A Locking Mechanism for Distributed Database Systems

Zhangbing Li
School of computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China
Key Lab of Knowledge Processing and Networked Manufacturing, College of Hunan Province, Xiangtan, China
Email: lzb_xt@126.com, zbli908@gmail.com

Zilan Zhu and Shaobo Zhang
School of computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China

***Abstract*—To overcome the disadvantages of the traditional locking mechanism for Distributed Database Systems (DDBS), such as the low transaction concurrency and the high communication costs, this paper studies the traditional mechanism with Two-phase Locking (2PL) protocol, and redefines the lock types and their compatibility matrix in DDBS, designs the structure of the Lock Table, improves the lock policy and the lock algorithm. Those make up of a new locking mechanism for DDBS. The experimental results show that this mechanism can intensively control and flexibly manage the requirements for locks with improved 2PL protocol and multi-granularity locking, and effectively ensure serializing scheduling of transactions and decrease the costs of communications while locking. It obtains the better transaction concurrency than the traditional mechanism.**

***Index Terms*—Locking Mechanism; Distributed Database; Two-Phase Locking Protocol; Distributed Transaction; Concurrency**

I. INTRODUCTION

In order to ensure the atomicity, consistency, isolation and durability (ACID) characters of transactions in database management system (DBMS), the locking mechanism is used for Distributed Database System (DDBS) as the Concentrated Database System (CDBS). Reference [1] indicates that lock is a software mechanism. It means that a user has held some resources (data segment, data sheets, record, etc.) to confirm the integrity and consistency of data when transactions access the same resource. Works [2, 3] present that lock is a very important technique for achieving the control of the concurrency. Its precise control is decided by lock type. They declare that the traditional mechanism adopts the Two-phase Locking protocol (2PL) for locks in DDBS as in CDBS. Several studies [1, 4-7, 10-11] consider that a distributed transaction in DDBS will cause the amount of messages for conveying and a large number of communication costs. With increasing of the transactions concurrently, the locking mechanism will bring up many problems through network, such as starvation to death, consistency of data copies and congestion of communication for the distribution transaction concurrence. Works in [8, 12] describe two protocols for the detection of deadlocks in distributed databases, and proposes a locking protocol to coordinate access to a distributed database. However, there are still some problems like the delay of communications and starvation of the concurrent transactions. These problems result in the concurrence degree is low. Therefore many DDBS prototypes could not be commercialization yet.

This paper proposes a new locking mechanism to resolve the problems. First, the traditional mechanism for locking in DDBS is studied, which includes the basic types of lock in RDBS and the comparison of the traditional mechanism for locking with 2PL between the CDBS and DDBS. Second, it defines some conceptions for locking in DDBS. The lock types, its relation of the partial order and the compatibility matrix are redefined in DDBS for the new locking mechanism. The lock granularity and the concurrence degree are redefined too. Third, it designs a new data structure for the Lock Table and improves the lock policy and the transaction schedule, then gives a new algorithm for locking with 2PL in DDBS. All those come into being a new locking mechanism. Finally, the simulation is performed, and the experiment results show that it has the better performance than that of the traditional mechanism in the aspects such as the time of locking and releasing, the communication costs for locking, the starvation and the concurrence degree of transactions.

II. TRADITIOAL MECHANISM FOR LOCKING IN DDBS

A. The Basic Types of Lock in Rdbs

The typical CDBS is Relation Database System (RDBS), which has the good mechanism for transactions processing and locking. References [1, 6] affirm that the locking mechanism in DDBS is similar to that in RDBS, but it is more complex because of data distribution and transaction concurrence in distributed database system.

Several works [2, 4, 5] assert that the Lock table and Two-phase Locking protocol (2PL) are used in Relation Database Management System (RDBMS) with multi-granularity protocol for assisting operation of lock scheduler. For each element, if locked, it is indicated by

doi:10.4304/jnw.9.8.2211-2217

lock table corresponding to the transaction. According to the literatures [3, 4], three RDBMS lock models are listed in table 1.

Where, Shared lock (S) supplies the services that the data locked can be read by any transaction which is the owner of the lock. Exclusion lock (X) takes the functions that only the owner can read or modify the data locked. Update lock (U) is that the owner can read the data locked. But after it is upgraded to X lock the data locked can be modified by the owner.

