Performance Transaction's Assessment Of Real Time Database System In Distributed Environment

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

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

Abstract

Database performance is a very important aspect of database usability. The objective of this paper is to proposed policy to forecast the performance of transaction under real time database system in distributed environment. A real time database system in distributed environment is a transaction processing system design to handle the workload within a deadline. The objective of such scheme is to complete the processing of transaction before the deadline expires. The performance of the system depends on the factors like as database system architectures, underlying processors, various operating conditions, disks speeds and workloads. Our works involves of forecasting the transaction performance depends on the basis of comparing with commit and abort of a transition in the scheme to give the result through simulation.

Key words: Real Time Database System, Distributed Sites, Transaction, Deadline, Distributed Environment.

1. INTRODUCTION

Today's Information Era database is an essential component of any Information system and in any environment either it is traditional, network, distributed, real-time. Data and the way it is organized is more important in any system. A database is a structured way to organize information. A real-time database is a processing system designed to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time. Real time applications are increasingly being implemented on different platforms such as centralize, distributed and mobile.

Many real time database applications are distributed in nature [2,6,17]. These include the like as factory automation, robotics, military tracking, aircraft control, shipboard control, stock arbitrage system, communication system, mobile communication systems, medical monitoring, computer integrated manufacturing (CIM), telephone switching, virtual environment, railway reservation, traffic control, sensory and banking systems etc [17,21,5]. The real time performance of Real time distributed database system (RTDBS) depends on several factors like as the database system architecture, disk speed, the underlying processor etc. Proposed model can be used to study the transaction atomicity for real time database system in distributed environment. The proposed model can be used under variety of workloads, setting and workload parameters. We mainly concentrate on the scheduling arrival rate of the workloads applied to the transaction deadline to measure the transaction performance. Distributed computing is a technique that is used to solve a single problem in a heterogeneous computer network system. A major issue in building a distributed database system is the transactions atomicity. When a transaction runs across into two sites, it may happen that one site may commit and other one may fail due to an inconsistent state of transaction. Two-phase commit protocol is widely used [19,2] to solve these problems. The choice of commit protocol is an important design decision for distributed database system. A commit protocol in a distributed database transaction should uniformly commit to ensure that all the participating sites agree to the final outcome and the result may be either a commit or an abort situation.

2. DATABASE SYSTEM IN DISTRIBUTED ENVIRONMENT

Database system (DBS) can be viewed as a collection of the data items which are shared by many users [26,27]. They are designed to manage huge amount of the data. The management of data basically involves the definition of structures for its storage and provision of mechanisms for manipulation of this stored information as per requirement. Thus, a DBS is a collection of objects, which satisfy the need of users besides a set of integrity constraints. Database Systems can be generally categorized as I. Centralized Database System and II. Distributed Database System

I. *Centralized Database System*: The centralized database systems are those that run on a single computer system. These systems may range from single-user database systems running on personal computers to high-performance database systems running on mainframe systems.

II. *Distributed Database System*: The distributed database systems (DDBS) consist of a collection of sites, connected together via some means of communication networks, in which, each site is a database system site in its own right but

the sites have agreed to work together, so that a user at any site can access data from anywhere in the network, exactly as if, the data are all stored at the user's own site [29].

3. REAL TIME DATABASE SYSTEMS IN DISTRIBUTED ENVIRONMENT

Real Time systems (RTS) are those for which accuracy depends not only on the logical properties of the produced results, but also on the temporal properties of these results [30, 31,32]. Typically, real time systems are associated with critical applications, in which human lives or expensive machineries may be at stake [33]. Hence, in such systems, an action performed too late (or too early) or a computation which uses temporally invalid data may be useless and sometimes harmful even if such an action or computation is functionally correct. The RTS continue to evolve; their applications become more and more complex, and often require timely access and predictable processing of massive amounts of real time data [34].

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. The systems with deadlines are called as real time system (RTDBS). A transaction in a database system can have any real time constraints. A real time database system is a transaction processing system that is designed to handle workloads, where each transaction has completion deadline. The deadlines are fall into three categories: (I) Hard deadlines, (II) Firm deadlines and (III) Soft Deadlines.

