Issues in Distributed Real Time Replication Database System

Dr. Ashish Srivastava[*]
Shyam Prakash Singh Kashyap[**]

Abstract

An issue in a distributed real time database system environment time after time involves caching or duplication. In many cases, a distributed device has access not only to data that is stored locally, and much of that data arrives via replication from other devices, Server, and other services. Certain manageable devices with limited resources, weak or discontinuous connectivity, and security vulnerabilities, data replication serves to increase availability, reduce communication costs, forward sharing, and improve survivability of critical information. Such systems generally provide weak consistency models in which read and update operations can be performed at any replica without coordination with other devices. Distributed real time computing is a new emerging computing paradigm of the future. Issues in this model pose many challenging problems to the database area. In this paper we identify these new challenges and plan to investigate their technical consequence.

Keywords : Database, Distributed Real Time database, replicated database, transaction.

1. INTRODUCTION

A real-time system is one that must process information and produce a response within a specified time, else risk severe consequences, including failure. That is, in a system with a real-time constraint it is no good to have the correct action or the correct answer after a certain deadline: it is either by the deadline or it is useless. Database replication based on group communication systems has been proposed as an efficient and flexible solution for data replication. Protocols based on group communication typically rely on a broadcast primitive called atomic [1] or total order [2] broadcast. Replication is the process of repetition and maintaining database objects in multiple databases that make up a distributed database system. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Replication provides user with fast, local access to shared data, and protects availability of applications because alternate data access options exist. Even if one site becomes unavailable, users can continue to query or even update the remaining locations

Another definition in that replication is the process of sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault tolerance, or accessibility. It could be data-replication if the same data is stored on multiple storage devices or computation replication if the same computing task is executed many times. A computational task is typically replicated in space, i.e. executed on separate devices, or it could be replicated in time, if it is executed repeatedly on a single device. The access to replicated entity is typically uniform with access to a single, non-replicated entity. The replication itself should be transparent to an external user. Also, in a failure scenario, a failover or replicas is hidden as much as possible.

The design and implementation of DRTDBS introduce several other interesting problems. Among these problems, predictability and consistency are fundamental to real time transaction processing, but sometimes these require conflicting actions . To ensure consistency, we may have to block certain transactions. Blocking of these transactions, however, may cause

[*] **Deptt. of Computer Application, M.B.S.P.G. College, Gangapur, Varanasi**
[**] **Deptt. of Computer Application, M.B.S.P.G. College, Gangapur, Varanasi**

unpredictable transaction execution and may lead to the violation of timing constraints. There are a number of other sources of unpredictability such as communication delays, site failures [3] and transaction's interaction with the underlying operating system and I/O subsystems . Other design issues of DRTDBS are data access mechanism and invariance, new metrics for database correctness and performance, maintaining global system information, security, fault tolerance, failure recovery etc. Replicated is the key characteristic in improving the availability of data distributed systems. Replicated data is stored at multiple server sites so that it can be accessed by the user even when some of the copies are not available due to server/site failures.[1] A Major restriction to using replication is that replicated copies must behave like a single copy, i.e. mutual consistency as well internal consistency must be preserved, Synchronization techniques for replicated data in distributed database systems have been studied in order to increase the degree of consistency and to reduce the possibility of transaction rollback. [2]

A fully replicated distributed real-time database provides high availability and predictable access times, independent of user location, since all the data is available at each node. Data replication proves the system performance when the data replicas are read operation. Database replication has long been used to interface related applications together. In replicated database systems, copies of the data items can be stored at multiple sites. The potential of data replication for high data availability and improved read performance is crucial to RTDBS. In contrast, data replication introduces its own problems. Access to a data item is no longer control exclusively by a single site; instead, the access control is distributed across the sites each storing the copy of the data item. It is necessary to ensure that mutual consistency of the replicated data is provided.

It is common to talk about active and passive replication in systems that replicate data or services. Active replication is performed by processing the same request at every replica. In passive replication, each single request is processed on a single replica and then its state is transferred to the other replicas. If at any time one master replica is designated to process all the requests, then we are talking about the primary-backup scheme (master-slave scheme) predominant in high-availability clusters. On the other side, if any replica processes a request and then distributes a new state, then this is a multi-primary scheme (called multi-master in the database field). In the multi-primary scheme, some form of distributed concurrency control must be used, such as distributed lock manager.

