UDC 004.031

Włodzimierz Khadzhynov[1], Dariusz Bernatowicz[2]
Technical University of Koszalin, Department of Electronics,
Śniadeckich 2, 75-453 Koszalin, Poland,
email: [1]hadginov@ie.tu.koszalin.pl, [2]dber75@o2.pl

Two-tier replication based on Eager Group – Lazy Master model

A scheme of two-tier replication based on Eager Group and Lazy Master models is presented. Initial transactions used permit remote nodes to read and update database. The algorithm of optimisation and commitment of initial transactions into form of base transaction are realised by initial transactions manager of master node.

Key words: *two-tier replication, initial transaction, base transaction, master node, slave node, algorithm of initial transaction optimisation.*

1. Introduction

The matter of replication is highly widespread and implemented in modern distributed database systems. Among the most secure schemes of transaction replication there are Eager Group and Lazy Master [1]. In their case, the problems of reconciliation do not occur, but there are some other problems, such as: servicing absence of remote nodes by Eager systems or instability of Master systems in a higher scale. Here is concerned the situation, which is often present in practice, where the architecture of company database system consists of a few efficient servers of global importance and any other amount of static or remote local nodes. Using only of replication model causes non-optimal utilisation of resources of company computer units.

In this paper, the Two-tier Eager Group – Lazy Master model of replication is proposed to avoid the problems above. In this approach, the operation of efficient global servers is based on Eager Group replication model, which allows the performing of data upgrade on-line. On the other hand, remote local nodes are using Lazy Master replication model. This architecture allows the solving of remote nodes problem and avoiding instability of loading.

2. Basic terms

Replication node is an independent working station with a replication module installed on [2]. Each node realises the function of data receiving and sending from the other node in a system. In advanced replication there are two kinds of replication nodes

— master nodes (master site) and slave nodes (slave site). In master nodes, transaction commitment and propagation of changes to other available nodes are performed in master nodes, as well as data changing. Slave nodes also perform data changing, but they transfer the commitment of transaction to master nodes.

Object of replication is name of an object of scheme, which is duplicated in all database systems and it creates replication environment [2]. The operations, performed on replication object in one database of environment, are propagated into duplicated objects of other databases. A set of logically connected objects of replication creates a group of replication objects. Each replication object may be included into only one group.

Group of replication nodes is a group among which upgrade of replicas is performed basing on the pattern from one of the group node. In synchronous replication scheme upgrade of all group nodes is realised as a single transaction.

The **transaction** means the smallest atomic set of instructions, which manipulates over data [3].

3. Characteristic of particular schemes of replication

3.1. Eager Group Replication

Eager Group replication scheme is presented in Fig. 1a. This kind of replication contains master nodes M, which are equal to each other. Each of them has the opportunity of data changing and those changes propagation to other replicas of particular group as a part of single transaction. Replicated objects are the same at all nodes in respect of structure and their content. Due to upgrading of all replicas during upgrading of any instance of object, in Eager replication scheme there is no serialisation anomalies and no need for reconciliation of conflicts, which are characteristic of Lazy replication [1]. Local operation, which breaks the consistency of remote node scheme, cannot be realise and causes aborting of the whole transaction.

Significant feature of Eager Group replication are waiting or locking states. They occur during of an attempt to the access to the same object by two colliding operations, which appeared from different transactions and at least one of them is in writing mode. A high increase of amount of locks with increasing the amount of nodes limits the scalability of the system, which makes operation of the system practically impossible. From functional point of view, achieving of data consistency and ordering of colliding operations is realised by using distributed locking or atomic broadcast.

During distributed locking the node is waiting for transaction realisation for granting locks on any other group nodes. If all nodes grant the locks the transaction is realised in all nodes. If not, the transaction is delayed and request is to be repeated after some time. In this case, two-phase commit protocol is used [3].

In replication of database based on atomic broadcast (ABCAST) group communication primitives are used. Total order usage guaranteed by ABCAST, is provided by transaction manager, which orders colliding transactions. Thank to reliable exchange and providing of global messages ordering, those systems manage exchanging messages between group nodes. In this case, it has to be assured, that two colliding operations are realised in ABCAST order for all nodes [4].

