```
mysql> show global status;
```
可以列出MySQL服务器运行各种状态值

一、慢查询

```
mysql> show variables like '%slow%';
+------------------+-------+
| Variable_name | Value |
+------------------+-------+
| log_slow_queries | ON |
| slow_launch_time | 2 |
+------------------+-------+
mysql> show global status like '%slow%';
+---------------------+-------+
| Variable_name | Value |
+---------------------+-------+
| Slow_launch_threads | 0 |
| Slow_queries | 4148 |
+---------------------+-------+
```
打开慢查询日志可能会对系统性能有一点点影响，如果你的MySQL是主－从结构，可以考虑打开其中一台从服务器的慢查询日志，这样既可以监控慢查询，对系统性能影响又小，另mysql有自带的命令mysqldumpslow可进行查询，例下列命令可以查出访问次数最多的20个sql语句mysqldumpslow -s c -t 20 host-slow.log

二、连接数

经常会遇见”MySQL: ERROR 1040: Too manyconnections”的情况，一种是访问量确实很高，MySQL服务器抗不住，这个时候就要考虑增加从服务器分散读压力，另外一种情况是MySQL配置文件中max_connections值过小：

```
mysql> show variables like 'max_connections';
+-----------------+-------+
| Variable_name | Value |
+-----------------+-------+
| max_connections | 256 |
+-----------------+-------+
```
这台MySQL服务器最大连接数是256，然后查询一下服务器响应的最大连接数：

```
mysql> show global status like 'Max_used_connections';
+----------------------+-------+
| Variable_name | Value |
+----------------------+-------+
| Max_used_connections | 245 |
+----------------------+-------+
```
MySQL服务器过去的最大连接数是245，没有达到服务器连接数上限256，应该没有出现1040错误，比较理想的设置是：

Max_used_connections / max_connections * 100% ≈ 85%

最大连接数占上限连接数的85％左右，如果发现比例在10%以下，MySQL服务器连接数上限设置的过高了。

三、Key_buffer_size

key_buffer_size是对MyISAM表性能影响最大的一个参数，下面一台以MyISAM为主要存储引擎服务器的配置：

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

```
+-----------------+------------+
| Variable_name | Value      |
+-----------------+------------+
| key_buffer_size | 536870912 |
+-----------------+------------+
```
分配了512MB内存给key_buffer_size，我们再看一下key_buffer_size的使用情况：
```
mysql> show global status like 'key_read%';
+-----------------------+-------------+
| Variable_name         | Value       |
+-----------------------+-------------+
| Key_read_requests     | 27813678764 |
| Key_reads             | 6798830     |
+-----------------------+-------------+
```
一共有27813678764个索引读取请求，有6798830个请求在内存中没有找到直接从硬盘读取索引，计算索引未命中缓存的概率：
key_cache_miss_rate ＝ Key_reads / Key_read_requests * 100%
比如上面的数据，key_cache_miss_rate为0.0244%，4000个索引读取请求才有一个直接读硬盘，已经很BT 了，key_cache_miss_rate在0.1%以下都很好（每1000个请求有一个直接读硬盘），如果key_cache_miss_rate在 0.01%以下的话，key_buffer_size分配的过多，可以适当减少。
MySQL服务器还提供了key_blocks_*参数：
```
mysql> show global status like 'key_blocks_u%';
+-----------------------+-------------+
| Variable_name         | Value       |
+-----------------------+-------------+
| Key_blocks_unused     | 0           |
| Key_blocks_used       | 413543      |
+-----------------------+-------------+
```
Key_blocks_unused 表示未使用的缓存簇(blocks)数，Key_blocks_used表示曾经用到的最大的blocks数，比如这台服务器，所有的缓存都用到了，要么 增加key_buffer_size，要么就是过渡索引了，把缓存占满了。比较理想的设置：Key_blocks_used / (Key_blocks_unused + Key_blocks_used) * 100% ≈ 80%

四、临时表

```
mysql> show global status like 'created_tmp%';
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Created_tmp_disk_tables | 21197   |
| Created_tmp_files       | 58      |
| Created_tmp_tables      | 1771587 |
+-------------------------+---------+
```
每次创建临时表，Created_tmp_tables增加，如果是在磁盘上创建临时表，Created_tmp_disk_tables也增加,Created_tmp_files表示MySQL服务创建的临时文件文件数，比较理想的配置是：
Created_tmp_disk_tables / Created_tmp_tables * 100% <= 25%
比如上面的服务器Created_tmp_disk_tables / Created_tmp_tables * 100% ＝ 1.20%，应该相当好了。我们再看一下MySQL服务器对临时表的配置：

```
mysql> show variables where Variable_name in ('tmp_table_size',
'max_heap_table_size');
+--------------------+-----------+
| Variable_name      | Value     |
+--------------------+-----------+
| max_heap_table_size | 268435456 |
| tmp_table_size     | 536870912 |
+--------------------+-----------+
```

