闭。它主要记录服务实例每次启动、停止的详细信息，以及运行过程中产生的警告和错误信息。MySQL 中与错误日志有关的参数是：log_error，我们一起来看一看它：

```
mysql> SHOW VARIABLES LIKE 'log_error';
+---------------+---------------------------------------+
| Variable_name | Value                                 |
+---------------+---------------------------------------+
| log_error     | /usr/local/mysql/data/mysqld.local.err |
+---------------+---------------------------------------+
```

其中，第二列 Value 打印的就是错误日志文件的完整路径。好的，这下你应该知道日志存储在哪里了。默认情况下，错误日志文件名会以 “主机名.err” 结尾。但是，需要知道，错误日志不会记录所有的错误信息，只有被 MySQL 声明为 “关键错误” 的事件发生时才会记录。

3 查询日志

同样，查询日志也是比较特殊的，相对于其他几类日志来说，查询日志是不建议打开的。这类日志通常用于项目的开发阶段，记录系统的完整行为，及时发现问题并解决问题。这也是一类比较简单的日志，一起来看看吧。

3.1 初识查询日志

查询日志记录了 MySQL 服务实例所有的操作，而不管这些操作是否执行成功。例如：日常的 CRUD、客户端与服务器连接和断开等等。你可以认为，对于查询日志来说，MySQL 发生了什么，就会记录什么。

正是由于查询日志的记录过程不会挑肥拣瘦，这也成了它最大的缺点。当数据库访问的特别频繁时，将会有大量的日志产生，由此将会大幅降低数据库的性能。所以，通常来说，查询日志都是关闭的。只有在调试或者特殊时期，需要追踪某些特殊的查询，可以临时打开查询日志。

3.2 与查询日志相关的参数

与查询日志相关的参数一共有三个，它们分别用于标识状态、日志路径以及输出类型。下面，我们依次来看一看这三个参数：

```
-- 第一个参数 general_log，标识查询日志是否处于开启状态，默认是 OFF，即关闭
mysql> SHOW VARIABLES LIKE 'general_log';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| general_log   | OFF   |
+---------------+-------+
```

开启查询日志有两种方式，一种是修改配置文件，但是需要重启服务；另一种是在运行时通过 SET 命令，但是只能对当前的会话生效。修改过程如下所示：

```
-- 修改配置文件：/etc/my.cnf
general_log_file       = /usr/local/mysql/data/general_log_mysql.log
general_log            = 1

-- 通过 SET 命令，使当前的会话生效
SET GLOBAL general_log = 'ON';
```

打开查询日志的开关之后，MySQL 服务实例将会自动创建查询日志文件，general_log_file 参数则用于指定文件的位置（同样可以自行修改），我们可以看一看：

```
-- 第二个参数 general_log_file
mysql> SHOW VARIABLES LIKE 'general_log_file';
+------------------+----------------------------------------------+
| Variable_name    | Value                                        |
+------------------+----------------------------------------------+
| general_log_file | /usr/local/mysql/data/B000000104678.log |
+------------------+----------------------------------------------+
```

最后一个参数是 log_output，它用于设置查询日志和慢查询日志（是的，这个参数控制了两类日志的行为）的输出类型。默认情况下，这个参数的值是 “FILE”，代表将日志写入文件。我们可以把它修改成 “TABLE”，这样，查询日志就会被写入 mysql 系统库的 general 表中（对于慢查询日志来说，会写入 slow_log 表中）。演示下这个过程吧：

```
-- 设置查询日志输出到表中
mysql> SET GLOBAL log_output = 'TABLE';

-- 一条任意的 SQL 查询
mysql> SELECT * FROM imooc_mysql.worker WHERE id = 1;
+----+------+------+--------+---------+
| id | type | name | salary | version |
+----+------+------+--------+---------+
|  1 | A    | tom  |   1800 |       0 |
+----+------+------+--------+---------+

-- 可以看到，mysql.general_log 表中记录了刚刚的查询语句
mysql> SELECT user_host, command_type, argument FROM mysql.general_log ORDER BY event_time DESC LIMIT 3;
+----------------------------------+--------------+-----------------------------------------------------+
| user_host                        | command_type | argument                                            |
+----------------------------------+--------------+-----------------------------------------------------+
| root[root] @ localhost [127.0.0.1] | Query        | SHOW COLUMNS FROM `general_log`                     |
| root[root] @ localhost [127.0.0.1] | Query        | USE `mysql`                                         |
| root[root] @ localhost []          | Query        | SELECT * FROM imooc_mysql.worker WHERE id = 1       |
+----------------------------------+--------------+-----------------------------------------------------+
```

