E. B. Fernández
T. Lang
C. Wood

Effect of Replacement Algorithms on a Paged Buffer Database System

Abstract: In a database system a buffer may be used to hold recently referenced pages. If this buffer is in virtual memory, the database paging system and the memory paging system affect its performance. The study of the effect of main memory replacement algorithms on the number of main memory page faults is the basic objective of this paper. We assume that the buffer replacement algorithm is least recently used (LRU), and page fault rates for LRU, random (R), and generalized least recently used (GLRU) main memory replacement algorithms are calculated and compared. A set of experiments validates these fault rate expressions and establishes some conditions for the practical application of the results.

Introduction

An important aspect of the performance analysis of a data management system (DMS) is the behavior of the buffer where previously referenced pages are held for possible future reference. If the DMS executes in a virtual memory system the buffer is in the virtual address space and, at a given time, some of the buffer pages are also in real storage. Two paging mechanisms participate in the process of accessing a database page, a DMS-controlled buffer paging and a virtual memory paging under the control of the operating system. This structure is of practical importance since it represents the operating environment for IMS and other commercial database systems, and several studies of its behavior have been made. In particular, the effect of the buffer size on performance has been evaluated analytically by using models that incorporate different assumptions [1-3], and also by actual measurement [1, 2, 4, 5].

While the performance of replacement algorithms has been studied analytically for virtual memory systems, for DMS only empirical evaluations have been made [2, 4, 5]. The database virtual buffer environment is different due both to the presence of two memory levels, and to the different characteristics of the reference strings found in these applications.

We analyze here the effect of main memory replacement algorithms on the performance of the buffer system, when the buffer replacement algorithm is LRU (least recently used). In a demand paging environment this performance can be characterized by the number of main memory page faults.

If the reference string is described by the least recently used (LRU) stack model [6], known probabilities of reference can be associated with specific positions of the stack. In non-database applications the stack probabilities usually have a decreasing characteristic with respect to stack distance, and for this case the LRU replacement algorithm has been shown to be optimal [7]. However, for database applications the stack probabilities could exhibit a different distribution, and it is shown in [8] that in general the optimal algorithm belongs to a class of which the LRU algorithm is a particular case. This optimal algorithm is denoted here as *generalized LRU* (GLRU). Another important replacement algorithm is the *random algorithm* (R), where the page to be replaced is selected with a uniform probability. These three algorithms (LRU, GLRU, and R) are considered here since they are of practical importance and possess tractable analytical characteristics.

The next section presents a model of the buffer system and introduces basic concepts. Models are then developed for the three memory replacement algorithms (LRU, R, and GLRU), and expressions are presented for

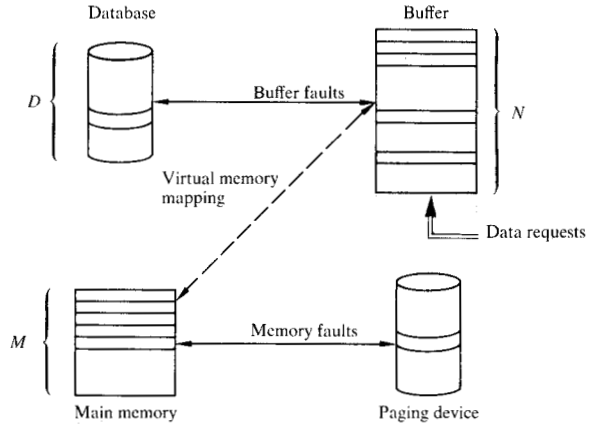


Figure 1 Model of a database buffer system.

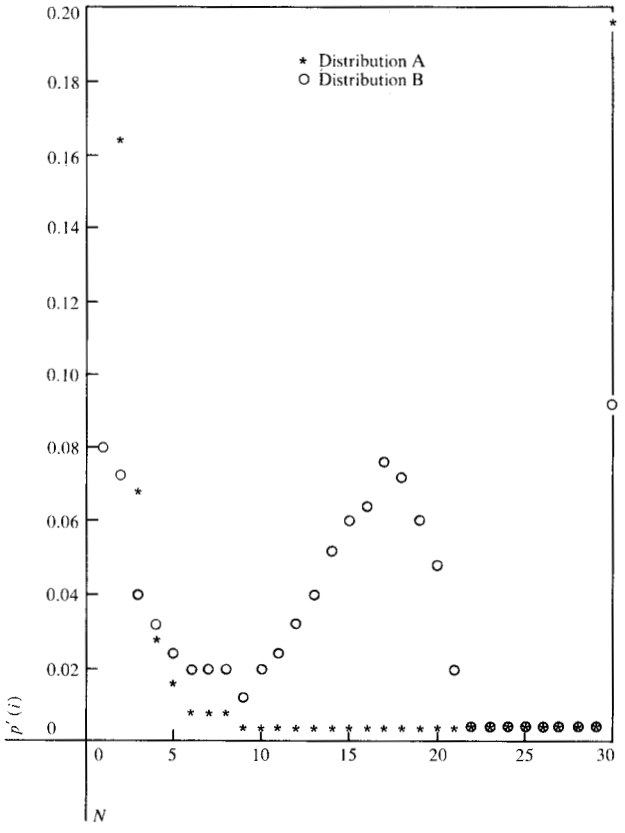


Figure 2 Representative probability distributions.

the expected number of main memory page faults per database reference. The nature of the reference strings in a database application is also discussed and two examples of stack reference probability distributions are given. The page fault rates are calculated for these distributions in order to compare the I/O cost for the different replacement algorithms, and a set of experiments are described to validate these results and evaluate their practical applicability.

Model and basic concepts

Consider a DMS consisting of a database of D pages, a virtual buffer of N pages, and M page frames available for the buffer in main memory (Fig. 1). When a database page is requested, the DMS first determines whether it resides in the virtual buffer. If the page is in the buffer, it is accessed through the virtual memory manager. If the page is not in the buffer, a buffer page has to be replaced by the requested database page, and then accessed through the virtual memory manager. Two paging mechanisms are thus involved in this process. We talk of *buffer faults* when an access to the database is needed because a requested page has not been found in the buffer, and of *memory faults* when an access to the virtual memory paging device is needed. The corresponding replacement algorithms are denoted as *buffer replacement algorithm* and *memory replacement algorithm*. The *cost ratio* r is the ratio between the cost of a database access and the cost of an access to the virtual memory paging device. The cost factor r depends on the specific devices used to store the database, on the paging device for virtual memory, and on the overhead associated with the corresponding access. Because of the smaller size of the virtual memory it is possible to use drums or fixed head disks as paging devices. This results in a practical range for r between 1 and 10. Furthermore, the introduction of high speed secondary memories, with technologies such as charge coupled devices or magnetic bubbles, could result in a larger r. Therefore, this two-level buffer would provide an effective way of taking advantage of a memory hierarchy. The *I/O cost* is the sum of the costs of memory faults and buffer faults. The expected I/O cost, measured in units of database faults, is given by $T = Q + (1/r)F$, where Q is the expected number of buffer faults per data request or the *miss ratio*, and F is the expected number of memory faults per data request (*page fault rate*). Q depends on the database organization, the application program, the size of the virtual buffer, and the buffer replacement algorithm. In multiuser systems, Q also depends on the memory replacement algorithm because memory faults cause task switches and hence changes in the pattern of references. When comparing the performance of memory replacement algorithms we neglect this latter effect, i.e., we assume that Q is empirically determined for one user at a time.

