Journal of Computing and Electronic Information Management

ISSN: 2413-1660

Research on synchronization method of distributed data stream in Internet of things

Huadong Wang*, Dongliang Zhang
School of Computer Science and Technology, Zhoukou Normal University, Zhoukou 466001, China

Abstract: In order to improve the synchronization efficiency and accuracy of distributed data stream, a synchronization method of distributed data stream in Internet of things is designed. First of all, the distributed data capture mainly includes control table structure design, synchronous "oscillation" prevention and trigger structure design. Then, on the basis of distributed data conversion and multi thread breakpoint transmission, this paper proposes a solution of concurrent synchronous data of one-to-multi nodes. Finally, the synchronization process of distributed data stream is proposed to realize the synchronization of distributed data stream in the Internet of things environment. The experimental results show that the synchronization method of distributed data stream in the Internet of things environment effectively improves the efficiency of data stream synchronization, and improves the accuracy of data stream synchronization.

Keywords: Internet of things environment; distributed data stream; synchronization method; multi thread breakpoint transmission.

1. Introduction

In the current information age, with the rapid development and increasing popularity of information technology, the application scope of database is becoming wider and wider, and the technology is becoming more and more mature, which plays an important role in all walks of life. In the environment of Internet of things, the important information of enterprises and institutions is obtained and transmitted. However, due to the actual situation of different units, the software and hardware environment is not the same [1]. Many relatively independent information service and management systems have been produced. They are cross regional and heterogeneous. However, if they are integrated, they will be a unified whole. With the development of economy and science and technology, it is necessary to integrate various heterogeneous data streams, which is not only conducive to unified management, but also conducive to access to information, realize resource sharing, and promote the scientific development among units [2]. Therefore, it is necessary to establish a data stream synchronization method on the existing basis, so that it can synchronize with the data of each branch, and when the data of each branch changes, the central database will also change. This is not only conducive to the management department to grasp the dynamic information, carry out decision-making analysis, improve the competitive advantage of enterprises, but also conducive to the government departments to improve the adaptability. In addition to the above functions, synchronization of distributed data stream can also back up the data, which is conducive to protecting the property of the state, enterprises and people. How to achieve data synchronization between heterogeneous databases safely, efficiently and quickly has become a research hotspot in various fields and an important direction of information technology development [3-4].

According to the above requirements, a synchronization method of distributed data stream in the Internet of things environment is designed and implemented to ensure the real-time data synchronization between professional database and shared, basic and spatial databases.

2. Acquisition of distributed data

In the Internet of things environment, there are many distributed data, so it is necessary to acquire the distributed data before data stream synchronization. If we design a method to acquire the change data of each database in the system, it will increase the complexity of the system. Therefore, this paper proposes a data acquisition method for most databases [5]. The current mainstream databases support triggers. Trigger is a kind of database mechanism, which can automatically acquire data operations. On this basis, a change data acquire method based on trigger, control table and time stamp TCT is proposed.

The basic idea of this method is to create add, delete and modify triggers for the tables that need to be synchronized, and to create a control table with timestamp field for heterogeneous databases [6]. When the data in the synchronization table changes, it needs to activate the corresponding trigger, store the extracted control information in the control table, and store the current time of the recording system in the timestamp field, so as to realize the distributed data capture.

2.1 Structure design of control table

In order to obtain data easily, a control table is established for all heterogeneous databases in the system to store some control information of the changed data, including the time stamp of the changed data, so as to change the name of the table where the data is located, the primary key value of the data, and the data operation [7]. When DML operation is performed in the source database, the trigger in the source database detects the change of data and saves the changed data to the control table [8].

The information of the change table is shown in Table 1.

Table 1 Information of Change table

Field name	Field type	Description
Id	Int	Change data number, primary key
C_Time	Datetime	Data change time
C_Field	Nvarchar(500)	Change field name
C_Table	Nvarchar(50)	Change the name of the table where the data is located
C_Key	Nvarchar(50)	Change the primary key value of data
C_Operate	Nvarchar(10)	Change data operation
C_Flag	Int	Synchronization identifier (0 / 1)

2.2 Prevention of synchronous "oscillation"

When data capture is synchronized, synchronization "oscillations" usually occur. In one-way synchronization, there is no trigger set in the target database, so there is no oscillation during synchronization. Both the source database and the target database have two-way synchronization trigger mechanism. When the SQL statements extracted from the synchronization operation are executed in the target database, the trigger will be awakened, and then the executed control information will be stored in the control table [9]. When the data in the target database needs to be synchronized with other databases, the control information will be extracted and used as synchronous data to continue synchronization, which will lead to "oscillation" phenomenon in the synchronization system and increase the system overhead [10].

