Proceedings of 2018 IEEE
International Conference on Mechatronics and Automation
August 5 - 8, Changchun, China

Optimization Design Method of Cache Extension in MySQL Database

Jianyun Ni, Shuzhi Xie, Zihao Li, Chao Jia

Tianjin Key Laboratory for Control Theory & Applications in Complicated Systems,
School of Electrical and Electronic Engineering, Tianjin University of Technology, Tianjin 300384
E-mail: xsz1358184042@163.com

***Abstract* - MySQL database has been widely used by many large, medium and small enterprises with its open source, lightweight and multi thread, support high concurrency. In the face of high concurrency access to large amounts of data in the network system, the overall design of the database plays an important role in determining system performance. How to optimize the database so that it does not affect the overall performance of the server system during data access becomes a top priority. Based on this situation, this thesis introduces how to optimize the design of MySQL database on large data volume and high concurrency issues. Starting from the choice of different storage engines, through a lot of experiments, verify and analyze different storage engines and caching technologies to solve the database query and Insert equal amount of concurrent execution efficiency.**

Index Terms* - *MySQL database; The engine choice; Caching technology

I. INTRODUCTION

With the development of network technology, there are two challenges in database usage, one is the increasing of data volume, the increase/delete/change/search operation of the database brings huge costs and finally reaches the maximum capacity. At this time, the data processing capability will also encounter the bottleneck. In the end, the key indicators such as system response and data processing are falling. The second is that the number of users accessing the data is also expanding, and in the entire network system, for MySQL, although the pressure can be mitigated by pre-caching [1] (Redis, etc.), read-write separation [2], and library sub-tables, there are still a number of restrictions with respect to the horizontal extension of other components of the system. In view of the above problems, The [3] has been used in the system to use the most frequently used database queries but the content update is very small, like the social type of web site, which is used to be used in an HTML, and it avoids a lot of database access requests. Literature [4] introduced the technology of separation of application server and multimedia server, separated the images with high resource consumption, and solved the problem of high multimedia resource consumption, but the optimization effect was not significant enough for some images with fewer images. Literature [5] and literature [6] aimed at large websites with complex applications, proposed database cluster, library table hash technology and load balancing technology, which can solve the problem of data volume and concurrency high. However, due to the high requirements of database cluster in architecture, cost and expansion, the maintenance and use cost of the website for small enterprises is too high, and pu is not adaptable. The document [7] is a way of improving performance and data security frequently used by large websites, and the mirroring technology can solve the difference of user access speed caused by different network access vendors and territories. Mainly solve the problem of data fault-tolerant rate. Although the above methods can solve the problems such as concurrency and big data volume problems, there are high cost and difficult maintenance problems. In view of the above problems, this paper starts with MySQL concurrent control analysis and caching analysis, compares the concurrency characteristics and index methods under different engines, and proposes a simple and universal solution for large data problems. By adding solid state hard disk as caching, the traditional mechanical disk I/O access is reduced. The bottleneck problem of the concurrency of the database. At last, the method is proved to be feasible through practical verification.

II. ANALYSIS OF DATABASE CONCURRENCY CONTROL

Concurrency control is a mechanism to make sure that the wrong thing is caused by concurrent operations in time. The task of concurrency control in a database management system(DBMS) is to ensure that the isolation and unity of transactions are not broken and the integrity of the database is not broken when the same data is accessed simultaneously in multiple transactions. Lock, timestamp, optimistic concurrency control (positive lock) and pessimistic concurrent control (pessimistic lock) are the main technical means for concurrent control.

In the MySQL database, the InnoDB engine supports both row lock table locks, row-level locking is locking granularity in MySQL, one of the most fine lock said only in view of the current operating line lock, can greatly reduce the conflict of database operations. The InnoDB row lock is implemented by locking the index entry on the index, which means that only if the data is retrieved through the index condition, InnoDB will use the row level lock, otherwise InnoDB will use the table lock.

The MyISAM engine only supports table-level locks, and is the largest lock in MySQL, indicating that the whole table of the current operation is locked, which is simple and has less resource consumption. However, due to the large size of the lock, other transactions will no longer perform any type of operation on the table when the data is added to the data, and

978-1-5386-6075-1/18/$31.00 ©2018 IEEE

only wait for the process to release the exclusive lock after the process is finished, which greatly affects the multi-transaction processing mechanism.Therefore, when accessing database meet larger concurrency, especially high concurrency rewrite operations, MyISAM lock on the whole table with exclusive, when other affairs to deal with the table, can only wait for the release of the lock.

