International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:8, 2009

An Analysis of Real-Time Distributed System under Different Priority Policies

Y. Jayanta Singh, and Suresh C. Mehrotra

***Abstract*—**A real time distributed computing has heterogeneously networked computers to solve a single problem. So coordination of activities among computers is a complex task and deadlines make more complex. The performances depend on many factors such as traffic workloads, database system architecture, underlying processors, disks speeds, etc. Simulation study have been performed to analyze the performance under different transaction scheduling: different workloads, arrival rate, priority policies, altering slack factors and Preemptive Policy. The performance metric of the experiments is missed percent that is the percentage of transaction that the system is unable to complete. The throughput of the system is depends on the arrival rate of transaction. The performance can be enhanced with altering the slack factor value. Working on slack value for the transaction can helps to avoid some of transactions from killing or aborts. Under the Preemptive Policy, many extra executions of new transactions can be carried out.

***Keywords*—**Real distributed systems, slack factors, transaction scheduling, priority policies.

I. Introduction

A distributed system is one where data are located on several computers that are linked together by a heterogeneous network. The advantages of such system are increased availability of resources, increased reliability, and increased execution speed in less time. The coordination of activities among computers is a complex task. If a transaction runs across two sites, it may commit at one site and may failure at another site, leading to an inconsistent transaction. Two-phase commit protocol is most widely used to solve these problems [1]. To ensure transaction atomicity, commit protocols are implemented in distributed database system. A uniform commitment is guarantee by a commit protocol in a distributed transaction execution 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, stock trading, threat analysis, process control 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 complete deadlines. To ensure transaction atomicity, commit protocol are implemented in distributed database system. For such a system a high coordination are required between distributed processing, other database issues, and real-time processing. Performance measures of the real time system concentrated towards overall optimization of transactions completed prior to the deadlines.

This paper shows a series of simulation study have been performed to analyze the performance under different transaction scheduling condition such as different workloads, arrival rate, CPU priority policies, altering slack factors and Preemptive Policy. The performance metric of the experiments is MissPercent that is the percentage of input transaction that the system is unable to complete before their deadline. The section II describes the concept of a real time database system. In section III, detail simulation model and simulation parameters are given. The detail experiment results and analysis are given in section IV. The overall conclusions are discussed in section V.

Y. J. Singh is with Faculty of Information Technology, 7th October University. Misurata, Libya (e-mail: y_jayanta@yahoo.com).

S. C. Mehrotra is with Dept of Information Technology and Computer Sc. Dr. B. A. Marathwada University, India (e-mail: mehrotrasc@rediffmail.com)

II. Real Time Database Concept

Database researchers have proposed varieties of commit protocols like Two phase commit, Nested two phase commit [2,3], Presumed commit and Presume abort [4], 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 [4,6,7]. 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 [8]. A real time database system is a Transaction processing system that designed to handle workloads where transactions have complete deadlines. A centralized timed Two-phase Commit protocol has been designed where the fate of a transaction is guaranteed to be known to all the participants of the transaction by a deadline [9]. 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. Simulation Details

A. Simulation Model

Many researchers have evaluated performance of database system. Literatures have been collected from the study of the real time processing model [11] and transaction processing addressing timeliness [12]. Such model consists of a database that is distributed in a non-replicated manner, over all the available sites (say 8 sites in this study) connected by a network [13,14,15]. This system will have six components: (i) a source: generate transactions, (ii) a transaction manager: models the execution behavior of the transaction, (iii) a concurrency control manager: implements the concurrency control algorithm, (iv) a resource manager: models the physical resources, (v) a recovery manager: implements the details commit protocol and (vi) a sink: collects statistics on the completed transactions. A network manager models the behavior of the communications network.