TABLE I. THREE RDBMS LOCKING MODEL

DBMS	Type of locking	Granularity of locking
DB2	IS, IX, SIX, S, U, X	Table space, table, record, index, data page
Oracle	S, X, IS, IX, SIX	Table, record, data page
Sql server 2005	S, U, X,Sch	Record, page, table, database

Intent locks include Intent Shared lock (IS), Intent Exclusion lock (IX) and Sharing Intent Exclusion lock (SIX). It indicates that the owner wants to acquire a Shared lock(S) or Exclusive lock (X) on some of the resources lower down in the hierarchy. It is usually placed at the table level. The owner can read the data of row when it holds Shared lock and can modify the data of the row when it holds Exclusion lock. SIX is generated when the applications request IX lock if they have held IX lock or S lock.

Scheme lock (Sch) includes Schema modification lock (Sch-M) and Schema stability lock(Sch-S). Sch-M lock is used when a table Data Definition Language (DDL) operation (such as adding a column or dropping a table) is being performed. Sch-S lock is used when queries come to be compiling. But it does not block any transaction. So DDL operations cannot be performed on the Sch locked table.

Practically, the oracle database adopts DML (Data Manipulation Language) locks as data locks, and takes DDL(Data Definition Language) locks as dictionary locks, Internal Locks/Latches, Distributed Locks and Parallell Cache Management Locks. The data are distributed by Data-Link mechanisms in oracle.

References [4, 5, 13] allege that DML locks protect data in a table. DML operations can hold data locks on in according with two different levels: specifically rows and entire tables. It includes row-level locks and table-level locks. Row locking provides the lowest level of locking to make the transaction concurrency best possible. A transaction gets an exclusive DML lock for modifying each row by anyone of the following SQL (Structured Query Language) statements: INSERT, UPDATE, DELETE, and SELECT with the “For UPDATE” clause. If a transaction obtains a row lock for a row, the transaction also holds a table lock on the table in terns of the row. The table lock prevents the DDL operations collisions that would override data changes in a current transaction.

DDL locks Protect the definition of an object while being used by a DDL operation. Recall indicates that a DDL statement implicitly commits.

Internal locks and latches protect Oracle internal database structures such as data files. Distributed Locks ensure the consistency of the data and other resources distributed among the various instances. Data-Link as a point or a link refers to a database object. Parallel Cache Management Locks are the Distributed Locks that cover one or more data blocks (table or index blocks) in the buffer cache. They are entirely handled by Oracle internal functions or instances instead of transactions.

B. Comparison of the Traditioal Mechanism for Locking with 2PL Between CDBS and DDBS

To ensure the consistency of data and the serializable scheduling of transactions with collision, the Two-phase Locking protocol (2PL) is widely used in DBS. It includes the first phase for holding locks and the second phase for releasing locks. However, there is a condition that all the requests of locking are prior to that of releasing. A transaction can request to hold locks of any type on data item, but cannot release locks in its stage of locking. By contraries, a transaction can release locks of any type on data item, but cannot apply for any lock again in its stage of unlocking. References [1-6] expound this theory in detail.

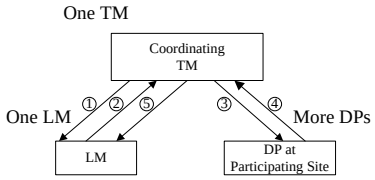

Figure 1. 2PL in centralized database

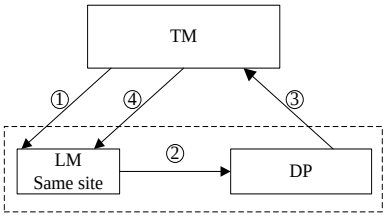

Figure 2. 2PL in distributed database

The feature of Two-phase Locking protocol in centralized database is as follows:

(1) Lock manager (LM) is responsible for the management of locks.

(2) Data dictionary assists LM to complete the distributions for locks.

Specific steps are listed as follows and shown as figure 1, where TM is transaction manager, LM is lock manager, DP is data processing.

(1) Lock request;

(2) Lock granted;

(3) Database operation;

(4) End of database operation;

(5) Release locks.

