Performance Analysis of Locking Algorithms for Distributed Database Systems(DDBS)

A

Dissertation submitted to
JAWAHARLAL NEHRU UNIVERSITY, New Delhi
in partial fulfillment of the requirements
for the award of the degree of

Master of Technology
in
Computer Science & Technology

By
MR. ANAND KUMAR

Under the Guidance of
PROF. P. C. SAXENA
&
PROF. C. P. KATTI

SCHOOL OF COMPUTER & SYSTEM SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
JANUARY, 2003

SCHOOL OF COMPUTER & SYSTEM SCIENCES

जवाहरलाल नेहरू विश्वविद्यालय

JAWAHARLALNEHRU UNIVERSITY

NEW DELHI – 110067 (INDIA)

CERTIFICATE

This is to certify that the dissertation entitled *"Performance Analysis of Locking Algorithms for Distributed Database Systems(DDBS)"* which is being submitted by **Mr. ANAND KUMAR** to the School of Computer & System Science, JAWAHARLAL NEHRU UNIVERSITY, NEW DELHI for the award of Master of Technology is a bonafide work carried out by him under our supervision.

This is original and has not been submitted in part or full to any university or institution for any Degree.

[illegible] 3-1-3

Prof. K. K. Bharadwaj
Dean
SC&SS, JNU

Professor-K. K. Bharadwaj
Dean
School of Computer & Systems Sciences,
Jawaharlal Nehru University
New Delhi - 110067

[illegible]

Prof. P. C. Saxena
Supervisor
SC&SS, JNU

[illegible]

Prof. C. P. Katti
Supervisor
SC&SS, JNU

PHONE : +91 -11 -26172426 FAX : +91 -11-26185886, 26198234 TELEX : (031) 73167

ACKNOWLEDGEMENT

I wish to convey my heartful gratitude and sincere acknowledgements to my Supervisors Prof. P. C. Saxena and Prof. C. P. Katti for their constant encouragement, guidance and affection throughout my work.

I am grateful to them for providing me enough infrastructures through Data Communication and Distributed Computing Group (DCDCG) laboratory to carry out my work. It is great honour to me for being a member of DCDC Group and I am thankful to all the members for their cooperation and understanding.

I would like to thank all faculty members for their help and useful suggestions during my work. I am also thankful to all of my classmates for their constructive criticisms and useful suggestions.

Anand Kumar

ANAND KUMAR

ABSTRACT

In the modern day information system world the popularity of distributed database systems (DDBS) is on rise. In these systems, many users will be accessing databases to retrieve and update data. Concurrent access to the databases must be synchronized for proper execution of the user transactions and to maintain consistency of the database. There are number of concurrency control algorithms proposed in the literature for synchronizing the transactions, among which locking algorithms are the most popular. The dissertation work proposes analytical models for studying the performance of locking algorithms in distributed database systems. Models are developed for fully replicated databases which can be further extended for partitioned and partially replicated databases.

The models are based on mean value analysis for data contention and queuing network models for resource contention. Only the average value of the database state, which is the number of locks locked, is used in modeling data contention. The model results in a set of simultaneous polynomial equations, which are solved by an iterative procedure, to estimate through put and response time. A simulator will be developed for validating the analytical models and the results from the analytical models agree quite well with the simulation results.

The results indicate that under certain restrictions, readwrite access is equivalent to exclusive access and, also, the same is true for non-uniform access of the database. The network size has considerable effect on the performance. The optimal granularities for transactions were found. In general, coarse granularity is preferred for large transactions and fine granularity for small transactions. Small transactions are preferred over larger transactions.

CONTENTS

Chapter 1

INTRODUCTION

Distributed database systems (DDBS) are becoming increasingly popular in the modern day information system world. Some examples of such systems are airline reservation systems, banking systems, medical systems and library information systems. The Database Management System allows the users to access and modify the data of the database. The concurrent access to the database have to be synchronized in order to maintain data consistency and to ensure each transaction will be executed correctly. There are number of concurrency control algorithms proposed in the literature for synchronizing the transactions. The performance of these algorithms should be well understood for designing distributed database systems. In this thesis an analytical model for studying concurrency control in distributed database systems is proposed.

Technological and management reasons for the increasing popularity of distributed database systems.

1. **Organisational and economic reasons :** Many organisations are decentralized, hence distributed database approaches fit more naturally into the structure of the organizations. Also, the cost of having large centralized computers is quite high when compared to small distributed systems.

2. **Interconnection of existing database :** When several databases already exist in an organisation and global applications arise, then managing information as a distributed database is a natural choice.

3. **Incremental growth :** Distributed database approaches support a smooth incremental growth with minimum impact.

4. **Reduced Communication overhead :** Many of the applications are local and having these applications at the local node clearly reduce the communication cost.

5. **Reliability and availability :** In distributed database systems the data can be stored redundantly at several nodes to increase reliability and availability. Also, this approach allows graceful degradation in the event of failures.

6. **Performance considerations:** The existence of several autonomous processors results in increased capacity through parallelism.

These distributed database systems are complex systems and there are number of issues in designing distributed database design. Some of these issues are:

1. **Logical design :** deals with the problems of mapping the real world into the realm of database. In the case of relational databases it corresponds to developing the relations.

2. **Fragmentation :** The purpose of fragmentation is to determine non-overlapping fragments of the database that can be allocated to different sites.

3. **Data distribution :** How the fragments are to be allocated over the network in order to achieve improvements in the performance like response time, cost etc.

4. **Query Optimization:** How the query processing has to be distributed in order to minimize cost and response time.

5. **Integrity and Concurrency Control:** Deals with the issue of maintaining consistent and coherent data in the database while users are concurrently accessing and modifying the data.

6. **Recovery :** Deals with the issue of how the data can be recovered to its consistent state in the event of system failures and user transaction failures.

7. **Privacy and Security:** Deals with the issue of giving access to authorized persons and preventing unauthorized accesses.

In this work only the performance of concurrency control algorithms are dealt with.

1.1 Concurrency Control

In a multi-user database management system a number of users may be simultaneously accessing the database. If this accessing is not controlled it can lead to corruption of the database and/or incorrect execution of user transactions. The concurrency control must maintain both internal consistency and mutual consistency. Internal consistency is the set of assertions that must be true for the data items of the database.

Mutual consistency requires that the various copies of the data item at different sites have the same value. Mutual inconsistency arises due to the unpredictable message delays between different computers. The main goal of concurrency control algorithms is to prevent interference among the users and maintain both internal and mutual consistency.

Concurrency control and Mutual Exclusion problems:

The database concurrency control problem is quite similar in some respects to the problems in classical concurrent systems such as operating systems. Operating systems coordinate the access by concurrent processes to system resources such as CPU, memory, and I/O devices. The basic problem is that of mutual exclusion i.e. to ensure that each resource is accessed by at most one process at any given time. Common solutions to these problems include semaphores, critical regions, and monitors. But, the problem of database concurrency cannot be solved by just guaranteeing mutual exclusion. This is illustrated by the following example. Let processes p1 and p2 access resources R1 and R2 one interleaving execution order is P1 uses R1 P2 uses R1 P2 uses R2. P1 uses R2. This ordering is acceptable in operating systems. But in the case of database systems, if R1, R2 corresponds to two accounts of a user and P1 is transferring money from account R1 toR2 and P2 is computing the sum of the two accounts, then the above computation will be unacceptable.

Another difference is that, the operating system handles resources that possess some robustness. Whereas the "data" handled by database systems are more delicate and improper handling leads to incorrect values of the data, and by propagation it may soil (corrupt) the entire database.

As a result, control schemes that work for one do not necessarily work for the other and hence most of the work done in the synchronization of concurrent processes cannot be directly used for concurrency control in the database: Concurrency control in centralized DBS is well understood and Two phase locking has been accepted as a standard solution.

1.2 PERFORMANCE ANALYSIS:

Performance evaluation has played an important role in the development of computer systems. The need for performance evaluation and predication exists from the initial conception of a system's architectural design to its daily operation after installation. It helps the designer to find out how the system behaves under different load conditions and also helps in identifying potential system bottlenecks.

Concurrency control (CC) algorithms affect the performance of the DDBS. The database designer would like to know how the CC affects the performance of the system. Also, there are many algorithms in the literature. Database designers are faced with the difficult task of choosing the most appropriate CC algorithms for their needs.

Performance studies and comparisons can be done by qualitative analysis and quantitative analysis. In the qualitative methods one mainly compares the message overhead added by the CC algorithms. Though this yields some insight into the behaviour of the algorithm it will not give a complete picture. As a result it is difficult to draw conclusive results from these studies.

Quantitative analysis is the main approach used for comparison. There has been extensive work done in this area for both centralized and distributed database concurrency control algorithms. Performance studies have been carried out by both simulation and analytical work. The major advantage of simulation methods over analytic methods is that they can be used to model the system with great detail involving complex flow control of the system. But their major disadvantage is that they require an enormous amount of effort and resource to gain comparable confidence in the results. This problem is quite acute in CC studies because of the large number of parameters involved and parametric studies of the entire parameter set are not practical. Analytical methods have the advantage of requiring much less time to obtain the solution and also it often yields greater insight into system's behaviour. But, the major disadvantage of this method is modeling is more difficult and the solution techniques may become intractable.

1.3 WORK OBJECTIVES

The aim of this work is to develop analytical models to study the performance of concurrency control in distributed database systems. The models proposed should be general enough to model various algorithms and also overcome many of the shortcomings. As locking approaches are the most popular form of concurrency control, the models and performance analysis will mainly concentrate on these schemes. By developing these models and performing studies it is hoped to shed some light on the underlying principles involved in distributed database design. Summary of the goals are: To

1. Develop analytical models
2. Validate these models using simulation
3. Predict how the system behaves under variation of different parameters effecting the performance and draw some definite conclusions about the tradeoffs involved in the design process.

1.4 Organization of this thesis

Chapter 2 describes the various concurrency control algorithms for distributed database systems and the performance studies carried out by other researchers. In the first half of the chapter the various concurrency control algorithms including two phase locking is reviewed, which is the algorithm modeled in this work. The second half of the chapter deals with performance analysis.

In Chapter 3 the details of the algorithm and the environment of the systems being modeled are given.

Chapter 4 presents the analytical model developed and its validation.

The simulation to be used for validating the model is described in chapter 5.

In chapter 6, the parametric analysis and the optimal granularity problem are presented.

Finally the summary and conclusions are presented in chapter 7.

