The Effect of Skewed Data Access on Buffer Hits and Data Contention in a Data Sharing Environment

Asit Dan, Daniel M. Dias and Philip S. Yu
IBM Research Division, T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

In this paper we examine the effect of skewed access on the buffer hit ratio in a multi-system data sharing environment, where each computing node has access to shared data on disks, and has a local buffer of recently accessed granules. In the literature, the effect of skewness in data access on increased data contention has been examined, since with skew most accesses go to few data items. For the same reason, skewness can also increase the buffer hit probability, alleviating the effect on data contention. We examine the resultant effect on the transaction response time, which depends not only on the various system parameters but also on the Concurrency Control (CC) protocol. Furthermore, the CC protocol can give rise to rerun transactions that have different buffer hit probabilities. In a multi-system environment, when a data block gets updated by a system, copies of that block in other system's local buffers are invalidated. We develop a comprehensive analytical buffer model that captures all these effects and integrate it with a CC model to estimate the overall transaction response time. The model is validated through simulations. We find that higher skew does not necessarily lead to worse performance, and that with skewed access optimistic CC is more robust than pessimistic CC. Examining the buffer hit probability as a function of the buffer size, we find that the effectiveness of additional buffer allocation can be broken down into multiple regions that depend on the degree of skewness.

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

1 Introduction

In recent years there has been considerable interest in coupling multiple systems for database transaction processing, for reasons of capacity, availability and cost. One method of coupling multiple systems is the data sharing approach [YU87, STRI82], also referred to as closely coupled clustering[KRON86], where all coupled computer systems have access to shared data stored on disk or on file servers[CHER88]. Each computing system, which we will refer to as a node, also has a local buffer for recently accessed data. In this paper, we focus on modeling the buffer hit ratio in these local buffers. More specifically, we examine the effect of skewed data access patterns on the buffer hit ratio, and its implications on overall system performance. In the literature, the effect of skewed access on the data contention level has been examined, implicitly assuming that the buffer hit ratio is unchanged [TAY85]. Therefore, with a larger fraction of accesses going to a few data items, the data contention level and hence the response time increase. However, with skewed access the buffer hit probability also increases, thus mitigating the effect on data contention. We study the effect of skewed access on both the buffer hit probability and data contention, and project the resultant effect on transaction response time. This approach can also be used to model a distributed shared memory system where memory modules and disks are distributed across a local area network[BELL90].

In the multi-system environment several inter-related factors complicate the analysis of system performance. Consider, for instance, a database consisting of a *hot set* that is frequently referenced, and a *cold set* that contains the remainder. When a system updates a block, all copies of this block in the local buffers of other systems become invalid. Note that hot set data in a buffer is more likely to get invalidated than cold set data. The Concurrency Control (CC) scheme used can also complicate the analysis. Under the Optimistic Concurrency Control (OCC) protocol, conflicting transactions can get aborted and rerun. With sufficient memory, rerun transactions exhibit a higher hit ratio than first run

transactions, since the referenced blocks except those that were invalidated are likely to continue to reside in the memory after access during the first run. Data brought in by rerun transactions also affect the buffer hit probability of first run transactions. The buffer hit ratio is thus affected by the abort probability. On the other hand, the buffer hit ratio not only determines the number of IO's but also impacts the abort probability since a lower buffer hit ratio leads to a longer execution time, and therefore to a higher probability of data contention. Skewed access also has other effects. Intuitively, skewed accesses may be expected to increase the data contention because a disproportionate fraction of the accesses goes to the hot set. Further, since the hot set is more likely to be retained in the buffer, skewed access may be expected to increase the buffer hit probability. However, a higher buffer hit probability can reduce the transaction response time and hence lower the data contention level. Furthermore, a higher data contention also implies a higher buffer invalidation rate, thus lowering the buffer hit ratio. The net effect of skewed accesses on the response time is thus hard to predict. We develop a comprehensive analytic model to capture these effects.

We explicitly model an LRU buffer capturing skewed access, cross-invalidation and the effect of rerun transactions, to predict the buffer hit ratio of first run and of rerun transactions. We note that the LRU policy and variants thereof are used in commercial database systems [TENG84]. The simple buffer model in [DAN90a] does not easily generalize to skewed data access. We develop a new approach to capture the effect of skewed data access on the buffer hit probability. It is based on two observations: (1) conservation of flow, which can be used to determine the push down rate to each LRU stack location, and (2) the hit probability of each type of granule (e.g. cold or hot) at a given stack location is approximately equal to the relative push down rate at that stack location. We consider both the OCC scheme and the standard two-phase locking scheme (which we will refer to as 2PL) to illustrate the interaction between the database buffer model and CC model and how they can be integrated. The CC and system resource models used are extensions for skewed data access of the models in [YU90a, YU90b, YU90c]. A hierarchical approach is taken to model the overall system performance, where the buffer submodel, the CC submodel and the hardware resource submodel are first developed separately and a higher level model integrates the three submodels to predict the overall transaction response time. The submodels interact with each other through some parameters, and hence, they are solved simultaneously using an iterative procedure. A detailed simulation model is also developed to validate the analysis. Because the analysis is simple, it is striking that it matches closely to results from simulation for such a complex environment. This is not only true for the aggregate measures like the response time but also true for all of the fine measures such as the buffer hit ratio for the first run and higher runs, the abort probability, and the lock contention probability.

There are few existing analytical models for the database buffer particularly for a multi-system environment. The importance of considering skewed data access for database applications is discussed in [CASA89], where a replacement strategy is modeled that fixes some buffers for the most frequently used blocks, and uses the remaining buffers to read in other blocks. As the authors point out, their model is optimistic for actual replacement strategies like LRU buffer management. Previous models of the database buffer for multi-system data sharing [YU87, DIAS88] have been empirical, based on trace driven simulations. We mention in passing that there are a number of studies in the literature on multiprocessor cache coherency. In general, the workloads are rather different for the database buffer and multiprocessor cache environments. The cache replacement policies considered in some of these works are also simple. In [YANG89] a random replacement policy is considered and in [GREE87] cache organization is based on direct mapping. The most relevant work in this context is the multiprocessor cache coherency model in [DUBO82], where the analysis is based on a priori knowledge of the curve of the hit ratio as a function of cache size.

In a data sharing environment, it is a hard problem to select the appropriate buffer size for each system to meet a given performance criterion. The composition of the buffer contents (hot versus cold granules) depends on not only the buffer size but also on the degree of skew, the number of nodes, the transaction rate and the concurrency control protocol. We find that although the buffer hit ratio of hot granules decreases with the number of nodes, the buffer hit of cold granules increases with the number of nodes. The buffer hit probability is also affected by the CC protocol in subtle ways. For instance, under OCC, rerun transactions have a higher cold granule hit probability than that for hot granules, while the reverse is true for first run transactions. We also find that the increase in buffer hit probability due to additional buffer allocation can be broken down into multiple regions that depend on the access skew, and that the OCC scheme is more resilient to data skew.

The paper is organized as follows. Section 2 outlines the data sharing environment and the transaction

model. The buffer model, and its integration with the concurrency control and system resource models, is described in Section 3. Model validation, and projections from the model are described in Section 4. In section 5, using the analytical model, we explore at length the effect of skewed access on the buffer hit probability and the transaction response time. A summary and concluding remarks appear in Section 6.