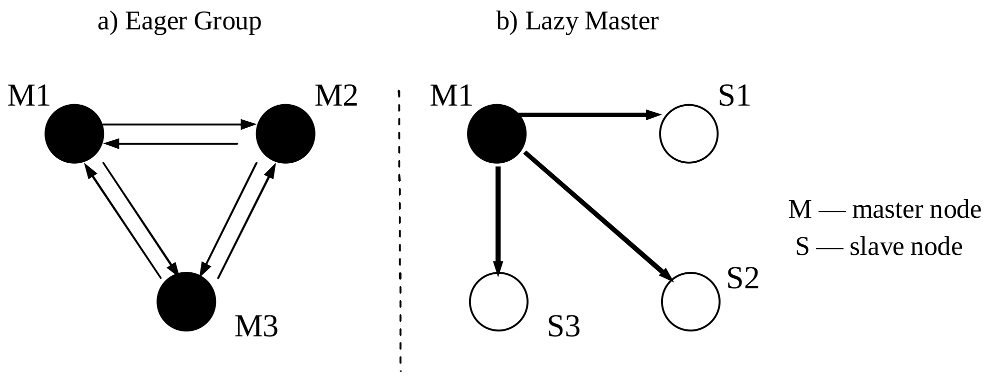

Fig. 1. Two replication schemes — Eager Group and Lazy Master

3.2. Lazy Master Replication

Lazy replication (named also asynchronous replication) is the most prevalent replication scheme in distributed database systems. In this kind of replication a transaction may be committed after upgrading one copy of replica. After transaction committing, upgrades are propagated to other replicas, which are up-to-date during separate refreshing transactions. An important feature of asynchronous replication appears from those replicated schemes in all nodes of environment which obtains the same state of consistency after a while.

A special case of asynchronous replication is Lazy Master (Primary Copy), presented in Fig. 1b. It is characterised by one copy of replica determined as a primary copy (which is stored in the main node M) and transaction updating on this replica only. Updates performed over a primary copy are propagated to other replicas in slave nodes S. In Lazy Master systems, likewise in Eager Group systems, failures reconciliation processes are not present and conflicts are caused by waiting and locking states [5].

4. Two-tier replication scheme

Lazy Group replication systems permit failure reconciliation also in scaled-up. Manual reconciliation of colliding transactions is unrealisable. The only solution is to abort the whole transaction, which needs to conciliate, and support any other update transactions. This causes that transactions are atomised, consisted and isolated, but non-durable. In this system, each transaction is preliminary as long as any other updates of replica will be propagated. In two-tier replication scheme two kinds of node may be distinguished — master and slave [6].

The architecture of a master node, which consists of system of database managing and replication module, is presented in Fig. 2. DBMS consists of local transaction manager LTM and transaction diary Log, which registers all changes performed by local requests over database. The replication module consists of a communication module, a consistency control module (CCM), a global transactions manager (GTM) and an initial transactions manager (ITM).

