AN APPROXIMATE ANALYSIS OF THE LRU AND FIFO BUFFER REPLACEMENT SCHEMES*

Asit Dan† and Don Towsley‡
University of Massachusetts
Amherst, MA 01003

ABSTRACT

In this paper, we develop approximate analytical models for predicting the buffer hit probability under the Least Recently Used (LRU) and First In First Out (FIFO) buffer replacement policies under the independent reference model. In the case of the analysis of the LRU policy, the computational complexity for estimating the buffer hit probability is $O(KB)$ where B is the size of the buffer and K denotes the number of items having distinct access probabilities. In the case of the FIFO policy, the solution algorithm is iterative and the computational complexity of each iteration is $O(K)$. Results from these models are compared to exact results for models originally developed by King [KING71] for small values of the buffer size, B, and the total number of items sharing the buffer, D. Results are also compared with results from a simulation for large values of B and D. In most cases, the error is extremely small (less than 0.1%) for both LRU and FIFO, and a maximum error of 3% is observed for very small buffer size (less than 5) when the access probabilities are extremely skewed. To demonstrate the usefulness of the model, we consider two applications. In our first application, we compare the LRU and FIFO policies to an optimal static buffer allocation policy for a database consisting of two classes of data items. We observe that the performance of LRU is close to that of the optimal allocation. As the optimal allocation requires knowledge of the access probabilities, the LRU policy is preferred when this information is unavailable. We also observe that the LRU policy always performs better than the FIFO policy in our experiments. In our second application, we show that if multiple independent reference streams on mutually disjoint sets of data compete for the same buffer, it is better to partition the buffer using an optimal allocation policy than to share a common buffer.

*This work was supported by a grant from Digital Equipment Corporation.

†Department of Electrical and Computer Engineering
‡Department of Computer and Information Science

1 Introduction

In this paper, we develop approximate analytical models for predicting the buffer hit probability under the Least Recently Used (LRU) and First In First Out (FIFO) buffer replacement policies under the independent reference model. This problem was first studied by King [KING71] who provided an exact analysis for both policies. This analysis was used by several researchers [RAO78,BABA83] to study the merits of different hardware cache organizations and buffer replacement policies. Unfortunately, the computational complexity of evaluating the buffer hit probability using King's model grows exponentially with the buffer size or the number of data items, and hence, is not very useful for large buffer sizes and large number of data items. Consequently, the later studies resorted to simulation in order to study the behavior of large buffers. More recently Flajolet, et.al. [FLAJ87] developed a simpler expression for the buffer hit probability under LRU. However, the computational complexity of evaluating this quantity remains unchanged. Apart from the exact analysis, various performance bounds were obtained by Franaszek and Wagner [FRAN74], and by Aven et. al.[AVEN76]. An asymptotic analysis has also been performed by Flajolet, et.al. [FLAJ87]. There does not appear to exist any simple analysis for estimating buffer hit probabilities.

The models developed in this paper are computationally efficient. The computational complexity for estimating the buffer hit probability of the LRU policy is $O(KB)$ where B is the size of the buffer and K denotes the number of items having distinct access probabilities. In the case of the FIFO policy, the solution algorithm is iterative and the computational complexity of each iteration is $O(K)$. The error in our analysis decreases with increasing values of B and K (less than 0.1% for $K > 20$) and it is also very small for small values of K and B(maximum error is less than 3%).

Dynamic buffer(cache) replacement policies such as LRU and FIFO have wide applications in hardware

cache management [SMIT82,STON89,RAO78], operating system memory management [COFF73], and data base buffer management [LANG77,TENG84,SACC87]. We believe, the simple approximate analytical models developed in this paper, will be useful to gain qualitative insights for various applications, where different sets of data have different access probabilities. For example, the model can be used to investigate the issues of skewness in data access, optimal set size in hardware cache, buffer coherency policies in multi-computer systems. The model has successfully been used for predicting buffer hit probability under a data sharing environment by one of the authors [DAN89].

We present two applications of our models in this paper. In the first application, we consider the behavior of LRU and FIFO in a database environment where there are two classes of data items, a *hot* set and a *cold* set. We show, for a wide range of parameters, that the buffer hit probability under LRU is close to that achieved under a static optimal policy. The static optimal policy has the disadvantage that it requires knowledge of the access probabilities of the data items. Hence, we observe that the LRU policy is preferable in an environment where the hotness of the data changes over time. The assumption of independent references not only simplifies the analysis and makes it tractable, but is also quite realistic for various applications such as database systems, where independent transactions access a small set of data [VERK85,KEAR89]. Finally, the LRU policy always performs better than the FIFO policy in all our experiments.

In our second application, we show that if multiple independent reference streams on mutually disjoint data sets compete for the same buffer space, it is better to partition the buffer amongst the reference streams using an optimal policy, rather than sharing a common buffer. The performance gain due to partitioning depends on the relative skewness of the data streams, and the buffer size. The policy can be used for partitioning of disk cache and partitioning of cache for the instruction and data set of a program [STON89].

We develop the analytical model for the LRU and FIFO policies in Section 2. Results from these models are compared to exact results obtained from King [KING71] and to results obtained from simulations in Section 3. Section 4 contains two applications of the models and Section 5 summarizes and concludes the paper.

2 Analysis of Replacement Policies

We consider a collection of D fixed size items that share a buffer that can store B items. An item may correspond to a line in a cache, a page in memory, or a granule in a database system. The collection of items is partitioned into K partitions labelled $k = 1, 2, \cdots, K$ where the k-th partition contains D_k items, $D = \sum_{k=1}^{K} D_k$, and the probability that any access lies in partition k is α_k, $\sum_{k=1}^{K} \alpha_k = 1$. Let $\{A_i\}_{i=1}^{\infty}$ be a sequence of i.i.d. r.v.'s where A_i denotes the partition from which the i-th item is requested. According to our assumptions, $\Pr[a_i = k] = \alpha_k$, $k = 1, \cdots, K$, $i = 1, 2, \cdots$. This corresponds to the *Independence Reference Model* (IRM) used in many studies of buffer behavior [KING71].

