Some Performance Issues in Distributed Real Time Database Systems

Udai Shanker*, Manoj Misra and Anil K. Sarje

Department of Electronics & Computer Engineering
Indian Institute of Technology Roorkee, Roorkee-247 667, INDIA
udaigkp@gmail.com, {manojfec | sarjefec}@iitr.ernet.in

Abstract

Database performance is an important aspect of database usability. Distributed real time database systems (DRTDBS) must be designed on all levels of database architecture to support timely execution of requests. The primary performance objective in DRTDBS is to minimize the number of missed deadlines. Due to the demanding nature of this objective, traditional approaches are inadequate. However, the research in DRTDBS has been mostly devoted to extending traditional transaction processing techniques to solve the issues important for the design of DRTDBS. In this environment, new policies/protocols must be designed to efficiently handle the transactions execution. Our works involves design of new priority assignment policies and commit protocols and comparison of their performances with existing policies/protocols.

1 Introduction

Many applications such as military tracking, medical monitoring, stock arbitrage system, network management, aircraft control, factory automation etc. that depend heavily on database technology for the proper storage and retrieval of data located at different remote sites have certain timing constraints associated with them. Such applications introduce the need for DRTDBS [12]. A DRTDBS is a collection of multiple, logically interrelated databases distributed over a computer network. They support transactions that have explicit timing constraints. The timing constraint of a transaction is expressed in the form of a deadline, which indicates that it must complete on/before some specific time in future. The transactions can be classified as hard, firm or soft type based on the effect of missing their deadlines [13]. A hard real time transaction must meet its deadline strictly. A missed deadline may result in a catastrophe. A firm real time transaction does not result in a catastrophe, if the deadline is missed. However, the results have no value after the expiry of deadline. A soft real time transaction has some value even after expiry of its deadline, but the value typically diminishes with time.

In contrast to traditional databases, where the primary goal is to minimize the response time of transactions and maximize throughput [14], the main objective of DRTDBS is to minimize the percentage of the transactions that miss their deadlines. The scheduling of real time transaction is far more complex than traditional real time scheduling as the database management algorithms for accessing and manipulating data in DRTDBS should not only ensure database consistency, but should also satisfy the timing constraints. The goal of this effort is to introduce various aspects of DRTDBS, the issues and challenges involved and the work carried out.

The following sections explore the basic issues and research challenges having key importance to the performance of DRTDBS followed by the contributions, major findings of our experimentations and scope for future work.

2 Performance Issues and Research Challenges

The implementation of DRTDBS is difficult due to the conflicting requirements of maintaining data consistency and also meeting transaction's deadlines. The difficulty comes from the unpredictability of the transactions' response times [15]. Each distributed transaction accessing 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

*M. M. M. Engg. College, Gorakhpur - 273010, India under QIP.

Proceedings of the VLDB 2006 PhD Workshop, Seoul, Korea, 2006

devices and shared data should be managed to make the best effort to satisfy the transaction deadlines.

2.1 Priority Assignment Policy

One of the most important issues in design of DRTDBS is transaction scheduling [16]. The transaction scheduling in DRTDBS involves both the CPU scheduling and the data scheduling and is done according to the priorities assigned to the transactions. As a result, the role of the priority assignment policy becomes an important issue in deciding the performance of the system because priorities determine the order of the transactions to access resources which in turn affects their likeliness to meet the deadlines. In traditional databases, when conflicts occur, the preferences tend to be based either on fairness or on resource consumption [14]. However, the transaction scheduling in DRTDBS is done according to the urgency of the transactions that decides their priorities. The priority assignment problem has been addressed by very few researches [17]. Generally, the priority of a transaction is determined on the basis of its deadline such as in earliest deadline first (EDF) priority assignment policy; both fairness and maximum resource utilization become secondary goal. This can cause two major problems. First, more CPU resource is wasted if closer to completion transactions are aborted in favour of higher priority transactions [13]. Second, longer transactions may be harder to finish creating a starvation problem [18]. Execution of a global transaction in a distributed system requires the execution of cohorts on different sites. Most heuristics [19,20,21] for priority assignment in DRTDBS consider that subtasks (cohorts) of a transaction are executed sequentially. Except ultimate deadline (UD), other heuristics are not suitable when the subtasks (cohorts) of a transaction are executed in parallel. The UD also becomes ineffective when data contention is non - trivial [21]. Moreover, Victor C. S. Lee et al. [21] have not studied the fairness property of these schemes.