(I) Hard deadlines: serious problem, this type of problem occurs when a task is not completed within the deadlines,

(II) Firm deadlines: the task is completed after the deadlines, and

(III) Soft Deadlines: the task diminishes its value if the task is completed after the deadlines.

NG considered on a commit protocol for check pointing transactions [35]. A distributed transaction is executed at various sites. The transaction may decide to commit at some sites whereas at one another sites it could decide to abort resulting in a violation of transaction atomicity in distributed environment [26].

Al-Houmaily et al. took atomicity with incompatible presumptions [37]. To overcome this problem distributed databases 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. Lee Inseon et al. worked on a causal commit protocol such as a new approach for distributed main memory database systems [38].

Commit protocol is required to ensure that either all the effects of the transaction persist or none of them persist in spite of the site or communication link failures and loss of messages.

Designing the real time system involved ensuring that there is enough processing power to meet the deadlines without the need of excessive hardware resources. The objective of system is to meet these deadlines, that is, to complete processing transaction before their deadlines expire. In RTDBS, the performance of the transaction commit is usually measured in terms of the numbers of transactions that complete before their deadlines.

There are collection of multiple, logically interrelated databases distributed over a computer network where transactions have explicit timing constraints, usually in the form of deadlines, it is called real time database system in distributed environment (DRTDBS).

The transaction that misses the completion of processing before its deadline is just considered as killed or aborted and discarded from the system without being executed to completion [5].

Database researchers have been working in this area for many early years and a variety of commit protocols have so far been proposed. These protocols include one phase protocols like Early Prepare (EP) [39,40], two phase protocols like the classical Two Phase Commit (2PC) [4], three phase protocol like Three Phase Commit (3PC) [11] and many of their optimizations.

A survey in RTDBS is in [14,15] and a detail of deadlines is discussed in [21,25,16, 36]. Early Prepare (EP) commit protocol & two-phase commit protocol in real-time designation has been investigated in [39,40, 16,22,3,18]. The works concentrated in management of deadline applied to a transaction and scheduling of arrival rates of workload with experimental performance of system under variety of workloads and different methods.

4. PROPOSED MODEL

The performance evaluation of the early prepare commit protocol (EP), we develop a detailed proposed model of a real time database system in distributed environment based on loose combination of the distributed database model presented in [5,12,23,24]. More details on the definitions and literature are available in [19,2,6,21,16,3,18,7,8,1, 20,36,39,41]. The proposed model consists of non-replicated

manner of database distributed to all available sites, say, for example, 8 sites in our case. According to our study, we modify the model of the Real Time Database System in Distributed Environment from the basic model presented in [5]. The model consists of six different components as shown in Fig.1.

Source: This component is mainly responsible for generating the transaction workload for a site. The workload model used by the source characterizes transactions in terms of the files that they access and the numbers of pages that they access and update in each file.

Transaction Manager: The transaction manager is responsible for accepting transactions from the source and modeling their execution. Each transaction in the workloads has a master process, numbers of cohorts and possibly a number of updaters. The master resides at the site, where the transaction is submitted. Each cohort makes a sequence of read and write requests to one or more files that are stored at its sites. A transaction has one cohort at each site, where it needs to access data. To select the execution sites for a transaction's cohorts, the decision rules is: If a files is present at the originating site, use the copy there; otherwise choose uniformly from the sites that have remote copies of the file.

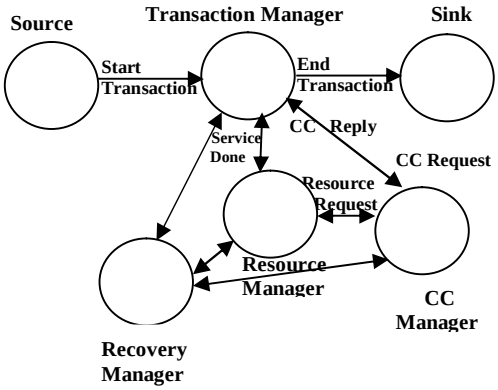

Fig. 1: The Framework of Real Time Database System in Distributed Environment Model

Recovery Manager: The recovery manager implements the details commit protocol.

