IDERA® | MySQL Whitepaper

A Guide to MySQL Performance Tuning

A GUIDE TO MYSQL PERFORMANCE TUNING

MySQL is a popular relational database management system (RDBMS). It is the preferred platform for business-critical eCommerce systems that companies rely on to service their customers. Production MySQL instances must perform at an optimal level.

This whitepaper explores the most effective methods of tuning MySQL servers and databases for optimal performance. It covers issues like configuration settings, deploying hardware resources, and optimizing SQL queries. The goal is to provide techniques that database administrators (DBAs) can use to enhance MySQL database performance.

TIPS FOR MYSQL PERFORMANCE TUNING

MySQL is a popular relational database management system (RDBMS). It is the preferred platform for business-critical eCommerce systems that companies rely on to service their customers. Production MySQL instances must perform at an optimal level.

This whitepaper explores the most effective methods of tuning MySQL servers and databases for optimal performance. It covers issues like configuration settings, deploying hardware resources, and optimizing SQL queries. The goal is to provide techniques that database administrators (DBAs) can use to enhance MySQL database performance.

- Limit yourself to changing one setting at a time. This allows you to test its benefits before making more modifications.
- Use the SET GLOBAL runtime command sparingly. It is useful for implementing or rolling back a change, but settings should be permanently defined in the configuration file.
- Ensure you are using the right configuration file and the correct section for the change.
- Refrain from using duplicate settings in the configuration file. Use version control to track the changes.

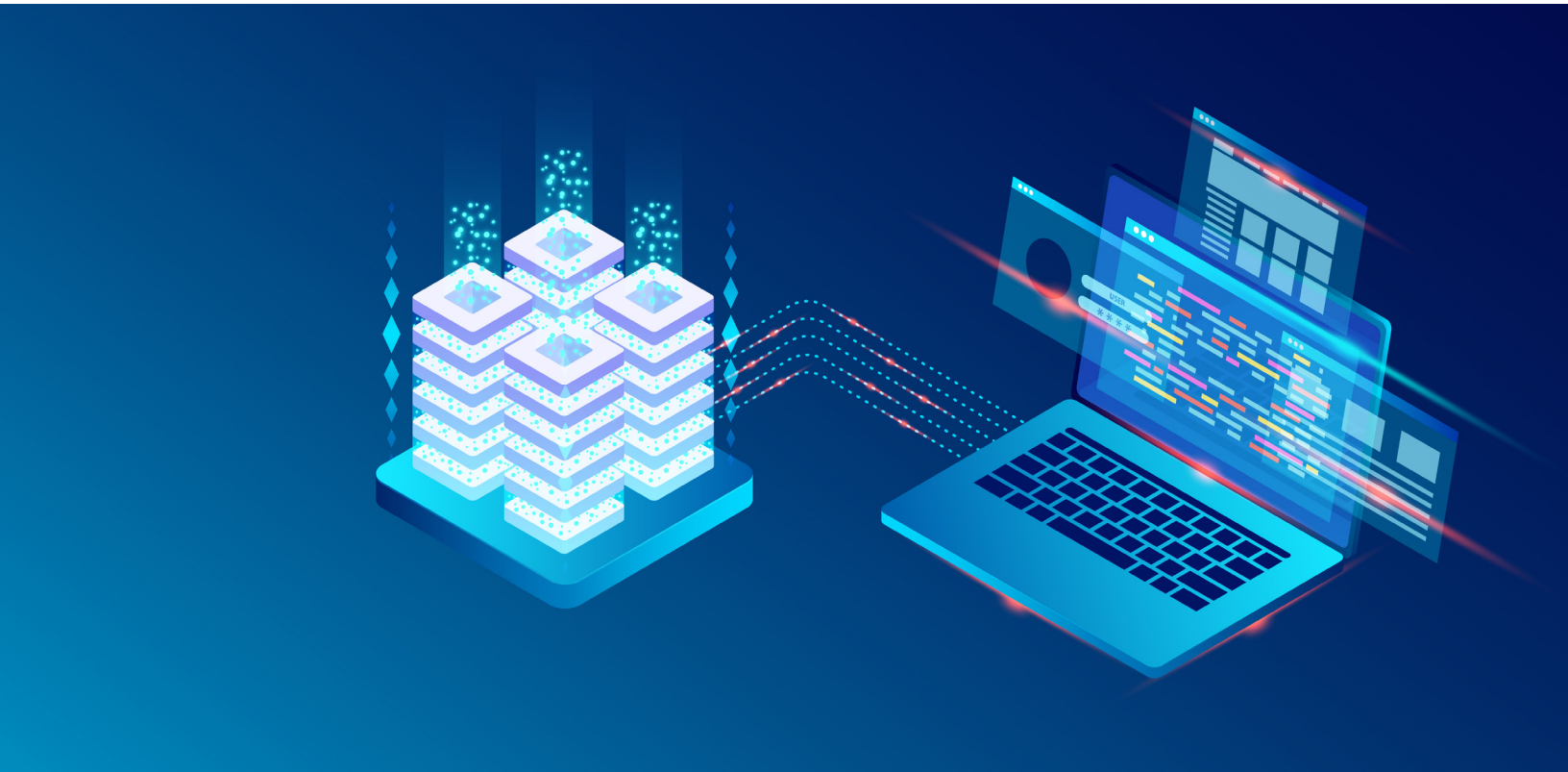
An organized approach to performance tuning produces better results and saves time. There is no magic bullet that will immediately solve performance issues. Optimizing MySQL systems requires an iterative method that should include benchmarking, monitoring, and evaluating how changes affect performance.

