ISSN:2229-6093

Shetan Ram Choudhary et al, Int.J.Computer Technology & Applications,Vol 4 (5),785-792

PERFORMANCE EVALUATION OF REAL TIME DATABASE SYSTEMS IN DISTRIBUTED ENVIRONMENT

Shetan Ram Choudhary[1], C.K. Jha[2], Ph.D

Department of Computer Science, Banasthali University, P.O. Banasthali Vidyapith-304022, Rajasthan, India

[1]srchoudhary.1972@gmail.com, [2]ckjha1@gmail.com

Abstract

Transaction scheduling plays an important role in deciding the performance of a real time database system (RTDBS) in distributed environment. It has been demonstrated that the priority based scheduling enhances the performance of a RTDBS in distributed environment. The performance is primarily measured by the number of transactions completed within a unit time. In real-time applications, timing and criticality characteristics of transactions must be taken into account. In this paper, we examine the performance of real time database systems in distributed environment. The deadline guarantee ratio and average response times are the primary performance measures. There have been performance studies on real-time database systems, but most of them were performed using simulation. This work demonstrates the feasibility of developing real time database systems in distributed with an acceptable performance.

Keywords: Real Time Database System, Transaction Scheduling, Distributed Environment.

1. INTRODUCTION

An integrated and shared base of persisted data used by different kinds of users in a variety of programs is said to be database. The management software that handles all requests from users for access to database is called database management system (DBMS). DBMS is also responsible for maintenance of the central base of data. A database buffer had to be maintained for purpose of interfacing main memory and disk. Distributed database management system (DDBMS) consists of collection of sites connected together via some kind of communication network in which –

- Each site is a database system site in it's own right.
- The sites have agreed to work together so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site.

There is a mounting need for real time data services in distributed environment. Modern electronics services and electronic commerce communication applications characterized by high volume of transactions, can't survive without an online support of computer system and updated database technology [57].

Many applications such as factory automation, military tracking, aircraft control, shipboard control, stock arbitrage system, networking management, sensory system, banking system, railway reservation system and traffic control, transaction should be processed within their deadlines using the fresh data in real time environment.

Any system where a timely response by the computer to external stimuli is vital in a real time system (RTS).The presence of multiple sites in distributed environment raise issue that is not present in centralized system. Typically RTS are associated with critical application in human lives or expensive machineries may be at stake. Hence in such system an action performed too late or too early or a computation which uses temporal invalid data may be useless and same time harmful. This type of action or computation is functionally correct. RTS continue to evolve their application, become more and more complex and often required timely access and predictable processing of massive amount of real time data with the need of the changing electronic communication system scenario.

Several of these application providing real time data services in distributed environment are essential. The issues involved in providing predictable real time data services in centralized data base system have researched as distributed real time database system (DRTDBS). The DRTDBS are collection of multiple, logically interrelated database distributed over a computer networks where transaction have explicit timing constraints usually in the form of deadline. In such a system data items must be controlled in order to maintain databases logically, consistency and satisfying timing constraints of various real time activities. The distributed system has few difficulties due to the distributed nature of the transaction which required database consistency.

The Real-time transaction deadlines fall into three categories: hard, firm and soft, which is based on the effect of missing their deadlines [41].

- Hard Real Time Transaction: It must meet its deadline strictly. A missed deadline may result in a catastrophe.
- Firm Real Time Transaction**:** It does not result in a catastrophe, if the deadline is missed. However, the results have no value after the expiry of deadline.
- Soft Real Time Transaction**:** In this system nothing catastrophic happens if some deadlines are missed but the performance will be degraded below acceptable level still substantial fraction of design efforts in these systems goes into making sure that task deadlines are met.

There are basically two types of distributed transaction execution models; viz., sequential and parallel [4, 41]. In sequential execution model, there can be at most

one cohort of a transaction at each execution site, and only one cohort can be active at a time. After successful completion of one operation, next operation in the sequence is executed by the appropriate cohort. At the end of execution of the last operation, the transaction can be committed. In parallel execution model, the coordinator of the transaction spawns all cohorts together and sends them for execution at respective sites [7]. All cohorts then execute in parallel. The assumption here is that the operations performed by one cohort during its execution at one site are independent of the results of the operations performed by some other cohort at some other site. In other words, the sibling cohorts do not share any information among themselves [54].

The implementation of Distributed Real Time Database Systems (DRTDBS) is difficult due the conflicting requirements of maintaining data consistency and meeting transactions deadline. The difficulty comes from the unpredictability of the response time of the transactions. Each distributed transaction processing a data item takes a variable amount of time due to

- concurrency control
- I/O and communication delays.