We are interested in the behavior of the LRU and FIFO replacement polices for such a system. Let $\underline{X}_n = (X_{1,n}, \cdots, X_{B,n})$ be the state of the system after the n-th request. Here $X_{i,n}$ denotes the occupancy of the i-th position in the buffer; $X_{i,n} = k$ iff an item from partition k resides in position i after the n-th request. Define $Y_{k,n}$ to be the number of items from partition k that are in the buffer after the n-th request, $Y_{k,n} = \sum_{i=1}^{B} \mathbf{1}(X_{i,n} = k)$, $k = 1, \cdots, K$. Here $\mathbf{1}(P) = 1$ if the predicate P is true and 0 otherwise. We are interested in the stationary behavior of the buffer when it exists, $\underline{X} = \lim_{n\to\infty} \underline{X}_n$, $Y_k = \lim_{n\to\infty} Y_{k,n}$, $1 \le k \le K$. It is easy to show that the LRU and FIFO policies exhibit such stationary behavior [COFF73]. The focus of our analysis is the determination of the stationary buffer hit probability for partition k, $h_k = E[Y_k]/D_k$, $k = 1, \cdots, K$.

2.1 The LRU Buffer Replacement Policy

The buffer can be thought of as a stack under LRU replacement policy. If the newly requested item is not found, it is brought from outside and placed on the stack top pushing all the others down by one position and the least recently used item (the one at the bottom of the stack) is removed from the buffer. However, if the newly requested item is found in the buffer then it is removed from the stack and placed at the top. If the item was originally the j-th element in the stack, then all items in positions $1, \cdots, j-1$ move down one position. The remaining items are unaffected by the move.

The top of the stack is located at position 1 in the buffer and the bottom of the stack in position B. The j-th most recently used item is found in position j, $j = 1, \cdots, B$. Let $\pi(\underline{x}) = \Pr[\underline{X} = \underline{x}]$ where $\underline{x} = (x_1, \cdots, x_B) \in \mathcal{S}$ the set of feasible buffer occupancy states. Here

$$\mathcal{S} = \{\underline{x}: \ \sum_{i=1}^{B} \mathbf{1}(x_i = k) \le D_k,\ 1 \le k \le K,\ \sum_{k=1}^{K} \sum_{i=1}^{B} \mathbf{1}(x_i = k) = B\}.$$

These probabilities satisfy

$$\pi(\underline{x}) = \sum_{j=1}^{B} \pi(x_2, \cdots, x_j, x_1, x_{j+1}, \cdots, x_B)\alpha_{x_1} + \sum_{l \in \{k: \sum_{i=2}^{B} \mathbf{1}(x_i=k) < D_k\}} \pi(x_2, \cdots, x_B, l)\alpha_{x_1} (D_{x_1} - \sum_{i=2}^{B} \mathbf{1}(x_i = x_1) - \delta_{l,x_1})/D_{x_1}.$$

Here $\delta_{x,y} = 1$ if $x = y$ and 0 otherwise.

King [KING71] has presented an exact solution for this Markov chain which yields $E[Y_k]$. Using a different approach which exploits the stack properties of LRU, Flajolet, et.al. [FLAJ87] have derived a simpler alternate expression for $E[Y_k]$. However, in both cases the computational complexity for evaluating $E[Y_k]$ grows exponentially as a function of B and D. Hence, we present an approximate analysis yielding $E[Y_k]$.

Let $Y_k(j)$ denote whether an item from partition k is stored at the j-th position. It takes value 1 if it is present in position j and 0 otherwise, $Y_k(j) = \mathbf{1}(X_j = k)$. We have $Y_k = \sum_{j=1}^{B} Y_k(j)$, $k = 1, \cdots, K$. Let $p_k(j) = \Pr[X_k(j) = 1]$, $1 \leq k \leq K$, $1 \leq j \leq B$. Clearly, $p_k(1)$ is the probability that the last access was from partition k, and is given by $p_k(1) = \alpha_k$. Let $r_k(j-1)$ be the conditional probability that an item from partition k moves from location $j-1$ to location j in the stack after a request, given that an item moves from location $j-1$ to j. We approximate the steady state probability, $p_k(j) = r_k(j-1)$, $1 \leq k \leq K$, $1 \leq j \leq B$. Let $b_k(j)$ represent the the average number of items from partition k contained in the first j positions of the buffer (the top j positions of the stack is equivalent to considering a buffer of size $j-1$). This is expressed as

$$b_k(j) = \sum_{l=1}^{j} E[X_l(j)] = \sum_{l=1}^{j} p_k(l), \quad j = 1, 2, \cdots, B, \quad (1)$$

and we have

$$p_k(j) = r_k(j-1) = \frac{\left(\alpha_k \left[1 - \frac{b_k(j-1)}{D_k}\right]\right)^+}{r(j-1)}, \quad j = 1, 2, \cdots, B-1;\ k = 1, \cdots, K, \quad (2)$$

where

$$r(j-1) = \sum_{i=1}^{K} \left(\alpha_i \left[1 - \frac{b_i(j-1)}{D_i}\right]\right)^+, \quad j = 1, 2, \cdots, B-1.$$

and $(a)^+$ takes value a whenever $a > 0$ and value 0 otherwise.

Using the above two equations recursively, we can calculate the buffer hit probability of partition k as $h_k = E[Y_k]/D_k = b_k(B)/D_k$.

The approximation in the above expression for $p_k(j)$, uses the average values of the number of granules of partition k in the top $j-1$ locations, rather than the exact distribution. This causes an error for small values of D and B. We have encountered situations where the estimates for the hit probability h_i exceed 1 for some i. To alleviate this, $b_i(j)$ is capped at D_i. This results in a lower value in $p_i(j)$ and the extra probability is redistributed to other partitions proportionally so that $\sum_{l=1}^{K} p_l(j) = 1$. In our iterative algorithm, once $b_i(j)$ reaches the value D_i, for all subsequent steps, l $(l > j)$ partition i is not considered, i.e., $p_i(l) = 0$.