The I/O cost as a function of buffer size was analyzed in [3] for three models which differ in their buffer search methods. For the three models, it is assumed that real pages, virtual pages, and database pages are all the same

size, and that the amount of main memory assigned to the buffer is fixed at M page frames. We are concerned here with one of these models (model C in [3]), where there exists a prefix table in main memory indicating which database pages are in the buffer (i.e., no memory faults for searching the buffer). However, instead of assuming both replacement algorithms to be LRU, as in [3], we assume the buffer replacement algorithm to be LRU and consider several main memory replacement algorithms. Model C corresponds closely to an IMS/VS VSAM system running under a virtual storage operating system.

In the environment of model C, main memory page faults occur in two situations. One is when the requested page is in the buffer, but not in the main memory. In such a case this page has to be transferred to main memory. The corresponding page fault rate is denoted as F_r (*reference paging*). The second is when the requested page is not in the buffer. In this case a buffer page is assigned to it, and this page is transferred to main memory (if not already there) to make the buffer replacement. This is the *double paging* effect, and the corresponding fault rate is called F_d.

Our interest here is in analyzing the effect of buffer size on I/O cost. For a buffer of size equal to or smaller than the available number of main memory pages, F is zero and, since Q is a decreasing function of the buffer size N, we consider only the case $N \geq M$. In [3] it is shown, for fixed M, that there may be a reduction in I/O cost for N beyond M.

In the following section we analyze the page fault rates for the LRU, R and GLRU memory replacement algorithms. Some specific assumptions of the LRU stack model are used to develop expressions for these paging rates.

Page fault rates for the memory replacement algorithms

In order to provide a broader perspective, general expressions for page fault rates for the buffer system are defined. These expressions are then restricted to the case in which the buffer replacement algorithm is LRU, and then F_r and F_d are obtained for GLRU, LRU, and R main memory replacement algorithms.

• *General model of the buffer system*

Let S be the *reference stack* that contains at a given time the database pages previously accessed and ordered according to their time of reference. Its ith position is $S(i)$, with probability of reference $p(i)$, and $S(0)$ contains the most recently referenced page. In the LRU stack model the probabilities are time invariant [6, 7], but this restriction is not required for some of the page fault expressions developed here.

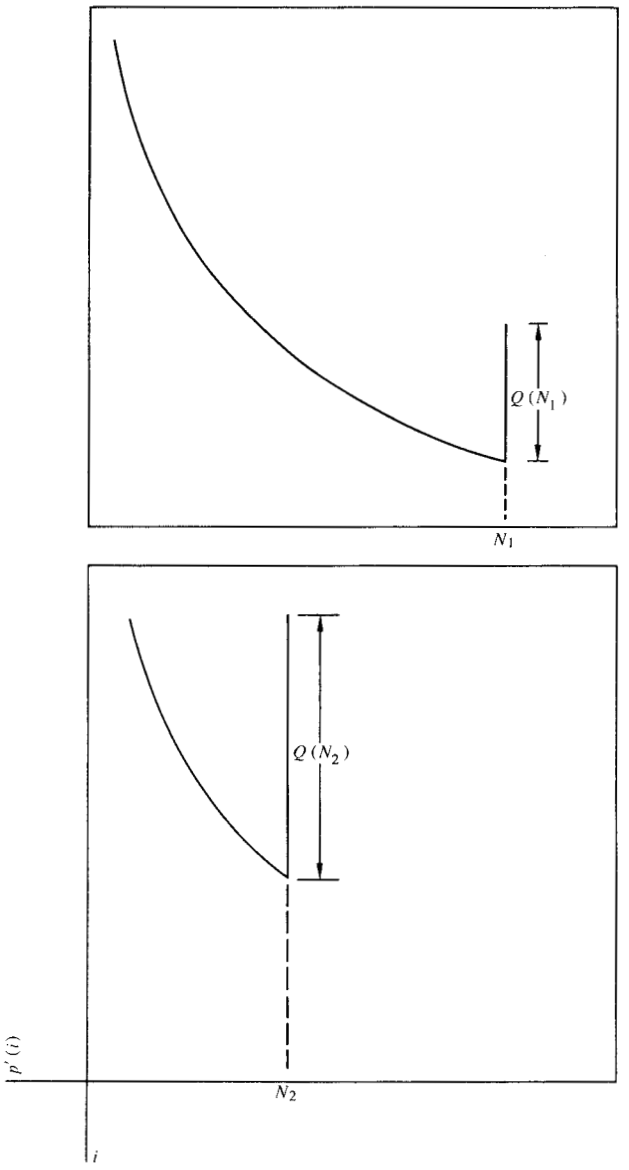


Figure 3 Effect of double paging on probability distributions.

Let B be the buffer, whose jth page is $B(j)$, and the main memory MM, whose kth page is $MM(k)$. In addition to $p(i)$, the following probabilities are associated with this system: $SB(i)$, the probability that $S(i)$ is in B; $BM(j)$, the probability that $B(j)$ is in MM; and $SM(i)$, the probability that $S(i)$ is in MM.

The main memory fault rate is defined by the expressions

$$F_r = \sum_{i=1}^{D} p(i)\,[P\{S(i) \in B \wedge S(i) \notin MM\}],$$

where P indicates probability, and

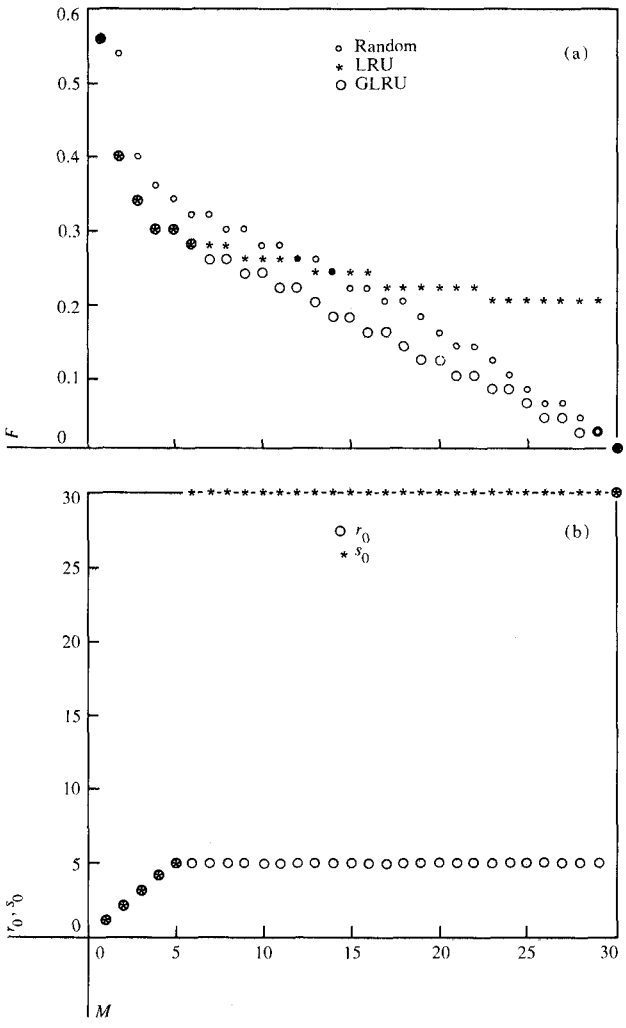


Figure 4 (a) Page fault rates vs M for distribution A and $N = 30$. (b) Optimal r_0 and s_0 vs M for distribution A and $N = 30$.

