N91-23974

Performance Analysis of Static Locking in Replicated Distributed Database Systems

Yinghong Kuang
Ravi Mukkamala
Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529.

Abstract

Data replications and transaction deadlocks can severely affect the performance of distributed database systems. Many current evaluation techniques ignore these aspects, because it is difficult to evaluate through analysis and time-consuming to evaluate through simulation. In this paper, we use a technique that combines simulation and analysis to closely illustrate the impact of deadlock and evaluate performance of replicated distributed database with both shared and exclusive locks.

1. Introduction.

A distributed database system (DDS) is a collection of cooperating nodes each containing a set of data items. A user transaction can enter such a system at any of these nodes. The receiving node, often referred to as the *coordinating* node, undertakes the task of locating the nodes that contain the data items required by a transaction.

In order to maintain database consistency and correctness in the presence of concurrent transactions, several concurrency control protocols have been proposed [1]. Of these, the most commonly used are time-stamping and locking protocols. Locking protocols have been widely used in both commercial and research environments. In static locking, prior to start of execution, a transaction needs to acquire either a shared-lock (for read operations) or an exclusive lock (for update operations) on each of the relevant data items.

Data replication is used to improve the performance of local transactions and the availability of databases. In replicated databases, one data item may have more than one copy in the system. Replica control algorithms are used to maintain the consistency among these copies. One of these is the read-one/write-all protocol. With this protocol an exclusive lock need to acquire an exclusive lock from every copy of the data item . For a shared lock to succeed, any one copy of the data item has to be share locked. When transactions with conflicting lock requests are initiated concurrently, they could be possibly blocked due to a deadlock.

There are two major ways to evaluate the performance of distributed systems: simulation and analysis. Simulation is a conceptually tractable technique, but requires large computation time. On the other hand, analysis is computationally faster but may not be tractable for all problems. In [4], Shyu and Li proposed an elegant analysis model to evaluate the response time and throughput of transactions in a non-replicated DDS. Assuming *exclusive* locking (i.e., only write operations), they model the queue of lock requests at an object as an M/M/1 queue [3]. This results in a closed-form for the waiting time distribution at a node, expressed in terms of the average rates of arrivals of requests and the average lock-holding time. With shared lock and replications added into the picture, it is very difficult to have a close model for it. Because of the limitations of simulation and analysis, we develop a technique that combines simulation and analysis.

This paper is organized as the follows. In Section 2, we describe the model used in our performance evaluation. In Section 3, we propose an evaluation technique. In Section 4, we illustrate the results. Finally, Section 5 has the conclusions.

2. Model

Our model has the following parameters:

- There are n nodes.
- There are d data items in a DDS.
- A data item may be located at exactly c number of nodes. The dc data copies are uniformly distributed across the n nodes.
- Each transaction accesses k data items.
- r is the read ratio. So among k data items to be accessed, rk are accessed only for read operations, and the rest are for read-write operations. Due to the read-one/write-all replica control policy, a transaction must procure rk shared locks for rk read operations and $(1-r)kc$ exclusive locks for the $(1-r)k$ read-write operations.
- Each data item is equally likely to be accessed by a transaction.
- Transaction arrivals into the system is a Poisson process with rate λ.
- The communication delay between any two nodes is exponentially distributed with mean $\bar{t}$.
- The average execution time of a transaction, once the locks are obtained, is $\bar{s}$.
- The deadlock mechanism is invoked every τ seconds.
- After an abortion of a transaction, it takes an average of ω seconds for this transaction to be restarted.
- μ is the service rate of transactions.
- b is the lock-holding time.
- λc is the arrival rate at each data copy.

[1]This research was supported in part by the NASA Langley Research Center under contracts NAG-1-1114 and NAG-1-1154.

CH2998-3/91/0000-0698$01.00©1991IEEE

3. Performance Evaluation Technique

Our technique consists of two stages. In the first stage, the average transaction response time and throughput are calculated by ignoring the deadlock. This is an iterative step involving simulation and analysis. In the second stage, the probabilities of transaction conflicts and deadlocks are computed by probability models. These probabilities are used, in turn, to compute the response time and throughput in the presence of deadlocks.

Stage 1:
Initially, we assume that there are no lock conflicts between transactions. Each transaction has to procure rk shared lock on data copies and $(1-r)kc$ exclusive locks on data copies. When a transaction has got all the lock grants from these data objects, it can go ahead with execution.

This procedure is summarized in the following 6 steps.

1. Initialize lock-holding time(b) to be $1/\mu$.
2. Given the total rate of transaction arrival(λ), the shared lock ratio(r), the number of data items(d), the number of data items required by each transaction(k) and the number of replications(c), derive the arrival rate at each data copy(λc).
3. With the arrival rate at each data copy(λc), the average lock-holding time(b), and the transmission time(t) we can simulate the queue at a data copy to arrive wait-time(w) distribution. With this distribution we can calculate the response time of transactions.
4. With the average service time of transactions($1/\mu$), and the transmission time, we can derive a new lock-holding time(b').
5. Set b to this new lock-holding time b'.
6. If the old and new lock-holding time are sufficiently close, stop the iteration. Otherwise, go back to step 3.

