THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY[1]

APPROXIMATING THE PERFORMANCE OF TWO PHASE LOCKING USING AN ITERATION SOLUTION MODEL

David D. Chen and Toby J. Teorey

CRL-TR-25-84

APRIL 1984

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

[1]Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agency.

ergn

UMR1184

ABSTRACT

APPROXIMATING THE PERFORMANCE OF TWO PHASE LOCKING

USING AN ITERATION SOLUTION MODEL

By

David D. Chen and Toby J. Teorey

One of the major objectives of a distributed database system is data shareability. However, in order to maintain consistency in the database, potentially expensive synchronization algorithms are required. A simulation model and an analytical approximation model are used to analyze the performance of the two phase locking protocol, one of the most practical and widely used concurrency control mechanisms. Performance is measured in terms of transaction response time. When results from these models are compared, the differences are acceptably small which shows the analytical (iteration solution) model to be a practical tool in evaluating system performance. It is shown that the ratio, not the individual values, of database size and the transaction arrival rate has the most significant effect on performance.

APPROXIMATING THE PERFORMANCE OF TWO PHASE LOCKING USING AN ITERATION SOLUTION MODEL

By

David D. Chen and Toby J. Teorey

I. INTRODUCTION

The use of database systems for managing large amounts of information has been widely recognized, and the implementation of these systems has been and will be a major activity. In the past, much effort has been devoted to inventing algorithms to implement and forming design tools for these database systems. However, little work has been done on the analysis of these algorithms and tools.

One of the most important advantages of using database system is the shareability of data. However, in order to maintain the consistency of the database, the user requests of the data must be synchronized into ordered sequences by a concurrency control mechanism. Thus a measurable system overhead is often introduced. In distributed database systems, synchronizing data access at different nodes can seriously degrade the system performance. In order to develop high performance concurrency control algorithms, a mathematical predictive model can be highly useful.

Completed: 03/22/84

We apply a mathematical model to study the performance of the two phase locking protocol, one of the most commonly used synchronization methods in distributed database systems [ESWA76, STON79].

Two phase locking (2PL) synchronizes read and write operations by explicitly detecting and preventing conflicts between concurrent operations. A transaction may lock data items to ensure their inaccessibility while in a temporarily inconsistent state. In the simplest case each data item has a unique lock which is held by at most one transaction at a time. If a transaction attempts to lock a data item that is already locked, it must either wait, abort itself, or preempt the earlier transaction. In general, two types of locks are issued by the data manager. That is, before reading data item *a*, a transaction must own a readlock on *a*. Before writing into *a*, it must own a writelock on *a*. The ownership of a lock is governed by the following rules [BERN80]:

1. Different transactions cannot simultaneously own conflicting locks. In the exclusive read case, two reads form the conflicting accesses; in the shared read case, two read operations do not conflict with each other. Thus, the definition of conflicting locks depends on the system discipline being used.

2. Once a transaction surrenders ownership of a lock, it may never obtain additional locks. Thus, every transaction obtains locks in a *two phase* manner. During the *growing phase* the transaction obtains locks without releasing any locks. By releasing a lock the transaction

enters the *shrinking phase*. During this phase the transaction releases locks and is prohibited from obtaining additional locks.

Furthermore, in two phase locking the execution order of transactions is set by the data manager based on the order of arrival of data access requests. Usually deadlocks are detected and resolved by the data manager. A simpler case, in which no distinction is made between readlock and writelock, will be analyzed. In Section II, the system specification is presented. A simulation model is then developed and simulation results are discussed in Section III. The iteration solution model which approximates the response time of transaction is given in Section IV. Comparisons between simulation models and the iteration solution model are made in Section V. The granularity property for two phase locking protocol is discussed in Section VI. Finally, in Section VII future directions are stated.

II. SYSTEM SPECIFICATION

Given a database system, the size of the database, denoted by *DZ*, is the total number of data items which may be a field, a record, a relation, or a file. The granularity of data item will be unspecified in this study. Each transaction requests to access different data items. The number of the requests is called the size of the transaction and is denoted by *TZ*. The retrieval time on a data item, say *a*, of a transaction is the service time *S(a)*. Before a transaction accesses a data item, it first requests the access permission from the data manager. The data

manager will grant the permission if no transaction is currently accessing the data item, and put a "lock" on the data item to prevent other transaction from accessing it.

Assume that the data items are requested on demand. When a transaction, say A, requests a data item whose lock has been held by another transaction, a conflict occurs and transaction A is called the *conflicting transaction*. The conflicting transaction has to wait in a first come first served fashion. When two transactions wait for each other, directly or indirectly, the deadlock occurs. The conflicting transaction which causes deadlock has to rollback to its original state, release all locks held, and restart its processing. It is assumed that a transaction will hold all its locks until the end of its session. Thus in the no deadlock case, the *lock holding time* $T(a,i)$ of the ith access request which accesses data item a of a transaction is

$$T(a,i) = S(a) + \sum_{j=i+1}^{TZ} [W(j) + S(j)], \qquad (1)$$

where $W(j)$ and $S(j)$ are the waiting timing and the service time of data item j. The timing diagram for a transaction is shown in Figure 1. The response time R is