Resource Manager: The resource manager manages the physical resources of sites like its CPU and its disk for reading and writing data or message from them. It also provides the CPU and I/O service to the transaction manager and concurrency control manager. This component is not fully implemented in our work.

Concurrency Control Manager: The concurrency control (CC) manager is responsible for handling concurrency control requests made by the transaction manager, including read and write access requests, requests to get permission to commit a transaction, and several types of master and cohort management requests to initialize and terminate master and cohort processes.

Sink: The sink deals with collection of statistics for the completed transactions.

Network Manager is a communication network interconnects the sites. All sites communication via messages exchange over the communication network. The network manager models the behavior of the communications network.

5. PROPOSED MODEL AND ITS PARAMETERS

Our works adopt the common model of transaction execution in distributed. There is one process called master which is executed at the site, where the transaction is submitted. There are many processes called cohorts, which are executed on behalf of the transactions at the different sites that are accessed in the transaction.

So, we called in other words, each transaction has a master process that runs at its originating site. The master process, in turns sites up a collection of cohort processes, which are involved in running the transaction. Cohorts are created by the master sending a STARTWORK message to local transaction manager at that site. After each cohort finishes executing its portions of a query, it sends a WORKDONE massage to the master and the master initiates the execution of a process after it receives such message from all its cohorts. When the transaction is initiated at the site of files and data items that it will access are chosen by the source, the master is, then loaded at its site of originating.

The Early Prepare Commit Protocol (EP) uses Presumed Commit (PC: an optimization over 2PC, where a cohort in prepared state goes ahead to commit in case of no reply from its master) to eliminate one round of messages for a distributed transaction that executes in the absence of failures. The communication is also reduced by EP, further by making each cohort enter the prepared state after it performs its work and before it replies to the master with the WORKDONE message.

A master using EP may have to force multiple MEMBERSHIP records, because the transaction membership may grow as transaction execution progresses. A master using the presumed commit protocol must record the identity of a cohort in its stable log before the cohort can enter its prepared state. Hence a Master using EP must record a Cohort's identity in its stable log before sending a work

request to that cohort. If the master knows the transaction membership before it begins executing the transaction, the master can force one membership record. On the other hand, if the master uses the results obtained from one cohort to determine the identity of a subsequent cohort and no two operations are done by the same cohort, the master must force a MEMBERSHIP record before sending each transaction operation to the appropriate cohort.

The steps of execution of EP are as: (i) Master forces one or more MEMBERSHIP log records and sends a STARTWORK request to each cohort. (ii) Each cohort executes its work request, forces a PREPARE log record and replies to master with a WORKDONE message. (iii) Master forces a COMMIT log record, sends a COMMIT message to each cohort, and forgets about the transaction. (iv) Each cohort appends to its log, but need not force, a COMMIT log record and then forgets about the transaction.

When EP is used and several work requests are sent to each cohort, each cohort forces several PREPARE log records. In contrast, when 2PC or PC is used, each cohort involved in a transaction forces only one PREPARE log record regardless of the number of work requests it received.

A cohort using EP synchronously forces each PREPARE record as early as possible, while a cohort using PC synchronously forces the PREPARE record as late as possible. A cohort using an intermediate approach could reply to the master after executing its work request, force the PREPARE record asynchronously, and notify the master once the record has been forced. This asynchronous approach reduces the number of synchronous log forces and may increase parallelism. It requires one more round of messages compared to EP, but one less round compared to PC.

We consider assuming that, without loss of generality that both the master and the cohorts are in an executing state. From this point, a cohort can move to either aborting or prepared state, depending upon whether it successfully finishes its work or not. If it finishes its job successfully, it forces a PREPARE log record and sends a WORKDONE message to its master. Once a cohort is in prepared state, it can no longer unilaterally decide to move to the aborting state; it has to wait for the master's decision of commit or abort, based on which it will move to aborting or committing state by forcewriting the appropriate log record.