The lock mechanism is also used in distributed database management system. The features of distributed database two-phase locking are listed as follows:

(1) There is a lock manager at each site;

TABLE II. THE COMPATIBILITY MATRIX OF LOCK TYPES

		Xthe lock type of request					
		LS	GS	LU	GU	LX	GX
The lock type of hold	LS	√	√	√	√	×	×
	GS	√	√	√	√	×	×
	LU	×	×	×	×	×	×
	GU	×	×	×	×	×	×
	LX	×	×	×	×	×	×
	GX	×	×	×	×	×	×

(2) Concurrency control is performed by cooperation among the lock managers at the corresponding sites, where data is involved in the set of transactions.

Specific steps are listed as follows and shown in figure 2:

(1) Lock request;

(2) Lock granted;

(3) End of database operation;

(4) Release lock.

III. NEW LOCKING MECHANISM IN DDBS

A. Lock Granularity and Compatibility Matrix of Lock Types in DDBS

Before discussing the lock mechanism, there are some conceptions to be defined below necessarily.

Define 1. A global transaction (GT) is a transaction lunched by DDBS from some site but contains some data objects in other sites. Accordingly, a local transaction (LT) is a transaction lunched by DDBS from local site which does not contain any data object in other sites.

Define 2. A global lock (GL) is a lock on the data object for a GT transaction in DDBS. Accordingly a local transaction (LL) is a lock on the data object for a LT transaction in local site of DDBS.

Define 3. A Lock Gross (LG) is a product of the number of all data objects for the GT transaction multiplying the total execution time of the transaction in DDBS. The formula is as follows:

$$LG = t_T \cdot N = t_T \cdot \sum_{j=1}^{m} N_j \qquad (1)$$

where, t_T is the total execution time of the transaction in DDBS which the unit is second, Nj is the number of the data objects of the transaction processing ($j=1\sim m$), N is the sum of Nj.

Define 4. A concurrence degree (CD) of transactions is a quotient that the sum of LGs when each transaction performs solely divides by the LG when all transactions perform concurrently successfully. The formula is as follows:

$$CD = \sum_{i=1}^{k} LG_i / LG = \sum_{i=1}^{k} t_{Ti} \cdot N_{Ti} / (t_{Tc} \cdot \sum_{i=1}^{k} N_{Ti}) \qquad (2)$$

where, t_{Ti} is the time of the i-th transaction performing solely, t_{Tc} is the time of all the transactions performing concurrently in DDBS, N_{Ti} is the number of the data objects which the i-th transaction processing ($i=1\sim k$), k is the sum number of the transactions. If any transaction is starved to death, the t_{Tc} will be a very big number and the CD value will be tending to zero.

1. The roles of global directory for locking

Several works [4, 5, 7, 8] declare that the global directory (GD) generally is stored in the main site of DDBS. The metadata includes the objects such as the name of data and the variety of the control information related to DDBS, for example, each site name and its IP address in distributed database. Those are placed in GD and accessed by DDBS. GD can be used for locating and searching the position of data objects in DDB quickly. It provides the source address of data objects when the lock table is created.

2. Lock granularity of this mechanism

According to the references [9-17] the concurrency control and recovery performance are affected by the lock granularity. The smaller the granularity size, the higher the concurrence degree is, vice versa. The data objects to be blocked can be some logical units in CDBS, such as the values of attribute or attribute set, records, tables, index and the whole database. It can also be some physical units, such as page, physical records, etc. Considering the average response time and concurrence degree of transactions, throughput of datum, costs of locking, flexibility of lock management agility blockade, granularity of locks and the partial order relation of lock types should be enhanced, which are adopted by this paper for databases, tables, segments, records, values of attribute or attributes set.

3. The compatibility matrix of Lock types

As the traditional RDBS, the lock types are divided into Shared(S), Update (U) and Exclusive(X) in the DDBS. But there are differences between local and global locks, so the types of locks in DDBS are defined as the local Shared lock (LS), the global Shared lock(GS), the local Update lock(LU), the global Update lock(GU), the local Exclusive lock(LX) and the global Exclusive lock(GX). Usually the global locks have higher level than the local locks. The relations among the lock types are shown as (LS,GS)<(LU,GU)<(LX,GX), which forms a partial order. Those consist of six levels or lock models. When the data object is holding a lock, the requests of locking will be accepted in terms of the compatibility of lock's type. References [4, 5] draw out the traditional lock compatibility matrix. The new lock compatibility matrix is formed as listed in Table 2.

