Estimating Cache Hit Rates from the Miss Sequence

Timothy Y. Chow, Terence P. Kelly, Daniel M. Reeves
Enterprise Systems and Software Laboratory
HP Laboratories Palo Alto
HPL-2007-155
September 17, 2007*

cache, hit rate, estimation, LRU stack distance, success function

This paper considers the problem of inferring cache hit rates from observations of references that miss in the cache. In the Web context, this is the problem of inferring a browser's cache hit rate by examining the requests that it issues to origin servers. We show that a case of this problem can be solved.

* Internal Accession Date Only

Approved for External Publication

Estimating Cache Hit Rates from the Miss Sequence

Timothy Y. Chow
tchow AT alum DOT mit DOT edu

Terence P. Kelly
kterence@hpl.hp.com

Daniel M. Reeves
dreeves@yahoo-inc.com

September 14, 2007

1 Introduction

This paper considers the problem of inferring cache hit rates from observations of references that miss in the cache. In the Web context, this is the problem of inferring a browser's cache hit rate by examining the requests that it issues to origin servers. We show that a case of this problem can be solved.

Specifically we consider a sequence of references to unit-sized objects generated by an LRU (Least Recently Used) stack distance model and filtered by an LRU cache of known size. The LRU stack distance model is well understood [8] and is reported to describe certain real-world reference sequences fairly well, e.g., Web accesses [1] and memory references in programs [9]. The problem of efficiently transforming a reference sequence into a corresponding LRU stack distance sequence is an interesting problem in its own right and has attracted considerable attention [4–7, 10]. For an extensive early survey of paging in storage hierarchies, see [3]. For analyses of cache removal policies under the independent reference model, see [11, 12].

2 Problem

We begin with a known universe of S symbols $\{A, B, C, \ldots\}$ initially arranged in arbitrary order in a *stack*. Stack *positions* are numbered from top to bottom; we say that symbols occupy positions. The upper K positions in the stack are called the *cache*; we consider only cases where $1 \leq K < S$. Associated with each stack position i is a probability p_i such that $\sum_{i=1}^{S} p_i = 1$. We emphasize that these probabilities are associated with *stack positions*, not with symbols.

The stack is repeatedly updated according to the following procedure: From time to time a stack *position* is drawn independently from the distribution defined by probabilities p_i. When a position is chosen, the symbol occupying it is removed from the stack, causing all symbols above it to "fall down" one position. The chosen symbol is replaced at the top of the stack, in position 1. If the chosen *position* is strictly greater than the cache size, then the chosen *symbol* is visible to us; otherwise we see nothing. The sequence of observed symbols is called the *miss sequence*.

Our problem is to estimate $H \equiv \sum_{i=1}^{K} p_i$, the *hit rate* of the cache. We observe only the (infinite) miss sequence; we have no information about the times at which references occur. The question is, does enough information about cache hits somehow "leak out" through the miss sequence to permit us to estimate H?

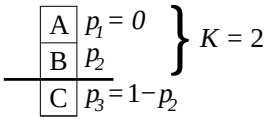

Figure 1: The case of $S = 3$ and $K = 2$.

3 Calculation of Subword Frequencies

It is easy to show that the problem defined in Section 2 cannot be solved as stated. Consider the case of $S = 2$ and $K = 1$. Regardless of the cache hit rate $H = p_1$, the miss sequence is "$\ldots ABABABAB\ldots$." More generally, we can add arbitrary probability mass to p_1 while maintaining the relative magnitudes of p_2 through p_S, and the miss sequence will be unchanged. So from now on we set $p_1 = 0$.

To avoid certain degenerate cases, we also assume that $p_i \neq 0$ for $i > 1$.

Perhaps the simplest statistics obtainable from the miss sequence are *subword frequencies*, e.g., the frequency with which the three-letter string "ABA" appears in the miss sequence. So one strategy for estimating the hit rate is to estimate the subword frequencies by empirical observation, express the subword frequencies as functions of the p_i, and then invert these functions to obtain estimates of the p_i, from which we can estimate the hit rate. We illustrate this process for the case of $S = 3$ and $K = 2$, with $p_1 = 0$, shown in Figure 1.