将查询日志写到数据表中可以方便的通过 SQL 去查询 MySQL 的工作过程（还可以加上一些过滤条件），但是这会严重降低服务器的性能，只能将它应用于调试，线上服务要绝对禁止。

4 慢查询日志

这类日志有个很形象的名字，也说明了日志记录的规则：慢查询。这是一类非常重要的日志，不仅在工作中发挥着巨大的作用，而且也是面试的常客。关于慢查询日志，你需要知道：它的概念是什么 ？怎样定义“慢”？又该怎样分析这类日志？好的，带着这几个问题一起来学习下吧。

4.1 初识慢查询日志

慢查询日志的关键词是“慢查询”，也就是说当你的 SQL 查询足够慢时，MySQL 就会记录这条 SQL。虽然说不能一概而论多久才是慢，但是，一般情况下，我们会定义 “商业应用” 的阈值是 1s，而 “用户应用” 的阈值则是 200ms。

MySQL 的慢查询日志可以有效的对执行时间过长、没有使用索引的查询进行跟踪。这些查询也就是我们通常使用的 CRUD 操作。但是，需要特别注意，慢查询日志只会记录执行成功的语句，这与查询日志不同。

4.2 与慢查询日志相关的参数

与慢查询日志有关的参数一共有5个，我们先来看一看与状态有关的两个（我会带着操作过程去解释）：

```
-- slow_query_log：标记慢查询日志是否开启的参数，默认是 OFF，即不开启
mysql> SHOW VARIABLES LIKE 'slow_query_log';
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | OFF   |
+----------------+-------+

-- slow_query_log_file：标记存放慢查询日志文件的完整路径
mysql> SHOW VARIABLES LIKE 'slow_query_log_file';
+---------------------+-----------------------------------------------+
| Variable_name       | Value                                         |
+---------------------+-----------------------------------------------+
| slow_query_log_file | /usr/local/mysql/data/B000000104678-slow.log |
+---------------------+-----------------------------------------------+
```

慢查询日志一旦开启，MySQL 实例将会自动创建慢查询日志文件。与查询日志类似，开启慢查询日志同样有两种方法，如下所示：

```
-- 修改配置文件：/etc/my.cnf
slow-query-log        = 1
slow_query_log_file   = /usr/local/mysql/data/slow_log_mysql.log

-- 通过 SET 命令，使当前的会话生效
SET GLOBAL slow_query_log = 'ON';
```

其实，在打开慢查询日志开关之前，我们应该先设置慢查询的时间阈值。这个阈值默认是 10s，这显然不能接受。可以像下面这样去修改、查询这个阈值：

```
-- long_query_time：控制慢查询的时间阈值参数
mysql> SHOW VARIABLES LIKE 'long_query_time';
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+

-- 设置慢查询时间阈值为 1s，注意，需要重新打开会话才能看到修改后的值
mysql> SET GLOBAL long_query_time = 1.0;
```

慢查询日志还有一个非常强大的功能，它可以捕获没有使用索引的查询语句，不论这条查询速度有多快（这一点非常重要）。同样，MySQL 也用了一个参数来控制这一行为：

```
-- log_queries_not_using_indexes：标识是否记录未使用索引的查询，默认是关闭
mysql> SHOW VARIABLES LIKE 'log_queries_not_using_indexes';
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| log_queries_not_using_indexes | OFF   |
+-------------------------------+-------+
1 row in set (0.00 sec)

-- 通过 SET 命令，打开这一开关
mysql> SET GLOBAL log_queries_not_using_indexes = 'ON';
Query OK, 0 rows affected (0.00 sec)
```

最后一个与慢查询日志相关的参数是 log_output，这在讲解“查询日志”时已经做了说明，这里不再赘述（可以尝试修改这个参数，让慢查询日志记录到 mysql.slow_log 表中，并分析其中的各个列值）。

