



链滴

如何使用 MySQL 慢查询日志进行性能优化 - Profiling、mysqldumpslow 实例详解

作者: [HiJiangChuan](#)

原文链接: <https://ld246.com/article/1637165900506>

来源网站: 链滴

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



如何使用慢查询日志 对 MySQL 进行性能优化 Profiling、mysqldumpslow 实例详解

卡拉云 kalacloud.com

当我们开始关注数据库整体性能优化时，我们需要一套 MySQL 查询分析工具。特别是在开发中大型目时，往往有数百个查询分布在代码库中的各个角落，并实时对数据库进行大量访问和查询。如果没有一套趁手的分析方法和工具，就很难发现在执行过程中代码的效率瓶颈，我们需要通过这套工具去定位 SQL 语句在执行中缓慢的问题和原因。

本教程带领大家学习和实践 MySQL Server 内置的查询分析工具——慢查询日志、[mysqldumpslow](#)、[rofiling](#)，详细讲解如何使用他们提升代码执行效率。如果你想根据自己的 workflow 开发一套数据库管理工具，推荐使用卡拉云。只要你会写 SQL，无需会前端也可以轻松搭建属于自己的后台查询工具详见本文文末。

一. 有关 MySQL 慢查询日志

1. 慢查询日志是什么？

MySQL 慢查询日志是用来记录 MySQL 在执行命令中，响应时间超过预设阈值的 SQL 语句。

记录这些执行缓慢的 SQL 语句是优化 MySQL 数据库效率的第一步。

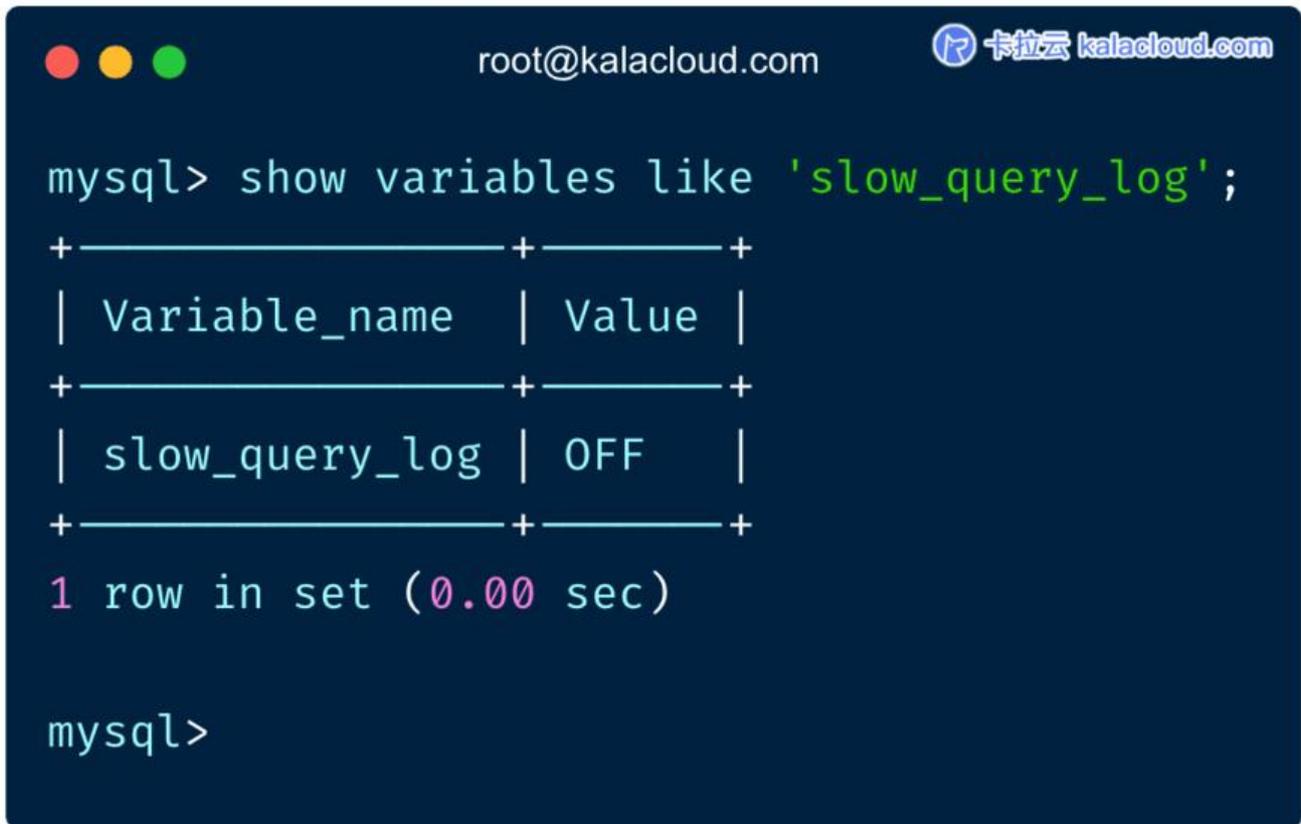
默认情况下，慢查询日志功能是关闭的，需要我们手动打开。当然，如果不是调优需求的话，一般也建议长期启动这个功能，因为开启慢查询多少会对数据库的性能带来一些影响。慢查询日志支持将记写入文件，当然也可以直接写入数据库的表中。