While maintaining the consistency of underlying database, scheduling and management of the system resources in DRTDBS should also take into account the timing constraints. Access to CPU, main memory, I/O devices and shared data should be managed to make the best effort to satisfy the transaction deadlines.

2. REVIEW OF LITERATURE

Transaction is a logical unit of work. It is also known as logical unit of recovery/logical unit of integrity. Each transaction should posses four important properties also known as the **ACID** (atomicity, consistency, isolation, durability) properties.

Atomicity: (All or none) Atomicity ensures that either all of transactions actions complete successfully or all of its effects are absent. **Consistency:** (Transaction preserves database consistency) Consistency ensures that a transaction when executed by itself without interference from other transactions maps the database from one consistent state to another. **Isolation:** (Transactions are isolated from one another) Isolation ensures that no transaction ever views partial effects of some other transaction, even when actions of transactions execute concurrently. **Durability:** Durability ensures that once a transaction happens, its updates survive, even if there are a subsequent system crash.

Concurrency consists of the lost update problem, the uncommitted dependency problem, the inconsistent analysis problem. Pessimistic Methods are used as locking and time stamping and locking for concurrency control are as exclusive locks and shared locks.

2.1 REAL TIME DATABASES

Real time databases have two properties. First, data has a finite life time after which it is aged out or becomes invalid. Second transactions have a life time after which their returned results are no longer useful and in addition could be harmful or catastrophic to the system if not returned within the specified lifetime called its deadline. Real-time transaction deadlines fall into two categories: hard and soft. A hard deadline is one cannot be violated. A transaction with hard deadline loses all values if not completed on time, possibly resulting in some catastrophic events occurs, soft deadline transactions will retain some of their value if deadline is missed.

2.2 REAL TIME DATABASE SYSTEMS IN DISTRIBUTED ENVIRONMENT

Real time systems are those for which correctness depends not only on the logical properties of the produced results, but also on the temporal properties of these results [5]. The database systems which are especially designed for the efficient processing of these types of real time data are referred to as distributed real-time database systems (DRTDBS).

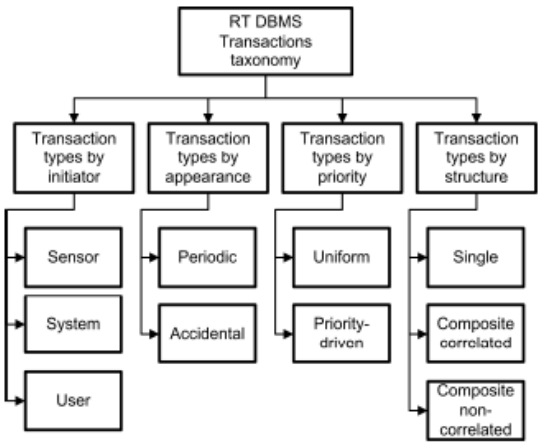

Fig.1 Transaction model in Real Time DBMS.

Distributed real time database systems can not be viewed as a combination of conventional DDBMS and RTS [56] see Fig.1, it has to process distributed transactions and guarantees their basic correctness criteria [8]. DRTDBS are collection of multiple, logically interrelated databases distributed over a computer network where transactions have explicit timing constraints usually in the form of deadlines.

2.3 REAL TIME TRANSACTION IN DISTRIBUTED ENVIRONMENT

When users programs interact with database, partially ordered sets of read and write operations are generated [17]. This sequence of operations on the database is called a transaction.

Hong-Ren Chen and Y.H. Chin was worked on the framework of a distributed real time database system.[58]

The distributed real time transaction processing is a form of transaction processing that supports transactions whose operations are distributed among different computers or among different databases from different vendors. So in a distributed real time transactions, the operations are executed at the site where the required data item resides and is associated

with time constraints. Transfer of money from one account to another, reservation of train tickets, filling of tax returns, entering marks of a student's grade sheet, etc. are some of the examples of distributed real time transactions. The transaction is an atomic unit of work, which is either completed in its entirely or not at all. Hence a distributed commit protocol is needed to guarantee the uniform commitment of distributed transaction execution [52]. The Commit operation implies that the transaction is successful, and hence all of its updates should be incorporated into the database permanently. An abort operation indicates the transaction has failed, and it requires the database management system to cancel all of its effects in the database system. In short, a transaction is an "all or nothing" unit of execution.