CHAPTER 2

2.0 BACKGROUND

Background literature surveys in both the concurrency control algorithms and the performance analysis are presented in this section. In section 2.1 concurrency control algorithms are described. Among these various algorithms the most popular one is the two phase locking algorithm and it is the one that is modeled in this thesis. Section 2.2 discusses the performance studies carried out by other researchers. These include both simulation and analytical modeling of centralized and distributed database systems. Among the analytical models the most important so far as this thesis is concerned is the Mean Value Analysis model for centralized data base system by Tay etal. The data contention model developed in this thesis is based on the mean value analysis, which is an extension of the Tay etal's model.

2.1 Algorithms

In this section, first we define some of the terms used in the later discussion. Afterwards, some of the concurrency control algorithms proposed will be described.

2.1.1 Distributed Database Terminology:

A database is a collection of shared, named, data items. These may be files, records, or fields within records and it is unspecified in our current discussion. A database state is defined as a mapping from the set of items of the database to the set of values. A distributed database is one in which the database is distributed over a number of sites of a computer network. There are number of different ways data could be distributed.

I Fully replicated – each site contains a complete copy of the database

II Partitioned database –each site contains a unique subset of the database

III Partially replicated – some partitions of the database are stored at more than one site.

Users interact with the database system by executing transactions. Transactions may be application programs written in a high level language or an on-line query written in a query language. A transaction obtains data from the database by issuing read commands and modifies the data by issuing write commands. The read–set of a transaction is the set of data items it reads and its write-set is the set of data items it writes. Two transactions conflict if the read–set or the write-set of one intersects with the write-set of another transaction. Access to the database is controlled by the database management system (DBMS).

The concurrency control algorithms must satisfy two major requirements. 1. Users must see the database as if the only transaction executed is theirs. 2. Transactions submitted must terminate in a finite time.

If the concurrent execution of a set of transactions is equivalent to some serial execution of the set of transactions, then the concurrent execution is said to be serializable. Any serializable execution also preserves database consistency. In the case of distributed database systems the same serial order should be maintained among various sites in order to maintain mutual consistency.

One of the cases by which a user transaction never terminates is by dead lock. Deadlock is caused by circular waiting of transactions. Other reasons are indefinite postponement, infinite chain of distinct waiting transactions, and waiting for a transaction which runs forever.

Once the transaction's computation is successful then it is ready to commit. The transaction should be committed at all the sites where it has accessed or changed the data, even if there are failures. This is ensured by the commit protocols.

It should be noted that serialization is a sufficient condition but not a necessary conditions for maintaining consistency. For example if the two transactions have commutative property between them, then they can be executed at two different sites in different orders and still maintain consistency.

2.1.2 Concurrency Control Algorithms

(I) **TWO PHASE LOCKING:** In two phase locking transactions are synchronized by explicitly detecting and preventing conflicts between concurrent operations. This is achieved by making the transactions lock the data item before they access it. A transaction must access a read lock on x before it can read data item x and similarly it must access write lock on x before it can write x. A transaction cannot own a read lock or a write lock on x if another transaction owns a write lock on x. The name is two phase locking because, transaction execution states can be divided into two phases – lock acquisition and lock releasing. In the first phase transactions accumulate all the locks. During the second phase the locks are released. Once a transaction surrenders its ownership of a lock it cannot obtain any additional lock, i.e. the transaction enters second phase. It has been proven that two phase locking guarantees serial order of execution of transactions. But this algorithm can lead to deadlocks. So additional facilities to detect dead locks or prevent dead locks should be incorporated.

If all the locks are acquired at once in the beginning of the transaction then it is know as static locking. If the locks are acquired as need during the execution of the transaction then it is known as dynamic locking.

(II) TIMESTAMP ORDERING: In time stamp ordering a serialization order is selected apriori and transaction executions are forced to obey this order. Each transaction is assigned a unique timestamp and all conflicting transactions are required to be processed in timestamp order. These timestamps are a monotonically increasing sequence of numbers which is often a function of the time of day. In the case of distributed systems, timestamps are generated by appending the site number to the timestamp generated at that site, thus ensuring uniqueness. Timestamps can be used to avoid deadlocks in database systems which use locks or it can be used without the locks. Timestamp ordering guarantees serialization order.

(III) **Optimistic control:_** This algorithm is based on the fact that conflicts are rare. Transactions are allowed to access the data without any restriction. Once the transaction is complete and ready to commit, it goes through a validation phase, in which it is checked with the other transactions to see whether any consistency constraints will be violated if we validate this transaction. If there is no consistency violation then the transaction is accepted otherwise it is aborted.

2.1.2.1 Algorithms based on locking:

(i) **Centralized two phase locking:** One of the site, known as the central site manages all synchronization. Before accessing data from any site, a transaction obtains the appropriate locks from the central site. Once the update is computed it is broadcasted to all the sites containing the update data items. The updated transaction releases the lock only after updating is finished. One of the main disadvantage of this scheme is that the central site becomes the bottleneck of the system. Also, even if a transaction has to access local data it has to obtain a lock from central site leading to poor response time and a high communication burden.

(ii) **Primary copy Algorithm:** This is a variation of the centralized two phase locking algorithm. Here one copy of each data item is selected as its primary copy. Before accessing data from any site, a

transaction must obtain the appropriate lock from its primary site. In this way the problem of heavy traffic at one central node is avoided.

(iii) **Two phase voting algorithm:** This is derived from the majority consensus algorithm of Thomas. A transaction that wants to write x sends its lock request to all the nodes which has x. Upon receiving the request x the site sends “Lock set” or “Lock blocked” to the requesting site. Once the transaction gets a majority of the “lock set” messages, it will act as if it received all ”Lock set” messages. Otherwise it waits for the “lock set” messages from the sites which originally sent “Lock blocked”. Dead lock can occur, so deadlock detection or prevention methods must be used.

2.1.3 Transaction Commit:

In the previous section various concurrency control methods that prevent interactions between transactions were described. Even using these algorithms inconsistency in distributed databases can arise if transactions are allowed to commit at some sites while at other sites they are aborted. A standard approach to avoid such partial commit is by using the Two phase commit.

2.1.3.1 TWO PHASE COMMIT:

Two phase commit ensures that either the transaction is committed at all the sites or aborted at all sites. This property, that transactions commit at all sites or none, is referred to as atomic commit. The site, where the transaction originated, coordinates with all other sites. This transaction originating site becomes the commit coordinator and all other sites which participates in the committing of the transaction become participants. When the transaction is ready to commit the two phase commit protocol is started.

In the first phase the coordinator writes a “Prepare” record in the log and sends a PREPARE message to all the participants. All the participants after receiving this PREPARE message, record all the values to be updated in stable storage, send a READY message, if it is ready to commit, otherwise it sends an ABORT message. The coordinator decides whether to commit or abort the transaction after receiving an answer from the participants. If all the messages received were READY then the coordinator decides to commit the transaction. Instead if some of the messages were ABORT then it decides to abort the transaction. In the second phase the coordinator records either Global-commit or Global-Abort on its stable storage and accordingly sends either a COMMIT or an ABORT message to all the participants. Participants receive this message and they either commit or abort according to the message received. Participants then send an ACK message to the coordinator. The coordinator receives this message and records “ complete” in its log. By applying this protocol the transaction is committed at all sites or aborted at all sites. It can be shown that the above protocol is resilient to almost all failures.

2.2 **Performance Analysis**

This section describes and discuses performance analysis of CC algorithms. The most common metrics are discussed.

2.2.1 Metrics :

The most common measures are response time and throughput. Other measures also used are restart rate, degree of concurrency and utilization of the system resources. These performance measures are functions of a number of systems parameters like the number of nodes, size of the database, transaction size, communication characteristics and computer system characteristics. Performance, also, depends on the database access methods, data distribution etc.

Two different performance metrics are Promptness and Coherence. Promptness is the speed with which multiple copies reach the same value with respect to an instant update of replicated data i.e., it tells us how up-to-date node is at a given instant of time. Coherence reflects the mutual difference of all replicated copies at a given time. These were used to discuss performance of time stamp mechanisms.

2.2.2 Simulation Studies :

Ries developed a simulation model to study concurrency control algorithms. The model is a closed queuing system with each site having a CPU and an I/O unit. His model considered a large number of input parameters (about 20) describing the database system. The main output parameter was utilization of CPU and I/O. The results indicated that coarse granularity is better if the transactions size is large But, fine granularity is better if the transaction are small and the lock placement is either random placement or worst case placement.

2.2.3 Analytical Studies :

In recent years a number of analytic methods have been proposed for studying the performance of CC algorithms. Many of these are based on queuing or Markovian models, other approaches like mean value analysis are also being used.

2.2.3.1 Mean Value Analysis :

Tay etal proposed a mean value analysis for a centralized database for locking schemes. In this scheme the entire analysis is done by using steady state average values and using a flow diagram. A set of algebraic equations can be easily written from the flow diagram using conservation principles. Flow diagrams will be described in more detail in the model section. The model is general enough to model multiple transaction classes, nonuniform access, sharable as well as exclusive locks, static and dynamic locking schemes, transactions with fixed length as well as transactions with variable lengths. In this model there is differentiation between data contention and resource contention.

Summary : Through there are various CC algorithms, most of them fall into one of the basic three algorithms – Locking, Timestamped algorithm, Optimistic Control. Two phase locking is one of the most popular algorithms. In the performance studies section both simulation and analytical studies were reviewed. The analytical model developed in this thesis is based on Mean Value Analysis and Queuing Network models.

CHAPTER 3

3.0 Database Environment

Performance analysis of concurrency control algorithms based on locking schemes is carried out by analytical modeling. As mentioned in the previous section this model is based on Mean Value Analysis. In this section first the environment of the database system modeled is described and later the details of the algorithm being modeled are presented.

3.1 Specification of the Environment

Before we start modeling the system, the environment in which the system works should be specified. A distributed database environment can be described in terms of the database, resources used and the transactions.

Database Model: The database consists of a set of data items and these data items are grouped into granules which are units of locking. A granule may be a page or a record Let N be the number of data items and D be the data granules, with each granule containing N/D data items. Each granule is considered to have C copies in the database system. If C is equal to 1 then it is a partitioned database, and if C is equal to S (number of sites) then it is a fully replicated database. In this thesis fully replicated case is considered.