Compared with the InnoDB engine and the MyISAM engine, in the actual application and development, in order to make full use of the system resources, and also because of the high concurrent demand of the system, many transactions are implemented in parallel. If you do not control parallel operations, you will likely lose updates, read obsolete data and read dirty data. And the MyISAM memory engine does not support transaction, and therefore, in the aspects such as concurrency control, data security, and failure recovery, it must be designed to use an InnoDB storage engine with a support transaction capability, to strictly follow and meet the ACID characteristics of the object, to ensure consistency of the account information division, to ensure that the transaction is rolled back to the initial state when a fault such as a crash occurs.Because of the concurrency control property of database, the MyISAM storage engine has insufficient performance on high concurrency. The experimental verification will also be discussed in the following analysis.

III. DATABASE CACHE ANALYSIS AND EXTENSION

A. *Comparative Analysis of Index Methods*

As one of the main functions of database, query is maintained in database system to satisfy the data structure of its particular search algorithm. These data structures refer to data in some way, so that the advanced search algorithm can be implemented on these data structures. This data structure is the index. It is also the reason that the database can get the data quickly and efficiently in a large number of data stores. The b-tree index is the most common type in the index, and most of the database uses a B+Tree data structure that adds sequential access Pointers to store data. In the MySQL database, the most commonly used MyISAM indexes and InnoDB indexes use the B+Tree index structure, but the B+Tree index is implemented in a very different way.

MyISAM storage engine using index files and data files in the separation, index file save the record pointer for the page, only through this page address pointer to read, and read by index. The InnoDB storage engine using clustering index data storage way to realize the b-tree indexes, namely the data line and the adjacent compact key value to be stored together, but the InnoDB gathered only one leaf pages (16 k) record (that is, the clustering indexes satisfy a certain range of records), and therefore includes adjacent key records may be far apart. In the actual development, there is a very useful field that occupies a large number of bytes. By separating the field, the cache efficiency of the table can also be improved to a certain extent. In general, the index itself is very large and cannot be stored in memory, so indexes tend to be stored on disk in the form of index files. Therefore, an index lookup process will produce disk I/O consumption, relative to the memory access, the consumption of disk I/O access to high several orders of magnitude, so in the process of the database operation to decrease as far as possible to find the disk I/O access number. Today, however, most computers are still mostly mechanical disks, and disk I/O is mostly consumed in the search time of the head. The biggest advantage of flash memory is the increase of random access speed. The sequential access of SDD is several times that of random access delay, while the access speed of disk is a hundred times different. Therefore, consider establishing a cache interval between the memory buffer pool and the traditional mechanical storage layer based on SSDS to shorten the I/O access time.

B. *Random Performance Analysis of SSD*

In order to better demonstrate the performance advantage of SSD in database system optimization, we performed a comparison of read/write performance between disk and SSD. In the experiment, we tested on the file level, respectively, to create a large file on the disk and SSD respectively, and then read and write sequential and random read and write operations in the file, each in the file. The size of the request is 8KB. The left part of the blue part is disk order and random read and write. The yellow part on the right is SSD sequence and random read and write part.

In contrast, SSD's random read and write operations show obvious performance advantages, which are dozens of times of disk random read and write. This is because each read and write of the disk requires a search operation, which takes time in milliseconds, so the disk can only reach a maximum of hundreds of IOPS, that is, hundreds to thousands of KB per second. SSD has no mechanical movement when accessing data, so its speed of I/O is very fast. As shown in Figure 1.

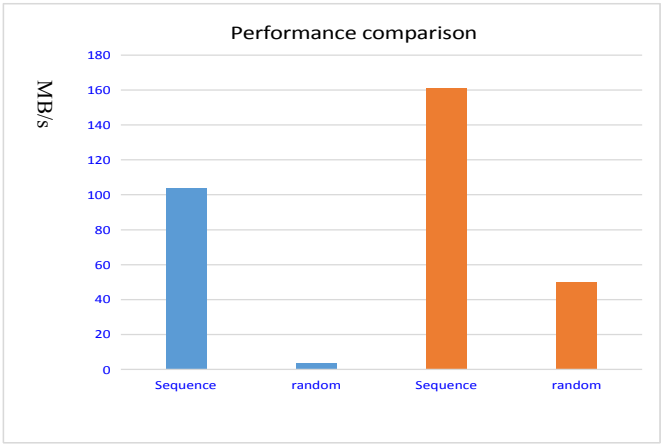

Fig. 1 Performance comparison between disk and SSD

C. *MySQL Cache Extension Mechanism*

MySQL cache mechanism simply means the SQL text and the query results are cached in a hash table, when the MySQL received the client sends the query, in front of the judge whether hit, MySQL will not parse SQL statements directly, but through the query, SQL database, the client agreement conditions, such as the key.To cache table query, if there is a