2 The Data Sharing Environment

The data sharing system considered consists of N loosely coupled nodes sharing a common database at the disk level (Figure 1). We assume that the database consists of D granules, where a granule is the unit of transfer between disk and buffer, i.e., a block. The execution of a transaction is modeled as consisting of three phases: initial set up, execution, and commit, as in [DAN90a, YU90b, YU90c]. Transactions arrive at each node according to a Poisson process with rate λ. Each granule access is assumed to be independent of all other granule accesses. We assume that each transaction accesses L granules from the shared database. The access pattern within the database may be skewed, i.e., some granules are accessed more frequently than others. Based on the frequency of data access, the data granules are grouped into M partitions, such that the probability of accessing any granule within a partition is uniform. Let β_i denote the fractional size of the database partition i, i.e., the size of partition i is $\beta_i D$. Let α_i denote the probability that any database access lies in partition i.

To improve the transaction response time, each node has a local buffer of size B and caches a part of the database in this buffer to reduce the transaction response time. Hence, copies of the same granule may be present at more than one node. Each node uses an LRU (Least Recently Used) buffer replacement scheme for its local buffer management. To access a data granule, a transaction requests a copy of the item from the local buffer manager. The buffer manager returns a copy of the granule to the requesting transaction if the granule is present in the buffer. Otherwise, a copy of that granule is brought in from the shared disk to the local buffer. In either case, the newly accessed granule is placed at the top of the LRU stack (Figure 2). In the case when a new granule is brought in from the disk, if there is no free buffer available, then the granule at the bottom of the LRU stack is pushed out of the stack. During the execution of a transaction, its updates are made on its local copies. At the commit time of the transaction, its updates are made permanent in the local buffer and in the shared disk, replacing the old copies. The identities of updated granules are also broadcast to all remote nodes so that each remote node can invalidate the old copies of the updated granules if present in their local buffers. At the remote nodes, the invalidated buffer locations are placed at the bottom of the LRU stack and are made available for bringing in new granules. The probability that a granule accessed from the i^{th} partition is also updated is denoted as p_{u_i}. Thus, the average rate at which granules of partition i are updated is given by $N\lambda L\alpha_i p_{u_i}$.

Note that all transactions may not successfully commit in their first run. The number of rerun transactions are particularly significant under OCC. (Under 2PL, transactions are only aborted during deadlock.) The rerun transactions need to bring back from the disk the granules that were invalidated by the remote nodes, if not brought in by other concurrently running transactions.

3 Integrated Systems Model

The execution time of a transaction depends on three main factors: 1) the buffer hit probability, which determines the number of I/Os to be performed by the transaction, 2) the concurrency control protocol used for resolving conflict in accessing data granules (waiting, abort etc.), and 3) the processing time and the queueing delay in accessing system resources such as CPU, etc.. We model buffer hit probability, system resource access times and concurrency control separately, and capture their interactions through a higher level model. The granule hold time (which is the time duration from the access of the granule by the transaction to either the completion of the transaction commit process or the abortion of the transaction) depends on the buffer hit probability estimated by the buffer model, and by the queueing and services times estimated by the resource model. The CC model estimates the transaction abort probability or the lock wait times based on the granule hold times, and this in turn affects both the buffer and resource models.

3.1 Buffer Model

To simplify our presentation, we first develop a simple buffer model ignoring the effect of database access by rerun transactions (aborted transactions). The simple model is quite accurate for 2PL where the number of aborted transactions due to deadlock is negligible except for a very high conflict situation. We then refine our model to include the effect of rerun transactions.

3.1.1 Buffer Model without Rerun Transactions

Since the data sharing system is homogeneous, we focus our attention on a single buffer. We extend the

analysis for multi-partition access under the LRU replacement scheme by Dan and Towsley [DAN90b] to capture inter-system buffer invalidation. To estimate the steady state probability of buffer hit, we first derive the average number of granules of each partition in the local buffer of any node. Let $Y_i(j)$ denote the average number of granules of partition i in the top j locations of the LRU stack. Therefore, the buffer hit probability of i^{th} partition is $Y_i(B)/(\beta_i D)$, and the overall buffer hit probability for a granule requested by a transaction is estimated as

$$h_1 = \sum_{i=1}^{M} \frac{\alpha_i Y_i(B)}{\beta_i D}. \quad (1)$$

Let $p_i(j)$ be the probability that the j^{th} buffer location from the top of the LRU stack contains a granule of partition i. Then,

$$Y_i(j) = \sum_{l=1}^{j} p_i(l). \quad (2)$$

We will set up a recursive formulation to determine $p_i(j+1)$ for $j \geq 1$ given $p_i(l)$ for $l = 1, ..., j$. Consider a smaller buffer consisting of the top j locations only. The buffer location $(j+1)$ receives the granule that is pushed down from location j. Let $r_i(j)$ be the rate at which granules of partition i are pushed down from location j. Our estimation of $p_i(j)$ is based on following two observations.

Conservation of flow: Under steady state condition, the long term rate at which granules of the i^{th} partition get pushed down from the top j locations of the buffer equals the difference of the miss rate and the invalidation rate of the i^{th} partition from the top j buffer locations (Figure 2); Otherwise, the average number of granules of partition i in the smaller buffer consisting of top j locations would become unbounded. The rate at which granules of partition i are brought to the smaller buffer consisting of the top j locations is $\lambda L \alpha_i (1 - Y_i(j)/\beta_i D)$, i.e., the rate of buffer miss in the top buffer. Hence, the push down rate, $r_i(j)$ is given by

$$r_i(j) = \lambda L \alpha_i (1 - \frac{Y_i(j)}{\beta_i D}) - (N-1)\lambda L \alpha_i p_{x_i} \frac{Y_i(j)}{\beta_i D}. \quad (3)$$

Relative push down rate: The expected value of the probability of finding a granule of the i^{th} partition in the $(j+1)^{st}$ buffer location over all time, $p_i(j+1)$, is approximately the same as the probability of finding a granule of the i^{th} partition in the $(j+1)^{st}$ buffer location in the event that a granule is pushed down from location j to location $(j+1)$. Formally, *Prob* {*location* $(j+1)$ *contains a granule of partition* i | *a granule is pushed from location* j *to location* $(j+1)$} $\approx$ *Prob* {*location* $(j+1)$ *contains a granule of partition* i}. Hence,

$$p_i(j+1) \approx \frac{r_i(j)}{\sum_{l=1}^{M} r_l(j)}, \quad j = 1...B-1. \quad (4)$$

Note that instantaneous value of $r_i(j)$ is dependent on the content of the top j buffer locations, and the more accurate estimation of $p_i(j)$ requires the precise distribution of the content of j buffer locations.

