



Performance Tuning the MySQL Server

Ligaya Turmelle
MySQL Support Engineer
ligaya@mysql.com



MySQL

- MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.
- a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack.
- Site: <http://www.mysql.com/>
- Download: <http://dev.mysql.com/downloads/>
- Online Manual:
<http://dev.mysql.com/doc/refman/5.1/en/index.html>

Before

- Before you start tweaking the server:
 - > Your single biggest gain in server performance will be from optimizing your queries
 - EXPLAIN
 - Indexes strategies
 - Database design
 - > Optimize the underlying server systems
 - OS
 - File system
 - Network
 - Storage

How MySQL uses memory

- MySQL uses memory in 2 ways – Global buffers and per connection buffers.
- Generally speaking global memory is allocated once when the server starts *
- Per connection memory allocations are for each connection - generally speaking on an “as needed” basis.
 - > These buffers should be “small”
- Global memory + (max_connections * session buffers)

* some of the Global buffers grow to their size like the table and thread caches

Where to get information

- `mysql> SHOW GLOBAL VARIABLES;`
 - > This command will show us what the current settings of the server are.

```

+-----+-----+
| Variable_name      | Value                |
+-----+-----+
| auto_increment_increment | 1                    |
| auto_increment_offset  | 1                    |
| automatic_sp_privileges | ON                   |
| back_log              | 50                   |
| basedir               | /Users/ligaya/mysql_installs/mysql-enterprise-gpl-5.0.50sp1-osx10.5-x86/ |
| binlog_cache_size     | 32768                |
| bulk_insert_buffer_size | 8388608              |
| character_set_client  | latin1               |
.....

```

Where to get information (con't)

- my.cnf file
 - > This file can also contain information not found in the global variables
- Server information
 - > How much RAM on the machine?
 - > Is the machine dedicated to MySQL?
 - > 64 bit processor?
 - > 64 bit MySQL server binary?

Where to get information (con't)

- `mysql> SHOW GLOBAL STATUS;`
 - > This command will give you information on what the server has been doing.
 - > You will want to issue this command twice with some time in between so you can see the change in the information over time
 - `mysql> SHOW GLOBAL STATUS;`
 - `mysql> SELECT SLEEP(180);`
 - `mysql> SHOW GLOBAL STATUS;`
 - > Sample Output:

Where to get information (con't)

```

+-----+-----+
| Variable_name          | Value          |
+-----+-----+
| Aborted_clients        | 6618           |
| Aborted_connects      | 3957           |
| Binlog_cache_disk_use  | 0              |
| Binlog_cache_use       | 0              |
| Bytes_received         | 1320510015     |
| Bytes_sent             | 2978960756     |
| Com_admin_commands     | 32124          |
| ...
| Threads_cached         | 0              |
| Threads_connected      | 1              |
| Threads_created        | 316771         |
| Threads_running        | 1              |
| Uptime                 | 1722523        |
| Uptime_since_flush_status | 1722523        |
+-----+-----+

```

Where to get information (con't)

- Uptime
 - > This information is taken from the status information and is given in number of seconds.
 - Ex: 1086122 seconds = ~19.9 days = 478.5 hrs
 - > This helps you calculate the values in a given time frame and since we have 2 Global status outputs we can see what the change in the values are for a given time under specific circumstances (high load, low load).
 - Ex: $(\text{bytes received}[2] - \text{bytes received}[1]) / (\text{Uptime}[2] - \text{Uptime}[1])$
 $(1320582719 - 1320510015) / (1722729 - 1722523) =$
72704 bytes/ 206 sec = 352.9 bytes/sec

What is the database doing?

- Com_xxx

- > The Com_xxx statement counter variables indicate the number of times each xxx statement has been executed. There is one status variable for each type of statement.

| | | |
|-------------------|--------|--|
| Com_create_table | 1349 | |
| Com_delete | 78813 | |
| Com_insert | 100357 | |
| Com_replace | 3130 | |
| Com_select | 984292 | |
| Com_show_tables | 459 | |
| Com_show_triggers | 898 | |
| Com_update | 285105 | |

- > You can then use this information with the Uptime to find out how well your database is handling the demands it is under – inserts/sec, deletes/sec, etc.