The master makes the decision (commit or abort) about the fate of the transaction. It can directly move to the aborted state from the executing state sending ABORT messages to the cohorts (i.e., if the transaction is aborted by the user). Master can as well move to the aborting state by sending ABORT message to all of its successful cohorts. It moves to the committing state by sending the COMMIT messages to all the cohorts. Note that the master requires to force write the ABORT log record while moving to its aborting state. From the executing state, the master moves to the committing state (by writing a COMMIT log record) only if it gets all WORKDONE messages from all the cohorts and then it forgets about the transaction, because of the PC feature of EP. Even if one ABORT message is received from any cohort, the master moves to the aborting state (by forcewriting an ABORT log record). From the aborting states, the master moves to the aborted state, by writing an END log record. The END log record is written only after the master has received all ACKs from the cohorts that were sent the decision. Note that, the cohorts do not write any more log records (that is, no END record) after moving to committing or aborting states, therefore, for the cohorts, these states are equivalent to committed or aborted states, respectively.

Finally, the master, after receiving ACK from all the prepared cohorts, writes an end log record and then "forgets" the transaction and makes free. Then the statistics results are collected in the sink. In our simulation experiments, we consider the transactions that are executed in parallel. A single formula is used to assign deadlines to all transactions. Each transaction is assigned a deadline and its formula is given by the following equation:

$$DT=AT+SF*RT \qquad \ldots\ldots\ldots\ldots\ldots (1)$$

Here,
DT = Deadline of Transaction;
AT = Arrival Time of Transaction;
RT = Resource Time of Transaction (T) and
SF = Slack Factor.

Slack factor (SF) that provides control over the tightness and slackness of deadlines. The resource for its execution in other word is an execution time of a transaction.

There are two issues related problem to resource time such as (i) It is a function of the number of messages and force-writes, which differ from one commit protocol to another commit protocol, and (ii) The workload generated utilizes information about transaction resource requirements in assigning deadlines. The performance evolution of the commit protocols of proposed model of a real time database system in distributed environment has been described. The settings of workload and system parameters used in our model. Summary of the proposed model parameters are given in table 1.

TABLE 1. PROPOSED MODEL PARAMETERS

Parameters	Description
NumSites orSelectfile	Number of sites in the Database
Dbsize	Number of pages in the database
ArrivalRate	Transaction arrival rate/site
Slackfactor	Slack factor in Deadline formula
FileSelection Time	Degree of Freedom (DistDegree)
WriteProb	Page update probability
PageCPU	CPU page processing time
PageDisk	Disk page access time
TerminalThink	Time between completion of 1 transaction & submission of another
Numwrite	Number of Write Transactions
NumberReadT	Number of Read Transactions

6. PARAMETER SETTINGS

The values of the parameter set in the simulation are given in table 2. Based on the transaction type, the cohorts may execute either in parallel or sequential. Each cohort makes a series of read and update access. In our model, the transaction has a single master process and multiple cohorts. The number of sites at which the transaction executed is simplified by the DistDegree parameters. At each of sites, the number of pages access by the cohorts varies uniformly between 0.5 to 1.5 times of Cohort Size. These pages are chosen randomly form among the database pages located at the sites. A page is read by the WriteProbe parameter. If the transaction's action deadline expires either before the completion of its local processing or before the master has written the global decision log receive, the transaction is killed and discarded. The CPU time to process a page is 10 milliseconds and the disk access time is 20 milliseconds.