2. REPLICATION MODELS IN DRTDBS

Distributed database is a collection of databases that can be stored in different computer networks sites. Each database may involve different database management systems and different architectures that distribute the execution of transactions. The object of distributed real time database systems (DRTDBS) is to the control of management of a distributed real time database in such a way that it appears to the user as a centralized database, where the transaction time constraints are imposed by the system. A centralized database systems runs on a single computer system, whereas a distributed real time database system consists of a collection of sites, connected together via some communication network, in which each site is a database system site in its own right but the sites have agreed to a work together, so that a user at any site can access data from anywhere in the network, exactly as if the data are stored at the user's own site in specified time.

Data replication is a common method to reduce the bandwidth consumption and the access latency in distributed systems. The existing replication methods mostly deal with the read-only data, thus the replica management mechanism is relatively simple. Most replication methods replicate files in all nodes or two endpoints on a query path. Naturally, the larger number of replicas, the better performance it can obtain. However, these methods lead to high overhead for unnecessary file replications and consistency maintenance.[3]

2.1 Replication in distributed systems

Whether one replicates data or computation, the objective is to have some group of processes that handle incoming events. If we replicate data, these processes are passive and operate only to maintain the stored data, reply to read requests, and apply updates. When we replicate computation, the usual goal is to provide fault-tolerance. For example, a replicated service might be used to control a telephone switch, with the objective of ensuring that even if the primary controller fails, the backup can take over its functions. But the underlying needs are the same in both cases: by ensuring that the replicas see the same events in equivalent orders, they stay in consistent states and hence any replica can respond to queries.

2.2 Availability & Performance

Replication provides fast, local access to shared data because it balances activity over multiple sites. Some users can access one server while other users access different servers, thereby reducing the load at all servers. Also, users can access data from the replication site that has the lowest access cost, which is typically the site that is geographically closest to them.

2.3 File Based Replication

File base replication is replicating files at a logical level rather than replicating at the storage block level. There are many different ways of performing this and unlike storage level replication; they are almost exclusively software solutions.

File level replication solution yield a few benefits. Firstly because data is captured at a logical level it can make an informed decision on whether to replicate based on the location of the file and the type of file. Hence unlike Disk Storage Replication where a whole volume needs to be replicated, file replication products have the ability to exclude cache, temporary files or parts of a file system that hold no business value. This can substantially reduce the amount of data sent from the source machine as well as decrease the storage burden on the destination machine. A further benefit to decreasing bandwidth is the data transmitted can be more granular than Disk Storage Replication. If an application writes 100byte, only the 100bytes is transmitted converse to a complete disk block which is generally 4k.

Replication Techniques in Distributed Systems organizes and surveys the spectrum of replication protocols and systems that achieve high availability by replicating entities in failure-prone distributed computing environments. The entities discussed in this book vary from passive un-typed data objects, to typed and complex objects, to processes and messages. Replication Techniques in Distributed Systems contains definitions and introductory material suitable for a beginner, theoretical foundations and algorithms, an annotated bibliography of commercial and experimental prototype systems, as well as short guides to recommended further readings in specialized subtopics.

2.4 Point-in-time replication

Point-in-time replication introduces periodic snapshots that are replicated instead of primary storage. If the replicated snapshots are pointer-based, then during replication only the

changed data is moved not the entire volume. Using this method, replication can occur over smaller, less expensive bandwidth links such as ISCSI or T1 instead of fiber optic lines.

2.5 Network load decline

Replication can be used to distribute data over multiple regional locations. Then, applications can access various regional servers instead of accessing one central server. This configuration can reduce network load dramatically.

2.6 group consumption

Replication can be used to distribute data over multiple regional locations. Then, applications can access various regional servers instead of accessing one central server. This configuration can reduce network load dramatically.

Conflict resolution: For instance, if a record is changed on two nodes simultaneously, an eager replication system would detect the conflict before confirming the commit and abort one of the transactions. A lazy replication system would allow both transactions to commit and run a conflict resolution during resynchronization. The resolution of such a conflict may be based on a timestamp of the transaction, on the hierarchy of the origin nodes or on much more complex logic, which decides consistently on all nodes.Database replication becomes difficult when it scales up. Usually, the scale up goes with two dimensions, horizontal and vertical: horizontal scale up has more data replicas, vertical scale up has data replicas located further away in distance. Problems raised by horizontal scale up can be alleviated by a multi-layer multi-view access protocol. Vertical scale up is running into less trouble since internet reliability and performance are improving.