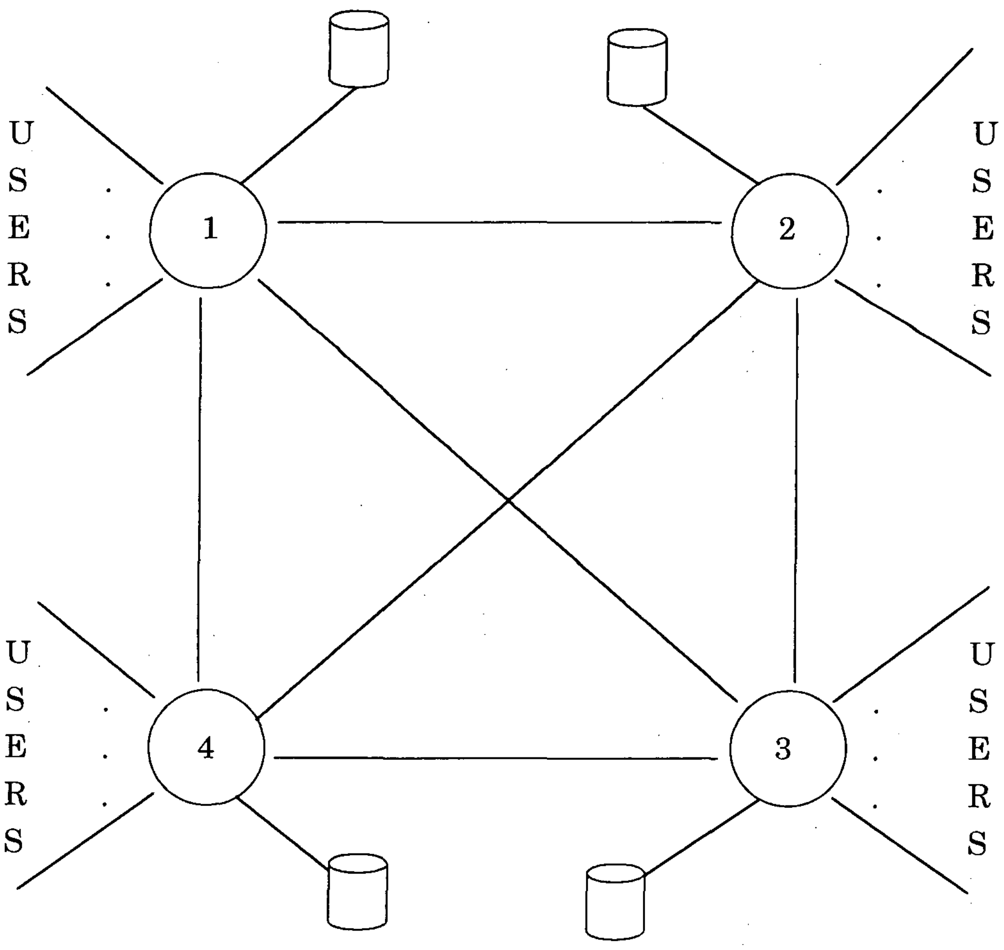


Fig. 3.1.1 : Fully connected Network with Four Sites

3.1.2 Resource Model

There are S computer sites in the system. Each of these sites is considered to be identical. Each site has a CPU, an I/O device and communication channels. The computer system is modeled by a single FCFS server and it will be referred to as the processor in the following sections. All the local computations and I/O accesses (for each request) are grouped into a single service unit. The communication channels are modeled as single exponential servers with FCFS scheduling. The communication network can have any topology. In this study it is restricted to fully connected topology.

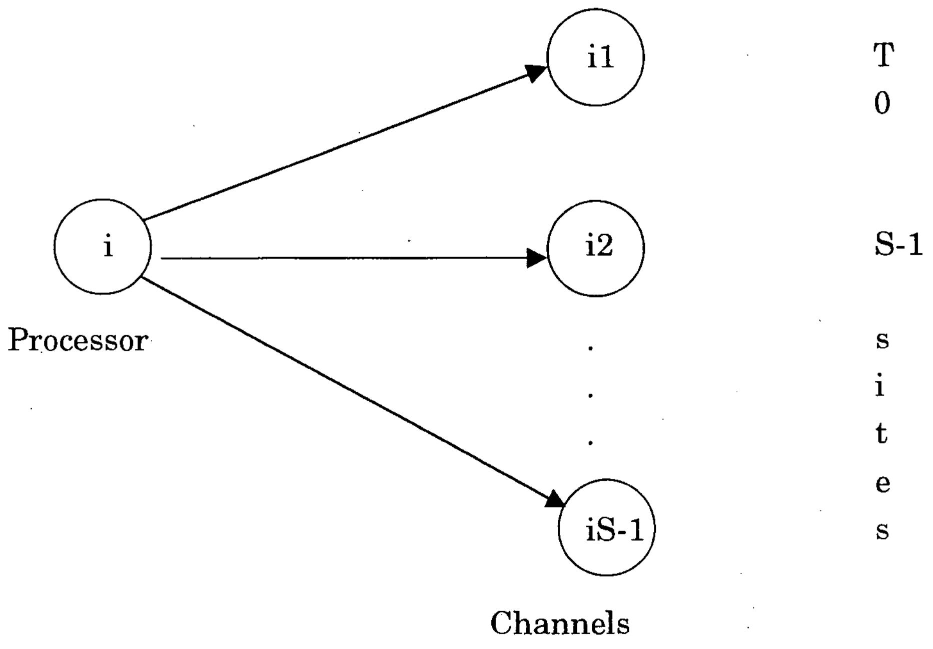


Fig 3.1.2 : Servers at site i

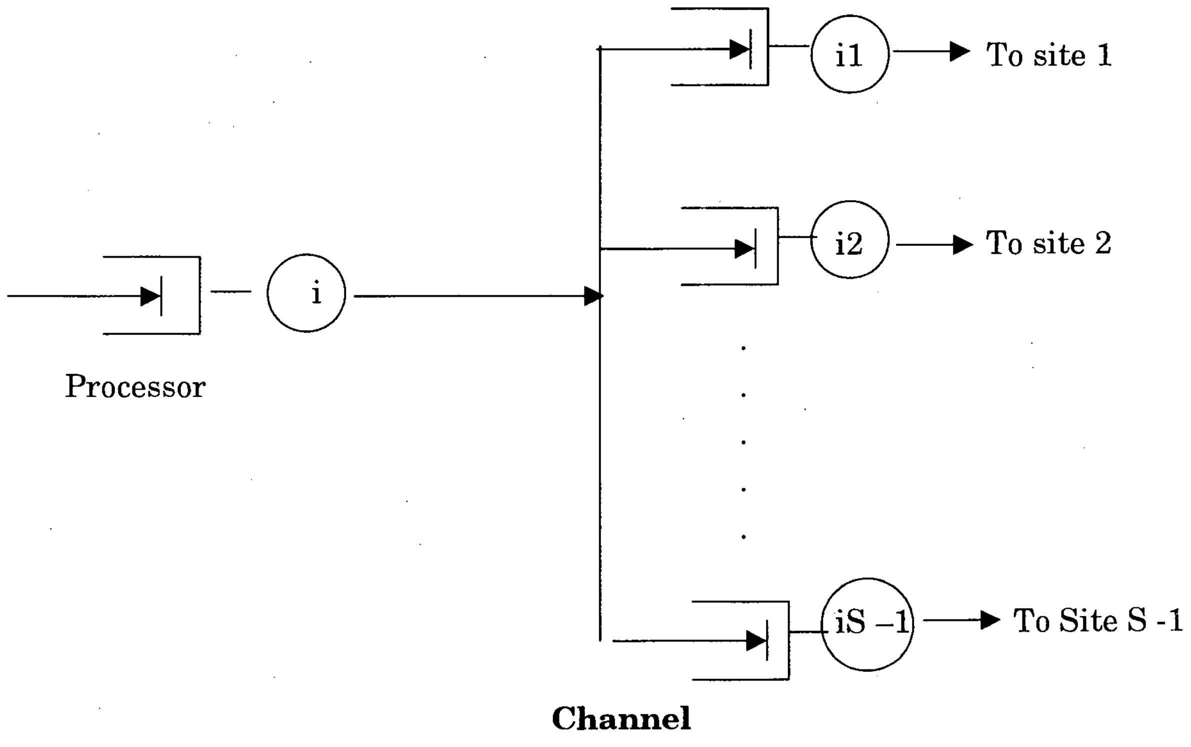


Fig. 3.1.3 : Queuing Diagram at site i

3.1.3 Transaction Model

Transactions arrive at each of the nodes (sites). Their arrival process is assumed to be a Poisson process with mean inter arrival rate of λ_i at site S_i. Each transaction reads a set of data items (read set) and updates a set of data items (write set). Transactions are required to lock the data granules before they access the data item in them. Also, transactions do not request locks (granules) they already hold. When a transaction requests a lock it is granted immediately, if it is not already held by some other transaction (no conflict). If there is a conflict the transaction is either restarted (in No Waiting Case) or will wait for the lock to be released (in Waiting Case). If the transaction is restarted then the transaction releases all the locks it held and resubmits its requests from the beginning. If the transaction is resubmitted immediately then it is most likely to conflict again. In order to avoid this, the transaction is delayed and then restarted. This delay known as the restart time is a tunable parameter.

The concurrency control algorithm modeled in this thesis is a locking algorithm, in particular it is Two phase locking with Two phase commit protocol. The detailed description is given below.

- **Transaction originating site**

First phase

```
Request All locks at Local site
            if there is a conflict then
            Abort transaction

else
            Process the transaction
            Compute the Update Set
            Broadcast LOCK-REQ alongwith
            Update set to all sites
            Wait for LOCK-GRANTED from all sites OR
            LOCK-REJECTED from any site;
```

Second phase

```
If any reply is LOCK-REJECTED
then
          ABORT the transaction
          Broadcast ABORT message
else
(* all replies LOCK-GRANTED*)
          Update local database
Release local locks and Broadcast
"UPDATE" message to all the sites
Wait for UPDATED message from all sites
```

TH-10220

```
After all replies received Transaction
Execution is complete
```

- **Participating site**

First phase

```
Create subtransaction once REQUEST message is received
If the sub-transaction conflicts then send LOCK-REJECTED
Abort sub-transaction
else (* no conflict *)
Send LOCK-GRANTED
                    Wait for UPDATE message
```

Second phase

```
If ABORT is received then
Abort sub-transaction
else (* UPDATE received *)
                    Update the database
                    Release locks
                    Send UPDATED message
```

CHAPTER 4

4.0 ANALYTICAL MODEL

The model developed in this thesis is an extension of mean value analysis model. It is rather simple and the solution time does not depend on the parameters of the system. The model is flexible enough to approximate different algorithms.

In distributed databases, because of multiple copies of the database, the CC algorithms are more complex and there are more parameters affecting the performance. As a result modeling of DDBS is more complex and involved. In a DDBS transactions can conflict (when locks are requested) at either local or remote sites. If the database is replicated then the transaction sees different database states at the local site and remote sites. This is true even in the case of fully replicated databases. So the analysis should address these two cases (local and external transactions) separately.