$$F_d = \sum_{i=1}^{D} p(i)[P\{S(i) \notin B \wedge \text{replaced page of } B \in MM\}].$$

In particular, if the buffer replacement is LRU the N most recently used pages are in the buffer, and

$$B(i) = S(i),\ i \leq N;$$

$$SB(i) = \begin{cases} 1 \text{ for } i \leq N; \\ 0 \text{ for } i > N. \end{cases}$$

Therefore,

$$SM(i) = BM(i),\ i \leq N;$$

and

$$F_r = \sum_{i=1}^{N} p(i)\,[1 - BM(i)]. \tag{1}$$

In the LRU buffer replacement algorithm the Nth position of the buffer is replaced. Since the probability that the Nth position is not in main memory is $1 - BM(N)$, we have

$$F_d = \sum_{i=N+1}^{D} p(i)\,[1 - BM(N)]. \tag{2}$$

It is convenient to define the distribution p', which corresponds to the probabilities of referencing the buffer pages, as

$$p'(i) = \begin{cases} p(i), & i < N; \\ \sum_{i=N}^{D} p(j), & i = N. \end{cases} \tag{3}$$

Then, from (1) and (2), the expression for the total number of page faults when the buffer replacement algorithm is LRU becomes

$$F = F_r + F_d = \sum_{i=1}^{N} p'(i)\,[1 - BM(i)]. \tag{4}$$

In the rest of the paper we assume LRU buffer replacement. We analyze now the effect of different main memory replacement algorithms on F_r and F_d. For convenience we define

$$Q(k) = 1 - \sum_{i=1}^{k} p(i), \tag{5}$$

which corresponds to the probability of not finding a referenced page in the first k pages of the buffer.

- *GLRU algorithm*

In [8] it is shown that for the LRU stack model the optimal BM distribution, which minimizes the number of page faults for a given distribution p', is a member of the class defined by

$$BM(i) = \begin{cases} 1, & 1 \leq i \leq r; \\ (M - r)/(s - r), & r < i \leq s; \\ 0, & s < i \leq N, \end{cases}$$

where $1 \leq r \leq M$ and $M \leq s \leq N$. For the case $r = s = M$, the second term of $BM(i)$ is zero, and this distribution corresponds to that obtained by the LRU algorithm. The optimal values of r and s, r_0 and s_0, depend on the distribution p' and can be found by enumeration techniques.

It is also shown in [8] that the following generalized LRU replacement algorithm (GLRU) provides this distribution and is, therefore, optimal for reference strings that satisfy the LRU stack model:

If $B(i)$ is referenced and is not in memory, *then if* $B(s_0)$ is in main memory and $i > s_0$ *then* replace $B(s_0)$, *else* replace $B(r_0)$ [$B(r_0)$ is always in main memory].

The reference and double paging rates in this case are

$$F_r = \sum_{i=r_0+1}^{s_0} p(i)\,[(s_0 - M)/(s_0 - r_0)] + \sum_{i=s_0+1}^{N} p(i)$$

$$= [Q(r_0) - Q(s_0)]\,[(s_0 - M)/(s_0 - r_0)] + Q(s_0) - Q(N), \quad (6)$$

and

$$F_d = \begin{cases} Q(N), & s_0 < N; \\ Q(N)\,[(s_0 - M)/(s_0 - r_0)], & s_0 = N. \end{cases} \quad (7)$$

These expressions are still valid if, instead of having time-invariant probabilities $p(i)$, the probabilities vary and their averages over a reference string are used. In this case, while this algorithm is still the optimal for its class, there may be another type of algorithm which is optimal overall. When applying this algorithm, it is possible to calculate r_0 and s_0 in a portion of the reference string and use these parameters in other parts of this string. The performance of the algorithm in those other strings depends only on the stability of r_0 and s_0. These requirements are considerably less restrictive than having time-invariant probabilities, and therefore these expressions (and the algorithm) are applicable to a wider range of practical situations.

• *LRU memory replacement algorithm*

For the LRU memory replacement algorithm (a special member of the previous class) the M most recently used buffer pages are in main memory, i.e.,

$$BM(i) = \begin{cases} 1, & 1 \le i \le M; \\ 0, & M < i \le N. \end{cases}$$

Hence, from (1), (2), and (4),

$$F_r = \sum_{i=M+1}^{N} p(i) = Q(M) - Q(N), \quad (8)$$

$$F_d = \sum_{i=N+1}^{D} p(i) = Q(N), \quad (9)$$

and

$$F = \sum_{i=M+1}^{N} p'(i) = Q(M). \quad (10)$$

As for the general class, we emphasize that these expressions are still valid if average values over a reference string are used for the stack probabilities.

• *Random memory replacement algorithm*

For the purpose of comparison, we now consider the random replacement algorithm, which is also of practical importance [2, 4]. In this algorithm the page to be replaced is selected with a uniform probability of $1/M$. We determine the effect of a reference at time t on the probabilities BM at time $t + 1$.

First we define the following events:

$a(i, t)$: the access to the buffer at time t is to page i;
$b(k, t)$: the kth page in the buffer is in main memory at time t [$\approx b(k, t)$ denotes the complement of $b(k, t)$].

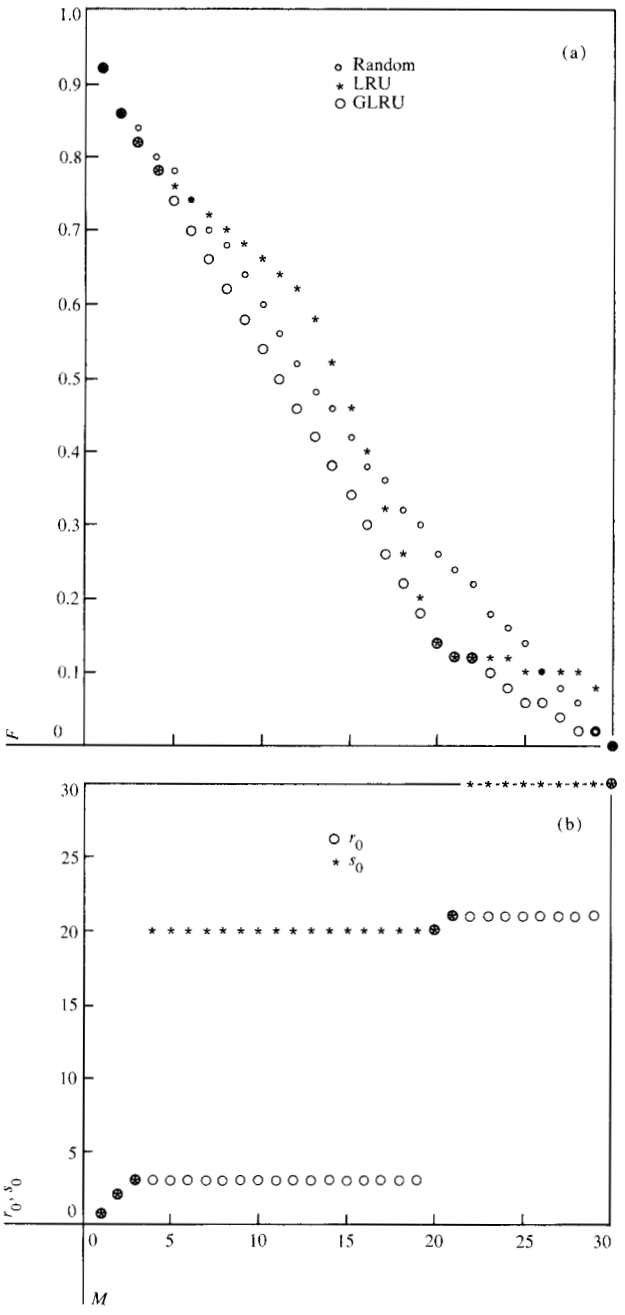


Figure 5 (a) Page fault rates vs M for distribution B and $N = 30$. (b) Optimal r_0 and s_0 vs M for distribution B and $N = 30$.