2.2 Distributed Real Time Commit Protocol

The atomic commit protocols play a key role in supporting global atomicity for the distributed real time transactions [22]. These protocols are used to ensure that all cohorts agree on the final outcome of the transaction. They typically require exchange of multiple messages in multiple phases among the participating sites, and also require to maintain logs of data to recover from failures. This significantly increases the execution time of the transactions and can adversely affect the system's ability to meet transaction deadlines. Due to distributed nature of the transactions and in presence of other sources of unpredictability such as data access conflicts, uneven distribution of transactions over the sites, variable local CPU scheduling time, communication delay, failure of coordinator and cohort's sites etc., it is not easy to meet the deadline of all transactions in DRTDBS. The unpredictability in the commitment phase makes it more serious because the blocking time of the waiting cohorts due to execute-commit conflict may become longer. Hence, due to unique characteristics of the committing transactions and unpredictability in the commitment process, design of an efficient commit protocol is another important issue that affects the performance of DRTDBS.

The Two Phase Commit (2PC) is still one of the most commonly used protocols in the study of DRTDBS. Most of the existing protocols proposed in the literatures are based on it. The 2PC based optimistic commit protocol (OPT) [24] for real-time databases try to improve system concurrency by allowing executing transactions to borrow data from the transactions in their commit stage. This creates dependencies among transactions. If a transaction depends on other transactions, it is not allowed to start commit processing and is blocked until the transactions, on which it depends, have committed. The blocked committing transaction may include a chain of dependencies as other executing transactions may have data conflicts with it. Enhancement has been made in the Permits Reading of Modified Prepared-Data for Timeliness (PROMPT) commit protocol, which allows executing transactions to borrow data in a controlled manner only from the healthy transactions in their commit phase [25]. However, it does not consider the type of dependencies between two transactions. The abort of a lending transaction aborts all the transactions dependent on it. The technique proposed by Lam K. Y. et al. maintains three copies of each modified data item (before, after and further) for resolving execute-commit conflicts [26]. This not only creates additional workload on the system but also has priority inversion problems. Based on the concepts of above protocols [25,26], Biao Qin and Y. Liu proposed a commit protocol Double Space (2SC) [27] which classifies the dependencies between lender and borrower into two types; commit and abort. The abort of a lending transaction only forces transactions in its abort dependency set to abort. The transactions in the commit dependency set of the aborted lending transaction continue as normal. However, 2SC creates inconsistency in case of write-write conflicts [2,11].

The protocols [24,25,26], that allow an executing cohort to borrow data from a committing cohort, do not allow the borrower to send WORKDONE/PREPARED message and block it until the lender commits. These protocols either use blind write model or update model [1].

2.3 Memory Efficient Commit Protocol

Important database system resources are main memory, CPU, disk and data items. Before the start of execution of a transaction, buffer space in main memory is allocated for the transaction. When the main memory is running low, a transaction may be blocked from execution. The amount of memory available in the system thus limits the number of concurrently executable transactions. In large scale real time database systems, the execution of transactions will be significantly slowed down if available memory is low. So, the effective use of available main memory space in data intensive applications is another challenging issue. During the execution of a transaction, temporary records are created to maintain the status of the transaction's execution. These temporary records are kept in the main memory until the transaction commits. This consumes a substantial amount of main memory. Since, unpredictability in the commitment phase may make the transaction to stay for a long period in the system; memory will be held up for a long period and will be not available for other transactions. So, this necessitates the design of commit protocols that save memory by creating less temporary objects.

2.4 Other Challenging Issues

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 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 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. Again, there is also no adequately designed technique for scheduling the CPU as being the primary resource in the DRTDBS.

Although, a lot of research has been done on these issues, there still exist many challenging and unresolved issues. Due to the heterogeneity of these issues, we have confined our work to only some of these issues. Our work involves design of new priority assignment policies and commit protocols and the comparison of their performance with existing policies/protocols. We assumed that the transactions are firm real time and data items accessed by the transactions are known before the start of execution of the transactions. Two locking approaches are used by the transactions to obtain a lock on data items viz., static two phase locking (S2PL) and dynamic two phase locking (D2PL). The deadlock freedom and lower communication overhead of locking by using S2PL makes it attractive for DRTDBS [23]. So, we used S2PL with higher priority concurrency control algorithm to access data in mutually exclusive way.