After a symbol in the miss sequence is observed, the cache may "churn" for a while before the next symbol in the miss sequence is generated. To analyze the churn, first number the $K!$ possible orderings of the cache elements in some fixed manner, e.g., 1.AB, 2.BA. Then write down the *transition matrix* T whose (i, j) entry $T_{i,j}$ is the probability that, given that the cache is currently in the ith ordering, a single stack update will result in the jth ordering. In our example,

$$T = \begin{pmatrix} 0 & p_2 \\ p_2 & 0 \end{pmatrix}. \tag{1}$$

The cache may churn for $0, 1, 2, \ldots$ updates, so to obtain the probability $T^*_{i,j}$ that the cache will churn from ordering i to ordering j, we need to compute

$$T^* = T^0 + T^1 + T^2 + T^3 + \cdots = (I - T)^{-1} = \frac{1}{1 - p_2^2} \begin{pmatrix} 1 & p_2 \\ p_2 & 1 \end{pmatrix}. \tag{2}$$

Now we are ready to calculate the transition probabilities associated with the generation of each symbol in the miss sequence. Number the $S!$ possible stack states in some fixed manner; in our running example, let us write 1.ABC, 2.ACB, 3.BAC, 4.BCA, 5.CAB, 6.CBA. Let $M_{i,j}$ denote the probability that, given that the stack starts in state i, the stack will be in state j immediately after the next miss. For example, let us calculate $M_{1,6}$. We start with ABC, and we want the probability that we get CBA immediately after the next miss. This is precisely the probability that the cache churns from AB to BA, and then C is picked and moved to the top. Thus $M_{1,6} = T^*_{1,2} p_3 = p_2/(1 + p_2)$. Similar calculations yield

$$M = \frac{1}{1 + p_2} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & p_2 \\ 0 & 0 & 1 & p_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_2 & 1 \\ 1 & p_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_2 & 1 & 0 & 0 \\ p_2 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3}$$

The matrix M gives us the probabilities associated with generating each symbol in the miss sequence, so to obtain subword frequencies for n-symbol subwords, we need to consider nth powers of M. More precisely, we can calculate the generating function for subword frequencies as follows. Each nonzero entry of M is associated with a certain missed symbol; for example, $M_{1,6}$ is associated with C. Form the matrix

$$\hat{M} = \frac{1}{1+p_2} \begin{pmatrix} 0 & 0 & 0 & 0 & C & p_2C \\ 0 & 0 & B & p_2B & 0 & 0 \\ 0 & 0 & 0 & 0 & p_2C & C \\ A & p_2A & 0 & 0 & 0 & 0 \\ 0 & 0 & p_2B & B & 0 & 0 \\ p_2A & A & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4}$$

From Markov chain theory we know that in the long run, each stack state is equally probable, so let $x = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$, and let y be the all-1's column vector. Then $x\hat{M}^n y$ is the generating function for length n subword frequencies (where, when calculating $\hat{M}^n$, we treat the symbols A, B, and C as *non-commuting* variables; i.e., when we multiply them, we concatenate them as strings and do not allow the ordering of the symbols to be disturbed). For example, when $n = 3$,

$$\begin{aligned} x\hat{M}^3 y &= (p_2(ABA + ACA + BAB + BCB + CAC + CBC) \\ &\quad + ABC + ACB + BAC + BCA + CAB + CBA)/6(1+p_2). \end{aligned} \tag{5}$$

The frequency of a subword is just its coefficient in this expression. For example, $Freq(ABA) = p_2/(6+6p_2)$ and $Freq(ABC) = 1/(6+6p_2)$.

In this example we are lucky, and can solve for p_2; namely, $p_2 = Freq(ABA)/Freq(ABC)$. However, in general, although we can calculate subword frequencies for any stack size and cache size using the above procedure, the resulting subword frequencies are complicated functions of the p_i, and it is not clear whether one can solve for all the p_i. Moreover, even if solving for the p_i is possible in principle, the matrices involved in our computation method grow like a factorial function, so this does not yield a practical solution for estimating the hit rate.