B. Design the Data Structure of Lock Table

The lock table adopts the grouping model as the traditional method. It is designed as a cross-link list which includes two parts in distributed database, the Data

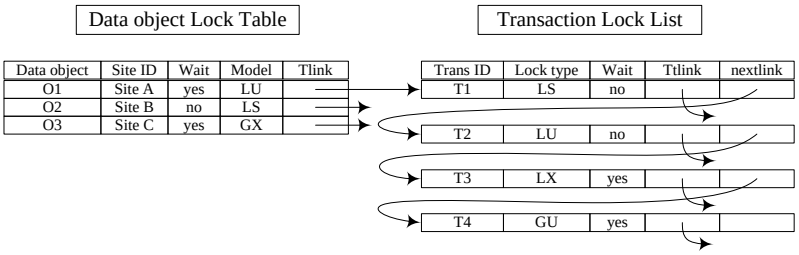

Figure 3. The data structure of Lock Table

Objects Lock Table and the Transactions Lock List. Its structure is shown as figure 3, which is dynamically managed and maintained by LM.

The Data Objects Lock Table is implemented by hash table and takes the compounding of Data object and Site ID as the hash code. Any database element that is not locked does not appear in the table. So the table size is proportional to the number of the objects locked, and not relevant to the whole database. It includes the attributes such as Data object, Site ID, Wait, Model and Tlink. The Data object yield is used to store the objects of transactions launched by TM for locking in distributed database. The Site ID field indicates the data object positions of network. The "Wait" field has two values which are "yes" and "no" respectively. The "yes" value presents some locks on the data object are waiting for, but the "no" value means no transaction is waiting for any lock. The Model field shows the integrative model of all locks on the same object. If its value is "GX" or "LX", it shows that this data-object has a lock type of "GX" or "LX" in some transactions, so any of new lock requests will be waiting. In the case of "GU" or "LU" of the Model value, it shows that this data object has a lock type of "GU" or "LU" in some transactions, so any of new lock requests will also be waiting except that the upgrade request of a transaction, which holds the lock of the "GU" or "LU", can be accepted. The "GU" or "LU" can be respectively upgraded to "GX" or "LX" independently. If the Model's value is "GS" or "LS", it shows that all locks on this data object are the type of Shared lock, and can accept the requests for locking as the type of "LS", "GS", "LU" or "GU" but "LX" and "GX" according to the compatibility of lock types and the lock policy. The Tlink yield is a link to the transactions which is locking on the data object. The scheduler of LM will get directly the necessary information without seeking in the Transaction Lock List.

The Transaction Lock List is a link list which includes the attributes as Trans ID, Lock type, Wait, Ttlink and Nextlink. The Trans ID field indicates the transaction ID number which requests to block the data objects, the Lock Type field indicates the types of locks, the Wait field is similar to that in the Data Object Lock Table, but the "no" value standards all the locks of the transaction are held successfully and "yes" is still waiting. The Ttlink yield is a link pointed to the list of all items in the transaction, while the Nextlink is a link pointed to the next Lock-list of the data object in other transaction. The Ttlink is useful for seeking all resources of a transaction to release locks while the transaction is submitting or aborting.

C. The Management and Control for Locking

The management and control for locking include the lock policy, the process of locking and unlocking. Several references [1, 4, 5, 7, 9, 10] present the traditional lock policy, but it is not perfectly suitable for the new locking mechanism in DDBS. Therefore it should be improved, and the process of locking and unlocking should be changed correspondingly.

1. The improved lock policy

The lock policy is improved for locks granting to a transaction as follows:

(1) First come and first service. The request of locking by some transaction arrives in LM firstly, the lock service is held first by the transaction according to the compatibility matrix of lock types. This policy requires the transactions agree to wait the longest time for a lock, and ensure that no transaction will starve to death, that is the status of a transaction always waiting for a lock.

(2) Shared locks as a priority. All the Shared locks for waiting will be granted firstly, and follow to grant an Update lock if there is any for waiting. Only there is no waiting for any locking, the Exclusive lock can be awarded to the transaction for the request. This policy allows the transactions waiting for an Update lock or an Exclusive lock to starve to death.

(3) Upgrades as a priority. If a transaction with a "GU" or "LU" lock is waiting to update to be "GX" or "LX" lock, it will be updated as a priority in turn, or use other policies above.

(4) Global locks as senior than Local locks. If the data object holds a Global lock, the requests of its local lock have to be waiting until the global lock is released.