$$R = \sum_{j=1}^{TZ} [W(j) + S(j)]. \qquad (2)$$

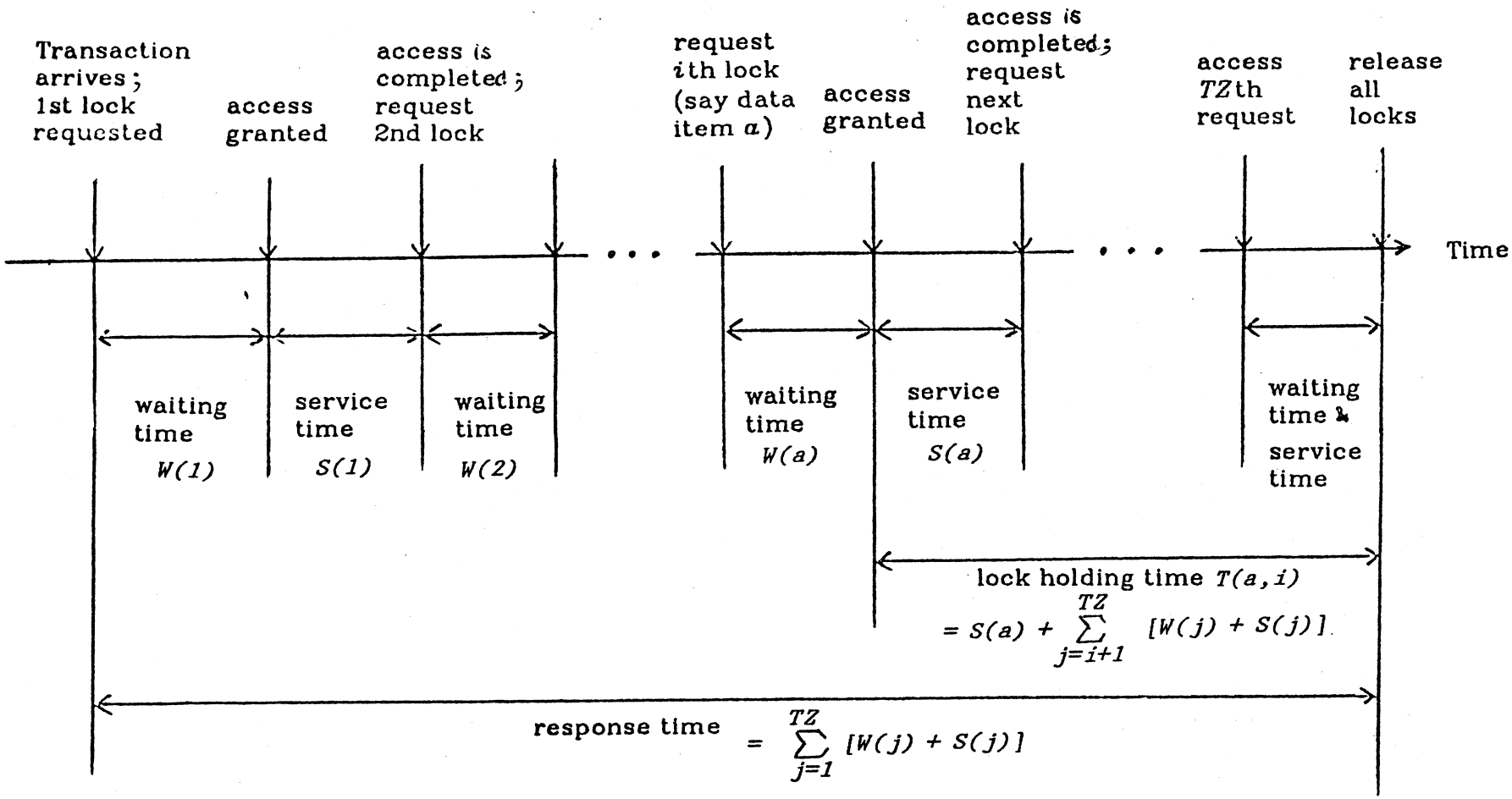


Figure 1. The time diagram for a transaction in the two phase locking.

III. THE SIMULATION MODEL

The system of two phase locking was proven to be mathematically intractable [LIN79]; thus without simplifying the problem, the simulation model is currently the only method available. A simulation model is developed in this section. It is assumed that *TZ* is fixed during the simulation. No two access requests in a transaction are requesting the same data item. Deadlocks are detected by analyzing the wait-for graph [BERN80] at the moment the access is received by the data manager. When deadlock occurs, the transaction which requests the lock and causes deadlock is aborted. Aborted transactions will request the same data items in the same sequence of lock requests as the transaction that was first generated. Figure 2 shows the flow of control of a transaction in the simulation model.

The simulation program was written in the C language and run on a VAX-11/780 machine. The service time *S(a)* is set to be one unit of time, and each simulation generated more than 1500 operation requests. Figure 3 lists the result of the simulated response time, the probability that a conflict arises when a data item is requested, the probability that a deadlock occurs when a data item is requested, and the conditional probability of a deadlock occurring given there is a conflict. Figure 4 gives the relationship between the response time and transaction arrival rate for several cases.

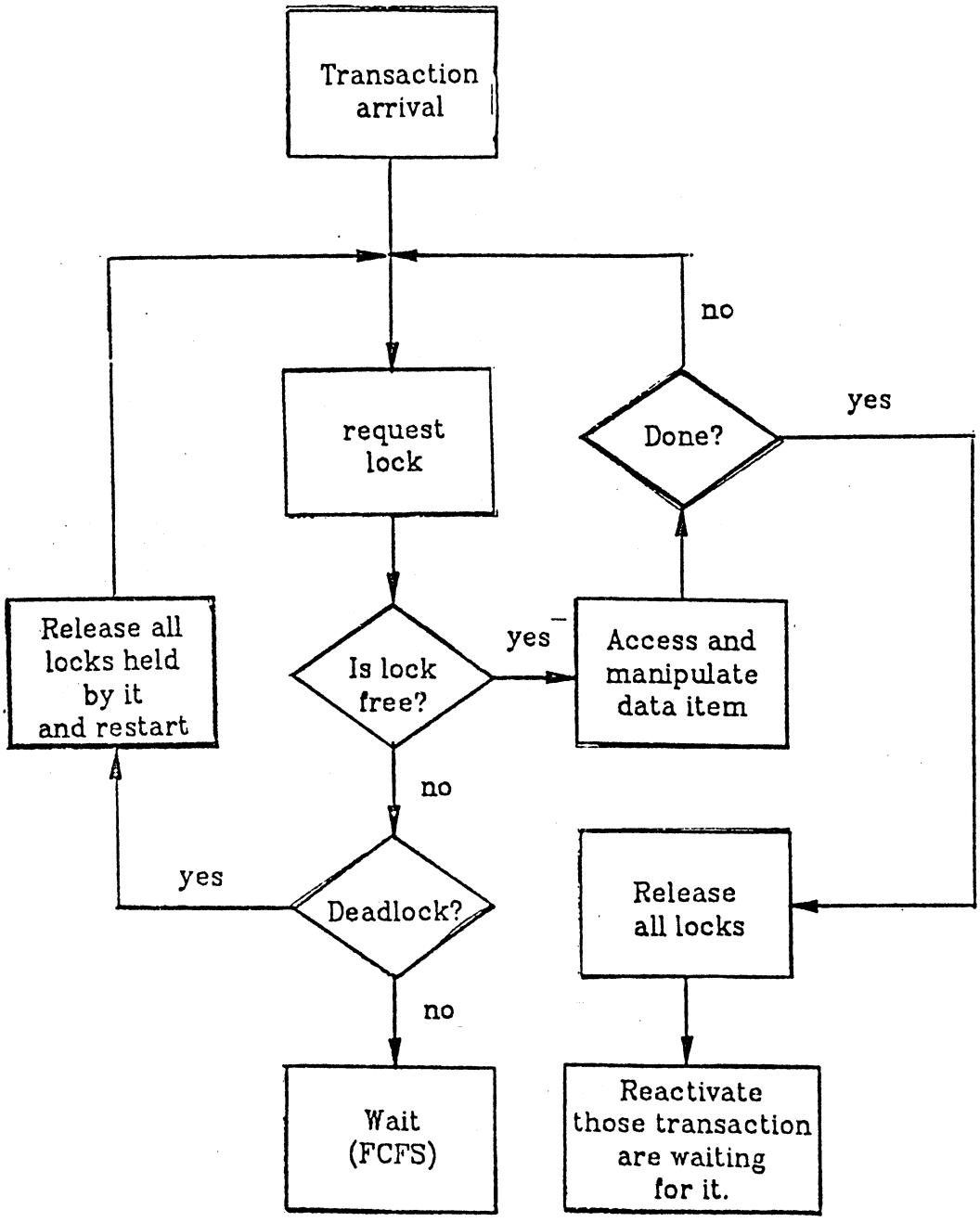


Figure 2. The flow of control of a transaction in the two phase locking.