4 Gaps and the Uniform Case

The results of the last section suggest that it might be promising to analyze the number of symbols that we will observe (in the miss sequence) before we observe the symbol currently at the top of the stack again. Call this random variable G, for "gap." For example, in the case $S = 3$ and $K = 2$ that we analyzed in Section 3,

$$\begin{aligned} P[G=1] &= Freq(ABA) + Freq(ACA) + Freq(BAB) + Freq(BCB) \\ &\quad + Freq(CAC) + Freq(CBC) \\ &= \frac{p_2}{1+p_2}. \end{aligned} \tag{6}$$

Let us now analyze G in the following slightly more general case, that we call the *uniform case*. Assume that $p_i = p$ is constant for $1 < i \le K$ and that $p_j = q$ is constant for $K < j \le S$, where $(K-1)p + (S-K)q = 1$; assume also that $p \neq 0$ and $q \neq 0$. Since S and K are assumed to be known, and q can be computed if p is known, there is just one unknown parameter to be estimated, namely p (or equivalently $H = (K-1)p$). This simplifies the problem significantly, although it is still far from trivial.

For any fixed value of K, the generating function

$$G(x) = \sum_{n \geq 1} P[G = n] x^n \tag{7}$$

in the uniform case may be computed as follows. The miss sequence may be broken up into three phases; the first phase lasts until just before the symbol currently at the top of the stack—call it Z—falls out of the cache, the second (very brief) phase is the single step when Z falls out of the cache, and the third phase lasts until Z is selected for the first time after falling out of the cache.

To analyze the first phase, we set up a Markov chain with K states, where the ith state represents Z being in position i in the cache. The transition probabilities are straightforward to calculate; for example, if Z is currently in position 4 (and $K > 4$), then the probability is p that Z moves to the top of the stack, the probability is $2p$ that Z stays where it is, the probability is $(K-4)p$ that Z advances to position 5 but no missed symbol is generated, and the probability is $1-H$ that Z advances to position 5 and some missed symbol *is* generated. Similar calculations yield the following transition matrix, where the variable x is used to keep track of transitions that generate a missed symbol, and where for compactness we introduce the notation $m_i(x) \equiv (K-i)p + (1-H)x$.

$$M(x) = \begin{pmatrix} 0 & m_1(x) & 0 & 0 & 0 & 0 & \dots & 0 \\ p & 0 & m_2(x) & 0 & 0 & 0 & \dots & 0 \\ p & 0 & p & m_3(x) & 0 & 0 & \dots & 0 \\ p & 0 & 0 & 2p & m_4(x) & 0 & \dots & 0 \\ p & 0 & 0 & 0 & 3p & m_5(x) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p & 0 & 0 & 0 & 0 & 0 & \dots & m_{K-1}(x) \\ p & 0 & 0 & 0 & 0 & 0 & \dots & (K-2)p \end{pmatrix}. \tag{8}$$

The cache starts in state 1, and must advance to state K just before Z falls out of the cache. This can happen after $0, 1, 2, \dots$ stack updates. Therefore the multiplicative factor that the first phase contributes to $G(x)$ is the $(1,K)$ entry of

$$M(x)^0 + M(x)^1 + M(x)^2 + M(x)^3 + \cdots = (I - M(x))^{-1}. \tag{9}$$

(The series summation may be justified by, for example, noting that $M(x)^n \to 0$ if $|x| < 1$.)

The second phase contributes a factor of $(1-H)x$ to $G(x)$.

During the third phase, each symbol outside the cache has an equal probability of being the next missed symbol, and we are simply waiting until Z gets picked. This phase contributes a factor of

$$\begin{aligned} \frac{1}{S-K} + \frac{1}{S-K}\left(\frac{(S-K-1)x}{S-K}\right) + \frac{1}{S-K}\left(\frac{(S-K-1)x}{S-K}\right)^2 + \cdots \\ = \frac{1}{S-K-(S-K-1)x} \end{aligned} \tag{10}$$

to $G(x)$. Putting all the factors together, we obtain the following theorem.