2.7 Active and Passive Replication in Distributed Systems

It is a common to talk about active and passive replication in systems that replicate data or services. Active replication is performed by processing the same request every replica. In passive replication, each single request is processed on a single replica and then its state is transferred to the other replicas. In the distributed systems research area replication is mainly used to provide fault tolerance. The entity being replicated is a process. Two replication strategies have been used in distributed systems: Active and Passive replication.

In active replication each client request is processed by all the servers. This requires that the process hosted by the servers is deterministic. Deterministic means that, given the same initial state and a request sequence, all processes will produce the same response sequence and end up in the same final state. In order to make all the servers receive the same sequence of operations, an atomic broadcast protocol must be used. An atomic broadcast protocol guarantees that either all the servers receive a message or none, plus that they all receive messages in the same order. The big disadvantage for active replication is that in practice most of the real world servers are nondeterministic. In passive replication there is only one server (called primary) that processes client requests. After processing a request, the primary server updates the state on the other (backup) servers and sends back the response to the client. If the primary server fails, one of the backup servers takes its place. Passive replication may be used even for nondeterministic processes. The disadvantage of passive replication compared to active is that in case of failure the response is delayed.

Synchronous replication - guarantees "zero data loss" by the means of atomic write operation, i.e. write either completes on both sides or not at all. Write is not considered complete until acknowledgement by both local and remote storage. Most applications wait for a

write transaction to complete before proceeding with further work, hence overall performance decreases considerably. Inherently, performance drops proportionally to distance, as latency is caused by speed of light.

Asynchronous replication - write is considered complete as soon as local storage acknowledges it. Remote storage is updated, but probably with a small lag. Performance is greatly increased, but in case of losing a local storage, the remote storage is not guaranteed to have the current copy of data and most recent data may be lost.

Semi-synchronous replication - this usually means [citation needed] that a write is considered complete as soon as local storage acknowledges it and a remote server acknowledges that it has received the write either into memory or to a dedicated log file. The actual remote write is not performed immediately but is performed asynchronously, resulting in better performance than synchronous replication but with increased risk of the remote write failing.

3. CHALLENGES IN DISTRIBUTED REALTIME REPLICATION

Data base management systems therefore play a vital role in today's organizations, from their reliability and availability directly depend the overall system dependability. Replication is a well known technique to improve dependability. By maintaining consistent replicas of a database one can increase its fault tolerance and simultaneously improve system's performance by splitting the workload among the replicas. Endeavor information systems are nowadays commonly structured as multi-tier architectures and invariably built on top of database management systems responsible for the storage and provision of the entire business data. Database

We address these issues by exploiting the partial replication of databases. We target large scale systems where replicas are distributed across wide area networks aiming at both fault tolerance and fast local access to data. In particular, we imagine information systems of multinational organizations presenting strong access locality in which fully replicated data should be kept to a minimum and a judicious placement of replicas should be able to allow the full recovery of any site in case of failure. The researches departs from work on database replication algorithms based on group communication protocols, in detail, multi-master certification-based protocols and weak voting protocols. [5 , 6] At the core of these protocols resides a total order multicast primitive responsible for establishing a total order of transaction execution. A well known performance optimization in local area networks exploits the fact that often the definitive total order of messages closely following the spontaneous network order, thus making it possible to optimistically proceed in parallel with the ordering protocol. Unfortunately, this optimization is invalidated in wide area networks, precisely when the increased latency would make it more useful. To overcome this we present a novel total order protocol with optimistic delivery for wide area networks. In a globalize state, an organization to achieve the desired degree of dependability must have replicas of their most sensitive data in several locations of a wide area network. This new kind of environment, with high point to point latency and limited bandwidth, poses several challenges to replicated databases that have been targeted to local environments usually with low latency and without bandwidth restrictions. Another issue relating to database replication that deserves some attention is the data being replicated. Replication increases the huge amount of storage on increase the load on the system. Some time this overloads the system processing and transaction problems. In a globalize organization, some of its data is relevant for all of its branches, but surely there is

other data that is only relevant to one or some of the branches but not to all of the branches. Given that some data is not relevant to some of the branches of the organization; it is questionable why to replicated all data to every replica. Should only relevant data be replicated to each replica, i.e., using partial replication instead of full replication, and it would be possible to achieve a high degree of dependability, and reduce the required network bandwidth and also local storage at each replica. Given the lack of database replication proposals for wide area networks, ensuring strong consistency and supporting partial replication, the goal of this thesis is to address and propose solutions for the problems arising from partially replicated databases and wide area networks.