In order to make the trigger recognize the source of DML operation, we assign a synchronization data administrator for each synchronization system, who is responsible for updating the synchronization data to the database. Trigger determines whether the current user is an administrator before execution. If yes, the trigger will not be triggered, otherwise it will be triggered. Table 2 shows the functions of determining the current operations administrator:

Table 2 Acquisition of the functions of the current user

Serial number	Database name	To getting the function of the current user
1	SQL Server 2008	Suser_name()
2	My SQL	Select user()
3	SQL server 2000	Suser_name()
4	Oracle	Select sys_context(„userenv“,“session_user“) from dual

2.3 Trigger structure design and Implementation

Firstly, an add trigger is created for the synchronization table. When adding to the synchronization table, the trigger first determines the current operation user. If it is Synchronizer Administrator, the trigger will not be triggered; otherwise, it checks whether the unsynchronized data in the control table (i.e. C_Flag=0) is the same as the primary key value of the current operation data [11-12]. In this case, it means that the inserted primary key value is not unique and the program ends. If it is not, the Insert statement is executed in the trigger to store the control information in the control table. The process is shown in Figure 1.

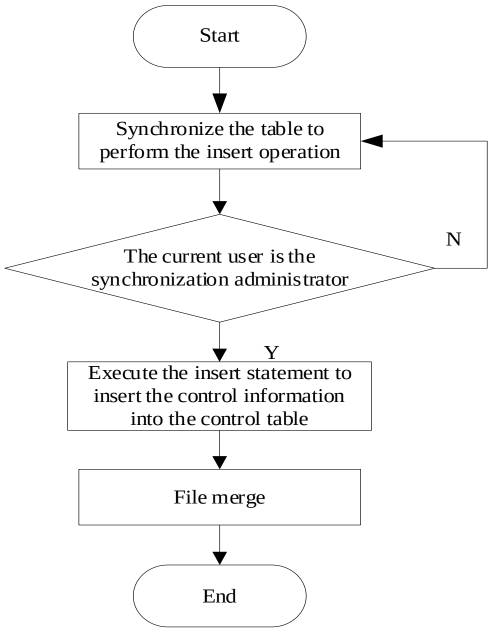
Start
Synchronize the table to perform the insert operation
The current user is the synchronization administrator
N
Y
Execute the insert statement to insert the control information into the control table
File merge
End

Figure 1 Flow chart of adding trigger

Secondly, a delete trigger is created for synchronous tables. When the synchronization table is deleted, the trigger program first determines the current action user. If it is Sync Administrator, the trigger will not be triggered; otherwise, when traversing the unsynchronized data in the control table, if the primary key value of the C\ keys field is different from the current operation data, and there is no operation record of this

data in the control table, the control information will be inserted into the control table. In the same case, the control table has the operation of the current data, but not synchronization. So for different C_Operate, there are several types of operations:
(1) If C_action type is insert, it indicates that the current data is newly inserted before synchronization, and the data will not be synchronized to other databases [13]. Then, it needs to delete other records directly from the control table.
(2) If C_operation type is update, it needs to modify the C\ u Time of C_Operate field of the data to be deleted, change it to the current running time, delete the data in the C_field, remove the data and insert the removed control information. Otherwise, it needs to remove the control information [14].
Thirdly, the modification trigger of synchronous table is created. When modifying a synchronization table, the trigger first determines whether the current action user is a synchronization administrator. If so, it will not be triggered; otherwise, there is no synchronized data in the traversal control table. If the primary key value of C_ key field is different from the current operation data and there is no operation record of the data in the control table, the control information is inserted into the control table [15]. In the same case, the current data in the control table also has operations, but not synchronization.

3. Distributed data conversion

The transformation process of distributed data mainly includes two parts: the transformation from source database to XML and the transformation from XML to target central database. Figure 2 illustrates the data transformation based on XML.