4.3 慢查询日志分析工具

MySQL 的慢查询日志是可读的，即可以直接阅读（使用文本编辑器即可打开，另外，如果想要构造慢查询，最简单的办法是：`SELECT SLEEP(3)`），但是，当慢查询日志文件太大，就很难找到自己想要的内容。为此，MySQL 提供了 mysqldumpslow 工具（默认安装），它的详细用法可以通过 `mysqldumpslow --help` 命令查看。下面，我来讲解几个常用的参数：

- -s：标识按照哪种方式排序
  - al：平均锁等待时间
  - at：平均查询时间
  - ar：平均返回数据行数
  - c：查询执行次数
  - l：锁等待时间
  - r：返回数据行数
  - t：查询时间
- -t N：标识返回前 N 条数据
- -g：grep，包含模糊匹配

最后，给出一些常用的查询方法，类似的情况，可以照猫画虎去完成（需要指定慢查询日志的正确路径，另外，可能需要 root 权限）。

```
# 查询访问次数最多的 10条 SQL 语句
mysqldumpslow -s c -t 10 /usr/local/mysql/data/B000000104678-slow.log

# 查询返回记录数最多的 10条 SQL 语句
mysqldumpslow -s r -t 10 /usr/local/mysql/data/B000000104678-slow.log

# 查询含有 like 的 SQL 语句
mysqldumpslow -g 'like' /usr/local/mysql/data/B000000104678-slow.log
```

5 二进制日志

二进制日志其实就是我们通常所说的 Binlog，它属于 MySQL 的逻辑日志，且使用二进制的格式保存在磁盘文件中，所以，也就不能直接查看。Binlog 的重要性不仅体现在工作中，也常常作为面试题出现，且会有一定的难度。所以，认真的学习、理解 Binlog 至关重要。

5.1 初识二进制日志

Binlog 用于记录数据库的所有变化，在 MySQL 中它的作用是主从同步。关于 Binlog 记录的信息，简单的说，所有涉及数据变动的操作，都会记录下来。Binlog 日志的记录格式一共有三种：ROW、STATMENT 以及 MIXED，下面，我来对它们进行说明：

ROW

- 说明：基于行的复制，不会记录每一条 SQL 语句的上下文信息，仅仅会保存哪条记录被修改
- 优点：记录了每一行数据的修改细节
- 缺点：日志量巨大，特别是批量修改的情况

STATMENT

- 说明：基于 SQL 语句的复制，记录每一条修改数据的 SQL，是 Binlog 的默认格式
- 优点：日志量小，节约磁盘 IO
- 缺点：在某些情况下会导致 Master-Slave 不一致，例如：执行 Sleep 函数

其中，MIXED 格式是 ROW 和 STATMENT 两种格式的混合，普通的复制使用的是 STATMENT 格式，对于 STATMENT 无法复制的操作则会使用 ROW 格式去记录。使用 MIXED 格式时，会由 MySQL 来决定使用哪种格式记录日志。

5.2 二进制日志的相关操作

其实在之前讲解“数据备份与恢复”时，就已经提到过 Binlog，但是，我并没有详细的对 Binlog 的相关操作做出说明，这里，我们就来看一看 Binlog 的相关操作。

对于 8.0 之前的版本来说，Binlog 是默认关闭的。标识 Binlog 开关的参数是 log_bin，且它是一个只读变量，也就是说我们不能通过 SET 命令的方式完成修改。我们一起来看看：

```
-- log_bin 的默认值是 OFF，标识关闭 Binlog
mysql> SHOW VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | OFF   |
+---------------+-------+

-- log_bin 是一个只读变量，不能通过 SET 命令修改
mysql> SET GLOBAL log_bin = 'ON';
ERROR 1238 (HY000): Variable 'log_bin' is a read only variable
```

所以，想要开启 Binlog，我们就必须要去修改 MySQL 的配置文件（my.cnf），然后重启 MySQL。配置文件中添加如下变量定义：

```
[mysqld]
log-bin = mysql-bin    # 开启 Binlog
binlog-format = ROW    # 定义为 ROW 格式（可以随意选择）
server_id = 1          # MySQL 实例 id
```