What is the database doing? (con't)

- Handler_*
 - > These are various internal counters. The ones below indicate how well the indexes on your queries are working.
 - **Handler_read_first:**
 - number of times the first entry was read from an index.
 - value is high, it suggests that the server is *doing a lot of full index scans*.
 - **Handler_read_key:**
 - number of requests to read a row based on a key.
 - value is high, it is a good indication that your *tables are properly indexed* for your queries.
 - **Handler_read_next:**
 - number of requests to read the next row in key order.
 - value is incremented if you are *querying an index column with a range constraint or if you are doing an index scan*.

What is the database doing? (con't)

- Handler_* (con't)
 - **Handler_read_prev:**
 - number of requests to read the previous row in key order.
 - mainly used *to optimize ORDER BY ... DESC.*
 - **Handler_read_rnd:**
 - number of requests to read a row based on a fixed position.
 - value is high if you are doing *a lot of queries that require sorting of the result.*
 - probably have a lot of *queries that require MySQL to scan entire tables or you have joins that don't use keys properly.*
 - **Handler_read_rnd_next:**
 - number of requests to read the next row in the data file.
 - value is high if you are doing a lot of table scans.
 - suggests that your *tables are not properly indexed* or your queries are not written to take advantage of the indexes you have.

What is the database doing? (con't)

- **Innodb_***
 - > This has various information on how the InnoDB tablespace is doing.
 - > Some of the information available:
 - **Innodb_buffer_pool_pages_data:**
 - number of pages containing data (dirty or clean).
 - **Innodb_buffer_pool_pages_free:**
 - number of free pages.
 - **Innodb_buffer_pool_pages_total:**
 - The total size of the buffer pool, in pages.

What is the database doing? (con't)

- **Key_***
 - > This provides you with information on how your key buffer is doing. (MyISAM)
 - **Key_blocks_unused:**
 - number of unused blocks in the key cache.
 - use this value to determine how much of the key cache is in use
 - **Key_blocks_used:**
 - number of used blocks in the key cache.
 - a high-water mark that indicates the maximum number of blocks that have ever been in use at one time.

What is the database doing? (con't)

- **Key_***
 - > This provides you with information on how your key buffer is doing. (MyISAM) (con't)
 - **Key_read_requests:**
 - number of requests to read a key block from the cache.
 - a cache hit
 - **Key_reads:**
 - number of physical reads of a key block from disk.
 - a cache miss
 - If large, then your `key_buffer_size` value is probably *too small*.
 - > cache miss rate = $\text{Key_reads} / \text{Key_read_requests}$.
 - > Key buffer efficiency: $1 - \text{cache miss rate}$

What is the database doing? (con't)

- **Qcache_***
 - > These counters provide you with information on how the query cache is doing.
 - **Qcache_free_blocks**: number of free memory blocks
 - **Qcache_hits**: number of query cache hits.
 - **Qcache_inserts**: number of queries added to the query cache.
 - **Qcache_lowmem_prunes**: number of queries that were deleted from the query cache because of low memory.
 - **Qcache_not_cached**: number of non-cached queries (not cacheable, or not cached due to the `query_cache_type` setting).
 - **Qcache_queries_in_cache**: number of queries registered in the query cache.
 - **Qcache_total_blocks**: total number of blocks in the query cache.

