Architecture and CRUD Complete Analysis and Algorithm of the Integration of Distributed Database Access in the Same Configuration

Qing Chen, Jing Liang, Leyi Gou and Zhousen Zhu

ABSTRACT

A distributed database with the same configuration is an important storage scheme for the extensible data of an information system. This storage scheme can effectively increase the access and storage capacity of the system and improve its response speed. However, the response speed of the traditional distributed database in the same configuration will gradually slow down when the traffic of the system increases or the access concurrency of the system expands. This article adopts the Mycat middleware and the form of Oracle database node and builds a new type of distributed database with the same configuration on the server side. It develops a dynamic switch and parser for a multiple data source, to parse and route CRUD operations in order to realize the dynamic connection control of the Mycat server and multiple Oracle database nodes and the effective distribution of data resources. Through analyzing the shard field of the CRUD statements, the integrated parser can implement the operation of the database by involving only a single or a few nodes without the Mycat server and communicate directly with the Oracle database of corresponding nodes. It can also implement the operation of the database by involving several or all of the database nodes to communicate.
and route with the Mycat server. This scheme can effectively reduce the operating pressure of distributed data source servers, reduce the execution time of CRUD operations, and greatly improve access speeds during large concurrent visits. At present, this achievement has been successfully applied to the cloud service of a large project with complex computing.

1. INTRODUCTION

In this paper, a distributed database of the same configuration with Oracle as the storage node is built on the server side using Mycat agent sharding. The data tables are divided and stored in different Oracle databases. In this form of distributed database, CRUD operations at the application layer need to be routed to the database storage node through the Mycat agent layer. With increasing system visits, in order to reduce the operating pressure of the Mycat server, the query process needs to be optimized to shorten the response time of CRUD operations [3]-[6]. Therefore, an integrated CRUD control and central converter based on the same configuration distributed database system was developed to analyze and route the CRUD operations in the application layer. The process of directly connecting and communicating with Oracle nodes for CRUD operations involves only a single database, without going through the Mycat server. Through testing, the parser can effectively reduce the execution time of CRUD operations and reduce the running pressure of the distributed data source server.

2. DISTRIBUTED DATABASE CRUD BIDIRECTIONAL COMPLETE ANALYSIS AND CENTRAL CONVERTER TECHNOLOGY

2.1 SQL Classification with Sharding Keywords

The same-configuration distributed database in this paper is dynamically allocated based on the database input from users. The conditional statement of the CRUD operation must contain the sharding field "zoneid". After SQL classification, different CRUD statements are analyzed to obtain the sharding key corresponding to the sharding field. The sharding rules established in this paper are used to obtain the connection information of the corresponding data source. The dynamic data source switching and central converter technologies explained in the next section are used to communicate with the corresponding data source connection.
2.2 Central Converter Technology

The core contents of the central converter in this paper are the fusion, the parsing, and the extraction of sharding keywords of CRUD operation statements. The core contents are as follows:

2.2.1 SQL CLASSIFICATION AND FUSION

Before extracting keywords, SQL statements and parameter strings from the operator need to be combined. This requires the following implementation and procedures:

1. Due to the different structures of SQL statements, the incoming SQL statements (containing placeholders) are divided into insert statements and non-insert statements using formula (1).

2. For insert statements, use the $I_i$ method. Put the value of the parameter string $param_i$ into the corresponding $sql_i$ to get the combined statement $JD_{sql_i}$. For non-insert statements, use the $FI_i$ method. Put the value of the parameter string $param_i$ into the parameter placeholder in $sql_i$ to get the combined statement $HD_{sql_i}$.

