Verifiable Access Technology of Hybrid Database in Distributed System Under Big Data

Jun Jia
Modern Vocational Technology College
799588307@qq.com

Abstract: Targeting at the problem that the existing hybrid database solutions can not simultaneously realize the client's verification and access function to the encrypted data in the public cloud database and the cross database query function in the heterogeneous database environment, this paper designs an access and integration middleware to realize the client's verification and access function to the heterogeneous hybrid database in the public cloud and the cross database query function. Access and integration middleware is composed of verification access component and heterogeneous database data integration method. The verification and access component reconstructs the protocol interaction process based on MySQL protocol, and encrypts and decrypts the encrypted data in the public cloud database through the existing hybrid database scheme, so as to realize the client's verification and access to the encrypted data in the cloud database. The heterogeneous database data integration method unifies the data from different types of databases, and then integrates them into the same database, and uses this database for query operation to realize cross database query. This paper first analyzes the hybrid database system model, then designs from the aspects of MySQL protocol analysis, verification access, data integration, and finally analyzes the access and integration middleware from the aspects of functionality and security.

1. Introduction
With the increase of data scale, more and more enterprises and institutions begin to use cloud computing to store and manage their own data in order to reduce the local storage burden and management overhead of data. However, storing data on the public cloud will face a series of security problems, such as privacy leakage, data theft and other. Because the current cloud computing services are mainly provided by some private enterprises or institutions, besides the ownership and management of data in the cloud are separated, the data in the public cloud is no secret for administrators and service providers [1]. The existing solution is to build a hybrid database system, encrypting the data and then storing it on the public cloud to ensure the security of data. Data will become meaningless and unreadable gibberish after confusion, replacement and other operations in encryption algorithm, so that even the administrator or service provider of cloud service can not obtain effective information through ciphertext, thus ensuring the security of data [2].

In order to build a hybrid database scheme, scholars and research teams at home and abroad have carried out extensive research. In 2011, the research team of Massachusetts Institute of Technology (MIT) proposed CryptDB [3] project at SOSP. CryptDB realizes the encrypted storage of data. When the user queries, it can return a plaintext query result set to the user without decrypting the stored ciphertext data. CryptDB system is mainly divided into three parts: client, agent and server. In CryptDB system, the encryption and decryption of data is carried out on the agent side. Firstly, the
Structured Query Language (SQL) is encrypted on the agent side. Specifically, some field names and table names are replaced with the corresponding ciphertext. The encrypted SQL can still maintain the original syntax requirements. Then, the encrypted SQL is sent to the server side, and the server side will return the ciphertext data to the agent side, which would decrypt the data and returns it to the client [4]. However, CryptDB only supports MySQL database. If ciphertext data is stored in other types of database systems, such as MSSQL and Oracle, CryptDB cannot provide services, and CryptDB does not support cross database query in heterogeneous database environment [5].

In order to solve the problem that the existing hybrid database solutions can not simultaneously realize the client's verification and access function to the encrypted data in the public cloud database and the cross database query function in the heterogeneous database environment, this paper proposes a heterogeneous hybrid database access and integration scheme in the hybrid cloud environment with a combination of the existing research results at home and abroad. In this scheme, an access and integration middleware is used to realize the client's verification access function and cross database query function to heterogeneous hybrid databases in the public cloud. At the same time, an automatic load balancing component is used to distribute client access to improve the processing capacity of middleware and dynamically adjust cluster computing resources.

2. Hybrid Database System Model

The system model of the existing hybrid database scheme includes three entities, namely client, hybrid database system and public cloud database, as shown in Figure 1.

![Figure 1 Hybrid database system model](image)

As shown in Figure 1, users often choose different types of database systems to store and manage data according to their own needs, such as MySQL, MSSQL and Oracle. However, the existing hybrid database solutions can not simultaneously realize the client's verification and access function to the encrypted data in the public cloud database and the cross database query function in the heterogeneous database environment [6]. To solve this problem, this paper designs an access and integration middleware to realize the client's verification and access function to the heterogeneous hybrid database in the public cloud and the cross database query function. And it is deployed at the same end with the client and hybrid database system. The access and integration middleware consists of verification access component and heterogeneous database data integration method. The verification access component is responsible for the verification access function of the client to the encrypted data in the public cloud database. The heterogeneous database data integration method can integrate the data from the heterogeneous database and realize the cross database query function.
2.1 Workflow
The workflow of verification access component is shown in Figure 2.