2.2 FIFO Buffer Replacement Policy

We now consider the FIFO buffer replacement policy. Under FIFO, the buffer can be thought of as a queue with position B considered as the head of the queue and position 1 as the tail of the queue. If a request is to an item already in the buffer, then the buffer remains unchanged. If a request is to an item not in the queue, then the item is placed in position 1 and all of the items within the buffer are moved one position. The item at the head of the queue (the oldest item in the buffer) is removed from the buffer.

The following equations describe the behavior of the stationary probability distribution of $\pi(\underline{x})$,

$$\pi(\underline{x}) = \pi(\underline{x}) \sum_{i=1}^{B} x_i/D_{x_i} + \sum_{l \in \{k: \sum_{i=2}^{B} \mathbf{1}(x_i=k) < D_k\}} \pi(x_2, \cdots, x_B, l)\alpha_{x_1} (D_{x_1} - \sum_{i=2}^{B} \mathbf{1}(x_i = x_1) - \delta_{l,x_1})/D_{x_1}, \quad \underline{x} \in \mathcal{S}.$$

King [KING71] has obtained an exact solution to this Markov chain. We present, instead an approximate analysis that yields estimates of $E[Y_k]$.

Let R be the probability that an item is removed from the buffer when a request is served. This is identical to the probability that a new item is brought in, hence

$$R = \sum_{k=1}^{K} \alpha_k (1 - \frac{E[Y_k]}{D_k}). \quad (3)$$

According to flow conservation, the probability that an item from partition k is removed from the buffer equals the probability that an item from partition k is brought in. The probability that an item from partition k resides in any position is $E[Y_k]/B$. Hence the probability

that an item from partition k is removed from the buffer is $RE[Y_i]/B$. The probability that an item from partition k is brought in is $\alpha_k(1 - E[Y_k]/D_k])$. Equating these two probabilities yields

$$\alpha_k(1 - E[Y_k]/D_k) = \frac{E[Y_k]}{B}R. \quad (4)$$

After some algebric manipulation we get

$$E[Y_k] = \frac{D_k}{1 + \frac{RD_k}{\alpha_k B}}. \quad (5)$$

By solving the above set of equations (3) and (5) we get $E[Y_k]$. This can be done in an iterative manner. However, the convergence of the algorithm is sensitive to the way $E[Y_k]$ is adjusted at each iteration step. We have found the following algorithm to work well.

step 1: Intialize $R := 1$; $Ysum := 0$;
step 2: Repeat while $(|Ysum - B| > \delta)$
$E[Y_k] := D_k/(1 + RD_k/\alpha_k B)$;
$k = 1, 2 \cdots K$
$Ysum := \sum_{k=1}^{K} E[Y_k]$
$R := R * Ysum/B$;
step 3: $h_k := E[Y_k]/D_k, k = 1, 2 \cdots K$

3 Validation of the Analysis

In this section, we validate our analytical models against exact results given by King's analysis for a small number of data items ($\leq$ 10) and against results obtained by simulation for larger number of data items. To introduce skewness in the access pattern of the data items, we choose their access probabilities according to truncated arithmetic and truncated geometric distributions [BABA83]. Under the truncated arithmetic probability distribution, the access probability of the i-th item is $p_i = \frac{2}{K(K+1)} * i$. Here, we assume that the access probability of each data granule is unique, i.e, $K = D$. The probability of the i-th item under the truncated geometric distribution is $p_i = \frac{(1-c)}{1-c^K} * c^i$, where c is a constant. The variance in p_i provides a measure of skewness in the access probabilities. It is higher for the truncated geometric distribution than for the truncated arithmetic distribution. In the next section we will give a more formal definition of skewness based on the optimal buffer hit probability. Figures 1 through 4 compare the buffer hit probability of the approximate analysis to that of exact analysis for LRU and FIFO replacement policies. The value of c for the geometric distribution is taken to be 2. The model prediction is optimistic for the LRU policy and pessimistic for the FIFO policy. The error in the approximation for both cases is very small and decreases with the number of data items. It is greater for the truncated geometric probability distribution (the skewness is higher for geometric distribution).

To validate our model for large buffer size, we simulate a skewed data access pattern for 2 and 3 partitions. Figures 5 and 6 show the buffer hit probability of each component as well as the overall buffer hit probability for 2 and 3 partitions case under the LRU policy. As can be observed from these graphs, the match between the approximate analysis and the simulation is excellent. In the case of 2 partitions, the total database size is 1000 items, and 80% of the data accesses go to 20% of the data items (i.e., the partition sizes are 200 and 800, and the corresponding access probabilities are 80% and 20%.). In the case of 3 partitions, the sizes of the partitions are 100, 200 and 400 and the respective probabilities that an access will go to the partitions are 57%, 29% and 14% (truncated geometric distribution). The simulation was run for a long initial period so as to fill up the buffer (20,000 accesses) and then for an additional duration of 20,000 accesses to gather statistics. The results from the simulation and the approximate analysis are found in figures 7 and 8 for both the LRU and the FIFO policies. The match between analysis and simulation is very good for the FIFO policy as well as the LRU policy. In both cases, LRU performs better than FIFO.

4 Applications

To demonstrate the usefulness of the model, we now consider two applications. In our first application, we compare the LRU and FIFO policies to an optimal static buffer allocation policy for a database consisting of two classes of data items, a hot set and a cold set. In our second application, we compare the policy of optimal partitioning of buffer to a policy of shared common buffer for the case of multiple independent reference streams on mutually disjoint data sets competing for the same buffer.

4.1 Database under Skewed Access

In this application we consider the problem of buffer allocation in a database system. We compare the performance of the LRU and FIFO policies with that of an optimal static allocation. We observe that the LRU policy provides most of the performance of the optimal policy. This is of interest because the static allocation requires knowledge of the access probabilities whereas the LRU policy does not. Furthermore, the LRU policy adapts easily to time varying changes in the access probabilities whereas the optimal policy does not.