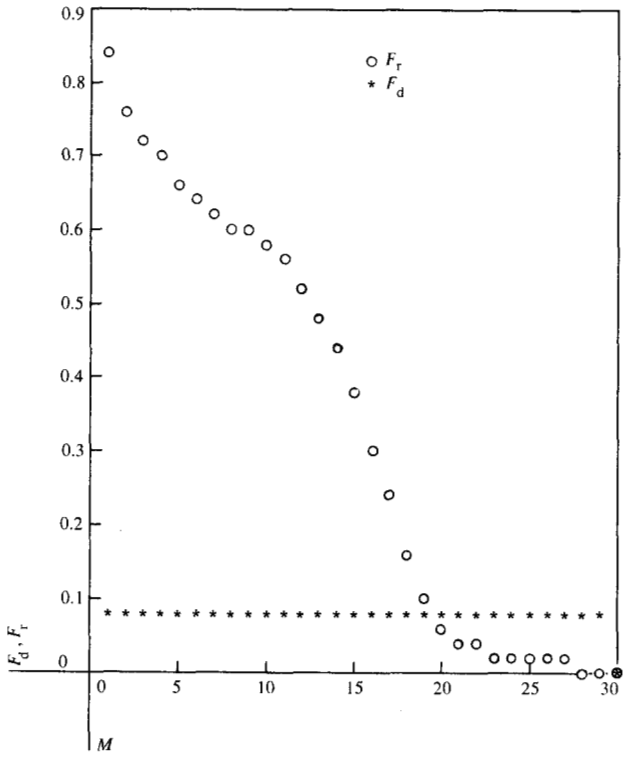
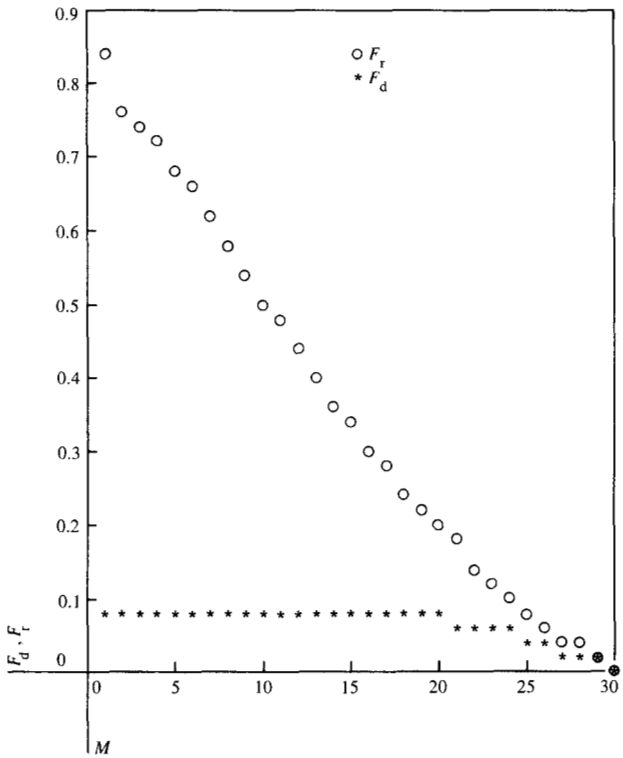


Figure 6 F_d and F_r vs M for LRU (distribution B and $N = 30$).

Figure 7 F_d and F_r vs M for random algorithm R (distribution B and $N = 30$).



Noting that the kth page in the buffer at time $t + 1$ may have been in either position k or $k - 1$ in the buffer at time t, depending on the reference at time t, we have:

$$
\begin{aligned}
P\{b(k, t+1)\} = \sum_{i=1}^{k-1} & [P\{b(k, t) \mid a(i, t) \wedge b(i, t)\} \\
& \times P\{a(i, t)\} \times P\{b(i, t)\} \\
& + P\{b(k, t) \mid a(i, t) \wedge \approx b(i, t)\} \\
& \times P\{a(i, t)\} \times P\{\approx b(i, t)\} \times (1 - 1/M)] \\
& + \sum_{i=k}^{N} [P\{b(k-1, t) \mid a(i, t) \wedge b(i, t)\} \\
& \times P\{a(i, t)\} \times P\{b(i, t)\} \\
& + P\{b(k-1, t) \mid a(i, t) \wedge \approx b(i, t)\} \\
& \times P\{a(i, t)\} \times P\{\approx b(i, t)\} \\
& \times (1 - 1/M)], \qquad (11)
\end{aligned}
$$

where $(1 - 1/M)$ is the probability of not replacing the predecessor of page k when the referenced page is not in the main memory (the predecessor of page k could be page k or page $k - 1$, depending on the stack distance of the reference). In equilibrium the probabilities are independent of t and thus the subscript t can be dropped. It can be shown that the conditional probability $P\{b(k) \mid a(i) \wedge b(i)\}$ is independent of $a(i)$. By definition, $P\{b(k)\} \triangleq BM(k)$, and $P\{a(i)\} = p'(i)$. Then, (11) becomes

$$
\begin{aligned}
BM(k) = \sum_{i=1}^{k-1} & p'(i)[P\{b(k) \mid b(i)\} \times BM(i) \\
& + P\{b(k) \mid \approx b(i)\} \times (1 - BM(i)) \times (1 - 1/M)] \\
& + \sum_{i=k}^{N} p'(i) [P\{b(k-1) \mid b(i)\} \times BM(i) \\
& + P\{b(k-1) \mid \approx b(i)\} \times (1 - BM(i)) \\
& \times (1 - 1/M)]. \qquad (12)
\end{aligned}
$$

The conditional probabilities can be written as

$$P\{b(i) \mid b(i)\} = P\{b(1) \mid \approx b(i)\} = 1,$$

$$P\{b(k) \mid b(i)\} = BM(k) \times w_k(i), \quad k > 1, \text{ and}$$

$$P\{b(k) \mid \approx b(i)\} = BM(k) \times w'_k(i), \quad k > 1,$$

where $w_k(i)$ and $w'_k(i)$ are defined below. Also, the following conditions must be satisfied:

$$\sum_{k=2}^{N} P\{b(k) \mid b(i)\} = \sum_{k=2}^{N} P\{b(k) \mid \approx b(i)\} = M - 1,$$

$$P\{b(i) \mid b(i)\} = 1,$$

and

$$P\{b(i) \mid \approx b(i)\} = 0.$$

As an approximation we assume that $w_k(i)$ and $w'_k(i)$ are independent of k for $k \neq i$. Consequently,

$$w_k(i) = \begin{cases} 1, & i = 1; \\ \dfrac{M - 2}{M - BM(i) - 1}, & 1 < i \leq N \text{ and } i \neq k; \\ 1/BM(i), & i = k, \end{cases}$$

and

$$w'_k(i) = \begin{cases} 0, & i = 1; \\ \dfrac{M - 1}{M - BM(i) - 1}, & 1 < i \leq N \text{ and } i \neq k; \\ 0, & i = k. \end{cases}$$