3 Contributions of Thesis

The work reported in the thesis provides better solutions for some of the issues mentioned above. Major contributions of our thesis may be described as follows:

1. A new scheme to determine the priorities of cohorts executing in parallel along with the method to compute the deadlines of the global and the local transactions have been proposed. In our scheme, each cohort is assigned an initial priority which is inversely proportional to the number of locks required by the cohort at its execution site. A temporary intermediate priority of the cohort is calculated when a data contention occurs and initial priority of newly arrived cohort (T_A) is higher than the priority of lock holding cohort (T_L). The intermediate priorities are based on the remaining execution time needed by T_L and the slack time available with T_A. This minimizes the abort of near completion low priority lock holding cohorts.

   The proposed priority assignment schemes have been compared with EDF priority assignment policy using S2PL concurrency control algorithm and 2SC commit protocol. We have implemented distributed real time simulator for main memory resident database. The simulation results show that the proposed scheme not only ensures fairness within the real time constraints, but also reduces Miss Percentage of transactions ranging from 3% to 10%. The proposed priority assignment scheme is capable to cope with the starvation problem encountered by long transactions. In this case, the improvement in long transaction Miss Percentage is up to 10%.

2. DRTDBS use a commit protocol to ensure transaction atomicity. Most of the existing commit protocols used in DRTDBS try to improve the system performance by allowing a committing cohort to lend its data to a lock requesting cohort, thus reducing data inaccessibility. This creates a dependency between the lender and the borrower. The dependencies created due to read/update type locks have been redefined, and then a static two phase locking with high priority (S2PL-HP) based commit protocol named as SWIFT [3] has been proposed. SWIFT is based on redefined dependencies that are created when a lock holding cohort lends its locked data to some other cohorts for reading or updating. The WORKSTARTED message is sent just before the start of processing phase of the cohort in place of sending WORKDONE message at the end of processing phase [9]. This improves performance of the system by overlapping the transmission time of WORKSTARTED message with the processing time of the cohorts. SWIFT also reduces the time needed for commit processing by allowing commit dependent only borrower to send its WORKSTARTED message instead of being blocked.

To ensure non-violation of the ACID properties, checking of completion of processing and the removal of dependency of cohort are done before sending the YES VOTE message to coordinator by the cohort. The important point of SWIFT is that the required modifications are local to each site and do not require inter-site communications. So, it is free from message overhead [7].

The performances of SWIFT have been compared with 2SC and PROMPT for both main memory resident and disk resident databases with and without communication delay. Results of simulation show a performance improvement of the order of 5% - 10% in transaction Miss Percentage. The performance of SWIFT has also been analyzed for partial read-only optimization, which minimizes intersite message traffic, execute-commit conflicts and log writes consequently resulting in a better response time. The effect of partial read only optimization has been studied both for the main memory and the disk resident databases at communication delay of 0ms and 100ms. The performance improvement in transaction Miss Percentage varies from 1% to 5%.

The impact of permitting the communication between the cohorts of the same transaction (sibling) in SWIFT has also been analyzed both for the main memory and the disk resident database at communication delay of 0ms as well as 100 ms. The cohort sends the abort messages directly to its siblings as well as its coordinator. A little improvement in transaction Miss Percentage was observed, i.e., up to 3%.

3. A new locking scheme has been developed for the database model that permits two types of write operations: blind write and update. In this new locking scheme, a lock not only shows the lock obtained by the lender but also the lock obtained by the borrower. The new locking scheme also ensures that a borrower can't be a lender simultaneously at the same site. This relieves the system from the burden of checking that a borrower is not trying to lend as compared with PROMPT and 2SC. All types of dependencies, that may arise by allowing a committing cohort to lend its data to an executing cohort under both update (read-before-write) model and blind write (write not ever read) model, have been redefined. A memory efficient commit protocol (MECP) has been proposed on the basis of new locking scheme and these all kind of dependencies that may arise by allowing a committing cohort to lend its data to an executing cohort As a result of the new locking scheme, in MECP, each site maintains only a single set of borrowers in comparison to PROMPT and 2SC [5,8], where two different sets are required.

