Best Practices for Configuring my.cnf Options

Options related to Performance in my.cnf

- **innodb_buffer_pool_size**
- This parameter decides the size in bytes of the buffer pool, the memory area where InnoDB caches table and index data.
- This parameter is one of the most important settings in your MySQL instance, and often 80 percent or more of memory is allocated here.
- For example, suppose that you have 100 gigabytes of data but your application accesses only 1 gigabyte of the data regularly. Then having only a few gigabytes in the buffer pool is enough.
- On the other hand, if you have 10 gigabytes of data and the application accesses all of it constantly, you want to have a large enough buffer pool to cover the data and index size.
- Output of show engine innodb status \G , you might see messages like evicted without access X.XX/s.
- If the evicted without access value is not zero, then it means data is getting read into the buffer pool and then pushed out again before it is accessed (also called buffer pool churn).
- If this value is tracked over time and you see it to go up, then you might need to increase the innodb_buffer_pool_size value.

Options related to Performance in my.cnf

- **innodb_log_file_size**
- This parameter determines the fixed size for MySQL’s redo logs. Tuning this value affects the crash recovery time and also overall system performance. The default value is 134,217,728 (about 128 MB).
- The default value for innodb_log_file_size might be small for your workload.
- We recommend increasing this value for any high change rate workload. For substantial insert, update, and delete activity, the recommended initial setting is as follows.
- innodb_log_file_size = 600M
- innodb_log_files_in_group = 2
- The innodb_log_files_in_group parameter defines the number of log files in the log group. Higher values than 2 for innodb_log_files_in_group produce no significant benefit

Options related to Performance in my.cnf

- **innodb_change_buffering**
- The change buffer is a special data structure used for caching changes in secondary index pages that are not in the buffer pool.
- The innodb_change_buffering parameter helps in reducing the substantial I/O operations used to keep secondary indexes up-to-date after data manipulation language (DML) operations.
- This parameter is used to control the extent of change buffering operations.
- The default value is all. We recommend that you disable change buffering by setting the value to none because all causes extremely long shutdown times for upgrades.

Options related to Performance in my.cnf

- **innodb_io_capacity**
- This parameter controls the maximum number of I/O operations per second that InnoDB can perform for InnoDB background tasks.
- Some examples of these tasks are flushing pages from the buffer pool and merging data from the change buffer, as described in the MySQL documentation.
- The default value for this parameter is 200.
- For I/O-intensive systems, a value of 1,000 usually works

Options related to Performance in my.cnf

- **innodb_io_capacity_max**
- This parameter defines the maximum value to which InnoDB is allowed to extend the setting of innodb_io_capacity if flushing activity falls behind.
- The default value is 2,000.
- We recommend as a best practice that you use smaller values for systems with low write loads and larger values for systems with high write activity.

Options related to Performance in my.cnf

- **innodb_stats_persistent**
- This parameter specifies whether the InnoDB table and index statistics produced by the ANALYZE TABLE command are stored on disk.
- These statistics are stored in the mysql.innodb_table_stats and mysql.innodb_index_stats tables.
- If the statistics aren’t stored, they need to be recalculated frequently, after every restart that can cause overhead.
- Available values are 1 (ON) or 0 (OFF). By default, this parameter is turned ON.

Options related to Performance in my.cnf

- **innodb_thread_concurrency**
- You can use this parameter to help solve a problem that arises if the number of active threads is significantly greater than the number of vCPUs.
- For a small instance size, you might be able to process a large number of sessions if most of the sessions are idle.
- However, the number of active threads can become significantly greater than the number of vCPUs. If this happens, the operating system tries to divide all the available vCPUs to available threads.
- The default value of this parameter is 0, which is interpreted as infinite concurrency (no concurrency checking).

Options related to Performance in my.cnf

- **innodb_flush_log_at_trx_commit**
- To achieve durability of any transaction, the log buffer has to be flushed to durable storage.
- However, writing to disk has an impact on performance.
- If performance is given more preference over durability in your system, you can tune the innodb_flush_log_at_trx_commit parameter to control how frequently the log buffer is flushed to disk.
- Possible settings are as follows:
- 0 – This setting does nothing at transaction commit but writes the log buffer to the log file and flush the log file every second. Although the OS tries to flush the log, this flush is not guaranteed. Therefore, in this case you should be able to afford the loss of some of the latest committed transactions if a crash occurs. Data loss also refers here to any potential loss of data, not just transactions. Thus, this setting can be a source of potential corruption.
- 1 – This setting writes the log buffer to the log file and flushes it to durable storage every time a transaction is committed. This is the default (and safest) setting. It guarantees that you don’t lose any committed transactions unless the disk or operating system “fakes” the flush operation.