The analytic model development will be in phases. In the beginning a simple algorithm is considered – Static lock with exclusive locks and fully replicated. All the locks are requested at the beginning of the transaction and whenever a conflict occurs the transaction will be aborted. Later different cases of this algorithm will be discussed.

4.1 DATA CONTENTION AND RESOURCE CONTENTION MODELING

The general approach to modeling is to derive a set of equations describing the behaviour of the system using the flow diagram for the data contention part and a queuing network model for the resource contention part. While deriving the set of equation for the data contention part, only the mean value will be used. In the next two subsections these aspects are described in detail.

4.1.1 Data Contention Modeling using Flow Diagram

Flow diagram for Static Locking

Before we draw the flow diagram, we repeat some of the assumptions regarding the database and transactions.

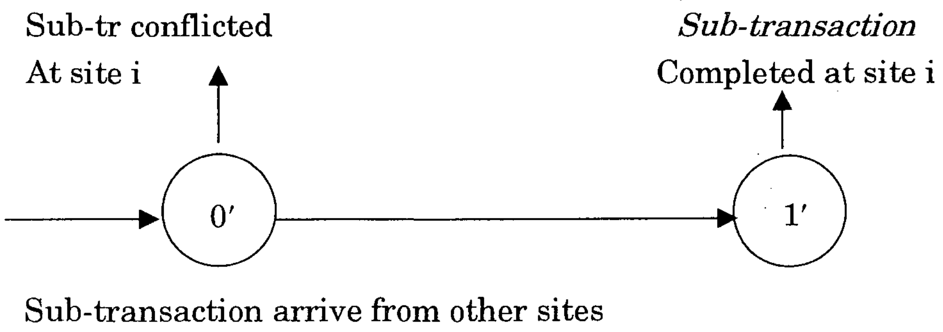
Assumptions

(i) The database is fully replicated

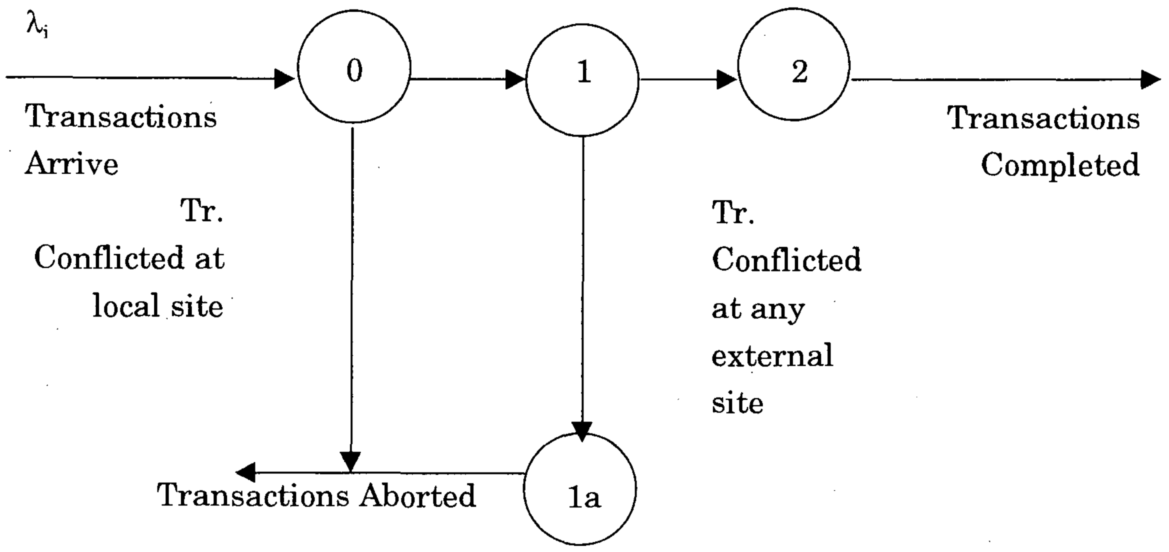
(ii) Locks are uniformly distributed

(iii) All the locks are requested in the beginning of the transaction

(iv) Transactions are aborted if they conflict.



Sub-transactions from other sites



Transactions from local site i

Fig. 4.1.1 : Flow Diagram at site i for Static Locking with No Waiting.

Transactions arrive at a rate of λ_i at site i. When the transaction starts it will be in stage 0. While in that stage request for all local locks will be made (static locking). If it is in conflict with other transactions, it is aborted. Otherwise it goes to stage 1. It then broadcasts LOCK-REQ to all the other sites and waits for replies. If all the replies are LOCK-GRANTED it goes to stage 2 otherwise (in case of conflict) it is aborted and an abort message is sent to all the other sites. In stage 2 the transaction sends an UPDATE message and once it receives all the replies, it leaves stage 2. The transaction is said to be completed after leaving stage 2. These sub-transactions request lock while in stage o′. The current site replies LOCK-GRANTED if there is no conflict and the sub-transaction goes to stage 1′ where it waits for an UPDATE message from the originator. When the originator site sends an UPDATE message, the remote site updates the data and send an UPDATED message to the originator. This completes the sub-transaction at the remote site. The set of equations for data contention are derived by finding the arrival rates into each stage, the departure rates from each stage and the number of locks in each stage. To determine the departure rates the probability of conflicts in each stage are evaluated. The number of locks in each stage are determined by evaluating the number of transactions in each stage, which are evaluated using Little's theorem.

4.1.2 Resource contention modeling

Computer network system are modeled as networks of queues. Each service centre is represented by a queue and a corresponding server. A diagram for a fully connected network and the corresponding queuing diagram for a three site system is given below. For the diagram below the CPU and I/O processor are combined into a single exponential server.

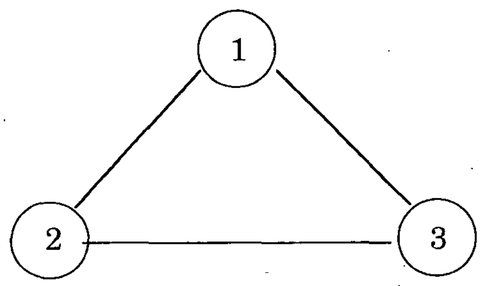


Fig 4.1.2 : Fully connected network with three sites

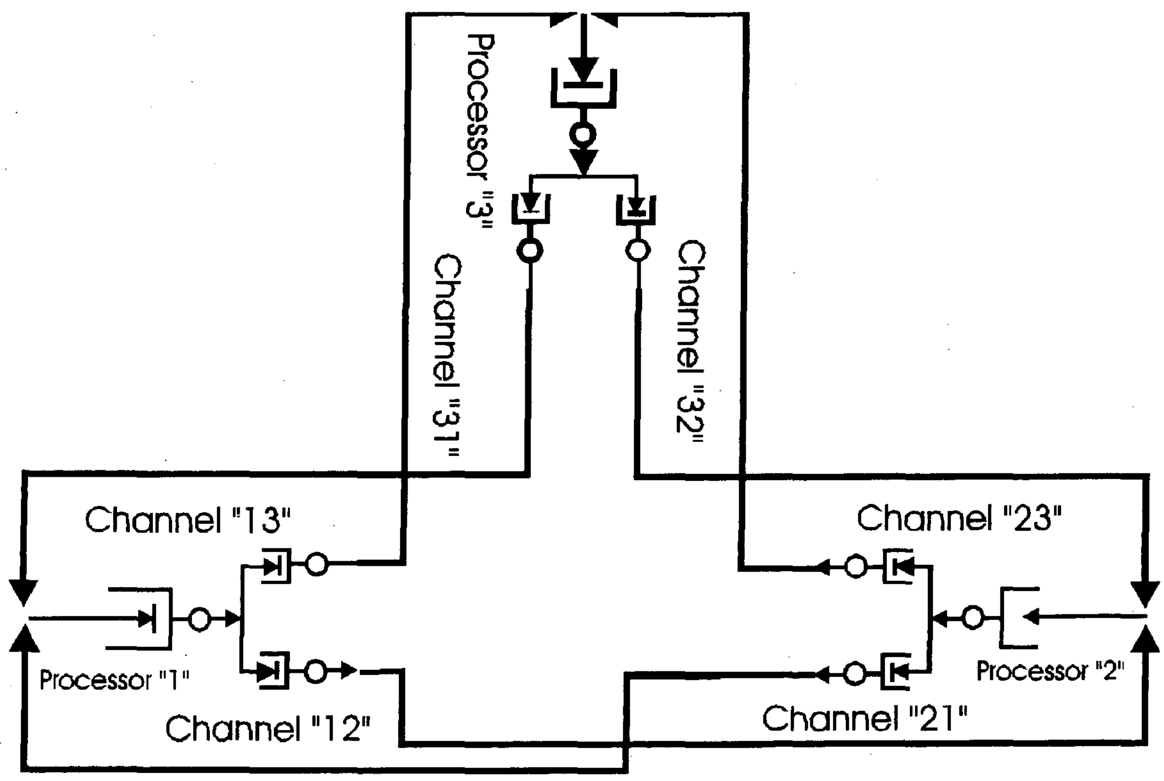


Fig 4.1.3 : Queuing Diagram for the Network in Fig 4.1.1

The transaction arrival process is assumed to be a Poisson process. At each site each of the servers is an exponential server and service times at each station are the same for all transaction types. The above network of queues behave like independent M/M/1 queues with input rate equal to the sum of all the arrivals at that node. Hence we can derive expressions for the time spent at each of the service centres.

4.2 STATIC LOCKING WITH EXCLUSIVE LOCKS

In this section the set of equations for a fully replicated database system in which each transaction requests exactly K exclusive locks will be derived.

Let

$P_i(1)$ = Probability that a transaction conflicts with other transactions at its local site i.

$P_i(1')$ = Probability that a transaction does not conflict with other transactions at its local site i = 1- $P_i(1)$

$P_{ij}(g|1')$ = Probability that a transaction from site i conflicted with the transactions from site j at site j, knowing that it did not conflict at local site i.

$$P_{ij}(g'|1') = 1 - P_{ij}(g|1')$$

$P_i(G|1')$ = Probability that the transaction from site i conflicted at any of the external sites, knowing that there was no conflict at local site i.

$$P_i(G'|1') = 1 - P_i(G|1')$$

$P_i(G', 1')$ = Probability that the transaction originating from site i does not conflict at either the local site or at any of the external sites.