TZ	DZ	Transaction arrivals/sec	Simulated resp. time (seconds)	Prob. of Conflict(PC)	Prob. of deadlock(PD)	PD/PC
3	64	0.2	3.0930	0.0259	0.00	0.00
3	64	0.4	3.1298	0.0373	0.0007	0.018
3	64	0.6	3.2146	0.06	0.0004	0.007
3	32	0.2	3.1208	0.031	0.00086	0.028
3	32	0.4	3.3126	0.0768	0.00089	0.0116
3	32	0.6	3.5116	0.1134	0.0022	0.0195
4	64	0.2	4.2181	0.032	0.0006	0.019
4	64	0.4	4.6541	0.080	0.0037	0.0466
4	64	0.6	5.0230	0.113	0.0029	0.0256
4	32	0.2	4.3783	0.0588	0.005	0.085
4	32	0.4	5.0217	0.129	0.007	0.0573
4	32	0.5	5.4760	0.176	0.012	0.066
4	32	0.6	-	-	-	-
5	64	0.2	5.5217	0.0507	0.0012	0.023
5	64	0.4	6.4421	0.1107	0.0071	0.0643
5	64	0.6	-	-	-	-
8	256	0.2	8.6722	0.025	0.0005	0.018
8	256	0.4	10.1687	0.065	0.0018	0.0275
8	256	0.6	13.7433	0.13	0.009	0.07
8	256	0.8	-	-	-	-

Figure 3. Simulation result of 2PL: where TZ is the number of data items requested by a transaction and DZ is the number of data items in the database.

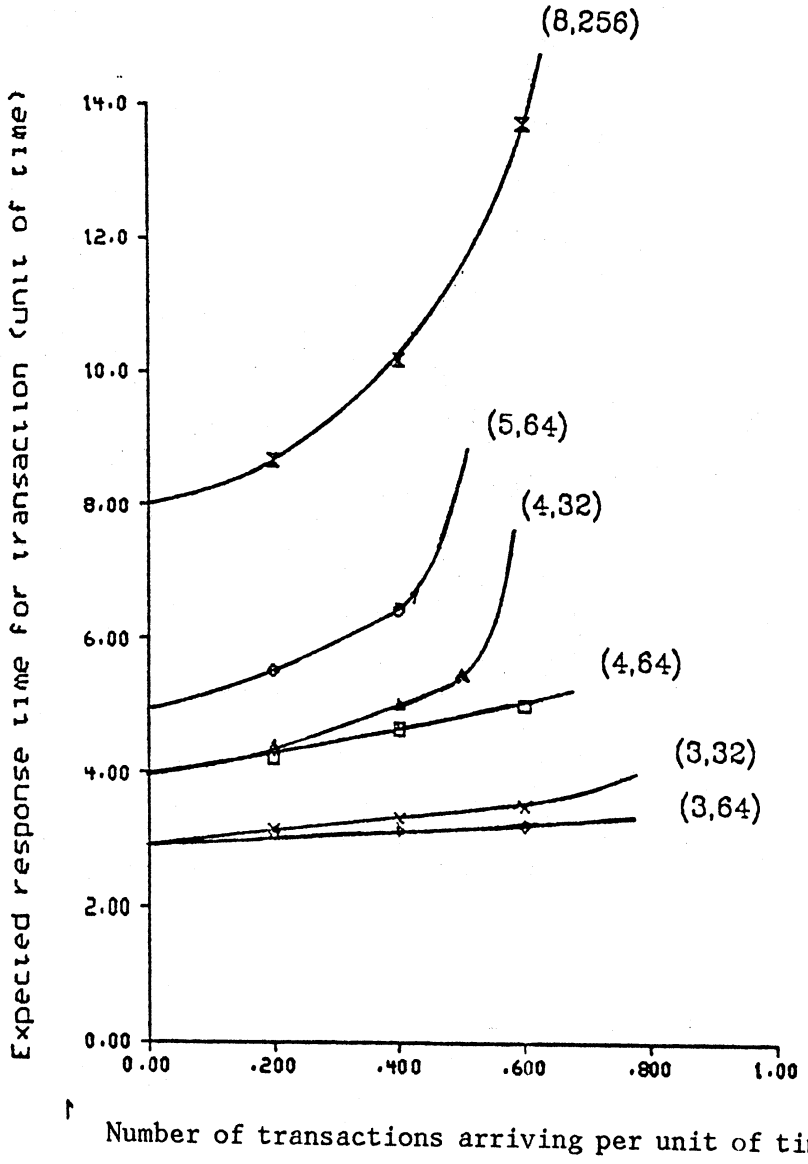


Figure 4. Response time for transaction versus the transaction arrival rate in two phase locking, where (TZ,DZ) is the pair of the number of data items requested by transaction and the number of data items in the database.

Several conclusions can be drawn from these cases:

1. When the system is not saturated, the probability of deadlock is much smaller than the probability of conflict. These results coincide with Lin's data [LIN81].

2. The distance between unsaturated and saturated transaction arrival rates is very narrow. Using the number of requests, $TZ = 4$ and the number of data items in the database, $DZ = 32$ as an example, the response times gradually increase as the arrival rates increase from 0.2 to 0.5 (number of transactions per unit of time). However, when the arrival rate increases to a little higher than 0.5 the system becomes saturated. The phenomenon suggests that the system should not be designed near the saturation point.

3. Because the data items requested by an aborted transaction are not regenerated in the simulation model, the aborted transaction tends to be aborted several times in a heavily utilized system. The system degrades rapidly as the system utilization factor increases.