The conditional probabilities do in fact satisfy the expression

$$BM(k) = P\{b(k) \mid b(i)\} \times BM(i) + P\{b(k) \mid \approx b(i)\} \times (1 - BM(i)).$$

Substituting for the conditional probabilities in (12) we have

$$BM(1) = 1,$$

$$BM(2) = \left\{ \sum_{i=2}^{N} p'(i)[BM(i) + M - 1] \right\} \Big/ \{M[1 - p'(1)]\},$$

$$BM(k) = \left[BM(k-1) \times \sum_{i=k}^{N} \frac{(M-1)^2 - BM(i)}{M - 1 - BM(i)} \times \frac{p'(i)}{M} \right] \div \left[1 - \sum_{i=1}^{k-1} \frac{(M-1)^2 - BM(i)}{M - 1 - BM(i)} \times \frac{p'(i)}{M} \right],$$

for $3 \leq k \leq N$.

The $BM(k)$ can now be found by numerical iteration, and from these, the page fault rates, by using expressions (1), (2), and (4). For the examples analyzed, the solutions for $BM(k)$ converged rapidly.

Characteristics of the reference stack

In order to provide a framework for comparing replacement algorithms, two probability distributions have been selected. Distributions A and B, illustrated in Fig. 2, have been obtained from the experimental miss ratios obtained by Tuel [1], and Sherman and Brice [2], respectively.

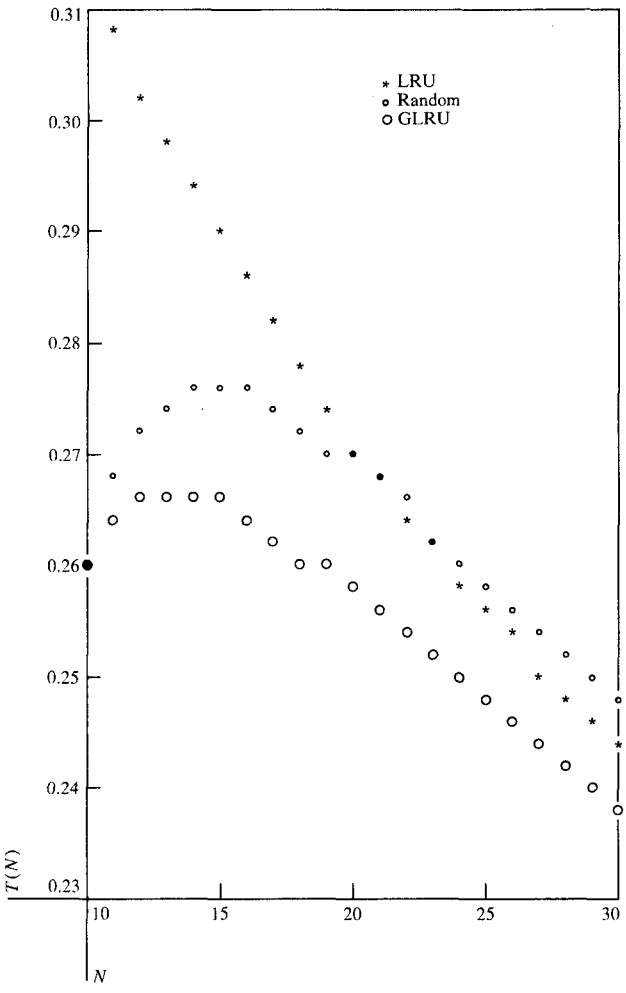


Figure 8 I/O cost vs N for distribution A, $r = 5$, $M = 10$.

(These miss ratios correspond to averages over a reference string.) The modified distributions p' [see Eq. (3)] are presented for $N = 30$. Distribution A (without the peak at $i = N$) is representative of the reference strings of some database applications, and also corresponds to the characteristics exhibited by most non-database programs [6]. Distribution B is representative of some database applications.

While non-increasing distributions seem to adequately represent many user and system programs, many database applications cannot be well characterized by them. There are several reasons for this.

1. The double paging effect can be represented by a peak at $i = N$ in the distribution p' as defined by (3) (Fig. 3).
2. A set of several transactions could produce a distribution such as B, because the pages of a given transaction are pushed down the stack by the other transactions before being rereferenced.

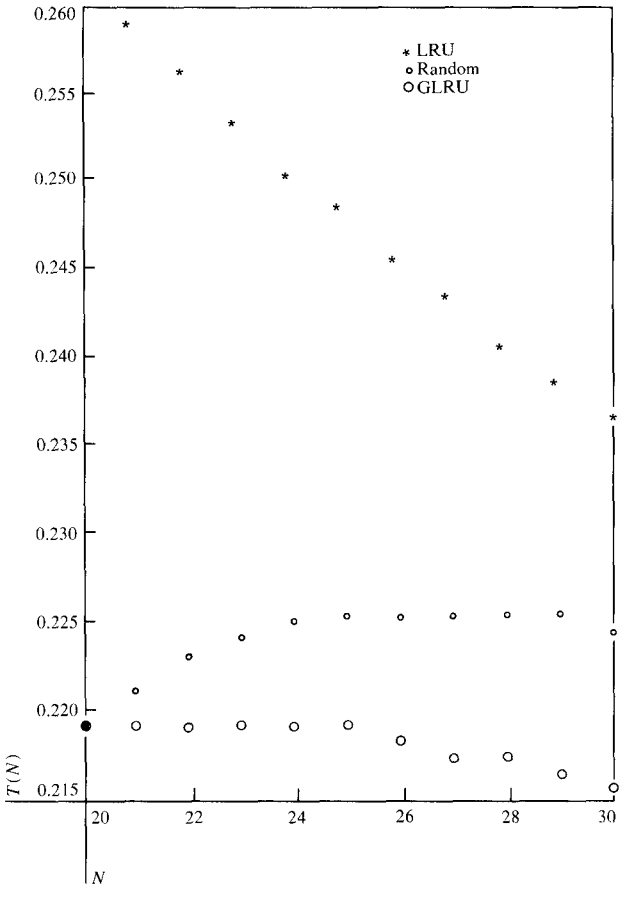


Figure 9 I/O cost vs N for distribution A, $r = 5$, $M = 20$.

3. A set of several transactions sharing common records (e.g., tables for indirect addressing of records), could give distributions such as B since these records are reused when they are in different stack positions [9].
4. The probability distributions of Spirn and Denning were obtained for programs where the essential behavior is dictated by instruction (and not data) referencing, which is normally very sequential, and where data requests are mainly satisfied from CPU registers rather than from memory. On the other hand, for database systems we are only concerned with data references to the virtual buffer.

For these reasons, the LRU algorithm may not be optimal in a database environment described by the LRU stack model. It is therefore of practical importance to compare the performances of different algorithms.

Comparison of replacement algorithms

The performances of the GLRU, random (R), and LRU algorithms are now compared, where the difference between the respective page fault rates is used as a means of comparison.