TABLE 2. ASSSUMED VALUES OF PROPOSED MODEL PARAMETERS

Parameters	Parameters
NumSites	8
Dbsize	vary(max.2400)
ArrivalRate	6 to 8 job/sec
Slackfactor	4
FileSelection Time	3
WriteProb	0.5
PageCPU	10ms
PageDisk	20ms
TerminalThink	0 to 0.5 sec

7. ANTICPATION OF RESULTS

We conducted an broad set of simulation experiments using the above mentioned parameters in Table 1 & 2 using simulation languages GPSS [13]. The simulation can use different simulation languages such as C++SIM, DeNet [10,9] etc. Commit percentage and Abort percentage were used as measures for the performance metrics in our simulation results. Commit percentage is the percentage of input transactions that the system is able to complete before their deadline and Abort percentage is the percentage of input transactions that the system is unable to complete before their deadline. We conducted simulation under normal and heavy loads with different settings of workload parameters like as Numsites, DBsize(File size), DistDegree (File selection time), and with other corresponding parameter values. We considered 8 distributed sites, with 8 files and other parameter values were kept constant. The one phase EP protocol significantly reduces the overhead of commit processing by eliminating an entire phase, making it especially attractive in the real time database system in distributed environment.

8. CONCLUSIONS

The performance of transactions was performed by comparison of transactions Commit and Abort percentages under two different workloads such as normal load and heavy load. Thus, for heavy load both Commit percentage and Abort percentage were high, it observed that could have been problem of improper distribution, like the work was assigned to a busy site. The increase in file selection time minimized the Abort percentage and gave improper performance. The performance improved in all sites under normal load. There is increase number of sites that the abort percentage was very low in the transaction.

