From: https://www.jianshu.com/p/b5f6e6008089

## 方法 1. SHOW PROFILE

show profile 是 Mysql5.1 版本之后引入的功能。默认是禁用的，但可以通过服务器变量在会话（连接）级别动态修改。

```
mysql> SET profiling=1;
```

然后，在服务器上执行的所有语句，都会测量其耗费的时间和其他一些查询执行状态变更相关的数据。这个工具功能非常强大，最有用的是在语句执行期间剖析服务器的具体工作。

工作原理：当一条查询提交给服务器时，show profile 将记录剖析信息到一张临时表，并且给查询赋予一个从 1 开始的整数标志符。
举例说明，创建表 shop，表结构如下。

```
mysql> show create table shop;
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
+
| Table | Create Table
|
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
+
| shop  | CREATE TABLE `shop` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '记录 ID',
  `shop_id` int(11) NOT NULL COMMENT '商店 ID',
  `goods_id` int(11) NOT NULL COMMENT '物品 ID',
  `pay_type` tinyint(1) NOT NULL COMMENT '支付方式',
  `price` decimal(10,2) NOT NULL COMMENT '物品价格',
  `comment` varchar(4000) DEFAULT NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY `shop_id` (`shop_id`,`goods_id`),
  KEY `price` (`price`),
  KEY `pay_type` (`pay_type`)
) ENGINE=InnoDB AUTO_INCREMENT=20001 DEFAULT CHARSET=utf8 COMMENT='商
店物品表'                    |
+-------+-------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
+
1 row in set (0.01 sec)
```

预先插入两万条数据，然后执行下面查询。

```
mysql> select * from shop order by shop_id , goods_id limit 9;
[query results omitted]
9 rows in set (0.00 sec)
```

查询返回 9 条结果，MySQL 客户端显示的查询时间只有两位小数，对于一些执行很快的查询精度是不够的。下面看 SHOW PROFILES 的结果。

```
mysql> show profiles;
+----------+------------+-----------------------------------------
------------+
| Query_ID | Duration   | Query
|
+----------+------------+-----------------------------------------
------------+
|        1 | 0.00059225 | select * from shop order by shop_id ,
goods_id limit 9 |
|        2 | 0.00007125 | shop profiles
|
+----------+------------+-----------------------------------------
------------+
2 rows in set (0.00 sec)
```

看到以更高的精度显示查询的响应时间。继续看接下来的输出：

```
mysql> show profile cpu, block io for query 1;
+------------------------------+----------+----------+------------
+--------------+---------------+
```

```
| Status                         | Duration | CPU_user | CPU_system |
Block_ops_in | Block_ops_out |
+--------------------------------+----------+----------+------------
+--------------+---------------+
| starting                       | 0.000029 | 0.000019 |   0.000009 |
0 |             0 |
| Waiting for query cache lock   | 0.000009 | 0.000005 |   0.000004 |
0 |             0 |
| checking query cache for query | 0.000079 | 0.000077 |   0.000002 |
0 |             0 |
| checking permissions           | 0.000014 | 0.000009 |   0.000003 |
0 |             0 |
| Opening tables                 | 0.000025 | 0.000023 |   0.000002 |
0 |             0 |
| System lock                    | 0.000014 | 0.000011 |   0.000003 |
0 |             0 |
| Waiting for query cache lock   | 0.000036 | 0.000034 |   0.000001 |
0 |             0 |
| init                           | 0.000029 | 0.000027 |   0.000002 |
0 |             0 |
| optimizing                     | 0.000010 | 0.000007 |   0.000003 |
0 |             0 |
| statistics                     | 0.000014 | 0.000012 |   0.000002 |
0 |             0 |
| preparing                      | 0.000014 | 0.000011 |   0.000002 |
0 |             0 |
| executing                      | 0.000006 | 0.000004 |   0.000003 |
0 |             0 |
| Sorting result                 | 0.000009 | 0.000006 |   0.000002 |
0 |             0 |
| Sending data                   | 0.000178 | 0.000168 |   0.000010 |
0 |             0 |
| end                            | 0.000008 | 0.000005 |   0.000003 |
0 |             0 |
| query end                      | 0.000009 | 0.000006 |   0.000002 |
0 |             0 |
| closing tables                 | 0.000011 | 0.000033 |   0.000021 |
0 |             0 |
| freeing items                  | 0.000012 | 0.000013 |   0.000010 |
0 |             0 |
| Waiting for query cache lock   | 0.000006 | 0.000008 |   0.000004 |
0 |             0 |
| freeing items                  | 0.000054 | 0.000000 |   0.000060 |
0 |             0 |
```

```
| Waiting for query cache lock   | 0.000006 | 0.000002 |   0.000003 |
0 |            0 |
| freeing items                  | 0.000005 | 0.000002 |   0.000003 |
0 |            0 |
| storing result in query cache  | 0.000006 | 0.000003 |   0.000002 |
0 |            0 |
| logging slow query             | 0.000005 | 0.000003 |   0.000003 |
0 |            0 |
| cleaning up                    | 0.000007 | 0.000004 |   0.000002 |
0 |            0 |
+--------------------------------+----------+----------+------------
+-------------+---------------+
25 rows in set (0.00 sec)
```

