HYBRID TRANSACTION MANAGEMENT IN DISTRIBUTED REAL-TIME DATABASE SYSTEM

Gyanendra Kumar Gupta[1], A. K. Sharma[2] and Vishnu Swaroop[3]
[1]Department of Computer Science & Engineering, KIT, Kanpur, U.P., India.
[2&3]Department of Computer Science & Engg., MMM Engg. College, Gorakhpur, U.P., India.

ABSTRACT

Managing the transactions in real time distributed computing system is not easy, as it has heterogeneously networked computers to solve a single problem. If a transaction runs across some different sites, it may commit at some sites and may failure at another site, leading to an inconsistent transaction. The complexity is increase in real time applications by placing deadlines on the response time of the database system and transactions processing. Such a system needs to process transactions before these deadlines expired. A series of simulation study have been performed to analyze the performance under different transaction management under conditions such as different workloads, distribution methods, execution mode-distribution and parallel etc. The scheduling of data accesses are done in order to meet their deadlines and to minimize the number of transactions that missed deadlines. A new concept is introduced to manage the transactions in "hybrid transaction management" rather than static and dynamic ways setting computing parameters. This will keep the track of the status of mix transaction static as well as dynamic so that we can improve the performance of the system with the advantages of static as well as dynamic.

KEYWORDS: *Real time system, hybrid transaction management, missed deadlines, database size.*

I. INTRODUCTION

As the world become smarter and more informatics, demands on IT will grow. Many converging technologies are coming up like rising IT delivery model-cloud computing. Demands of the real time distributed database are also increasing. Many transaction complexities are there in handling concurrency control and database recovery in distributed database systems. Two-phase commit protocol is most widely used to solve these problems [1] and commit protocols are implemented in distributed system. A uniform commitment is guarantee by a commit protocol in such system to ensure that all the participating sites agree on a final outcome. Result may be either a commit or an abort condition.

Many real time database applications in areas of communication system and military systems are distributed in nature. In a real time database system the transaction processing system that is designed to handle workloads where transactions have deadlines. A series of simulation study have been performed to analyze the performance of the system under different transaction management condition such as different workloads, distribution methods, execution mode-Distribution and Parallel, impact of dynamic slack factors to throughput etc. The section 2 describes the concept of a real time database system. The section 3 describes the transaction details. In section 4, proposed model and their parameters are given. The detail of anticipation of result and analysis are given in section 5. The overall conclusions are discussed in section 6.

II. REVIEW OF LITERATURE

Many database researchers have proposed varieties of commit protocols like two phase commit and Nested two phase commit [2, 3], Presumed commit [4] and Presume abort [3], Broadcast two phase commit , Three phase commit [5,6] etc. These require exchanges of multiple messages, in multiple

phases, between the participating sites where the distributed transaction executed. Several log records are generated to make permanent changed to the data disk, demanding some more transaction execution time [4, 7, 8]. Proper scheduling of transactions and management of its execution time are important factors in designing such systems.
Transactions processing in any database systems can have real time constraints. The scheduling transactions with deadlines on a single processor memory resident database system have been developed and evaluated the scheduling through simulation [9]. A real time database system is a Transaction processing system that designed to handle workloads where transactions have complete deadlines. In case of faults, it is not possible to provide such guarantee. Real actions such as firing a weapon or dispensing cash may not be compensatable at all [10]. Proper scheduling of transactions and management of its execution time are the important factors in designing such systems. In such a database, the performance of the commit protocol is usually measured in terms of number of transactions that complete before their deadlines. The transaction that miss their deadlines before the completion of processing are just killed or aborted and discarded from the system without being executed to completion [11].

III. TRANSACTION DETAILS

This study is in continuation of [12, 13] work in the same domain [14, 15]. The study follows the real time processing model [16, 17, 18] and transaction processing addressing timeliness [19]. This model has six components: (i) a source (ii) a transaction manager (iii) a concurrency control manager (iv) a resource manager (v) a recovery manager (vi) a sink to collects statistics on the completed transactions. A network manager models the behaviour of the communications network. The definitions of the components of the model are given below.

3.1 The source:

This component is responsible for generating the workloads for a site. The workloads are characterized in terms of files that they access and number of pages that they access and also update of a file.

3.2 The transaction manager:

The transaction manager is responsible for accepting transaction from the source and modelling their execution. This deals with the execution behaviour of the transaction. Each transaction in the workload has a general structure consist of a master process and a number of cohorts. The master resides at the sites where the transaction was submitted. Each cohort makes a sequence of read and writes requests to files that are stored at its sites. A transaction has one cohort at each site where it needs to access data. To choose the execution sites for a transaction's cohorts, the decision rule is: if a file is present at the originating site, use the copy there; otherwise, choose uniformly from among the sites that have remote copies of the files. The transaction manager also models the details of the commit and abort protocols.