2.4 PRIORITY ASSIGNMENT POLICY

A real time database system is a part of a large and complex real time system. The tasks in real time system and transactions in distributed real time database systems are similar in the sense that both are units of work as well as units of scheduling [30, 32, 33, 58]. However, tasks and transactions are different computational concepts and their difference affect how they should be scheduled and processed. Unlike transactions, task in real time systems do not consider consistency of the data items used. Though many real time task scheduling techniques are still used for scheduling real time transactions, the transaction scheduling in the real database systems needs a different approach than that of which is used in scheduling tasks in the real time systems.

Liu and Layland [40] have developed a rate monotonic static assignment scheme to determine the schedulability of a set of periodic tasks for centralized RTS. The proposed priority assignment techniques can be broadly classified into three categories: static, dynamic and hybrid. A scheduling algorithm is said to be *static* if priorities are assigned to tasks once and for all. A scheduling algorithm is said to be *dynamic* if the priority of a task changes from request to request. One of the most used algorithms belonging to this class is Earliest Deadline First (EDF), according to which priorities assigned to tasks are inversely proportional to the absolute deadlines of active jobs where deadline of a job depends on the arrival time of its next occurrence. A scheduling algorithm is said to be *hybrid* if the priorities of some of the tasks are fixed and priorities of the remaining tasks vary from request to request. Though many real time task scheduling techniques are still used for scheduling real time transactions, the transaction scheduling in real time database systems needs a different approach than that used in scheduling tasks in real time systems.

The performance of different scheduling policies for soft deadline based transactions was first addressed by Abbot R. and Garcia-Monila H. [2]. They have conducted study on the performance of three priority assignment techniques: FCFS, EDF and LSF, with different concurrency control methods namely serial execution (SE), high priority (HP), and conditional restart (CR) through simulation. The pioneering work in RTDBS performance evaluation of various scheduling options for a real time database system with disk and shared locks is reported again by Abbot R. and Garcia-Monila H. [1]. The scheduling algorithms used for this study are FCFS, EDF and LSF along with the concurrency control algorithms such as wait, wait-promote, high priority & conditional restart.

Pang et al. investigated the problem of "bias" against longer transactions under "earliest-deadline-based" scheduling policies in a centralized RTDBS [45, 46]. Their approach to solve the problem of bias assigns virtual deadlines to all transactions. A transaction with an earlier virtual deadline is served before one with a later virtual deadline. The virtual deadline of a transaction is adjusted dynamically as the transaction progresses and is computed as a function of the size of the transaction.

In a real-time database system, an application may assign a value to a transaction to reflect the return it expects to receive if the transaction commits before its deadline [24 25]. Haritsa et al. [26] addressed the problem of establishing a priority ordering among transactions characterized by both values and deadlines that results in maximizing the realized value. They proposed the Adaptive Earliest Deadline (AED) protocol for priority assignment as well as for load control of the transactions. AED was later improved to Adaptive Earliest Virtual Deadline (AEVD) policy using virtual deadline based on both arrival time and deadline. Datta et al. addressed some of the weaknesses in AEVD, and proposed the Adaptive Access Parameter (AAP) method for explicit admission control [12].

Dogdu Erdogan and Ozsoyoglu Gultekin proposed new priority assignment and load control policies for repeating real-time transactions [15]. Based on the execution histories of the transactions, they showed that a widely used priority assignment technique EDF is biased towards scheduling short transactions favorably and proposed protocols that attempt to eliminate the discriminatory behavior of EDF by adjusting the priorities using the execution history information of transactions. They introduced the notion of "fair scheduling" of transactions in which the goal was to have "similar" success ratios for all transaction classes (short to long in size).

The problem of assigning deadlines to the parallel and the serial subtasks of the complex distributed tasks is addressed by Kao B. and Garcia-Monila H. [31]. They studied the problem of automatically translating the deadline of a real time activity to deadlines for all its sequential and parallel sub tasks constituting the activity.Lam et al. have conducted study on the effects of different priority assignment heuristics using optimistic concurrency control protocol and high priority two phase locking [36, 39].

To reduce the miss percentage of transactions and the wastage of time for remote transaction due to communication delay, a new real time scheduler called Flexible High Reward (FHR) is proposed by Chen Hong-Ren et al. [10].

2.5 COMMIT PROTOCOL

A distributed transaction is executed at more than one site. In such an environment, the transaction may decide to commit at some sites at some other sites it could decides to abort resulting in a violation of transaction atomicity [42, 48]. To overcome this problem, distributed database systems use a distributed commit protocol which ensures the uniform commitment of the distributed transaction. i.e. all the participating sites agree on the final outcome (commit/ abort) of the transaction [6, 38]. Commit protocol ensures that either all the effects of the transaction persist or none persist despite of the site or communication link failures and loss of message.