Notations : The term 1, 1′, g, g′, G and G′ inside the parenthesis of the p's are not variables, instead they are used as qualifiers to the conflict terms. The "i" stands for local site, "g" for an external site (global) and "G" refers to an event involving all the external sites. The "bars" indicate the NOT of the term. Referring to the flow diagram (Fig. 4.1.1), $P_i(1)$ corresponds to probability of conflict in stage 0, $P_{ij}(g|1')$ corresponds to probability of conflict in stage 0′, due to transactions at site j and $P_i(G|1')$ corresponds to probability of conflict in stage 1. We assume that all sites are identical with respect to the service rates and the inputs transaction rates. Hence, all P_i (1), P_{ij} (g) and P_i (G,1)'s are equal and we can drop the subscript i. A new transaction must first acquire locks at its local node. Since locks on data granules are exclusive and each transaction requests exactly K locks, the probability that the truncation does not conflict with other transactions at its local site is

$$P(1') = \frac{\binom{D - N_L}{K}}{\binom{D}{K}}$$

Where N_L denotes the number of locks already locked at the local site. The numerator in the above expression gives the number of ways of choosing K locks from $D - N_L$ unlocked granules. The denominator gives the total number of ways in which K locks can be chosen from the database with D granules.

$$P(1') = \prod_{j=0}^{K-1} \frac{(D - N_L - j)}{(D-j)}$$

The number of locks requested by a transaction is small when compared to the number of locks in the database. Also only a small portion of these locks are locked by the active transaction, i.e. $D - N_L >> K$. Hence the expression for $P(1')$ can be approximated by

$$P(1') = \left(1 - \frac{N_L}{D}\right)^K \qquad (1)$$

Once the transaction gets the locks at its local site then it will request locks from all the other sites. The cohort sites reject the request only if the locks requested have already been acquired by another transaction. To compute the probability that a transaction conflicts at any of the external sites, first the probability that the transaction does not conflict at an external site with the transactions from that site is computed. The sub-transactions will conflict only with transactions in stage 1 (N_1 Locks). Locks from stage 2 is known globally and these locks were tested while acquiring local locks. The probability that a transaction does not conflict at an external site with the transactions from that site given that there is no conflict at the local site is :

$$P(g' \mid 1') = \frac{\begin{pmatrix} D - N_L \\ K \end{pmatrix}}{\begin{pmatrix} D - (N_L - N_1) \\ K \end{pmatrix}}$$

$$= \prod_{j=0}^{K} \frac{(D - N_L - j)}{(D - (N_L - N_1) - j)}$$

$$\approx \left(1 - \frac{N_1}{D - (N_L - N_1)}\right)^K$$

The transaction can conflict at any of the external sites. We assume that the transactions arriving at different sites are independent. Hence, the probability that a transaction does not conflict at a particular site is independent of the transaction conflicting at other sites. Therefore,

$$P(G' \mid 1') = \prod_{j=1}^{S-1} P(g' \mid 1')$$

$$= \left[P(g' \mid 1') \right]^{S-1}$$

The probability that a transaction does not conflict at all is :

$$P(G', 1') = P(1')\ P(G' \mid 1') \qquad \text{(due to Bayes Theorem)}$$

So,

$$P(G', 1') = \left(1 - \frac{N_L}{D}\right)^{K} \left(1 - \frac{N_1}{D - (N_L - N_1)}\right)^{K(S-1)} \qquad (2)$$

Evaluation at N_L and time spent in each stage:

N_L is the number of locks locked at a site, which is the sum of locks held by transactions in each stage,

$$N_L = N_0 + N_1 + N_{2a} + N_2 + N_0' + N_1' \qquad (3)$$

Where N_j is the number of locks held by transactions in stage j.

Little's theorem states that for any stage,

$$L = \lambda * T$$

where,

L	=	Number of transactions in a stage
λ	=	Arrival rate into that stage
T	=	Time spent in that stage

By knowing the arrival rates and the time spent in each of these stages, we can calculate the number of transactions in each stage. Since each transaction requests exactly K locks, the number of locks in stage i is

$$N_i \quad = \quad \text{Arrival rate} * \text{Time spent} * K$$

The transactions access the physical servers like processor's to lock, unlock and update the database and channels, to transmit messages from one site to another site. If t_p and t_c are the waiting time + service time at the processor and channel respectively, then we can calculate the time spent in each stage in terms of t_p and t_c. For calculating t_p and t_c we need the arrival rates at the processor and channel. So the arrival rates at the servers due to transactions in each stage are also evaluated in the following section. Let λ be the arrival rate of transactions at any site. Each site has a processor and S-1 communication channels. Let λ_P be the arrival rate of transactions (or sub-transactions) at the processor and λ_C be the arrival rate at each of the channels. Also, λ_{ip} and λ_{ic} denote the arrival rate at the processor and channel respectively due to transactions in stage i.

Stage 0:

A transaction upon arrival enters stage O. While in this stage it will access the processor to determine the locks required for this transaction and then lock them if they are free. Hence the time spent in stage 0 is :

$$T_O = t_p$$

The transaction does not hold any locks while in this stage (it is trying to lock), hence: $N_O = 0$

As indicated above, each transaction requests exactly one processor access to lock the database and these transactions arrive at a rate of λ transactions / unit time. Consequently, the arrival rate at the processor due to the transactions in this stage is: $\lambda_{OP} = \lambda$

The transactions do not access the channel while in this stage, hence: $\lambda_{OC} = 0$

Stage 1:

Once a transaction has received the local locks it enters stage 1. While in this stage, the transaction sends lock request messages (LOCK-REQ) to all the sites. Since the database is fully replicated, hence it has to get locks from all the sites. It waits until either all LOCK-GRANTED messages are received or until a LOCK-REJECTED message is received. If T_{1q} represents the time that the transaction waits in stage 1 when there is No conflict at any of the sites and T_{1p} represents the case when the transaction conflicts. Then the various delays are:

1. Communication delay for request message from site i to j = t_C

2. Processor delay for locking at site j = t_P

3. Communication delay for reply message from j to i = t_C

The transaction has to wait for all (from S-1 sites) the replies if the reply message is lock granted.

$$T_{1q} = \max_{\text{S-1 samples}} . \{ t_c + t_p + t_c \}$$

If F(x) indicates the probability distribution function of one of the sampling process, then the PDF of max. Of S-1 samples is given by $(F(x))^{(S-1)}$. The evaluation of the mean of this delay term becomes tedious and complex if we use the exact expression for the F(x). To reduce this complexity we can approximate the expression by a single exponential distribution. If the variance of x is less than the square of the mean then the distribution can be approximated by a distribution whose probability density function (pdf) is

$$f(x) = 1/m \left(\exp(-(x-d) / m)\right)$$

and the parameters m and d are set to expressions given below

$$d = \text{Mean} - \sqrt{\text{Variance}}$$

$$m = \sqrt{\text{variance}} = \text{standard deviation}$$

The mean of the max of n samples can then be evaluated using the following equation

$$\underset{\text{S-1 samples}}{\text{Mean of max}\{x\}} = \text{Mean} + \sqrt{\text{Variance}\left(\sum_{j=2}^{S-1} 1/j\right)}$$

Each of the servers are exponential, hence the variance = mean

$$T_{1q} = \{t_c + t_p + t_c\} + \sqrt{t_c^2 + t_p^2 + t_c^2}\left(\sum_{j=2}^{S-1} 1/j\right) \qquad (4)$$

If the transaction conflicted then it does not have to wait for all the replies and in this case the time spent is

$$T_{1p} = t_c + t_p + t_c \qquad (5)$$

The arrival rate into stage O is $\lambda P(1')$ and hence the number of locks in this stage is

$$N_1 = K(\lambda P(1'))(T_{1q} P(G' \mid 1') + T_{1q} P(G \mid 1'))$$

$$N_1 = K\lambda(P(G', 1')\ (T_{1q} - T_{1q}) + P(1')T_{1p}) \qquad (6)$$

The transactions do not access the processor while in this stage, hence: $\lambda_{1P} = 0$

The transactions broadcast lock request messages to all the other sites, hence: $\lambda_{1C} = \lambda P(1')$

Stage 1 a:

A transaction which has encountered conflict at a remote site enters stage 1a. It releases the local locks and broadcasts ABORT messages to all sites except the site which reported the conflict, but it does not wait for any reply messages. It incurs a delay of t_p to release the locks.

$$T_{1a} = t_p$$

The arrival rate into stage 1a is λ P (1′) P (G′| 1′) and the time spent by a transaction in this stage is T_{1a}. Therefore,

$$N_{1a} = \lambda\, k\, P(G, 1')\, T_{1a} \tag{7}$$

Transactions access the processor to release the locks. The arrival rate at the processor due to this stage is :

$$\lambda_{1ap} = \lambda\, P(1')\, P(G|\, 1')$$

Lock release messages are broadcasted to all sites except the site which reported the conflict, i.e. the lock release messages are sent to (S-2) sites. Since all the sites are identical the conflicts are equally likely. Therefore an external site receives an ABORT message on the average (S-2) / (S-1) times the rate at which transactions conflict at external sites. Consequently, the arrival rate at the channel due to transactions in this stage is

$$\lambda_{1ac} = \lambda\, P(1')\, P(G|\, 1') \left(\frac{S-2}{S-1} \right)$$

Stage 2:

When a transaction receives locks from all the sites then it enters stage 2. The transaction first updates its database then sends update messages to all the sites. After it receives UPDATED messages from all the sites, the transaction leaves the system. The various delays are:

1. Update processing at local site = t_p

2. Communication delay for update message from site i to j = t_c

3. Processing delay at the other sites for updating and releasing locks = t_p

4. Communication delay for reply messages from site j to i = t_c

$$T_2 = t_p + \max_{\text{S-1 samples}} \{ t_c + t_p + t_c \}$$

$$T_2 = t_p + t_c + t_p + t_c + \sqrt{t_c^2 + t_p^2 + t_c^2} \left(\sum_{j=2}^{S-1} 1/j \right) \qquad (8)$$

The arrival rate into this stage is

$$N_2 = k \lambda T_2 P\ (G', 1') \qquad (9)$$

Transactions access the processor to update the database and release the locks. The arrival rate at the processor due to this stage is

$$\lambda_{2p} = \lambda P\ (G', 1')$$

Update messages are broadcasted to all sites. The arrival rate at the channel due to this stage is : $\lambda_{2C} = \lambda P\ (G', 1')$

Stage 0′:

Sub-transactions in this stage are requesting locks at the local node. One access to the processor is required. Hence, the time spent in this stage is :

$T_{O'} \quad = \quad t_p \; ; N_{O'} \quad = \quad 0$ 10)

The arrival rate of sub-transactions into this stage is (S-1) λ. Each sub-transaction requests one processor access and at the end of this access sends back the reply message. Hence,

$\lambda_{O'P} \; = \; (S-1)\,\lambda\,P\,(1')$ (11)

and

$\lambda_{O'C} \; = \; (S-1)\,\lambda\,P\,(1')$

Stage 1′:

The sub-transactions after locking the database send LOCKED messages to their originators and wait for either an UPDATE or ABORT message. This waiting time is composed of:

1. Communication delay for the reply message = t_c

2. Waiting time for other replies at the transaction originating site. In order to compute this, term, consider the expression for T_{1q}. The last term in that expression can be interpreted as the extra waiting time for all the replies to reach the originating site before the transaction can proceed to stage 2. Now consider a sub-transaction in $T_{1'}$, if the

reply message of this sub-transaction reaches the originator first then it has to wait for replies for other sites, but if it reaches last then its waiting time is 0. Hence we can approximate the average wait time as the half of the wait time in T_{1q}.