Replication protocols in the database community concentrate mainly on the data and it's semantic. The properties of the communication primitives and how they can help in ensuring data consistency are usually disregarded. This results in lack of acceptance of protocols ensuring strong data consistency, which are considered too expensive and presenting poor performance. he existence of such primitives eases the development of replication protocols which may rely on the communication primitives to ensure data consistency, without regarding to the data semantics. Recently several research efforts have been developed in order to combine protocols from both communities. These efforts result in group based replication protocols. Results obtained by these protocols are encouraging, giving indicators that, in a replicated database, data consistency does not need to be sacrificed in favor of performance.

A real time distributed database prototyping environment was used to built the reminder program. The environment provides the user with multiple threads of executions and guarantees the consistency of concurrently executing processes several requests can be submitted at the same time & may read/write operation take place simultaneously at different sites.[7]

4 .Issues in Distributed Real Time Database Systems

Issues in Distributed Real Time replicated Database Systems What proven architectural approach should you follow to create nearly-identical copies of the data and to manage the integrity of the copies if they can be updated at both the source and target within a replication. The time expressed in the form of a deadline is a critical factor to be considered in distributed real time transaction [7]. The completion of transactions on or before its deadline is one of the most important performance objectives of DRTDBS. There are several important factors that contribute to the difficulty in meeting the deadlines of a distributed transaction [8]. One of the most significant factors is the data conflict among transactions . A number of real time concurrency control protocols have been proposed in the past. When a data conflict occurs between an executing and a committing transaction, a commit protocol has to work with concurrency control protocol to handle it and to ensure the transaction atomicity. Design issues: Replication set size. Decide whether to replicate an entire table, a subset of a table, or data from more than one table. This is a tradeoff among the amount of data that changes, the overall table size, and the complexity of the link. A Replication set data changes at the target have to occur and if the source wants to see the changes, then try to make the changes naturally non-conflicting to avoid the need for conflict detection and resolution and the Replication frequency are decide the appropriate timing of the replication for the requirements and optimize the use of computing resources[9,10,11].

In replicated database system, copies of the data items can be stored at multiple sites. The potential of data replication for high data availability and improved read performance is

crucial to RTDBS. On the other hand; data replication introduces its own problems. Access to a data item is no longer control exclusively by a single site; instead the access control is distributed across the sites each storing a copy of the data item. It is necessary to ensure that mutual consistency of the replicated data is provided, in other words, replicated copies must behave like a single copy. This is possible by preventing conflicting accesses on the different copies of the same data item, and by making sure that all data sites eventually receive all updates. Multiple copy updates lead to a considerable overhead due to the communication required among the data sites holding the copies. Therefore measure issue is the development of replication protocol / policy. Other design issue of DRTDBS is data access mechanism and invarieance, replication of data items at various sites, new metrics are database for database correctness and performance, maintaining global system information, security, fault tolerance, failure recovery, optimizing the use of memory, deadlines assignment strategies, possibility of distributed deadlocks etc. Also there is no adequately designed technique for scheduling the CPU as the primary resources in DRTDBS.