2. 配置并打开慢查询日志

(1) 在 MySQL Server 中临时开启慢查询功能

在 MySQL Server 中，默认情况慢查询功能是关闭的，我们可以通过查看此功能的状态

```
show variables like 'slow_query_log';
```



```
root@kalacloud.com
mysql> show variables like 'slow_query_log';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | OFF   |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

如上图所示，慢查询日志（slow_query_log）的状态为关闭。

我们可以使用以下命令开启并配置慢查询日志功能，在 **mysql** 中执行以下命令：

```
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL slow_query_log_file = '/var/log/mysql/kalacloud-slow.log';
SET GLOBAL log_queries_not_using_indexes = 'ON';
SET SESSION long_query_time = 1;
SET SESSION min_examined_row_limit = 100;
```

SET GLOBAL slow_query_log：全局开启慢查询功能。

SET GLOBAL slow_query_log_file：指定慢查询日志存储文件的地址和文件名。

SET GLOBAL log_queries_not_using_indexes：无论是否超时，未被索引的记录也会记录下来。

SET SESSION long_query_time：慢查询阈值（秒），SQL 执行超过这个阈值将被记录在日志中。

SET SESSION min_examined_row_limit：慢查询仅记录扫描行数大于此参数的 SQL。

****特别注意：**在实践中常常会碰到无论慢查询阈值调到多小，日志就是不被记录。这个问题很有可能是 **min_examined_row_limit** 行数过大，导致没有被记录。**min_examined_row_limit** 在配置中常被略，这里要特别注意。

接着我们来执行查询语句，看看配置。（在 MySQL Server 中执行）

```
show variables like 'slow_query_log%';
```

```
show variables like 'log_queries_not_using_indexes';  
show variables like 'long_query_time';  
show variables like 'min_examined_row_limit';
```

```
mysql> show variables like 'slow_query_log%';
```

Variable_name	Value
slow_query_log	ON
slow_query_log_file	/var/log/mysql/kalacloud-slow.log

```
2 rows in set (0.00 sec)
```

```
mysql> show variables like 'log_queries_not_using_indexes';
```

Variable_name	Value
log_queries_not_using_indexes	ON

```
1 row in set (0.00 sec)
```

```
mysql> show variables like 'long_query_time';
```

Variable_name	Value
long_query_time	1.000000

```
1 row in set (0.01 sec)
```

```
mysql> show variables like 'min_examined_row_limit';
```

Variable_name	Value
min_examined_row_limit	100

```
1 row in set (0.01 sec)
```

```
mysql>
```

以上修改 MySQL 慢查询配置的方法是用在[临时监测数据库运行状态](#)的场景下，当 MySQL Server 重启时，以上修改全部失效并恢复原状。

扩展阅读：[六类 MySQL 触发器使用教程及应用场景实战案例](#)

(2) 将慢查询设置写入 MySQL 配置文件，永久生效

虽然我们可以在命令行中对慢查询进行动态设置，但动态设置会随着重启服务而失效。如果想长期开慢查询功能，需要把慢查询的设置写入 MySQL 配置文件中，这样无论是重启服务器，还是重启 MySQL，慢查询的设置都会保持不变。

MySQL conf 配置文件通常在 `/etc` 或 `/usr` 中。我们可以使用 `find` 命令找到配置文件具体的存放位。