![Workflow diagram](image)

**Figure 2 Workflow of verification access component**

The details are as follows:
1. **Establish a client connection.**
   Verify the access component and creates the connection pool $P_p$ of the cloud database and the connection pool $P_{cache}$ of the MySQL database $DB_{cache}$ for caching data.
   The verification access component creates the ServerSocket based on TCP protocol, monitoring the specified port and receiving the connection request. When receiving the connection request from the client, the socket connection $S_{client}$ between server and client is created. At the same time, when there is a new client connection, a new socket connection $S_{DB}$ with $DB_{cache}$ is created [7].

2. **User authentication.**
   Verify that the access component receives the MySQL handshake initialization packet from the connection $S_{url}$ and forwards it to the client by connecting $S_{client}$;

3. **Resolve client command requests.**
   After the user passes the authentication, the client can send a command request to the authentication access component. The verification access component receives the MySQL Command Packet from $S_{client}$, parses the data packet, and obtains the client command request $C$ from the command field of the packet[8];
   The verification access component parses the command $C$ to obtain the database tag $L_d$ and the data table $L_t$ to query [9].

4. **Process client command requests.**
   According to the library mark $L_d$ obtained by parsing command $C$, the verification access component determines whether $C$ is operating on a single database or multiple databases. When the number of $L_d$ does not exceed one ($L_d \leq 1$), it is a single database query, otherwise it is a cross...
database query [10]. If it is a cross database query, it needs to call heterogeneous database data integration method for data integration processing;

(5) Returns the processing result to the client.
Verify that the access component judges the type of command C. If C belongs to the command type that needs to return the query result set, the third step is executed; otherwise, the second step is executed;

(6) Clean up the cache data table.
After the query, delete the cache data table Tcache in the cache database DBcache.

2.2 Load Optimization of Access Cluster
Definition: S represents the current load state of the service cluster, with idle, overload and normal states, and T represents tasks. The load of a cluster is the average number of tasks $Ave_T$ waiting to be processed in each task processing node in the cluster [11].

Theorem: in a stable system, the average number of users $Ave_U$ is equal to the effective arrival rate $\lambda_U$ of users multiplied by the average waiting time $t_{uw}$ of users, that is, $Ave_U$ can be expressed by the following formula:

$$
Ave_U = \lambda_U \cdot t_{uw} \tag{1}
$$

Proposition 4.1 the automatic load balancing component designed in this paper can realize the load balancing function, and can automatically adjust the computing resources of the service cluster, so that the overall load of the cluster is in a reasonable range, and the average load of all task processing nodes in the cluster is controlled between idle threshold $Th_{idle}$ and overload threshold $Th_{over}$ [12]. It is proved that $t_T$ represents the time needed to process a task, $\lambda_T$ represents the task arrival rate, and $R$ represents the task processing efficiency of a task processing node. The results are as follows:

$$
R = \frac{1}{t_T} \tag{2}
$$

Then the average waiting time $t_{uw}$ of task T is expressed as:

$$
\frac{1}{k_n} = \frac{1}{R - \lambda_T} \cdot \frac{1}{1 - r_T \cdot R - \lambda_T} = \frac{t_T}{1 - r_T} - t_T \tag{3}
$$

According to theorem 4.1, the average number of tasks $Ave_T$ waiting to be processed in the cluster $Ave_T$ satisfies the following formula:

$$
Ave_T = \lambda_T = \lambda_T = t_T \cdot t_T - \lambda_T \cdot t_T, m \tag{4}
$$

The following is an analysis of the status of the cluster processing tasks under three different load states:

(1) $S = idle$. Let $Sum_T$ denote the total number of tasks, and these tasks need to be completed within a given time interval $\lambda_T$, $N$ denote the number of current task processing nodes in the cluster, if the following formula is satisfied, then the cluster is idle:

$$
Ave_T = \frac{Sum_T}{N} \leq Th_{idle} \tag{5}
$$

When the cluster is idle, it satisfies

$$
0 \leq \frac{Sum_T}{N} \leq \alpha \cdot \Delta t (0 < \alpha < 1) \tag{6}
$$

Let $a = \frac{1}{2}$, which means when $\frac{Sum_T}{N} = \frac{1}{2}$, the cluster just reaches the idle threshold, it can thus be concluded that
$$Th_i = \frac{\text{Sum}_{th_i}}{N} = \frac{\lambda f}{2t_f}$$  \hspace{1cm} (7)