At the end of stage 1 the response time without the consideration of transaction deadlocks is obtained.

Stage 2:
This stage considers transaction conflicts and computes the deadlock probability. Here the probabilities of transaction deadlock and restart are computed. These are then used to compute response time and throughput in the presence of deadlocks.

Assume there are two transactions T1 and T2. Let RS, WS be the read and write sets of transactions respectively.

1. Let fs_i be the probability that the readset of T1 has i data items overlapping with the writeset of T2, i.e. $|RS(T1) \cap WS(T2)| = i$.
2. Let fe_{ij} be the probability that given $|RS(T1) \cap WS(T2)| = i$, the writeset of T1 has j data items overlaping with the readset and writeset of T2, i.e. the probability that $|WS(T1) \cap (RS(T2) \cup WS(T2))| = j$.

Clearly,

$$fs_i = \frac{\binom{k-rk}{i}\binom{d-k+rk}{rk-i}}{\binom{d}{rk}} \quad (1)$$

$$fe_{ij} = \frac{\binom{k-i}{j}\binom{d-rk-k+i}{k-rk-j}}{\binom{d-rk}{k-rk}} \quad (2)$$

It can also be noted that $fs_i fe_{ij}$ is the probability that:
$|\text{Read-set(T1)} \cap \text{Write-set(T2)}| = i$
$\wedge\ |\text{Write-set(T1)} \cap (\text{Write-set(T2)} \cup \text{Read-set(T2)})| = j$.
If PW_{ij} is the probability that T1 waits for T2,

$$PW_{ij} = p1 + p2 - p1 * p2 \quad (3)$$

$$p1 = 1 - [1 - (1/2)^c]^i \quad (4)$$

$$p2 = (1 - (1/2)^{cj}) \quad (5)$$

where $p1$ is the probability that T1 waits for T2 for shared locks in readset
and $p2$ is the probability that T1 waits for T2 for exclusive locks in writeset.
Probability that T1 waits for T2 is now given by

$$Pw = \sum_{i=0}^{\min(x,k-x)} \sum_{j=0}^{\min(k-x,k-i)} fs_i fe_{ij} PW_{ij} \quad (6)$$

With this probability of waiting and the formulas in [4] we can calculate the probability of a transaction deadlock, the probability of a transaction restart and the probability of a transaction to be blocked by other transactions. And with these probabilities and the time between deadlock detection(τ), we can calculate the response time with consideration of deadlock. (Details are ometted here.)

4. Results

Using this technique, we obtained a number of interesting results that illustrate the effect of deadlocks and number of replications on database performance. These are summarized in Figures 1-5. We make the following observations.

- Transaction response times are quite sensitive to the ratio of shared locks (Figure 1 and 2). Here, we compare the response times when deadlocks are ignored (DI, computed in Stage 1) with those obtained when deadlocks are considered (DC, computed in Stage 2). The effect of deadlocks is more predominant at higher transaction loads and with smaller values of r. When $r = 2/3$, the effect of deadlocks is not significant on response time.
- If we compare Figure 1 and 2 with Figure 3 and 4, it can be observed that the increase in replications results in the larger response time when read ratio is smaller than 1/3.
- Fig. 5 shows the response times with different replication numbers. Here we can see that with both cases when read ratio is 2/3 and 1/3, the response time increases as the number of replications increases. But with read ratio equals 1/3, the increasing rate is much smaller than that with read ratio equals 2/3.

5. Conclusions

In [4], Shyu and Li presented an elegant technique to evaluate the performance of distributed database systems in the presence of deadlocks. Their technique assumed only exclusive locks and thus representing the worst-case effects of deadlocks.

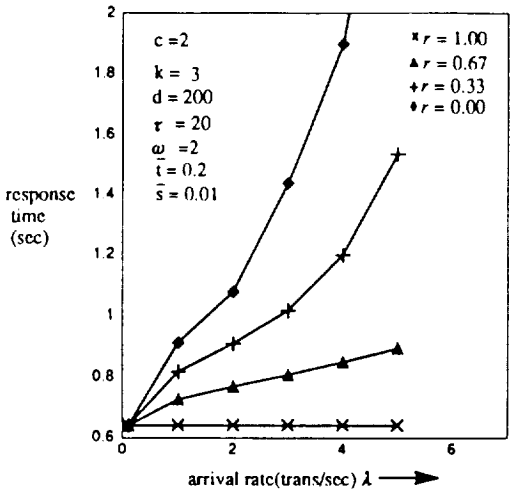

Figure.1 Comparison of response time with different read ratio when deadlock is ignored.

