CIRS: A State-Conscious Concurrency Control Protocol for Replicated Real-Time Databases

Vishal Pathak
M.M.M.Engg.College
Gorakhpur, India

Ajay Pratap
M.M.M.Engg.College
Gorakhpur, India

Rabin Kr. Singh
M.M.M.Engg.College
Gorakhpur, India

Abhishek Kr. Singh
M.M.M.Engg.College
Gorakhpur, India

Abstract—**Replication [5] is the technique of using multiple copies of a server or a resource for better availability and performance.Each copy is called a replica. The main goal of replication is to improve availability, since a service is available even if some of its replicas are not. This helps mission critical services, such as many financial systems or reservation systems, where even a short outage can be very disruptive and expensive.A prerequisite for realizing the banefits of replication, however, is the devlopement of high performance concurrency machenism. Current applications, such as Web-based services, electronic commerce, mobile telecommunication system, etc., are distributed in nature and manipulate time-critical databases. In order to enhance the performance and the availability of such applications, one of the main techniques is to replicate data on multiple sites of the network. Therefore, the major issue is to develop efficient replica concurrency control protocols that are able to tolerate the overload of the distributed system. In fact, if the system is not designed to handle overloads, the effects can be catastrophic and some primordial transactions of the application can miss their deadlines.**

In this paper we present CIRS (Concurrency control In Replicated realtime Systems) a state conscious concurrency control protocol in replicated distributed environment which is specially for firm realtime database system. CIRS mechanism uses S2PL (Static Two Phase Locking) for deadlock free environment. It also includes veto power given to a cohort after receiving PREPARE message from its coordinator. Also with some more assumptions like sending an extra message in execution phase but after completion of execution at local copy which is described later in this paper the proposed mechanism has a significant increased performance over O2PL and MIRROR in decreasing execution time of the current transaction and it also decreases the waiting time of transactions in wait queue.

I. INTRODUCTION

Distributed real time database systems (DRTDBSs) can be defined as database systems that support real time transactions. A distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. The network must allow the users to share the data i.e. user at location A must be able to access the data at location B. They are used for a wide spectrum of applications such as air traffic control, stock market trading, banking, telemedicine etc. In DRTDBS, there are two types of transactions: global and local. The global transactions are distributed real-time transaction executed at more than one site whereas the local transactions are executed at generating site only. A commonly model used for distributed real time transaction consists of a process, called coordinator, which is executed at the site where the transaction is submitted, and a collection of other processes called cohorts executing at various sites where the required data items reside.

Transactions in a real time database are classified into three types, viz. hard, soft and firm. The classification is based on how the application is affected by the violation of transaction time constraints. This paper reports efficient solutions for some of the issues important to the performance of replicated firm deadline based DRTDBS.

The performance of DRTDBS depends on several factors such as specification of transaction's deadline, priority assignment policy, scheduling transactions with deadlines, time cognizant buffer and locks management, commit procedure etc. One of the primary performance determinants is the policy used to schedule transactions for the system resources. The resources that are typically scheduled are processors, main memory, disks and the data items stored in database.

On possible goal of replication is to have replicas behave functionally like nonreplicated servers. This goal can be stated precisely by the concept of one-copy serializability, which extends the concept of serializability to a system where multiple replicas are present. An execution is one-copy serializable if it has the same effect as a

serial execution on a one-copy database. We would like a system to ensure that its executions are one-copy serializable. In such a system, the user is unaware that data is replicated. There are two approaches to sending a transaction's updates to replicas: synchronous and asynchronous. In the synchronous approach, when a transaction updates a data item, say *x*, the update is sent to all replicas of *x*. These updates of the replicas execute within the context of the transaction. This is called synchronous because all replicas are, in effect, updated at the same time. Although sometimes this is feasible, often it is not, because it produces a heavy distributed transaction load. In particular, it implies that all transactions that update replicated data have to use two-phase commit, which entails significant communications cost. Fortunately, looser synchronization can be used, which allows replicas to be updated independently. This is called asynchronous replication, where a transaction directly updates one replica and the update is propagated to other replicas later. In a replicated environment means when there is more than one copy of a data item distributed on different sites then there is one major issue how to update replicas of data item on different site with efficient concurrency control mechanism. While few works have been done in area of replica concurrency control like 2PL, O2PL and OCC but they are not upto the standard of present requirement.

In this paper we focus on concurrency control and one copy serializability in replicated firm real time database system. In firm real time system the major issue is deadline of completion means if the transcation misses its deadline then the transaction is of no usage, hence it is killed if the trasaction misses its deadline.

2. RELATED WORK

In this section we will concentrate on the work that is under consideration for purposed work and we will also discuss some related replica concurrency control protocols.