We begin with a description of the optimal static allocation. Given the K partitions with known sizes and access probabilities, we allocate B_k units of the buffer

to partition k, $1 \leq k \leq K$ so as to maximize the overall buffer hit rate, $H = \sum_{k=1}^{K} \alpha_k B_k / D_k$.

If $\alpha_k/D_k > \alpha_j/D_j$, then reallocating buffer from partition j to partition k increases H. Hence, the optimal solution is obtained in the following manner. Order the K partitions in decreasing value of α_k/D_k. First allocate a buffer of size $\max(B, D_1)$ to partition 1. If $B > D_1$ then allocate a buffer of size $\max(B - D_1, D_2)$ to partition 2. The procedure is continued until all the buffers are allocated. The general solution is given by,

$$B_k = \max\left(0, \min(D_k, B - \sum_{l=1}^{k-1} D_k)\right), \; k = 1, \cdots, K. \tag{6}$$

Consider a database of size D consisting of two partitions where the first partition is of size βD and the second of size $(1-\beta)D$. Let α denote the fraction of accesses made to the first partition. Let $A = (\alpha, \beta)$ define an access pattern to a database. We will order the access patterns in terms of skewness. Two access patterns are different if they differ in one or both the attributes. Let $H(A, B, D)$ represent the buffer hit probability under the optimal buffer allocation policy, where the access pattern, buffer size and database size are given by A, B and D respectively. We say that access pattern A_1 is more skewed than access pattern A_2, if $H(A_1, B, D) \geq H(A_2, B, D)$ for all values of B and D and if $H(A_1, B, D) > H(A_2, B, D)$ for at least one set of values of B and D. Figures 9 through 12 compare the buffer hit probability of LRU and FIFO schemes to that of static optimal allocation scheme, for various degree of skewness and buffer size. The database size, D, in all of these cases is 10,000. The straight lines (dotted lines) correspond to the policy when the buffer is allocated to a particular partition first and then the remaining buffer is allocated to the other partition. As the relative frequency of access to the first partition (α) changes, the optimal buffer hit probability follows the straight line corresponding to that partition, until it is no longer optimal to prefer that partition. The intersection of the straight lines represents that point.

We make the following observations from these figures. Both LRU and FIFO track the optimal allocation policy for all parameters. However, their relative performance depends on the parameters A, B, D. When the buffer size is small ($B = 3,000$), and the data access patterns are highly skewed ($\beta = 0.4$, $\alpha > 0.4$), both LRU and FIFO fail to retain sufficient hot data granules in the buffer. Hence, they perform considerably worse than the optimal allocation (compare figures 9 and 10). This degradation is much less for a larger buffer size (B=8,000, compare figures 11 and 12).

In all cases the LRU policy performs better than the FIFO policy. Consider the case, where both the hot-set and the buffer sizes are small ($\beta = 0.2$ and $B = 3000$). As the LRU retains more hot granules than FIFO, the difference in their performance is significant (figure 9). This difference is less for a larger hot-set size and small buffer size (figure 10). For a larger buffer size ($B = 800$), if the hot-set is small then all three policies retain the hot-set and their performances are close (figure 11). However, for a larger hot-set size, FIFO fails to retain the hot-set even for a large buffer size, and performs worse (figure 12).

Another observation to be made from all these figures is that at the point where no partition is preferred the performance of all three policies are the same. As the hotness migrates, i.e., as α changes, the performance of LRU and FIFO changes very little. However, the performance of the optimal allocation is very sensitive to correct knowledge of the access probabilities, and a large penalty may be incurred if the wrong partition is chosen. This points out the danger of a static allocation in an environment where the access probabilities are not well known.

As observed from the earlier figures, the difference in performance between the LRU and the optimal allocation policies, depends on the relative size of the buffer compared to the database. We next explore the degradation of LRU performance as a function of the database size while keeping the buffer size fixed (figures 13 through 15). As we increase the database size, both of the partition sizes increase proportionally, but their relative sizes remain constant. For a small database, both the LRU and optimal policies retain the hot-set and the difference between their performances is negligible. The first break-point occurs when the LRU policy begins to lose some of the hot granules (figure 13). From this point onwards the performance of LRU degrades more quickly with increasing database size. The second break-point occurs when the optimal policy fails to satisfy the hot-set, i.e., the size of the hot-set is greater than the buffer size (figure 14). From this point onwards, the difference between the performance of the optimal and the LRU policies decreases. Figure 16 shows the percentage degradation in performance of the LRU policy for various access patterns. The degradation reaches a maximum (note, a peak occurs at the second break-point) when the hot-set is small and a significant fraction(say, 50%) of accesses goes to the cold-set. For a system with multiple partitions, several break-points can be observed, each corresponding to a point when one less partition can not be satisfied under the optimal policy. In figure 17, we plot the buffer hit probabilities of the LRU and the optimal policies as well their difference (in same unit), for a three partition case.

4.2 Optimal Buffer Partitioning

We have observed from our first application that the identification of the partitions and the knowledge of their access probabilities are essential for obtaining the optimal buffer allocation. In many applications the exact knowledge of the partitions may be hard to gather without a substantial overhead. However, the partitions may be grouped into identifiable groups. In the database context, this may correspond to multiple database relations (files) sharing a common buffer and access to each relation is skewed dividing each relation into multiple partitions. The exact knowledge of skewness in each relation may not be known. A second example in the context of program execution is the instruction and data set of a program, where the reference streams to instruction and data can be separated and where accesses within the data and instruction sets are skewed.

We observed in the previous subsection that LRU is the policy of choice for buffer management for each relation when nothing is known about the skewness in each relation. However, it is not clear whether a common buffer should be shared by multiple relations under one LRU policy or the buffer should be partitioned in some optimal way among the relations. In the later case, each relation uses its portion of the buffer under its LRU policy. For a real life application, an online adaptive algorithm may be used for the optimal partitioning of the buffer [STON89]. From the analysis point of view, we will assume knowledge of the access pattern in each relation to derive the near optimal partition sizes. We will describe below an algorithm to derive the near optimal partition sizes along with an informal argument for why this should be a near optimal buffer allocation for this problem. In both cases, the performance metric of interest is the overall buffer hit probability.