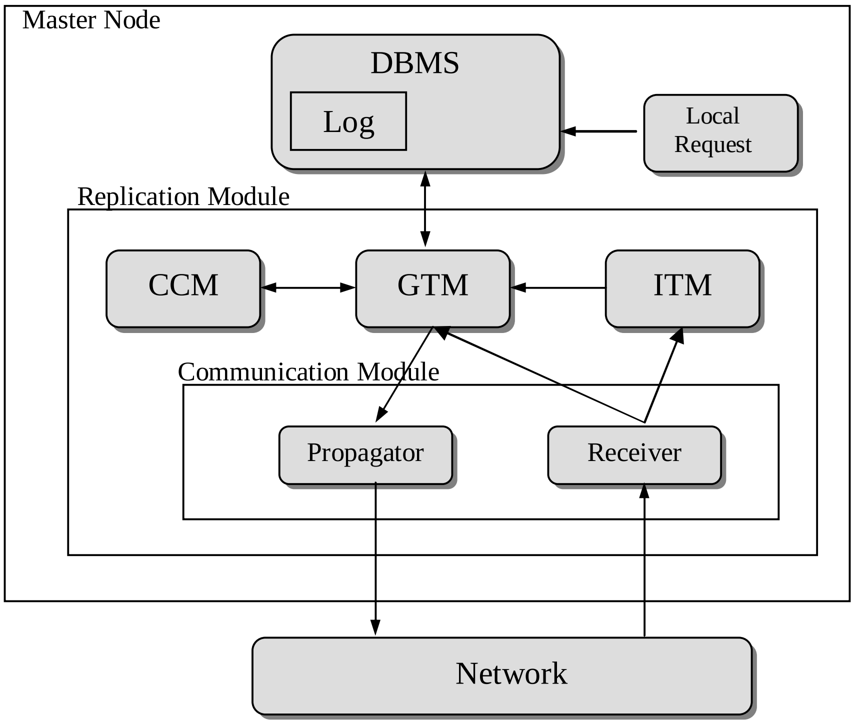

Fig. 2. Architecture of a master node

The communication module consists of Propagator and Receiver components, which realise the function of receiving and sending updates messages for master nodes as well as for slave nodes.

Transactions may be propagated to GTM or ITM depending on their kind. Transactions from other master nodes are transferred by Receiver to the main manager of transactions, which sends the request of transaction performing to local transactions manager. In the case of initial transaction requested by slave nodes, Receiver transfers them to initial transactions manager. Transactions committed by ITM are transferred to LTM and to the other master and connected slave nodes with help of GTM.

5. Transaction model

The transaction model determines the properties of transactions, which have an access to copy of replica in any node. Each transaction consist of reading operation $r_i(X)$ or writing operation $w_i(X)$ performed over objects (X). Transactions are realised atomically, which means that transaction commits c_i or aborts a_i results of all operation itself.

In the two-tier system we can distinguish two kinds of transactions — base and initial. An initial transaction is created in a slave node and propagated to MWT of master node. In case of commitment, the initial transaction is transferred to GMT, where is executed as a base transaction.

Any conflicts of base transactions are regulated by consistency control protocols in the way of serialisation of transactions. In order to guarantee the order of correct execution, colliding transactions have to be separated from each other. Different levels of agreement are used. Those levels are a compromise between correctness and efficiency in maximization of a agreement level by reducing transaction conflict profile. Base transactions are performed with a single serialisation copy; therefore state of base system is a result of serialisation. The transaction becomes durable, when the base transaction is finished, so replicas of all connected nodes are convergent to the base state of the system.

6. Replication process

In a slave node, utility software makes changes over local data by generation of initial transactions. As a result of those changes, differences between data in master and slave nodes occur and generate initial versions of data. The changes performed are recorded into the initial transaction history also called the Log. For any initial transaction, a timestamp is added in order to maintain the data consistency and to solve the conflicts during updating by the initial transaction manager of a master node. Beside timestamp, there is a type of operation stamp («I», «U» or «D» respectively for Insert, Update or Delete operations). After specified time (for local nodes it may be a constant time interval), initial transactions are capsulated into bundles and sent to the master level for commitment as a base transaction. The bundles have a parametric character; therefore their size may be evaluated experimentally due to the efficiency of the system. The encapsulation and sending of transaction take place on a record level, which means that after modification of 100 records of replicated table, a master node will receive 100 messages of record changing. The transaction point is set in the transaction diary and it informs in which point of diary the next step of data propagation starts.

In the initial transaction manager of a master node the processes of correctness checking and optimisation of new transactions are performed. Because updating is performed on a record level the timestamps of analysed transactions are checked first. If the content of the initial transaction timestamp is older than the content of the base transaction timestamp the updating is getting out-of-date and is aborted. Further initial transactions are checked basing on repetition criteria. In this approach, the transactions performing changes over the same record are checked whether they are two or more in a particular bundle of transactions. If so, the smallest number of indispensable transactions is chosen basing on the following algorithm. If two or more transactions will update the same record respectively only the last one is being realised and the others are aborted. After initial choosing, the transactions are transferred to the global transaction manager in order to perform as base transactions. GTM requests locking of all updated object in all master nodes and after so the initial transaction is being realised over all nodes during the same base transaction. After transaction realisation, all locks are released. The committed base transactions are also transferred to slave nodes during its updating. A slave node rejects initial versions of an object and accepts updates from a replica in a master node. Next, the transaction point is set to point on the last transaction of realised update. The state of data in a slave node is convergent to the state of data in a master node when all base transactions of replica update will be done.

7. Algorithm for optimisation

In ITM, the repetition of criteria is checked for each operation on a particular record during initial transaction processing.