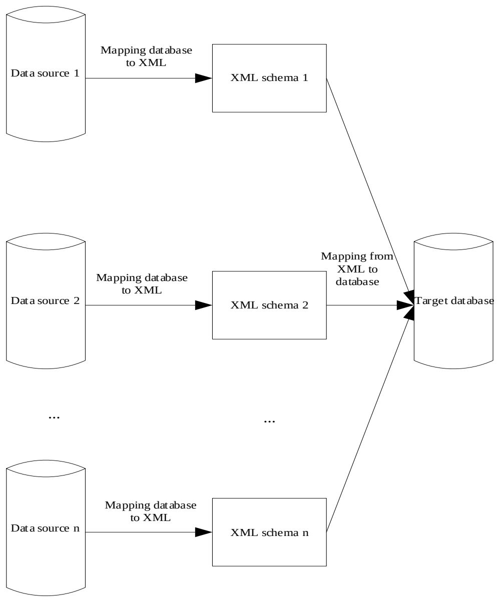

Figure 2 Schematic diagram of data conversion based on XML

The conversion steps are as follows:
Step 1: read data from the source database by using XML file as the intermediary,

convert the data according to the mapping relationship between local data type and intermediate data type, and convert the heterogeneous data source data into XML standard file according to the defined mapping relationship.
Step 2: transfer the data to the destination synchronously through the XML channel.
Step 3: on the target side, read and analyze the XML file, convert the obtained data from the local data type to the intermediate type, and then add the converted data to the target database [16].
Through the above data conversion, aiming at the problems of data conversion failure and inconsistent data between the sender and the receiver in the process of database synchronization, a three-times data packaging error correction method is proposed [17].
The steps are as follows:
Step 1: Check whether there are XML packets that cannot be synchronized through the coordination monitoring program;
Step 2: Parse the XML package that failed to synchronize, and get the primary key value of the specific data contained in the package
Step 3: When the third SQL statement is used, the latest synchronization value of the same SQL statement will be obtained again, and the failed database data will be obtained again [18-19];
Step 4: Record the newly encapsulated XML package, synchronize the database and delete the failed XML package;
Step 5: Sending server sends the packaged XML package again;
Step 6: The receiving server obtains the new XML data packet and writes the unpacked data to the receiving database to complete the distributed data conversion.

4. Multi-threading breakpoint continuation

When generating breakpoints, the most commonly used technology is to transmit packets from the source node to the target node. Multi-threading breakpoint continuation data has the following advantages: Thread technology can make full use of system resources for efficient packet transmission; breakpoint continuation technology can save network bandwidth, shorten transmission time and improve transmission efficiency when the network is unstable [20]. The principle of multithreading breakpoint recovery is shown in Figure 3.

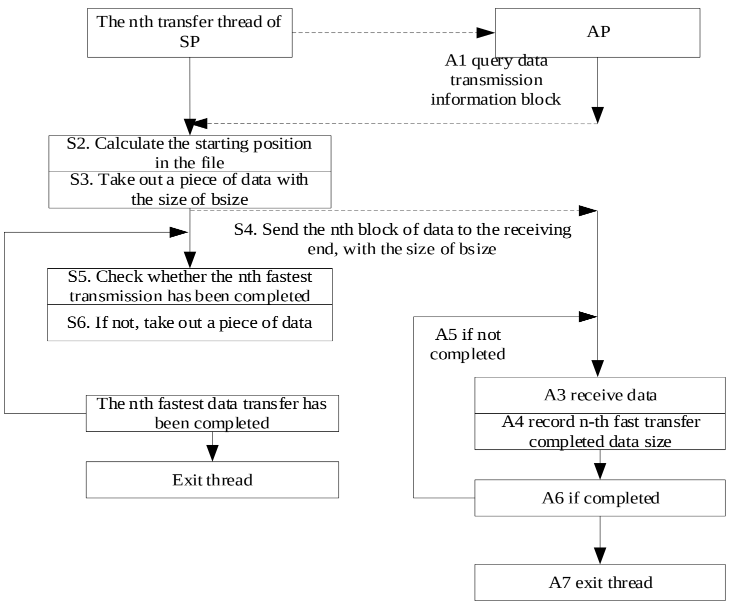
The nth transfer thread of SP
AP
A1 query data transmission information block
S2. Calculate the starting position in the file
S3. Take out a piece of data with the size of bsize
S4. Send the nth block of data to the receiving end, with the size of bsize
S5. Check whether the nth fastest transmission has been completed
S6. If not, take out a piece of data
A5 if not completed
A3 receive data
A4 record n-th fast transfer completed data size
The nth fastest data transfer has been completed
Exit thread
A6 if completed
A7 exit thread

Figure 3 Principle of multi-threading breakpoint continuation

In the figure above, SP represents the sender, AP represents the receiver, Sn (n is a positive integer) represents the step size of the sender, and An (n is a positive integer) represents the step size of the receiver. In SP, the file is divided into n blocks of the same size, and N threads are created to connect AP. After receiving each connection request, AP feeds it back to SP for data transmission. If SP receives a message that can transfer files, it will send an information block (including the starting position of the block) to AP to request data transfer. After receiving the request, AP sends a data transmission block to SP, SP transfers the data specified in the information block to AP, AP and SP thread exit, AP records the breakpoint log when it interrupts, and prepares for further transmission.