INITIAL STEPS

Some initial decisions must be considered before we delve into the details of the techniques that can improve the performance of MySQL servers and databases. These choices can have a big impact on your system's baseline performance capabilities.

- Use the latest version of MySQL that supports your databases. Newer versions offer built-in performance enhancements that provide the results you need without extra work.
- Select InnoDB as the MySQL database engine. InnoDB is the default engine since MySQL 5.5. Go with MyISAM or another alternative if there are compelling circumstances. There are several reasons that InnoDB is the preferred engine, including:
- InnoDB supports row-level locking where MyISAM only supports table-level locking;
- InnoDB supports transactions, including ACID (Atomicity, Consistency, Isolation, Durability) transactions;
- InnoDB has an internal storage manager to manages indexes and base tables for increased efficiency;
- InnoDB supports foreign keys and other relationship constraints.

With the latest version of MySQL and InnoDB in place, you can start looking at methods to improve system performance.



CHOOSING THE APPROPRIATE MYSQL CONFIGURATION SETTINGS

Many configuration settings influence the performance of a MySQL database and server. Remember to change one setting at a time so its effects get evaluated before proceeding. It's much easier to roll back a single change than to recreate the work after many modifications.

MySQL configuration settings get changed by modifying the configuration file.

On Linux systems, the configuration file is - **/etc/mysql/my.cnf.**

On Windows systems, the configuration file is located at -

C: \ProgramData\MySQL\MySQL Server x.x\my.ini. The MySQL version currently running replaces x.x in the file's path.

Below are some impactful variables set in the configuration file to review and/or change.

innodb_buffer_pool_size

The memory buffer pool is used by MySQL to store data in memory, minimizing the number of disk reads necessary to satisfy requests. Ideally, you should provision 2 to 4GB of memory for the operating system with the rest going to MySQL. For example, with 32GB available, a setting of **innodb_buffer_pool_size = 28GB** would reserve 4GB for the OS with the remainder for MySQL.

This is a very important variable for performance tuning. Take full advantage of this variable. System memory resources require MySQL to be installed on a dedicated server.

innodb_log_file_size

This variable represents the size of the committed log files in MySQL and setting it correctly can have a big impact on database performance. Smaller log files require more checkpoint flushes when data is written from the buffer pool to disk. Larger files reduce this number and improve system performance. The default is 48MB which is usually too small for most databases. The optimal size should result in no more than one checkpoint flush event per hour.

innodb_flush_method

The method used to flush data to InnoDB data and log files can have a substantial impact on performance. There are multiple settings available and choosing the correct one will require benchmarking and testing. The potential performance gains are worth the effort.

innodb_dedicated_server

This variable was introduced in MySQL 8.0.3 as a way to automatically configure the preceding three variables based on the amount of memory available on the server. Setting this variable to True activates the automatic configuration of the buffer_pool_size, log_file_size, and flush_method variables. It is recommended to only use this variable when a MySQL instance is on a dedicated server where it is not sharing resources with other applications.

innodb_io_capacity

This variable defines the maximum number of I/O operations that can be performed per second by the InnoDB engine. It can be difficult to find the right setting for this variable and may necessitate iterative changes and benchmarking.

innodb_flush_log_at_trx_commit

This variable is concerned with MySQL's ACID compliance. The default value of 1 indicates full ACID compliance where logs are written and flushed to the disk after every transaction commit. This protects data availability and integrity in the event of an outage. Changing this variable may provide some performance gains, but needs to be done carefully. Systems performing financial transactions should keep the default setting.

query_cache_size

Tuning this variable changes the cache size for pending SQL queries.

max_connection

This variable determines the maximum number of allowable database connections. Setting it too low results in extended user wait times and errors due to too many connections already being open. The default is 151. Increasing the number of connections needs to take into account disk speed and available memory. In cases where you need to exceed 1,000 connections, you may need to also modify operating system parameters to handle the workload.

slow_query_log

Setting this parameter creates a log of files that exceed the defined threshold. The threshold is set in seconds and is indicated with the long_query_time variable. Queries that take longer than the long_query_time to run will be identified and logged. The additional log_queries_not_using_indexes setting will do exactly that and provide insight into queries that may benefit from the use of an index.

Effective Hardware Provisioning

Provisioning the four main hardware resources used by MySQL improves system performance. This can be tricky, as there are limitless combinations for deploying hardware resources.

Storage

The storage subsystem that stores MySQL files has to handle the volume of data transmitted in an acceptable time frame. As a database grows, initial storage provisioning may not be enough and lead to performance degradation. Improving the speed of MySQL's storage devices, such as moving from hard drives to faster solid-state drives can enhance performance.