All of our experimental observations suggest that the buffer hit probablity of the LRU policy under the IRM model is a concave function of the buffer size. This conjecture has been shown to be true by van den Berg [VAN89]. Consequently, the optimal partitioning of the buffer can be formulated as a simple integer convex programming problem which can be solved using a greedy algorithm first proposed by Fox [FOX66]. This algorithm allocates one unit of buffer at each step to the group that will yield the highest incremental change in buffer hit probability. As we do not evaluate the buffer hit probability exactly, we use the approximate analysis of section 2.1. Since the algorithm for obtaining estimates for the hit probability under the LRU policy is recursive and requires a single step for the addition of a buffer unit, it is easily merged with Fox's algorithm.

Figure 18 compares the two buffer policies for a database application with two relations (groups). Each relation consists of 1000 data granules, but the access probabilities within each relation is different. In relation 1, 80% of the accesses go to 20% of the items, and in relation 2, 60% of the accesses go to 40% of the items. The relative frequency of accessing each relation is given by the parameter γ. Increasing γ increases the difference in the access frequency between the most frequently accessed partition (hot partition of relation 1) and the least frequently accessed partition (cold partition of relation 2). For a large buffer size both shared and partitioned policies retain the hot data items of both the relations. But for a small buffer, the performance gain due to partitoning is significant. In all cases, the partitioned policy performs better than the shared policy. This provides evidence that the buffer pool mechanism used in DB2 system [TENG84] is a good one.

5 Summary

In this paper, we have developed approximate analytical models for predicting the buffer hit probability under the Least Recently Used (LRU) and First In First Out (FIFO) buffer replacement policies under the independent reference model. The computational complexity of the analysis of the LRU policy is $O(KB)$ where B is the size of the buffer and K denotes the number of items having distinct access probabilities. In the case of the FIFO policy, the solution algorithm is iterative and the computational complexity of each iteration is $O(K)$. We have compared results from these models to exact results from models originally developed by King [KING71] for small values of the buffer size, B, and the total number of items sharing the buffer, D. In most cases, the error is extremely small (less than 0.1%) for both LRU and FIFO policy, and a maximum error of 3% was observed for very small buffer size(less than 5) and extreme skewness in access probabilities. Results of the approximate models are also compared with the results from simulations for large values of B and D and the match is found to be excellent.

To demonstrate the usefulness of the model, we have considered two applications. In our first application, we compared the LRU and FIFO policies to an optimal static buffer allocation policy for a database consisting of two classes of data items, hot and cold. Both LRU and FIFO track the optimal allocation policy, and the performance of LRU is always better than FIFO. Both LRU and FIFO do not require explicit knowledge of the access frequency of the data items. On the other hand, the optimal allocation policy requires the precise knowledge of the access frequency of all data granules. The penalty in performance for preferring a wrong partition to keep in buffer, is rather large. In the case that the hotness may migrate from one class to an-

other over time, this makes LRU the preferred policy. We have also explored the difference in performance between the optimal policy and the LRU policy as a function of database size, for a fixed buffer size. In our second application, we show that if multiple independent reference streams on mutually disjoint data sets compete for the same buffer, it is better to partition the buffer using an optimal allocation policy than to share a common buffer.

References

[AVEN76] Aven, O.I., L.B. Boguslavsky, and Y. A. Kogan, "Some Results on Distribution-Free Analysis of Paging Algorithms," *IEEE Transactions on Computers*, Vol. C-25, No. 7, pp. 737–745, July 1976.

[BABA83] Babaoglu, O. and D. Ferrari, "Two-Level Replacement Decisions in Paging Stores," *IEEE Transactions on Computers*, Vol. C-32, No. 12, pp. 1151–1159, December 1983.

[COFF73] Coffman, E. G. and P. J. Denning, *Operating Systems Theory*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

[DAN89] Dan, A., D. Dias and P. S. Yu, "Buffer Model under Skewed Data Access for a Data Sharing Environment"(Work in progress).

[FLAJ87] Flajolet, P., D. Gardy and L. Thimonier, "Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-organizing Search", Tech Rep. 720 INRIA, Domaine Voluceau, Rocquencourt, BP 105 78153 Le Chesnay Cedex (France), August 1987.

[FOX66] Fox, B.,"Discrete Optimization via Marginal Analysis," *Management Science*, Vol. 13, pp. 210–216, November 1966.

[FRAN74] Franaszek, P. A. and T. J. Wagner,"Some Distribution-Free Aspects of Paging Algorithm Performance," *Journal of the ACM*, Vol. 21, No. 1, pp. 31–39, January 1974.

[KEAR89] Kearns, J. P. and S. DeFazio, "Diversity in Database Reference Behavior," *Performance Evaluation Review*, Vol. 17, No. 1, pp. 11–19, May 1989.

[KING71] King, W. F., "Analysis of Paging Algorithms," In *Proc. IFIP Congress*, pages 485–490, Ljublanjana, Yugoslavia, aug 1971.

[LANG77] Lang, T., C. Wood, and I. B. Fernandez, "Database Buffer Paging in Virtual Storage Systems," *ACM Transactions on Database Systems*, Vol. 2, No. 4, pp. 339–351, December 1977.

[RAO78] Rao, G. S., "Performance Analysis of Cache Memories," *Journal of the ACM*, Vol. 25, No. 3, pp. 378–395, July 1978.

[SACC87] Sacco, G. M., "Index Access with a Finite Buffer," In *Proc. of 13th VLDB Conference*, pages 301–309, Brighton, 1987.

[SMIT82] Smith, A. J., "Cache Memories," *ACM Computing Surveys*, Vol. 14, No. 3, pp. 473–530, September 1982.

[STON89] Stone, H. S., J. L. Wolf, and J. Turek, *Optimal Partitioning of Cache Memory*, Research Report RC14444 (64697), IBM, March 1989.

[TENG84] Teng, J. Z. and R. A. Gumaer, "Managing IBM Database 2 Buffers to Maximize Performance, "*IBM System Journal*, Vol. 23, No. 2, pp. 211–218, 1984.

