Operating Systems
R. S. Gaines Editor

Cold-Start vs. Warm-Start Miss Ratios

Malcolm C. Easton
IBM Thomas J. Watson Research Center
Ronald Fagin
IBM San Jose Research Laboratory

In a two-level computer storage hierarchy, miss ratio measurements are often made from a "cold start", that is, made with the first-level store initially empty. For large capacities the effect on the measured miss ratio of the misses incurred while filling the first-level store can be significant, even for long reference strings. Use of "warm-start" rather than "cold-start" miss ratios cast doubt on the widespread belief that the observed "S-shape" of lifetime (reciprocal of miss ratio) versus capacity curve indicates a property of behavior of programs that maintain a constant number of pages in main storage. On the other hand, if cold-start miss ratios are measured as a function of capacity and measurement length, then they are useful in studying systems in which operation of a program is periodically interrupted by task switches. It is shown how to obtain, under simple assumptions, the cache miss ratio for multiprogramming from cold-start miss ratio values and how to obtain approximate cold-start miss ratios from warm-start miss ratios.

Key Words and Phrases: miss ratio, cold start, warm start, storage hierarchy, lifetime function, multiprogramming, S-shape

CR Categories: 3.70, 4.32, 9.35

Author's present addresses: M.C. Easton, IBM Thomas J. Watson Research Center, Yorktown Heights NY 10598; R. Fagin, IBM Research Laboratory, 5600 Cottle Rd, San Jose CA 95193. The second author carried out most of this research while at the IBM Thomas J. Watson Research Center.

1. Introduction

In evaluation of performance of a paged, two-level storage hierarchy, the fraction of references which are to pages not in first-level storage is a basic statistic. Such references are called *misses* or *page faults* and the fraction of references that are misses is called the *miss ratio*. In practice, values of the miss ratio are obtained by simulation of operation of the storage hierarchy in response to a particular *reference string* (typically the sequence of page addresses referenced by a selected program). The values obtained depend on the storage management policy, the choice of the reference string, the capacity (in pages) of the first level of storage, the contents of the first level at the start of the simulation, and the length of the reference string. For simplicity, we will only consider the storage management policy that brings a page into the first level when it is requested (not before) and replaces the least recently referenced page (this is also called LRU or "least recently used" replacement).

Rather than choose one reference string, the common practice is to analyze a number of strings obtained from various "typical" programs. This is done for each of a number of capacities.

The dependence of miss ratio on storage management policy, reference string, and capacity is obvious. On the other hand, the dependence on initial conditions and on the length of the reference string is frequently ignored in the literature. The usual procedure is to start with first level storage initially empty and to process either the entire reference string of a program's execution or a "long" section of such a string. There is an initial period during which the first level fills. The number of references required to fill the first level increases rapidly as its capacity increases. Nevertheless, even for large capacities, it is often assumed that miss ratio measurements over "long" reference strings with an initially empty first-level store are representative of miss ratios for operation when the program maintains a constant number of pages in the first level. We will show that incorrect application of this assumption can lead to incorrect interpretations of data and thus to errors in modeling of computer system performance. For example, the widely publicized "S-shape" [1, 3, 12] of plots of the reciprocal of miss ratio versus capacity can be explained by the fact that for large capacities, most of the page faults are incurred while filling the first level. For similar reasons, an understanding of the effect of page size on miss ratio is obscured by failure to consider the importance of initial conditions.

In order to clarify the situation, we introduce terminology and notation that explicitly give information on initial conditions and on the length of the reference string used in the measurement. The *cold-start miss ratio* (to the first-level store) with capacity CAP measured for a string of length T, denoted $COLD(CAP,T)$, is the LRU miss ratio starting with an initially empty first-level store. (If the measurement starts at time t with an initially

empty first-level store, then we say that the miss ratio is measured from a *cold start* at time t.) The *warm-start miss ratio* (to the first-level store) with capacity CAP measured over T references, denoted $WARM(CAP,T)$, is the LRU miss ratio starting with a *filled* first-level store. The initial contents of the first level and LRU stack positions of the stored pages are obtained from the portion of the reference string prior to the start of measurement. An important difference between these miss ratios is that while $WARM(CAP,T)$ can be zero, $COLD(CAP,T)$ includes the effect of initial misses and so is positive for all finite T. For many values of (CAP,T), these miss ratios are nearly equal. In particular, it is easy to see that they differ by no more than CAP/T if both measurements start from the same point of the string. However, if CAP/T is of the same order of magnitude as the miss ratio, then the distinction between the two becomes important.