There are two ways to save the file content and breakpoint information when the breakpoint continues to be executed: 1. Save the file content and breakpoint information respectively to generate two temporary files; 2. Write the offset of the file to the temporary file of breakpoint information to keep the file content consistent and synchronized with the breakpoint information every time the data block transferred. In order to better maintain the consistency and synchronization of file content and breakpoint information, and avoid errors caused by deletion or loss of one party, the first method is used in this study.

In the process of transmission, the file is partitioned. The less the number of file blocks is, the longer the breakpoint recording time of each file is. If the transmission is interrupted and then transmitted, the longer the breakpoint log recording time is, the shorter the breakpoint log recording time is. In order to improve the transmission efficiency of breakpoint recovery upload, it is necessary to partition files reasonably. Suppose that an interrupt is randomly generated in a transmission process, the file size of this transmission is F, the file transmission speed is constant $V(\mathrm{B/s})$, the time of writing breakpoint log file is constant t, and the size of each file block is divided into X bytes.

At a certain time, the interrupt rate of each byte transmission is:

$$p = \frac{1}{F} \tag{1}$$

If the expected value of repeat transfer size generated by each file block transfer process is E, then:

$$E = p \times \frac{(0 + X - 1) \times X}{2} \tag{2}$$

The total time needed for the whole transmission process is as follows:

$$T = \frac{\frac{F}{X} E + F}{V} + \frac{F}{X} t + \frac{X - 1}{2V} + \frac{F}{X} t \tag{3}$$

The derivative of time T is T'.

$$T' = \frac{1}{2V} - \frac{F \times t}{X^2} \tag{4}$$

The specific business implementation process is shown in Figure 4.

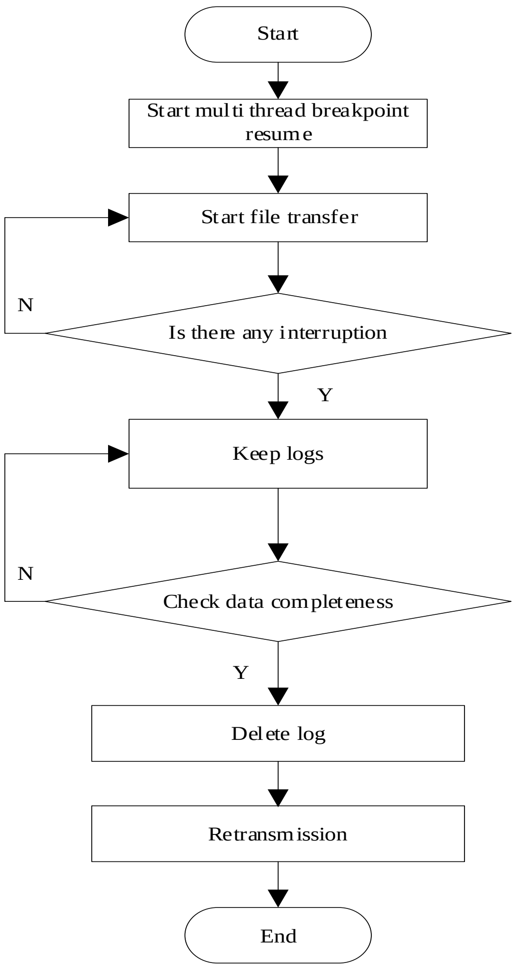

Figure 4 Flow chart of multi-threading breakpoint continuation

5. The solution of synchronous data of one-to-multi nodes

When data is synchronized from one node to multiple nodes at the same time, the same data is usually inserted into the control table and an exception is raised, which violates the primary key constraint of multithreading concurrent operation. If node a synchronizes data to B and C at the same time, node a uses thread 1 to synchronize data with node B, while node a uses thread 2 to synchronize data with node C. The specific implementation process is shown in Figure 5.