In this case, it can be concluded from formula (4) that:

$$\text{Ave}_r = 2 \cdot \frac{\Delta t}{2Th_i} \cdot 1 - \lambda_i \cdot \Delta t / 2T_i$$  \hspace{1cm} (8)

(2) $S = \text{overload}$

If the following formula is satisfied, the cluster is overloaded:

$$\text{Ave}_r = \frac{\text{Sum}_{th_i}}{N} \geq T_r$$  \hspace{1cm} (9)

When the cluster is in overload state

$$\text{Sum}_{th_i} \cdot t_i \geq \alpha \cdot \Delta t (\alpha > 1)$$  \hspace{1cm} (10)

In other words, when the system \( \text{Sum}_{th_i} \cdot t_i \) just reaches the overload threshold, it can thus be concluded that: at this time, within formula (4):

$$Th_{ow} = \frac{\text{Sum}_{th_i}}{N} = \frac{\alpha \cdot \Delta t}{t_r}$$  \hspace{1cm} (11)

Based on the automatic load balancing component, when the cluster just meets the condition of overload state, it needs to open \( N_{new} \) new task processing nodes to solve the problem of system overload

$$N_{new} = N$$  \hspace{1cm} (12)

Thus:

$$\text{Ave}_r = \frac{1}{2} Th_{ow}$$  \hspace{1cm} (13)

(2) $S = \text{normal}$

If the following formula is satisfied, the cluster is in normal state:

$$Th_{idle} < \text{Ave}_r = \frac{\text{Sum}_{th_i}}{N} < Th_{ow}$$  \hspace{1cm} (14)

In this case:

$$\alpha \cdot \Delta t < \frac{\text{Sum}_{th_i} \cdot t_i}{N} \leq \Delta t (0 < \alpha < 1)$$  \hspace{1cm} (15)

3. Testing and Analysis

3.1 Test Environment

The cloud computing platform involved in the test in this paper adopts OpenStack cloud computing management platform. The main hardware environment in the test includes client PC and the infrastructure of building OpenStack cloud platform. The OpenStack cloud platform is composed of control node, computing node, network node and storage node [13]. The detailed configuration parameters of each equipment in the test are shown in Table 1.
### Table 1 Test hardware configuration information

| Numbering | Equipment name   | CPU core | RAM   | hard disk |
|-----------|-----------------|----------|-------|-----------|
| 1         | Control node    | 6        | 32GB  | 1T        |
| 2         | Calculate node  | 6        | 32GB  | 2T        |
| 3         | Network node    | 12       | 32GB  | 5T        |
| 4         | Storage node    | 12       | 16GB  | 1T        |
| 5         | Client          | 4        | 8GB   | 1T        |

The software environment involved in this test is shown in Table 2.

### Table 2 Test software environment

| Numbering | Name of software   | Types of software                  | Version | Description                                                                 |
|-----------|--------------------|------------------------------------|---------|-----------------------------------------------------------------------------|
| 1         | OpenStack          | Cloud computing management platform | Juno    | Open source cloud computing management platform                             |
| 2         | Windows            | operating system                   | 10      | Operating system of the client PC                                           |
| 3         | Ubuntu             | operating system                   | 12.07   | Virtual machine image used for testing                                     |
| 4         | Ubuntu Server Cloud| operating system                   | 12.07   | Operating system task processing node virtual machine image                 |
| 5         | Encrypted Database System | Encrypted database system | 2.2     |                                                                             |
| 6         | MySQL              | Database Management System          | 5.5.53  | Open source relational database                                             |
| 7         | MSSQL              | Database Management System          | 2019R2  | Microsoft SQL Server database                                               |

### 3.2 Verification Access and Data Integration Test

The test results are shown in Figure 3. Firstly, the test results show that the access and integration middleware has realized the verification access to different types of hybrid databases. Secondly, the test results show the time cost of the middleware to query different amounts of data in MySQL and MSSQL.
The following conclusions can be drawn from Figure 3:

With the increase of the amount of data, the query time is getting longer and longer. Because there is the need to realize the client's verification access to encrypted data, the middleware needs to call the hybrid database system to decrypt the encrypted data. The larger the amount of data is, the longer the decryption time becomes, and the larger the amount of data is, the larger the query result set forwarded by the middleware to the client becomes, the contract time is thus getting longer.