Equations 2, 3 and 4 can be solved iteratively, with the base condition of $p_i(1) = \alpha_i$. At the point, when $Y_i(j)$ is very close to its limit $(\beta_i D)$, $Y_i(j)$ may exceed $\beta_i D$ because of the approximation in the above equations. This is corrected by resetting $Y_i(j)$ to $\beta_i D$ whenever $Y_i(j)$ exceeds $\beta_i D$ and $r_i(j)$ is taken to be zero for all subsequent steps for that partition. Note that, although $r_i(j)$ is a function of the transaction rate (λ), $p_i(j)$ and therefore, h_1 are independent of λ, because λ cancels out in Equation 4.

3.1.2 Buffer Model with Rerun Transactions

The simple buffer model described in the previous subsection ignored rerun transactions. Here, we consider the impact of rerun transactions on the buffer hit probability of first run transactions, as well as the buffer hit probability of rerun transactions. The buffer hit probability of rerun transactions is 1 if the delay between two runs is not so large as to flush out any of the granules brought in during the previous run, and if none of the granules of the rerun transaction are lost due to invalidation. We assume that the buffer size is large enough to accommodate the working set of all active transactions and we ignore the effect of flushing in our subsequent analysis. In [DAN89] a condition is derived for flushing not to occur. With high invalidation rate the buffer hit probability of rerun transactions becomes less than 1, since some of the granules brought in during the previous run are invalidated. The rerun transactions bring back those invalidated granules leading to better use of buffer locations that would otherwise contain invalid blocks, and therefore the buffer hit probability of the first run transactions is also improved. The probability of abort and the number of

granules lost during a rerun will depend on the concurrency control protocol. We postpone their derivation until we detail our concurrency control protocol in Section 3.2. Let $L_{r,i}$ denote the average number of granules of partition i brought in by a r^{th} run transaction. Also, let P_r denote the probability that a transaction is executed at least r times. The buffer hit ratio of a r^{th} run transaction can be approximated as

$$h_r = 1 - \frac{\sum_{i=1}^{M} L_{r,i}}{L}, \quad r > 1. \tag{5}$$

To determine the buffer hit probability of the first run transaction we modify Equation 4 to reflect the additional granules brought into the buffer by rerun transactions. Hence,

$$r_i(j) \approx \lambda L\alpha_i(1 - \frac{Y_i(j)}{\beta_i D}) + \lambda \sum_{r=2}^{\infty} P_r L_{r,i} - (N-1)\lambda L\alpha_i p_{u_i} \frac{Y_i(j)}{\beta_i D}, \tag{6}$$

where P_r and $L_{r,i}$ are given in Equations 8 and 9 in Section 3.2. Note that, P_r in Equation 6 is a function of λ, and λ does not cancel out in Equation 4. Hence, unlike the previous case, the buffer hit probability in the presence of rerun transactions is dependent on the transaction rate. Note that for the single node case, there is no buffer invalidation. Therefore, $p_i(j)$ and the buffer hit probability are independent of transaction rate.

3.2 Concurrency Control Model

Since the focus of this paper is the buffer model for data sharing in the presence of data skew, the Concurrency Control (CC) model is outlined here for completeness only. Details of the CC models for 2PL and OCC can be found in [YU87, YU90a, YU90b, YU90c] where a mean value model is used as in other studies in the area such as those in [TAY85, DAN88]. There are many variations of the OCC protocol. Here, we use the pure OCC protocol where transactions are aborted only at the end of the execution phase, if they conflict with committing transactions. We will briefly describe below the model for OCC to illustrate the approach and refer the reader to the citations for details of the 2PL model.

The transaction model consists of $L+2$ states, where L is the fixed number of granules accessed, as described in Section 2. State 0 models the initial set up phase, and is modeled as contributing time R_{INPL} to the average transaction response time, corresponding to an average of I_{INPL} instructions and O_{INPL} I/Os per transaction for setup. Following this, a transaction progresses to states 1, 2, ... , L, in that order. At the start of each state $i > 0$ the transaction accesses a new granule, and moves to state $i+1$ when the next granule is accessed. In state i, the transaction has informed the CC manager of access to i data granules. After state L, if the transaction is aborted, it returns to state 1, and progresses as before. Otherwise, it enters commit processing in state $L+1$. In the r^{th} run of a transaction, the average time in state i is modeled as R_i^r, corresponding to execution of an average of I_i instructions, and an average of $(1 - h_r)$ I/Os. At commit time, after accessing L granules, the concurrency control manager is so informed. If the transaction entering commit was marked for abort, it is restarted after a (fixed) wait time of $T_{Backoff}$. Otherwise, the transaction enters commit processing, broadcasts invalidations, and marks all conflicting transactions for abort. It then writes commit records to the log and propagates the updates to the shared disk. All of these take an average time of T_{Commit}. During commit processing, exclusive access on the granules accessed is retained, and any ongoing transactions that access granules that conflict with the committing transaction are marked for abort.

We extend the approximation in [YU90b, YU90c] for the probability of abort P_A^1 in the first run of a transaction to capture skewed data access. This is approximated as,

$$P_A^1 \approx 1 - \prod_{l=1}^{L} \left[\left(\prod_{i=1}^{M} (1 - \frac{l\alpha_i}{D\beta_i})^{LN\lambda\alpha_i p_{u_i} R_l^1} \right) \left(1 - \sum_{i=1}^{M} \frac{LN\lambda\alpha_i^2 p_{u_i} T_{Commit}}{D\beta_i} \right) \right]. \tag{7}$$

The rationale for the above approximation is as follows. By state l, a transaction has accessed on an average $l\alpha_i$ granules from the i^{th} partition. Hence, the factor $(1 - l\alpha_i/D\beta_i)$ is the probability that a granule of partition i accessed by a transaction entering the commit phase does not conflict with a transaction that has accessed $l\alpha_i$ granules from the same partition, assuming equiprobable access to each granule within a partition. $L\lambda\alpha_i p_{u_i} R_l^1$ is the average number of granules of the i^{th} partition updated by transactions entering commit during the period of average duration R_l^1 that the transaction is in state l. Hence, the first term in the outer product accounts for the probability that none of the transactions entering commit conflict with the transaction at state l. The second term in the outer product accounts for the contention probability on accessing the new granule at the l^{th} state, with transactions holding exclusive access on granules during commit processing. There are $L\lambda\alpha_i p_{u_i} T_{Commit}$ granules of the i^{th} partition held by transactions in commit processing. Since the

probability that the newly accessed granule lies in the i^{th} partition is α_i, the second product term is the probability that accessing the new granule does not conflict with the committing transactions. This approximation in [YU90b, YU90c] for uniform access was found to compare very well with simulations, and simulation results of section 4 provide evidence that the extension for skewed access is accurate. In a similar manner, the probability of abort P_A^r for the r^{th} run transaction is approximated as in Equation 7 replacing R_l^1 by R_l^r.