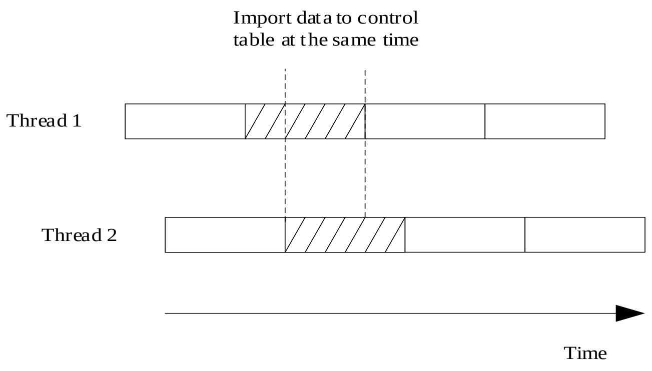

Figure 5 Problems caused by multi-threading concurrent synchronization data

When node a is synchronized with node B and node C, it seems that there is a small time-interval between them. The slash part of thread 1 and thread 2 is the execution process of importing data into the control table, and the overlapping part between them is to insert the data into the control table at the same time.

For this kind of problem, queue is usually used to solve, and the specific operation steps are as follows:

(1) Connect n main threads in the queue to construct the queue structure.

(2) Determine whether the queue contains the main thread.

(3) If there is a main thread in the queue, take one of the main threads out of the queue and start it to synchronize the data.

(4) Start the monitoring thread in the main thread that just left the queue. There is a one-to-one correspondence between the two threads.

The monitoring thread determines whether the operation of the main synchronization thread on the control table has been completed.

(5) If the operation of the control table is not completed, wait for a period of time, and then judge, cycle judgment, until the end of the operation of the control table.

(6) If the operation is completed on the control table, skip to (2) and make cycle execution.

(7) Finally, if it is determined that there is no main thread in the queue, it means that the concurrent synchronization is complete.

6. Implementation of distributed data stream synchronization

6.1 Data consistency detection

In the aspect of data consistency checking, the basic design idea of data consistency checking algorithm is: After the local database system is running, the running status is recorded in the checking database, and then the previous flag bit is modified to the corresponding value according to the corresponding operation of the remote standby database system. Finally, in the verification process, the verification data including false is scanned.

The specific steps of the algorithm are as follows:

Step 1: After the operation of the local master database is completed, it can judge whether the database operation is successful, and construct a check object in the check database. If the operation is successful, the operation record is the primary key, the operation timestamp, and a flag bit, indicating whether the operation is applied to the standby database (the value here is set to false); if the operation fails, it ends directly;

Step 2: After the remote database operation is completed, it can judge whether the database operation is successful. In the case of successful execution, the verification library is quered and compared by primary key. When there is a record in the verification library (if the flag bit is false, it becomes true), it should update the record, but if it does not exist, it should directly insert the record into the verification library (default is false); if the execution fails, it should directly end the update record.

Step 3: Data verification: When all the flag bits are true, the data of the local database is consistent with that of the remote database, and the check ends; when all these records are in the local database, if these records are false, the records will be written to the remote database. No matter whether the insertion or deletion is successful or not, the records with the mark bit of true should be deleted, and the remaining records with the mark bit of false should be retained for verification in the next inspection cycle. The calibration cycle can be set according to the actual production and equipment level.

6.2 Conflict detection and resolution

Because the database will encounter similar deadlock between database transactions or threads in the actual operation, each operation may have multiple operators, so the operation peer will synchronize with multiple different peers (called synchronization point), that is to control the target. Because the communication delay between the peer node and each synchronization point is different, the time for the synchronization request to arrive at the synchronization point is first arrived and then arrived, so we need to deal with this situation as follows:

Step 1: If the operation op1 has only one operand (the same processing as op2), other

operations cannot conflict with it, so the causal operation returned by the synchronization site should be followed directly.
Step 2: When the operation op1 has multiple operands, if the operands returned by the synchronization site are all opi operation op1, I = 2 opi 3, then op1 will not immediately execute the operators brought back in these messages, but will wait for a period of time (this time is determined by the whole network delay).
Step 3: If a message containing opi operation, sent by opi generating node or forwarded by other peer nodes is received during the waiting process, it indicates that there is no conflict in the network. At this time, the site can execute op1.
Step 4: If the waiting time exceeds the preset value, but the message containing opi operation or other peer forwarding sent by the generating node of these opis has not been received, it can be considered that there is a conflict in the network, because the site of these opis may enter the waiting state after receiving the op1 operation opi, until the waiting time exceeds the value.
Step 5: When a message containing opi operation or other peer-to-peer forwarding is received from the generating node in the reason operation opi, if the waiting time exceeds the preset value and the message containing opi operation or other peer-to-peer forwarding has not been received, it will be considered that there is a conflict in the network, because the sites where these opis are located may also enter the waiting state after receiving reason operation op1, the waiting time between them exceeds this value.