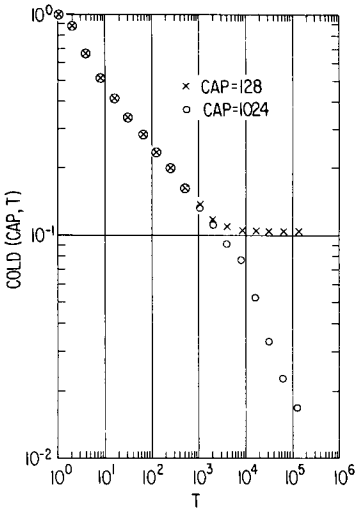
In the following sections we argue that warm-start miss ratios should be measured if behavior of a program running uninterrupted for indefinitely long periods of time is being studied.

On the other hand, under some reasonable assumptions, cold-start miss ratios of a program obtained for moderate values of T (e.g. 10^2–10^5) can be used to compute that program's miss ratio under multiprogramming (when the effect of task-switching must be considered). Thus cold-start miss ratios are useful in studying certain aspects of multiprogramming performance.

If a long reference string is available, then a reasonable procedure for obtaining cold-start miss ratio values is to determine for each pair (CAP,T) an average value of $COLD(CAP,T)$. To obtain this value we compute the number of misses to an initially empty first-level store of size CAP, for each substring of length T, and then we average the results. A practical procedure for carrying out this computation is described in [8]. If CAP is greater than the number of distinct pages in every substring of length T, then this procedure is equivalent to finding average working-set size (i.e. the average number of distinct references in a substring of T references). Figure 1 shows average values of $COLD(CAP,T)$ measured for two values of CAP and several values of T. The measurement was carried out on a string of references to 32-byte pages (a typical page size for a cache) generated by an astronomical orbit calculation program.

If a reference string is available, then one can obtain estimates of $COLD(CAP,T)$ for specific (CAP,T) values by direct measurement. However, there is a considerable body of existing miss ratio data for various values of CAP obtained using a single "large" value of T. We will give a method for obtaining approximate values of $COLD(CAP,T)$ for various values of T from such data.

Fig. 1.



2. Shape of the Lifetime Function

In modeling of computer system behavior, the usual assumption is that the reference string is a realization of a stationary stochastic process (we will formally define *stationarity* in Section 4). More specifically, there is a set of integers called the set of *page names* and a sequence of random variables $\ldots R_{-2}, R_{-1}, R_0, R_1, R_2 \ldots$ whose values are page names (intuitively, the value of R_t is the name of the page referenced at time t). Therefore for a particular value of CAP, under LRU management, there is a random variable $W_{\alpha,t}(CAP)$ whose value is the number of (warm-start) page faults generated by $\langle R_\alpha, R_{\alpha+1}, \ldots, R_{\alpha+t-1}\rangle$, where R_i generates a (warm-start) page fault ($\alpha \leq i \leq \alpha + t - 1$) iff the value of R_i is not contained among the previous CAP distinct values of $\langle \ldots, R_{i-2}, R_{i-1}\rangle$.

For a stationary sequence we define the (*limiting*) *miss ratio* to be

$$M(CAP) = \lim_{t\to\infty} \frac{W_{\alpha,t}(CAP)}{t}. \tag{2.1}$$

The strong law of large numbers [5] implies existence of $M(CAP)$ with probability one. A little thought shows that this limit does not depend on α.

For a stationary sequence, we define the (*limiting*) *lifetime* to be

$$L(CAP) = \lim_{t\to\infty} \frac{t}{W_{\alpha,t}(CAP)}. \tag{2.2}$$

By existence of $M(CAP)$ with probability one and by (2.1), with probability one $L(CAP)$ exists and $L(CAP) = 1/M(CAP)$. The lifetime can be thought of as the average time between page faults.

A reasonable method for estimating the value of $L(CAP)$ is by measuring $t/(W_{\alpha,t}(CAP))$ for finite t. Note that this estimate is the reciprocal of a warm-start miss ratio. If the first reference recorded is the value of R_1, then the value of CAP must be such that the set of values taken by $\{R_1, R_2, \ldots, R_{\alpha-1}\}$ contains at least CAP distinct values. It is not possible to obtain the value of $W_{\alpha,t}(CAP)$ otherwise, since then it depends on R_i for $i < 1$ and this information is not available. If the first $\alpha - 1$ references

of the recorded string contain β distinct references then, using the αth reference as a starting point, we can compute warm-start miss ratios over the remainder of the string for $CAP = 1, 2, \ldots, \beta$ and thus estimate $L(CAP)$ for these values, by finding the reciprocal of the warm-start miss ratio.