Consequently, this extra wait time for sub-transactions in 1′ is

$$\frac{1}{2}\sqrt{t_c^2 + t_p^2 + t_c^2}\left(\sum_{i=2}^{S-1} 1/i\right)$$

3. Processing delay at the originating site = t_p

4. Communication delay for the update or release message = t_c

5. Processor delay to update and/or release locks = $t_{p'}$

Consequently the time spent in Stage 1′ is

$$T_{1'} = t_c + \frac{1}{2}\sqrt{t_c^2 + t_p^2 + t_c^2}\left(\sum_{i=2}^{S-1} 1/i\right) + t_p + t_c + t_p \qquad (12)$$

Transactions broadcast LOCK-REQ to all sites immediately after obtaining local locks. As a result any two transactions, from different sites that conflict with each other at other than their local sites will do so at each of the sites and only one of these two transactions will hold the locks at any site until an ABORT message is received. Hence, a sub-transaction will hold a lock half the time on the average, if the originating transaction conflicts at any of the external sites. Thus,

$$N_{1'} = (S\text{-}1)\,\lambda\,P(1')\left(P(G'|1') + \frac{P(G|1')}{2}\right) T_{1'}\,k \qquad (13)$$

The first term in the parenthesis is for the case when the transactions do not conflict at any site. Consequently, the arrival rate at the processor due to transactions in stage 1′ is :

$$\lambda_{1'p} = (S\text{-}1)\,\lambda\,P(G', 1') + (S\text{-}1)\,\lambda\,P(G, 1')\,\frac{(S\text{-}2)}{(S\text{-}1)}$$

and the arrival rate at the channel due to transactions in stage 1′ is :

$$\lambda_{1'c} = \lambda\,P(G', 1') + \lambda\,P(G, 1')\,\frac{(S\text{-}2)}{(S\text{-}1)}$$

The second term in the above two expressions are for the conflict case. In which case the lock release messages are broadcasted to all sites except the site which reported the conflict.

The total arrival rate at the processor's is

$$\lambda_p = \lambda_{0p} + \lambda_{1p} + \lambda_{2p} + \lambda_{0'p} + \lambda_{1'p}$$

$$= \lambda + \lambda P(1') + (S\text{-}1)\,\lambda P(1') + (S\text{-}1)\,\lambda P(G', 1') + (S\text{-}1)\,\lambda P(G,1')\,\frac{(S\text{-}2)}{(S\text{-}1)}$$

Therefore,

$$\lambda_p \quad = \quad \lambda \ (1 + SP(1') + (S\text{-}1)\ P\ (G', 1') + (S\text{-}2)\ P\ (G,1')) \qquad (14)$$

and the total arrival rate at the channel is

$$\lambda_c \quad = \quad \lambda_{0c} + \lambda_{1c} + \lambda_{2c} + \lambda_{0'c} + \lambda_{1'c}$$

$$= \quad \lambda P\ (1') + \lambda P\ (G', 1') + \lambda P\ (G, 1') \frac{(S\text{-}2)}{(S\text{-}1)} \ +$$

$$\lambda P\ (1') + \lambda P\ (G', 1') + \lambda P\ (G, 1') \frac{(S\text{-}2)}{(S\text{-}1)}$$

So,

$$\lambda_c \quad = \quad 2\lambda \left(P(1') + P\ (G', 1') + P\ (G, 1') \frac{(S\text{-}2)}{(S\text{-}1)} \right) \qquad (15)$$

If T_R denotes the response time of a successful transaction then,

$$T_R \quad = \quad T_0 + T_1 + T_2 \qquad (16)$$

As mentioned in section 4.1.2 the queuing system is a network of M/M/1 queues. The sojourn time (waiting + service time) at any server is computed independently of the others and using the standard M/M/1 equations, we have

$$t_p = \frac{1}{\mu_p - \lambda_p} \qquad (17)$$

$$t_c = \frac{1}{\mu_c - \lambda_c} \qquad (18)$$

where μ_p is the mean service rate of the processor; and μ_c is the mean service rate of the channel.

4.2.1 Solution Approach and Validation

Equations 1 through 18 form a set of simultaneous polynomial equations of the order of K * (S-1). The input parameters are the number of sites in the system S, the size of the database D, the number of locks requested by each transaction K, the arrival rate of transactions λ and the service rates of the processors (μ_p) and the channels (μ_c). An iterative method is used to solve these equations to yield the various output parameters such as the probability of conflict, response time etc. In order to validate the model a simulator will be developed. The input parameters used for the comparison will be carried out for various values of K, S, μ_p and μ_c. The range of service rates for the channel and processor will be chosen such that the system varies from light load to heavy load.

4.2.2 Discussion on Results

The probability of conflict increases as the channel service time increases and, it also increases as the processor service time increases. Also the number of conflicts increases as the number of lock requests increases. The rate of change of conflicts with the channel utilization is much higher when the number of locks requested is small. The transaction throughput and the response time is affected mainly by resource contention.

4.3 READ WRITE CASE

In this section we consider the case where the transactions request locks which are either read or write locks. The read locks are for data items which do not get modified by the transactions and hence they can be shared with read locks of other transactions. The write locks are for data items which get modified by the transaction and hence they have to be exclusive locks. The read locks conflict with only write locks of other transactions, whereas the write locks conflict with both read and write locks of other transactions. Apart from this discussion the rest of the assumptions remain the same.

4.3.1 Read-Write Lock Model

Let b be the probability that the lock requested by a transaction is a write lock and it is independent of other locks requested by the transaction. Then the probability of accessing j write locks and k – j read locks is

$$\binom{k}{j} b^j (1-b)^{k-j}$$

The probability that the transaction does not conflict at the local site knowing that it requested j write locks and k – j read lock is

$$P(1') = \left(1 - \frac{N_{WL} + N_{RL}}{D}\right)^{j} \left(1 - \frac{N_{WL}}{D}\right)^{k-j}$$

Where

N_{WL} = # of write locks locked at the local site

N_{RL} = # of read locks locked at the local site

Hence, the probability that a transaction does not conflicts at the local site is given by

$$P(1') = \sum_{j=0}^{k} \binom{k}{j} \left(1 - \frac{N_{WL} + N_{RL}}{D}\right)^{j} \left(1 - \frac{N_{WL}}{D}\right)^{K-j} b^{j} (1-b)^{k-j}$$

Simplifying it we obtain,

$$P(1') = \left\{ b\left(1 - \frac{N_{WL} + N_{RL}}{D}\right) + (1-b)\left(1 - \frac{N_{WL}}{D}\right) \right\}^{K}$$

$$= \left(1 - \frac{N_{WL} + b\,N_{RL}}{D}\right)^{K} \qquad (1)$$

The, transactions after receiving local locks will try to lock at other sites. The transactions with more read locks are less likely to conflict than the transactions with more write locks. Hence while evaluating the probability of conflict at an external site this dependency has to be taken into consideration. The probability that a transaction requesting j write locks and hence k-j read locks NOT conflicting at the local site and NOT conflicting at an external site is :

$$P(g', 1') = P(g' \mid 1')\, P(1')$$

$$= \left(1 - \frac{N_{WL} + N_{RL}}{D}\right)^{j} \left(1 - \frac{N_{W1} + N_{R1}}{D - (N_{WG} + N_{RG})}\right)^{j} *$$

$$\left(1 - \frac{N_{WL}}{D}\right)^{k-j} \left(1 - \frac{N_{W1}}{D - N_{WG}}\right)^{k-j}$$

Where, $N_{WG} = N_{WL} - N_{W1}$

$N_{RG} = N_{RL} - N_{R1}$

$$P(g', 1') = \sum_{j=0}^{k} \binom{k}{j} b^{j} (1-b)^{k-j} P(g', 1' \mid j \text{ W locks})$$

After simplification,

$$P(g', 1') = \left\{ b\left(1 - \frac{N_{WL} + N_{RL}}{D}\right)\left(1 - \frac{N_{W1} + N_{R1}}{D - (N_{WG} + N_{RG})}\right) + (1-b)\left(1 - \frac{N_{WL}}{D}\right)\left(1 - \frac{N_{W1}}{D - N_{WG}}\right) \right\}^{k}$$

The conflicts at external sites are independent of each other, hence the probability that a transaction requesting j write locks and k-j read locks does not conflict at any site including the local site is

$$P(G', 1' \mid j \text{ W locks}) = \left(1 - \frac{N_{WL} + N_{RL}}{D}\right)^{j} \left(1 - \frac{N_{W1} + N_{R1}}{D - (N_{WG} + N_{RG})}\right)^{(S-1)j} *$$

$$\left(1 - \frac{N_{WL}}{D}\right)^{k-j} \left(1 - \frac{N_{W1}}{D - N_{WG}}\right)^{(S-1)(k-j)}$$

The probability of not conflicting at any site including the local site is

$$P(G', 1') = \left\{ b \left(1 - \frac{N_{WL} + N_{RL}}{D}\right) \left(1 - \frac{N_{W1} + N_{R1}}{D - (N_{WG} + N_{RG})}\right)^{S-1} + \right.$$

$$\left. (1-b)\left(1 - \frac{N_{WL}}{D}\right) \left(1 - \frac{N_{W1}}{D - N_{WG}}\right)^{S-1} \right\}^{K} \qquad (2)$$

Time spent in each stage :

It is assumed that the processing times required for locking or unlocking read and write locks are the same. As a result the time spent in each of the stages is the same as in the previous section.

Number of Read and Write Locks :

The number of transactions in each stage is calculated using Little's Theorem. The procedure for evaluating the various quantities follows the one outlined in the EXCLUSIVE LOCK case. The sojourn and also the arrival rates for each stage are the same as in the EXCLUSIVE LOCK case.