2. The process of locking

Let's suppose that a transaction T requests to hold a lock on the data object A. if A has not an item in the Data object Lock Table, the corresponding item will be created and the request will be accepted. If there is existing the item of A, the value of the yield "Model" and "Wait" can help to decide how the lock can be added.

If there is an item on A and the Model="GX" ,"LX" or "GU","LU", it shows there is an Exclusive lock or Update lock on A for global or local in the Transaction Lock List, and the lock requests arrived will be usually refused. If so, a new item of the Transaction Lock List will be created and added the end of it with "Wait" yield as "yes". The new lock of the list item should be correctly linked by the field of Tlink and Nextlink. Otherwise, if

the Model=”GS” or “LS”, the lock request of the shared or Update type lock will be accepted and granted. In this case if the request lock is Update type, the Model should be set to “GU” or “LU” value. If an upgrade request is coming, the “GU” or “LU” lock item in the Transaction Lock List should be found; and if the Wait yield is “no”, the upgrade request will be accepted and the lock of “GX” or “LX” will be granted correspondingly, meanwhile the Model yield will be update as “GX” or “LX” simultaneously. Otherwise, the request will be refused.

3. The process of unlocking

Supposing a transaction T requests to unlock on the data object A, the lock item of the transaction linked the A will be deleted. If the Lock type of T is “LS”, the Model yield in the Data object Lock table not need to be modified. If the Lock type of T is “GS”, it is necessary to search the whole Transaction Lock List to find whether there are some locks at the lower level. If there are some “GS” locks, the Model yield in the Data object Lock table not need to be modified when deleting the lock item in the Transaction Lock List, or it will be modified to be lower level according to the partial order of the lock type, it means that “GS” is updated to “LS” in the Model yield. Similarly if the lock of “LU”, “GU”, “LX” or “GX” is to release and there are still some locks like “GS” or “LS” on the A, it will be correspondingly updated to “GS” or “LS” for degrading the level of the lock model in the Model yield because the locks released have exclusive and unique property. If the Transaction Lock List is null, the data object A will be removed from the Data object Lock table.

D. The Locking Algorithm with 2PL in DDBS

The improved Two-phase locking protocol (2PL) in DDBS are described as:

(1) All transactions should hold necessarily all their locks before they start. A GT transaction has to get all of its locks including the global and the local.

(2) A transaction will release all of its locks when it is submitted or aborted.

The innovative architecture of transactions for locking with the improved 2PL protocol is shown as figure 4.

The basic ideas of locking algorithm are listed as follows:

(1) If the transaction scheduler launches a transaction in some site, According to the SQL sentence optimized, the TM sends the requests for locks to the LM.

(2) If all locks are held in the LM, a start signal is transferred to the DP to process the data of the transaction.

(3) If the transaction is a GT transaction, according to the GD dictionary the requests for Global locks will be sent to the other LM in corresponding site on the network, and the sub-transactions if necessary will be transferred to the other TM simultaneously.

(4) The sub-transaction of GT holds all of its locks through the LM in its site.

(5) If all locks of the sub-transaction are held in the LM, a start signal is transferred to the DP to process the data in its site.

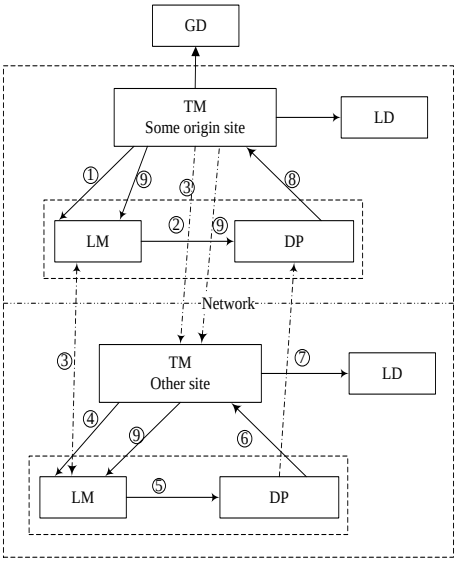

Figure 4. The innovative architecture of transactions for locking with the improved 2PL in DDBS

(6) The DP notices the end message of the data processing to the TM in this site, and may transfers the results to the DP of the GT transaction in origin site.

(7) The DP in origin site sends a final signal to the origin TM for preparing to process the data.

(8) No matter whether the DPs are processing in success or failure, they will send the messages to the original TM, which decides the transaction is submitted or aborted.