```
sudo find /etc -name my.cnf
```



```
root@kalacloud.com
root@kalacloud.com:~$ sudo find /etc -name my.cnf
[sudo] password for root:

/etc/mysql/my.cnf
/etc/alternatives/my.cnf

root@kalacloud.com:~$
```

找到位置后，使用 `nano` 编辑 `my.cnf` 将慢查询设置写入配置文件。

```
sudo nano /etc/mysql/my.cnf
```

```
[mysqld]
```

```
slow-query-log = 1
slow-query-log-file = /var/log/mysql/localhost-slow.log
long_query_time = 1
log-queries-not-using-indexes
```

使用 `nano` 打开配置文件，把上面的代码写在 `[mysqld]` 的下面即可。 `ctrl+X` 保存退出。

```
sudo systemctl restart mysql
```

重启 MySQL Server 服务，使刚刚修改的配置文件生效。

****特别注意：**直接在命令行中设置的慢查询动态变量与直接写入 `my.cnf` 配置文件的语法有所不同。

扩展阅读：[10种 MySQL 管理工具 横向测评 - 免费和付费到底怎么选?](#)

举例：动态变量是 `slow_query_log`，写入配置文件是 `slow-query-log`。这里要特别注意。

更多 MySQL 8.0 动态变量语法可查看 [MySQL 官方文档](#)。

二. 使用慢查询功能记录日志

到这里我们已经配置好慢查询功能所需要的一切。下面咱们写一个示例，在这个示例中我们来一起学习如何查看和分析慢查询日志。

你可以打开两个连接到服务器的命令行窗口，一个用来写 MySQL 代码，另一个用来查看日志。

注意：以下教程中，有些代码是在命令行中执行，有些是在 MySQL Server 中执行，请注意分辨。

登录 MySQL Server，创建一个数据库，写入一组示例数据。

```
CREATE DATABASE kalacloud_demo;
USE kalacloud_demo;
CREATE TABLE users ( id TINYINT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) );
INSERT INTO users (name) VALUES ('Jack Ma'),('Lei Jun'),('Wang Xing'),('Pony Ma'),('Zhang YiMing'),('Ding Lei'),('Robin Li'),('Xu Yong'),('Huang Zheng'),('Richard Liu');
```

为了保证大家与教程配置保持一致，咱们一起使用动态变量，再设置一边慢查询参数。

在 MySQL Server 中执行以下 SQL 代码：

```
SET GLOBAL slow_query_log = 1;
SET GLOBAL slow_query_log_file = '/var/log/mysql/kalacloud-slow.log';
SET GLOBAL log_queries_not_using_indexes = 1;
SET long_query_time = 10;
SET min_examined_row_limit = 0;
```

现在我们有了一个表中有数据的示例数据库。慢查询功能也已经打开，我们特意把时间阈值 (`long_query_time`) 设置为 10 并且把最小行 (`min_examined_row_limit`) 设置为 0。

接着我们来运行一段代码测试一下：

```
USE kalacloud_demo;
SELECT * FROM users WHERE id = 1;
```

使用主键索引对表进行 `select` 查询，这种查询速度非常快，又使用了索引。因此慢查询日志中不会有何记录。

我们打开慢查询日志，验证一下是否有记录，在命令行中执行以下命令：

```
sudo cat /var/log/mysql/kalacloud-slow.log
```

可以看到 `kalacloud-slow.log` 还没有任何记录。

```
root@kalacloud.com

root@kalacloud.com:~$ sudo cat /var/log/mysql/kalacloud-slow.log
[sudo] password for root:

/usr/sbin/mysqld, Version: 8.0.26-0ubuntu0.20.04.2 ((Ubuntu)). started with:
Tcp port: 3306 Unix socket: /var/run/mysqld/mysqld.sock
Time          Id Command      Argument
```

接着我们在 MySQL Server 中执行以下代码：

```
SELECT * FROM users WHERE name = 'Wang Xing';
```

这段查询代码使用非索引列 (name) 来进行查询，所以慢查询日志在会记录下这个查询。

我们打开日志查看记录变化：