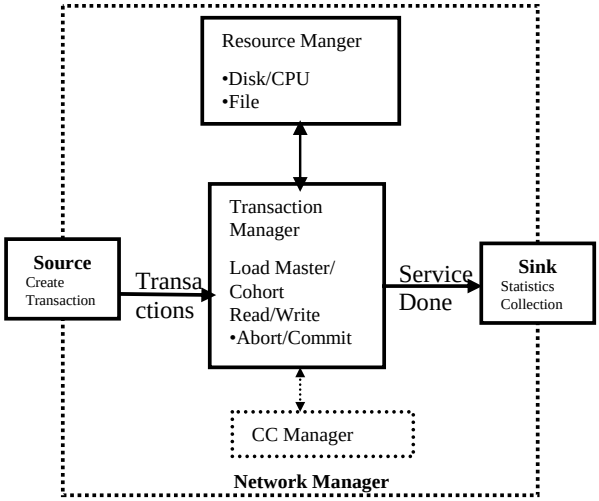

Fig. 1 Real time distributed database model

A typical real time database model is shown in Fig. 1. The definition of the components of the model is given below. The study is concentrated in managing the transaction scheduling under different environments. The study has concentrated to minimized numbers of the percentage of miss transactions under different conditions in order to optimize performance of the system.

The transaction manager: The transaction manager is responsible for accepting transaction from the source and modeling their execution. This deals with the execution behavior 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 trans-action manager also models the details of the commit and abort protocols.

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

B. Execution: Simulation Parameters and its Set Values

In a common model of a distributed transaction, 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 data items that 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. Then the statistics are collected in the Sink [11,13,14,15]

The database is modeled as a collection of Database size (Bsize) pages that are uniformly distributed across all the number of sites (NumSites). At each site, transactions arrive under Poisson stream with rate ArrivalRate, and each transaction has an associated firm deadline. The deadline is assigned using the formula. 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.

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

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 a parameter called the File selection time (DistDegree). 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 of 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 value. The CPU time to process a page is 10 milliseconds while disk access times are 20 milliseconds. Summary of the simulation parameters and its set values are given in Table I.

TABLE I
SIMULATION PARAMETERS AND ITS SET VALUES

Parameters	Description	Set values
NumSites or Selectfile	Number of sites in the Database	8
Dbsize	Number of pages in the database.	Vary (max.2400)
ArrivalRate	Transaction arrival rate/site	6 to 8 job/sec
Slackfactor	Slack factor in Deadline formula	4
FileSelection Time	Degree of Freedom (DistDegree)	3
WriteProb	Page update probability	0.5
PageCPU	CPU page processing time	10ms
PageDisk	Disk page access time	20ms
Terminal Think	Time between completion of one transaction and submission of another	0 to 0.5sec
Numwrite	Number of Write Transactions	-
Number ReadT	Number of Read Transactions	-

IV. EXPERIMENTS AND RESULTS

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 at a time. The experiment has been performed using different simulation language such as, in study [13] using C++Sim, and in study [11] using DeNet. Literatures are also collected from several recent studies [16,17,18,19,20,21].

For this study, GPSS World [22] is used as a simulator. 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 MissPercent that is the percentage of input transaction that the system is unable to complete before their deadline. If the transaction's action deadline expires either before completion of its local processing, or before the master has written the global decision log record, the transaction are killed and discarded. The MissPercent values in range of 0% to 20% are taken to represent system performance under "Normal" loads, while ranges of 21% to 100% represent system performance under "heavy" loads.

A. Traffic Analysis under Different Environments

This section discusses the statistical results of this simulation under different environments-Centralized and Distributed systems. As cohorts are kept at different locations, the distributed systems have higher percentage of miss transactions than that of centralized system. The higher miss percentage of transaction creates problem in managing atomicity of the transaction in such a system. This leads to design of a new distributed commit-processing protocol to have a real-time committing performance. The comparison of Centralize and distributed performances is shown in Figs. 2. Rest of the study will report on how the system will bring to optimized condition