However, a number of measurements of lifetime functions have been obtained by measuring $1/COLD(CAP,K)$ where K is the length of the available reference string. Using this method, Belady and Kuehner [1] observed that plots of the lifetime as a function of capacity for several programs have approximately an S-shape, that is, a convex ("up") portion followed by a concave ("down") portion. They comment that the flattening out of the lifetime function curve for large capacities might be due to two factors:

(1) The effect of initial loading misses, so that the lifetime function for large capacities is bounded by the length of the recorded string divided by the number of initial loading misses (misses caused by the first reference to each page).

(2) The effect of diminishing returns due to program locality. Possible factor number (2) is repeated frequently in the literature [3, 6]. Under the scenario of factor (2), the program has already accumulated its "working set" somewhere near the point of inflection [7]. Indeed, the lifetime function is modeled in [3] by a curve with this assumption built in, and values of this function are used as parameters of a stationary page fault process.

In order to examine the plausibility of the two explanations suggested above, we examined several program traces of length 1.5×10^6 references, with measurements beginning after 5×10^5 references. The page size was 2048 bytes. We found that the lifetime function measured by the warm-start method did not usually flatten out for large capacities. Figures 2–4 show the startling difference

Fig. 2.

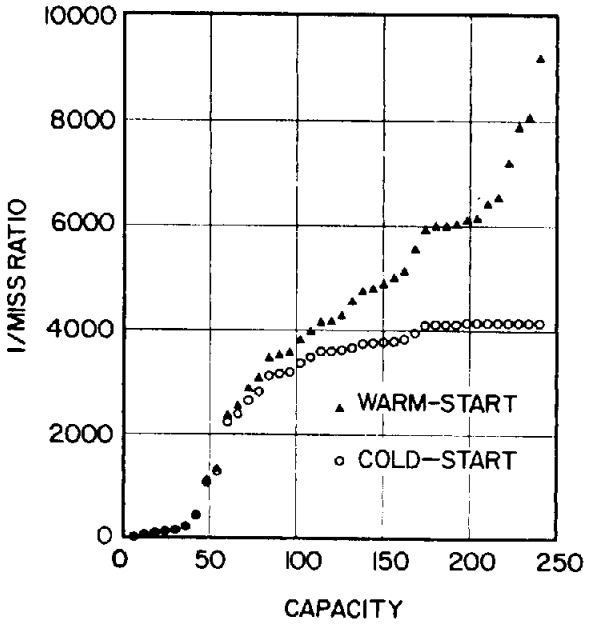


Fig. 3.

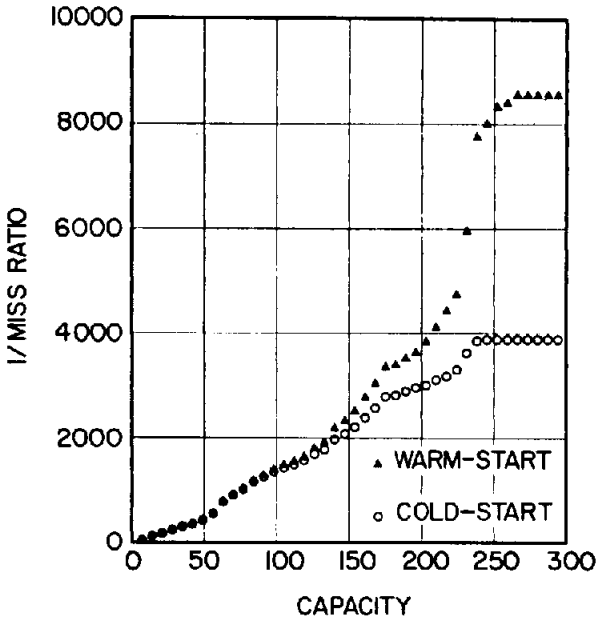
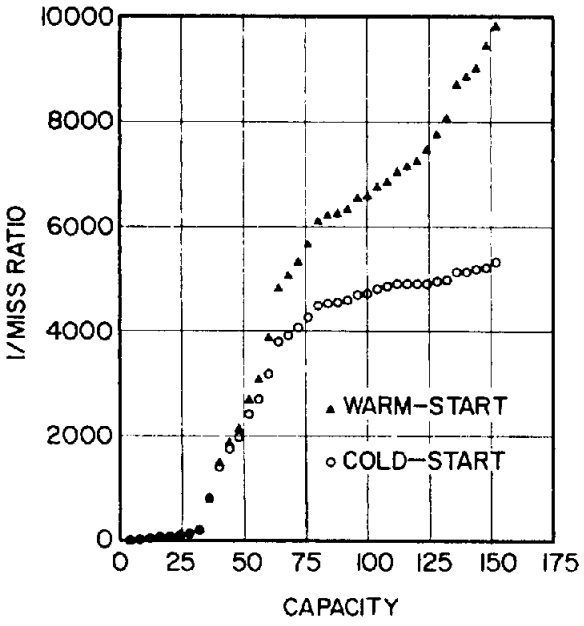