corresponding data, have direct access to the data as a result, does not need to query the database query process. If a cache is not found, then execute SQL queries, jump to disk to find to call data, after the execution of the SQL query results, will return to the query SQL query result and will result in the secondary buffer.The schematic diagram is shown below.

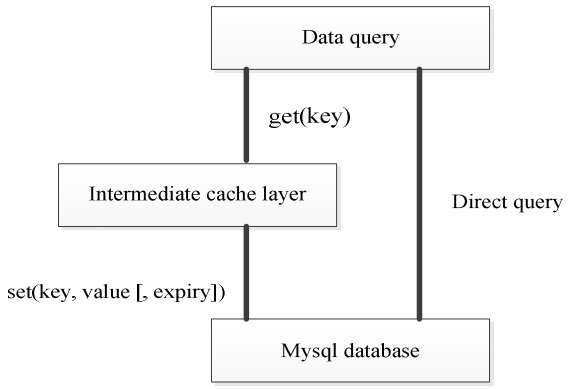

Fig. 2 Caching mechanism

When the next time you need to request the page again, if not in the main buffer pools, first determines whether the page in the secondary buffer pool, if in the reading to the main buffer pool, then not need to access the disk on the page. If the page changes inside the buffer pool, and also exists in the secondary buffer pool, then don't need to modify the secondary buffer pool page, because a page may have been modified many times within the buffer pool, auxiliary time synchronization in the buffer pool pages can make SSD write performance problems exposed, so at this time of the auxiliary buffer pool pages, add to the end of the free list.[9] Once the page of the main buffer pool is removed, it is placed in the secondary buffer pool again. It can be seen that the design of the auxiliary buffer pool is a place for a large number of read operations.

By increasing the data buffer between memory and disk, effective on existing data cache in the Shared storage pressure, at the same time increased the number of bytes of data read every time, will be more data into to the secondary buffer, thereby increasing data pre-reading efficiency, can effectively reduce the memory read data waiting time. The related parameters and formula as shown below:

$$\text{Buffer pool hit rate} \approx \frac{read\_requests}{read\_requests + read\_ahead + reads} \quad (1)$$

reads:Represents the number of times a page is read from a physical disk.

read_aheads:Number of previews.

read_requests:The number of times a page is read from the buffer pool.

It can also be seen from formula (1) that by reducing the number of times and the number of prereading of data from physical disk, the hit probability of the buffer pool is improved. Therefore, in theoretical analysis, it is feasible to improve the processing performance of the database by increasing the auxiliary cache, and finally, we will analyze the design method through experimental verification.

IV. Concurrent Experimental Analysis

Through the above analysis, this section will be in high concurrency and experimental analysis on the problem of large amount of data, contrast MyISAM engine and InnoDB storage engines in the transaction concurrency control and locking mechanism difference influence on concurrency, verify the large amount of data and contain table fields in query and insert operations such as the impact on the system cache. Using MySQLslap, a benchmarking tool that comes with MySQL, test the behavior of the system under different pressures to assess the performance of the system. The MySQL version of the test is MySQL 5.7.21, running on Intel(R) Core(TM) i5-4210 CPU @ 2.60GHz 2.59GHz, and the available memory is 3.89GB.

A. Concurrency Tests Based on Different Storage Engines

This test is mainly conducted on the concurrency comparison experiment of InnoDB and MyISAM storage engines. Simulation more client of concurrent operation, using MySQLslap benchmarking, based on the built table data on MyISAM and InnoDB do concurrency 50 ~ 350 respectively, and repeat the test times for 200 contrast test, the ordinate of a single data in seconds, query time abscissa for concurrent number.