- *LRU and GLRU*

From (6), (7), and (10) we have

$$F_{\text{LRU}} - F_{\text{GLRU}} = \sum_{i=M+1}^{N} p'(i) - \sum_{i=r_0+1}^{s_0} p'(i)\left(\frac{s_0 - M}{s_0 - r}\right) - \sum_{i=s_0+1}^{N} p'(i)$$

$$= \sum_{i=M+1}^{s_0} p'(i)\left(\frac{M - r_0}{s_0 - r_0}\right) - \sum_{i=r_0+1}^{M} p'(i)\left(\frac{s_0 - M}{s_0 - r_0}\right)$$

$$= \{[(M - r_0)(s_0 - M)]/(s_0 - r_0)\}\,[\bar{p}'(M + 1, s_0) - \bar{p}'(r_0 + 1, M)],$$

where $\bar{p}'(a, b)$ is the average of $p'(i)$ between $i = a$ and $i = b$. This difference depends on M and the distribution p'. It is zero when

$$\min_{i \le r_0 < M} \bar{p}'(r_0 + 1, M) \le \max_{M < s_0 \le N} \bar{p}'(M + 1, s_0),$$

which is the condition for LRU to be optimal [7]. $F_{\text{LRU}} - F_{\text{GLRU}}$ is a maximum when $p(i) = 0$ for $i \le M$ and $p(M + 1) = 1$ (worst case for LRU). In this case, $r_0 = 1$ and $s_0 = M + 1$, resulting in $(F_{\text{LRU}} - F)_{\max} = (1 - 1/M)$. For example, for $M = 10$, $(F_{\text{LRU}} - F_{\text{GLRU}})_{\max} = 0.9$.

- *Random and GLRU*

Although the random replacement algorithm can never be optimal [i.e. $(F_{\text{RAND}} - F_{\text{GLRU}}) > 0$], its worst case is never as bad as the LRU worst case because max F_{RAND} is never 1 [i.e., $(F_{\text{RAND}} - F_{\text{GLRU}})_{\max} < (F_{\text{LRU}}) - F_{\text{GLRU}})_{\max}$]. As an example, for $M = 10$, $(F_{\text{RAND}} - F_{\text{GLRU}})_{\max} = 0.278$.

- *Random and LRU*

$F_{\text{LRU}} - F_{\text{RAND}}$ is a maximum for the same reference distribution p' as the one that gives a maximum for $(F_{\text{LRU}} - F_{\text{OPT}})$, namely: $p'(i) = 0$ for $i \ne M + 1$ and $p'(M + 1) = 1$. For the case when $M = 10$, $(F_{\text{LRU}} - F_{\text{RAND}})_{\max} = 0.82$. Again for $M = 10$, $F_{\text{RAND}} - F_{\text{LRU}}$ can be shown to be a maximum for the following distribution: $p'(i) = 0$ for $i \ne 10$ and $i < N$, $p'(10) = 0.8$, and $p'(N) = 0.2$ $(N >> M)$. In this case, $(F_{\text{RAND}} - F_{\text{LRU}})_{\max} \approx 0.31$.

- *Effect of main memory size*

In general, LRU is better than R when main memory is large enough to contain the locality of the database references $[Q(M) - Q(N) << 1]$, and double paging is not significant $[Q(N) << 1]$. For M close to N $(M < N)$, and for significant values of $Q(N)$, R is better than LRU because, as M increases, the double paging rate remains constant for LRU while it decreases for R. Since the reference pag-

ing decreases with increasing M for both LRU and R, the double paging effect eventually dominates for LRU.

• *Graphic comparison*

The I/O costs for the three replacement algorithms are now compared for the distributions A and B. Because the values of Q used to obtain these distributions are averages over reference strings, the costs obtained are exact for the GLRU and LRU algorithms, while they are only approximated for the random algorithm. (The costs for this case depend on the time-invariance of the probabilities of the stack model.)

The I/O costs depend on M, N, and r. First, we compare the total cost for varying M. Since N will be fixed for this comparison, $Q(N)$ is constant and we can concentrate on the page fault rates. Figures 4(a) and 5(a) show the performance of the three algorithms as a function of M for distributions A and B, respectively, and for $N = 30$. The corresponding values of r_0 and s_0 are also given [Figs. 4(b) and 5(b)]. Figure 6 shows F_r and F_d for the LRU algorithm applied to distribution B. Figure 7 does the same for the random algorithm.

From Figs. 4(a) and 5(a), it can be seen that in the region where M approaches N, LRU does poorly because of the dominance of the double paging effect (Fig. 6). When M is close to N the random algorithm is close to the GLRU algorithm [Figs. 4(a) and 5(a)], since the probability of having a page in main memory increases with M, and double paging is not significant in this range for this algorithm (Fig. 7). In the region between $M = 6$ and $M = 15$ the random algorithm does better than LRU [Fig. 5(a)], due to the high reference paging needed by LRU in this case (Fig. 6). For distribution A, LRU is better than R for $M \leq 13$ [Fig. 4(a)]. For distribution A, LRU is optimal for $M \leq 6$, while for distribution B, LRU is optimal in the ranges $1 \leq M \leq 3$ and $20 \leq M \leq 21$.

We now consider the total cost for varying N. Figures 8, 9, and 10 compare T (expected I/O cost) as a function of N for the three replacement algorithms, the two distributions A and B, a suitable value of r, and two values of M.

For distribution A, a value of $r = 5$ has been used to obtain a decrease in cost. For $M = 10$ (Fig. 8), the GLRU algorithm produces a decrease in cost with respect to the case for which $N = M$ for $N > 20$. Because $r = 5$, the difference in I/O cost between the algorithms is not very significant. For $M = 20$ (Fig. 9), in the range of N considered, only the GLRU algorithm gives a decrease in cost.

For distribution B, $r = 2$ has been selected. (A lower r is sufficient here since this distribution has a steeper slope at $M = 10$ than does distribution A.) It can be seen from Fig. 10 that with this value of r, a decrease in cost (with respect to the case $N = M$) is obtained for values of $N >$ 15. In the range of values of N considered, the GLRU algorithm is significantly better than both R and LRU, with LRU being the worst.

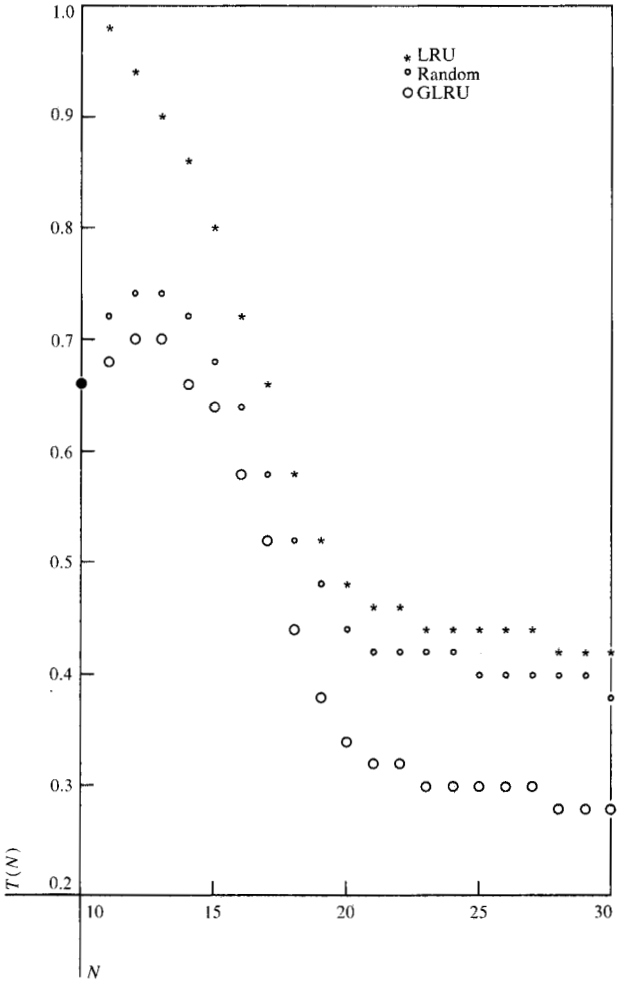


Figure 10 I/O cost vs N for distribution B, $r = 2$, $M = 10$.