Fig. 4.



between lifetime functions obtained by cold-start and warm-start method for three different program address traces. (Because of statistical uncertainties, the graphs were plotted only at capacities such that at least one hundred warm-start misses were observed.) In each case, the cold-start and warm-start measurements were carried out over identical portions of a reference string. The cold-start lifetime in each case levels off at a value equal to the number of references (10^6) divided by the number of distinct references in the measured portion of the string. The results indicate that for large capacities, far more page faults occur in filling the initially empty first-level store than in the remainder of the string. Thus, the

cold-start miss ratios do not represent behavior of a filled first-level store for large capacities and factor (1) seems to explain the flattening. We suggest use of warm-start miss ratios as estimates for the limiting lifetime function. Furthermore we suggest that in analyses of systems in which a program maintains a constant number of pages in main storage, the lifetime for the program should not be assumed to be S-shaped.

3. Effect of Page Size on Miss Ratio

In this section we note another difference between cold-start and warm-start miss ratios. In [10] and [11] it is shown that in certain simple models of page reference strings, the expected miss ratio is nearly independent of page size, if the size of first-level storage (in bytes) is held fixed. (In these models the expected miss ratio is equal to the limiting miss ratio $M(CAP)$ defined in the previous section, with probability one.) This steady-state analysis corresponds to using warm-start miss ratios; indeed, when simulations are carried out using these models, observed warm-start miss ratios are approximately independent of page size, as predicted. On the other hand, if *cold-start* miss ratios are measured for a very large capacity, then doubling the page size causes the cold-start miss ratio to be approximately halved (this effect appears dramatically in the miss ratio curves of Chu and Opderbeck [4]). The reason is simple: if first-level storage is large enough to hold the entire program, then all cold-start page faults occur while filling the initially empty first-level store. If the first level can hold the entire program, then there are approximately half as many of these initial loading misses when the page size is twice as large.

Thus in order to understand the effect of page size on miss ratio, the effect of page size on the number of initial loading misses should be separated from the effect of page size on the limiting miss ratio. The relative importance of the two factors depends on the length of time the program runs.

4. Effect of Task Interruptions on Miss Ratio

We now discuss cases in which initial loading misses play a major role in determining miss ratio. Miss ratios are usually obtained from reference strings generated by a single program. In such an analysis, it is assumed that once CAP distinct pages have been referenced, from then on first-level storage always contains the CAP pages that have been referenced most recently. In multiprogramming systems, however, task switches often interrupt the execution of a particular program. Before execution of the program resumes, some or all of its pages in first-level storage may be replaced by pages of other tasks.

One specific case in which this occurs is in storage hierarchies using a cache (high-speed first-level store). (For a description and bibliography see [16].) The unit of transfer between cache and main storage is called a *line*. During execution, the lines referenced by a program enter the cache, pushing out lines of other tasks. When one of these other tasks resumes execution, some or all of its lines have been displaced, so its miss ratio is generally higher than when running uninterrupted. (Compare columns 5 and 6 of Table I, which contain some empirical results. These results were obtained from the string of references generated by an astronomical calculation program, ORBIT, and from a PL/I compiler, PLCOMP. In both cases the reference string consisted of 5×10^5 references to 32-byte lines.)

A similar phenomenon can occur under certain management policies for main storage. If a task is considered to be currently inactive (e.g. no human response to an interactive program has occurred for a period of time), then the pages associated with the task are replaced by pages requested by other tasks. Later, when the task becomes active again, its pages are brought into main storage on demand. As in the cache case, the loss of pages during an inactivity period causes the miss ratio to increase over the value when run without interruption.

We will discuss the cache case in some detail. We will assume throughout that *all* cache lines belonging to a program are pushed out of the cache between that program's execution intervals. In many systems this is nearly always true. Otherwise, this can be considered a "worst case" assumption.

We will describe a rather general stochastic model for the reference sequence and task switch process. In this context, we consider the cold-start miss ratio to be a random variable. We show that the long-run cache miss ratio for a program under task switching can be expressed as a weighted average of expected cold-start miss ratios for that program's uninterrupted reference string. The weights are determined by the probability distribution of lengths of execution intervals. When the latter distribution is not known, we show that the assumption that all execution intervals are of equal length yields the maximum cache miss ratio for a given task switch rate.