Given P_A^r, the parameter P_r, i.e., the probability that a transaction has an r^{th} run, can be approximated as

$$P_r \approx \prod_{i=1}^{r-1} P_A^i \quad r > 1. \tag{8}$$

The average number of granules of the i^{th} partition lost by a 2nd run transaction due to buffer invalidation can be approximated as

$$L_{2,i} \approx \frac{L\alpha_i N\lambda L\alpha_i p_{u_i}}{D\beta_i} \frac{(N-1)}{N} \left(\frac{T_{1,read}/2}{P_A^1} + T_{Backoff} \right) \tag{9}$$

where $T_{1,read} = LR_l^1$. $N\lambda L\alpha_i p_{u_i}$ is the rate at which granules of partition i are updated and $(N-1)/N$ is the probability that the update is remote. Hence, $(L\alpha_i \times (N-1)\lambda L\alpha_i p_{u_i} \times T_{1,read}/2)/(D\beta_i))$ is the average number of granules of partition i lost by a transaction during its execution. Given that a transaction is aborted, the conditional probability of a granule accessed by the aborted transaction has a higher probability of being invalidated, and hence, the term P_A^1 appears in the denominator. The aborted transaction may lose additional granules during the backoff period and this is approximately given by the expression $(L\alpha_i \times \lambda L\alpha_i p_{u_i} \times T_{Backoff}/(D\beta_i)) \times (N-1)/N$. The expression for $L_{r,i}$ for $r > 2$, can be obtained similarly by replacing $T_{1,read}$ by $T_{r-1,read} = LR_i^{r-1}$ and P_A^1 by P_A^{r-1}.

For 2PL, we assume the number of rerun transactions to be negligible, and Equation 4 (instead of Equation 6) is used to determine the buffer hit ratio. The analysis for lock contention probability(P_{cont}) and average wait time(W) are described in detail in [YU87, YU90a].

3.3 System Resource Model

We assume that each node consists of K tightly coupled processors and that the database is spread over multiple disks. The processors can be modelled as an M/M/K server with FCFS discipline where the disks are modelled as an infinite server. We first consider the OCC case. The processor utilization can be estimated as

$$\rho = \frac{\lambda}{K \times MIPS} \left(I_{INPL} + \sum_{l=1}^{L} I_l + (I_c + (N-1)I_v) + \sum_{r=2}^{\infty} P_r \left(\sum_{l=1}^{L} I_l \right) \right) \tag{10}$$

where MIPS is the processor speed. In the above, the probabilities of abort are estimated in terms of the average times in each state R_i^r, and the utilization ρ is expressed in terms of the abort probabilities. Now, R_i^r can be estimated from ρ based on an M/M/K assumption as in [YU90b, YU90c] and the overall average transaction response time is estimated as

$$R = R_{INPL} + \sum_{l=1}^{L} R_l^1 + T_{Commit} + \sum_{r=2}^{\infty} P_r (T_{Backoff} + \sum_{l=1}^{L} R_l^r). \tag{11}$$

We note that P_A^r for $r > 2$ quickly approaches a constant value and the above equations can be closely approximated by closed form expressions.

Similarly, the overall average transaction response time for 2PL is estimated as [YU87, YU90a]

$$R = R_{INPL} + \sum_{l=1}^{L} R_l^1 + LP_{cont}W + T_{Commit}. \tag{12}$$

4 Validation and Results

To validate our model, we use a detailed discrete event simulation model, simulating all three components of the integrated system model: LRU buffer replacement policy, concurrency control (2PL and OCC) and FIFO queueing model for CPU. The skewed access pattern is modeled as access to two kinds of data (*hot* and *cold*). The simulation explicitly keeps track of the buffer contents at each node according to the LRU policy. For buffer misses, an I/O delay is modeled. The simulation also keeps track of the data accessed by each transaction and explicitly simulates buffer hits, data contention, transaction aborts, locking of data granules, waiting for locks to become available, queueing and processing at the CPU, I/O waits, and commit processing.

For 2PL, in the case of lock requests leading to a data contention, the transaction is placed in a wait state until the lock is released by the transaction holding the lock (while the approximate analysis estimates the average lock wait time). For 2PL, if a lock request leads to a deadlock, the transaction making the request is aborted and restarted after a back-off delay. (The approximate

analysis assumes that the probability of deadlock is very small compared to the contention probability, which is also confirmed by the simulations.) For OCC, at commit time transactions are checked to see if they have been marked for abort; if not, any running transactions with conflicting access are marked for abort. (By comparison, the analysis estimates the probability of transaction abort.) At transaction commit time, for each update the buffer at each remote node is checked and copies of the updated block are invalidated. Overheads for the buffer invalidation are also explicitly simulated. Each rerun transaction makes the same references as its first run, and buffer hits result only if a copy is still in the buffer (while the analysis assumes that the buffer is large enough so that buffer flushing does not occur).

Tightly coupled processors are modeled as having a common queue, while loosely coupled nodes have separate queues. The CPU service times are constants that correspond to the CPU MIPS rating and the specific instruction pathlengths given below. (They are not exponentially distributed as in the M/M/K analytical model of the CPU.) The CPU is released by a transaction when lock contention occurs, for each I/O, during broadcast invalidation, and during backoff after an abort. The simulation model measures the buffer hit probabilities of first run and rerun transactions for both hot and cold data accesses. It also measures the transaction response time and various effects of concurrency control on performance measures such as the probability of abort or conflict, waiting time, etc..

Since the various components of the integrated system model interact with each other, the model components cannot be validated in isolation and the validation of each component is sensitive to the correctness of the other components. We will first focus our attention on the validation of the buffer model and will also explore the effect of various parameters on the buffer hit probability. We will then validate the integrated model and examine the effect of buffer hit probability on the transaction response time, and the sensitivity of the buffer hit ratio and response time to skew.

Figure 3 shows the various transaction and system parameters which are kept fixed for all simulations, unless otherwise specified. The transaction parameters (number of accesses and instructions) are similar to those in [YU87], derived from customer traces. We have chosen the database size and access rule parameters to reflect two types of applications. In the first application, the database size is relatively small (10K granules, or about 40 Mbytes) and most of the database accesses go to the hot-set (80-20 rule i.e., 80% of the accesses goes to 20% of the database or 2K granules.). Note that this is a stress case and the buffer hit probability and response time are very sensitive to various parameters (number of nodes, transaction rate, etc.). In the second application, the database size is increased (50K granules or about 200 Mbytes). However, we keep the hot-set size comparable (5% of the database or 2.5K granules) in order to obtain reasonable buffer hits for the same buffer sizes. Furthermore, a large fraction of the accesses (50%) goes to the large cold-set. For this case, the buffer hit probability is less sensitive to various parameters than for the previous case, because of the larger fraction of cold accesses. As we will see, our model is robust, and it works equally well for both the applications modeled. Various buffer sizes are considered to study the sensitivity of buffer hit ratio to buffer size. All buffer sizes chosen satisfy the minimum requirement derived in [DAN89] to avoid buffer flushing. In the simulations, this indeed was found to be the case. We note that further study of larger database sizes indicates that as long as the hot-set size is comparable to the above cases, similar results to that for the second application described above are obtained with the cold-set hit ratio becoming negligibly small.

In the following sections we find a remarkable match between the analysis and simulation results for both the buffer hit probability and overall measures of performance such as the response time. This is true even for extreme values in resource utilization and data contention. We emphasize that while the analysis decomposes the model into separate components and makes approximations in analyzing the LRU buffers, CPU, and data contention, the simulation makes no such decomposition and accurately simulates each of these aspects.