6.3 Real-time synchronization of incremental data

After checking and conflict detection, real-time incremental files are extracted to realize data stream synchronization. It is assumed that two computers in different regions are connected through the network, one is on the server side, the other is on the client side. At startup, the file f is the same on both the client and the server. Now some content changes in the client file, such as adding or deleting some content, which are recorded as f_1. The steps taken are to synchronize the files of the remote server and the client.
In the first step, the server divides the file into a series of non-overlapping data blocks in m bytes, totally n blocks. If the last block is less than m bytes, it will be filled with m bytes.
The second step is to calculate the checksums of all data blocks on the server side. The checksums of each data block are represented as rolling check and strong check respectively, which are recorded as $h(x_i, y_i)$ and sent to the client side.
In the third step, the client searches for the file f_1 from the beginning, generates the scrolling checksums and compares them with the checksums x_i generated by the

server. If the strong checksums of two data blocks are equal, the next data block will be compared. It is found that the data block matched with x_i is compared with y_i after calculating the strong check sum. If it is not, the two data blocks are different. The first offset of the data block will be recorded and the comparison will continue until all the offsets are obtained.

Step 4: after the comparison, the server receives different contents from the client and starts to synchronize different parts of the file.

At the same time, but in the actual inspection process, when the file blocks are not equal, the inspection values are equal. Therefore, in order to correspond to this situation, it needs to make a strong comparison after a weak comparison.

The basic strategy is to calculate the 32-bit scroll check value of each block and search the check table of the f file.

The first level uses a 16 bit rollover check (derived from a 32-bit check) and a series of hash values, sorted by a 16 bit rollover check. Each item in the hash table points to the first hash value of the list element. If there is no element in the hash value list, it will contain a null value.

If the hash value of the hash table is not null, the secondary check is started. Level 2 parity includes scanning the sort table pointed by the hash table entry and comparing the current value with the whole 32-bit parity table. If the search reaches the same check value, the third level check is started when the whole 16-bit hash item scan stops.

The third level check includes calculating the strong hash of the current file offset, and comparing the current list item with the strong hash (if the two strong hashes match successfully), it finds the file block in f_1 that is the same as f .

After the successful match, the client will send it to the server where the program block f_1 starts and ends, as well as the index (position) of the data block in f_1. The next check starts from the end of the successful match data block. If the matching fails, the next check starts from the next byte of the starting position of the data block, and the above process is repeated continuously to realize the synchronization of distributed data stream in the Internet of things environment.

7. Experimental comparison

In order to verify the effectiveness of the distributed data stream synchronization method in the Internet of things environment, the experimental analysis is carried out, and the synchronization effect of the two methods is compared with the traditional methods.

The network connection diagram between the data center and each station applied in the experiment is shown in Figure 6.

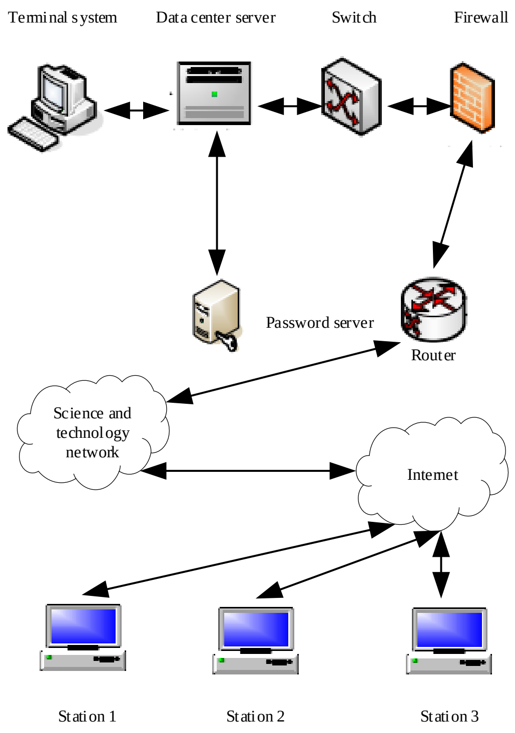

Figure 6 Network connection diagram of data center and stations

In this study, we focus on the real-time synchronization of data files stored in different operating systems to analyze the efficiency of the two methods.