We begin by defining precisely our notion of stationarity. A sequence of random variables $\ldots H_{-2}, H_{-1}, H_0, H_1, H_2$ each having as range the set $G = \{g_1, g_2, \ldots\}$ is said to be *stationary* if for every two sets of α integers $\{i_1, i_2, \ldots, i_\alpha\}$ and $\{j_1, j_2, \ldots, j_\alpha\}$ and for every integer β:

$$\Pr\{H_{i_1} = g_{i_1}, \ldots, H_{i_\alpha} = g_{j_\alpha}\} = \Pr\{H_{i_1+\beta} = g_{j_1}, \ldots, H_{i_\alpha+\beta} = g_{i_\alpha}\}.$$

Taking $\alpha = 1$, for example, we see that $E[H_l]$ is the same for all l.

We assume that the reference string is a realization of $\langle R_t \rangle$, a stationary sequence. Let CAP be the number of cache lines available to the task. In this section, we will use $\overline{COLD}(CAP,T)$ to denote the *expected* cold-start miss ratio, i.e., the expected value of the random variable equal to $1/T$ times the number of misses generated by $R_t, \ldots, R_{t+T-1}$ with a cold start at time t. Let τ_i, $i = 1, 2,$

Table I. Execution Interval Determined by Main Storage Page Faults; Cache Line Size = 32 Bytes.

Col	1	2	3	4	5	6
Program	Mean length of execution interval $\bar{T}$	Cache capacity (lines) CAP	Average value of $\overline{COLD}(CAP,\bar{T})$	Cache miss ratio from application of (4.3) or equivalently, of (4.4)	Measured cache miss ratio assuming none of program's lines are present in cache at start of each execution interval	Measured cache miss ratio when program is run uninterrupted
ORBIT	2910	256	0.10	0.089	0.087	0.077
		512	0.10	0.082	0.079	0.064
		1024	0.10	0.057	0.057	0.014
PLCOMP	5510	256	0.050	0.042	0.045	0.033
		512	0.050	0.040	0.041	0.028
		1024	0.050	0.037	0.037	0.021

... be a random variable giving the length of the ith execution interval. Thus if $\tau_1 = j$ then the first execution interval contains $R_1, \ldots, R_j$. The second interval begins with R_{j+1}, etc. Let $\eta_i(CAP)$, $i = 1, 2, \ldots$ be a random variable giving the number of misses to a cache of size CAP in the ith execution interval, under the assumption that the cache contains no lines of the task at the start of the interval. We assume that random variables $\{\tau_i\}$ and $\{\eta_i(CAP)\}$ have finite expected values and variances and that $\langle\tau_i\rangle$ and $\langle\eta_i(CAP)\rangle$ are stationary sequences. We also assume that each has autocorrelation asymptotically zero (that is, the correlation between τ_i and τ_{i+x} goes to 0 as $x \to \infty$; similarly for $\langle\eta_i\rangle$.)

We define the *long-run cache miss ratio* to be the limit (as $k \to \infty$) of

$$\sum_{i=1}^{k} \eta_i(CAP) / \sum_{i=1}^{k} \tau_i \tag{4.1}$$

The assumptions imply that $\langle\tau_i\rangle$ and $\langle\eta_i(CAP)\rangle$ satisfy the weak law of large numbers, from which it follows that (4.1) converges in probability to:

$$E[\eta_i(CAP)])/E[\tau_i] \tag{4.2}$$

Letting P_T denote $\Pr[\tau_i = T]$, and $\bar{T} = \sum_{T=1}^{\infty} T \cdot P_T$, the expected interval length, we can write (4.2) as:

$$\sum_{T=1}^{\infty} (E[\eta_i(CAP) | \tau_i = T] P_T)/\bar{T}$$

Finally, we assume that $T \cdot \overline{COLD}(CAP,T)$, the expected number of cold-start misses in T references, is equal to $E[\eta_i(CAP) | \tau_i = T]$. This means that the expected number of cold-start misses in an arbitrarily chosen substring of length T, starting with an empty first-level store, is the same as the expected number in an execution interval of length T. This feature of the model is easily justifiable if the task switches are caused by time slicing or by interrupts associated with other tasks, since then the task switches are caused by a mechanism independent of the program in execution. However, even if task switching is primarily caused by main store page faults, then the cache fault process, taking place on a "microscopic level" relative to main store references, may not depend significantly on the times of occurrence of the main store page faults. For example, a program transferring control from page to page occasionally has a page fault. But the code executed on pages causing faults need not have significantly different statistical properties from the code executed on other pages.