Theorem 1. *In the uniform case, the generating function of equation (7) is given by*

$$G(x) = \frac{(I - M(x))^{-1}_{1,K}(1-H)x}{S-K-(S-K-1)x}. \tag{11}$$

For $K = 2, 3, 4$, $G(x)$ works out to be, respectively,

$$\frac{x[p+(1-p)x]}{[1+p-px][S-2-(S-3)x]}$$

$$\frac{x[p+(1-2p)x][2p+(1-2p)x]}{[1+p-(p+2p^2)x-(p-2p^2)x^2][S-3-(S-4)x]}$$

$$\frac{x[p+(1-3p)x][2p+(1-3p)x][3p+(1-3p)x]}{[1-p^2-(p+2p^2+3p^3)x-(p+p^2-12p^3)x^2-(p-6p^2+9p^3)x^3][S-4-(S-5)x]}$$

By taking Taylor expansions of these expressions, we can compute an expression for $P[G = n]$ for any desired n. This can be compared with the empirical value that is estimated from the miss sequence, and then one can solve for p.

Unfortunately, in general we have no closed-form expression for $(I - M(x))^{-1}$, and therefore Theorem 1 is of limited use unless K is very small.

However, we do have a formula for the coefficient of x in $G(x)$, i.e., for $P[G = 1]$.

Theorem 2. *In the uniform case,*

$$P[G=1] = \frac{(K-1)!\,p^{K-1}}{(S-K)(1+p)(1-p)(1-2p)(1-3p)\cdots(1-(K-3)p)}. \tag{12}$$

This solves the uniform case in theory; we simply estimate $P[G = 1]$ from the miss sequence and then find the smallest positive value of p satisfying equation (12). However, in general $P[G = 1]$ will be very small, so estimating it will require a very long fragment of the miss sequence.

We conclude this paper with the proof of Theorem 2. The key step is the following lemma.

Lemma 1. *The eigenvalues of the matrix*

$$X = \begin{pmatrix} 0 & K-1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & K-2 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & K-3 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 2 & K-4 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & 3 & K-5 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & \dots & K-2 \end{pmatrix} \tag{13}$$

are $-1, 0, 1, 2, 3, \dots, K-4, K-3, K-1$.

Proof of Lemma 1. For $1 \le n \le K-1$, let v_n be the row vector

$$v_n = \left(\underbrace{0, 0, \dots, 0}_{K-n-1 \text{ zeroes}}, \binom{n}{0}, -\binom{n}{1}, \binom{n}{2}, -\binom{n}{3}, \dots, (-1)^n \binom{n}{n} \right). \tag{14}$$

We claim that $v_n X = (K-n-2)v_n$, and hence that $K-n-2$ is an eigenvalue of X. To see this, let us consider each entry of $v_n X$ in turn. Unless $n = K-1$, the first entry of $v_n X$ is an alternating sum of binomial coefficients that equals zero (confirming our claim); if $n = K-1$, then the first entry of

$v_n X$ is -1, again confirming our claim. The subsequent entries up to entry number $K-n-1$ are easily seen to be zero. For $i = 0, 1, \ldots, n$, entry number $K-n+i$ of $v_n X$ equals

$$\begin{aligned}(-1)^i \binom{n}{i}(K-n+i-2) &+ (-1)^{i-1}\binom{n}{i-1}(n-i+1) \\ &= (-1)^i \binom{n}{i}\left(K-n+i-2-\frac{i}{n-i+1}(n-i+1)\right) \\ &= (-1)^i \binom{n}{i}(K-n-2), \end{aligned} \tag{15}$$

completing the confirmation of the claim.

