2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA) | 978-1-6654-9079-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICIBA56860.2023.10165291

Research on Performance Optimization of MySQL Database

Pengxiang Gao[1] ,Qi Chen[1] ,Xin Xie[1] ,Chen Wang[1]
1. Institute of Computer Application, China Academy of Engineering Physics, Mianyang, China
pxgaonpu@qq.com, 284279301@qq.com, 861040668@qq.com
Corresponding Author: Chen Wang Email:uestsc_wc@163.com

Abstract*—The application system based on the MySQL database has gradually penetrated into various fields, and the performance of the MySQL database has attracted more and more attention. This paper studies the performance optimization of MySQL databases from five aspects: SQL statement optimization, index optimization, table structure optimization, schema optimization, and server-side optimization.

***Keywords*—database;MySQL;performance optimization**

I. Introduction

As one of the most popular relational databases today, MySQL has become the preferred relational database for most Internet companies because of its free and open source, small size, high efficiency, simple and easy to use, and abundant resources[1~3]. With the increasing complexity of business systems and the continuous expansion of database scale, the amount of data storage has grown exponentially. In addition, users have higher and higher requirements for response time, and the performance problems of database systems are becoming more and more prominent[3]. MySQL performance optimization is the process of optimizing and tuning the MySQL database system to improve its processing power, responsiveness, throughput, reliability, and security to meet the performance needs of the application, thereby providing a better user experience and quality of service. Therefore, it is of great theoretical and practical significance to study how to use optimization technology to optimize the performance of MySQL database [4].

The structure of this article is as follows: The first section briefly introduces the definition and significance of MySQL and MySQL performance optimization, and introduces the research content of this paper. The second section introduces the performance optimization methods of MySQL database. Section III summarizes the full text.

II. Performance optimization method

A. SQL Statement Optimization

Over time, With the accumulation of time, the amount of business data gradually increases, and the impact of SQL execution efficiency on the running efficiency of applications gradually increases. SQL statement optimization refers to the syntax of SQL itself, but there is a better way to write it to achieve the same goal. Here are some suggestions for optimizing MySQL insert and query statements.

(1) optimize MySQL insert statement

1) If many rows are inserted from the same client, use multiple value tables to insert statements at once;

2) If many lines are inserted from different clients, the INSERT DELAYED statement can be used for higher speed;

3) When loading a table from a text file, use LOAD DATA INFILE;

4) REPLACE INSERT with the replace statement depending on the application;

(2) optimize MySQL query statements

1) SELECT queries specific field names, instead of SELECT selecting all fields;

2) Avoid full table scanning and build indexes on columns involved in WHERE, ORDER BY, GROUP BY and DISTINCT;

3) Avoid using fuzzy query before %, which will cause index invalidation and full table scanning;

4) When the fields on either side of the WHERE query condition OR are not indexed, use OR as little as possible;

5) Avoid expression operations on fields in the WHERE clause

6) In the joint query, the small table is used to drive the big table, and the index field of the driven table is used as the restriction field on;

7) Large offset LIMIT query, filter first and then sort;

8) For paging query, complete the sorting paging operation on the index, and then associate back the contents of other columns required by the original table query according to the primary key;

9) IN many cases, it is a good choice to use EXISTS instead of IN;

10) Avoid frequent creation and deletion of temporary tables to reduce the consumption of system table resources.

B. Index Optimization

An index is a structure that sorts the values in one or more columns of a database table. It can be used to quickly access specific information in the database table [5~6]. When selecting the columns to construct an index, you can build an efficient index by following these principles:

1) Data columns of primary keys and foreign keys must be indexed;

2) Frequently queried columns are best indexed;

3) Establish indexes for fields that often appear after keywords WHERE, ORDER BY, GROUP BY and DISTINCT;

4) Avoid indexing frequently accessed columns;

5) Avoid using too many indexes: too many indexes can slow down write operations;

6) For the composite index, the index is established according to the frequency of the field appearing in the query condition.

It is important to note that index optimization is not a one-time process. It requires continuous monitoring and optimization to ensure stable and continuous optimization of query performance.

C. Table Structure Optimization

MySQL database is a database based on row storage, and the database IO operation is in the form of PAGE. If the space occupied by each row record is reduced, the number of rows that can be accessed by each IO will be increased. In addition, since our memory is limited, increasing the number of rows per PAGE increases the amount of cached data per block of memory, and also increases the chance of a data hit in the memory swap, known as the cache hit ratio.