Under all these assumptions, the long-run cache mass ratio is:

$$\left(\sum_{T=1}^{\infty} P_T T \cdot \overline{COLD}(CAP,T)\right) \Big/ \bar{T} \tag{4.3}$$

Relation (4.3) has been tested empirically for several program traces where task switches were caused *only* by main store page faults and where values of P_T and $\overline{COLD}(CAP,T)$ were estimated from the reference string being tested. Assuming stationarity of $\langle R_i\rangle$, it is not hard to show that the procedure of averaging the cold-start miss ratios over all substrings of length T gives an unbiased estimate for $\overline{COLD}(CAP,T)$. Values obtained from (4.3) by substituting average cold-start miss ratios for $\overline{COLD}(CAP,T)$ were compared with the observed average cache miss ratio (total misses divided by number of references) obtained assuming emptying of the cache at each task switch. The compared values for the strings tested agreed within 6 percent for values of $CAP = 256$, 512, and 1024 lines, with each line of size 32 bytes. Typical results appear in columns 4 and 5 of Table I.

One application of (4.3) is to yield a lower bound for long-run cache miss ratio under multiprogramming as cache capacity is increased, if we assume that the cache is nevertheless too small to retain any of the task's lines at the start of the task's next execution interval. This bound is obtained by using $\overline{COLD}(\infty,T)$ for $\overline{COLD}(CAP,T)$ in (4.3). This value $\overline{COLD}(\infty,T)$ is simply $1/T$ times the expected number of distinct cache lines in a substring of length T.

For the remainder of this section, we will hold CAP fixed. Denote $T \cdot \overline{COLD}(CAP,T)$, the expected number of cold-start misses after T references, by $\bar{N}(T)$. Then our formula (4.3) for the long-run cache miss ratio becomes

$$\left(\sum_{T=1}^{\infty} P_T \bar{N}(T)\right) \Big/ \bar{T} \tag{4.4}$$

Note that $\bar{N}(T)$ has been defined only for integral values of T. It will be convenient to define $\bar{N}(T)$ for nonintegral values of T by linear interpolation.

Suppose that the probability distribution $\{P_T\}$ of execution interval lengths is unknown, but that the expected execution interval length, $\bar{T}$, is known. Then in (4.4), the assumption that all intervals have length $\bar{T}$ yields for the cache miss ratio the value of $\bar{N}(\bar{T})/\bar{T}$ (that is, $\overline{COLD}(CAP,\bar{T})$, if $\bar{T}$ is an integer). At the conclusion of this section, we will show that this value, $\bar{N}(\bar{T})/\bar{T}$, is an upper bound for (4.4). So for fixed $\bar{T}$, uniform interval lengths gives the highest cache miss ratio. If task switches are caused primarily by main storage page faults, then $\bar{N}(\bar{T})/\bar{T}$ may be considerably larger than (4.4). (In Table I, compare the estimates in column 3 for $\overline{COLD}(CAP,\bar{T})$, which are also estimates for $\bar{N}(\bar{T})/\bar{T}$, with the values in column 4, which are obtained from (4.3) or (4.4), and note that the values in column 3 are larger than the corresponding values in column 4.) Measurements by Ghanem [13, 14] suggest that $\{P_T\}$ has approximately a hyperexponential distribution when all task switches are caused by page faults. If all task switches are caused by page faults, then use of such a distribution should give more accurate cache miss ratios than those obtained with intervals of equal length. We will now prove that under the stochastic assumptions of this section, $\bar{N}(\bar{T})/\bar{T}$ is greater than or equal to (4.4), that is, that the expected value of an entry in column 3 of Table I is greater than or equal to the expected value of the corresponding entry in column 4.

THEOREM 1. $(\bar{N}(\bar{T}))/\bar{T} \geq (\sum_{T=1}^{\infty} P_T \bar{N}(T))/\bar{T}$.

PROOF. It is sufficient to show that the function $T \rightarrow \bar{N}(T)$ is concave (down) since then, by a fundamental property of concavity [15, p. 70], we have $\bar{N}(\sum_{T=1}^{\infty} P_T T) \geq \sum_{T=1}^{\infty} P_T \bar{N}(T)$. In fact, we need only show that for each integer $T \geq 1$,

$$\tfrac{1}{2}(\bar{N}(T-1) + \bar{N}(T+1)) \leq \bar{N}(T). \tag{4.5}$$

Since $\bar{N}(T)$ is defined by linear interpolation for nonintegral T, it is clear from geometric considerations, and not difficult to prove, that (4.5) implies concavity of the function (in other words, if $\bar{N}$ is "concave at integer points," then it is "concave everywhere").