```
sudo cat /var/log/mysql/kalacloud-slow.log
```

```
root@kalacloud.com

root@kalacloud.com:~$ sudo cat /var/log/mysql/kalacloud-slow.log
[sudo] password for root:

/usr/sbin/mysqld, Version: 8.0.26-0ubuntu0.20.04.2 ((Ubuntu)). started with:
Tcp port: 3306 Unix socket: /var/run/mysqld/mysqld.sock
Time          Id Command      Argument
# Time: 2021-09-24T04:07:59.931096Z
# User@Host: root[root] @ localhost [] Id: 9
# Query_time: 0.001032 Lock_time: 0.000449 Rows_sent: 1 Rows_examined: 10
use kalacloud_demo;
SET timestamp=1632456479;
SELECT * FROM users WHERE name = 'Wang Xing';
```

我们可以看到这个非索引查询，已经被记录在慢查询日志中了。

再举个例子。我们提高最小检查行 (min_examined_row_limit) 的检查行数设置为 100，然后再执行查询。

在 MySQL Server 中执行以下代码：

```
SET min_examined_row_limit = 100;
SELECT * FROM users WHERE name = 'Zhang YiMing';
```

执行后，再打开 `kalacloud-slow.log`，可以看到条小于 100 行的查询，没有被记录到日志中。

特别注意：如果慢查询日志中，没有记录任何数据，可以检查以下内容。

(1) 创建日志的目录权限问题，是否有对应的权限。

```
cd /var/log
mkdir mysql
chmod 755 mysql
chown mysql:mysql mysql
```

(2) 另一个可能是查询变量配置问题，把 `my.conf` 文件内有关慢查询的配置清干净，然后重启服务重新配置。看看是不是这里出的问题。

扩展阅读：[如何将 MySQL 的查询结果保存到文件](#)

三. 慢查询日志记录参数详解

接着我们来讲解慢查询日志应该如何分析



```
root@kalacloud.com
root@kalacloud.com:~$ sudo cat /var/log/mysql/kalacloud-slow.log
[sudo] password for root:

/usr/sbin/mysqld, Version: 8.0.26-0ubuntu0.20.04.2 ((Ubuntu)). started with:
Tcp port: 3306 Unix socket: /var/run/mysqld/mysqld.sock
Time          Id Command      Argument
# Time: 2021-09-24T04:07:59.931096Z
# User@Host: root[root] @ localhost [] Id: 9
# Query_time: 0.001032 Lock_time: 0.000449 Rows_sent: 1 Rows_examined: 10
use kalacloud_demo;
SET timestamp=1632456479;
SELECT * FROM users WHERE name = 'Wang Xing';
```

日志中信息的说明：

- **Time**：被日志记录的代码在服务器上的运行时间。
- **User@Host**：谁执行的这段代码。
- **Query_time**：这段代码运行时长。
- **Lock_time**：执行这段代码时，锁定了多久。
- **Rows_sent**：慢查询返回的记录。
- **Rows_examined**：慢查询扫描过的行数。

这些被记录的信息非常有意义，所有超过阈值的代码都会被记录在日志中，我们可以通过这些信息找到 MySQL 查询时效率不佳的代码，有助于我们优化 MySQL 性能。

扩展阅读：[如何在 MySQL 里查询数据库中带有某个字段的所有表名](#)

四. 使用 mysqldumpslow 工具对慢查询日志进行分析

实际工作中，慢查询日志可不像上文描述的那样，仅仅有几行记录。现实中慢查询日志会记录大量慢查询信息，写入也非常频繁。日志记录的内容会越来越长，分析数据也变的困难。好在 MySQL 内置了 `mysqldumpslow` 工具，它可以把相同的 SQL 归为一类，并统计出归类项的执行次数和每次执行的耗等一系列对应的情况。

我们先来执行几行代码让慢查询日志记录下来，然后再用 `mysqldumpslow` 进行分析。

上文我们把 `min_examined_row_limit` 设置为 100，在这里，我们要将它改为 0，慢查询才能有记录在 MySQL Server 中执行以下代码：