Stage 0 :

A transaction upon arrival enters stage O. While in this stage it will determine the necessary locks for this transaction and then lock them if they are free. The transaction does not hold any locks while in this stage (it is trying to lock), hence the number of write and read locks are

$N_{WO} = 0$ and $N_{RO} = 0$

Stage 1:

Once a transaction has received all the local locks it enters stage 1. While in this stage, the transaction sends lock request messages to all the sites (database is fully replicated, hence it has to get locks from all the sites). It waits until either all LOCK-GRANTED messages are received (T_{1q}) or until a LOCK-REJECTED messages is received (T_{1p}).

The arrival rate of transactions into stage 0 is λ P(1') and the number of write and read locks in this stage is:

$$N_{W1} = \sum_{j=0}^{k} j \begin{pmatrix} k \\ j \end{pmatrix} b^{j} (1-b)^{k-j} \quad \lambda P(1' \mid j \text{ W locks})$$

$$(T_{1q} P(G' \mid 1' j \text{ W locks}) + T_{1p} P(G \mid 1' j \text{ W locks}))$$

But as, $p(A \mid B)\, p(C \mid A, B) \quad = \quad p(C, A \mid B)$

$$N_{W1} = \sum_{j=0}^{k} j \binom{k}{j} b^{j} (1-b)^{k-j}$$

$$\lambda \left(T_{1q} P(G', 1' \mid j \text{ W locks}) + T_{1p} P(G, 1' \mid j \text{ W locks}) \right)$$

After simplification the term for N_{W1} reduces to

$$N_{W1} = \lambda (T_{1q} - T_{1p}) P(G', 1') k b_2 + \lambda T_{1p} P(1') k b_1 \qquad (3)$$

where,

$$b_1 = \frac{b}{b + (1-b) \left(\dfrac{D - N_{WL}}{D-(N_{WL} + N_{RL})} \right)}$$

and

$$b_2 = \frac{b}{b + (1-b) \left(\dfrac{D - N_{WL}}{D-(N_{WL} + N_{RL})} \right)^{S} \left(\dfrac{D - N_{WG} + N_{RG}}{D - N_{WG}} \right)^{S-1}}$$

Similarly the number of read locks locked at a site (L_{R1}) is

$$L_{R1} = \sum_{j=0}^{k} (k-j) \binom{k}{j} b^{j} (1-b)^{k-j}$$

$$\lambda \left(T_{1q} P(G', 1' \mid j \text{ W locks}) + T_{1p} P(G, 1' \mid j \text{ W locks}) \right)$$

$$L_{R1} = \lambda\,(T_{1q} - T_{1p})\,P\,(G', 1')\,k\,(1-b_2) + \lambda\,T_{1p}\,P\,(1')\,k\,(1-b_1) \qquad (4)$$

The read locks are sharable. So the number of granules read locked is given by:

$$N_{R1} = D\left(1 - \left(1 - \frac{1}{D}\right)^{L_{LR1}}\right) \qquad (5)$$

Stage 1a:

Transactions which have encountered a conflict at some remote site enter stage 1a. The number of write and read locks in this stage is:

$$N_{W1a} = \lambda\,T_{1a}\,(P\,(1')\,k\,b_1 - P\,(G', 1')\,k\,b_2) \qquad (6)$$

$$L_{R1a} = \lambda\,T_{1a}\,(P\,(1 - b_1)\,k\,(1 - b_1) - P\,(G', 1')\,k\,(1 - b_2) \qquad (7)$$

$$N_{R1a} = D\left(1 - \left(1 - \frac{1}{D}\right)^{L_{LR1a}}\right)$$

Stage 2:

When a transaction receives locks from all the sites then it enters stage 2. The arrival rate into this stage is $\lambda * P(G', 1')$. Write and Read locks in this stage are

$$N_{W2} = \lambda\,T_2\,P\,(G', 1')\,k\,b_2 \qquad (9)$$

$$L_{R2} = \lambda\,T_2\,P\,(G', 1')\,k\,(1- b_2\,) \qquad (10)$$

$$N_{R2} = D\left(1 - \left(1 - \frac{1}{D}\right)^{L_{LR2}}\right) \qquad (11)$$

Stage 0′ :

Sub-transactions in this stage are requesting locks at the local node. They require one access to the processor.

$$N_{WO'} = 0 \text{ and } N_{RO'} = 0 \qquad (12)$$

Stage 1′:

The sub-transactions after locking the database enter stage 1′. Transactions broadcast LOCK-REQ to all sites immediately after obtaining local locks. As a result any two transactions that conflict with each other will do so at each of the sites and only one of these two transactions will hold the locks at any site until a ABORT message is received. Hence a sub-transaction will hold a lock half the time on the average, if the originating transaction conflicts at any of the external sites. Hence,

$$N_{W1'} = (S-1)\,\lambda\, T_{1'}\left(\frac{P(G', 1')\, b_2 + P(1')\, b_1}{2}\right) k \qquad (13)$$

$$L_{R1'} = (S-1)\,\lambda\, T_{1'}\left(\frac{P(G', 1')\,(1\text{-}b_2) + P(1')\,(1\text{-}b_1)}{2}\right) k \qquad (14)$$

$$N_{R1'} \quad = \quad D\left(1 - \left(1 - \frac{1}{D}\right)^{L_{R1'}}\right) \tag{15}$$

and the number of locks at each site are

$$N_{LW} = N_{W0} + N_{W1} + N_{W1a} + N_{W2} + N_{WO'} + N_{W1'} \tag{16}$$

$$N_{LR} = N_{R0} + N_{R1} + N_{R1a} + N_{R2} + N_{RO'} + N_{R1'} \tag{17}$$

Using equations 1-17 and expressions for t_p, t_c, T_0, T_1, T_2, $T_{0'}$, $T_{1'}$ and T_R from section 4.2 we can solve for the various probabilities of conflict, number of locks and the response times. An iterative method is used to solve this system of equations.

Discussion on Results

The input parameters are the same as in section 4.2 except the probability of lock request being write lock (b). When b will be increased, then there will be more write locks in the system, and hence probability of conflict will be higher. The probability of conflict for the same working condition is less in Read-Write case than in previous exclusive lock case. This is due to the fact that a granule can be read locked by more than one transaction and also, the read locks requested by the transaction conflict only with the write lock which are a smaller subset. If the number of read and write lock locked is small compared to the size of the database then, the terms b_1 and b_2 are very close to b. i.e. $b_1 = b = b_2$. Also, the expression for the number of read locks locked can be approximated as the number of read locks requested $N_{Ri} = L_{Ri}$ for all i. With these approximations the Read-Write lock case is equivalent to exclusive locks with the number of locks in database changed from D to D/ b(2-b). As a result by knowing the behaviour of the system for exclusive locks we can predict the behaviour for Read-Write locks.

4.4 FURTHER EXTENSION

Till now only uniform demand of locks has been taken into consideration. It can be further extended for non-uniform demand of locks. It can also be extended for transactions with intermediate number of locks to be demanded. Among the various database models only fully replicated database is taken into consideration which has the scope of further extension for partitioned database cases and partially replicated database cases.

CHAPTER 5

5.0 SIMULATION MODEL

An event driven simulator will be developed to verify the analytical model. The simulator will not only simulate the locking algorithms in distributed database, but it can also be used to simulate a general computer network system. The simulator will be written in C language.

An event queue keeps track of the events that are to be processed in increasing order of time. Simulation time is recorded by simulation clock and it is set to 0 at the beginning of the simulation. An event from the head of the event queue will be taken out and the simulation clock will be updated to the new time and the event will be scheduled.

Inputs to the simulator are:

1. Number of sites (nodes) in the network

2. Number of servers at each site

3. The network topology, routing table which is a S * S matrix, whose elements indicate the next node to be visited in the communication path.

4. Number of locks in the database.

5. Transaction parameters such as number of locks / transaction, probability of write lock, etc.

6. The interarrival distribution (exponential, uniform or deterministic), service distribution of the processors and channel servers.

Processors and communication channels are modeled as servers of a queuing system. When a transaction makes a request to access a server it enters a corresponding queue. These queues are FCFS queues. Transactions from the head of the queue are dequeued and serviced. At the end of the service, a scheduler will be invoked which finds the next step in the transaction execution and schedules the transaction accordingly. Each transaction has a script which contains information like the node at which the transaction originated, transaction identifier, locks it requested, time at which it arrived and the current state of the transaction. The locks requested by the transactions are determined using pseudo random generators of appropriate distribution. Also, the lock type (Read or Write, Uniform or Non-Uniform) are determined at the transaction creation time.

The database state is represented by a lock table which holds information like the locks being locked, the type of lock. When a transaction makes a lock request then the lock table is referred to find whether the request can be granted or not. If the request is granted then the lock is updated to indicate the locks acquired by this transaction.

5.1 SIMULATION OUTPUT ANALYSIS

The method of batch means will be used to collect and process the statistics of the simulation runs. In this method, each simulation is run for time T simulation time units and this time is divided into m batches, each of T/m time units. The results are collected at the end of each batch interval. If $X_1, X_2, \ldots\ldots X_m$ are the means of the one of the performance measures (say, response time) for m batches. The estimate of the mean is

$$E[x] = \frac{1}{m} \sum_{i=1}^{m} X_i$$

and 100 (1 - α)% confidence interval of the estimated mean is given by

$$E[x] \pm t_{\alpha/2;\, m-1} \frac{\sigma_x}{(m)^{1/2}}$$

where,

$$\sigma_x^2 = \frac{m \sum_{i=1}^{m} X_i^2 - \left(\sum_{i=1}^{m} X_I \right)^2}{m\,(m-1)}$$

and $t_{\alpha/2;\, m-1}$ is chosen from the student – t distribution with m-1 degrees of freedom. For the above expressions to be accurate there are certain assumptions that must hold good about X's, which are:

1. X_i, $i \geq 1$, must be a covariance stationary process. This condition can be satisfied, if the sampling is done after the system is in steady state.
2. X_i must be un-correlated, which is satisfied if the batch size chosen is large enough.
3. X_i must be approximately normally distributed, which again is satisfied if the batch size is large (from central moment theorem).

CHAPTER 6

6.0 PARAMETRIC ANALYSIS :

One of the main advantage of analytical modeling over simulation modeling is that extensive parametric analysis can be carried out inexpensively. In this chapter parametric analysis for fully replicated databases is carried out. Also the problem of determining the optimum granularity size is considered.