Options related to Performance in my.cnf

- 2 – This setting writes the log buffer to the file at every commit operation, but it doesn’t flush the buffer. This setting flushes data to disk only once per second. Although the OS tries to flush the log, this flush is not guaranteed.
- The most important difference between this and the 0 setting (and what makes 2 the preferable setting) is that setting to 2 means that no transactions are lost if the MySQL process crashes. However, you can still lose transactions if the entire server crashes or loses power. This is the same data loss as if the crash is caused by the OS. Similar to the value 0, this setting can potentially cause corruption, and data loss is not only loss of transactions.
- Unless the value is set to 1, InnoDB doesn’t guarantee ACID properties.

Options related to Performance in my.cnf

- **tmp_table_size and max_heap_table_size**
- of an internal temporary table is created as an in-memory table but becomes too large, MySQL automatically converts it to an on-disk table.
- The maximum size for in-memory temporary tables is determined from whichever of the values of tmp_table_size and max_heap_table_size is smaller.
- Operations involving on-disk tables are significantly slower than in-memory temporary tables.
- The default value for both of these parameters is 16,777,216 bytes.
- We often come across issues when a customer running a complex query that takes a long time (typically 5–15 minutes) sees copying to tmp table on disk from show full processlist.
- This kind of issue often come with symptoms like large write IOPS and disk queue depth (over 100).
- In such cases, increasing the variables for both tmp_table_size and max_heap_table_size to 64 MB, 128 MB, or even 512 MB resolves the problems. We recommend that you compare performance before and after this change.

Options related to Performance in my.cnf

- **foreign_key_checks**
- Often data import process becomes slower when foreign key checks are enabled.
- The foreign_key_checks parameter is enabled by default, and default value of this parameter is 1 (ON). However, it is not available in the parameter group.
- If your goal is to ensure performance rather than data integrity, to improve import performance you can turn off this parameter by setting it to 0 before running the import SQL statements.

Options related to Performance in my.cnf

- **UNIQUE_CHECKS**
- You can also greatly improve insert performance for large tables by running SET UNIQUE_CHECKS=0 to disable UNIQUE_CHECKS before running INSERT for the data.
- Then we run SET UNIQUE_CHECKS=1 when the insert is done. UNIQUE_CHECKS is also enabled by default, and the default value of this parameter is 1 (ON),

Options related to Performance in my.cnf

- **optimizer_switch**
- As the MySQL documentation says, the optimizer is the set of routines that decide what execution path the DBMS should take for queries.
- The optimizer is responsible for selecting the most efficient query plan to get your results.
- This parameter is used to control the behavior of optimizer and is composed of a set of flags. By turning these flags ON or OFF, we can control the optimizer’s behavior.
- The default values are the following: index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, engine_condition_pushdown=on, index_condition_pushdown=on, mrr=on, mrr_cost_based=on, block_nested_loop=on, batched_key_access=off, materialization=on, semijoin=on, loosescan=on, firstmatch=on, subquery_materialization_cost_based=on, use_index_extensions=on.
- Default values are generally recommended.

Options related to Performance in my.cnf

- **Innodb_read_io_threads and Innodb_write_io_threads**
- These parameters are the number of I/O threads for read and write operations, respectively, in InnoDB. Both of them have default value of 4.
- Increasing these values increase the threads for the specific InnoDB operation.
- Tuning them and setting innodb_read_io_threads = 16 and innodb_write_io_threads = 4 for heavy OLTP workloads can help your instance keep up.

Options related to Performance in my.cnf

- **innodb_status_output_lock**s
- This parameter enables or disables the InnoDB lock monitor.
- Set it to 1 to turn on and to 0 to turn off the monitor.
- The default value is 0.
- When enabled, the parameter displays additional information about locks in SHOW ENGINE INNODB STATUS output and also prints periodic output to the MySQL error log.
- It helps in identifying locks, particularly in cases where no other session is accessing the locked rows and thus there is no conflict.

Options related to Performance in my.cnf

- **table_open_cache**
- This parameter defines the number of table definitions that can be stored in the definition cache. The default value is 4,000.
- As a rule of thumb, set this parameter to a value large enough to keep most of the tables being used open all the time. Opening and closing tables significantly slows operation.
- You might start with a value twice max_connections or twice the total number of tables and then tune from there.

Options related to Performance in my.cnf

- **thread_cache_size**
- This is a configuration variable that defines how many connection handling threads should be cached for reuse, instead of releasing and reallocating them. Thread creation and destruction, which happens at each connect and disconnect, can be expensive.
- The default value of thread_cache_size is 14. Usually this should be set to at least 16.
- If an application has large jumps in the number of concurrent connections and Threads_Created is growing quickly, this variable should be increased to a higher value.
- The goal is not to have threads created in normal operation.