Let $U_{i,j}$ be a random variable whose value is 1 if R_{i+j-1} takes a value that causes a cold-start miss with a cold start at time i. Otherwise the value of $U_{i,j}$ is zero. Then

$$\begin{aligned}\bar{N}(T) &= E[U_{i,1} + U_{i,2} + \ldots U_{i,T}] \\ &= E[U_{i,1}] + E[U_{i,2}] + \ldots + E[U_{i,T}] \\ &= a_1 + a_2 + \ldots + a_T,\end{aligned}$$

where from each j, we define a_j to be $E[U_{i,j}]$, the expected value of $U_{i,j}$. (Stationarity of $\langle R_i \rangle$ implies that these expected values are the same for all i.) If the $(i + T - 1)$-th reference is a miss with a cold start at time $i - 1$, then it is a miss with a cold start at time i. Thus $U_{i-1,T+1} \leq U_{i,T}$ which implies $a_{T+1} \leq a_T$. Therefore $\bar{N}(T+1) - \bar{N}(T) = a_{T+1} \leq a_T = \bar{N}(T) - \bar{N}(T-1)$, from which (4.5) follows, and the proof is complete. □

5. Simple Recipe for Cold-Start Miss Ratios

The previous section showed the usefulness of cold-start miss ratios in evaluating the cache miss ratio under multiprogramming. In this section we will give a simple recipe for obtaining approximations to cold-start miss ratios $COLD(CAP,T)$ from estimates for values of $M(CAP)$, the limiting miss ratio. Thus, assume that one has available values $MISS(1)$, $MISS(2)$, $MISS(3)$, ... , $MISS(CAP)$, ... , where $MISS(CAP)$ is an estimate for the limiting LRU miss ratio with capacity CAP. For example, $MISS(CAP)$ could be an observed value of $COLD(CAP,\bar{T})$ or of $WARM(CAP,\bar{T})$ for some large value of $\bar{T}$. Using these values, we will obtain approximations to $COLD(CAP,T)$ for certain pairs (CAP,T); other values can be obtained by interpolation.

Define $LIFE(CAP)$ to be $1/MISS(CAP)$. If $MISS(CAP)$ is exactly the limiting miss ratio then $LIFE(CAP)$ is the limiting lifetime, that is, the average time between page faults, with capacity CAP (thus $LIFE(CAP)$ is then $L(CAP)$ as defined in (2.2)). Of course, if $MISS(CAP)$ is an approximation to the limiting miss ratio, then $LIFE(CAP)$ is an approximation to the limiting lifetime. For each integer k, define

$$LIFE^*_{CAP}(k) = \sum_{i=0}^{k-1} LIFE(\min\{i, CAP\}), \tag{5.1}$$

where we adopt the convention that $LIFE(0) = 1$. If $LIFE^*_{CAP}(k_0) = T_0$, then *the estimate given by our recipe for* $COLD(CAP,T_0)$ *is* k_0/T_0.

We tested our recipe on several program address traces of length $\bar{T} \sim 10^6$ references. We used measured values of $1/WARM(CAP,\bar{T})$ as values for $LIFE(CAP)$ in (5.1) and found that our estimate was almost always within 10–15 percent of the directly observed average cold-start miss ratio. In a slightly expanded version of this paper [9], the authors show that in a certain precise sense this recipe is good in the well-known "LRU stack model" [2, p. 275; 17]. (In this model there are fixed probabilities $\{r_i\}$ such that the probability that the next reference is to the page in LRU stack position i is r_i, independent of past history.) It is shown in that report that if limiting lifetimes are used in the recipe and if the number of cold-start misses in T references is reasonably large, then with high probability the value of the random variable equal to the cold-start miss ratio for T references is close to the value obtained from the recipe.

We will now give a rough intuitive explanation as to why our recipe is reasonable. Let T_0 denote $LIFE^*_{CAP}(k_0)$. We will show that T_0 is approximately the average time (number of references), starting from an initially empty first-level store, until the k_0th page fault. (In fact,

we will show that in the LRU stack model, if we take for $LIFE(CAP)$ the exact value of the limiting lifetime, that is,

$$LIFE(CAP) = 1/\sum_{i>CAP} r_i, \tag{5.2}$$

then the value T_0 is *exactly* the expected time after a cold start until the k_0th page fault.) Since the k_0th cold-start page fault occurs on the average at time T_0, a reasonable estimate for the cold-start miss ratio at time T_0 is k_0/T_0, which is our recipe value.

We will now demonstrate our claim that in the LRU stack model, $T_0 = LIFE^*_{CAP}(k_0)$ is the expected time until the k_0th page fault. In the LRU stack model, it is easily verified that the time until a page is referenced which is distinct from the l distinct pages most recently referenced has a geometric distribution with expected value $LIFE(l)$. Thus $LIFE(0) = 1$ is the time till the first page fault; $LIFE(1)$ is the expected time between the first and second page fault; $LIFE(\min\{2, CAP\})$ (which is $LIFE(2)$ if $CAP \geq 2$, or $LIFE(1)$ if $CAP = 1$) is the expected time between the second and third page fault; and so on. Hence T_0, which is the sum of k_0 such terms, is the expected time till the k_0th page fault. If conversely, the expected number of cold-start misses over time T_0 (assuming T_0 is an integer) were k_0, then our recipe would exactly give the expected value $COLD(CAP, T_0)$ of the cold-start miss ratio in this model. However, we remark without proof that this need not be the case, even in the LRU stack model.