Since n ranges from 1 to $K-1$, this shows that $-1, 0, 1, 2, \ldots, K-3$ are eigenvalues of X. It is also easily seen that the all-1's vector is an eigenvector of X with eigenvalue $K-1$. This gives us K distinct eigenvalues, and since X is a $K \times K$ matrix, this list of eigenvalues is complete. □

Proof of Theorem 2. By equation (7), we see that $P[G=1]$ may be calculated by dividing $G(x)$ by x and then setting $x = 0$. By Theorem 1, this means we want to calculate

$$P[G=1] = \frac{(I-M(0))^{-1}_{1,K}(1-H)}{S-K}. \tag{16}$$

Now $M(0)$ is just the matrix X in Lemma 1 with every entry multiplied by p. Hence the eigenvalues of $M(0)$ are

$$-p, 0, p, 2p, 3p, \ldots, (K-4)p, (K-3)p, (K-1)p, \tag{17}$$

and therefore the eigenvalues of $I-M(0)$ are

$$1+p, 1, 1-p, 1-2p, 1-3p, \ldots, 1-(K-4)p, 1-(K-3)p, 1-(K-1)p. \tag{18}$$

To compute $(I-M(0))^{-1}_{1,K}$, we take $(-1)^{K-1}$ times the determinant of the submatrix obtained by deleting the first column and last row of $I-M(0)$ (conveniently, this submatrix is lower triangular with diagonal entries $-p, -2p, \ldots, -(K-1)p$), and then divide by the determinant of $I-M(0)$ (i.e., the product of the eigenvalues of $I-M(0)$). We obtain

$$(I-M(0))^{-1}_{1,K} = \frac{(-1)^{K-1}\cdot(-1)^{K-1}(K-1)!\,p^{K-1}}{(1+p)(1-p)(1-2p)\cdots(1-(K-4)p)(1-(K-3)p)(1-(K-1)p)}.$$

By equation (16) we need to multiply this expression by $1-H = 1-(K-1)p$ and then divide by $S-K$; this yields the theorem. □

5 History & Future Work

Brian Noble inspired this problem with a question during the second author's dissertation proposal in November 2000. This paper reached roughly its present form in May 2005. Recently we have become optimistic that further progress is possible, and we welcome collaboration on this problem.

References

[1] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS96)*, December 1996. Reference [2] is longer and older.

[2] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. Technical Report TR-96-11, Boston University Computer Science Department, 1996. `http://www.cs.bu.edu/techreports/`.

[3] Oleg Ivanovich Aven, Edward Grady Coffman, and Yakov Afroimovich Kogan. *Stohastic Analysis of Computer Storage*. D. Reidel Publishing Company (member of Kluwer Academic Publishers Group), 1987. ISBN 90-277-2515-2.

[4] B. T. Bennett and V. J. Kruskal. LRU stack processing. *IBM Journal of Research and Development*, 19(4):353–357, July 1975.

[5] Terence Kelly and Daniel Reeves. Optimal Web cache sizing: Scalable methods for exact solutions. *Computer Communications*, 24:163–173, February 2001. `http://ai.eecs.umich.edu/~tpkelly/papers/`.

[6] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 9(2):78–117, 1970.

[7] Frank Olken. Efficient methods for calculating the success function of fixed space replacement policies. Technical Report LBL-12370, Electrical Engineering and Computer Science Department, University of California, Berkeley; and Computer Science and Mathematics Department, Lawrence Berkeley Lab, May 1981. This is the author's Berkeley Masters thesis.

[8] B. Ramakrishna Rau. Properties and applications of the least-recently-used stack model. Technical Report CSL-TR-77-139, Digital Systems Laboratory, Department of Electrical Engineering and Computer Science, Stanford University, May 1977.

[9] Jeffrey R. Spirn. Distance string models for program behavior. *Computer*, 9(11):14–20, November 1976.

[10] James Gordon Thompson. Efficient analysis of caching systems. Technical Report UCB/CSD 87/374, Computer Science Division (EECS), University of California at Berkeley, October 1987. This is the author's Ph.D. dissertation.

[11] J. van den Berg and D. Toswley. Properties of the miss ratio for a 2-level storage model with LRU or FIFO replacement strategy and independent references. *IEEE Transactions on Computers*, 42(4):508–512, April 1993.

[12] Sarut Vanichpun and Armand M. Makowski. The output of a cache under the independent reference model: where did the locality of reference go? *ACM SIGMETRICS Performance Evaluation Review*, 32(1), June 2004.