The algorithm of optimisation presented below is used for realisation of a single operation instead of a few following operations, which concern the same record. As it was said, any operation over data is performed on the record level, so the algorithm returns the same result as all operations performing over particular record. Basic assumption is to realise the first and the last SQL operation at the same sequence as performing operation over a table and the result of those two operations is determined by realisation of the following operation. After reading operation type stamps, two of following transactions, which concern the same record, are compared according to criteria from Table.

Algorithm for optimisation. Table of operations

First / second	Insert$_i$(V_2)	Update$_i$(V_2)	Delete$_i$
Insert$_i$(V_1)	*X*	Insert$_i$(V_2)	Nothing
Update$_i$(V_1)	*X*	Update$_i$(V_2)	Delete$_i$
Delete$_i$	*X*	*X*	Delete$_i$

The first operation is presented in the first column of Table and the following operation is presented in the first row. The Insert and Update operations have parameters V_1 and V_2, which determine data introduced into a record R_i. The fields marked «*X*» show the operations, which cannot be realised after each other. The result of two operation comparing is used as an argument of the first operation of the next comparison. This process continues until the processing of all operations concerning particular record will be finished. The timestamp of the last operation of a processed group will be assigned to the result of comparison independently from which operation will be chosen as a result of comparison.

The following example shows all phases of comparison operation for particular record and the final result achieved.

Example 1
[Delete$_i$, Insert$_i$(V_1)], Update$_i$(V_2), Delete$_i$ $\rightarrow$
[Update$_i$(V_1), Update$_i$(V_2)], Delete$_i$ $\rightarrow$
[Update$_i$(V_2), Delete$_i$] $\rightarrow$
Delete$_i$ $\rightarrow$ final operation – deleting of record

The final result of example 1 and the last operation realised is a Delete operation. A special case, which is also shown in a table above, is the presence of Insert and Delete operations after each other. In this situation these two operations are omitted and the final result depends upon the previous operation only.

Example 2
[Update$_i$(V_1), Delete$_i$], Insert$_i$(V_2), Delete$_i$ $\rightarrow$
[Delete$_i$, Insert$_i$(V_2), Delete$_i$] $\rightarrow$
Delete$_i$ $\rightarrow$ final operation – deleting of record

Example 3
[Insert$_j$(V$_1$), Delete$_j$] $\rightarrow$
nothing $\rightarrow$ lack of operation over record

In example 2, only the second operation (Delete) is being realised and this operation deletes a particular record R_i, whereas, in the third example operation will be omitted.

8. Conclusions

A scheme of replication presented in this paper solves basic problems in Eager Group and Lazy Master replication schemes. It assures as follows:

– high availability and scalability with avoiding non-stability;

– allows servicing of remote nodes;

– provides a single copy of serialisation of a realised transaction;

– maintaining of convergence of data with avoiding illusion of a system;

– keeping of realised transactions durability by commitment of initial transactions into the form of base transactions and its realisation in all available nodes.

The optimisation and the commitment of initial transactions is performed by efficient servers (master nodes) and thus minimising workload of slave nodes. In the case of very large workload of master nodes or using less efficient units in those nodes, delays may occur. Such delays take place at waiting for locks in other master nodes.

1. *Gray J., Helland P., O'Neil P., Shasha D.* The Danger of Replication and a Solution // ACM SIGMOD Int. Conf. on Management of Data. — Montreal, 1996.

2. *Bębel B., Wrembel R.* Replikacja danych w bazach danych Oracle9i // VII Seminarium PLOUG. — Warszawa, 2003.

3. *Bernstein P., Hadzilacos V., Goodman N.* Concurrency Control and Recovery in Database Systems. — Massachusetts: Addison Wesley, 1987.

4. *Agrawal D., Alonso G., A. El Abbadi, Stanoi I.* Exploiting Atomic Broadcast in Replicated Database // EURO-PAR Int. Conf. on Parallel Processing. — 1997.

5. *Paciti E., Minet P., Simon E.* Fast Algorithms for Maintaining Replica Consistency in Lazy-Master Replicated Database // Int. Conf. of Very Large Data Bases. — Edinburgh, 1999.

6. *Kemme B., Alonso G.* Database Replication based on Group Communication. Technical Report. — Department of Computer Science. — Zurich, 1998.

Received 15.07.2004