只有256MB以下的临时表才能全部放内存，超过的就会用到硬盘临时表。

五、Open Table情况

```
mysql> show global status like 'open%tables%';
+--------------+-------+
| Variable_name | Value |
+--------------+-------+
| Open_tables | 919 |
| Opened_tables | 1951 |
+--------------+-------+
```

Open_tables 表示打开表的数量，Opened_tables表示打开过的表数量，如果Opened_tables数量过大，说明配置中 table_cache(5.1.3之后这个值叫做table_open_cache)值可能太小，我们查询一下服务器table_cache值：

```
mysql> show variables like 'table_cache';
+--------------+-------+
| Variable_name | Value |
+--------------+-------+
| table_cache | 2048 |
+--------------+-------+
```

比较合适的值为：

```
Open_tables / Opened_tables     * 100% >= 85%
Open_tables / table_cache * 100% <= 95%
```

六、线程使用情况

```
mysql> show global status like 'Thread%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Threads_cached | 46 |
| Threads_connected | 2 |
| Threads_created | 570 |
| Threads_running | 1 |
+------------------+-------+
```

如果我们在MySQL服务器配置文件中设置了thread_cache_size，当客户端断开之后，服务器处理此客户的线程将会缓存起来以响应下一个客户 而不是销毁（前提是缓存数未达上限）。Threads_created表示创建过的线程数，如果发现Threads_created值过大的话，表明MySQL服务器一直在创建线程，这也是比较耗资源，可以适当增加配置文件中thread_cache_size值，查询服务器 thread_cache_size配置：

```
mysql> show variables like 'thread_cache_size';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| thread_cache_size | 64 |
```

```
+------------------+-------+
```
示例中的服务器还是挺健康的。

七、查询缓存(query cache)

```
mysql> show global status like 'qcache%';
+-------------------------+-----------+
| Variable_name           | Value     |
+-------------------------+-----------+
| Qcache_free_blocks      | 22756     |
| Qcache_free_memory      | 76764704  |
| Qcache_hits             | 213028692 |
| Qcache_inserts          | 208894227 |
| Qcache_lowmem_prunes    | 4010916   |
| Qcache_not_cached       | 13385031  |
| Qcache_queries_in_cache | 43560     |
| Qcache_total_blocks     | 111212    |
+-------------------------+-----------+
```

MySQL查询缓存变量解释：

Qcache_free_blocks：缓存中相邻内存块的个数。数目大说明可能有碎片。FLUSH QUERY CACHE会对缓存中的碎片进行整理，从而得到一个空闲块。

Qcache_free_memory：缓存中的空闲内存。

Qcache_hits：每次查询在缓存中命中时就增大

Qcache_inserts：每次插入一个查询时就增大。命中次数除以插入次数就是不中比率。

Qcache_lowmem_prunes： 缓存出现内存不足并且必须要进行清理以便为更多查询提供空间的次数。这个数字最好长时间来看；如果这个数字在不断增长，就表示可能碎片非常严重，或者内存 很少。（上面的 free_blocks和free_memory可以告诉您属于哪种情况）

Qcache_not_cached：不适合进行缓存的查询的数量，通常是由于这些查询不是 SELECT 语句或者用了now()之类的函数。

Qcache_queries_in_cache：当前缓存的查询（和响应）的数量。

Qcache_total_blocks：缓存中块的数量。

我们再查询一下服务器关于query_cache的配置：

```
mysql> show variables like 'query_cache%';
+------------------------------+-----------+
| Variable_name                | Value     |
+------------------------------+-----------+
| query_cache_limit            | 2097152   |
| query_cache_min_res_unit     | 4096      |
| query_cache_size             | 203423744 |
| query_cache_type             | ON        |
| query_cache_wlock_invalidate | OFF       |
+------------------------------+-----------+
```