9. REFERENCE

1. A.S. Tanenbaum, "Computer Networks", Third Editions, *Prentice Hall of India Pvt. Ltd*, 2001.
2. Bipin C. Desai, "An Introduction to Database System", *Galgotia Publication Pvt. Ltd., New Delhi.(India)*, 1st Edition pp.520, 1993.
3. H. Jayant, "Performance Analysis of Real Time Database System", *Technology Report* 92-96. University of Maryland, USA, 1991.
4. J. Gray, " Notes on Database Operating Systems", Operating System: An advanced course, *Lecture notes in Computer Science*, 60, 1978.
5. Jayant H, J.M.Carey and M. Livney, "Data Access Scheduling in Firm Real Time Database System", *Real Time System Journal*, 4(3), 1992.
6. Jayant H, Ramesh G, Kriti R, and S.Seshadri, "Commit Processing in Distributed Real Time Database System" *Pro of 17th IEEE Real Time System Symposium USA*, December, 1996.
7. Jayant, Krithi, Ramesh, "The PROMPT Real Time Commit Protocol", *IEEE transaction on Parallel anddistributed System*, vol 2, no 2, Feb, 2000.
8. Kenneth Reed, *Data Networks* (Handbooks), IE: 61-65, 1998.
9. Livny, M, DeNet User's Guide, Version 1.0, *Comp. Sc. Dept University of Wisconsim, Madison*, 1988.
10. M.C. Little and D.L. Mc Cue, *C++SIM User's Guide Public release 1.5*, *Dept. of Computing Sc. University of New Castte Upon Tyne, U.K.*, 1994.
11. M.Qszu, P.Valdureiz, "Principles of Distributed Database System", *Prentice Hall*, 1991.
12. Michale J.Carey and Miron Livny, "Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication" *Proc. Of 14th VLDB conference*, Los Angeles, California, August, 1988.
13. Minutesmansoftware,GPSS, world North Carolina ,USA,4E.[*GPSS Book*] (Student Version 4.3.5), 2001. http://www.minutmansoftware.com
14. O. Ulusoy, " Research Issues in Real Time Database system " , *Technology Report BUCEIS-94-32, Department of Computer Engineering and Information Science, Bilkent University, Turkey*, 1994.
15. O. Ulusoy, "Processing Real-time Transaction in a Replicated Database Systems", *Intel Journal of Distributed and Parallel Database* 2(4), 1994.
16. Robert Abbott and Hector Garcia-Molina, "Scheduling Real time transaction: a performance Evaluation, *Programme of 14th VLDB conference Los Angeles*, 1988.
17. S. Son, "Real Time Database System", *A new challenging Data Engineering*, 13 (4), December, 1990.
18. S.Davidson,I.Lee and V. Wolfe., "A protocol for Times Atomic Commitment", *Proc. of 9th International Conference on Distributed Computing System*", 1989.
19. Silberchatz, Korth, Surdarshan, "Database System Concept", *4th International Edition, McGraw Hills,. New Delhi.(India)*, pp 91, 1984.
20. Udai Shanker, Manoj Misra, Anil K. Sarje, "Distributed real time database systems: background and literature review", Distrib Parallel Databases, 127-149(23), 2008.
21. Y. Jayanta, S.C. Malhotra, "Management of Atomicity problem in worst possible environment", *Library progress (International)*, 22(1), 25, 2002.
22. Y. Yoon, "Transaction Scheduling and Commit Processing for Real time Distributed database System", *Ph. D thesis Korea Advance Institute of Sc. And Technology*, May, 1994.
23. Y.Jayanta & S.C. Mehrotra, "A new Commit processing under Distributed Database Real time Database System", *Skyline Business Journal, Sharjah, UAE*, 2005.
24. Y.Jayanta & S.C. Mehrotra, "Performance analysis of a Real Time Distributed Database System through simulation", *15th IASTED International Conference on APPLIED SIMULATION AND MODELLING(ASM), Rhodes, Greece*, June, 2006. http://www.actapress.com
25. Y.Jayanta & S.C. Mehrotra, "Simulation of commit processing under Distributed Real Time Database System", *IETE, National Conference, Aurangabad, India*, January, 2004.
26. Ramakrishnan Raghu and Gehrke, Johannes, "Database Management System," McGraw Hill Publication, New York, January 2003.
27. Ullman Jeffrey D., "Principle of Database Systems," Galgotia Publication Pvt. Ltd. 1992.
28. Gehani Narain, Ramamritham K., Shanmugasundaram, J. and Shmueli, O., "Accessing Extra Database Information: Concurrency Control and Correctness," Information Systems, Vol. 23, Issue 7, 439-462, Nov. 1996.
29. Garcia-Molina Hector and Lindsay Bruce, "Research Directions for Distributed Databases," ACM SIGMOD Record, Vol 19, Issue 4, December, 1990.
30. Aldarmi Saud A., "Real Time Database Systems: Concepts and Design," Department of Computer Science, University of York, April 1998.
31. Stankovic John A., "Misconception about Real-Time Computing," IEEE Computer, pp. 10-19, Oct. 1988.
32. Vrbsky Susan V. and Tomic Sasa, "Satisfying Timing Constraints of Real Time Databases," Journal of Systems and Software, Vol. 41, pp. 63-73, 1998. www.mpcs.org/MPCS98/Final_papers/Paper.48.pdf
33. Lam Kam -Yiu, Law Gary C.K. and Lee Victor C.S., "Priority and Deadline Assignment to Triggered Transactions in Distributed Real-Time Active Databases," Journal of Systems and Software, Vol. 51, No. 1, pp. 49-60, April, 2000.
34. Ulusoy Ozgur, "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.
35. NG Pui, "A Commit Protocol for Checkpointing Transactions," Proceedings of the 7th Symposium on Reliable Distributed Systems, Columbus, OH, USA, pp. 22–31, Oct. 10-12, 1998.
36. Shetan Ram Choudhary, C.K.Jha, "Performance Evaluation of Real Time Database Systems in Distributed Environment", Int. J. Computer Technology & Applications, Vol 4(5), 785-792, 2013.
37. Al-Houmaily Yousef J. and Chrysanthis P.K., "Atomicity with Incompatible Presumptions," Proceedings of the 18th ACM Symposium on Principles of Database Systems (PODS), Philadelphia, June 1999.
38. Lee Inseon, Heon Y. and Park Taesoon, "A new Approach for Distributed Main Memory Database Systems: A Causal Commit Protocol," IEICE Transactions on Information and System, Vol. E87-D, No.1, pp. 196-204, January 2004.
39. James W. Stamos and Flaviu Cristian, "A Low_Cost Atomic Commit Protocol, In Proceedings of the 9th IEEE Symposium on Reliable Distributed Systems, October, 1990.
40. James W. Stamos and Flaviu Cristian, "Coordinator Log Transaction Execution Protocol", International Journal on Distributed and Parallel Databases, 1(4), 1993