4.1 Buffer Hit Ratio

Since OCC is more complicated to analyze (because of the effect of rerun transactions), we have chosen OCC for these examples, unless otherwise specified. Also, to show the robustness of our model, we will validate the model under the workload of application 1 (high skew). Figure 4 shows the effect of buffer size on the buffer hit probability of first run transactions for different numbers of nodes. The analytical estimate matches very well with the simulation results. For a small buffer size, most of the buffer space is occupied by hot granules. The buffer hit probability increases with the buffer size almost linearly up to the point where very little hot data gets replaced. Any further increase in the buffer space, makes the additional buffer available for cold granules, and subsequent increase in buffer hit probability is small.

In a multi-system environment, the maximum buffer

size that can be effectively used by the hot-set is less than the hot-set size, and is determined by the point where buffer miss rate for the hot set is equal to its invalidation rate[DAN90a]; we will refer to this as the saturation point for the hot set, or the break-point. Since the invalidation rate increases with the number of nodes, the maximum buffer hit probability for the hot-set and the over all buffer hit probability decreases as the number of nodes increases. However, the cold-set is less affected by invalidation, and beyond the saturation point for the hot-set the buffer hit probability curves are parallel to each other. To better understand the effect of invalidation on the buffer hit probability of hot and cold granules, we plot the buffer hit probability for each type of data separately in Figure 5 for a buffer size of 3K granules (12 Mbytes). Note that the buffer size is larger than the hot set size (2K granules). The overall buffer hit probability of first run transactions (indicated as "weighted" in the figure) closely follows the buffer hit probability of hot granules. The invalidation rate increases with the number of nodes, and hence, the buffer hit probability of the hot granules goes down (Equation 6). Since the size of the cold data set is large, very few cold granules are lost from the buffer due to invalidation. The buffer hit probability of the cold granules actually increases with the number of nodes as the effective buffer size that can be exploited by the hot granules decreases.

The buffer hit probability of a rerun transaction depends only on the invalidation rate, and not on the buffer size, unless the buffer size is so small that a granule accessed by a first run transaction is flushed out before the reaccess on its rerun. Figure 6 shows the effect of invalidation on the buffer hit probability of a rerun transaction. Since the size of the cold data set is large, very few cold granules are lost from the buffer due to invalidation. On the other hand, the buffer hit probability of the hot granules falls slowly with an increasing number of nodes. The explanation is that aborted transactions have misses during their rerun corresponding to conflicts with transactions running on other nodes; while the number of such conflicts increases, the probability that more than one conflict occurs for a single transaction is small. Again, most of the decrease in the buffer hit probability is due to the invalidation effect on the hot set.

4.2 Transaction Response Time

Figure 7 shows the effect of buffer size on the transaction response time for a highly skewed workload (application 1). The parameter in this chart is the number of nodes N. The match between the analytical prediction and simulation is excellent for all cases. The increase in buffer hit probability translates to a decrease in response time, and hence, these curves are inversely related to the buffer hit curves (Figure 4). For a single node, the response time decreases significantly until the saturation point for the hot-set. Once the hot-set saturation occurs, the response time drops at a nearly linear but slower rate. The response time for the multi-node case is higher due to several factors: the higher data contention (higher transaction load), the lower buffer hit probability, and increased commit delay. The saturation point occurs for a smaller buffer size for the multi-node case than for the single node case due to the invalidation effect. Comparing Figures 4 and 7, notice that the saturation point occurs at the same buffer sizes for corresponding parameters. The invalidation rate also depends on the update probability. The dashed-curve for a 2 node case shows that the saturation point occurs at a larger buffer size for a lower probability of update (0.25) of the hot-set. The conclusion to be drawn from these observations is that, beyond a certain buffer size, the response time cannot be reduced simply by increasing the private buffer size.

Our next set of graphs are for the integrated model using the workload of application 2. Figure 8 shows the estimate of the probability of abort under OCC for buffer sizes of 1K, 3K and 8K granules (i.e., 4 Mbytes, 12 Mbytes and 32 Mbytes) respectively. It shows the effect of scaling up the system by increasing the number of nodes but keeping the transaction load per node as constant. The first run probability of abort decreases with an increase in the buffer size. However, the rerun probability of abort is almost the same for both buffer sizes since it is rare to have more than one granule invalidated and thus the the rerun execution time is nearly the same. Figure 9 shows the response time corresponding to Figure 8. For a small buffer size, invalidation has a smaller impact on the buffer hit probability, and hence, the increase in response time is primarily due to increased data contention and commit delay (broadcast invalidation messages). Since the buffer hit probability is more sensitive to invalidation, the response time increases more sharply for a larger buffer size.

5 Effect of Skewness

Skewness in the access pattern can result in a disproportionate access rate to a fraction of the database, and therefore tends to increase the level of data contention. However, skewness also results in a higher buffer hit probability, since the frequently accessed granules tend to remain in the buffer. In turn, a larger buffer hit probability decreases the average granule hold time, and this decreases the level of data contention. Further, an increase in data contention also leads to an increase in the

invalidation rate and consequently, to a reduced buffer hit probability. Hence, the resultant effect on the response time is hard to anticipate. In this section, we will use the analytical model to explore the effect of skewness.

To vary the skewness, we vary the frequency of accessing the hot-set (α), keeping the size of the hot-set (βD) as constant. The remaining parameters are the same as that of application 2. Figures 10 and 11 compare the effect of skewness on the buffer hit probability for a multi-system environment with $N = 6$ and $\lambda = 7$ and 25, respectively for OCC. Also shown is the case of 2PL with $\lambda = 7$. Increase in α not only increases the buffer hit probability of the hot-set, but also implies that a higher fraction of accesses goes to the hot-set. Hence, the overall buffer hit probability increases in a non-linear manner. For a small buffer size (0.5K), most of the granules present in the buffer are hot, and for a very large buffer (8K), the maximum fraction of the hot set that can be retained is in the buffer. Hence, for both these cases, increase in α does not significantly change the composition of the buffer, and the increase in the overall buffer hit probability is close to linear due to the increase only in the access rate to the hot set. For intermediate buffer sizes, increase in skewness increases the proportion of the hot-set in the buffer until the maximum fraction of the hot-set that can be retained in the buffer is reached. Beyond this point, the buffer hit probability increases again linearly with α; further, the buffer hit probability increases very little with increasing buffer size (compare $B = 2K$, $B = 4K$, and $B = 8K$ for OCC and 2PL in Figure 10) and the buffer composition does not change with skewness.

As explained in Section 3, the buffer hit probability under 2PL (negligible rerun transactions) does not depend on the transaction rate, while this is not so for OCC (significant rerun transactions). The rerun transactions bring back some of the invalidated granules. This has a compensating effect on the buffer hit probability of the first run transactions. The number of granules brought back by a rerun transaction increases with the transaction rate. Thus, as the transaction rate increases, the buffer hit probability of the first run transaction under OCC becomes higher than that under 2PL. For a low transaction rate ($\lambda = 7$), the buffer hit probabilities under both protocols are very close. However, for a higher transaction rate ($\lambda = 25$), the buffer hit probability under OCC is significantly higher than that for a lower transaction rate, for a large buffer size ($B > 500$). Under 2PL, such a high transaction rate can not be supported due to high data contention. It is only included in Figure 11 for comparison purposes. Recall from Figure 4 that the maximum buffer size needed to satisfy the hot-set depends on the invalidation rate. Nevertheless, Figure 11 shows that a larger fraction of the hot-set can be retained in the buffer under OCC than for 2PL.