(1) selection of data type

1) Number type: Avoid DOUBLE type; For fixed precision decimals, it is not recommended to use DECIMAL. It is recommended to multiply by a fixed multiple and convert to integer storage. It is recommended to multiply by a fixed multiple and convert to integer storage, which can greatly save storage space. For integer storage, if the data volume is large, you are advised to select TINYINT/INT/BIGINT.

2) Character type: Avoid TEXT data type; CHAR is recommended for fixed-length fields, and VARCHAR is recommended for indefinite-length fields.

3) Time type: Try to use TIMESTAMP because it only needs half the storage space of DATETIME. DATE is recommended for data types that only need to be accurate to a certain date, because it only needs 3 bytes of storage space, less than TIMESTAMP.

4) ENUM &SET: For status fields, you can try to use ENUM to store them, because it can greatly reduce the storage space. To store predefinable attribute data, try using the SET type.

5) LOB type: It is strongly discouraged to store LOB type data in the database.

(2) the choice of character encoding

Character sets directly determine how data is stored and encoded in MySQL. Because the space occupied by different character sets to represent the same content varies greatly, using appropriate character sets can help us reduce the amount of data as much as possible, thus reducing the number of IO operations.

D. Architecture Optimization

MySQL performance optimization from the architectural level mainly includes the use of cache, read-write separation, sub-database and sub-table and other methods.

(1) use cache

It is inevitable that there will be some slow queries in the system. These queries are either large in data volume or complex in query, which makes the query occupy the connection for a long time. If the effectiveness of this kind of data is not particularly strong, we can put this kind of data into the cache system, and obtain the data directly from the cache system within the validity period of the data cache, which can reduce the pressure on the database and improve query efficiency.

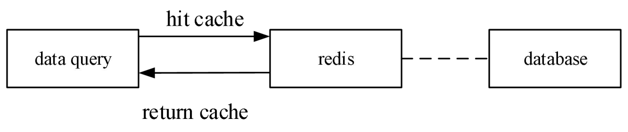

Fig. 1. MySQL architecture of using caching

(2) read-write separation

For the performance problem of a single MySQL server, we can use multiple database servers at the same time, set one of them as the master node, and set the remaining nodes as slave nodes. The data written by the user is only written to the master node, while the read request is allocated to each slave node.The MySQL read-write separation architecture is shown in Figure 2.

(3) sub-database and sub-table

MySQL's read-write separation architecture has a very good effect on reducing the pressure on the main database server. However, with more and more business data, if the data volume of a certain table increases sharply, the query performance of a single table will drop sharply. At this time, we can disperse the data of a single node to multiple nodes for storage, which is sub-database and sub-table. The sub-database and sub-table are divided into vertical sub-database, vertical sub-table, horizontal sub-database and horizontal sub-table. Horizontal distribution is mainly to solve the storage bottleneck; vertical distribution is mainly to reduce concurrency pressure.

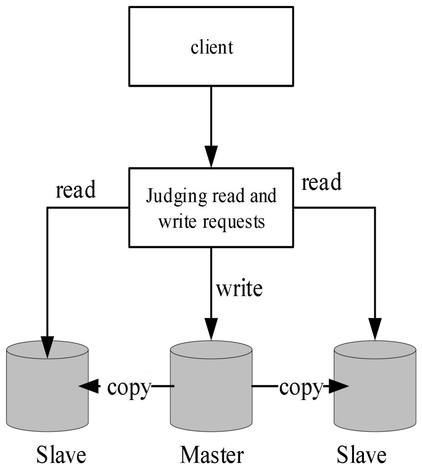

Fig. 2. MySQL architecture with read-write separation

E. Server Optimization

MySQL server optimization mainly optimizes from three aspects: storage engine, configuration parameters, and hardware configuration.

(1) storage engine

In MySQL, a storage engine is a technology used by the database management system to store and retrieve data. Different storage engines have different characteristics and advantages. Here are some suggestions for choosing a storage engine.

TABLE I. COMPARISON OF THREE MYSQL STORAGE ENGINES

Storage engine name	Application scenario
InnoDB	It is the default transactional storage engine of MySQL, supports ACID transactions, has high reliability and data consistency, and supports row-level locks and foreign key constraints. InnoDB is suitable for application scenarios that require high concurrency and data consistency.
MyISAM	It does not support transactions and foreign keys, but has better performance and is suitable for application scenarios with more reads and fewer writes, such as reporting systems and log analysis.
Memory	The Memory storage engine stores data in memory at a very fast speed, but is limited by memory capacity. Memory is suitable for application scenarios with frequent reads and writes but a small amount of data, such as session management, cached data, etc.