[VAN89] J.L. van den Berg, private communication, November 1989.

[VERK85] Verkamo, A. I., "Empirical Results on Locality in Database Referencing," In *Proceedings of the ACM SIGMETRICS Conference on Measurement and modeling of Computer systems*, pages 49–58, 1985.

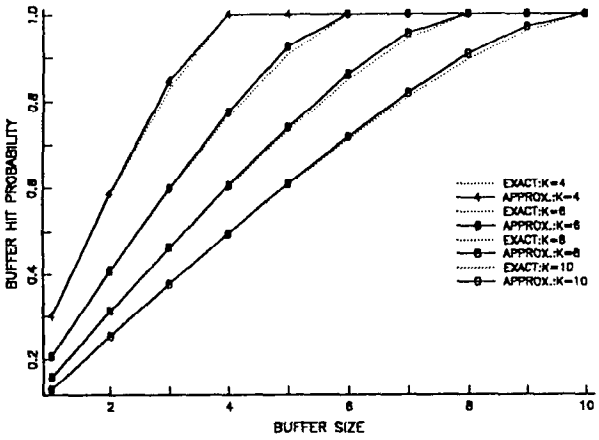

Figure 1: Comparison with King's analysis
(LRU: Arithmetic Probability Distribution)

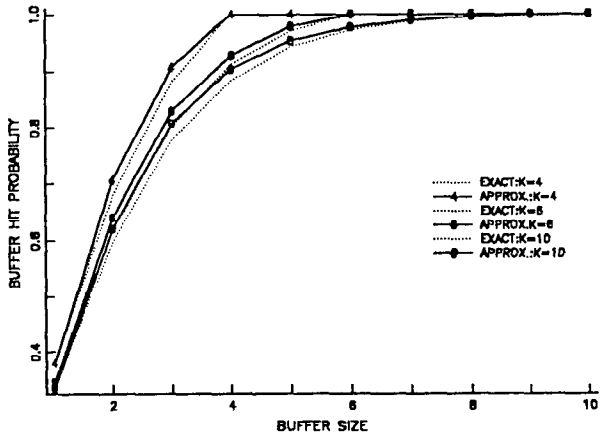

Figure 2: Comparison with King's analysis
(LRU: Geometric Probability Distribution)

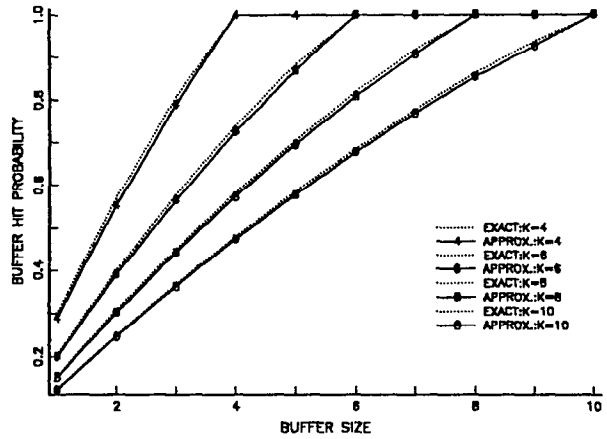

Figure 3: Comparison with King's analysis
(FIFO: Arithmetic Probability Distribution)

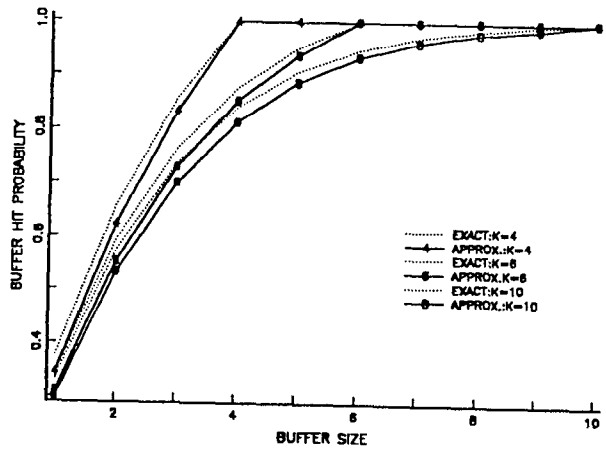

Figure 4: Comparison with King's analysis
(FIFO: Geometric Probability Distribution)

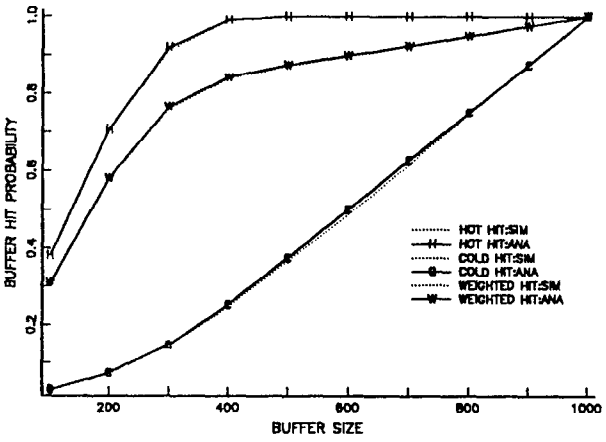

Figure 5: Comparison with simulation (LRU: 2 Partitions)

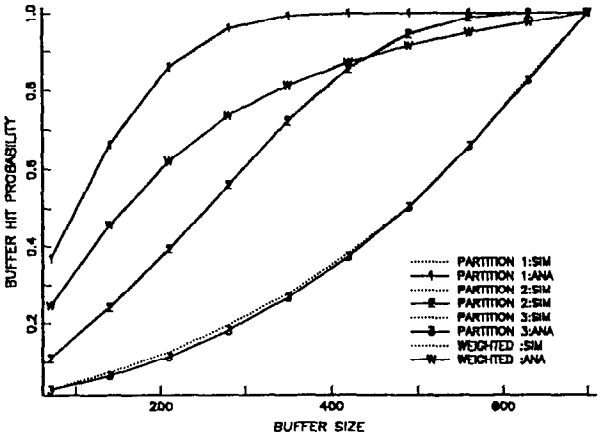

Figure 6: Comparison with simulation (LRU: 3 Partitions)