Figure 12 compares the transaction response time for a transaction load of $\lambda = 7.0$ per node under 2PL and OCC. Under 2PL, for a small buffer size ($B = 0.5K, 1K$) and for a large buffer size ($B = 8K$) the response time increases with skewness. With increasing skewness, the increase in data contention dominates the increase in buffer hit probability. For an intermediate buffer size ($B = 2K, 4K$), the response time initially decreases a little due to increased buffer hit probability until the point where maximum hot-set that can be retained in the buffer is reached. This is smaller than the hot-set size and beyond this point the response time increases with skewness. Under OCC, the response time increases slightly with increase in skew for very small ($B = 0.5K$) and large ($B = 8K$) buffer sizes. For intermediate cases, the response time either decreases with increase in skew ($B = 1K$) or first decreases and then increases with increasing skew ($B = 2K, 4K$).

6 Summary and Conclusions

In this paper we studied the combined effect of skewed data access on both the buffer hit probability and the data contention probability, and the overall effect on transaction response time in a multi-system data sharing environment. We examined the effects of varying the number of nodes, buffer size, update probability, transaction rate, the degree of skewness, and the CC protocol on the buffer hit probability, the data contention probability (2PL) the transaction abort probability (OCC) and transaction response time. We developed a simple approximate analytical model for database buffers with skewed access in this environment. We focused on the case where the database consists of a hot set of frequently referenced granules, and a remaining cold set of granules. We modeled the LRU buffer replacement policy at each node, and estimated the probabilities of hot and cold granules at all levels of the LRU chains, taking into account the buffer invalidations received from other nodes. These LRU buffer occupancy probabilities are then used to estimate the overall buffer hit probabilities for hot and cold granule requests. The model is generalized to handle skewed access to an arbitrary number of partitions of the database. The buffer model was integrated with models for the CC protocol (both OCC and 2PL), and with queueing model for resources, in order to estimate overall transaction response time. A detailed simulation model was developed to validate the analytical model. The results from the simulation and analytical models

showed excellent agreement.

Examining the buffer hit ratio with increasing number of nodes, we found that while the hot set buffer hit ratio decreases significantly with the number of nodes, the cold set hit ratio can actually increase. This is because the hot set is particularly susceptible to buffer invalidation since it is likely to reside in several buffers. Due to this hot set invalidation a larger part of the buffer becomes available for cold granules. The overall hit ratio closely follows the hot set hit ratio, and both are larger than the cold set hit ratio. For rerun transactions under OCC, the cold set hit ratio is close to unity while the hot set hit ratio decreases slowly with increase in the number of nodes. This is because most aborts are primarily due to contentions on hot set data, which are therefore invalidated if the contention is with a transaction running on another node, causing a buffer miss on rerun. Aborts are predominantly due to a single contention, usually causing a single buffer miss on rerun, provided that the data has not been flushed out of the buffer before the rerun occurs. Because of the high buffer hit ratio for rerun transactions under OCC, the abort probability of rerun transactions was very small. As a result, the OCC protocol was found to be more robust at high levels of data skew.

Examining the transaction response time as a function of buffer size shows several regions which exhibit different effectiveness for additional allocated buffer. In the first region, data referenced during the first run of transactions are flushed from the buffer before reference during the rerun of aborted transactions. The response time decreases sharply as the buffer size exceeds this threshold [DAN89]. In the second region, the invalidation rate for hot set granules is less than the re-reference rate; when the buffer size becomes large enough so that this condition is no longer satisfied, additional buffer cannot be used effectively by the hot set, and is used instead by cold set granules. In the second region, the response time shows a close to linear decrease with increasing buffer size. Beyond this region, the decrease in response time falls off to a lower rate. (A data base with multiple partitions in terms of access frequency, will exhibit multiple such regions.) The size of the first region is determined by the working set size. The second region is determined by the 'effective' hot set size that can be buffered, and is affected by the hot set size, the degree of skewness and the number of systems. Our buffer model can provide an accurate prediction of the buffering effect and insights into the factors affecting performance.

Finally, we examined the effect of varying the degree of skew on the buffer hit probability and the response time. Assuming that the hot set size is constant, an increase in the fraction of requests that go to the hot set gives rise to two competing effects. First, the level of data contention increases, since a larger fraction of the requests are made to the same hot set. However, the overall buffer hit probability also increases because a larger fraction of the buffer is occupied by the hot set. For small buffer sizes that cannot accommodate a significant fraction of the hot set, or very large buffers that can accommodate the maximum fraction of the hot set that can be retained in the buffer, the effect of increasing data contention predominates and the response time increases with increase in the skew. For intermediate buffer sizes, the effect of higher buffer hits can predominate over the contention effect and the average response time can actually decrease with increase in the skew.

References

[BELL90] Bellew, M.,Hsu, M., and V. O. Tam, "Update Propagation in Distributed Memory Hierarchy", *Sixth International Conference on Data Engineering*, Los Angeles, CA, Feb. 1990, pp. 521–528.

[CASA89] Casas, R. I., and K. C. Sevcik, "A Buffer Management Model for Use in Predicting Overall Database System Performance", *Fifth International Conference on Data Engineering*, Los Angeles, CA, Feb. 1989, pp. 463–469.

[CHER88] Cheriton, D., "The V Distributed System," *Communications of the ACM*, Vol. 31, No. 3, pp. 314–333, March 1988.

[DAN88] Dan, A., D. F. Towsley, and W. H. Kohler, "Modeling the Effects of Data and Resource Contention on the Performance of Optimistic Concurrency Control Protocols," *Fourth International Conference on Data Engineering*, Los Angeles, CA, Feb. 1988, pp. 418–425.

[DAN89] Dan, A., D. M. Dias, and P. S. Yu, "Buffer Modelling for a Data Sharing Environment with Skewed Data Access", *IBM Research Report RC15296*, 1989.

[DAN90a] Dan, A., D. M. Dias, and P. S. Yu, "Database Buffer Model for the Data Sharing Environment," *Sixth International Conference on Data Engineering*, Los Angeles, Feb. 1990, pp. 538–544.

[DAN90b] Dan, A., and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," *Proc. of the ACM SIGMETRICS*, Boulder, CO, MAY 1990.

[DIAS88] Dias, D. M., B. R. Iyer, and P. S. Yu, "Trade-offs Between Coupling Small and Large Processors for Transaction Processing," *IEEE Transactions on Computers*, Vol. C-37, No. 3, pp. 310–320, March 1988.

[DUBO82] Dubois, M., and F. Y. Briggs, "Effects of Cache Coherency in Multiprocessor," *IEEE Transactions on Computers*, Vol. C-31, No. 11, pp. 1083–1099, November 1982.

[KRON86] Kronenberg, N., H. Levy and W. D. Strecker, "VAXcluster: a Closely-Coupled Distributed System," *ACM Transactions on Computer System*, Vol. 4, pp. 130–146, May 1986.