IV. ITERATION SOLUTION MODEL

An iteration solution model is now developed for two phase locking, which simplifies the system in many ways and provides results reasonably close to the more expensive simulation model. Let us first focus our

attention on transactions which access a data item called *a*. Data item *a* may appear in the *i*th lock request, where i=1,...,*TZ*. Let *T(a,i,m-1)* be the *lock holding time* of data item *a* derived at *m-1*st iteration for a transaction that requests data item *a* at *i*th position. The lock holding time for a data item is the time period from the time the lock of that data item is granted to the time the lock is released. Also let *W(j,m)* denote the waiting time for accessing data item *j*. Thus, we have the following equations.

$$T(a,1,m-1)=S(a)+\sum_{j=2}^{TZ}[S(j)+W(j,m-1)] \qquad (3)$$

$$T(a,i,m-1)=S(a)+\sum_{j=i+1}^{TZ}[S(j)+W(j,m-1)] \qquad (4)$$

$$T(a,TZ,m-1)=S(a) \qquad (5)$$

Assuming that every transaction which has data item *a* as one of the requested data items is a Poisson arrival process, we have the arrival rate for each data item as $\lambda TZ/DZ$. Thus for each request position of each data item, the arrival rate is λ/DZ. Since it is difficult to solve these random variable functions in closed form, an iteration method is developed. The iteration is as follows:

1. For the first iteration, let the lock holding time of data item *a* contain only the service times of the data items requested after *a*.

That is, $W(j,0) = 0$ for all j. From Eq. (4), the holding time $T(a,i,0)$ can be derived. Then based on $T(a,i,0)$ we can estimate the expected value and second moment of the waiting time $W(j,1)$ for accessing the data item a.

2. We assume the distribution of waiting times of all the data items are the same. Then, with the probability distribution of waiting time $W(j,m-1)$ (actually, we can't obtain the exact distribution except in the form of Laplace transform and the nth moment.), we derive the waiting time $W(a,m)$ for data item a which has these waiting times $W(j,m-1)$ as part of lock holding time $T(a,i,m-1)$.

3. Repeat step 2 until the distribution of each service time $T(a,i,m)$ has converged.

Even with this approach it is still difficult to solve using iteration. Fortunately, we are only interested in the expected waiting time. After a certain number of iterations the following equation should hold.

$$E[W(a,m)] = E[W(a,m-1)] \qquad (6)$$

From M/G/1 queuing theory, we have the following equation for the expected waiting time [KLEI75].

$$E[W(a,m)] = \frac{\sum_{i=1}^{TZ} (\lambda/DZ)\, E[(T(a,i,m-1))^2]}{2 + [1 - \sum_{i=1}^{TZ} (\lambda/DZ)\, E[T(a,i,m-1)]]} \qquad (7)$$

In order to obtain a closed form formula for the expected response time, the additional assumptions are made as follows:

1. All the service times for data item j, $S(j)$'s are distributed the same.

2. The expected waiting times $E[W(j,m-1)]$ for all data items are the same. In an application environment where transactions randomly access the database, each $W(j,m-1)$ should have the same distribution. As shown in [LIN81], a 20/80 application environment is simply a heavier loaded random accessing application environment.

3. All the covariances $E[S(k)*W(j,m-1)]$ are the same. The assumption is somewhat questionable, especially when k and j are equal. However, when the transaction size and the database size are large, then even if k and j are equal, these two random variables would be less correlated. Thus $E[S(k)*W(j,m-1)]$ are made the same.

4. Assume $E[W(j,m-1)*W(k,m-1)]$ and $E[W(j,m-1)^2]$ are relatively small, with respect to $E[S(a)]$ and $E[S(a)^2]$, and can be ignored. This assumption will be valid if the computation converges.

5. Assume $E[S(a)*W(j,m-1)]=E[S(a)]*E[W(j,m-1)]$. Again, if the size of transaction is large enough, the $S(a)$ and $W(j,m-1)$ should be less correlated.

With these assumptions we are ready to derive the expected value for $W(j,m)$. First, we derive the expected value for $T(a,i,m-1)$ from Eq. (4)

$$E[T(a,i,m-1)]=E[S(a)+\sum_{j=i+1}^{TZ}(S(j)+W(j,m-1))] \qquad (8)$$

$$=(TZ-i+1)*E[S(a)] + (TZ-i)*E[W(j,m-1)]$$

We also can derive the second moment for $T(a,i,m-1)$.

$$E[T(a,i,m-1)^2]=(TZ-i+1)*E[S(a)^2]+(TZ-i+1)*(TZ-i)*(E[S(a)])^2$$

$$+ 2*(TZ-i+1)*(TZ-i)*E[S(a)]*E[W(j,m-1)] \qquad (9)$$

By substituting Eqs. (6), (8), and (9) in Eq. (7), the following equation is derived.

$$6*TZ*(TZ-1)*(E[W(a,m)])^2$$

$$+(TZ*(TZ+1)*(4*TZ+2)*E[S(a)]-(12*DZ/\lambda))*E[W(a,m)]$$

$$+3*TZ*(TZ+1)*E[(S(a))^2]+2*TZ*(TZ+1)*(TZ-1)*(E[S(a)])^2$$

$$= 0. \qquad (10)$$

And as it is shown in Eq. (10), only the ratio, not the individual values, of database size and transaction arrival rate is important. To estimate the expected waiting time E[*W(a,m)*], we need only to solve the quadratic equation (10).

From Eq. (2), the expected response time *R* for a transaction is

$$R = TZ * (E[S(a)]+E[W(a,m)]). \qquad (11)$$

Figures 5 and 6 show the relationship between waiting time and transaction arrival rate and the relationship between waiting time and transaction size.

V. COMPARISON OF SIMULATION AND ANALYTICAL RESULTS

As a measure of the applicability and usefulness of the iteration model, several comparisons are made between the analytical model and other simulation models. The simulation model developed in here, the simulation model developed by Lin [LIN82], and Ries' simulation model [RIES79] are used.