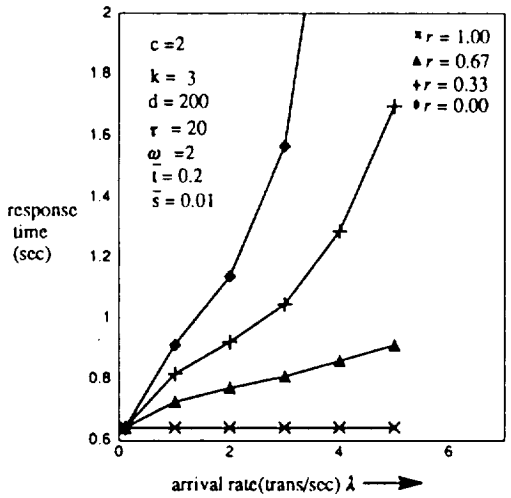

Figure.2 Comparison of response time with different read ratio when deadlock is considered.

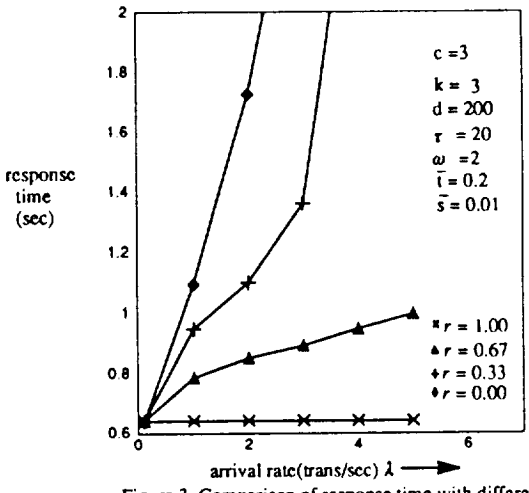

Figure.3 Comparison of response time with different read ratio when deadlock is ignored.

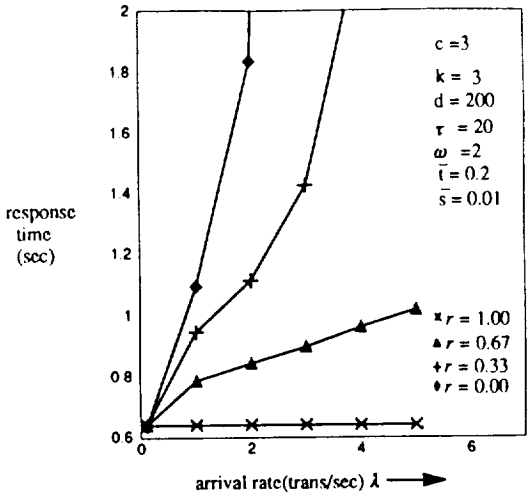

Figure.4 Comparison of response time with different read ratio when deadlock is ignored.

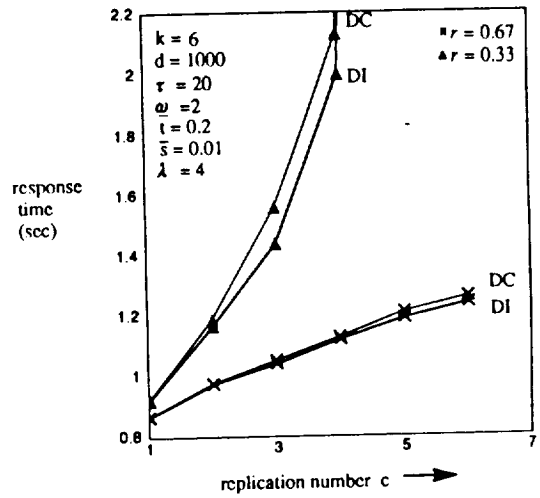

Figure. 5 Comparison of response time with different read ratio with and without deadlock.

DC: Deadlock Considered.
DI: Deadlock Ignored.

In this paper, we have extended their technique to combine sim ulation and analysis. And with this extended technique we allo both shared and exclusive locking and also replications in ou model. We evaluated the the effect of number of data items, th number of data items accessed by each transaction, the ratio o read operations on transaction response time and the number o replications. These results show the importance of considerin both shared and exclusive lock requests, the deadlock proba bilities as well as the number of replications of database fo response time evaluations.

References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[2] A. Hac, "A decomposition solution to a queueing network model of a distributed file system with dynamic locking," *IEEE Trans. Software Eng.*, vol. SE-12, no. 4, pp. 521-530, Apr. 1986.

[3] L. Kleinrock, *Queueing Systems, Vol. I*, New York: Wiley-Interscience, 1975.

[4] S.-C. Shyu and V. O. K. Li, "Performance analysis of static locking in distributed database systems," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 741-751, June 1990.

[5] M. Singhal and A. K. Agrawala, "Performance analysis of an algorithm for concurrency control in replicated database systems," *Proc. ACM SIGMETRICS Conf. Measurement Modeling Comput. Syst.*, 1986, pp. 159-169.

[6] Y. C. Tay, R. Suri, and N. Goodman, "A mean value performance model for locking in databases: The no-waiting case," *J. ACM*, vol. 32, no. 3, pp. 618-651, July 1985.