7.1 Comparison of initial synchronization time

After the traditional method and the method in this study are synchronized, the comparison results of synchronous data traffic are shown in Figure 7.

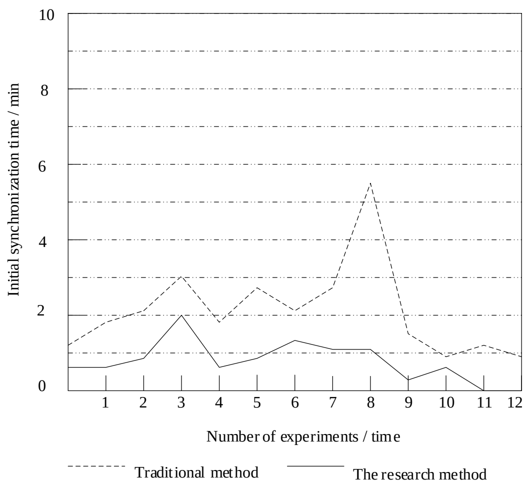

Figure 7 Comparison results of initial synchronization time

Through the analysis of Figure 7, it can be found that the synchronization method in this study is less affected by heterogeneous data in the synchronization process, and the synchronization data can meet the needs of practical application, which is better than the traditional method.

7.2 Comparison of incremental transmission rate

The incremental transmission rate of the proposed method and the traditional synchronization method is analyzed. The results are shown in Figure 8.

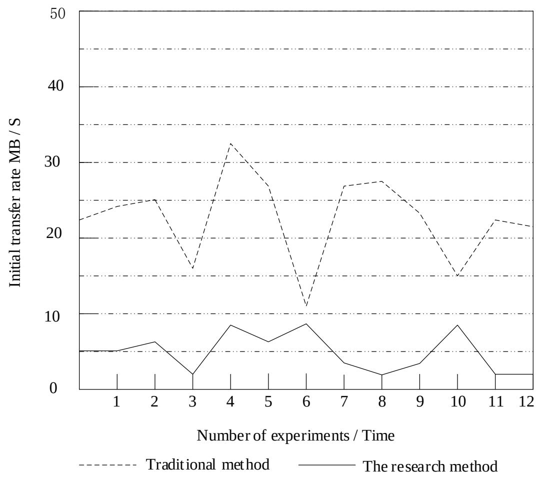

Figure 8 Comparison of incremental transmission rate

According to figure 8, it can be found that the transmission rate of the traditional method in incremental data transmission is slower than that of the method in this study.

7.3 Comparison of average incremental transmission time in everyday

The average time consumption of incremental transmission between the proposed method and the traditional method is compared, and the comparison results are shown in Figure 9.

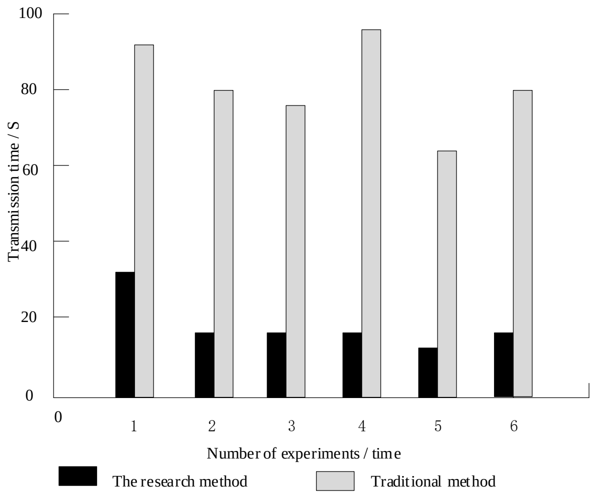

Figure 9 Comparison of average incremental transmission time in everyday

By analyzing Figure 9, it can be found that the average incremental transmission time in everyday of the synchronization method in this study is less than that of the traditional method. It shows that this research method can not only solve the problem of data file synchronization in heterogeneous environment, but also reduce the amount of data transmitted on the network and improve the synchronization efficiency after the initial synchronization of data files.