看到消耗时间最多是“发送数据（Sending data）”，且是CPU密集型。“Sending data”状态含义很具有误导性，所谓的“Sending data”并不是单纯的发送数据，而是包括“收集 + 发送 数据”。这里的关键是为什么要收集数据，原因在于：mysql使用“索引”完成查询结束后，mysql得到了一堆的行id，如果有的列并不在索引中，mysql需要重新到“数据行”上将需要返回的数据读取出来返回给客户端。

结合Sending data的定义，将目标聚焦在查询语句的返回列上面，根据表结构知道comment的内容最大，把查询语句的comment列去掉，重新查询一次：

```
mysql> select id,shop_id, goods_id, pay_type, price from shop order
by shop_id , goods_id limit 9;
[query results omitted]
9 rows in set (0.00 sec)
```

下面重新看下SHOW PROFILES的结果。

```
mysql> show profiles;
+----------+------------+-------------------------------------------
-------------------------------------------+
| Query_ID | Duration   | Query
|
+----------+------------+-------------------------------------------
-------------------------------------------+
|        1 | 0.00059225 | select * from shop order by shop_id ,
goods_id limit 9                                    |
|        2 | 0.00007125 | shop profiles
|
|        3 | 0.00054275 | select id,shop_id, goods_id, pay_type,
price from shop order by shop_id , goods_id limit 9 |
```

```
+----------+-----------+-----------------------------------------
-----------------------------------------+
3 rows in set (0.00 sec)
mysql> show profile for query 3;
+------------------------------+----------+
| Status                       | Duration |
+------------------------------+----------+
| starting                     | 0.000030 |
| Waiting for query cache lock | 0.000011 |
| checking query cache for query | 0.000090 |
| checking permissions         | 0.000011 |
| Opening tables               | 0.000025 |
| System lock                  | 0.000014 |
| Waiting for query cache lock | 0.000039 |
| init                         | 0.000028 |
| optimizing                   | 0.000009 |
| statistics                   | 0.000013 |
| preparing                    | 0.000014 |
| executing                    | 0.000005 |
| Sorting result               | 0.000009 |
| Sending data                 | 0.000127 |
| end                          | 0.000008 |
| query end                    | 0.000008 |
| closing tables               | 0.000012 |
| freeing items                | 0.000013 |
| Waiting for query cache lock | 0.000006 |
| freeing items                | 0.000044 |
| Waiting for query cache lock | 0.000005 |
| freeing items                | 0.000005 |
| storing result in query cache | 0.000006 |
| logging slow query           | 0.000005 |
| cleaning up                  | 0.000006 |
+------------------------------+----------+
25 rows in set (0.00 sec)
```

从上面的结果看到“Sending data”已经由 0.000178s 降到 0.000127s。

### 方法 2. SHOW STATUS

**show status** 可以显示 mysql 服务器的状态，直接查询 status 而不过滤，查询出会有三百多条信息。因此常用的方法是 **show status like ‘%xxx%’** 进行感兴趣的状态过滤。具体学习可以参考这篇文章：MySQL 优化：使用 show status 查看 MySQL 服务器状态信息。

使用 SHOW STATUS，排查出一个问题，请查看拙作 MySQL server has gone away 报错原因分析。

下面这个方法实际就是以较高的频率比如一秒执行一次 show global status 命令来捕获数据，通过某些计数器（比如 Threads_running、Threads_connected、Questions、Queries）的变化来发现问题。这三个数据的趋势对于服务器级别偶尔停顿的敏感性很高。一般发生此类问题，而其他两个则至少有一个出现尖刺。

```
datou:~$ /usr/bin/mysqladmin ext -i1 -uroot -h127.0.0.01 |awk
'/Queries/{q=$4-
qp;qp=$4}/Threads_connected/{tc=$4}/Threads_running/{printf
"%5d %5d %5d\n",q,tc,$4}'
  144     2     1
    1     2     1
    1     2     1
    1     2     1
    1     2     1
    1     2     1
    1     2     1
```

方法 3. SHOW PROCESSLIST

通过不停地捕获 SHOW PROCESSLIST 的输出，来观察是否有大量线程处于不正常的状态或者有其他不正常的特征。例如查询很少会长时间处于"statistics"状态，这个状态一般是指服务器在查询优化阶段如何确定表关联的顺序--这通常都是非常快的。

使用 SHOW PROCESSLIST 命令时，在尾部加上\G 可以垂直的方式输出结果，这样将每一行记录的每一列都单独输出为一行， 可以方便地使用 sort|uniq|sort 一类的命令来计算某个列值出现的次数：

```
$ mysql -e 'SHOW PROCESSLIST\G' | grep State: | sort | uniq -c | sort
-rn
    744 State:
    67 State: Sending data
    36 State: freeing items
    8 State: NULL
    6 State: end
    4 State: Updating
    4 State: cleaning up
    2 State: update
    1 State: Sorting result
    1 State: logging slow query
```