   The performance of MECP is compared with PROMPT and 2SC and is marginally better with these commit protocols in term of Miss Percentage of the transaction, but it reduces the memory requirement to a great extent. This makes it suitable for data intensive applications with high transaction arrival rate where system's main memory size is a bottleneck.

4 Conclusions

In this paper, we discussed major challenging issues important for designing of DRTDBS. Our research addresses three major challenges. The solution of first one introduces new heuristic and temporary intermediate priority assignment policy to determine the priorities of transactions while for second issue, a <u>s</u>tatic two phase locking with higher priority based, <u>w</u>rite-update type, <u>i</u>deal for <u>f</u>ast and <u>t</u>imeliness commit protocol has been proposed. In third case, a new distributed real time commit protocol MECP has been presented that uses a new locking scheme. In nutshell, we have developed, implemented and evaluated the new priority assignment policies/commit protocols to deal with firm real time cohorts executing in parallel fashion. By considering the problem of DRTDBS, we believe that our insights and solutions will significantly contribute to solving the problems.

The work presented in our thesis is only a starting point. Many other issues are still to be resolved and warrant further investigation. Following are some suggestions to extend this work.

- Alternative approaches such as analytical methods and experiments in actual environment can be used to evaluate the effects of the proposed priority assignment policies, deadline computation method and commit protocols on the performance of DRTDBS.
- Our performance studies are based on the assumption that there is no replication. Hence, a study of relative performance of various topics discussed here deserves a further look under assumption of replicated data.
- The integration and the performance evaluation of proposed commit protocols with 1PC and 3PC protocols.
- Although tremendous research efforts have been reported in the hard real time systems in dealing with hard real time constraints, very little work has been reported in hard real time database systems. So, the performance of the proposed work can be evaluated for hard real time constrained transactions.
- Our work can be extended for Mobile DTRDBS, where memory space, power and communication bandwidth is a bottleneck. The MECP will be well suited to hand held devices and possibility of its use for commit procedure can be explored.
- The fault tolerance and the reliability are highly desirable in many real time applications because in these applications, continued operation under

catastrophic failure and quick recovery from failure is very crucial. These aspects may also be dealt.

- In our work, we assumed that each site has a system with a single processor. An obvious extension of our work is for multiprocessor environment.
- More work is needed to explore the impact of communication in between cohorts of the same transaction (siblings) on the overall system performance.
- Our research work can also be extended for grid database systems.

References

[1] Udai Shanker, Manoj Misra and Anil K. Sarje. Distributed Real Time Database Systems: Background and Literature Review. In International Journal of Distributed and Parallel Databases, Springer Verlag (communicated)

[2] Udai Shanker, Manoj Misra and Anil K. Sarje. A Fast Distributed Real Time Commit Protocol. In Journal of Computer Science & Informatics, Computer Society of India, Vol. 35, No. 4, Oct.-Dec. 2005.

[3] Udai Shanker, Manoj Misra and Anil K. Sarje. SWIFT - A Real Time Commit Protocol. In International Journal of Distributed and Parallel Databases, Springer Verlag, Volume 20, Issue 1, July 2006, pages 29-56.

[4] Udai Shanker, Manoj Misra and Anil K. Sarje. Priority Assignment Heuristic and Issue of Fairness to Cohorts Executing in Parallel. In WSEAS Transactions on COMPUTERS, Issue 7, Volume 4, July 2005, pages 758-768.

[5] Udai Shanker, Manoj Misra and Anil K. Sarje. The MEWS Distributed Real Time Commit Protocol. In WSEAS Transactions on COMPUTERS, Issue 7, Volume 4, July 2005, pages 777-786.

[6] Udai Shanker, Manoj Misra and Anil K. Sarje. OCP-the Optimistic Commit Protocol. In Proceedings of the 17th Australasian Database Conference (ADC 2006), Hobart, Tasmania, Australia, Jan. 16-19, 2006 (Published by Australian Computer Society in Conferences of Research and Practice in Information Technology, Vol. 49, pages 193-202, ACS 2006 in association with ACM Digital Library)

[7] Udai Shanker, Manoj Misra and Anil K. Sarje. Dependency Sensitive Distributed Commit Protocol. In Proceedings of the 8th International Conference on Information Technology (CIT 05), Bhubaneswar, India, Dec. 20 - 23, 2005, pages 41-46.