7.4 Comparison of data synchronization accuracy

When analyzing the accuracy of synchronized data, data synchronization is combined with integrity check to redirect the output to a log file. In this way, when the whole file system needs to be backed up, only the unmodified files need to be detected. When the DataSync process is finished, the data synchronization is run again. In this process, Rsync will also compare and verify the data files to detect whether the data has been tampered, as well as the subtle changes of the data and the integrity of the data. Hash function is used to generate hash value for data and save it. If the value is equal to the saved hash value, it is considered that the data has not been modified. Otherwise, it is considered that the data has been modified. This method can detect the accuracy of data before and after synchronization.

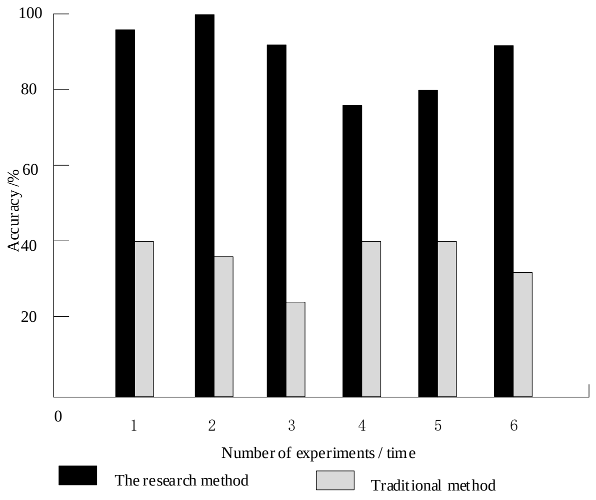

Figure 10 Comparison of data synchronization accuracy

Through the above experiments, it can be proved that there is no error during the file transmission after the synchronization of the research method, which indicates that the data at both ends are consistent after the data synchronization of the research method. After comparison, the synchronization method in this study has higher synchronization accuracy and better application effect than traditional methods.

8. Conclusion

The problem of database synchronization in distributed environment is the focus of current research, which ensures the data consistency of each node in the system. It is very important to adopt a reasonable and effective database synchronization method. According to the actual requirements of distributed system for database synchronization, the real-time data synchronization technology is studied. The main research contents are as follows

Firstly, we focus on the synchronization method of heterogeneous database. By

analyzing all kinds of data structures, it can divide them into three categories according to the complex program.
Secondly, in order to capture the change data, a new data capture method is proposed. This method only records the control information of the updated data items in the change table, which greatly saves the storage space and does not need to modify the original application table, thus reducing the operation and trouble. Because it combines the use of triggers, it is suitable for any database management system supporting triggers, so it is multifunctional.
This study only made a simple discussion, however, there are still some shortcomings Firstly, it only focuses on the synchronous replication between relational databases. The synchronization of databases, object databases and other databases needs to be studied.
Secondly, for the security of the information to be transmitted, there is not much consideration in data migration.

References

[1] Szustak, Lukasz. Strategy for data-flow synchronizations in stencil parallel computations on multi-/manycore systems[J]. The Journal of Supercomputing, 2018, 74(4): 1-13.

[2] Huang X, Ma Y. Finite-time h ∞ sampled-data synchronization for markovian jump complex networks with time-varying delays ☆[J]. Neurocomputing, 2018, 296(28): 82-99.

[3] Liu C M, Lai C C. A group-based data-driven approach for data synchronization in unstructured mobile p2p systems[J]. Wireless Networks, 2018, 24(7): 2465-2482.

[4] Felipe V, Lopes Eduardo J, et al. Traveling wave-based solutions for transmission line two-terminal data time synchronization[J]. IEEE Transactions on Power Delivery,2018, 33(6): 3240-3241.

[5] Wan Y, Zhang X, Dai Y, et al. An azimuth cut-off wavelength compensation method based on multiview synthetic aperture radar synchronization data[J]. Remote Sensing Letters,2020, 11(3): 245-254.

[6] Cheng J, Park J H, Karimi H R, et al. A flexible terminal approach to sampled-data exponentially synchronization of markovian neural networks with time-varying delayed signals[J]. IEEE Transactions on Cybernetics,2018, 48(8): 2232-2244.

[7] Li H, Gao X, Li R. Exponential stability and sampled-data synchronization of delayed complex-valued memristive neural networks[J]. Neural processing letters,2020, 51(1): 193-209.

[8] Wang X, Liu X, She K, et al. Extended dissipative memory sampled-data synchronization control of complex networks with communication delays[J]. Neurocomputing, 2019, 347(28): 1-12.