```
SET min_examined_row_limit = 0;
```

接着我们执行几条查询命令：

```
SELECT * FROM users WHERE name = 'Wang Xing';  
SELECT * FROM users WHERE name = 'Huang Zheng';  
SELECT * FROM users WHERE name = 'Zhang YiMing';
```

根据前文的慢查询设置，这三条记录都将被记录在日志中。

现在，我们切换到命令行的窗口中，执行 `mysqldumpslow` 命令：

```
sudo mysqldumpslow -s at /var/log/mysql/kalacloud-slow.log
```

返回的数据：



```
root@kalacloud.com  
root@kalacloud.com:~$ sudo mysqldumpslow -s at /var/log/mysql/kalacloud-slow.log  
[sudo] password for root:  
  
Reading mysql slow query log from /var/log/mysql/kalacloud-slow.log  
Count: 3 Time=0.00s (0s) Lock=0.00s (0s) Rows=1.0 (3), root[root]@localhost  
SELECT * FROM users WHERE name = 'S'  
  
Died at /usr/bin/mysqldumpslow line 162, ◇ chunk 3.
```

我们可以看到，返回的数据中，已经把三条类似的 SQL 语句记录抽象成一条记录 `SELECT * FROM use s WHERE name = 'S'` 并且针对这条记录列出了对应的总量和平均量的记录。

常见的 `mysqldumpslow` 命令 平时大家也可以根据自己的常用需求来总结，存好这些脚本备用。

- `mysqldumpslow -s at -t 10 kalacloud-slow.log`：平均执行时长最长的前 10 条 SQL 代码。
- `mysqldumpslow -s al -t 10 kalacloud-slow.log`：平均锁定时间最长的前10条 SQL 代码。
- `mysqldumpslow -s c -t 10 kalacloud-slow.log`：执行次数最多的前10条 SQL 代码。
- `mysqldumpslow -a -g 'user' kalacloud-slow.log`：显示所有 `user` 表相关的 SQL 代码的具体值
- `mysqldumpslow -a kalacloud-slow.log`：直接显示 SQL 代码的情况。

`mysqldumpslow` 的参数命令

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
           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)
-t NUM     just show the top n queries
-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
```

常用的参数讲解：

-s

- al: 平均锁定时间
- at: 平均查询时间 [默认]
- ar: 平均返回记录时间
- c: count 总执行次数
- l: 锁定时间
- r: 返回记录
- t: 查询时间

-t: 返回前 N 条的数据

-g: 可写正则表达，类似于 grep 命令，过滤出需要的信息。如，只查询 X 表的慢查询记录。

-r: rows sent 总返回行数。

mysqldumpslow 日志查询工具好用就好用在它特别灵活，又可以合并同类项式的分析慢查询日志。们在日常工作的使用中，就能够体会 **mysqldumpslow** 的好用之处。

另外 **mysqldumpslow** 的使用参数也可在 [MySQL 8.0 使用手册](#) 中找到。

扩展阅读：[如何查看 MySQL 数据库、表、索引容量大小？找到占用空间最大的表](#)

五. Profiling - MySQL 性能分析工具

为了更精准的定位一条 SQL 语句的性能问题，我们需要拆分这条语句运行时到底在什么地方消耗了少资源。我们可以使用 Profiling 工具来进行这类细致的分析。我们可通过 Profiling 工具获取一条 QL 语句在执行过程中对各种资源消耗的细节。

进入 MySQL Server 后，执行以下代码，启动 Profiling

```
SET SESSION profiling = 1;
```

检查 profiling 的状态

```
SELECT @@profiling;
```

返回数据：0 表示未开启，1 表示已开启。



```
root@kalacloud.com  
mysql> SELECT @@profiling;  
+-----+  
| @@profiling |  
+-----+  
|           1 |  
+-----+  
1 row in set, 1 warning (0.03 sec)
```

执行需要定位问题的 SQL 语句。

```
USE kalacloud_demo;  
SELECT * FROM users WHERE name = 'Jack Ma';
```

查看 SQL 语句状态。

```
SHOW PROFILES;
```

打开 profiling 后，SHOW PROFILES; 会显示一个将 Query_ID 链接到 SQL 语句的表。

```
root@kalacloud.com 卡拉云 kalacloud.com

mysql> SHOW PROFILES;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00032750 | SELECT DATABASE() |
| 2 | 0.06947625 | show databases |
| 3 | 0.09744325 | show tables |
| 4 | 0.00206050 | SELECT * FROM users WHERE name = 'Jack Ma' |
+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)
```