Its experimental result are as follows:

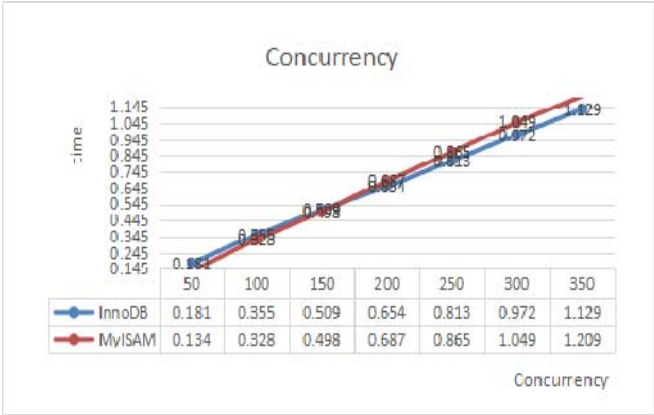

Fig. 3 Comparison and analysis of concurrency

Analysis of experimental results shows that under the same configuration environment, when low concurrency, MyISAM is must read and write performance advantages, but with the continuous improvement of concurrency, InnoDB showed a better ability to adapt, this is because the MyISAM tables between and disk data loading process, in order to protect the key buffer, use MyISAM table lock mechanism, make the MyISAM tables of read operation, does not block other users read requests for the same table, but it will block write requests to the same table, to write operations, MyISAM tables will block other users read and write requests for the same table, i.e. when a thread after get the table write locks, only hold locks may update operations on the table, and the

other threads will wait until the lock release. When the client has a large number of write operations or update operations, in the same table operation, the lock wait for other transactions will be caused, thus the performance of MyISAM table drops sharply. And InnoDB table adopts row level locking mechanism, which greatly reduces the lock wait time and improves the concurrency performance of the table accordingly. Therefore, in systems with high concurrency, the InnoDB engine is preferred.

B. Caching and Table Testing

In this section, we mainly discuss the optimization effect of adding secondary cache combined with table technology to analyze data query insertion and so on under the condition of the same concurrent volume, same configuration and so on. In the experiment, two table fields and table attributes are created respectively, both of which are InnoDB storage engines, and one of them is split vertically. Add the custom function rand_string(n INT),rand_text(n INT),rand_num(), etc., to generate random strings, text, and Numbers to ensure randomness in the data batch inserts.

Part of the code is as follows:

```
create function con_string(n INT)
returns varchar(255)
begin
declare char_int varchar(100) default 'ABCDEFGHIJKLMNOPQRSTUVQXYZ0123456789';
declare return_str varchar(255) default '';
declare i int default 0;
while i< n do
set return_str = concat(return_str, substring(char_int, floor(1+rand()*52),1));
set i = i + 1;
end while;
 return return_str;
end ##
```

Define the stored procedure insert_equipment(in start int(10),in max_num int(10)), and call the stored procedure to insert the table record in bulk. To improve execution efficiency, autocommit = 0, so that the process is always in the transaction until the commit commit completes the current transaction. At the same time, in order to avoid the conflict with the default statement end of mysql in the stored procedure, the delimiter ## statement is used to change the stored procedure terminator, and then restore after the stored procedure is finished.

```
create procedure insert_equipment(in start int(10),in max_num int(10))
begin
declare i int default 0;
set autocommit = 0;
set i = i+1;
Insert into equipment values ((start+i),rand_string(6),rand_num(),rand_text(10),'fine',c
urdate(),rand_text(40));
until i = max_num
end repeat;
commit;
end ##
```

Call the stored procedure and insert 1000000 records in batch.

```
call insert_equipment1(1,1000000)##
```

Experiments, by a table of some commonly used and not commonly used column divided, additional secondary cache, made of high speed I/O can store more data in the memory, in the operations such as query, effectively reduce the number of I/O, improve the cache hit ratio. At the same time the use of CACTI monitoring software to monitor the system CPU usage, observe the real-time dynamic change, in a different table for batch operation can be found that contain table fields of table will occupy more space of system, when the batch operation, user utilization rate has increased dramatically, extremely easy because of the heavy load of a system crash or outage situation happened, and although increased after system sustained high utilization rate, but help to share the system pressure, improve the system stability.