- Two Phase Commit Protocol (2PC)
- Three Phase Commit Protocol (3PC)

2.6 REAL TIME COMMIT PROTOCOLS

Due to series of synchronous message and logging cost, commit processing can result in a significant increase in the transaction execution time. In a real time environment, this is clearly undesirable. It may also result in priority inversion, because, once a cohort reaches the prepared state, it has to retain all its data locks until it receives the global decision from the coordinator. This retention is fundamentally necessary to maintain atomicity. Therefore, if high priority transaction requests access to a data item that is locked by a "prepared cohort" of lower, it is not possible to forcibly obtain access by pre-empting/ aborting the low priority cohort. In this sense, the commit phase in DRTDBS is inherently susceptible to priority inversion. More importantly, the priority inversion interval is not bounded since the time duration, that a cohort is in the prepared state, can be arbitrarily long. This is especially more problematic in distributed context. Therefore, in order to meet the transaction deadlines, the choice of a better commit protocol is very important for DRTDBS. For designing the commit protocols for DRTDBS, we need to address two questions.

(i) How do we adopt the standard commit protocols into real time domain/

(ii) How can be decrease the number of missed transactions in the system?

Researchers have proposed some real time commit protocols in the literature to address this issue. Soparkar et al. have proposed a protocol that allows individual sites to unilaterally commit [53].

A centralized timed 2PC protocol guarantees that the fate of a transaction (commit or abort) is known to all the cohorts before the expiry of the deadline when there are no processor, communication or clock faults [13, 14].

According to study of Ramesh Gupta et al., the relative performance of different commit protocols [18-23, 27]. Using a detailed simulation model for firm-deadline DRTDBS, the authors have evaluated the deadline miss performance of a variety of standard commit protocols including 2PC, PA, PC and 3PC. Then they have proposed and evaluated the performance of a new commit protocol called OPT designed specifically for the real-time environment [3,18,21].

Harista et al. proposed a new protocol Permits Reading of Modified Prepared-Data for Timeliness (PROMPT) that is also designed specifically for the real-time environment and includes features such as controlled optimistic access to uncommitted data, active abort, silent kill and healthy lending [28,29,35].

Lam et al. proposed deadline-driven conflict resolution (DDCR) protocol which integrates concurrency control and transaction commitment protocol for firm real time transactions [34, 37].

Pang Chung-leung and Lam K. Y. proposed an enhancement in DDCR called the DDCR with similarity (DDCR-S) to resolve the executing-committing conflicts in DRTDBS with mixed requirements of criticality and consistency in transactions [44].

Based on PROMPT and DDCR protocols, B. Qin and Y. Liu proposed double space commit (2SC) protocol [47]. They analyzed and categorized all kind of de-pendencies that may occur due to data access conflicts between the transactions into two types commit dependency and abort dependency.

Ramamritham et al. [50] have given three common types of constraints for the execution history of concurrent transactions. The paper [9] extends the constraints and gives a fourth type of constraint. Then the weak commit dependency and abort dependency between transactions, because of data access conflicts, are analyzed. Based on the analysis, an optimistic commit protocol Two-Level Commit (2LC) is proposed, which is specially designed for the distributed real time domain. It allows transactions to optimistically access the locked data in a controlled manner, which reduces the data inaccessibility and priority inversion inherent and undesirable in DRTDBS.

2.7 MEMORY OPTIMIZATION

The important data base system resources are the data items that can be viewed as logical resource, and CPU, disks and the main memory which are physical resources [16]. Though the cost of the main memory is dropping rapidly and its size is increasing, the size of database is also increasing rapidly. In real time applications, where databases are of limited size or are growing at a slower rate than the main memory capacities are growing, they can be kept in the main memory. However there are many real time applications that handle large amount of data and require support of an intensive transaction processing [49]. The amount of data they store is too large to be stored in the non volatile main memory. Examples include telephone switching, satellite image data, radar tracking, media servers, etc. In these cases, the database cannot be accommodated in the main memory easily. Hence many of these types of database systems are disk resident. The buffer space in the main memory is used to store the execution code, copies of files and data pages and temporary objects produced. With the new functionalities and features of the light weight devices, there is a need of new policy/ protocols so that

memory utilisation can be improved [51]. Ramamritham K. and Sen R. utilized a novel storage model, ID based storage, which reduces storage costs considerably. They present an exact algorithm for allocating memory among the database operators. Because of its high complexity, a heuristic solution based on the benefit of an operator per unit memory allocation has also been proposed.