[GREE87] Greenberg, A. G., and I. Mitrani, "Analysis of Snooping Caches", *Performance*, December 1987.

[STRI82] Strickland, J. P., P. P. Uhrowczik and V. L. Watts, "IMS/VS: An Evolving System," *IBM Systems Journal*, Vol. 21, pp. 490–510, 1982.

[TAY85] Tay, Y. C., N. Goodman and R. Suri, "Locking Performance in Centralized Databases," *ACM Trans. Database Systems*, Vol. 10, No. 4, pp. 415–462, 1985.

[TENG84] Teng, J. Z., and R. A. Gumaer, "Managing IBM Database 2 Buffers to Maximize Performance," *IBM Systems Journal*, Vol. 23, No. 2, pp. 211–218, 1982.

[YANG89] Yang, Q., L. N. Bhuyan, and B. Liu, "Analysis and Comparison of Cache Coherence Protocols for a Packet-Switched Multiprocessor," *IEEE Transactions on Computers*, Vol. C-38, No. 8, pp. 1143–1153, August 1989.

[YU87] Yu, P. S., Dias, D. M., Robinson, J. T., Iyer, B. R. and Cornell, D. W., "On Coupling Multi-Systems Through Data Sharing", *Proceedings of the IEEE*, Vol. 75, No. 5, May 1987, pp. 573-587.

[YU90a] Yu, P. S., D. M. Dias, and S. S. Lavenberg "On Modelling Database Concurrency Control", *IBM Research Report RC15386*, 1990.

[YU90b] Yu, P. S. and D. M. Dias, "Concurrency Control Using Locking with Deferred Blocking," *Sixth International Conference on Data Engineering*, Los Angeles, Feb. 1990, pp. 30–36.

[YU90c] Yu, P. S. and D. M. Dias, "Impact of Large Memory on the Performance of Optimistics Concurrency Control Schemes", *Proc. PARBASE-90: Intl. Conf. on Databases, Parallel Architectures, and Their Applications*, Miami Beach, FL, March 1990, pp. 86–90.

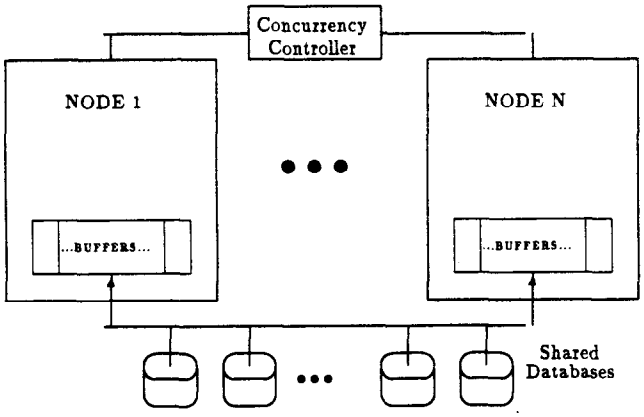

Figure 1: The Data Sharing Environment

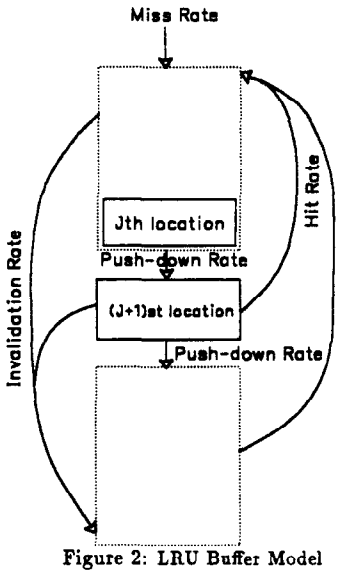

Figure 2: LRU Buffer Model

Application #1	*Database size*	*10K*
	Access rule	*80-20*
Application #2	*Database size*	*50K*
	Access rule	*50-5*
Transaction parameters	*Transaction size*	*15*
	Prob. of update of Hot-set	*0.3*
	Prob. of update of cold-set	*0.1*
System parameters	*CPU per node*	*2*
	MIPS per CPU	*20.0*
	I/O access time	*25ms*
CPU overheads (instructions)	*Initial set up*	*150K*
	Transaction step	*20K*
	Commit (Constant)	*2K*
	Commit (Variable)	*3K*
I/O overheads	*I/O per set up*	*5*

Figure 3: Transaction and system parameters

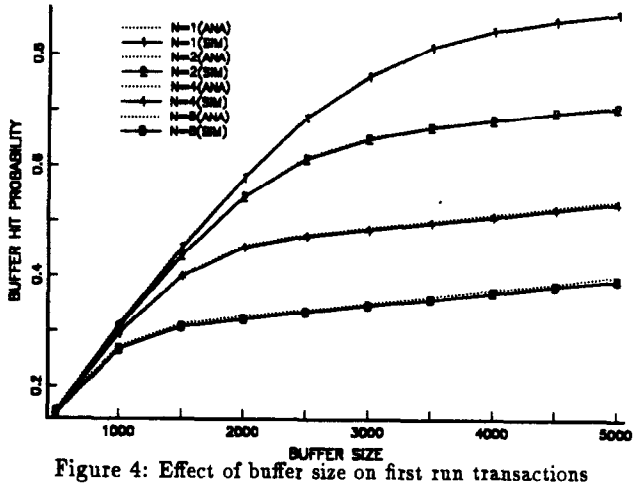

Figure 4: Effect of buffer size on first run transactions

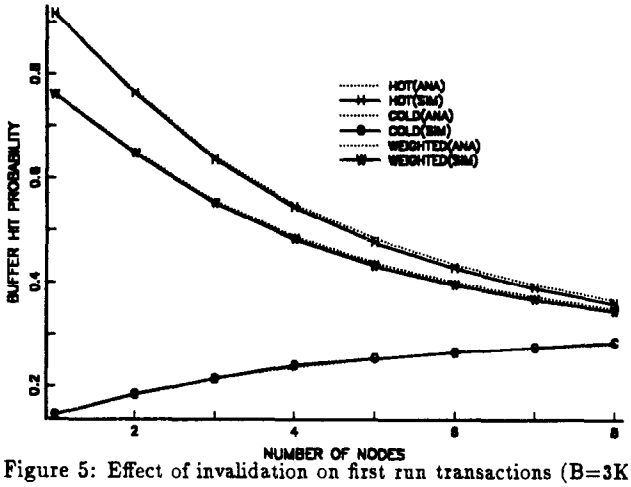

Figure 5: Effect of invalidation on first run transactions (B=3K)