Query_ID: SQL 语句的 ID 编号。

Duration: SQL 语句执行时长。

Query: 具体的 SQL 语句。

执行以下 SQL 代码，将 [# Query_ID] 替换为我们分析的 SQL 代码 **Query_ID** 的编号。

`SHOW PROFILE CPU, BLOCK IO FOR QUERY [# Query_ID];`

即

`SHOW PROFILE CPU, BLOCK IO FOR QUERY 4;`

```
root@kalacloud.com 卡拉云 kalacloud.com

mysql> SHOW PROFILE CPU, BLOCK IO FOR QUERY 4;
+-----+-----+-----+-----+-----+
| Status | Duration | CPU_user | CPU_system | Block_ops_in | Block_ops_out |
+-----+-----+-----+-----+-----+
| starting | 0.000147 | 0.000146 | 0.000000 | 0 | 0 |
| Executing hook on transaction | 0.000008 | 0.000007 | 0.000000 | 0 | 0 |
| starting | 0.000014 | 0.000014 | 0.000000 | 0 | 0 |
| checking permissions | 0.000009 | 0.000008 | 0.000000 | 0 | 0 |
| Opening tables | 0.000098 | 0.000099 | 0.000000 | 0 | 0 |
| init | 0.000015 | 0.000014 | 0.000000 | 0 | 0 |
| System lock | 0.000017 | 0.000017 | 0.000000 | 0 | 0 |
| optimizing | 0.000019 | 0.000019 | 0.000000 | 0 | 0 |
| statistics | 0.000033 | 0.000033 | 0.000000 | 0 | 0 |
| preparing | 0.000035 | 0.000035 | 0.000000 | 0 | 0 |
| executing | 0.000088 | 0.000088 | 0.000000 | 0 | 0 |
| end | 0.000010 | 0.000009 | 0.000000 | 0 | 0 |
| query end | 0.000008 | 0.000007 | 0.000000 | 0 | 0 |
| waiting for handler commit | 0.000013 | 0.000014 | 0.000000 | 0 | 0 |
| closing tables | 0.000018 | 0.000017 | 0.000000 | 0 | 0 |
| freeing items | 0.001259 | 0.000067 | 0.000000 | 0 | 0 |
| logging slow query | 0.000138 | 0.000135 | 0.000000 | 0 | 8 |
| cleaning up | 0.000136 | 0.000138 | 0.000000 | 0 | 0 |
+-----+-----+-----+-----+-----+
18 rows in set, 1 warning (0.01 sec)
```