3.3 The concurrency control manager:

It deals with the implementation of the concurrency control algorithms. In this study, this module is not fully implemented. The effect of this is dependent on algorithm that chooses during designing the system.

3.4 The resource manager:

The resource manager models the physical resources like CPU, Disk, and files etc for writing to or accessing data or messages from them.

3.5 The sink:

The sink deals for collection of statistics on the completed transactions.

3.6 The Network Manager:

The network manager encapsulates the model of the communications network. It is assuming a local area network system, where the actual time on the wire for messages is negligible.

IV. TRANSACTION MODEL AND THEIR PARAMETER

The proposed model is discussed below. A common model of a distributed transaction is that there is one process, called as Master, which is executed at the site where the transaction is submitted, and a set of processes, called Cohorts, which executes on behalf of the transaction at these various sites that are accessed by the transaction. In other words, each transaction has a master process that runs at its site of origination. The master process in turn sets up a collection of cohort's processes to perform the actual processing involved in running the transaction. When cohort finishes executing its portion of a query, it sends an execution complete message to the master. When the master received such a message from each cohort, it starts its execution process.

When a transaction is initiated, the set of files and data items that, it will access are chosen by the source. The master is then loaded at its originating site and initiates the first phase of the protocol by sending PREPARE (to commit) messages in parallel to all the cohorts. Each cohort that is ready to commit, first force-writes a prepared log record to its local stable storage and then sends a YES vote to the master. At this stage, the cohort has entered a prepared state wherein it cannot unilaterally commit or abort the transaction but has to wait for final decision from the master. On other hand, each cohort that decides to abort force-writes an abort log record and sends a NO vote to the master. Since a NO vote acts like a veto, cohort is permitted unilaterally abort the transaction without waiting for a response from the master.

After the master receives the votes from all the cohorts, it initiates the second phase of the protocol. If all the votes are YES, it moves to a committing state by force-writing a commit log record and sending COMMIT messages to all the cohorts. Each cohort after receiving a COMMIT message moves to the committing state, force-writes a commit log record, and sends an acknowledgement (ACK) message to the master. If the master receives even one NO vote, it moves to the aborting state by force writing an abort log record and sends ABORT messages to those cohorts that are in the prepared state. These cohorts, after receiving the ABORT message, move to aborting state, force-write an abort log record and send an ACK message to the master. Finally, the master, after receiving acknowledgement from all the prepared cohorts, writes an end log record and then forgets and made free the transaction. The statistics are collected in the Sink [11, 16, 17, 26]. The database is modeled as a collection of DBsize pages that are uniformly distributed across all the NumSites sites. At each site, transactions arrive under Poisson stream with rate Arrival Rate and each transaction has an associated firm deadline. The deadline is assigned using the formula

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

Here DT, AT, SF and RT are the deadline, arrival rate, Slack factor and resource time respectively, of transaction T. The Resource time is the total service time at the resources that the transaction requires for its execution. The Slack factor is a constant that provides control over the tightness or slackness of the transaction deadlines.

In this model, each of the transaction in the supplied workload has the structure of the single master and multiple cohorts. The number of sites at which each transaction executes is specifying by the File selection time (DistDegree) parameter. At each of the execution sites, the number of pages accessed by the transaction's cohort varies uniformly between 0.5 and 1.5 times Cohort Size. These pages are chosen randomly from among the database pages located at that site. A page that is read is updated with probability of WriteProb. Summary of the simulation parameter is given in table I.

Parameter Settings

The values of the parameter set in the simulation are given in table II. The CPU time to process a page is 10 milliseconds while disk access times are 20 milliseconds.

Table I. Proposed model parameters

Parameters	Description
NumSites or Selectfile	Number of sites in the Database
Dbsize_generating_site	Number of pages in the database at same location.
Dbsize_remote_site	Number of pages in the database at remote location.
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 transaction & submission of another
Numwrite	Number of Write Transactions
NumberReadT	Number of Read Transactions

Table II. Assumed values of proposed model parameters

Parameters	Set Values	Parameters	Set Values
NumSites	8	FileSelection Time	3
Dbsizevary	Max. 200 for generating site and 2200 for remote site	PageCPU	10ms
ArrivalRate	6 to 8 job/sec	PageDisk	20ms
Slackfactor	4	TerminalThink	0 to 0.5 sec
WriteProb	0.5	Numwrite/Number Read T	vary

V. ANTICIPATION OF RESULTS

The experiment has to be perform using different simulation language like C++Sim, DeNet etc. For this study, GPSS World can be use as a simulator [20]. Literatures are also collected from several recent studies [21, 22, 23, 24, 25, and 26]. The study for performance evaluation starts by first developing a base model. Further experiments were constructed around the base model experiments by varying a few parameters and process of execution at a time.
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. A study can be analyzing the performance of the system under different workload with varying the arrival rate of the transaction, dynamic slack factors, execution mode etc. A study can be analyzed the performance using this new concept of transaction to manage the transactions in “hybrid transaction management” rather than static and dynamic ways setting computing parameters technique along with varying database size for generating site and remote site technique. The anticipated experimental results are discussed below.