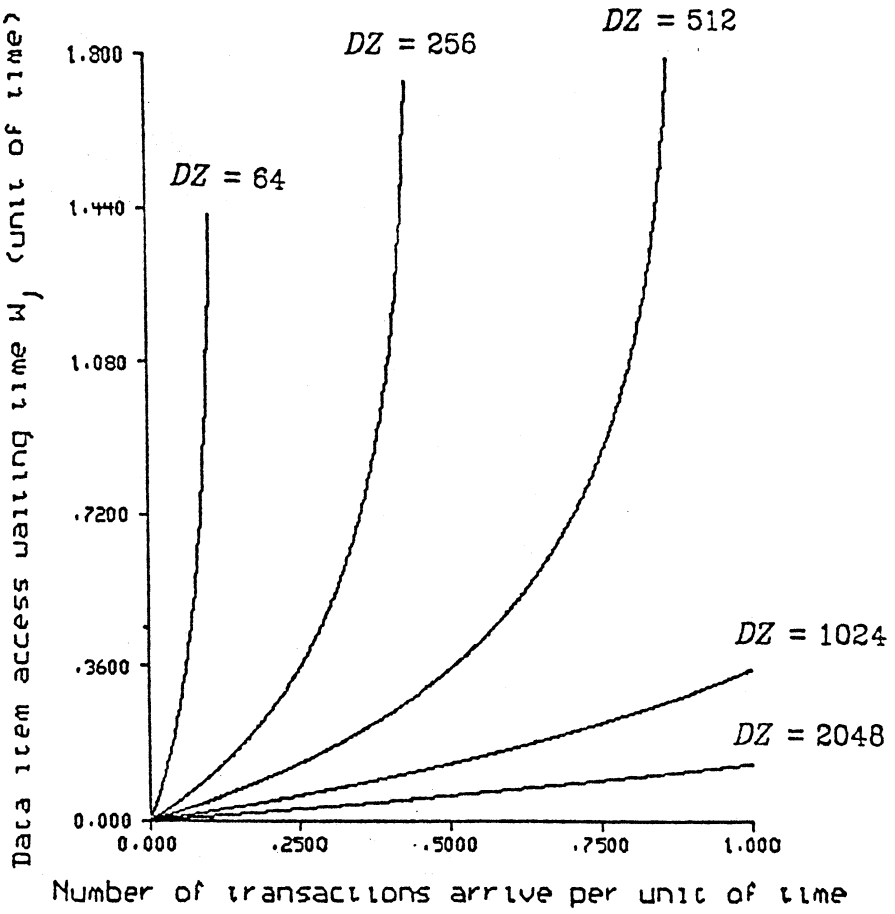


Figure 5. The waiting time versus the transaction arrival rate for the two phase locking where the data item service time is exponentially distributed with mean=1 and the number of data items requested by transaction is equal to 10 (iteration solution model is used).

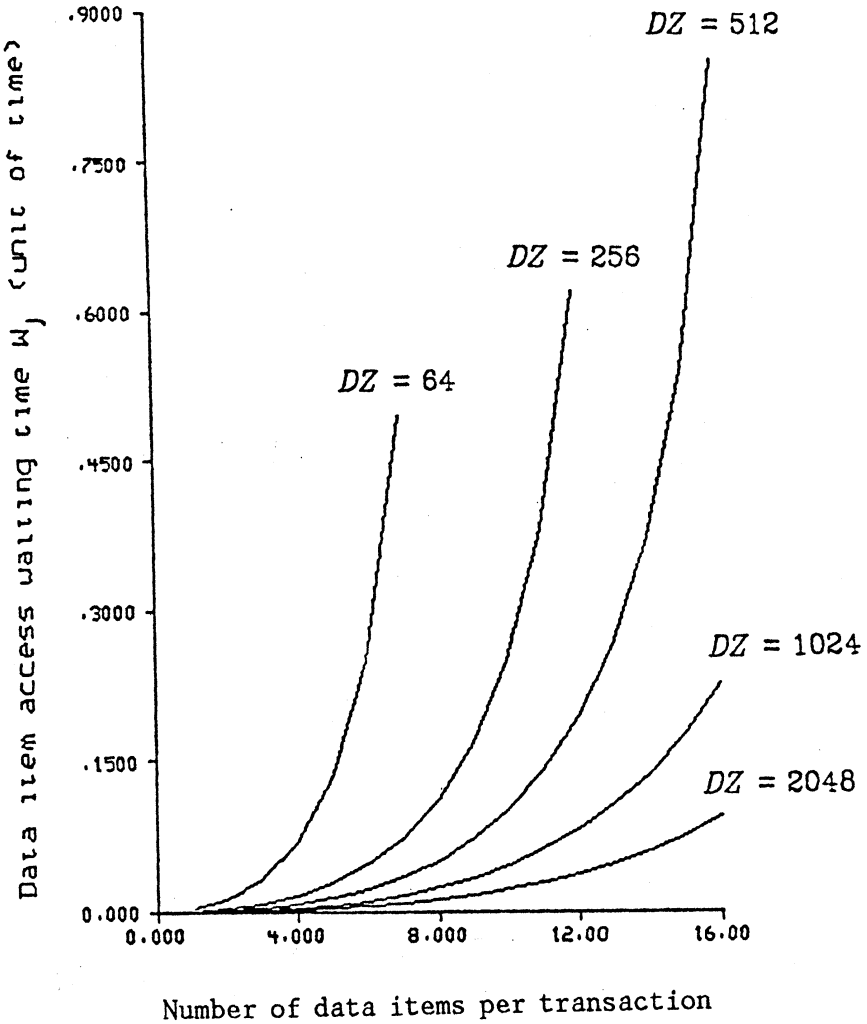


Figure 6. The waiting time versus the number of data items requested by transaction for the two phase locking where the data item service time is exponentially distributed with mean=1 and the transaction arrival rate is equal to 0.2 transaction per unit of time (iteration solution model is used).

Completed: 03/22/84

All four models use the total number of lockable units, which are called data items, to characterize the database size. The transaction size is fixed for all three models, except that Ries assumed that the transaction size is exponentially distributed. Locks are exclusive in all four models. That is, no data item can be shared among transactions. In terms of the sequence of lock requests, all three models assume that transaction requests lock on demand and releases all locks at the end of its session. However, in Ries' model, it is assumed that a transaction can start processing only when it obtains all requested locks and a transaction releases locks after its access is completed. All four models assume that data items have the same probability to be accessed. In addition, Lin also studied the 20/80 application environment, which 20% of data items are accessed by 80% of transactions. In his conclusion, he claimed that the 20/80 application environment is simply a more heavily loaded random access application case.

The data item access time is assumed to be the same for all data items in the iteration solution model, this simulation model, and Lin's model. However, Ries studied the effect of CPU and I/O on locking. Thus, in his simulation model the data item access item is the combination of CPU time, I/O time, and system overhead to lock and unlock.

In Ries' model deadlock will never occur. In the other three models, deadlock is possible. The results of all the simulations show that the probability of deadlock is small if the system is not saturated. Thus, the effect of deadlock is ignored in the iteration solution model.

Both Lin's model and our simulation model abort and restart the conflicting transaction which causes deadlock. However, the sequence of lock requests of the aborted transaction is not regenerated in our simulation, while a new sequence is generated in Lin's model.

One of the major differences among the models is that the iteration solution model and our simulation model use an open queue system in which transaction arrival rate characterizes the system load, while Lin's model and Ries' model use a closed loop system in which the multiprogramming level (*MP*), or fixed number of transactions in the system, is used to represent the activity of the system. The assumption of multiprogramming level is valid in a heavily loaded system, since there are enough transactions to keep up with the multiprogramming level. However, in a lowly utilized system, or in a large multi-application system, the transaction arrival rate is the more appropriate parameter. Figure 7 shows the similarities and differences among the four models.