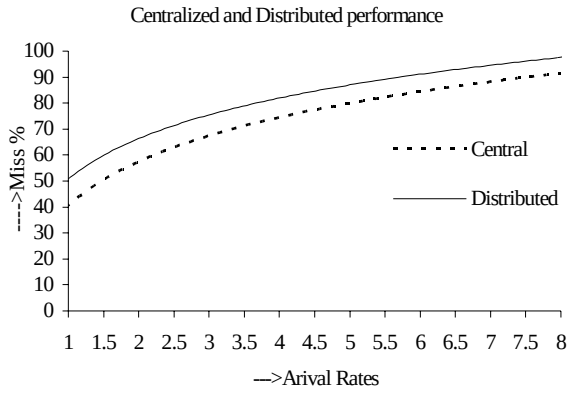

Fig. 2 Comparison of Centralize and distributed performances

B. *Impact of Arrival Rates to Individual Sites*

In this set of experiments, the impact of increasing the arrival rates was observed on the performances of the each of sites under normal load and heavy load. Fig. 3 presents the results obtained. Here s1, s2 etc are representing the 8 sites. Under both conditions, the miss percentage is reduced at the lower values of arrival rate of the transactions for each of the sites. In both the normal and heavy load the arrival rate play an important role to give a minimized miss percentage. The success ratios of the transaction are also increase by lowering the arrival rate.

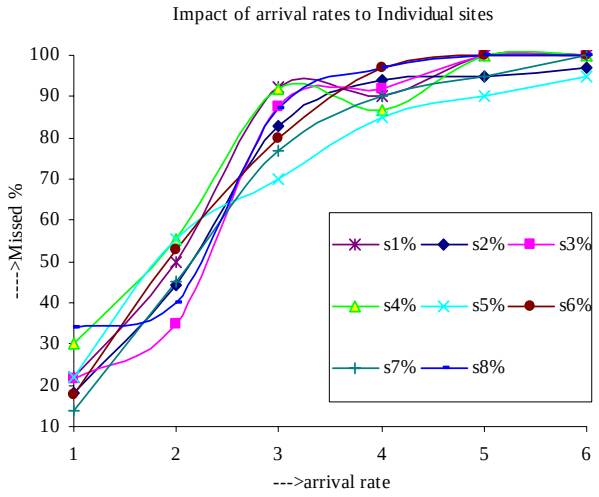

Fig. 3 Impact of arrival rates to Individual sites

C. *Impact of Arrival Rates to Throughput*

In this set of experiments, the impact of the arrival rates was observed on the throughput of the system. The throughput is the number of transactions completed prior to the deadline divided by the simulation clock elapsed. Fig. 3 presents the results obtained. The study represents the data for all the participating 8 different sites. The throughput initially increases with increase in arrival rate under normal workload. But it drops rapidly at very high loads. So the lowering arrival rates of transactions are recommended to have a less number of missed transactions.

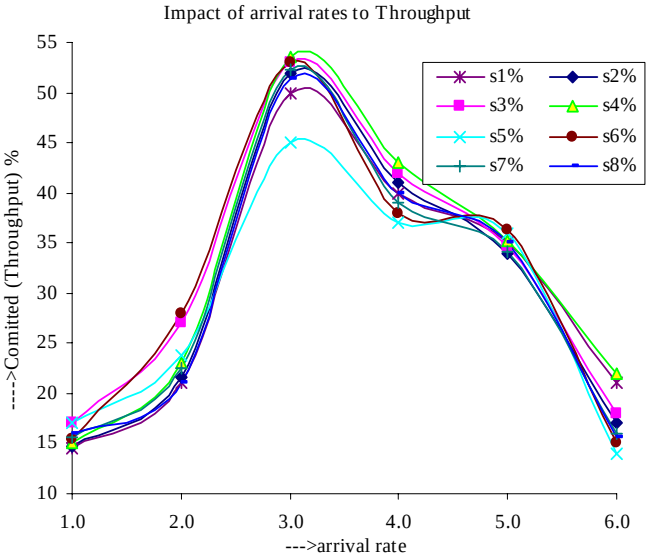

Fig. 4 Impact of arrival rates to Throughput

D. *Impact of CPU Priority Policies*

In this set of experiments, system behavior was studied under impact of CPU priority assignment policies. The transactions in the CPU queue were prioritized depending on a FIFO basis. A very similar type of Figure 4 is obtained as output. This study shows that there is no effect on the performance of the system by altering CPU priority policies.