5.1. Comparison of Centralized and Distributed systems

This anticipated experiment compares the performance of the system under centralized and distributed [13]. The distributed systems have higher percentage of miss Transactions than centralized system. This higher miss percentage is due to distance between cohorts. This leads to design of a new perfect distributed commit processing protocol to have a real-time committing performance.

5.2. Impact of distribution methods

This anticipated experiment is to be conducted to know the impact of difference between distribution methods to the performance of the system [13]. As an example, we take Exponential distribution and Poisson distribution. The assignment and committing of transactions to cohorts are passed under scheduler using Exponential distribution and Poisson distribution and the statistics of the simulation outputs are to be noted. The Exponential might give more uniform assignment and committing of transactions than Poisson. Poisson might throws higher numbers of transactions giving more collisions of transactions and large number miss percentage of transactions than Exponential. So on many experiments of such similar types might be conducted by using more different distribution rules.

5.3. Impact execution mode: Distribution and Parallel

This anticipated experiment compares the output of the system putting the cohorts in parallel with that of distribution execution [13]. From this we might conclude following points. Parallel execution of the cohorts might reduce the transaction response time. The time might require for the commit processing is partially reduced. This is because the queuing time is shorted in parallel and so there are much fewer chances of a cohort aborting during waiting phase.

5.4. Impact of slack to Throughput

In this set of experiments, the impact of slack factor to observed on the throughput of the system [13]. The throughput initially might decreases with increase in slack factor due to constraint of distributed real time database. Still there are lots more to study required about other parameters to improve the throughput of the overall system.

5.5. Transaction Management

The transactions can be managed in many different ways. In most of the earlier work done simply static or dynamic ways with only database size computing [13,26]. A new concept is introduced to manage the hybrid transactions management with database size for originating site and remote site rather than database size computing parameters, where the values of the parameters are changes or adjust automatically depending on the requirements during the execution the experiment.

VI. CONCLUSIONS

A series of simulation study have been performed to analyze the performance under different transaction management situation such as different workloads, distribution methods, execution mode-Distribution and Parallel, impact of dynamic slack factors to throughput. The scheduling of data accesses are done in order to meet their deadlines and to minimize the number of transactions that missed deadlines.
Parallel execution of the cohorts reduces the transaction response time than that of serial or distributed execution. The time required for the commit processing is partially reduced, because the queuing time is shorted in parallel and so there are much fewer chances of a cohort aborting during waiting phase. The throughput initially increases with increase in slack factor. But it drops rapidly at very high work loads. The slack factors can be providing by static or dynamics ways.
A new concept is introduced to manage the hybrid transactions in database size for originating site and remote site rather than database size computing parameters. With this approach, the system gives a significant improvement in performance. This approach will keep tracks of timing of the transactions to help them from aborts. This approach will give advance information about the remaining execution time of the transactions. This will help the system to inject extra time to such transactions with the merit of static as well as dynamic ways with the track and does recording of the status of the status of failing transaction so that we can provide an extra slack time to improve the performance of the system. In all the conditions the arrival rate of transaction plays a major role in reducing number of miss percentage and improved performance.

REFERENCES

[1] Silberschatz, Korth, Sudarshan, 2002, Database system concept,4th (I.E), McGraow-Hill Pub. 698-709,903
[2] Gray. J, 1978,“Notes on Database Operating Systems”, Operating Systems:An Advanced Course, Lecture notes in Computer Science
[3] Mohan, C, Lindsay B and Obermark 1986, Transaction Management in the R* Distributed Database Management Systems, ACM TODS, 11(4).
[4] Lampson B and Lomet D, 1993, “A new Presumes Commit Optimization for Two phase Commit”, Pro.of 19th VLDB Conference.
[5] Oszu M, Valduriez P, 1991, Principles of Distributed Database Systems, Prentice-Hall.
[6] Kohler W, 1981, A survey of Techniques for Synchronization and Recovery in Decentralized Computer System, ACM Computing Surveys, 13(2)