Conclusions

We have shown that knowledge of the initial conditions and the length of the reference string measured are important in properly interpreting miss ratio data. Among the errors that we believe have been committed by ignoring these factors is the assumption that an S-shaped lifetime function represents behavior of a program that maintains a constant number of pages in main storage. On the other hand, for a system in which a program's execution is periodically interrupted, we have shown the usefulness of the two-parameter "cold-start miss ratio" in analyzing miss ratio performance. Further, we have provided a simple recipe for obtaining approximate values of cold-start miss ratios from limiting miss ratios.

Acknowledgments. We thank B. Ingebrand for supplying instruction trace tapes and M.Z. Ghanem for supplying an algorithm for producing a reference string from an instruction trace. We are grateful to D. Slutz, E. McNair, and T. Moeller for allowing us to use programs they have developed, and to A.C. McKellar for reading a draft of the manuscript.

Received November 1975; revised October 1976

References

1. Belady, L.A., and Kuehner, C.J. Dynamic space-sharing in computer systems. *Comm. ACM 12*, 5(May 1969), 282–288.
2. Coffman, E.G. Jr., and Denning, P.J. *Operating Systems Theory.* Prentice-Hall, Englewood Cliffs, N.J., 1973.
3. Chamberlin, D.D., Fuller, S.H., and Liu, L.Y. An analysis of page allocation strategies for multiprogramming systems with virtual memory. *IBM J. Res. Develop. 17* (Sept. 1973), 404–412.
4. Chu, W.W., and Opderbeck, H. Performance of replacement algorithms with different page sizes. *Computer 7*, 11 (Nov. 1974), 14–21.
5. Doob, J.L. *Stochastic Processes.* Wiley, New York, 1953.
6. Denning, P.J., and Graham, G.S. Multiprogrammed memory management. IEEE Proc. Interactive Computer Systems, **63** (June 1975), pp. 924–939.
7. Denning, P.J., and Kahn, K.C. A study of program locality and lifetime functions. Proc. Fifth Symp. on Operating Systems Principles, Austin, Texas, 1975, pp. 207–216 (available from ACM, New York).
8. Easton, M.C. Computation of cold-start miss ratios. *IEEE Trans. Comptrs. C-27*, 5 (May 1978), 404–408.
9. Easton, M.C., and Fagin, R. Cold-start vs. warm-start miss ratios and multiprogramming performance. IBM Res. Rep. RC 5715, IBM T.J. Watson Res. Ctr., Yorktown Heights, N.Y., Nov. 1975.
10. Fagin, R. Asymptotic miss ratios over independent references. *J. Computer Syst. Sci. 14*, 2 (April 1977), 222–250.
11. Fagin, R., and Easton, M.C. The independence of miss ratio on page size. *J.ACM. 23*, 1 (Jan. 1976), 128–146.
12. Ghanem, M.Z. Study of memory partitioning for multiprogramming systems with virtual memory. *IBM J. Res. Develop. 19*, 5 (Sept. 1975), 451–457.
13. Ghanem, M.Z. Experimental study on the behavior of programs. IBM Res. Rep. RC 5460, IBM T.J. Watson Res. Ctr., Yorktown Heights, N.Y., June 1975.
14. Ghanem, M.Z. Personal communication.
15. Hardy, G.H., Littlewood, J.E., and Polya, G. *Inequalities.* Cambridge U. Press, London, 1964.
16. Kaplan, K.R., and Winder, R.O. Cache-based computer systems. *Computer 6*, 3 (March 1973), 30–36.
17. Shedler, G.S., and Tung, C. Locality in page reference strings. *SIAM J. Comptng. 1*, 3 (Sept. 1972), 218–241.

Corrigendum. Scientific Applications

Gabor T. Herman, Arnold Lent, and Peter H. Lutz, "Relaxation Methods for Image Reconstruction," *Comm. ACM 21*, 2 (February 1978), 152–158.

Figure 1 was adapted from a paper by R. S. Ledley, G. DiChiro, A. J. Luessenhop, and H. L. Twigg, "Computerized Transaxial X-ray Tomography of the Human Body, *Science 186*, 4160 (Oct. 1974), 207–212. It was incorrectly credited to Reference [11] when it should have been credited to Reference [12], the above-mentioned paper by Ledley, DiChiro, Luessenhop, and Twigg.