In the rest of this section, we compare the iteration solution model with our simulation model and Lin's simulation model. Figure 8 lists the results from our simulation model and the results from solving the quadratic equation of Eq. (10). The differences, which are

$$\frac{|\text{Resp time (iter. sol. model)} - \text{Resp time (simul)}|}{\text{Resp time (iter. sol. model)}}, \quad (12)$$

indicate that the iteration solution model estimates the response time

	Iteration solution model	Simulation model	Lin's simulation model [LIN82]	Ries' simulation model [RIES79]
Characterize database (DZ)	fixed	fixed	fixed	fixed
Transaction size (TZ)	fixed	fixed	fixed	exponential distribution
Type of locks	exclusive	exclusive	exclusive	exclusive
When locks requested	request-as-needed	request-as-needed	request-as-needed	obtain all required locks before start
When locks released	at the end of session	at the end of session	at the end of session	when access is completed on that data item
Access distributions	random	random	random 20/80	random
Data item access time S_a	same for all data items	same for all data items	same for all data items	CPU time + I/O time + system locking overhead
Deadlock	not considered	yes	yes	never occur
Deadlock resolution	—	the conflicting transaction is aborted and restarted	the conflicting transaction is aborted and restarted	—
Aborted transaction	—	same lock request sequence	lock request sequence is regenerated.	—
Type of model	open system with transaction arrival rate	open system with transaction arrival rate	closed loop system with fixed multiprogramming level	closed loop system with fixed number of transactions in system
Model level	functional level	functional level	functional level	CPU, I/O level
Measures	waiting time response time	response time prob. of conflict prob. of deadlock	waiting time prob. of conflict prob. of deadlock	useful CPU, I/O lock CPU, I/O response time

Figure 7. Comparisons among two phase locking models.

TZ	DZ	Transaction Arrival rate	Iteration solution model		Simulation	Differences
			Waiting time	Response time R	Response time RS	$\frac{\lvert R-RS \rvert}{R}$
3	64	0.2	0.022881	3.068643	3.0930	0.79%
3	64	0.4	0.047992	3.143976	3.1398	0.133%
3	64	0.6	0.075726	3.227178	3.2146	0.39%
3	32	0.2	0.047992	3.143976	3.1208	0.737%
3	32	0.4	0.106577	3.319731	3.3126	0.2148%
3	32	0.6	0.180450	3.54135	3.5116	0.84%
4	64	0.2	0.051780	4.20712	4.2181	0.26%
4	64	0.4	0.116006	4.464024	4.6541	4.26%
4	64	0.6	0.198744	4.794976	5.0230	4.75%
4	32	0.2	0.116006	4.464024	4.3783	1.9%
4	32	0.4	0.311656	5.246624	5.0217	4.28%
4	32	0.5	0.482211	5.928844	5.4760	7.6%
4	32	0.6	0.812692	7.250768	—	—
5	64	0.2	0.104183	5.520915	5.5217	0.014%
5	64	0.4	0.268785	6.343925	6.4421	1.5%
5	64	0.6	0.602523	8.012615	—	—
8	256	0.2	0.09503	8.760242	8.6722	1.01%
8	256	0.4	0.237568	9.900554	10.1687	2.71%
8	256	0.6	0.488035	11.904277	13.7433	15.4%
8	256	0.8	1.266470	18.131763	—	—

Figure 8. Comparisons between iteration solution model and our simulation model. (Constant service time with $E[S(a)]=1.$)

very accurately in most cases. Several observations are:

1. The last column shows the difference between the two results. The range of the difference is within 10%. When the rate of conflict is low, the difference is less than 1%.

2. When the transaction arrival rate is large, the iteration solution model underestimates the response time. The difference between response times obtained from simulation and those derived from the quadratic equation increases as the transaction arrival rate increases. A major reason is that the theoretical result does not consider the performance degrading effect of deadlock.

3. Deadlocks make the system unable to handle all the transactions while the theoretical result predicts that the system is adequate. This happens when the probability of conflict is larger than 15%. However, as shown in the simulation result, the distance between saturated and unsaturated transaction arrival rates is very narrow. In general, the system should not be too close to the saturation point in a real application case.

In order to compare the results from Lin's model and the results from the iteration solution model, we have to derive the expected response time and the "equivalent" transaction arrival rate from Lin's model. Two statistical data are collected by Lin: the probability of conflict *PC*, and the average waiting time *WT* of a conflicting lock request after the

conflict. Since the response time is the sum of the service time and waiting time, we can derive the expected response time *RS* by the following equation:

$$RS = (S(a) + WT * PC) * TZ \qquad (13)$$

From Little's Theorem [LITT61], the mean number of customers in the system (waiting and in service) is equal to the arrival rate times the mean time for a customer spent in the system. Thus, the transaction arrival rate can be obtained from the following equation:

$$MP = \lambda * RS. \qquad (14)$$

With Eqs. (13) and (14), we are able to compare Lin's simulation with the iteration model. Because the variance between closed loop system (Lin's simulation model) and open system (the iteration model), the differences between these two models vary considerably from case to case. However, in the most cases, the iteration solution model gives good estimation, where the differences are within 10%. More detailed analyses are in [CHEN83].

VI. RESPONSE TIME ANALYSIS

Let us take the example of a centralized database system. Let the total number of records in the database be *DR* and the total number of records accessed by a transaction be *TR*. The basic lockable unit is the

data item which is a collection of records. The number of records in each data item is called the size of the data item. The problem of granularity is to find the best size of data items that minimizes the expected response time for transactions, assuming the two phase locking protocol is used.

Let *DZ* be the number of data items in the database and *TZ* be the number of data items accessed by transactions. Usually three placement cases are studied [RIES79]. Those are the best placement, random placement, and worst placement. In the best placement case, all records retrieved by a transaction are clustered together. As the result of the best placement, the minimum number of data items will be accessed. That is

$$TZ = CEILING(TR * DZ / DR) \qquad (15)$$

In the worst placement case, the records retrieved by a transaction tend to be scattered in different data items. The number of data items accessed by transaction is

$$TZ = mininum(DZ, TZ). \qquad (16)$$

In the random placement case, each record accessed by a transaction has the same probability of being in any data item. From the result of Yao [YAO77], the number of data items accessed by the transaction is