Processor/CPU

Processor speed is a major determining factor in the responsiveness of a MySQL database. As with the other hardware resources responsible for MySQL performance, CPU requirements can change over time based on the database's evolving workload. Degraded performance may indicate that the processor needs to be upgraded or the database moved to a new and more powerful server.

Memory

Adding more or faster memory can contribute to improved performance. A larger buffer pool will reduce the number of disk reads and improve query response speed. After installing more memory, remember to update the innodb_buffer_pool_size to take advantage of the updated provisioned resources.

Network

Inadequate network bandwidth causes problems including inconsistent connectivity, lost data packets, and excessive latency. Planning is important to the initial network configuration and capacity constraints to ensure resources meet the expected workload.

Performance issues related to network consumption may occur due to changes in usage patterns after the database goes into production. What was once enough to meet user demand may need adjusting to address evolving workloads. Procuring hardware resources is usually not the responsibility of the database team. By identifying why performance is lagging, DBAs can make a case to the decision-makers for more resources.

Optimizing the Database Schema

A database's structure has a significant impact on its performance. The following techniques can optimize the schema of a MySQL database.

- Limit the number of columns - More columns in a table lead to more processing time. While you can use up to 4096 columns in a MySQL table, keeping it under 100 when possible will provide better performance.
- Use suitable data types - MySQL provides over 20 data types designed to address different needs. Using the correct data type such as Text instead of LongText can reduce table index size and result in faster performance.
- Normalize tables - The process of normalizing a database ensures that data is non-redundant. This saves space and contributes to faster database performance.

Tuning SQL Queries

Execution speed impacts MySQL database response time and customer satisfaction. Slow queries that impact critical systems put a business at a competitive disadvantage.

Once configuration changes are complete, it's time to inspect performance of individual SQL queries. First, identify both the queries that execute more and those that take the longest amount of time to complete. Concentrating on these two classes of queries produces likely candidates for optimization.

DBAs can use many techniques to tune SQL queries in their MySQL databases. Here are some of the most productive methods of improving the performance of SQL queries.

Avoid using leading wildcards in queries - Leading wildcards result in the most expensive scans when searching for requested data. Try to use wildcards at the end of a phrase if it is necessary to use one.

Employ indexes where appropriate - The proper use of indexes can dramatically increase query performance by minimizing the number of data rows that need to be searched. Having too few, too many, or the wrong indexes are all query performance killers and need to be optimized to achieve better response times.

Specify columns when using SELECT - Selecting more data than is needed slows MySQL performance. Selecting specific columns rather than searching a table with a SELECT * statement will improve query performance.

Use GROUP BY rather than SELECT DISTINCT - Modifying queries in this way improves performance because *SELECT DISTINCT* results in inefficient database operation.

Use the EXPLAIN function - This function reads and evaluates queries when appended to a query and can help identify inefficiencies that can be addressed by changing SQL code.

Queries are often added to a database, making it important to check performance. As database workloads evolve, efficient queries exhibit degraded performance. You should check queries to ensure performance is up to expectations.

Using Database Tools for MySQL Performance Tuning

Multiple types of tools are available to assist database teams in tuning their systems for optimal performance. Two categories of software applications are especially useful to DBAs looking to improve MySQL response time.

Monitoring tools

MySQL monitoring tools serve multiple purposes when used for performance tuning. A reliable monitoring tool will:

- Identify long-running queries and database locks;
- Report on resource utilization such as disk space, CPU usage, and network traffic;
- Provide a historical baseline against which tuning efforts can be measured.

With a viable monitoring solution, DBAs tune their systems with superior information. This position makes it more likely they will be successful in achieving performance gains.

Query tuning tools

When problem queries come to light, tools that automate the tuning process will improve team productivity. Applications are available to help database teams tune queries by providing features like:

- Identifying missing or unused indexes;
- Providing visual query building and analysis functionality;
- Performing wait-time analysis to help identify problem queries;
- Enabling load testing to check the performance of alternative SQL queries.

A reliable database monitoring tool and query optimizer helps DBAs find and fix the queries that are slowing down database performance. Lack of these tools will make their jobs much harder and may impede their ability to improve performance to the desired degree.

Conclusion

Tuning MySQL performance is a combination of science and art. The science is evident in each variable, setting, and hardware resource that impacts performance. Keep in mind the defined parameters and best practices surrounding each of these items when tuning MySQL databases.

The art comes in with trying to make sense of multitudes of connected parts that impact database performance. Tuning takes experience, reliable information, and at times intuition. Certain DBAs seem to have an innate sense of where the problem lies and get to the root of the problem.

Whether you are a tuning expert or an average DBA, a methodical approach provides a great starting place for improving system response time.

SQL DIAGNOSTIC MANAGER

Monitoring MySQL and MariaDB databases is more efficient when you have the right monitoring tools. Get alerts when something goes wrong, tune your database infrastructure, and easily find and fix problems before they develop into more serious issues or costly outages.

Start your free, 14-day trial of SQL Diagnostic Manager for MySQL today.

Start for FREE