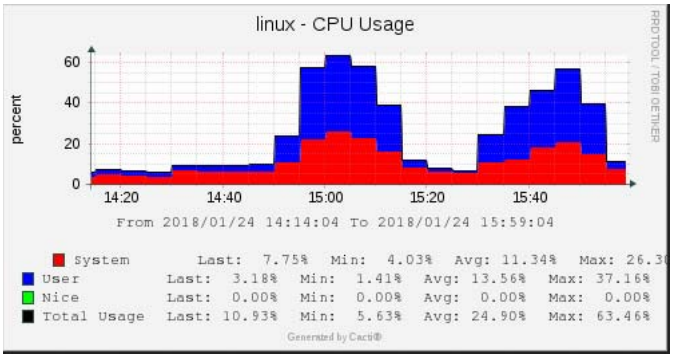

Fig. 4 CPU utilization in stored procedures

V. CONCLUSION

Facing the intricate appeared in practical application, a variety of problems, in terms of performance, improve database and have different solution for a problem, and constantly adjust, according to the requirements of the late optimization plan will change accordingly, so the MySQL optimization is a long and tedious process. In this paper, by combining database design principle analysis under high concurrency and large amount of data of database optimization, and expound the different aspects in order to solve this problem and related experiments, table and cache optimization technique is proposed. The results show that the proposed database optimization design method is effective on high concurrency problems with large table data fields.

ACKNOWLEDGMENT

The authors acknowledge with thanks the financial support by China Petroleum Chemical Co Tianjin branch project,Tianjin Application Foundation and Frontier Technology Research Program (Youth Project) (Grant No.: 15JCQNJC42700), and The Development of Science and Technology Foundation of the Higher Education Institutions of Tianjin (Grant No.:20120705).

REFERENCES

[1] Du, S., Li, C., Mao, X., & Yan, W. (2017). The Optimization of LRU Algorithm Based on Pre-Selection and Cache Prefetching of Files in Hybrid Cloud. *International Conference on Parallel and Distributed Computing, Applications and Technologies* (pp.125-132). IEEE.

[2] Liu, X., Xu, Z., & Pan, S. (2013). A distributed metadata management method based on separation of read and write - taking "digital city" applications as an example. *Science of Wuhan University(Geomatics and Information),* 38(10), 1248-1252.

[3] Liu, Y.. (2015). Web page static research and performance analysis based on Smarty template engine. *Computer and Digital Engineering* (2), 295-298.

[4] Kim, E., & Liu, J. C. L. (2017). An integrated prefetching/caching scheme in multimedia servers. *Journal of Network & Computer Applications*.

[5] Roukh, A., Bellatreche, L., Tziritas, N., & Ordonez, C. (2016). Energy-Aware Query Processing on a Parallel Database Cluster Node. *International Conference on Algorithms and Architectures for Parallel Processing* (pp.260-269). Springer, Cham.

[6] Xu, J., & Yang, X. (2017). Research on key technologies of technological service and management based on cluster load balancing. *Cluster Computing*(3), 1-7.

[7] Otte, L., Danel, R., Řepka, M., Vančura, V., & Johanides, D. (2015). Comparison of database mirror technologies for use in fault-tolerant information system solutions. *Carpathian Control Conference* (pp.360-365). IEEE.

[8] Lee, H., & Lee, S. W. (2016). Performance improvement plan for MySQL insert buffer. *International Conference* (pp.99-101).

[9] Ruan, X., & Chen, H. (2017). Improving Shuffle I/O performance for big data processing using hybrid storage. *International Conference on Computing, NETWORKING and Communications* (pp.476-480). IEEE.

[10] Kang, W. H., Lee, S. W., & Moon, B. (2016). *Flash as cache extension for online transactional workloads.* Springer-Verlag New York, Inc.

[11] Zhao, H., Gai, K., Li, J., & He, X. (2015). Novel Differential Schema for High Performance Big Data Telehealth Systems Using Pre-cache. *IEEE, International Conference on High PERFORMANCE Computing and Communications, 2015 IEEE, International Symposium on Cyberspace Safety and Security, and 2015 IEEE, International Conf on Embedded Software and Systems* (pp.1412-1417). IEEE Computer Society.

[12] Swardika, I. K., & Santiary, P. A. W. (2017). Speed of spatial query of satellite data on various database storage engine.*Electronics Symposium* (pp.465-470). IEEE.

[13] Almeida, R., Bernardino, J., & Bernardino, J. (2015). Performance Evaluation MySQL InnoDB and Microsoft SQL Server 2012 for Decision Support Environments. *Eighth International C* Conference on Computer Science & Software Engineering* (pp.56-62). ACM.

[14] Lv, Q., & Xie, W. (2014). A real-time log analyzer based on mongodb. *Applied Mechanics & Materials,* 571-572, 497-501.