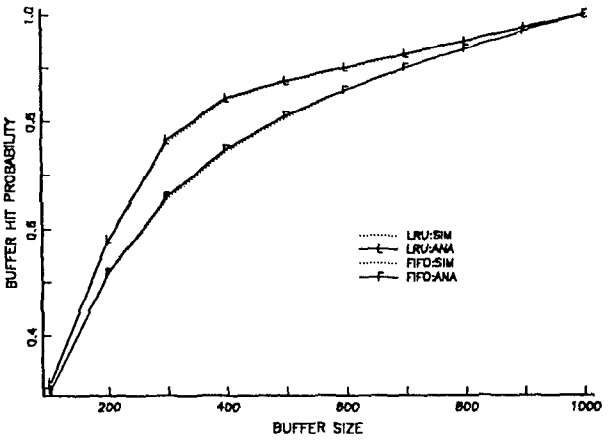

Figure 7: Comparison of LRU and FIFO policies (2 Partitions)

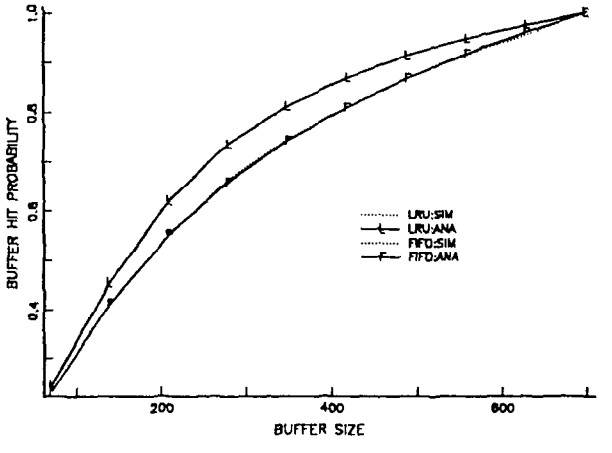

Figure 8: Comparison of LRU and FIFO policies (3 Partitions)

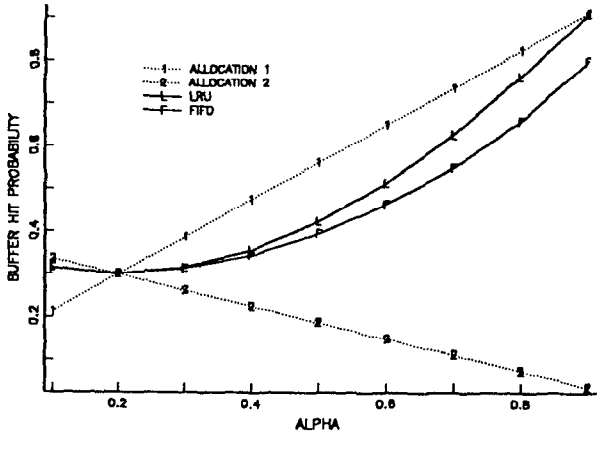

Figure 9: Comparison of Buffer schemes ($B = 3K$; $\beta = 0.2$)

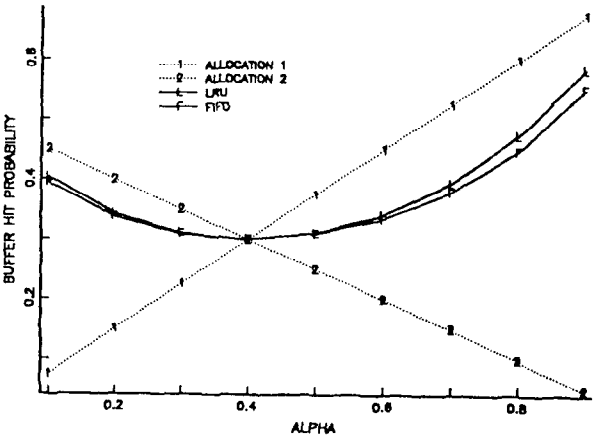

Figure 10: Comparison of Buffer schemes ($B = 3K$; $\beta = 0.4$)

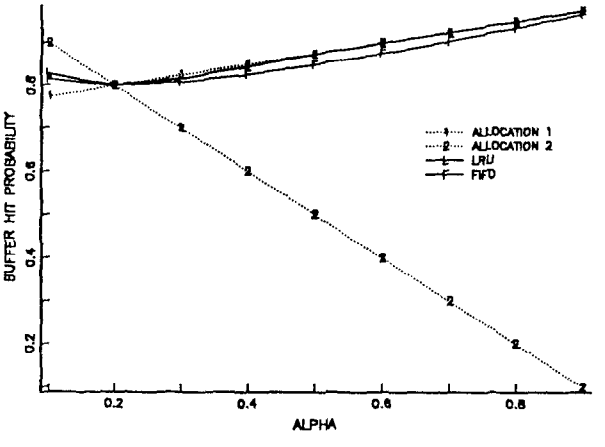

Figure 11: Comparison of Buffer schemes ($B = 8K$; $\beta = 0.2$)

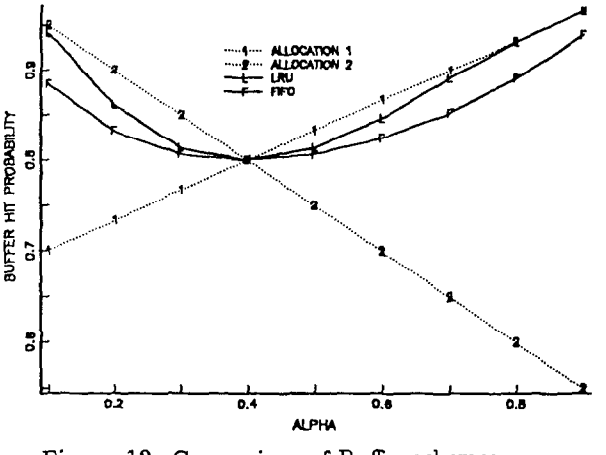

Figure 12: Comparison of Buffer schemes ($B = 8K$; $\beta = 0.4$)