What is the database doing? (con't)

- Qcache_*
 - > One quick and easy way to determine if you benefit from using the query cache – Query Cache Hit Rate
 - QC Hit Rate = $Qcache_hits / (Qcache_hits + Com_select)$
 - > Also check to see how often you are invalidating the queries in the cache.
 - Qcache_inserts vs Com_select
 - Want Qcache_inserts \ll Com_select - bigger the difference, the better
 - > Note: keep in mind warming up the cache

What is the database doing? (con't)

- Just a few general information statuses on what the database is doing
 - > Connections: number of connection attempts (successful or not) to the MySQL server.
 - > Queries: number of statements executed by the server. This variable *includes* statements executed within stored programs, unlike the Questions variable. This variable was added in MySQL 5.0.76.
 - > Questions: The number of statements executed by the server. As of MySQL 5.0.72, this includes *only statements sent to the server by clients* and no longer includes statements executed within stored programs, unlike the Queries variable.
 - > Slow_queries: The number of queries that have taken more than long_query_time seconds.

What is the database doing? (con't)

- General information statuses (con't)
 - > Sort_merge_passes: number of merge passes that the sort algorithm has had to do. If this value is large, you should consider increasing the value of the sort_buffer_size system variable.
 - > Table_locks_immediate: number of times a table lock was granted immediately. Typically associate with MyISAM.
 - > Table_locks_waited: number of times we had to wait to get a table lock. Typically associate with MyISAM.
 - If high you may need to consider changing storage engines.
 - > Threads_cached: number of threads in the thread cache.
 - > Threads_connected: number of currently open connections.
 - > Threads_created: number of threads created to handle connections.
 - If Threads_created is big, you may want to increase the thread_cache_size value.
 - The cache miss rate can be calculated as $\frac{\text{Threads_created}}{\text{Connections}}$.

What is the database doing? (con't)

- More information on JOINS
 - > Select_full_join:
 - number of joins that perform *table scans* because they do not use indexes. If this value is not 0, you should carefully check the indexes of your tables.
 - > Select_range_check:
 - number of joins without keys that check for key usage after each row. If this is not 0, you should carefully check the indexes of your tables.
 - > Select_full_range_join:
 - number of joins that used a *range search* on a reference table.
 - > Select_scan:
 - number of joins that did a *full scan* of the first table.
 - > Select_range:
 - number of joins that used *ranges* on the first table.

Variables for tweaking

- **Globals**

- > Main ones:

- **innodb_buffer_pool_size:**

- The size in bytes of the memory buffer InnoDB uses to cache data and indexes of its tables. The larger you set this value, the less disk I/O is needed to access data in tables.
 - On a dedicated database server, you may set this to up to 80% of the machine physical memory size. However, do not set it too large because competition for physical memory might cause paging in the operating system.

Variables for tweaking

- **Globals**

- > Main ones:

- **innodb_log_file_size:**

- larger the value, the less checkpoint flush activity is needed in the buffer pool, saving disk I/O
 - larger the log files longer the crash recovery time
 - size in bytes of *each* log file in a log group.
 - combined size of log files must be less than 4GB
 - Sensible values range from 1MB to 1/N-th of the size of the buffer pool, where N is the number of log files in the group (`innodb_log_files_in_group`).

Variables for tweaking

- **Globals**

- > Main ones (con't)

- **key_buffer_size:**

- Index blocks for MyISAM tables are buffered and are shared by all threads. `key_buffer_size` is the size of the buffer used for index blocks.
 - The maximum allowable setting for `key_buffer_size` is 4GB on 32-bit platforms. As of MySQL 5.0.52, values larger than 4GB are allowed for 64-bit platforms.
 - Using a value that is 25% of total memory on a machine that mainly runs MySQL is quite common. However, if you make the value too large your system might start to page and become extremely slow.

Variables for tweaking

- **Globals**

- > Lesser ones:

- **table_cache:**

- The number of open tables for all threads. Increasing this value increases the number of file descriptors that mysqld requires.
 - You can check whether you need to increase the table cache by checking the `Opened_tables` status variable. If the value of `Opened_tables` is constantly increasing and you don't do `FLUSH TABLES` often (which just forces all tables to be closed and reopened), then you should increase the value of the `table_cache` variable.