(9) Whether the GT transaction is submitted or aborted, the TM in original site which launches the GT transaction will request the local LM and corresponding LMs in other sites to release all locks of this GT transaction interrelated.

If the copies of data object need to be updated synchronously, in order to keep the consistency of data copies, there is an independent component to start a consistent sub-transaction to request for the locks. References [14-18] provide some methods to solve this problem.

IV. SIMULATION EXPERIMENTS

Before simulating, we implemented a transaction management system with the new locking mechanism by VC++ 6.0, which includes the modules of TM, LM and DP components. It takes the text files and their lines as the data objects. An Excel file is used for the GD dictionary storing metadata of all data objects. Other Excel files are used for the LD dictionary storing metadata of data objects at the local site. The processes based on sockets simulate the actions of transactions in DDBS to read and write the data. The locking system is respectively tested in a local network and a Metropolitan Area Network (MAN). They all are Gigabit LAN. The results of experiments are shown as Table 3.

TABLE III. THE EXPERIMENTS RESULTS OF CONCURRENT TRANSACTIONS

	Number of Transactions	Number of data objects	Average lock time(s)		Average release time(s)		Starve to death		Number of communications		times of datum process(s)	
			O	I	O	I	O	I	O	I	O	I
Local network	1	5	0.5	0.5	0.1	0.1	0	0	60	55	8	7
	5	20	1.3	1.2	0.5	0.4	0	0	542	461	15	13
	20	78	3.7	3.1	1.1	0.9	0	0	2605	1832	42	36
	100	382	12.1	9.5	3.7	2.2	4	3	*	*	*	*
MAN	1	5	0.7	0.7	0.2	0.2	0	0	63	56	9	8
	5	20	1.6	1.5	0.9	0.7	0	0	572	485	17	15
	20	78	4.7	4.1	2.5	1.9	0	0	3121	2681	53	47
	100	382	14.1	11.5	5.3	4.4	6	5	*	*	*	*

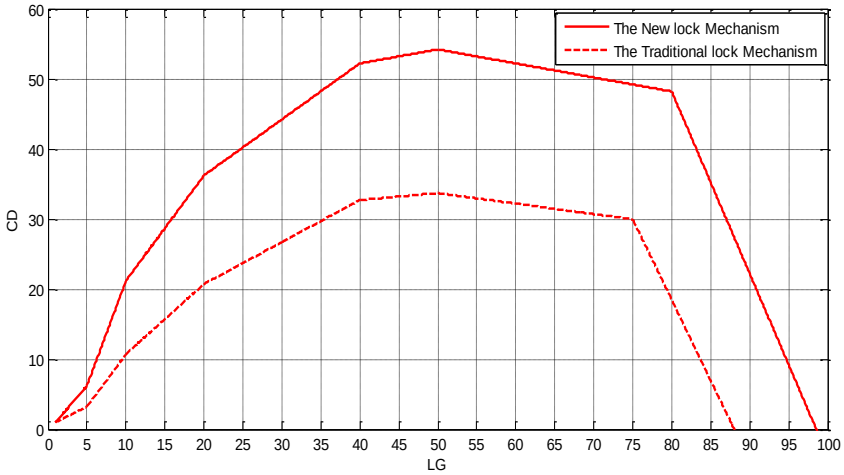

Figure 5. The comparison of the concurrence degree between the new and the traditional lock mechanism

In above table, the unit of times is second, the O stands for the Original lock algorithm and the I is the Improved lock algorithm. The star sign indicates a very large or infinite number.

The experimental results from the table show that the new locking mechanism has a shorter average time for locking and unlocking than that of the traditional mechanism under the same number of concurrency transactions. At the same time it has a number of communications for locks fewer than of the traditional mechanism. Therefore its cost for locking is lower. Further more, it is important that the new locking mechanism reduces the number of the transactions which are starved to death. Thus the concurrence degree of transactions is increased in DDBS.

The comparison of the concurrence degree between the new and the traditional mechanism for locking according to the formula 2 is as the figure 5. The figure shows that the concurrence degree increases with the LG under the value of 50 and reduces quickly greater than that of 80. From the results of experiments, there are obviously outcomes that the improved algorithm for locking is more quickly than the tradition algorithm in DDBS, the starvation problem of transactions has been smoothed and the concurrence degree is increasing obviously, the communication costs for locking are cut down abundantly. It is more security for the data consistency in DDBS.