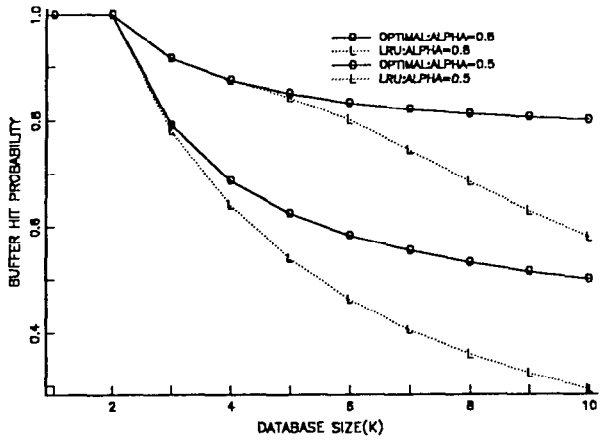

Figure 13: Effect of Database Size ($B = 2K$; $\beta = 0.2$)

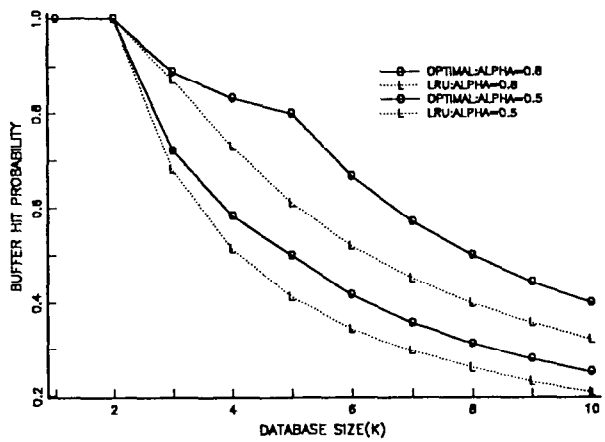

Figure 14: Effect of Database Size ($B = 2K$; $\beta = 0.4$)

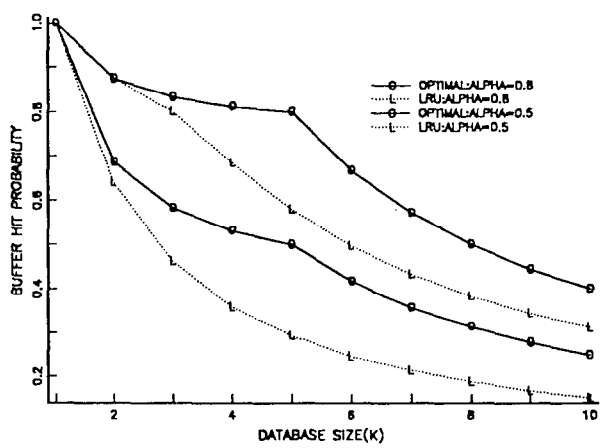

Figure 15: Effect of Database Size ($B = 1K$; $\beta = 0.2$)

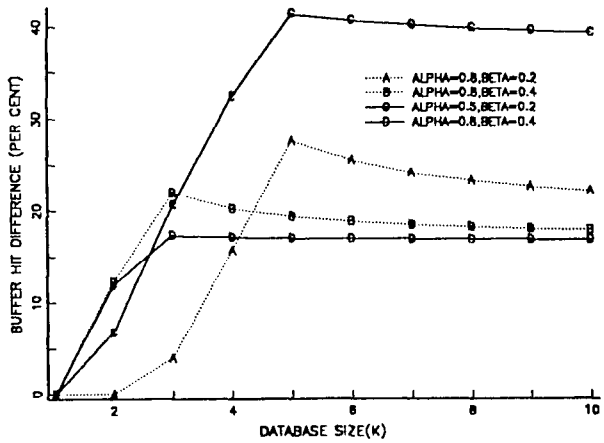

Figure 16: Degradation in Buffer Hit Probability of LRU policy ($B = 1K$)

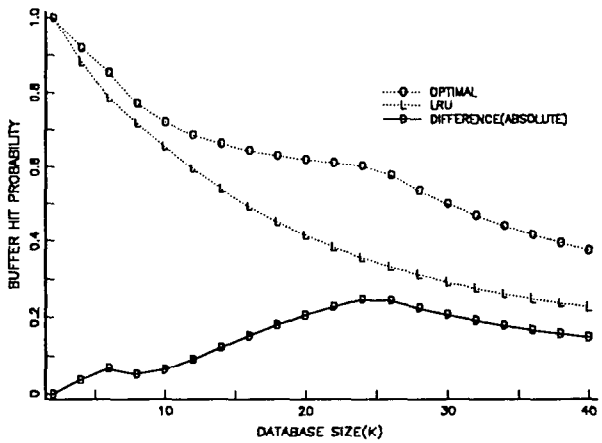

Figure 17: Degradation in LRU Buffer Hit Probability (3 partitions: $B = 1K$; $\alpha : 0.6, 0.3, 0.1$; $\beta : 0.08, 0.3, 0.62$)

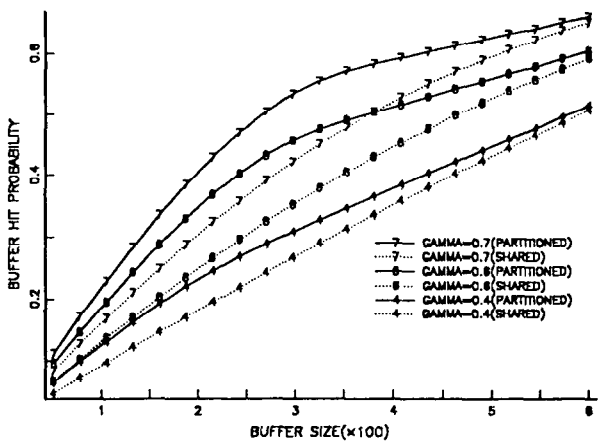

Figure 18: Comparison of Partitioned and Shared Buffer Policies