6.1 Metrics for Performance Analysis :

The probability of conflict and the response times are used for comparison and validation Besides these relative throughput, which is defined as the ratio of rate of successful completion of transaction to the arrival rate of the transaction is also used for the purpose of performance analysis. This term is equal to the probability that a transaction does not conflict either at the local site or at any of the external sites P (G′, 1′). The other output parameters, like the useful utilization of processors and channels have similar behavior to the relative throughput.

6.2 FULLY REPLICATED DATABASE

The network taken in to consideration is fully connected (i.e. each site has a complete copy of the database) and all sites are identical. At each site there is a processor and S-1 channels (S is the number of sites in the system). One of the main parameters of a distributed system is the size of the network.

6.2.1 Effect of Network Size on Performance

The performance of the system with various network sizes under different loads are studied. Variation of relative throughputs and response times, with network size for different combinations of service times are studied. The relative throughput does not vary significantly with respect to the number of sites. Even when the number of locks is increased, the relative throughput doesn't vary considerably. The response times vary considerably with network size. There are two delay factors affecting the response time, one is waiting time for reply messages and the second one is the time delay at the physical servers. For larger networks, the waiting times to receive the reply messages are higher than the smaller networks, as they have to wait for replies from more sites. However, the time delay at the physical servers are higher for smaller networks than the larger networks, because the arrival rate at the channels are higher.

6.2.2 Effect of K on Performance

The variation of k, the number of locks requested by a transaction, has a significant effect on the performance. In order to study the effect of k on the performance, other parameters such as service times of the transaction at the processor and the channel are kept constant. At low values of k the relative throughput is very close to one. As k is increased the performance starts deteriorating. The drop in relative throughput is nearly linear when the service rates of channels and processors are small. If the channel service time is high then the drop in performance is very rapid and it is less sensitive to the changes in the processor service time. However at lower channel service times, changes in processor service times have a significant effect on the performance. So smaller transactions are preferred over larger transactions, even if the processing requirements are the same for both types of transactions.

6.2.3 Effect of Read Locks on Performance

As the probability of read locks increases the relative throughput starts increasing. The increase is more rapid for slower communication links. When all the locks requested by the transactions are read locks then there is no conflict between them, as the read locks are sharable.

6.3 OPTIMAL GRANULARITY

In this section the problem of optimal granularity for fully replicated database systems is studied. In section 4.2, we considered the transactions to be requesting granules (locks) from the database system. But in reality transactions access the data items and these data items are grouped into granules which are units of locking. Let I_D be the number of data granules, with each granule containing I_D/D data items. Transactions are required to lock the data granules before they access the data items in them. Let d be the number of data items required by the transaction and let k denote the number of data granules that are to be locked to access the k data items. The number of data granules required depends on how these data items are placed in the data granules. If the data items are randomly placed then the number of data granules required is

$$K = D\left\{1-\binom{I_D - I_D/D}{d} \Big/ \binom{I_D}{d}\right\}$$

Where the terms in parenthesis are the binomial coefficients. The above expression reduces to

$$K = \left(1 - \prod_{j=0}^{(I_D/D)-1} (I_D - d - j)/(I_D - j) \right) D$$

Granule Size : The selection of granule size for a database system is a trade off issue. There are two extreme cases. The first case is, each data time is a granule (lock). This corresponds to finest granularity. The second is, the entire database is considered as one data granule. In the first case, the transaction do not conflict with each other unless they need to access the same data item(s). But the overhead involved in locking, and releasing the locks are high. In the second case, the transaction need to access only one lock, which keeps the locking overhead to minimum. But, once a transaction has locked the database then all other transactions conflict, which reduces the throughput of the system.

The above two cases form the two extremes and in between them there are number of possibilities. Infact the number of granules can vary from 1 to I_D. If the granules are coarse i.e. the granules contain large number of data items, then two transactions may conflict with each other even if they are accessing different data items because the data items may be in a common granule. Thus, reducing the concurrency factor (number of transactions concurrently executing in the system). But, if the granularity is fine, then it seems that transactions is less likely to conflict with other transactions, but the overhead to lock the granules increases. The problem is to choose the granularity such that the system performance are optimal, i.e. the response times are minimum, throughput is maximum etc.

Let, t_{pl} = mean precessor service time to either lock or unlock a granule

t_p= mean processor service time to process a data item

t_c= mean channel service time to transmit a data item

Then, the service rate terms in equation 17 and 18 in section 4.2 are given by

$\mu_p = 1 / (t_p d + t_{pl}\, k)$ and $\mu_c = 1/ (t_c d)$

Where d is the number of data items requested by the transactions which are contained in k granules.

In order to find the optimal granularity D the number of locks is varied from 1 to I_D, the number of data items in the database. The probability of conflict expressions in the model assume that $D >> N_L$ and also $D >> K$. These assumptions do not especially if D is very small. For this case we derive an alternate expression and it is given below. Let P(j) be the steady state probability that i transactions are holding locks at a site and $P_c(j)$ be the probability of conflict when it sees j transactions upon its arrival. Then, the probability that a transaction conflicts P_c is :

$$P_C = \sum_{j=0}^{\infty} p_c(j)\, p(j)$$

Now, the probability that a transaction conflicts when it sees transactions in the system is :

$$P_c(j) = 1 - \frac{\binom{D-kj}{k}}{\binom{D}{k}} \quad \text{for} \quad j \le D/k$$

$$\text{and} = 1 \quad \text{for} \quad j \ge D/k$$

There are only d locks. There fore the number of transactions holding locks is less tha or equal to D/K. Hence P(j)=0 for j >D/K and the probability of conflict is :

$$P_C = \sum_{j=1}^{\lfloor D/k \rfloor} p_c(j)\, p(j)$$

The only terms not known in the above expression are the P(j). The exact expressions for these terms are difficult, if not impossible to compute. We will derive an approximation by using the mean number of locks in the system. The average number of locks held in a database (N) is given by

$$N_L = \sum_{j=0}^{\lfloor D/k \rfloor} k(j)\, p(j)$$

And the probability of conflict is

$$P_c = \left(1 - \frac{\binom{D-k}{k}}{\binom{D}{k}} \right) N_L/k$$

Results : The relative throughput is minimum when the number of granules equals 1. As the granularity is decreased the performance improves and the maximum occurs when the number of granules is equal to the size of the database. When the granularity is very coarse then the number of transactions concurrently executing is limited by the number of granules. As the number of granules increases more and more transactions can concurrently execute. This improvement is far greater than the locking overhead. The response times for the successful transactions increases as D increases but the increase is small. The optimal granularity depends on the transaction size. If the transactions are small then fine granularity gives the optimal performance. The optimal performance for large transactions is when the entire database is one single lock. Thus for larger transactions coarse granularity is better and that for smaller transactions fine granularity is better.

6.4 SUMMARY

In this chapter, parametric analysis was carried out. From this we draw the following conclusions.

For Fully replicated databases:

1. The number of sites has very little effect on the throughput. However, the response time vary significantly with the number of sites.

2. Smaller transactions are preferred over larger transactions.

3. The read locks improve performance and for large communication delay the performance is less sensitive to processor delays.

4. Fine granularity is preferred for small transactions and coarse granularity is preferred for the large transactions.

CHAPTER 7

7.0 SUMMARY AND CONCLUSION :

Section 7.1 summarizes the main points of this dissertation. The conclusions are given in section 7.2. Finally, future research directions are suggested in section 7.3.

7.1 SUMMARY

In order to obtain the performance of various concurrency control algorithms for distributed database systems, results obtained by previous researchers were studied and analyzed. These included work done by number of researchers, among whom Ries, Pun and Belford, Len and Notte, Singhal and Tay etal are on the forefront. The models were based on mean value analysis for data contention and queuing network models for resource contention. The database state was represented by the average number of locks locked. The state that a transaction sees at its local site and at the external sites are different and this was taken into account while developing the model. The results of parametric analysis which included parameters such as network size, the number of locks, probability of readlocks, and degree of locality of reference on the performance were presented. Also, the problem of optimal granularity was addressed.

7.2 CONCLUSIONS

1. The model developed in this thesis can be used to study the performance of locking algorithms in distributed database systems. The model will be in good agreement with simulation results.

2. The model developed is flexible enough to incorporate different locking schemes .

3. The problem of optimal granularity was discussed and it was observed that fine granularity is preferred for small transactions and the coarse granularity is preferred for large transactions.

4. Small transactions are preferred over large transactions

5. Read locks decrease the probability of conflict and under certain restrictions this model is equivalent to uniform access with database size increased.

6. Non-uniform access increases the probability of conflicts although equations for uniform cases are given in this work which can be extended for non-uniform access and under certain restrictions this model is equivalent to uniform access with the database size reduced.

7.3 FURTHER RESEARCH

There are number of areas in which this work can be extended. A few areas are discussed below.

1. This model considers the no waiting case. The model can be extended to include the waiting case by deriving an expression for the transaction wait time when they conflict

2. The basic algorithms studied in this thesis was locking. There are other algorithms based on Time stamped algorithms and Optimistic Control Algorithms. Extension of the model to these algorithms would give a unified way of comparing the various algorithms.

3. The processing elements were combined and modeled as single servers. The model can be extended to include a detailed model of the CPU, Disk I/O. Also, the service times for different stages of the transaction were assumed to be same. This assumption can be relaxed using some approximation techniques.

4. The current model uses exponential service time distributions. It can be extended to consider general service distributions.

5. This thesis dealt with fully replicated database. The model can be extended for partitioned databases and partially replicated databases

6. In this work the network was considered to be fully connected. The model can be extended to other network topologies like hypercube, star, mesh, ring etc.

REFERENCES

(1) Kennet C Sevcik and Hai Wang, “Solution Properties and convergence of approximate Mean value analysis algorithm”, A.C. Sigmetrics:March 2002

(2) Wang, H. and Sevcik, K. C., “Experiments with improved approximate mean value analysis algorithms”, Performance Evaluation, vol. 39, no. 1-4 pp. 189 – 206, 2000.

(3) Ross, S.M.,”Introduction to probability models”, Seventh edition, 2001

(4) Desia, B.C., “An Introduction to Database Systems”, Galgotia Publications Pvt. Ltd.:1999

(5) Silberschatz, A.,Korth, H.F., Sudarshan, S. “Database System Concepts”, Third edition, T.M.H.,Inc.1997

(6) Agarwal, R. and Carey, M.J.”Models for studying Concurrency Control Performance Alternatives and Implications” ACM SIGMOD, 1985.

(7) (6)Tay, Y.C, Suri,R. and Goodman,N.,”A Mean value Performance model for locking in databases:The No-waiting”, Journal of ACM,Vol.32, July 1985.