2.8 REAL - TIME BUFFER MANAGEMENT

Data storage in a database management system is organised into hierarchy, the lowest level of the hierarchy being secondary storage (disk), followed by primary memory and high speed cache. A DBMS must optimize these storage areas organization and access with constraint policies to facilitate increased data manipulation throughput and to provide real time support [55]. Primary memory is organised into a limited number of physical pages which are shared between active transactions, these pages and their management form the critical link between the databases high level functions (concurrency control, transaction processing, recovery, etc.) and the physical realization of data within the logical database as viewed by these upper level functions. For each logical page reference, the Buffer manager performs the following tasks [43]

(i) The buffer is searched for requested page.
(ii) If the page is not currently in primary memory buffers, the buffer manager must retrieve the page from secondary memory.
  (a) If free frames are available, the page is assigned to a physical memory frame.
  (b) If there are no free frames, a replacement page must be selected and then moved to secondary storage.
(iii) The requested page, once retrieved, is placed in the buffer memory frame.
(iv) The page reference is recorded in the buffer managers page table.
(v) The address of the corresponding buffer frame is returned to the transaction manager to be used in physically accessing data items held on that page.

3. PROPOSED WORK

The performance of the system depends on the factors such as database system architectures, underlying processors, disks speeds, various operating conditions and workloads. 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; however it may cause several unpredictable transaction executions and lead to the violation of timing constraints. There are number of other sources of unpredictability such as communication delays, site failure and transactions 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, maintain global system information, security, fault tolerance, failure recovery, etc.

Although a lot of research has been done on these issues, there still exist many challenging and unresolved issues. Many real-time applications need to share data that are distributed among multiple site. In different applications remote data access consist of multi-hop network operation and take substantially more time than the local data access. Another problem is that due to long remote data access time, by the time a transaction gets all the data it needs, some of the data item may have already become stakes.

(i) The time expressed in the form of dead line is a critical factor to be considered in distributed real time transaction [11].
(ii) The completion of transaction on or before its deadline is one of most important performance objective of DRTDBS.
(iii) One of the most significant factors is the data conflict among transactions. The data conflict that occurs among executing conflict.
(iv) Scheduling of distributed transactions.
(v) Optimizing the use of memory.
(vi) Management of distributed transactions.
(vii) Deadline assignment strategies.
(viii) Possibilities of distributed deadlocks.

Embedding a DBMS and an operating system (OS) environment, in which it is usually treated like a normal application program, can result in aggregating effects on buffer management. If DBMS runs in a virtual address space, program code as well as the DBMS buffer is paged by OS memory management, unless they are made resident in main memory. While replacement of buffer pages is done by the DBMS according to the logical references, paging of main memory frames is performed by independent OS algorithms based on the addressing behavior within the main memory frames. In such an environment, the kinds of faults can occur as: Page faults, Buffer faults, Double-page faults.

Because the DBMS can keep several pages per transaction in fix status, it is possible that a shortage of buffer frames will occur (a resource deadlock); an additional page is requested, yet no page in the buffer can be replaced if all are flagged with FIX status. This situation is especially threatening with small buffer sizes.

Real time database systems and simulation are the fields of special interest in distributed environment. As we have seen that the development of commit protocols for the traditional database system has been an area of intensive research in the past decade. However, in case of real time commit protocols in distributed environment, very little amount of the work has been reported in the literature.

4. METHODOLOGY

The present investigation entitled performance evaluation of real time database system in distributed environment will be developed and then simulation of

strategies/algorithms analyzes the performance base model. Further experiments will be constructed among the base model with varying a few parameters at a time An event driven based simulator is written in C language, to evaluate the performances of protocols [39]. The simulation can use different simulation languages such as GPSS, C++SIM and DeNet. The concurrency control scheme used is S2PL in conjunction with temporary intermediate priority. GPSS [59] is used in our simulation experiment. The deadline of the transaction is determined by the method given as below

$$\text{Deadline } (D_i) = A_i + SF * R_i$$

Where A_i is the arrival time of transaction (T_i) at a site. SF is the slack factor. R_i is the minimum transaction response time.

Multiple database sites can be simulated in a single physical program through establishment of virtual sites. Simulation also allows one to expand the research to study a very large database system comprised of several database sites by setting certain parameters in the simulator. The database system is assumed to consist of several data nodes. The data are logically arranged as pages of memory. The performance metric of the experiments is Miss Percent that is the percentage of input transaction that the system is unable to complete before their deadline. If the transaction action deadline expires either before completion of its local processing, or before the master has written the global decision log record, the transaction is killed and discarded.