注意，修改完成之后，一定要重启你的 MySQL 服务器，而不是重新打开一个 MySQL 客户端会话。好吧，配置修改之后，自行去验证下是否打开了 Binlog 日志（查看下 log_bin 参数的值即可）。

想要查看当前系统的所有 Binlog 日志，可以查看 log_bin_index 参数，如下所示：

```
-- log_bin_index 参数标识的是 Binlog 索引文件的位置
mysql> SHOW VARIABLES LIKE 'log_bin_index';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| log_bin_index | /usr/local/mysql/data/mysql-bin.index |
+---------------+-----------------------------------------+
```

之后，可以使用 cat 命令查看下这个文件（可能需要 root/管理员 权限）：

```
□ ~ sudo cat /usr/local/mysql/data/mysql-bin.index
Password:
./mysql-bin.000001
./mysql-bin.000002
./mysql-bin.000003
./mysql-bin.000004
./mysql-bin.000005
./mysql-bin.000006
./mysql-bin.000007
```

这个文件中记录了 Binlog 的相对路径（相对于 mysql-bin.index 文件），所以，实际上 Binlog 日志也位于 /usr/local/mysql/data 中。另外，关于 Binlog 还有几个常用的操作命令需要掌握：

```
-- 查看所有的 Binlog 日志列表
SHOW MASTER LOGS

-- 查看 Master 的状态，即最后一个 Binlog 日志的编号名称，及操作事件的 POS（位置）值
SHOW MASTER STATUS

-- 刷新 Binlog，产生一个新编号的 Binlog 文件
FLUSH LOGS

-- 清空所有的 Binlog 日志（慎用）
RESET MASTER
```

5.3 二进制日志的清理

随着系统在不断运行，日志量也会逐步增加，所以，自然也就会有清理日志的工作。MySQL 提供了两种方式来清理 Binlog，一种是手动的，一种是自动的。

手动清理 Binlog 是通过 PURGE 命令，它会同时删除 Binlog 文件和 Binlog 索引文件记录。语法以及示例如下：

```
-- PURGE 命令语法
PURGE { BINARY | MASTER } LOGS { TO 'log_name' | BEFORE datetime_expr }

-- PURGE 命令示例
-- 删除 mysql-bin.000003 之前的 Binlog 文件，也就是 mysql-bin.000001 和 mysql-bin.000002
mysql> PURGE BINARY LOGS TO 'mysql-bin.000003';

-- 删除 2019-12-08 零点之前的 Binlog 文件
mysql> PURGE MASTER LOGS BEFORE '2019-12-08 00:00:00';
```

自动清理 Binlog 是通过配置 MySQL 的 expire_logs_days 参数（在 my.cnf 中配置）。例如，设置 expire_logs_days = 10，表示系统保留10天的 Binlog，第11天将会删除第1天的 Binlog。但是，这种方式会出现瞬间过高的 IO，从而导致业务出现性能抖动。

6 总结

MySQL 日志系统按照日志的功能做了“细致的”分类，这其实也对我们日常设计日志系统提供了参考。虽然看起来日志种类繁多，但是让每一类日志只负责“一部分”功能，也降低了整体的学习难度。学习日志系统，一定要亲力亲为，尝试打开、关闭各类日志，并分析日志内容。最终，能够熟练的掌握利用日志排查问题、优化系统等等技能。

7 问题

为什么 MySQL 要把日志分成很多类，这样做的优缺点是什么 ？

错误日志只会记录“关键错误”，你能举例说明哪些是关键错误吗 ？

你能总结 MySQL 各个种类的日志所负责的功能吗 ？

根据你的理解，谈一谈线上应用应该开启哪几类日志 ？ 为什么 ？

Binlog 的内容是二进制的，你平时用到过吗 ？ 是怎么解析的呢 ？

8 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档：MySQL Server Logs

MySQL 官方文档：Server System Variable Reference

MySQL 官方文档：mysqlbinlog — Utility for Processing Binary Log Files

MySQL 官方文档：Binary Logging Options and Variables

}

14 你应该知道的系统数据库及常用系统表

16 视图应该怎样去应用和管理呢？