Status 是执行查询过程中的具体步骤，**Duration** 是完成该步骤所需的时间（以秒为单位）。

我们可以根据这些细节来具体分析，如何优化对应的 SQL 代码。

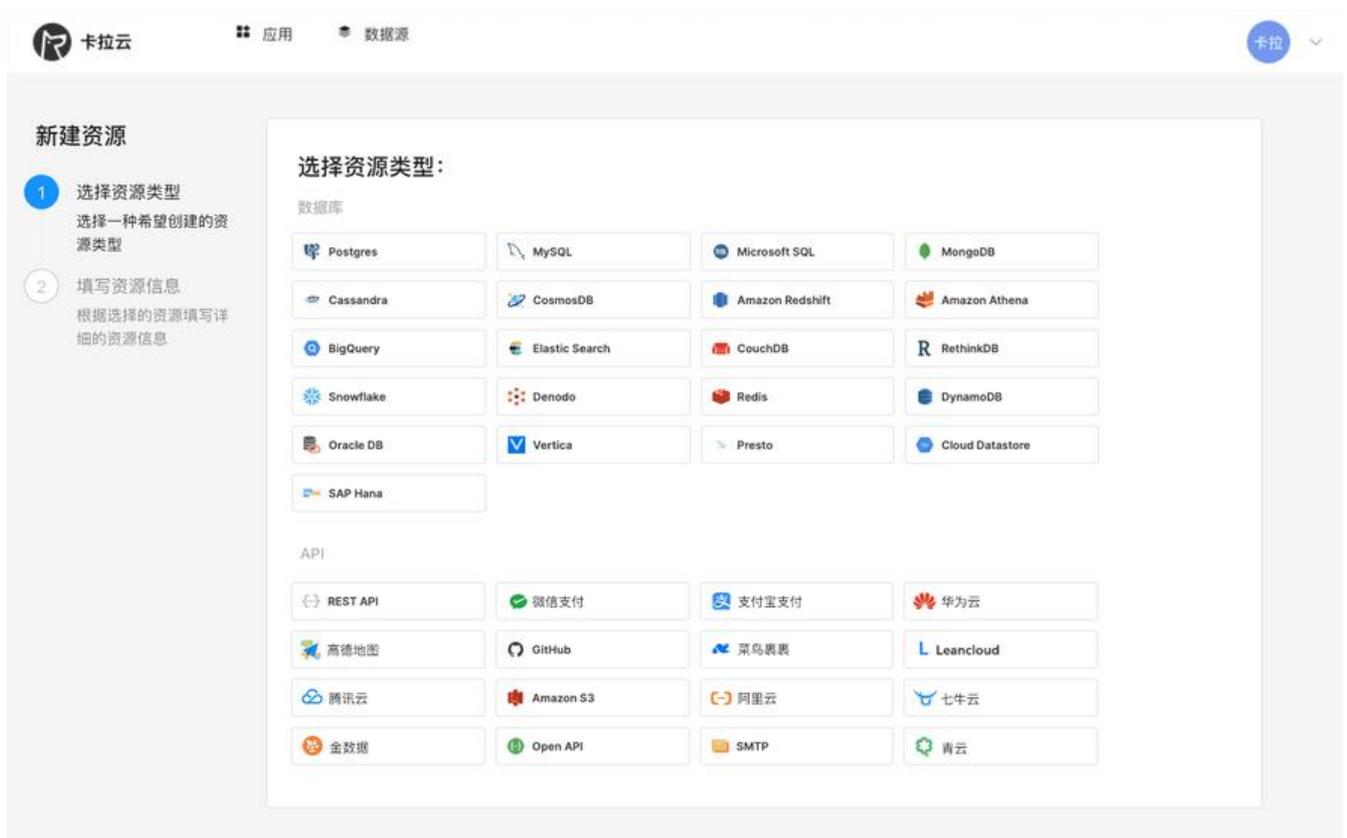
六. 慢查询教程总结

慢查询是让我们看到数据库真实运行状态的工具，对服务器和数据库性能优化有着指导性的意义。无论是生产环境、开发、QA，都可以谨慎的打开慢查询来记录性能日志。

我们可以先把动态变量 `long_query_time` 设置的大一些，观察一下，然后在进行微调。有了慢查询日志，我们就有了优化性能的方向和目标，再使用 `mysqldumpslow` 和 `profiling` 进行宏观和微观的日志分析。找到低效 SQL 语句的细节，进行微调，最终使我们的系统可以获得最佳执行性能。

至此，MySQL 慢查询日志我们就讲解完了，如果你周期性的查看 log 日志，可以使用卡拉云搭一个日志看板，自己不仅查看、分析数据方便，还可以一键分享给组内的小伙伴共享数据。

卡拉云是新一代低代码开发工具，免安装部署，可一键接入包括 MySQL 在内的常见数据库及 API。仅可以像命令行一样灵活，还可根据自己的工作流，定制开发。无需繁琐的前端开发，只需要简单拖，即可快速搭建企业内部工具。**数月的开发工作量，使用卡拉云后可缩减至数天**，欢迎使用我开发的拉云。



卡拉云可快速接入的常见数据库及 API

卡拉云可根据公司工作流需求，轻松搭建数据看板，并且可分享给组内的小伙伴共享数据

卡拉云 用户留存对比 保存 预览

新用户当日完成新手任务后，次日及七日留存比

组件

```

text1: {} # Item
text2: {} # Item
text3: {} # Item
text4: {} # Item
text5: {} # Item
text6: {} # Item
text7: {} # Item
text8: {} # Item
text9: {} # Item
button1: {} # Item
value: "导出数据"
color: "#3c92dc"
disabled: false
eventHandlers: [] # Item
}
container1: {} # Item
datetimepicker1: {} # Item
tabbedcontainer1: {} # Item