Experimental results

A series of measurements was performed by using a section of the reference string described in [10] and [11]. This section corresponds to the operation of a hierarchical database system (IBM's IMS), with 6-segment types. The objectives of these experiments were to compare the fault rates obtained from the model developed here with measurement on a real reference string; and to determine the applicability of the GLRU algorithm in a real environment. Tables 1, 2, and 3 summarize the results for the model validation. Table 1 indicates the stack probabilities for a series of 10 000 record references. Tables 2 and 3 compare the paging rates obtained with the model by using the stack probabilities of Table 1 and the actual measurement on the reference string.

Table 1 Stack probabilities of reference string.

Stack position	*Probability*
1	0.9106
2	0.0313
3	0.0008
4	0.0023
5	0.0054
6	0.0012
7	0.0077
8	0.0029
9	0.0081
10	0.0088
Remainder	0.0128

Table 2 Validation of the model, fixed $M = 4$ and variable N.

Algorithm	N	r_0	s_0	*Model*			*Measurement*		
				F_r	F_d	F	F_r	F_d	F
GLRU	5	2	5	28	165	194	25	167	192
	6	2	6	48	242	290	44	242	286
	7	2	7	104	244	349	102	254	356
	8	2	8	135	252	387	127	254	381
	9	2	9	203	212	415	202	211	413
	10	2	10	279	157	436	283	158	441
LRU	5			54	496	550	Same as model		
	6			66	484	550			
	7			143	407	550			
	8			172	378	550			
	9			253	297	550			
	10			341	209	550			
R	5			54	205	259	32	197	229
	6			83	311	394	45	297	342
	7			154	316	470	110	316	426
	8			188	326	514	146	319	465
	9			267	272	539	221	264	485
	10			354	199	553	300	198	498

Table 3 Validation of the model, fixed $N = 10$ and variable M.

Algorithm	M	r_0	s_0	*Model*			*Measurement*		
				F_r	F_d	F	F_r	F_d	F
GLRU	2	2	10	372	209	581	372	208	580
	3	2	10	325	183	508	331	181	512
	4	2	10	279	156	435	283	158	441
	5	2	10	232	131	363	245	130	375
	6	2	10	186	104	290	206	104	310
	7	2	10	140	78	218	152	77	229
	8	2	10	93	52	145	101	52	153
	9	2	10	46	26	72	46	25	71
LRU	2			372	208	580	Same as model		
	3			364	208	572			
	4			341	208	549			
	5			287	208	495			
	6			275	208	483			
	7			198	208	406			
	8			169	208	377			
	9			88	208	296			
R	3			413	207	620	333	206	539
	4			353	199	552	292	192	484
	5			297	182	479	245	180	425
	6			239	158	397	202	148	350
	7			181	127	308	150	122	272
	8			122	89	211	95	85	180
	9			61	47	108	54	49	103

For the LRU algorithm, experiment and model give the same values since the fault rates in this case depend only on the measured average probabilities. In addition, examination of expressions (6) and (7) shows that for the GLRU algorithms the paging rates depend again on the average probabilities (for a fixed pair of r and s values). Agreement between model and measurement is very good because of this reason, the small discrepancies being due to the fact that the statistical nature of the expressions results in exact values only for very long sequences of references. Finally, there is good agreement in the results for the random algorithm. The differences are due to the stronger dependency on the LRU stack model in this case, and also to the approximate solution of the expressions.

As indicated previously, the performance of the GLRU algorithm depends on the stability of r_0 and s_0 along the reference string. Even in the presence of variations this performance can be adequate if the variation of r_0 and s_0 has little effect on the paging rates. Table 4 indicates the stack probabilities for seven reference sequences of 5000 records each. Their averages over this complete string (35 000 references) are also given. Table 5 shows the variation of r_0 and s_0 as a function of M for $N = 10$, for the seven reference strings and for the complete string. It can be seen that these values are relatively stable. The value $s_0 = N$ is characteristic when the stack probabilities are rapidly decreasing and the double paging peak predominates.

Brice and Sherman [4] have performed a series of experiments, some of which are relevant to this study. In particular, they report on double paging rates and reference paging rates for LRU buffer replacement and random memory replacement. Table 6 compares their results with those obtained from our model for that particular case. It can be seen that there is a reasonable agreement. (These results are approximate because they depend on the exact form of Q, which was interpolated from only five points.)

Conclusions

Studies of the effect of replacement algorithms on the performance of database systems that use a paged buffer are of practical importance since many commercial data management systems use buffers to improve performance. Analyses of the effect of varying both the buffer and the main memory replacement algorithms are very complex and simulation studies may be needed. However, if the buffer replacement algorithm is LRU, at least some main memory replacement algorithms can be analyzed. Three main memory replacement algorithms (LRU, R, and GLRU) have been considered in this paper. Expressions for reference paging rate and for double paging rate were developed for these three algorithms, and their relative performances compared. The expressions for the LRU

Table 4 Stack probabilities of four reference strings for study of r_0 and s_0.

Stack position	*String 1*	*String 2*	*String 3*	*String 4*	*String 5*	*String 6*	*String 7*	*Complete string*
1	0.8512	0.9150	0.9210	0.9293	0.9294	0.9175	0.9267	0.9399
2	0.1165	0.0278	0.0004	0.0004	0.0000	0.0053	0.0059	0.0088
3	0.0000	0.0011	0.0000	0.0000	0.0000	0.0055	0.0029	0.0034
4	0.0011	0.0029	0.0000	0.0000	0.0000	0.0072	0.0019	0.0037
5	0.0011	0.0064	0.0034	0.0021	0.0000	0.0080	0.0026	0.0038
6	0.0000	0.0016	0.0000	0.0000	0.0039	0.0161	0.0034	0.0024
7	0.0000	0.0073	0.0086	0.0004	0.0008	0.0044	0.0042	0.0034
8	0.0000	0.0012	0.0207	0.0152	0.0000	0.0008	0.0047	0.0006
9	0.0000	0.0107	0.0000	0.0041	0.0042	0.0069	0.0054	0.0021
10	0.0022	0.0090	0.0030	0.0045	0.0000	0.0055	0.0081	0.0023
Remainder	0.0277	0.0168	0.0429	0.0440	0.0617	0.0225	0.0341	0.0294

and GLRU algorithms depend on stack probabilities averaged over reference strings, but the expressions for the random algorithm (R) depend on the time invariance of these probabilities. While the GLRU is always optimal within its class, it is also globally optimal if the stack probabilities are time invariant.