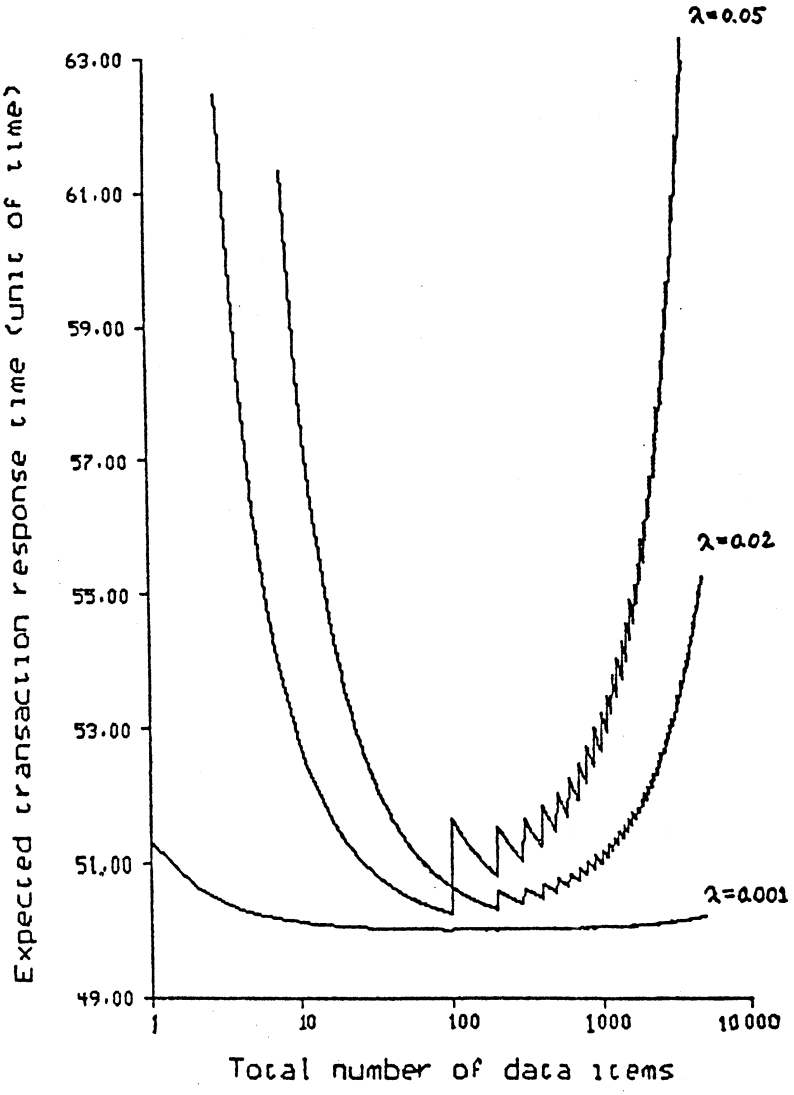
$$TZ = TR * (1 - [C(DR-(DR/DZ),TZ) / C(DR,TR)]) \qquad (17)$$

where the expression of $C(DR-(DR/DZ),TZ)$ and $C(DR,TZ)$ represent the number of different ways TR records can be selected from $DR-(DR/DZ)$ and DR records, respectively.

Let the system we wish to evaluate have 5000 records and each transaction access 50 different records. The processing time for each record is equal to one unit of time. It is also assumed that the system overhead for lock and unlock is small enough to be neglected. Figures 9, 10, and 11 show the expected response times for each transaction versus the number of data items in the system for the best placement, random placement, and worst placement cases, respectively.

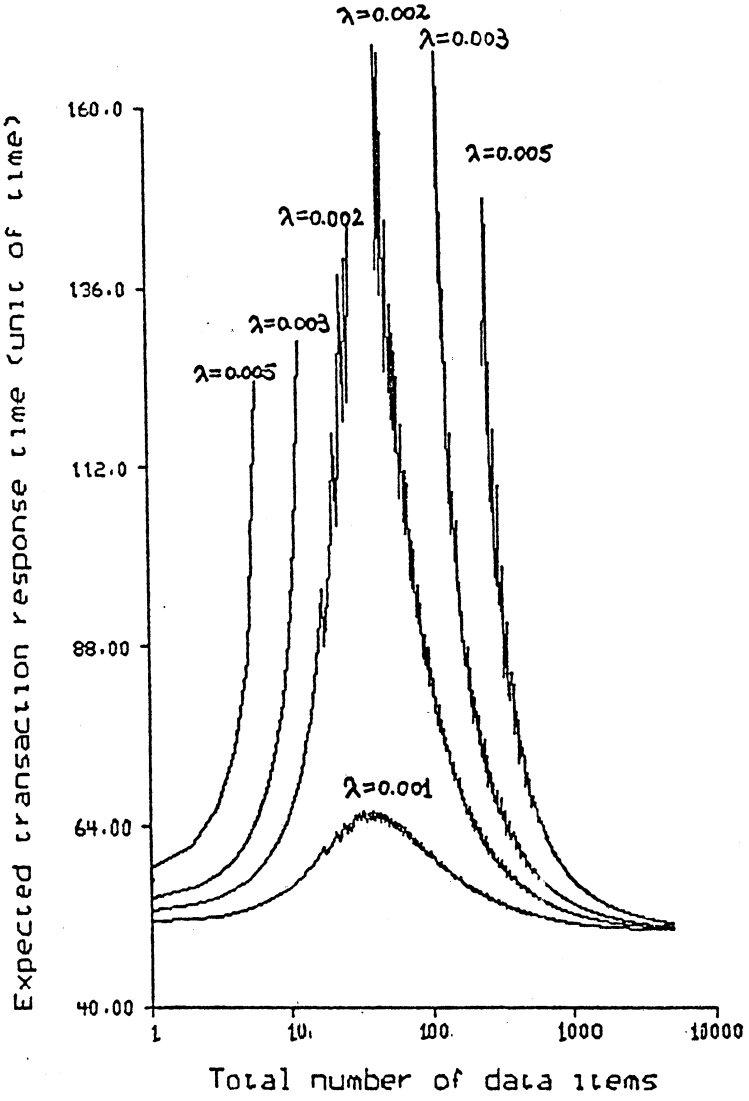
In this example, for both random and worst placement cases, the worst expected response time is near the point that the total number of data items is equal to the number of records accessed by a transaction. Better response times are at both ends of the axis of the number of data item; that is, either fine or coarse data items should be used in these two cases. However, it is believed that when the system overhead is large the larger size of data items should be beneficial.

