Real-time Transaction Scheduling Method for Distributed Database

Sun Qiao[1, a], Zhang Shaowei[2, a], Fu Lanmei[2, a], Sun Jiasong[2, a]

[1] Beijing GuoDianTong Network Technology Co. Ltd, 10070, Beijing, china

[2] E. E. Department of Tsinghua University, 100084, Beijing, China

[a]email: littlesmart@yeah.net

Keywords: Real-time transaction schedule; Distributed database; transaction management; Server cluster

Abstract. The existing distributed database is mainly based on the limited scale network environment. Many systems implement algorithms are difficult to apply when it needs to be expanded or involved into the field of distributed query processing, distributed transaction processing and data replication. In this paper, one real-time transaction management architecture and its details are proposed to improve the efficacy of SQL processing based on distributed database system. The seven primary functional models in the server cluster part and four functional models in the distributed database part are designed to enhance the query function such as insert or update in our network environment. The experiments and results proved that our method is more efficient than traditional methods.

Introduction

The rapid development of mobile internet gives the challenge to the massive data storage, management, computing and I/O design. The distributed storage system were produced such as GFS has been widely used. Distributed network management solution can be a unified management and scheduling of storage node resources with high reliability and maintainability. The distributed storage technology based on the mobile Internet needs to store and process large amount of data in order to meet the demand of mass data. Database technology is the foundation of all kinds of applications, a large number of application systems are based on the database system, so the performance of the database system is directly related to the success or failure of the application system and the value of the promotion and use. Due to the expansion of database application requirements and the development of computer hardware environment, especially the development of network technology, the distributed database system are more important than ever before[1][2][3].

The research of distributed database system began in the middle of 1970s. The current network and distributed technology gradually moving toward integration, distributed computing mostly based on network research, get a better price; and based on multiprocessor mainframe to distributed computing, due to its high price, difficult in many application fields were widely popular. Parallel processing technology is mostly used in fields such as data mining, high performance computing and so on [4][5]. It was used to calculate large amount, high real-time requirement and large amount of I/O. Now it tends to adopt a more generalized concept of distributed database, namely a by several independent and autonomous database system, which each member databases provide some functions to exchange data and provide services with other members.

Mobile Internet to achieve large amounts of data, query data is a multi-node distributed storage and the real-time transaction scheduling, the existing methods generally part of the query task is decomposed into sub transactions and the sub-transaction put in different nodes were treated, the priority assessment, consensus sequence of problems [6][7]. Improve the efficiency of distributed database transaction real-time scheduling, and optimize the decomposition process of the transaction. This method can greatly improve the performance of the distributed database system.

The remainder of this paper is structured as follows. In Section II, we give an overview of the real-time transaction management model. Section III introduced the implement of our real-time

transaction management architecture. The experiments and results are given in Section IV. We summarize our contribution in Section V.

Real-time Transaction Management Model

Transaction is the core concept in the database system. The concept of transaction was proposed in [1]. It can be used in the relational database to solve the problem that it is difficult to ensure the database size increases, complex structure, sharing user data brought about an increase in data integrity, security, concurrency and reliability. The concept of a transaction is the division of the operation of a database into a basic atomic unit called a transaction. In a distributed database system, a distributed transaction is a global transaction, usually consisting of a main transaction and a sub transaction executed at different nodes. Generally, the primary transaction is responsible for the starting, committing, and termination of the transaction; the sub-transactions are completed on the database access operations on the corresponding nodes. The so-called "global affairs" means a transaction that requires access or update data on multiple nodes, the so-called "local transaction" refers to a transaction that only accesses or updates data on a node [8][9].

With the centralized database system, in order to ensure database consistency and reliability and to ensure access to data correctness and validity, transactions in a distributed database system should also has four features: atomicity, consistency, independence and durability [10][11]. In a distributed database system, the request of any one application will eventually be transformed into a sequence of database storage operations on the corresponding nodes in the network. Compared with traditional transaction processing, the distributed transaction is executed on multiple nodes, and the error of any one node can cause the global error. So the distributed transaction is much more complex than the centralized transaction.

Distributed transaction control model is a common method to coordinate the implementation of the sub transactions among the members of a distributed transaction. The control model of distributed transaction execution is: the master-slave model, the triangle model and the hierarchical model[12][13].