3. For the statement $JD_{sql_i}$, use the $Z_i$ method in formula (3) to obtain the corresponding value $I_z$ with "zoneid" from the parameter array. For the statement $HD_{sql_i}$, use the $FZ_i$ method in formula (4) to obtain its value from multiple "zoneid" parameter equations, and obtain $[]$ after de-weight.

$$JD_{sql_i} = \sum_{t=0}^{L_{param_i}} I_i(sql_i, param_i) \quad (1)$$

$$HD_{sql_i} = \sum_{n=0}^{L_{sql_i}} \sum_{t=0}^{L_{param_i}} F_i(sql_i, param_i) \quad (2)$$

$$I_z = Z_i(JD_{sql_i}) \quad (3)$$

$$S_z[] = FZ_i(HD_{sql_i}) \quad (4)$$

2.2.2 EXTRACTION OF SHARDING KEYWORDS

For insert statements, $I_z$ obtained from formula (3) extracts the value of the shard field "zoneid" from $I_z$. After security filtering, it is the shard key for this operation.

For non-insert statements, an array of $[]$ is obtained from formula (4). If the length of $[]$ is 1, $sql_i$ only contains a single or a few of the same shard
keys. This shows that the operation involves a single or a few database nodes and direct access to the shard key occurs. If the length of \( [\ ] \) is greater than 1, \textit{sqli} contains multiple shard keywords. This shows that the operation involves multiple database nodes, and the shard key value corresponding keyword of Mycat is “0”.

### 2.3 Database Connections of CRUD Operations

For the two types of CRUD statements mentioned in 2.2, database connections are conducted respectively. Insert statements, according to the obtained shard key, establish the data source for the current thread alias and then retrieve the corresponding data source connection. Unlike non-insert statements, insert statements involve only a single database, so insert operations connect and communicate with the corresponding data source nodes. Non-insert statements, according to the obtained shard key, set the data source alias of the current thread to obtain the corresponding data source connection.

### 3. TEST AND RESULT ANALYSIS

Methods of testing in this paper include: uploading Excel files on the front-end interface and sending them to the back-end to break up into separate items; inserting and updating data for different orders of magnitude after analyzing the central converter (based on the Spring boot framework) for parameter strings and SQL statements; parsing SQL and parameter strings through the central parser in the background (based on the Spring boot framework) and then querying and deleting data for different orders of magnitude.

#### 3.1 Test Results and Analysis

Multiple tests for data of different orders of magnitude were conducted. The test results with large fluctuations were removed and the remaining results were averaged. The test results are shown in Figure 1:
In this test, 10, 100, 1000, and 10000 pieces of data were inserted and updated, while 1000, 10000, 100000, and 100000 pieces of data were queried and deleted respectively. Each type of operation is divided into two cases (as follows) for comparative testing: communicating with the corresponding data source directly after being parsed by the central converter, and communicating with the Mycat server directly without going through a central converter.

3.2 The Analysis

According to the operation results, the following conclusions were obtained through analysis:

(1) According to Figure 3.1, data insert and update operation times with a central converter are shorter than those without a central converter, indicating that central converter parsing can reduce the run time of data insertion and update operations. With the increase of data volume, the run times of insert and update operations are longer, and the run times of the central converter are reduced.

(2) According to Figure 3.1, data query and deletion operation times analyzed by the central converter are shorter than those without the central converter, indicating that the operation time of query and deletion operations can be reduced by the analysis of the central converter. With the increase of data volume, the longer the run times of query and deletion operations, the longer the run times of the central converter can be reduced.
(3) Most query and deletion operations only involve a single database, which is directly connected with the corresponding database node after passing through the central converter, and does not pass through the Mycat server. This allows all CRUD operations to avoid passing through the Mycat server and effectively relieves the operation pressure of the Mycat server.

In conclusion, the central converter described in this paper can effectively relieve the running pressure of the Mycat server and improve the running speed of CRUD operations.

4. CONCLUSIONS

In this paper, using the Mycat middleware and Oracle database node, the distributed database with the same configuration is built on the server side. The dynamic switch of multiple data sources and central parser are also added in the application layer of the system to analyze and route the CRUD operations of the operator. This scheme can reduce the running pressure of the Mycat server, reduce the execution time of CRUD operations, and effectively improve the access speed in the case of large visits.

ACKNOWLEDGEMENTS

Fund Projects Participated in the Project: Science and Technology Key Research and Development Project of Sichuan Science and Technology Department (NO.2018GZ0270); National Natural Science Foundation of China (NO.11575121)

REFERENCES