(2) database configuration parameters

The configuration items that often need to be optimized in MySQL configuration are summarized in the table below.

TABLE II. SUMMARY OF MYSQL OPTIMIZED CONFIGURATION ITEMS

Parameter Type	Function
The buffer pool parameters	Innodb_buffer_pool_size, key_buffer_size, etc. are used to set the buffer pool size of the InnoDB and MyISAM storage engines to improve the database access speed.
Thread pool parameters	It includes thread_cache_size, thread_handling, etc., which are used to set the size and management strategy of the thread pool to improve concurrent processing capabilities.
Connection pool parameters	It includes max_connections, wait_timeout, etc., which are used to set the size of the connection pool and the connection timeout time to avoid system bottlenecks caused by too many connections.
Query cache parameters	It includes query_cache_size, query_cache_type, etc., which are used to set the size and type of query cache to improve query efficiency.
Log parameters	It includes log_slow_queries, log_error, etc., which are used to record slow queries and error logs for troubleshooting and performance analysis.
Lock parameters	It includes innodb_lock_wait_timeout, innodb_locks_unsafe_for_binlog, etc., which are used to set the lock waiting timeout time and the security of the lock mechanism.
Memory parameters	It includes max_heap_table_size, tmp_table_size, etc., which are used to set the size of the memory table to avoid a large number of temporary tables occupying disk space.
Other parameters	It includes max_allowed_packet, innodb_flush_log_at_trx_commit, etc., which are used to set the maximum packet size and the refresh method of the transaction log, etc.

It should be noted that different MySQL versions and storage engines may have different configuration parameters. In addition, you need to be cautious when adjusting parameters to avoid problems such as system instability or data loss.

(3) hardware configuration

The hardware configuration of the MySQL server determines the lower limit of the MySQL database, and hardware configuration can usually be optimized from three aspects: CPU, memory, and storage.

1) CPU: The performance of MySQL is closely related to the speed of the CPU and the number of cores. Therefore, choosing a CPU with strong performance and a large number of cores can improve the processing capacity of MySQL.

2) Memory: MySQL uses memory to cache data, so enough memory is needed to cache data for queries and operations. It is generally recommended to control the memory usage of MySQL to about 50% of the physical memory.

3) Disk: The read and write operations of the MySQL database require high disk I/O speed, so the use of high-performance disks can improve the processing capacity of MySQL. It is generally recommended to use solid-state drives (SSD) or high-performance storage to store MySQL data.

III. Conclusions

This paper conducts in-depth research on the performance optimization of MySQL database, and gives optimization suggestions from five aspects: SQL statement optimization, index optimization, table structure optimization, architecture optimization, and server optimization, which have certain implications for MySQL performance optimization research.

Acknowledgment

Upon the completion of this paper, I would like to take this opportunity to express my sincere gratitude to all the colleagues and friends who have offered me help. Finally, thank you for reading this article.

References

[1] Satoto K I, Isnanto R R, Kridalukmana R, et al. Optimizing MySQL database system on information systems research, publications and community service[C]//2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). IEEE, 2016: 1-5.

[2] MySQL tutorials: http://dev.MySQL.com/

[3] Gyorödi C, Gyorödi R, Sotoc R. A comparative study of relational and non-relational database models in a Web-based application[J]. International Journal of Advanced Computer Science and Applications, 2015, 6(11): 78-83.

[4] Schwartz B , Zaitsev P , Tkachenko V . High Performance MySQL: Optimization, Backups, and Replication[M]. O'Reilly Media, Inc. 2012.W. Felter, A. Ferreira, R. Rajamony and J. Rubio, An updated performance comparison of virtual machines and Linux containers,ISPASS, Philadelphia, PA, 2015, pp.171-172.

[5] Kohli N, Verma N K. Performance issues of hospital system using MySQL[C]//2010 3rd International Conference on Computer Science and Information Technology. IEEE, 2010, 6: 497-501.

[6] Győrödi C A, Dumşe-Burescu D V, Győrödi R Ş, et al. Performance impact of optimization methods on MySQL document-based and relational databases[J]. Applied Sciences, 2021, 11(15): 6794.