E. *Impact of Slack Factor*

The Slack factor is a constant that provides control over the tightness or slackness of the transaction deadlines. Calculate the estimated slack value for the transaction that is the difference between the deadline and the estimated processing time. Normally the system kills or aborts all transactions which are unlikely to be completed before their deadlines. If the slack factor value is negative, it aborts the transaction and removed it from the queues.

This set of experiment is conducted to safe some of the transactions before killing or aborting them. After computing the slack value of all transactions, the system will know possible total number of transactions which have +ve and -ve slack values. If there is large number of Transaction with +ve slack value means that the system will have some relaxed time or bonus time. If the system is not having firm slack condition, alter the slack value to next higher level to safe some of the dying transactions. During alteration of slack value, it should initiate to execute same number of –ve slack value truncation to that of +ve slack value. In other words, it is going to comprised some –ve slack transactions if there remains some extra time for transactions with +ve slack value. The Fig. 4 shows the outcomes of the study. The constant slack value gives normal performance. By increasing the slack value, it gives low values of missed percentage of transaction. Lowering the slack value gives large value of missed percentage. However the system can safe some losing

transaction depending on number transaction which have +ve slack value.

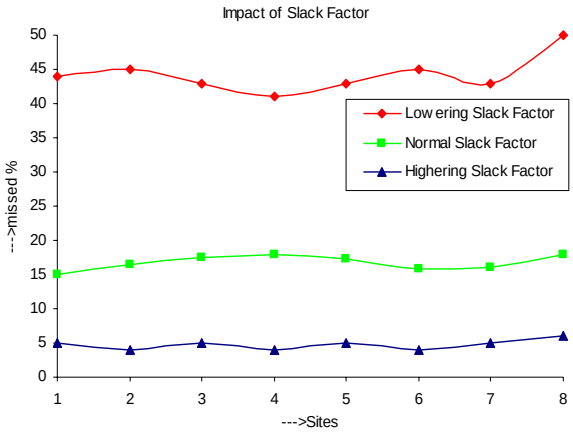

Fig. 5 Impact of Slack factor to Individual sites

F. Impact of Preemptive Policy

In most of previous study, the slack value is given as hard value. Other options are not provided to alter this value. So even the system has a large number of transactions with +ve slack value, system does not do anything extra, simply waits to ends its allotted time. This set of experiment is setup to carry out extra execution of new transaction with compromising with that numbers of transactions which have +ve slack value. If there is considerable large number of transactions with +ve slack factor value, then same number of new transaction can be scheduled. The study can observe some of the transactions with +ve slack value and it could help in generating some new transactions. In this way, extra execution of transactions can be carried out with the same amount of system time.

V. Conclusion

The real time distributed processing system has been simulating under different environments and conditions. The arrival rate of transaction plays a major role in reducing number of miss percentage and improved performance. The throughput of the system initially increases with increase in arrival rate. But it drops rapidly at very high work loads. So the more studies are required to observe the correlation between missed % and the throughput to have an optimized performance.

There is no effect on the performance of the system by altering CPU priority policies. Calculating the estimated slack value for the transaction can helps to avoid some of transactions from killing or aborts. If the system is not having firm slack condition, altering the slack value to next higher level can safe some of the dying transactions. If a system has considerable number of transaction with +ve slack factor, many extra executions of transactions can be carried out. Execution of new transaction can be done with compromising with that numbers of transactions which have +ve slack value and –ve slack value.

Acknowledgment

It is gratefully acknowledge Minuteman Software, P.O. Box 131, Holly Springs, North Carolina, 27540-0131, USA for providing free Study materials in their site. www.minutemansoftware.com.