```

查询

```

get_user_list: {} # Item
timeout: 10000
sqlstatement: "select * from users;"
triggertype: "AUTOMATIC"
eventHandlers: [] # Item
enabled: true
data: {} # Item
- 0: {} # Item
id: 4
name: "小鹏"
age: "189"
phone: "15246297807"
gender: "女"
debt_amount: "300"
risky: "低风险"
user_wechat_name: "云选可查询"
user_wechat_id: "f1YunReal"

```

发帖留存
点赞留存
评论留存

选择用户注册日期:

当日新注册用户: 3678 发帖新用户次日留存: 35 发帖新用户七日留存: 15

当日发帖新用户: 415 未发帖新用户次日留存: 未发帖新用户七日留存: !

当日点赞新用户: 1785

正在执行查询...get_users 0.5秒

已启用
 get_user_list

get_user_list
已启用
本查询连接数据源
卡拉云 Demo 留存数据 - MYSQL

get_users
已启用
通用设置

常用组件

- 文本
用于显示文本内容, 支持 Markdown 和 HTML
- 输入框
允许用户输入文本
- 下拉框
从列表中选择值或触发操作
- 图片
通过图片链接展示图片
- 按钮
允许用户执行操作, 如导出数据, 执行查询等
- 表格
展示和允许用户操作表格数据
- 容器
用于将多个组件集合在一起
- 文件选择
允许用户选择文件以便上传
- 弹窗
允许弹窗并在内部放置组件
- 富文本
所见即所得的文本编辑器

仅需拖拽一键生成前端代码，简单一行代码即可映射数据到指定组件中。

卡拉云 优惠券核销 保存 预览

优惠券发放后台

coupon_id	code	优惠券名称	优惠券类型	核销状态	金额	满	减	用户类型	发券人	发券时间	过期时间	发券
892	U9xJ8B3H	新春满减	2	已使用		300	100	2	销售铁蛋	2023-01-01	2031-01-01	新增
896	Yx16so6	现金直减	1	已使用	200			3	销售小黑	2023-01-02	2024-01-01	知平
891	XFcu8pfn	老用户回馈券	1	未使用	100			1	运营铁柱	2021-11-01	2021-12-31	合作
895	nKUq8mLK	快乐父亲节	2	已过期		200	10	1	运营铁柱	2020-01-23	2020-12-31	朋友
894	3AMz7Dol	阳光普照	1	未使用				2	运营小棒	2023-01-01	2024-12-01	拉新
893	T24EQ6WU	老用户激活券	1	已使用	100			3	主管张铁人	2021-07-09	2022-12-31	回传

优惠券信息

优惠券名称:

优惠券类型:

现金券金额:

满减券, 满: 减:

优惠券基本规则

固定期, 有效期至:

领券后, 生效天数:

用户类型:

发券人操作信息

发券人:

发券原因:

所有查询运行完毕

通用 高级
insert_coupon 已启用

- get_coupon
- update_coupon
- insert_coupon

触发方式: MANUAL

SQL语句

```

INSERT INTO users (coupon_name,type,amount,over,by,date,day,user_type,operator,reason) VALUES ("{{coupon_name.value}}","{{type.value}}","{{amount.value}}","{{over.value}}","{{by.value}}","{{date.value}}","{{day.value}}","{{user_type.value}}","{{operator.value}}","{{reason.value}}");

= INSERT INTO users (coupon_name,type,amount,over,by,date,day,user_type,operator,reason) VALUES ("老用户回馈券","1","100","","",2021-12-31","","","运营铁柱","合作厂商渠道销售");

```

成功时触发

卡拉云可直接添加导出按钮，导出适用于各类分析软件的数据格式，方便快捷。立即开通[卡拉云](#)，快搭建属于你自己的后台管理系统。

有关 MySQL 教程，可继续拓展学习：

- [如何远程连接 MySQL 数据库，阿里云腾讯云外网连接教程](#)
- [如何在 MySQL / MariaDB 中导入导出数据，导入导出数据库文件、Excel、CSV](#)
- [如何在两台服务器之间迁移 MySQL 数据库 阿里云腾讯云迁移案例](#)
- [MySQL 中如何实现 BLOB 数据类型的存取，BLOB 有哪些应用场景？](#)
- [如何使用 MySQL Workbench 操作 MySQL / MariaDB 数据库中文指南](#)