The results will be interpreted and comparative study of policies for Distributed Real Time Database Systems development or Simulation of these algorithm will also be done.

5. EXPECTED OUTCOME

The proposed priority assignment scheme may be capable to reduce miss percentage of transactions and will be implemented in distributed real time simulator for main memory resident database.

The performances of developed simulation may be improvement of the few order in transaction miss percentage and will minimize intersite message traffic, execute-commit conflicts and log writes with better response time. It will be compared with commit protocols for both main memory resident and disk resident databases with and without communication delay.

A new locking scheme will be developed for the database model. The performance of developed model will be compared with commit protocol and may be marginally better with these commit protocols in term of miss percentage of the transaction, but it will reduce the memory requirement to a great extent. This will make suitable for data intensive applications with high transaction arrival rate.

6. REFERENCES

[1] Abbott Robert and Garcia-Molina, H., "Scheduling real-time transactions with disk resident data," in Proceedings of the 15th International Conference on Very Large Databases, Amsterdam, The Netherlands, pp. 385–395, 1989.

[2] Abbott Robert and Garcia-Monila, H., "Scheduling real-time transaction: a performance evaluation," in Proceedings of the 14th International Conference on Very Large Databases, pp. 1–12, August 1988.

[3] Agrawal D., Abbadi El, A., Jeffers R. and Lin, L., "Ordered shared locks for real-time databases" International Journals of Very Large Data Bases (VLDB Journal) Vol. 4, Issue 1, pp. 87–126, January 1995.

[4] Agrawal D., Abbadi El A. and Jeffers R, "Using delayed commitment in locking protocols for real-time databases," in Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Diego, CA, pp. 104–113, 2-5 June 1992.

[5] Aldarmi Saud A., "Real Time Database Systems: Concepts and Design," Department of Computer Science, University of York, April 1998.

[6] Al-Houmaily Yousef J. and Chrysanthis P.K., "Atomicity with incompatible presumptions," in Proceedings of the 18th ACM Symposium on Principles of Database Systems (PODS), Philadelphia, June 1999.

[7] Audsley N.C., Burns A., Richardson M.F. and Wellings, A.J., "Data consistency in hard real-time systems," YCS 203, Department of Computer Science, University of York, June 1993.

[8] Bestavros Azer, "Advances in Real Time Database Systems Research," ACM SIGMOD Record, Vol. 24, No. 1, pp. 3-8, 1996.

[9] Biao Q., Yunsheng L. and Jin, C.Y., "A commit strategy for distributed real-time transaction," Journal of Computer Science Technology, Vol. 18, Issue 5, pp. 626-631, 2003.

[10] Chen H.-R., Chin Y.H. and Tseng S.-M., "Scheduling value-based transactions in distributed real-time database systems," in Proceedings of the 15th International Parallel and Distributed Processing Symposium, pp. 978-984, 23–27 April 2001.

[11] Datta Anindya, Son S. H. and Kumar Vijay, "Is a Bird in the Hand Worth More Than Two in the Bush? Limitations of priority Cognizance in Conflict Resolution for Firm Real – Time Database Systems," IEEE transactions on Computers, Vol. 49, No. 5, pp. 482-502, May 2000.

[12] Datta A., Mukhejee S., Konana F., Yiguler I.R. and Bajaj, A., "Multiclass transaction scheduling and overload management in firm real-time database systems," Information System, Vol 21, Issue 1, 29–54, 1996.

[13] Davidson S.B., Lee I. and Wolfe V., "A protocol for timed atomic commitment," in Proceedings of the IEEE 9th International Conference on Distributed Computing Systems, pp. 199-206, 5-9 June 1989.

[14] Davidson S.B., Lee I. and Wolfe V., "Timed Atomic Commitment," IEEE Transactions on Computer, Vol 40, Issue 5, 573-583, 1991.

[15] Dogdu E. and Ozsoyoglu G, "Real-time transactions with execution histories: priority assignment and load control," in Proceedings of the 6th International Conference on Information and Knowledge Management, Las Vegas, NV, pp. 301-308.

[16] Garcia-Molina H. and Salem K., "Main memory database systems: an overview," IEEE Transactions on Knowledge Data Engineering, Vol 4, Issue 6, pp. 509-516, 1992.

[17] Gehani N., Ramamritham K., Shanmugasundaram, J. and Shmueli, O., "Accessing extra database information: concurrency control and correctness," Information System, Vol. 23, Issue 7, 439-462, 1996.