In the contrary, the best placement case has its best performance at the point where the size of data item (in records) closely approximates the number of records accessed by a transaction; and from this optimal point the performance tends to degrade as the size of the data item



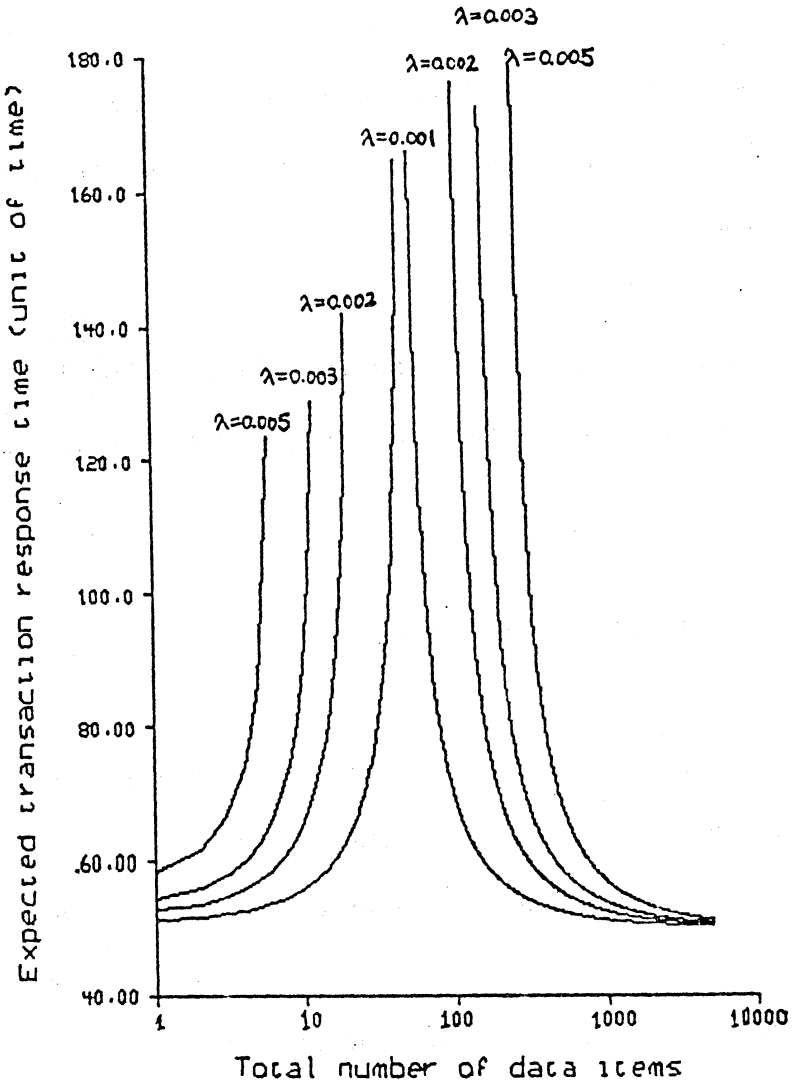
Total number of records in database = 5000
Number of records accessed by transaction = 50
Processing time for each record = 1.0 unit of time
System overhead for lock and unlock = neglected
Transaction arrival rate (λ) = number of transactions that arrive per unit of time

Figure 9. The expected response time of transaction for various number of data items in the database for 2PL in the best placement case.



Total number of records in database = 5000
Number of records accessed by transaction = 50
Processing time for each record = 1.0 unit of time
System overhead for lock and unlock = neglected
Transaction arrival rate (λ) = number of transactions that arrive per unit of time

Figure 10. The expected response time of transaction for various number of data items in the database for 2PL in the random placement case.



Total number of records in database = 5000
Number of records accessed by transaction = 50
Processing time for each record = 1.0 unit of time
System overhead for lock and unlock = neglected
Transaction arrival rate (λ) = number of transactions that arrive per unit of time

Figure 11. The expected response time of transaction for various number of data items in the database for 2PL in the worst placement case.

decreases or increases. Discontinuities in the graphs of Figures 10 and 11 result from jumps in the transaction size as the number of data items increases.

As a final word of this section, the example shows that the analytic model is easily used to predict the system performance in various system configurations.

VII. CONCLUSIONS AND FUTURE DIRECTIONS

The performance of the two phase locking algoritm has been examined. A simulation model and an analytical model (called iteration solution model) which approximates the lock holding time were developed. The analytical model is superior to the simulation model in its capability in sensitivity analysis and computation efficiency. Thus it is highly desirable to have an analytical model to predict the system performance for a database, even as complicated as the two phase locking protocol. The iteration solution model served as a new method in evaluating this protocol.

New areas such as the effect of transactions having a variety of arrival distributions and including the system overhead for locking, will be valuable to investigate. To explore the possibility of using approximation techniques in analyzing time stamp ordering algorithm will be another meaningful step in evaluating system performance. Another beneficial research area to explore is the performance evaluation for the

the transaction in which each transaction requests *TZ* data items and the types of operations are either reads or writes. For a two phase locking protocol with exclusive locks for both reads and writes, the problem of evaluating the transaction response time is reduced to the iteration solution model discussed in this paper. However, if the two phase locking protocol allows a read operation to share the data item retrieval with other read operations, the read shareability adds another level of difficulty to the evaluation problem. More studies and experiments need to be performed and the validity of the iteration solution method on the shared-read type of transaction model needs to be investigated further.

VIII. REFERENCES

[BERN80] Bernstein, P. A. and Goodman, Nathan, "Fundamental algorithms for concurrency control in distributed database system," Technical Report CCA-80-05 February 15, 1980.

[CHEN83] Chen, D.D., *Performance of Concurrency Control and Data Allocations in Distributed Database Systems,* PH.D. dissertation, U of Michigan, Ann Arbor, April, 1983.

[ESWA76] Eswaran, K.P., Gray, J.N., Lorie, R.A., and Traiger, I.L., "On the notions of concurrency and predicate locks in a relational database system," *CACM*(19,11) November 1976.

[KLEI75] Kleinrock, L., *Queueing Systems, Volume I, Theory,* John Wiley &

Sons, Inc., NY 1975.

[LIN79] Lin, W.T.K., "Concurrency control in multiple copy distributed database system," *Proceedings 4th Berkeley Workshop Distributed Data Management and Computer Networks,* August, 1979.

[LIN81] Lin, W.T.K., "Performance evaluation of two concurrency control mechanisms in a distributed DBMS," *ACM-SIGMOD 1981 International Conference on Management of Data,* April, 1981.

[LIN82] Lin, W.T.K. and Nolte, J., "Performance of two phase locking," *6th Berkeley Workshop on Distributed Data Management and Computer Networks,* February, 1982.

[LITT61] Little, J.D.C., "A proof for the queuing formula: $L=\lambda W$," *Operations Research*(9,3) 1961, pp. 383-387.

[RIES79] Ries, D.R., *The Effects of Concurrency Control on Database Management System Performance,* PH.D. dissertation, U of California, Berkeley, April, 1979.

[STON79] Stonebraker, M., "Concurrency control and consistency of multiple copies of data in distributed INGRES," *IEEE Transactions on Software Engineering* (SE-5,3) May 1979, pp. 188-194.

[YAO77] Yao, S.B., "Approximating block accession in database organization," *CACM*(20,4), April 1977, pp. 260-261.