Variables for tweaking

- **Globals**

- > Lesser ones:

- **thread_cache_size:**

- threads the server should cache for reuse.
 - When a client disconnects, the client's threads are put in the cache if there are fewer than `thread_cache_size` threads there.
 - Requests for threads are satisfied by reusing threads taken from the cache if possible, and only when the cache is empty is a new thread created.
 - increase to improve performance if you have a lot of new connections.
 - May not provide a notable performance improvement if you have a good thread implementation.
 - $(100 - ((\text{Threads_created} / \text{Connections}) * 100)) =$
thread cache efficiency

Variables for tweaking

- **Globals**

- > Lesser ones:

- **query_cache_size:**

- The amount of memory allocated for caching query results.
 - default value is 0, which disables the query cache.
 - allowable values are multiples of 1024; other values are rounded down to the nearest multiple.
 - Note that query_cache_size bytes of memory are allocated even if query_cache_type is set to 0.

Variables for tweaking

- **Per Session**

- > **max_heap_table_size:**

- sets the maximum size to which MEMORY tables are allowed to grow.

- > **tmp_table_size:**

- maximum size of internal in-memory temporary tables. (The actual limit is determined as the smaller of max_heap_table_size and tmp_table_size.)
 - If an in-memory temporary table exceeds the limit, MySQL automatically converts it to an on-disk MyISAM table.

Variables for tweaking

- Per Session (con't)

- > **read_buffer_size:**

- Each thread that does a sequential scan allocates a buffer of this size (in bytes) for *each* table it scans. If you do many sequential scans, you might want to increase this value.
 - The value of this variable should be a multiple of 4KB. If it is set to a value that is not a multiple of 4KB, its value will be rounded down to the nearest multiple of 4KB.
 - The maximum allowable setting is 2GB though we do not normally recommend having it higher than 8M.

Variables for tweaking

- Per Session (con't)

- > **read_rnd_buffer_size:**

- When reading rows in sorted order following a key-sorting operation, the rows are read through this buffer to avoid disk seeks.
 - Setting the variable to a large value can improve ORDER BY performance by a lot.
 - The maximum allowable setting is 2GB though we do not normally recommend having it higher than 8M.

- > **sort_buffer_size:**

- Each thread that needs to do a sort allocates a buffer of this size. Increase this value for faster ORDER BY or GROUP BY operations.

Variables for tweaking

- Per Session (con't)

- > **bulk_insert_buffer_size:**

- MyISAM uses a special tree-like cache to make bulk inserts faster for INSERT ... SELECT, INSERT ... VALUES (...), (...), ..., and LOAD DATA INFILE when adding data to *non-empty* tables.
 - limits the size of the cache tree in bytes per thread.

- > **join_buffer_size:**

- size of the buffer that is used for plain index scans, range index scans, and joins that do not use indexes and thus perform full table scans.
 - the best way to get fast joins is to add indexes.

Variables for tweaking

- Per Session (con't)

- > **max_allowed_packet:**

- maximum size of one packet or any generated/intermediate string.
 - The packet message buffer is initialized to `net_buffer_length` bytes, but can grow up to `max_allowed_packet` bytes when needed.
 - value by default is small, to catch large (possibly incorrect) packets. You must increase this value if you are using large BLOB columns or long strings. It should be as big as the largest BLOB you want to use.

Variables for tweaking

- Per Session (con't)

- > **myisam_max_sort_file_size:**

- maximum size of the temporary file that MySQL is allowed to use while re-creating a MyISAM index (during REPAIR TABLE, ALTER TABLE, or LOAD DATA INFILE).
 - If the file size would be larger than this value, the index is created using the key cache instead, which is slower.
 - The value is given in bytes.

- > **myisam_sort_buffer_size:**

- size of the buffer that is allocated when *sorting* MyISAM indexes during a REPAIR TABLE or when creating indexes with CREATE INDEX or ALTER TABLE

Questions?



Performance Tuning the MySQL Server

Ligaya Turmelle
MySQL Support Engineer
ligaya@mysql.com