[18] Gupta R., Haritsa J.R., Ramamritham K. and Seshadri S., "Commit processing in distributed real-time database systems," in Proceedings of Real-time Systems Symposium, Washington DC, IEEE Computer Society, San Francisco, December 1996.

[19] Gupta R., Haritsa J.R., Ramamritham K. and Seshadri S., "Commit processing in distributed real-time database systems," Technical Report TR-96-01, Database System Lab, Supercomputer Education and Research Centre, I.I.Sc. Bangalore, India, 1996.

[20] Gupta R., Haritsa J.R. and Ramamritham K., "More optimism about real-time distributed commit processing", Technical Report TR-97-04, Database System Lab, Supercomputer Education and Research Centre, I.I. Sc. Bangalore, India, 1997.

[21] Gupta R., Haritsa J.R. and Ramamritham K., "Revisiting commit processing in distributed database systems," in Proceedings of the ACM International Conference on Management of Data (SIGMOD), Tucson, May 1997.

[22] Gupta R. and Haritsa J.R., "Commit processing in distributed real-time database systems", in Proceedings of the National Conference on Software for Real-Time Systems, Cochin, India, pp. 195-204, January 1996.

[23] Gupta Ramesh, "Commit processing in distributed on-line and real-time transaction processing systems," M.Sc (Engineering) thesis, Supercomputer Education and Research Centre, I.I.Sc. Bangalore, India, 2000.

[24] Haritsa J.R., Carey M.J. and Livny, M., "Data access scheduling in firm real-time database systems," Journal of Real-Time Systems, Vol. 4, Issue 3, 203-242, 1992.

[25] Haritsa J.R., Carey M.J. and Livny M., "Value-based scheduling in real-time database systems", Technical Report TR-1204, CS Department, University of Wisconsin, Madison, 1991.

[26] Haritsa, J.R., Livny M. and Carey M.J., "Earliest deadline scheduling for real-time database systems," in Proceedings of 12th IEEE Real-Time Systems Symposium (RTSS), San Antonio, TX, pp. 232–242, December 1991.

[27] Haritsa J.R., Ramamritham K. and Gupta, R., "Characterization and optimization of commit processing performance in distributed database systems," Technical Report, University of Massachusetts, March 1998.

[28] Haritsa J.R., Ramamritham K. and Gupta R., "Real-time commit processing," Real-time Database Systems: Architecture and Techniques, Kluwer Academic Publishers, Dordrecht, Vol. 593, Kluwer International Series in Engineering and Computer Science, eds. Tei – Wei Kuo, and Kam –Yiu Lam, pp. 227–243, 2001.

[29] Haritsa J.R., Ramamritham K. and Gupta R., "The PROMPT real-time commit protocol," IEEE Transactions on Parallel and Distributed Systems, Vol. 11, Issue 2, pp. 160–181, 2000.

[30] Hong D.-K., Johnson T. and Chakravarthy S., "Real-time transaction scheduling: a cost conscious approach," in Proceedings of the SIGMOD Conference, pp. 197-206, 1993.

[31] Kao B. and Garcia-Molina H., "Subtask deadline assignment for complex distributed soft real-time tasks," Technical Report 93-149, Stanford University, 1993.

[32] Kim Y.-K. and Son S.H., "Predictability and consistency in real-time database systems," in Son, S.H. (ed.), Advances in Real-Time Systems, pp. 509-531, Prentice Hall, New York, 1995.

[33] Kim Y.-K., "Predictability and consistency in real time transaction processing," PhD thesis, University of Virginia, May 1995.

[34] Lam K.-Y., Cao J., Pang C.-L. and Son S.H., "Resolving conflicts with committing transactions in distributed real-time databases," in Proceedings of the Third IEEE International Conference on Engineering of Complex Computer Systems, Como, Italy, pp. 49–58, 8-12 September, 1997.

[35] Lam K.-Y. and Kuo T.-W., "Real-Time Database Systems: Architecture and Techniques," Kluwer Academic, Dordrecht, 2001.

[36] Lam K.-Y., Lee V.C.S., Kao B., and Hung S.L., "Priority assignment in distributed real-time databases using optimistic concurrency control," IEE Proceedings on Computer and Digital Techniques, Vol. 144, No. 5, pp. 324-330, Sept. 1997.

[37] Lam K.-Y., Pang C., Son S.H. and Cao J., "Resolving executing-committing conflicts in distributed real-time database systems," Journal of Computer, Vol. 42, No. 8, pp. 674-692, 1999.