Many issues affecting the design of A DRTDBS to maintain its requirements; Data Consistency and Scalability are the main issues that are considered in this paper. All of those critical systems need data to be obtained and updated in a timely fashion, but sometimes data that is required at a particular location is not available when it is needed and getting it from remote site may take too long before which the data may become invalid, this potentially leads to large number of transactions miss their deadline and violating the timing constraints of the requesting transaction. One of the solutions for the above-mentioned problem is replication of data in real-time databases. By replicating temporal data items, instead of asking for remote data access requests, transactions that need to read remote data can now access the locally available copies which help transactions meet their time and data freshness requirements. The real-time scheduling part of our scheme has three components: a policy to determine which transactions are eligible for service, a policy for assigning priorities to transactions, and a policy for resolving conflicts between two transactions that want to access the same data object. None of these policies needs any more information about transactions than the deadline and the name of the data object currently being accessed. Transactions that are unable to meet their deadlines are immediately aborted. When a transaction is accepted for service at the local site where it was originally submitted, it is assigned a priority according to its deadline. The transaction with the earliest deadline has the highest priority. High priority is the policy that we employed for resolving transaction conflicts. Transactions with the highest priorities are always favored. The favored transaction, i.e. the winner of the conflict, gets the resources that it needs to proceed. Some of the other issues in DRTDBS are scheduling of distributed transactions, optimizing the use of memory, management of distributed transactions, deadline assignment strategies, difficulty in maintaining global system information, possibility of distributed deadlocks etc. [12].

5. Transaction Processing In Replicated Data in the DDBMS

A transaction is a logical unit of work constituted by one or more SQL statements executed by a single user. A transaction begins with the user's first executable SQL statement and ends when it is committed or rolled back by that user. A remote transaction contains only statements that access a single remote node. A distributed transaction contains statements that access more than one node. A distributed transaction is a transaction that includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a

distributed database. The term replication refers to the operation of copying and maintaining database objects in multiple databases belonging to a distributed system. The time expressed in the form of a deadline is a critical factor to be considered in distributed real time transactions. Completion of transactions on or before its deadlines is one of the most important performance objectives of DRTDBS. There are several important factor that contribute to the difficulty in meeting the deadlines of a distributed transactions.The terms distributed database system and database replication are related, yet distinct. In a pure (that is, not replicated) distributed database, the system manages a single copy of all data and supporting database objects. Typically, distributed database applications use distributed transactions to access both local and remote data and modify the global database in real-time. While replication relies on distributed database technology, database replication offers applications benefits that are not possible within a pure distributed database environment. Most commonly, replication is used to improve local database performance and protect the availability of applications because alternate data access options exist.[13,14,16] For example, an application may normally access a local database rather than a remote server to minimize network traffic and achieve maximum performance. Furthermore, the application can continue to function if the local server experiences a failure, but other servers with replicated data remain accessible. A new component, which is a replication manager module, has been recently added to the system, in order to maintain replicated data[18].

6. Consistency Levels

Synchronization depends on the level at which correctness is sought. This can be roughly categorized as replica-level correctness and transaction-level correctness. At the replica level, correctness or coherency requirements are expressed per item in terms of the allowable divergence among the values of the copies of each item. There are many ways to characterize the divergence among copies of an item. For example, with quasi copies [8], the coherency or freshness requirements between a cached copy of an item and its primary at the server are specified by limiting (a) the number of updates (versions) between them, (b) their distance in time, or (c) the difference between their values. At the transaction level, the strictest form of correctness is achieved through global serializability that requires the execution of all transactions running at mobile and stationary hosts to be equivalent to some serial execution of the same transactions. In case of replication, one copy serializability provides equivalence with a serial execution on a one-copy database. One-copy serializability does not allow any divergence among copies.There is a large number of correctness criteria proposed besides serializability. Central are also criteria that treat read-only transactions, i.e., transactions with no update operations, differently. Consistency of read-only transactions is achieved by ensuring that transactions read a database instance that does not violate any integrity constraints (as for example with snapshot isolation), while freshness of read-only transactions refers to the freshness of the values read [17]. Finally, relaxed-currency serializability allows update transactions to read out-of-date values as long as they satisfy some freshness constraints specified by the users [19].

7. CONCLUSION

It is interesting to consider the entire scenario and how it compares to a true distributed real-time database DRTDBMS solution based on synchronous updating approaches. The first solution point is that the application is not blocked by a problem that is related to distributed real time transactions. It performs a local update, which is quick, while the DRTDBMS

manages the distribution asynchronously in time. The second solution point, of course, is that there is latency between the updates performed at the primary and subsequent copies. This raises application issues which the user needs to be able to live with. A third solution point is that in larger and/or less reliable environments a DRTDBMS approach just would not work, while replication's architecture can. A DRTDBMS would attempt to perform one -2 phase commit transaction with a no of branches. Even if all branches nodes were on-line, the distributed DRTDBMS would hold locks on all branches targets until all branches were willing to commit. In an unreliable WAN scenario or any situation with many nodes, clearly the distributed DRTDBMS solution just flat does not work.