2.1. Distributed Two Phase Locking

In this algorithm as described in [1] when a transaction arrives at a site in distributed system it is divided in subtransactions known as cohorts and processes which update replicas of data item are known as replica updaters .If a request by a cohort is a read lock on a data item then any lock is not required on the replicas at remote sites but if the request is a write lock then write locks are required on all the replicas of the dataitem (see figure 1). In this protocol write locks are obtained as the transaction executes, with the transaction blocking on a write request until all of the copies of the data items to be updated have been successfully locked by a local cohort and its remote updaters on replicas. Only the data locked by a local cohort is updated in the data processing phase of transaction. Remote copies locked by updaters are updated after those updaters have received list of items to be updated with the PREPARE message during the first phase of commit protocol. Write locks are only released after they are commited are aborted.

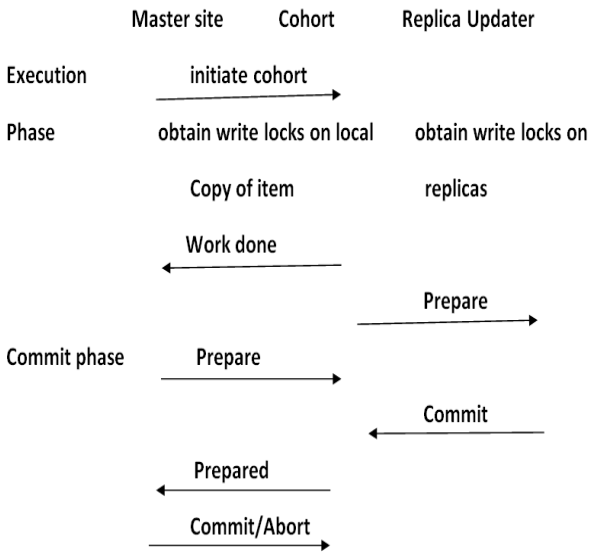

Figure 1

2.2 Distributed Optimistic Two Phase Locking

There is another algorithm as described in [2] in distributed environment for replica concurrency control as O2PL. In this algorithm when a cohort requests for a write lock, it is immediately given to it if lock is available.

However it defers requesting write locks on replicas at remote site in the second phase of commit protocol (as see Figure 2).

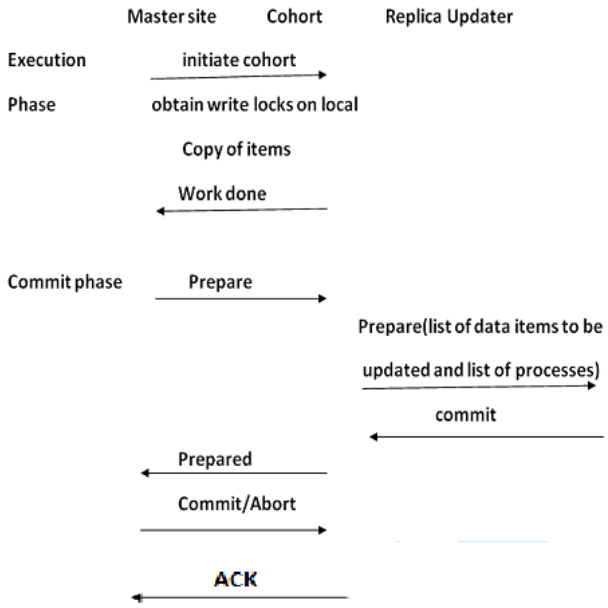

Figure 2

In this protocol when a cohort updates a dataitem it requests for write locks on replicas after it has received PREPARE message from its master site. Actually what happens that after getting PREPARE message from its coordinator the cohort sends a PREPARE message to all of its remote updators of the corresponding dataitem. With the PREPARE message it also sends list of the dataitems to be updated and the processes used in updating the dataitems. After that remote updaters obtain the locks on data item to be updated and sends COMMIT message to the cohort after completing the updation. Now after getting COMMIT message from replica updaters the cohort sends PREPARED message to its coordinator. Since the locks are deferred to the second phase of commit protocol there is a chance of both block and abort also due to arriving of higher priority transaction than the executing one.

3. THE CIRS PROTOCOL

Although S2PL [4] is a deadlock free mechanism but it slows down the concurrent processing of multi transactions. This is due to locking of all the data till the end of the commit phase. Also if a higher transaction arrives at a site than executing one then current transaction is aborted and lock is made available to higher priority one. This makes the wastage resources. Hence we propose here a new mechanism with augmentation of S2PL.

We will use here a term Healthy Point(HP) with Block(B)/Donot Abort(DA) which means if a cohort reaches its Healthy Point(HP) than it will not be aborted against a higher priority transaction at that site. It means DA is used here. And if a lower priority transaction demands a lock then it will be blocked against a higher priority executing one. It means B is used here.