[38] Lee I., Heon Y. and Park T., "A new approach for distributed main memory database systems: a

causal commit protocol," IEICE Transactions on Information System, Vol. E87-D, No.1, pp. 196-204, 2004.

[39] Lee V.C.S., Lam K.-Y. and Kao B., "Priority scheduling of transactions in distributed real-time databases," International Journal of Time-Critical Computing Systems, Vol. 16, pp. 31-62, 1999.

[40] Liu C.L. and Layland, J.W., "Scheduling algorithms for multiprogramming in a hard real-time environment," Journal of the ACM, Vol. 20, No. 1, pp. 46–61, Jan. 1973.

[41] Mittal A. and Dandamudi S.P., "Dynamic versus static locking in real-time parallel database systems," in Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mixco, 26–30 April 2004.

[42] NG Pui, "A commit protocol for checkpointing transactions," in Proceedings of the 7th Symposium on Reliable Distributed Systems, Columbus, OH, USA, pp. 22–31, 10-12 October 1998.

[43] Ozsoyoglu Gultekin and Snodgrass Richard T., "Temporal and Real - Time Databases: A survey," IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 4, pp. 513-532, August 1995.

[44] Pang C.-L. and Lam K.Y., "On using similarity for resolving conflicts at commit in mixed distributed real-time databases," in Proceedings of the 5th International Conference on Real-Time Computing Systems and Applications, 27-29 October 1998.

[45] Pang H.H., Carey M.J. and Livny, M., "Multiclass query scheduling in real-time database systems," IEEE Trans. Knowl. Data Eng. Vol. 7, No. 4, pp. 533–551, 1995.

[46] Pang H.H, "Query processing in firm real-time database systems," PhD thesis, University of Wisconsin, Madison, 1994.

[47] Qin B. and Liu Y., "High performance distributed real-time commit protocol," Journal of Systems Software, Vol. 68, No. 2, pp. 145-152, 2003.

[48] Ramakrishnan R. and Gehrke, J., "Database Management System," McGraw Hill, New York, 2003.

[49] Ramakrishnan R. and Ullaman J.D., "A survey of research on deductive database systems," www-db. stanford.edu/~ullman/dscb/ch1.pdf.

[50] Ramamritham K. and Chrysanthis P.K. "A taxonomy of correctness criteria in database applications," Journal of Very Large Data Bases, Vol. 5, pp. 85–97, 1996.

[51] Ramamritham K. and Sen, R., "DELite: database support for embedded lightweight devices," in EMSOFT, Pisa, Italy, pp. 3–4, September 27–29, 2004.

[52] Ramsay S., Nummenmaa J., Thanisch P., Pooley R.J. and Gilmore S.T., "Interactive simulation of distributed transaction processing commit protocols," in Luker, P. (ed.) Proceedings of Third Conference of the United Kingdom Simulation Society (UKSIM'97), Department of Computer Science, University of Edinburgh, pp. 112-127, 1997.

[53] Soparkar N., Levy E., Korth H.F. and Silberschatz A., "Adaptive commitment for real-time distributed transaction," Technical Report TR-92-15, 1992, Dept. of Computer Science, University of Texas, Austin and also in the Proceedings of the 3rd International Conference on Information and Knowledge Management, Gaithersburg, MD, USA, pp. 187-194, 1994.

[54] Ulusoy, O., "A study of two transaction processing architectures for distributed real-time database systems," Journal of Systems Software, Vol. 31, No. 2, pp. 97-108, 1995.

[55] W. Effelsberg, T. Haerder, "Principles of database buffer management," ACM transactions on Database Systems, Vol. 9, No. 4, pp. 560-595, Dec. 1984.

[56] A. V. Zharkov. Distributed Real-Time Database Management System Prototype for Transaction Handling Methods Simulation. In Proceedings of scientific and technical conference "Microsoft Technologies in Programming Theory and Practice". N. Novgorod, 2007.

[57] Susan B. Davidson and Aaron Watters, "Partial Computation in Real-Time Database Systems", *University of Pennsylvania*, 1988.

[58] Hong-Ren Chen and Y.H. Chin, "Efficient Priority Assignment Policies for Distributed real time database systems", Proceedings of the 2007 WSEAS International Conference on Computer Engineering and Applications, Gold Coast, Australia, January 17-19, 2007.

[59] Minutesmansoftware,GPSS, world North Carolina ,USA,4E.[*GPSS Book*] (Student Version 4.3.5), 2001. http://www.minutmansoftware.com