[7] Nystrom D, Nolin M, 2006, Pessimistic Concurrency Control and Versioning to Support Database Pointers in Real-Time Databases, Proc. 16th Euromicro Conf. on Real-Time Systems
[8] Ramamritham,Son S. H, and DiPippo L,2004, Real-Time Databases and Data Services, Real-Time Systems J., vol. 28, 179-216.
[9] Robert A and Garcia-Molina H, 1992, Scheduling Real-Time Transactions, ACM Trans. on Database Systems, 17(3).
[10] Levy E., Korth H and Silberschatz,1991,An optimistic commit protocol for distributed transaction management, Pro.of ACM SIGMOD Conf.
[11] Jayant. H, Carey M, Livney,1992, “Data Access Scheduling in Firm Real time Database Systems”, Real Time systems Journal, 4(3)
[12] Jayanta Singh and S.C Mehrotra et all, 2010,“Management of missed transaction in a distributed system through simulation”, Proc. Of IEEE.
[13] Udai Shanker, “Some Performance Issues In Distributed Real Time Database System”, PhD Thesis, Computer Science and Engineering Department M.M.M.E.C, Gorakhpur, December, 2005.
[14] Jayanta Singh and S.C Mehrotra, 2006, “Performance analysis of a Real Time Distributed Database System through simulation” 15th IASTED International Conf. on APPLIED SIMULATION & MODELLING, Greece
[15] Jayanta Singh and S.C Mehrotra, 2009 "A study on transaction scheduling in a real-time distributed system”,EUROSIS’s Annual Industrial Simulation Conference, UK.
[16] Jayant H. 1991, “Transaction Scheduling in Firm Real-Time Database Systems”, Ph.D. Thesis, Computer Science Dept. Univ. of Wisconsin, Madison.
[17] Jayant H. Carey M and Livney M, 1990 “Dynamic Real-Time Optimistic Concurrency Control”, Proc. of 11th IEEE Real-Time Systems Symp.
[18] Jayant H., Ramesh G. Kriti.R, S. Seshadri, ”Commit processing in Distributed Real-Time Database Systems”, Tech. Report-TR-96-01, Pro. Of 17th IEEE Real-Time Systems Symposium, USA,1996
[19] Han Q, 2003, Addressing timeliness /accuracy/ cost tradeoffs in information collection for dynamic environments, IEEE Real-Time System Symposium,Cancun, Mexico
[20] Minutesmansoftware, GPSS world, North Carolina, U. S. A. 2010.
[21] Xiong M. and Ramamritham K., 2004, Deriving Deadlines and Periods for Real-Time Update Transactions, IEEE Trans. on Computers, vol. 53,(5).
[22] Gustavsson S and Andler S 2005, Decentralized and continuous consistency management in distributed real-time databases with multiple writers of replicated data, Workshop on parallel and distributed real-time systems, Denver, CO
[23] Xiong M, Han S., and Lam K, 2005, A Deferrable Scheduling for Real-Time Transactions Maintaining Data Freshness, IEEE Real-Time Systems Symposium.
[24] Jan Lindstrom, 2006 "Relaxed Correctness for Firm Real-Time Databases," rtcsa, pp.82-86, 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06).
[25] Idoudi, N. Duvallet, C. Sadeg, B. Bouaziz, R. Gargouri, F,2008, Structural Model of Real-Time Databases: An Illustration, 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008).
[26] Jayanta Singh and S.C Mehrotra et al, 2010, “Dynamic Management of transactions in a distributed real-time processing system”, International Journal of Database Management Systems, Vol.2, No.2, May 2010.

Authors Profile

Gyanendra Kumar Gupta received his Master degree in Computer Application in year 2001 and M.Tech in Information Technology in year 2004. He has worked as Faculty in different reputed organizations. Presently he is working as Asst. Prof. in Computer Science and Engineering Deptt. , KIT, Kanpur. He has more than 10 years teaching experience and 3 years industry experience. His area of interest includes DBMS, Networks and Graph Theory. His research papers related to Real Time Distributed Database and Computer Network are published in several National & International Conferences. He is pursuing his PhD in Computer Science.

A.K. Sharma received his Master degree in Computer Science in year 1991 and PhD degree from IIT, Kharagpur in year 2005. Presently he is working as Associate Professor in Computer Science and Engineering Department, Madan Mohan Malaviya Engineering College, Gorakhpur. He has more than 23 years teaching experience. His areas of interest include Database Systems, Computer Graphics, and Object Oriented Systems. He has published several papers in National & International conferences & journals.

Vishnu Swaroop received his Master degree in Computer Application in year 2002 presently he is working as Computer Programmer in Computer Science and Engineering Department, Madan Mohan Malaviya Engineering College, Gorakhpur. He has more than 20 years teaching and professional experience. His area of interest includes DBMS, & Networks. His research papers related to Mobile Real Time Distributed Database and Computer Network are published in several National & International conferences. He is pursuing his PhD in Computer Science.