The first control model is composed of a distributed transaction manager, which is used as a master controller with one or more local transaction manager as the slave controller. In this model, the distributed transaction manager sends a message to a local transaction manager, and accepts messages from where they are returned, to control all the operations that need to be synchronized. The distributed transaction manager activates every local transaction manager and indicates what to do. According to the instructions, the local transaction manager completes the respective access to the local database, and returns the results to the distributed transaction manager.

In triangle control model, the control is shared between the distributed transaction manager and the local transaction manager. The local transaction manager wants to send and receive data between the local transaction managers, and does not have to be distributed transaction manager as an agent. This control model avoids unnecessary data transmission between distributed transaction manager and local transaction manager.

In level control model, each local transaction manager itself can be used as a distributed transaction manager. This control model allows for additional design, when a local transaction manager receives a request from a local transaction manager and the local transaction manager itself becomes a global transaction manager. It can optimize the received transaction into another distributed transaction, and distribute it to the relevant local transaction manager.

Implementation of New Real-time Transaction Management Model Architecture

In this paper, the real-time transaction scheduling we proposed is composed by many large single real time database. It doesn’t matter about where data stored or which server node data read from. In our architecture, the number of server nodes will affect the overall performance and reliability. The functional structure of our transaction management module is shown in figure 1. There are two parts named as server cluster and distributed database.

In the first part, there are seven primary functional models: Status checking, data migration, communication platform, computer communication, transaction management, rout and index. The status checking module is the monitor module of each module and node in the distributed real-time database system. The strategy of the main modules of the distributed real time database system is developed in this module. It is mainly responsible for checking all the information of each node. The data migration and data management modules are responsible for the management of data extracted from distributed database. As shown in the figure, all client access the server cluster through computer communication platform. The transaction management modules will manage all kinds of the requests proposed. Transaction management module is responsible for the management of transaction from generation to transaction, which is the unified management module of internal and external request. Generally, the route and index are dynamic and adapted in the cluster to handle the data stream.

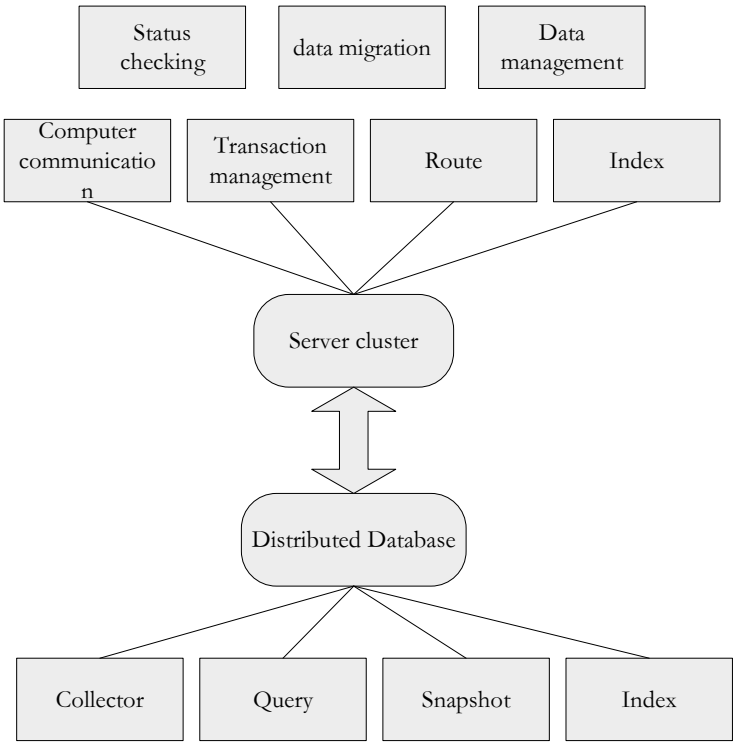

Fig.1. New real-time transaction management architecture

In the second part, there are four functional models: collector, query, snapshot and index. The collector is statically bound to the fixed snapshot service, which provides real time data. The query requests of the real time data are different from the entry to the database, and the snapshot service provides a subscription to the query service for real-time data, while the data is stored in the archive service. Historical data and real-time data of the subscription are through the query service, query service in the snapshot service to obtain real-time data, access to historical data in the local query. And the Snapshot module is the main source of historical data, while providing subscription service, it is the core module of real-time database, real-time data push and real-time data subscription is the core function of real-time database system. The principle of application of transaction management module to ensure the real-time and throughput of data push and subscription, so as to ensure the function of the core is not affected. The index module is a module which is responsible for the dynamic partition and adjustment of the distributed real-time database system.