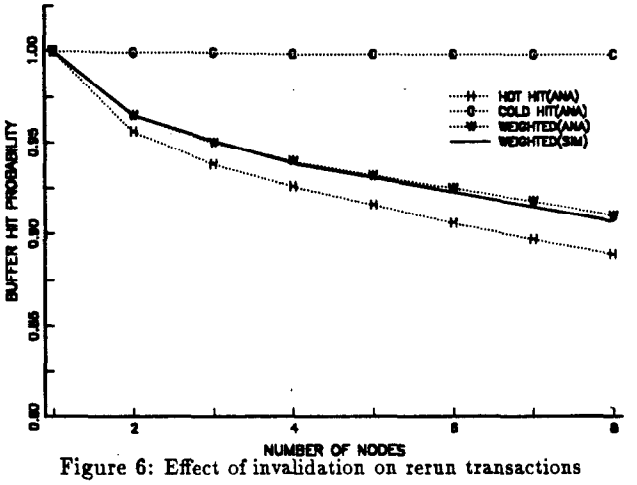

Figure 6: Effect of invalidation on rerun transactions

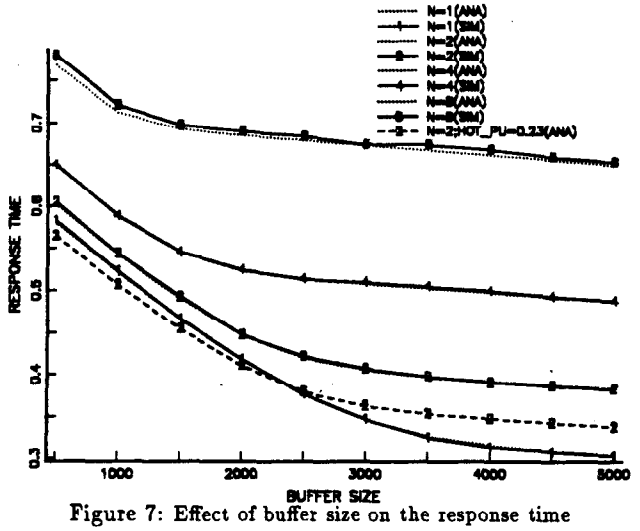

Figure 7: Effect of buffer size on the response time

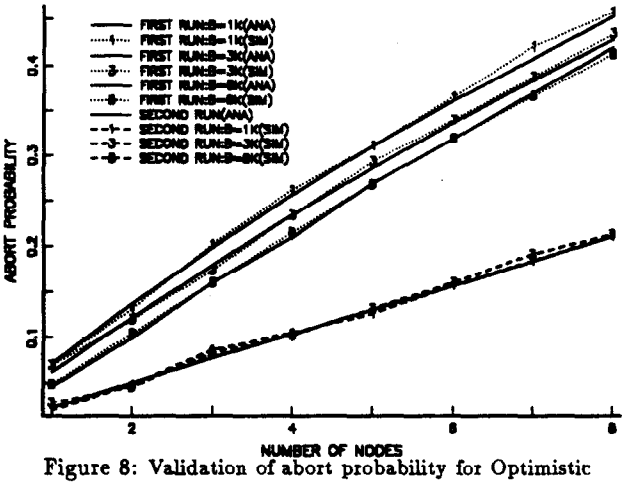

Figure 8: Validation of abort probability for Optimistic

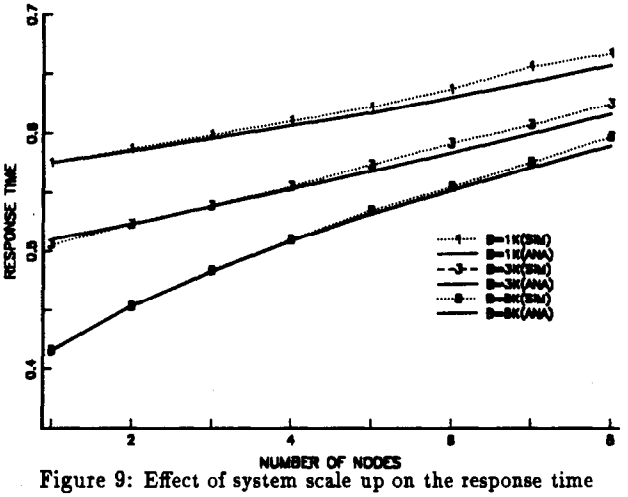

Figure 9: Effect of system scale up on the response time

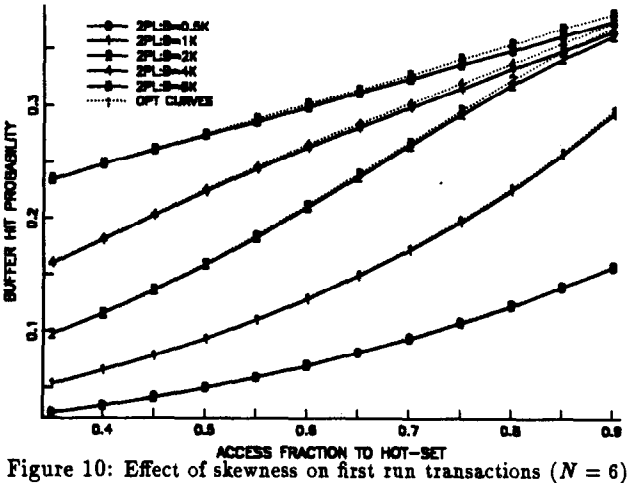

Figure 10: Effect of skewness on first run transactions ($N = 6$)

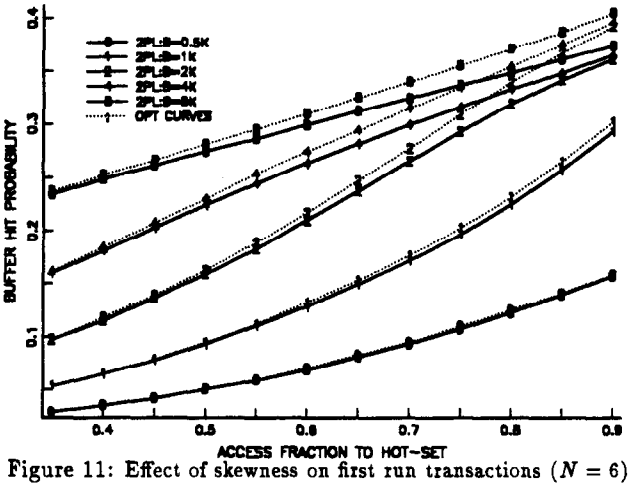

Figure 11: Effect of skewness on first run transactions ($N = 6$)

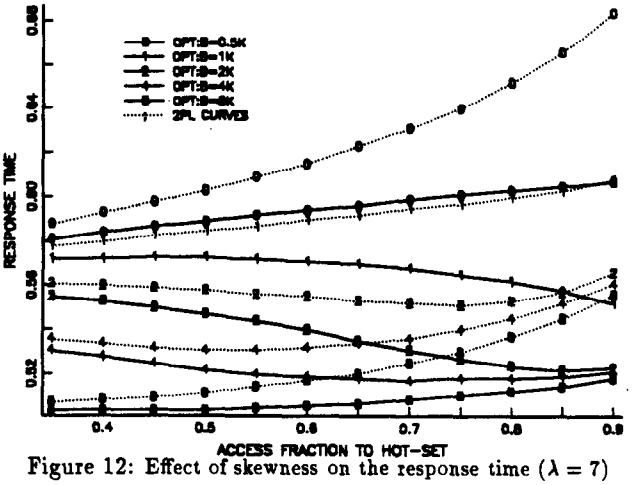

Figure 12: Effect of skewness on the response time ($\lambda = 7$)