3.3 Load Balancing Test

In order to evaluate the performance of the automatic load balancing component, this paper designs a test to evaluate the processing performance of the component under different loads and verify its detailed running status. The specific test scheme is to simulate 1000, 2000, 5000 and 10000 users respectively in the service cluster (hereinafter referred to as the automatic load balancing cluster) with the application of the automatic load balancing component, and use the access and integration middleware to conduct a table matching test mentioned in the cross database query test_a4 and test_b4 to test the time required for the access and integration middleware to process all the query requests after applying the automatic load balancing component, and compare with the ordinary service cluster (hereinafter referred to as the ordinary cluster) without using the automatic load balancing component. The number of initial task processing nodes of both clusters are set to 3. The test results are shown in Figure 4.
As shown in Figure 4 (a), the maximum number of task processing nodes opened by the automatic load balancing cluster when processing high concurrency requests will increase with the increasing number of tasks.

As shown in Figure 4 (b), the processing time of automatic load balancing cluster is shorter than that of ordinary cluster in processing user requests of the same size, which indicates that the processing performance of automatic load balancing cluster is better than that of ordinary cluster. And the greater the load pressure is, the greater the processing performance gap becomes. After using the automatic load balancing component, the service cluster takes less time to process large-scale tasks, and increases slowly with the increase of tasks.

Through the test, we can know that the automatic load balancing component designed in this paper has good performance in task processing when dealing with large-scale tasks and high concurrent requests. The combination of automatic load balancing component and access and integration middleware will improve the query efficiency of middleware, especially the performance of concurrent query.

4. Conclusion

By deploying a private OpenStack cloud computing platform, the database system is deployed on some virtual machines as a simulation of public cloud database. Firstly, the access and integration middleware is tested from two aspects: single database query and cross database query. The test results show that the access and integration middleware realizes the verification access to hybrid database, and can perform cross database query operation in heterogeneous hybrid database environment. Then the automatic load balancing component is tested. The results show that the automatic load balancing component has good performance in dealing with large-scale tasks and high concurrent requests. The combination of automatic load balancing component and access and integration middleware improves the query efficiency of middleware, especially for concurrent queries.

References

[1] Mohiuddin I, Almogren A. Workload aware VM consolidation method in edge/cloud computing forIoT applications[J]. Journal of Parallel and Distributed Computing, 2019, 123: 204-214.

[2] Carneiro T, Da Nóbrega R V M, Nepomuceno T, et al. Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications[J]. IEEE Access, 2018, 6: 61677-61685.

[3] Kumar L, Mutanga O. Google Earth Engine Applications Since Inception: Usage, Trends, andPotential[J]. Remote Sensing, 2018, 10(10): 1509.
[4] O’Brien P, WH Young S, Arlitsch K, et al. Protecting privacy on the web: A study of HTTPS and Google Analytics implementation in academic library websites [J]. Online Information Review, 2018, 42(6): 734-751.

[5] Picado J, Lang W, Thayer E C. Survivability of Cloud Databases-Factors and Prediction [C] //Proceedings of the 2018 International Conference on Management of Data. ACM, 2018: 811-823.

[6] Wang J, Chen X, Li J, et al. Towards achieving flexible and verifiable search for outsourced database in cloud computing [J]. Future Generation Computer Systems, 2017, 67: 266-275.

[7] Paladi N, Gehrmann C, Michalas A. Providing user security guarantees in public infrastructure clouds [J]. IEEE Transactions on Cloud Computing, 2017, 5(3): 405-419.

[8] Zhang Y, Xu C, Liang X, et al. Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation [J]. IEEE Transactions on Information Forensics and Security, 2017, 12(3): 676-688.

[9] Wang G, Liu C, Dong Y, et al. IDCrypt: A Multi-User Searchable Symmetric Encryption Scheme for Cloud Applications [J]. IEEE Access, 2018, 6: 2908-2921.

[10] Tiwari D, Gangadharan G R. SecCloudSharing: Secure data sharing in public cloud using ciphertext-policy attribute-based proxy re-encryption with revocation [J]. International Journal of Communication Systems, 2018, 31(5): e3494.

[11] Wang X A, Xhafa F, Ma J, et al. PRE+: dual of proxy re-encryption for secure cloud datasharing service [J]. International journal of web and grid services, 2018, 14(1): 44-69.

[12] Shen X J, Chang Q, Liu L, et al. CCLBR: Congestion control-based load balanced routing in unstructured P2P systems [J]. IEEE Systems Journal, 2018, 12(1): 802-813.

[13] Arunarani A R, Manjula D, Sugumaran V. Task scheduling techniques in cloud computing: A literature survey [J]. Future Generation Computer Systems, 2019, 91: 407-415.