References

[1] Silberschatz, Korth, Sudarshan-2002, *Database system concept,*4th (I.E), McGraow-Hill Pub. 698-709,903
[2] Gray. J,978 “Notes on Database Operating Systems”, Operating Systems: An Advanced Course, *Lecture notes in Computer Science, 60*
[3] Mohan, C, Lindsay B and Obermark R, 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 Conf*
[5] Oszu M, Valduriez P, 1991, *Principles of Distributed Database Systems,* Prentice-Hall,1991
[6] Kohler W, 1981,survey of Techniques for Synchronization and Recovery in Decentralized Computer System, *ACM Computing Surveys,* 13(2)
[7] Ramamritham,Son S. H, and DiPippo L, 2004, Real-Time Databases and Data Services, *Real-Time Systems J*., vol.28,179-216
[8] Robert A and Garcia-Molina H, 1992, Scheduling Real-Time Transactions, *ACM Trans. on Database Systems,* 17(3)
[9] Davidson S., Lee I and Wolfe V.,1989, A protocol for Times Atomic Commitment, Proc. of 9th Intl. Conf. On Distributed Computing System
[10] Levy E., Korth H and Silberschatz A.1991, An optimistic commit protocol for distributed transaction management, *Pro.of ACM SIGMOD Conf.*
[11] Haritsa J., Carey M, Livney M,’92, Data Access Scheduling in Firm Real time Database Systems, *Real Time systems J,* 4(3)
[12] Han Q, 2003, Addressing timeliness /accuracy/ cost tradeoffs in information collection for dynamic environments, *IEEE Real-Time System Symposium,Cancun, Mexico*
[13] Haritsa J., Ramesh G. Kriti.R, S. Seshadri, 1996, ”Commit processing in Distributed Real-Time Database Systems”, Tech. Report-TR-96-01, Pro. Pro. Of 17th *IEEE Real-Time Systems Symposium, USA*
[14] Haritsa J., Carey M and Livney M, 1990,“Dynamic Real-Time Optimistic Concurrency Control”, *Proc. of 11th IEEE Real-Time Systems Symp.*
[15] Haritsa J. 1991, “Transaction Scheduling in Firm Real-Time Database Systems”, *Ph.D. Thesis, Computer Science Dept. Univ. of Wisconsin, Madison*
[16] Xiong M. and Ramamritham K., 2004, Deriving Deadlines and Periods for Real-Time Update Transactions, *IEEE Trans. on Computers*, vol. 53,(5)
[17] Kang W, Son, S., Stankovic J, and Amirijoo M, 2007, I/O Aware Deadline Miss Ratio Management in Real-Time Embedded Databases, *IEEE RTSS*
[18] 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*
[19] Xiong M, Han S., and Lam K, 2005, A Deferrable Scheduling for Real-Time Transactions Maintaining Data Freshness, *IEEE Real-Time Systems Symposium, FL*
[20] 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)*
[21] 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)*
[22] Minutesmansoftware, *GPSS world*, North Carolina, U. S. A. 2001(4E). [GPSS-Book]

Y. Jayanta Singh is working as a Lecturer in Faculty of Information Technology, 7th October University, Misurata, (Libya). He obtained his M.Sc, Ph.D in Computer Science from Dr. B.A.Marathwada University, (India) in 2000 and 2004 respectively. He had worked with Keane (Canada&India), Skyline University College (UAE), TechMahindra (India). His research papers are published in International and National Journals and in conference proceedings. His areas of interest are Simulation and modeling, Real time Database, parallel and high speed computing and Software Engineering etc.

Suresh. C. Mehrotra, F.N.A,Sc., FIETE is working as a professor in Dr. Babasaheb Ambedkar Marathwada University, Aurangabad (India). He received his master degree in Physics from Allahabad University (India) in 1970 and Ph.D. in Physics from Austin (USA) in 1975. He is recipient of Welch Foundation Fellowship (1975), Alexander Von Humboldt Foundation Fellowship (1983-85), FOM (Netherland). He has published more than 150 papers in areas of Time Domain Reflectometery, Speech Processing, and Network Simulation.