Experiments and Results

In order to evaluate the performance of the above method, we evaluated and tested in the present network environment. The specific configuration is: 15 servers (2CPU, IBM-x3650M4-2U 6 Xeon core 12 thread E5-2620, 48G memory, 2T hard disk), the operating system for Ubuntu11.10 and

Windows7. In our 10Mbps network environment, we completed the test of local/remote disaster recovery replication fault toleration. All the local/remote server configuration are same. The SQL statements in the insert, update, select respectively, to perform some statements, respectively, and calculated the average execution time; in the test, we tested 100 table is in the database, list of the largest total 50000 records, each record in a total of 15 fields, each field not more than 50 bytes.

Fig.2. Testing results of execution time by traditional architecture

Fig.3. Testing results of execution time by new architecture

It shows in the figure 2 and figure 3 that the average execution time of the insert, update, and select is different but have the similar changing tendency. In figure 2, the insert SQL function has the most execution time in three functions. The update and select have the close results. In figure 3, the insert SQL function has less execution time than which in figure 2. The update and select have the different results. The three functions all decreased than which in figure 2. For the whole point of view, the efficiency of our architecture is efficient.

Conclusion

With the wide application of the distributed computing environment, distributed database system has become an important part of the information processing, it eliminates the many disadvantages of traditional centralized database, suitable for a variety of architectures. In this paper, one real-time transaction management architecture and its details are proposed to improve the efficacy of SQL processing based on distributed database system. The experiments and results proved that our method is more efficient than traditional methods. Next we will focus on the intelligent optimization mechanism of query transaction based on server cluster.

Acknowledgement

This research was financially supported by Science and Technology Project of the State Grid Corporation of China (SGBJDK00KJJS1500180) and the State Grid Information & Telecommunication Group CO.,LTD. (SGITG-KJ-JSKF[2015]0010).

References

[1] Gray, J. Notes on Database Operating Systems. Operating Systems: an Advanced Course [J]. Lecture Notes in Computer Science, vol.60, PP: 397-405. Springer, Berlin, 1978.

[2] Agrawal D, Abbadi A E. The Generalized Tree Quorum Protocal: An Efficient Approach for Managing Replicated Data [J]. ACM Trans, On Database Systems .1992.

[3] Kemper A, Wiesner C. HyperQueries: Dynamic Distributed Query Processing on the Internet[C].VLDB 2001, Proceedings of, International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy. 2001:206-207.

[4] Zhang J, You S, Gruenwald L. Lightweight Distributed Execution Engine for Large-Scale Spatial Join Query Processing[C]. IEEE International Congress on Big Data. IEEE Computer Society, 2015.

[5] Akbarinia R. Data Access in Dynamic Distributed Systems: Basics, Concepts and Techniques of P2P Query Processing[M]. VDM Verlag, 2009.

[6] Zhuang Y, Li Q, Chen L. Multi-query Optimization for Distributed Similarity Query Processing[C]. international Conference on Distributed Computing Systems. IEEE, 2008:639-646.

[7] Eom Y, Hwang D, Lee J, et al. EM-KDE: A locality-aware job scheduling policy with distributed semantic caches[J]. Journal of Parallel & Distributed Computing, 2015, 83:119-132.

[8] Pawel Jurczyk, Li Xiong. Dynamic Query Processing for P2P Data Services in the Cloud[C].

[9] Montgomery C D. Dynamic communication channel allocation method and system: US, US 7388872 B2[P]. 2008.

[10] Database and Expert Systems Applications, 20th International Conference, DEXA 2009, Linz, Austria, August 31 - September 4, 2009. Proceedings. 2009:396-411.

[11] Saravanan R, Vivekananth P. Range of Query Processing in Peer to Peer Networks[M]. Instrumentation, Measurement, Circuits and Systems. Springer Berlin Heidelberg, 2012:1-10.

[12] Rubin S H. Adaptive case-based reasoning system using dynamic method for knowledge acquisition: US, US8447720[P]. 2013.

[13] Rodolfo A P R, Graciela V A, José A M F, et al. Vertical fragmentation design of distributed databases considering the nonlinear nature of roundtrip response time[C]. International Conference on Knowledge-Based & Intelligent Information & Engineering Systems. Springer-Verlag, 2010:173-182.