各字段的解释：

query_cache_limit：超过此大小的查询将不缓存

query_cache_min_res_unit：缓存块的最小大小

query_cache_size：查询缓存大小

query_cache_type：缓存类型，决定缓存什么样的查询，示例中表示不缓存 select sql_no_cache 查询

query_cache_wlock_invalidate：当有其他客户端正在对MyISAM表进行写操作时，如果查询在query cache中，是否返回cache结果还是等写操作完成再读表获取结果。

query_cache_min_res_unit的配置是一柄”双刃剑”，默认是4KB，设置值大对大数据查询有好处，但如果你的查询都是小数据查询，就容易造成内存碎片和浪费。
查询缓存碎片率 = Qcache_free_blocks / Qcache_total_blocks * 100%
如果查询缓存碎片率超过20%，可以用FLUSH QUERY CACHE整理缓存碎片，或者试试减小query_cache_min_res_unit，如果你的查询都是小数据量的话。
查询缓存利用率 = (query_cache_size - Qcache_free_memory) / query_cache_size * 100%
查询缓存利用率在25%以下的话说明query_cache_size设置的过大，可适当减小；查询缓存利用率在80%以上而且Qcache_lowmem_prunes > 50的话说明query_cache_size可能有点小，要不就是碎片太多。
查询缓存命中率 = (Qcache_hits - Qcache_inserts) / Qcache_hits * 100%
示例服务器 查询缓存碎片率 ＝ 20.46%，查询缓存利用率 ＝ 62.26%，查询缓存命中率 ＝ 1.94%，命中率很差，可能写操作比较频繁吧，而且可能有些碎片。

八、排序使用情况

```
mysql> show global status like 'sort%';
+-------------------+------------+
| Variable_name     | Value      |
+-------------------+------------+
| Sort_merge_passes | 29         |
| Sort_range        | 37432840   |
| Sort_rows         | 9178691532 |
| Sort_scan         | 1860569    |
+-------------------+------------+
```

Sort_merge_passes 包括两步。MySQL 首先会尝试在内存中做排序，使用的内存大小由系统变量Sort_buffer_size 决定，如果它的大小不够把所有的记录都读到内存中，MySQL 就会把每次在内存中排序的结果存到临时文件中，等MySQL 找到所有记录之后，再把临时文件中的记录做一次排序。这再次排序就会增加 Sort_merge_passes。实际上，MySQL会用另一个临时文件来存再次排序的结果，所以通常会看到 Sort_merge_passes增加的数值是建临时文件数的两倍。因为用到了临时文件，所以速度可能会比较慢，增加 Sort_buffer_size 会减少Sort_merge_passes 和 创建临时文件的次数，但盲目的增加Sort_buffer_size 并不一定能提高速度

九、文件打开数(open_files)

```
mysql> show global status like 'open_files';
+--------------+-------+
| Variable_name | Value |
+--------------+-------+
| Open_files | 1410     |
+--------------+-------+
mysql> show variables like 'open_files_limit';
+------------------+-------+
| Variable_name | Value |
+------------------+-------+
| open_files_limit | 4590     |
+------------------+-------+
```

比较合适的设置：Open_files / open_files_limit * 100% <= 75%

十、表锁情况

```
mysql> show global status like 'table_locks%';
+----------------------+-----------+
| Variable_name        | Value     |
```

```
+----------------------+-----------+
| Table_locks_immediate | 490206328 |
| Table_locks_waited | 2084912 |
+----------------------+-----------+
```

Table_locks_immediate 表示立即释放表锁数，Table_locks_waited表示需要等待的表锁数，如果Table_locks_immediate / Table_locks_waited >5000，最好采用InnoDB引擎，因为InnoDB是行锁而MyISAM是表锁，对于高并发写入的应用InnoDB效果会好些。示例中的服务器Table_locks_immediate / Table_locks_waited ＝ 235，MyISAM就足够了。

十一、表扫描情况

```
mysql> show global status like 'handler_read%';
+----------------------+-------------+
| Variable_name              | Value       |
+----------------------+-------------+
| Handler_read_first | 5803750          |
| Handler_read_key        | 6049319850     |
| Handler_read_next          | 94440908210 |
| Handler_read_prev          | 34822001724 |
| Handler_read_rnd        | 405482605 |
| Handler_read_rnd_next | 18912877839 |
+----------------------+-------------+
mysql> show global status like 'com_select';
+--------------+-----------+
| Variable_name | Value           |
+--------------+-----------+
| Com_select | 222693559 |
+--------------+-----------+
```

计算表扫描率：

表扫描率 ＝ Handler_read_rnd_next / Com_select

如果表扫描率超过4000，说明进行了太多表扫描，很有可能索引没有建好，增加read_buffer_size值会有一些好处，但最好不要超过8MB。