The proposed mechanism is:

HP of a cohort:

A cohort reaches its HP after sending a PREPARE message to its replica updaters in its execution phase i.e. in first phase of 2PC.

HP of a replica updater:

A replica updater reaches HP after gaining locks on needed data items.

By this mechanism some significant improvements can be noted in S2PL.Since after HP a cohort has a less probability of abortion hence a blocked transaction can borrow data from executing one. It means waiting and executing time of blocked transaction will get reduced which is very needed in a firm RTDBMS. Also by sending

PREPARE message to its replica updaters as shown in Figure 3 the waiting time of a cohort between sending PREPARE message to its updaters and receiving COMMIT message from them will get reduced. Hence over all time of execution of transaction will get reduced.

Algorithm:

For a cohort:

HP=false;

EXfinish (execution finished) =false;

If (message=INITIATE COHORT) {

Start execution of cohort;

EXfinish=true;

}

If (EXfinish=true) {

Send PREPARE message to its replica updaters;

HP=true;

Send WORKDONE message to its coordinator;

}

For a replica updater:

If (message=PREPARE&&lock obtained=true) {

HP=true;

After execution send COMMIT message to its cohort;

}

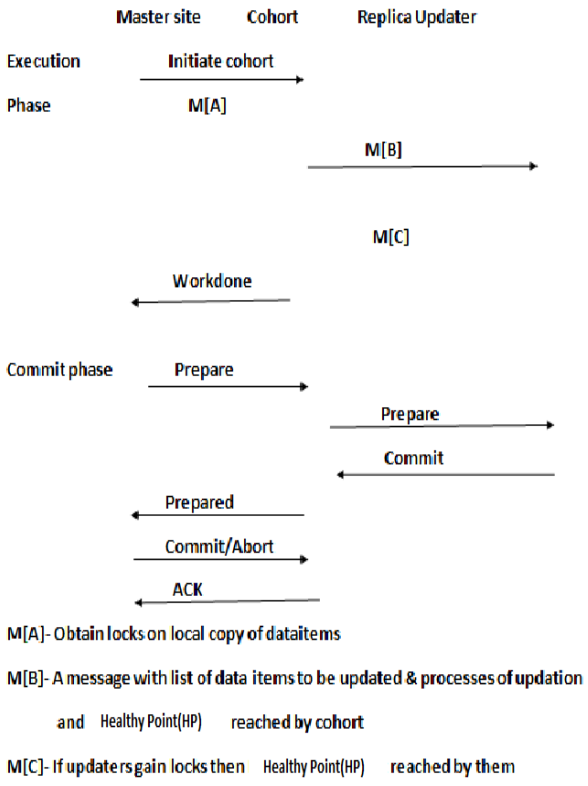

Figure 3

MAIN CONTRIBUTIONS:

1. By this proposed mechanism a deadlock free environment is provided.
2. The waiting time of blocked transaction is reduced.
3. Also the execution time of current transaction is reduced.
4. Wasting of resources is minimized.
5. Easy implementation with integration of S2PL and 2PC.

4. ANALYSIS AND CONCLUSION

In this paper we have taken the problem of accessing replicated data in firm RTDBS mainly concurrency control problem in conflict mode. This proposed mechanism can be easily integrated and implemented in current systems.

However if we talk about advantages of this proposed CIRS protocol over other protocols like O2PL, OCC and MIRROR then this mechanism will lead in some areas like probability of deadlock is decreased. Also by sending M[B] message shown in Figure 3, the waiting time of cohort after sending PREPARE message to its updaters and receiving COMMIT message from them is decreased than in MIRROR due to which whole time of Execution phase and Commit phase will get decreased. Also in this paper it is said that a blocked transaction can borrow data item after reaching the Healthy point by the executing transaction, in this way the waiting time of transaction in queue will get decreased.

Hence from above discussion we recommend this CIRS Protocol over other protocols in replicated environment of firm RTDBS.

5. REFERENCES

[1] Gray,J.,"Notes On Database Operating Systems,"in Operating Systems: An Advanced Course, R. Bayer, R. Graham, and G.Seegmuller,eds.,Springer-Verlag,1979.

[2] Carey, M., and Livny, M., "Conflict- Detection Trade offs for Replicated Data," ACM Transactions on Database Systems, Vol.16,pp.703-746,1991

[3] M. Xiong *et al*. "MIRROR: A State-Conscious Concurrency Control Protocol for Replicated Real-Time Databases", Computer Science Computer Science Department Faculty Publication Series,1999

[4] U. Shanker et *al,"*The SWIFT-Real Time Commit Protocol", International Journal of Distributed and Parallel Databases, Springer Verlag, Vol. 20, Issue 1, 2006, pp. 29-56

[5] El Abbadi, A., Toueg, S., 1989. Maintaining availability I partitioned replicated databases. ACM Trans. Database Syst. 14 (2), 264–290.