[8] Udai Shanker, Manoj Misra and Anil K. Sarje. A Memory Efficient Fast Distributed Real Time Commit Protocol. In Proceedings of the 7th International Workshop on Distributed Computing (IWDC 2005), Indian Institute of Technology Kharagpur, India, Dec. 27-30, 2005, pages 500-505.

[9] Udai Shanker, Manoj Misra and Anil K. Sarje. Optimizing Distributed Real-Time Transaction Processing During Execution Phase. In Proceedings of the 3rd International Conference on Computer Application (ICCA2005), University of Computer Studies, Yangon, Myanmar, March 9-10, 2005, pages 1-7.

[10] Udai Shanker, Manoj Misra and Anil K. Sarje. Priority Assignment to Cohorts Executing in Parallel. In Proceedings of the 3rd International Conference on Computer Application (ICCA2005), University of Computer Studies, Yangon, Myanmar, March 9-10, 2005, pages 39-45.

[11] Udai Shanker, Manoj Misra and Anil K. Sarje. A New Commit Protocol for Distributed Real-Time Database Systems. In Proceedings of the IASTED International Conference on Databases and Applications (DBA 2005), Innsbruck, Austria, Feb. 14-16, 2005, pages 122-127.

[12] Lee Juhnyoung. Concurrency Control Algorithms for Real - time Database Systems. PhD Thesis, Department of Computer Science, University of Virginia, 1994.

[13] Kao Ben and Garcia - Monila H. An Overview of Real - time Database Systems. In Advances in real - time systems, pages 463 - 486**,** 1995**.**

[14] Yu Philip S., Wu Kun - Lung, Lin Kwei - Jay and Son S. H. On Real - Time Databases: Concurrency Control and Scheduling. In Proceedings of the IEEE, Volume 82, No.1, pages 140 - 157, Jan. 1994.

[15] Ramamritham K. Real-time Databases. Distributed and Parallel Databases, Special Issue: Research Topics in Distributed and Parallel Databases, Vol. 1, Issue 2, pages 199 - 226, April 1993.

[16] Bowers David S. Directions in Databases. In Lecture Notes in Computer Science, 826, Springer - Verlag, pages 23 - 54.

[17] Lee Victor C. S., Lam Kam - Yiu and Kao B. Priority Scheduling of Transactions in Distributed Real - Time Databases. In International Journal of Time-Critical Computing Systems, Vol. 16, pages 31 - 62, 1999.

[18] Huang Jiandong, "Real Time Transaction Processing: Design, Implementation and Performance Evaluation," PhD thesis, University of Massachusetts, May 1991.

[19] Kao Ben and Garcia - Molina H. Deadline Assignment in a Distributed Soft Real - Time System. In Proceedings of the 13th International Conference on Distributed Computing Systems, pages 428 - 437, 1993.

[20] Kao Ben and Garcia - Molina H. Subtask Deadline Assignment for Complex Distributed Soft Real - time Tasks. Technical Report 93 - 149, Stanford University, 1993.

[21] Lee Victor C. S., Lam K. Y., Kao Benjamin C. M., Lam K. W. and Hung S. L. Priority Assignment for Sub - Transaction in Distributed Real - time Databases. 1st International Workshop on Real - Time Database Systems, 1996.

[22] O'Neil Patrick, Ramamritham K. and Pu C. Towards Predictable Transaction Executions in Real - Time Database Systems. Technical Report 92 - 35, University of Massachusetts, August, 1992.

[23] Lam Kam-Yiu. Concurrency Control in Distributed Real - Time Database Systems. PhD Thesis, City University of Hong Kong, Hong Kong, Oct. 1994.

[24] Gupta Ramesh, 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.

[25] Haritsa Jayant R., Ramamritham K. and Gupta R. The PROMPT Real -time Commit Protocol. IEEE Transactions on Parallel and Distributed Systems, Vol. 11, No. 2, pages 160 - 181, 2000.

[26] Lam Kam - Yiu, 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, pages 674 - 692, 1999.

[27] Qin Biao and Liu Y. High Performance Distributed Real - time Commit Protocol. Journal of Systems and Software, Elsevier Science Inc., Vol. 68, Issue 2, November 15, pages 145 -152, 2003.