如果要查看不同的列，只需要修改 grep 的模式即可。在大多数案例中，State 列都非常有用。从这个例子的输出中看到，有很多线程处于查询执行的结束部分的状态，包括"freeing items"、end"、"cleaning up"和"logging slow query"。事实上，在案例中的这台服务器上，同样模式或类似的输出采样出现

了很多次。大量的线程处于"freeing items"状态是出现了大量有问题查询的很明显的特征和指示。

上面的命令行不是唯一查找问题方法。如果 MySQL 服务器的版本较新，也可以直接查询 INFORMATION_SCHEMA 中的 PROCESSLIST 表。上面演示的这个例子是由于 InnoDB 内部的争用和脏块刷新所导致，但有时候原因可能比这个要简单得多。

在 processlist 中，看到哪些运行状态时要引起关注，主要有下面几个：

processlist 状态

## 方法 4. 使用慢查询日志

#### 方法 4.1 修改 mysql 的配置文件 my.cnf

查询 my.cnf 文件路径方法：

```
datou:/var/log/mysql$ which mysqld
/usr/sbin/mysqld
datou:/var/log/mysql$ /usr/sbin/mysqld --verbose --help |grep -A 1
'Default options'
[query results omitted]
Default options are read from the following files in the given order:
/etc/my.cnf /etc/mysql/my.cnf /usr/etc/my.cnf ~/.my.cnf
```

MySQL 读取各个 my.cnf 配置文件的先后顺序是：

- /etc/my.cnf
- /etc/mysql/my.cnf
- /usr/etc/my.cnf
- ~/.my.cnf

按优先级逐个排查/etc/my.cnf /etc/mysql/my.cnf /usr/etc/my.cnf ~/.my.cnf，直到 my.cnf 文件存在。
在 my.cnf 文件[mysqld]里面加上以下内容：

```
# Here you can see queries with especially long duration
slow_query_log_file = /var/log/mysql/mysql-slow.log # 日志位置
slow_query_log = 1          # 设置开启
long_query_time = 0.1               # 慢查询超时记录时间，单位 秒
# log_queries_not_using_indexes # 对没有使用索引的查询进行记录
```

重启 mysq： sudo service mysql restart，查看慢查询配置是否生效使用下面的方法 2，配置生效后就可以在查询中跟踪慢查询。
执行查询 sql：

```
mysql> select * from shop where comment like '%a%' order by comment limit 9;
[query results omitted]
9 rows in set (0.21 sec)
```

看到 mysql-slow.log 输出慢查询日志，查询时间为 0.211029s，与 MySQL 客户端输出的查询时间一致。

```
yuejunzyj:/var/log/mysql$ cat /var/log/mysql/mysql-slow.log
/usr/sbin/mysqld, Version: 5.5.57-0ubuntu0.14.04.1-log ((Ubuntu)).
started with:
Tcp port: 3306  Unix socket: /var/run/mysqld/mysqld.sock
Time                 Id Command    Argument
# Time: 170915 11:43:48
# User@Host: root[root] @ localhost [127.0.0.1]
# Query_time: 0.211029  Lock_time: 0.000101 Rows_sent: 9
Rows_examined: 20009
use study;
SET timestamp=1505447028;
select * from shop where comment like '%a%' order by comment limit 9;
```

**方法 4.2 用命令开启慢查询**

```
mysql> show variables like "%long%";         //查看一下默认为慢查询的时间 10 秒
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+
1 row in set (0.00 sec)

mysql> set global long_query_time=2;          //设置成 2 秒，加上 global，下次进 mysql 生效
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like "%slow%";          //查看一下慢查询是不是已经开启
+---------------------+---------------------------------+
| Variable_name       | Value                           |
+---------------------+---------------------------------+
```

```
| log_slow_queries    | OFF                              |
| slow_launch_time    | 2                                |
| slow_query_log      | OFF                              |
| slow_query_log_file | /usr/local/mysql/mysql-slow.log |
+---------------------+----------------------------------+
4 rows in set (0.00 sec)

mysql> set slow_query_log='ON';                          //加上
global，不然会报错的。
ERROR 1229 (HY000): Variable 'slow_query_log' is a GLOBAL variable
and should be set with SET GLOBAL
mysql> set global slow_query_log='ON';            //启用慢查询，下次
进 mysql 生效
Query OK, 0 rows affected (0.28 sec)

mysql> show variables like "%slow%";              //查看是否已经开启
+---------------------+----------------------------------+
| Variable_name       | Value                            |
+---------------------+----------------------------------+
| log_slow_queries    | ON                               |
| slow_launch_time    | 2                                |
| slow_query_log      | ON                               |
| slow_query_log_file | /usr/local/mysql/mysql-slow.log |
+---------------------+----------------------------------+
4 rows in set (0.00 sec)
```

小礼物走一走，来简书关注我