As distributed real time operational online applications become more widely used across large enterprises there is going to be a requirement for increasing numbers of data copies to support timely local response. This is because the propagation uncertainties and costs associated with real time networks and/or DRTBMS solutions are a headache to deal with.

Replication cannot be used where absolute data synchronization is required for the real time application. Replication, or the copying of data in distributed real-time databases to multiple locations to support distributed applications, is an important new tool for businesses in building competitive service advantages. New replicator facilities from several vendors are making this technology much more useful and practical than it's been in the past. The available all the protocols for distributed real time database are not quite sufficient. In this article we will go into enough detail on replication for the reader to understand the importance of replication, its benefits and some of the related technical issues. More work to be done by the researchers to provide a fruitful replicated database solution for distributed real time database fulfilling all the real time constraints.

I. REFERENCES

[1] Hadzilacos and S. Toueg. A modular approach to fault tolerant broadcasts and related problems. Technical Report TR94-1425, Dep. of Computer Science, Cornell University, Ithaca, New York (USA), 1994

[2] P.A. Bernstein, D.W. Shipman, and J. B. R. Jr. Concurrency control in a system for distributed databases (sdd-1). ACM Trans. Database Syst., 5(1):18–51, 1980.

[3] Udai Shanker, “Some Performance Issues In Distributed Real Time Database System”, PhD Thesis, December, 2005.

[4] S. Elnikety, F. Pedone, and W. Zwaenopoel. Database replication using generalized snapshot isolation. In SRDS. IEEECS, 2005.

[5] M, Valduriez P, 1991, Principles of Distributed Database Systems, Prentice-Hall.P. Bernstein, V. Hadzilacos and N. Goodman,

[6] G Schussel, "Replication the nexy generation of Distributed Database technology via Internet www.dci.com/speaker/archive/replica.html.

[7] E. Leontiadis, V.V. Dimakopoulos and E. Pitoura. Creating and Maintaining Replicas in Unstru ctured Peer-to-Peer Systems, EURO-PAR 2006

[8] S. Ceri, M.A.W. Houtsma, A.M. Keller, P. Samarati: A Classification of Update Methods for Replicated Databases, via Internet, May 5, 1994.

[9] Sang Hyuk Son, "Replicated Data Management in Distributed Database Systems", ACM Simgod,, Vol. 17, Issue 4, Dec 1998, Newyork,USA, pp-62-69.

[10] Xin Sun; Jun Zheng; Qiongxin Liu; Yushu Liu; " Dynamic Data Replication Based on Access Cost in Distributed Systems",. Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT '09, Seoul ,2009, pp829 – 834.

[11] G Schussel, "Replication the nexy generation of Distributed Database technology via Internet www.dci.com/speaker/archive/replica.html.

[12] B Kemme and G Alonso, "A new approach to developing and implementing eager database replication protocols", ACM Trans database Systems, 25(3).333-379, 2000.

[13] S.H. Son, "Environment for integrated development and evaluation Real Time Distributed Database Systems", Journal of System Integration, Vol. 2 No. 1, Feb, 1999, pp 67-90.

[14] Robert A and Garcia-Molina H,1992, Scheduling Real-Time Transactions, ACM Trans. on Database Systems, 17(3).

[15] R. Alonso, D. Barbara, and H. Garcia-Molina. Data Caching Issues in an Information Retrieval System, ACM Transactions on Database Systems (TODS), 15(3), September 1990, pp 359-384.

[16] Jayanta Singh and S.C Mehrotra,2009 "A study on transaction scheduling in a real-time distributed system", EUROSIS's Annual Industrial Simulation Conference, UK.

[17] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil., and P. O'Neil. A Critique of ANSI SQL Isolation Levels. In Proceedings of the ACM SIGMOD Conference, 1995, pp1-10

[18] H. Garcia-Molina and G. Wiederhold. Read-Only Transactions in a Distributed Database. ACM Transactions on Database Systems, 7(2), June 1982, pp209-234.

[19] P.A.Bernstein, A. Fekete, H. Guo, R. Ramakrishnan, and P. Tamma. Relaxed-currency serializability for middle-tier caching and replication. In SIGMOD Conference, 2006,pp599–610.

●