Some conclusions that can be drawn from this comparison are:

1. For fixed memory size M, LRU can be optimal in some cases, but it is particularly bad for values of buffer size N just larger than M. On the other hand, the random replacement algorithm is never optimal, but its worst case behavior is not as bad as the worst case behavior for the LRU algorithm.
2. When we plot a typical distribution p' (Fig. 11), it is possible to distinguish three regions according to the behavior of the main memory replacement algorithms. In region 1, R may be better than LRU when the locality of the database cannot be contained in this memory size. In region 2, LRU is optimal since the probabilities satisfy the optimality condition [8]. In region 3, R is again better than LRU because of the double paging effect.
3. For a required page-fault rate a tradeoff is possible between the memory size M and the complexity of the replacement algorithm; for example, in Fig. 5(a) for $F = 0.4$ the GLRU algorithm requires $M = 13$, and the LRU, $M = 16$.
4. The relative I/O cost of these three algorithms depends M, N, r, and p. Since the implementation of each of the algorithms has different complexity, to select an algorithm for a specific situation an analysis similar to the one indicated here should be performed, in order to see whether the improvement justifies the additional complexity.

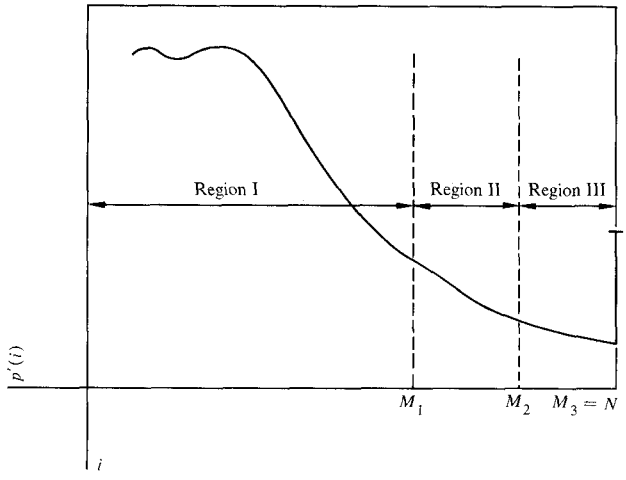
Table 5 Variation of r_0 and s_0 with M for $N = 10$ and for seven reference strings.

String		*M*							
		2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	r_0	2	2	2	2	2	2	2	2
	s_0	2	10	10	10	10	10	10	10
2	r_0	2	2	2	2	2	2	2	2
	s_0	2	10	10	10	10	10	10	10
3	r_0	1	1	1	1	1	1	1	1
	s_0	10	10	10	10	10	10	10	10
4	r_0	1	1	1	1	1	1	1	1
	s_0	10	10	10	10	10	10	10	10
5	r_0	1	1	1	1	1	1	1	1
	s_0	10	10	10	10	10	10	10	10
6	r_0	1	1	1	1	1	1	1	1
	s_0	10	10	10	10	10	10	10	10
7	r_0	1	1	1	1	1	1	1	1
	s_0	10	10	10	10	10	10	10	10
Complete	r_0	2	2	2	2	2	2	2	2
	s_0	2	10	10	10	10	10	10	10

Table 6 Comparison of model to Brice and Sherman experiments.

M	*N*	*Model*		*Brice/Sherman*	
		F_d	F_r	F_d	F_r
5	10	531	147	590	113
	15	375	388	405	374
	20	85	687	94	682
10	15	247	197	275	199
	20	78	478	85	448
15	20	50	246	59	196

Figure 11 Regions of the modified probability along M.



5. For a suitable value of r, the I/O cost decreases with increasing buffer size. Even for $r = 1$, it was shown in [3] that the total I/O cost for the LRU memory replacement algorithm may be reduced for $N > M$ if M increases with N. If the variation of M with N is known, the expressions in this paper may be used to compare the I/O cost of the three algorithms.

An additional conclusion can be obtained by comparing the page fault rate of the GLRU algorithm with the reference paging of the LRU algorithm. In the particular experiments performed, the paging rate of the GLRU algorithm was usually larger than the LRU reference paging rate. In systems where this behavior occurs, it seems advantageous to attempt to eliminate double paging. Once double paging has been eliminated, it is necessary to obtain a new GLRU algorithm that minimizes reference paging.

Double paging could be eliminated (or reduced) by using one of the following strategies:

1. *Communication between the DMS and the operating system.* This requires an integrated design of these systems, which is not always possible in practice. It also requires that buffer pages and virtual memory pages be the same size or that they be aligned along their boundaries.
2. *Single level store.* In this case, both the database and its buffer lie in the same address space. However, because of the different reference characteristics of database and non-database programs this approach may not result in optimal performance.
3. *Reservation of real pages for buffer use.* Again this does not produce optimal total performance. It has been shown [3] that by using fast paging devices the total I/O cost could be reduced by having a larger virtual buffer.

Experimental validation has shown that good agreement exists between the calculations of the model and the actual measured values, and that the parameters of the GLRU algorithm are relatively stable, indicating that it is possible to apply this algorithm in databases which have adequate parameter stability. This appears to be a common situation since these parameters depend only on average probabilities taken over reference strings.

References

1. W. G. Tuel, Jr., "An Analysis of Buffer Paging in Virtual Storage Systems," *IBM J. Res. Develop.* **20**, 518 (1976).
2. S. W. Sherman and R. S. Brice, "Performance of a Database Manager in a Virtual Memory System," *ACM Trans. Database Syst.* **1**, 317 (1976).
3. T. Lang, C. Wood, and E. B. Fernández, "Database Buffer Paging in Virtual Storage Systems," *ACM Trans. Database Syst.* **2**, 339 (1977).
4. R. S. Brice and S. W. Sherman, "An Extension of the Performance of a Database Manager in a Virtual Memory System Using Partially Locked Virtual Buffers," *ACM Trans. Database Syst.* **2**, 196 (1977).
5. A. Reiter, "A Study of Buffer Management Policies for Data Management Systems," Technical Summary Report 1619, Mathematics Research Center, University of Wisconsin, Madison, 1976.
6. J. R. Spirn and P. J. Denning, "Experiments with Program Locality," *Proc. AFIPS* **41**, 611 (1972).
7. E. G. Coffman, Jr. and P. J. Denning, *Operating Systems Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973, p. 276.
8. C. Wood, E. B. Fernández, and T. Lang, "Minimization of Demand Paging for the LRU Stack Model of Program Behavior," *Research Report G320-2689*, IBM Los Angeles Scientific Center, 1977.
9. J. Rodriguez-Rosell, "Empirical Data Reference Behavior in Data Base Systems," *Computer* **9**, 9 (1976).
10. W. G. Tuel, Jr. and J. Rodriguez-Rosell, "A Methodology for Evaluation of Data Base Systems," *Research Report RJ 1668*, IBM Research Laboratory, San Jose, CA, 1975.
11. D. P. Gaver, S. S. Lavenberg, and T. G. Price, Jr., "Exploratory Analysis of Access Path Length Data for a Data Base Management System," *IBM J. Res. Develop.* **20**, 449 (1976).

Received May 31, 1977; revised November 1, 1977

E. B. Fernández and C. Wood are located at the IBM Scientific Center, Data Processing Division, 9045 Lincoln Blvd., Los Angeles, California 90045. T. Lang is with the Computer Science Department, University of California, Los Angeles, California 90024.