V. CONCLUSION AND FUTURE WORK

Through studying the traditional mechanism for locking with 2PL in distributed database system, this paper designs a new architecture for the transactions scheduling in DDBS, and improves the lock policy and the lock algorithm with the 2PL and multi-granularity in DDBS, which come into being a new locking mechanism for transactions. The experiments show that the new mechanism has enhanced the concurrence degree of transactions and cut down the cost of communications, and provides a more efficient approach for the consistency of the distributed data. In the future, the component for the consistency of data copies in DDBS will be designed an implemented. The new locking mechanism will be expanded to apply in the storage for the cloud data.

ACKNOWLEDGEMENTS

This research is supported by NSFC, under grant number 51075142, and by Science and Technology Planning Project of Hunan Province, under grant number: 2011FJ4300 (B11219), 2012FJ3047 (11C053) and 2013FJ4048.

REFERENCES

[1] Cory Devor, C. Robert Carlson. “Structural locking mechanisms and their effect on database management

system performance", *Information Systems*, Volume 7, Issue 4, 1982, Pages 345–358

[2] Thomasian A. "Two-phase locking performance and its thrashing behavior", *ACM Trans Database Syst*, 1993, 18(4): 579–625.

[3] Kedem Z M, Silberschatz A. "Locking Protocols: Rrom Exclusive to Shared Locks". *Journal of the ACM*, Volume 30, Number 4, 1983.

[4] P. A. Bernstein and E. Newcomer. Principles of Transaction Processing. *Morgan Kaufmann Publishers, SanMateo, CA, USA, 1*997.

[5] M. Tamer Ozsu etc.. Principles of Distributed Database Systems (2E). *TsingHua University Press.* 120-135. 2002

[6] Ceri S, Navathe B, Wiederhold G. "Distribution Design of Logical Database Schemas", *IEEE Transactions on Software Engineering*, Vol. SE-9, Number 1, 1983.

[7] M. Tamer Ozsu. "Modeling and Analysis of Distributed Database Concurrency Control Algorithms Using an Extended Petri Net Formalism", *IEEE Transactions on Software Engineering*, vol. SE-11, no. 10, pp. 1225-1240, Oct. 1985.

[8] Daniel A. Menasce, Geraild J. Popek, and Richard R. MuntZ. "A Locking Protocol for Resource Coordination in Distributed Databases", *ACM Transactions on Database Systems*, Vol. 5, No. 2, June 1980, Pages 103-138.

[9] Gray J N, Lorie R A, Putzolu G R. "Granularity of Locks and Degrees of Consistency in a Shared Data Base", *In Proceedings of VLDB*, 1975.

[10] Yannakakis M, Papadimitrioum C H, Kung H T. "Locking Protocols:Safety and Freedom from Deadlock", *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, 1979.

[11] D. Z. Badal. "The distributed deadlock detection algorithm", *ACM Trans. Comp. Syst.*, 4(4) pp. 320–337, November 1986.

[12] Daniel A. Menasce, Richard R. MuntZ. "Locking and Deadlock Detection in Distributed DataBases", *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 3, p. 195-202, MAY 1979.

[13] Oracle documentation. "Overview of Locking Mechanisms", http://docs. oracle. com/cd/A57673_01/ DOC/server/doc/SPS73/chap7. htm, 2013. 08

[14] Özgür Ulusoy. "Analysis of concurrency control protocols for real-time database systems", *Information Sciences*, Volume 111, Issues 4, November 1998, Pages 19–47

[15] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. "Understanding replication in databases and distributed systems", *In Proc. of the 20th International Conference on Distributed Computing Systems(ICDCS)*, pp. 464–474, Taipei, Taiwan, April 2000. IEEE-CS Press.

[16] Kam-Yiu Lama, Tei-Wei Kuob, Ben Kaoc, etc. "Evaluation of concurrency control strategies for mixed soft real-time database systems", *Information Systems*, Volume 27, Issue 2, April 2002, pp. 123–149

[17] Li Zhangbing, Che Wujiang. "A New Algorithm for Data Consistency Based on Primary Copy Data Queue Control in Distributed Database", 2011, DOI: 10. 1109/ICCSN. 2011. 6014424

[18] Jin Jing, Omran Bukhres, Ahmed Elmagarmid. Distributed Lock Management for Mobile Transactions. *15th International Conference on Distributed Computing Systems, Vancouver, BC, Canada*, June 1995.