1. 网络优化
2. 内存优化
3. sql语句优化

通过 show status 来优化 MySQL 数据库

关键字: mysql

1，查看 MySQL 服务器配置信息

```
mysql> show variables;
```

2，查看 MySQL 服务器运行的各种状态值

```
mysql> show global status;
```

3，慢查询

```
mysql> show variables like '%slow%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| log_slow_queries | OFF   |
| slow_launch_time | 2     |
+------------------+-------+
mysql> show global status like '%slow%';
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Slow_launch_threads | 0     |
| Slow_queries        | 279   |
+---------------------+-------+
```

配置中关闭了记录慢查询（最好是打开，方便优化），超过 2 秒即为慢查询，一共有 279 条慢查询

4，连接数

```
mysql> show variables like 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 500   |
+-----------------+-------+

mysql> show global status like 'max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 498   |
+----------------------+-------+
```

设置的最大连接数是 500，而响应的连接数是 498

max_used_connections / max_connections * 100% = 99.6% （理想值 ≈ 85%）

5, key_buffer_size

key_buffer_size 是对 MyISAM 表性能影响最大的一个参数，不过数据库中多为 Innodb

```
mysql> show variables like 'key_buffer_size';
+-----------------+----------+
| Variable_name   | Value    |
+-----------------+----------+
| key_buffer_size | 67108864 |
+-----------------+----------+

mysql> show global status like 'key_read%';
+-------------------+----------+
| Variable_name     | Value    |
+-------------------+----------+
| Key_read_requests | 25629497 |
| Key_reads         | 66071    |
+-------------------+----------+
```

一共有 25629497 个索引读取请求，有 66071 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% =0.27%

需要适当加大 key_buffer_size

```
mysql> show global status like 'key_blocks_u%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Key_blocks_unused | 10285 |
| Key_blocks_used   | 47705 |
+-------------------+-------+
```

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

Key_blocks_used / (Key_blocks_unused + Key_blocks_used) * 100% ≈ 18% （理想值 ≈ 80%）

6,临时表

```
mysql> show global status like 'created_tmp%';
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Created_tmp_disk_tables | 4184337 |
| Created_tmp_files       | 4124    |
| Created_tmp_tables      | 4215028 |
+-------------------------+---------+
```

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：

Created_tmp_disk_tables / Created_tmp_tables * 100% ＝ 99% （理想值<= 25%）

```
mysql> show variables where Variable_name in ('tmp_table_size', 'max_heap_table_size');
```

```
+---------------------+-----------+
| Variable_name       | Value     |
+---------------------+-----------+
| max_heap_table_size | 134217728 |
| tmp_table_size      | 134217728 |
+---------------------+-----------+
```

需要增加 tmp_table_size

7,open table 的情况

```
mysql> show global status like 'open%tables%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   | 1024  |
| Opened_tables | 1465  |
+---------------+-------+
```

Open_tables 去重
Opened_tables 没有去重

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   | 1024  |
+---------------+-------+
```

Open_tables / Opened_tables * 100% =69% 理想值 （>= 85%）
Open_tables / table_cache * 100% = 100% 理想值 (<= 95%)

8, 进程使用情况

```
mysql> show global status like 'Thread%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_cached    | 31    |
| Threads_connected | 239   |
| Threads_created   | 2914  |
| Threads_running   | 4     |
+-------------------+-------+
```

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 | 32    |
+------------------+-------+
```

9，查询缓存(query cache)

```
mysql> show global status like 'qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 2226     |
| Qcache_free_memory      | 10794944 |
| Qcache_hits             | 5385458  |
| Qcache_inserts          | 1806301  |
| Qcache_lowmem_prunes    | 433101   |
| Qcache_not_cached       | 4429464  |
| Qcache_queries_in_cache | 7168     |
| Qcache_total_blocks     | 16820    |
+-------------------------+----------+
```

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            | 33554432 |
| query_cache_min_res_unit     | 4096     |
| query_cache_size             | 33554432 |
| 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。

10,排序使用情况

```
mysql> show global status like 'sort%';
+-------------------+----------+
| Variable_name     | Value    |
+-------------------+----------+
| Sort_merge_passes | 2136     |
| Sort_range        | 81888    |
| Sort_rows         | 35918141 |
| Sort_scan         | 55269    |
+-------------------+----------+
```

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，见 How fast can you sort data with MySQL?（引自 http://qroom.blogspot.com/2007/09/mysql-select-sort.html）

另外，增加 read_rnd_buffer_size(3.2.3 是 record_rnd_buffer_size)的值对排序的操作也有一点的好处，参见：http://www.mysqlperformanceblog.com/2007/07/24/what-exactly-is- read_rnd_buffer_size/

11.文件打开数(open_files)

```
mysql> show global status like 'open_files';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_files    | 821   |
+---------------+-------+

mysql> show variables like 'open_files_limit';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| open_files_limit | 65535 |
+------------------+-------+
```

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

12, 表锁情况

```
mysql> show global status like 'table_locks%';
+-----------------------+---------+
| Variable_name         | Value   |
+-----------------------+---------+
| Table_locks_immediate | 4257944 |
| Table_locks_waited    | 25182   |
+-----------------------+---------+
```

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.

13. 表扫描情况

```
mysql> show global status like 'handler_read%';
+-----------------------+-----------+
| Variable_name         | Value     |
+-----------------------+-----------+
| Handler_read_first    | 108763    |
| Handler_read_key      | 92813521  |
| Handler_read_next     | 486650793 |
| Handler_read_prev     | 688726    |
| Handler_read_rnd      | 9321362   |
| Handler_read_rnd_next | 153086384 |
+-----------------------+-----------+
```

各字段解释参见 http://hi.baidu.com/thinkinginlamp/blog/item/31690cd7c4bc5cdaa144df9c.html，调出服务器完成的查询请求次数：

```
mysql> show global status like 'com_select';
+---------------+---------+
```

```
| Variable_name | Value   |
+---------------+---------+
| Com_select    | 2693147 |
+---------------+---------+
```

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。