Cache Replacement Algorithms for the Renewal Arrival Model

Kaisa Kyläkoski and Jorma Virtamo
Laboratory of Telecommunications Technology
Helsinki University of Technology
P.O.B. 3000, FIN-02015 HUT, Finland
Email: kkylakos@luuri.hut.fi

July 1998

Abstract

Caches provide an important means for reducing latency times experienced by the users. This paper gives a short review of the most important arrival models and some basic results related to these. The three arrival models are the Independent Reference Model (IRM), the Least Recently Used (LRU) stack model and the renewal model. We also discuss the problem of good replacement algorithm and, more specifically, study the performance of a set of algorithms by simulation, assuming renewal arrival process.

Keywords – arrival model, replacement algorithm, cache

1   Introduction

WWW caches provide an important means for reducing network traffic and latency times experienced by the users. Accordingly, modeling the cache behavior has recently become an active field of research. A cache is basically defined by three factors: the cache size, the arrival process of document requests and the cache replacement algorithm.

Arrival models were studied already in the 1960's and 1970's in the context of program behavior. In these studies an arrival model described how the program referenced the pages in its memory. The two basic arrival models are the IRM (Independent Reference Model) and the LRU (Least Recently Used) stack model. IRM describes the request string as a sequence of i.i.d. random variables whereas the LRU-stack model assumes that when the pages are ordered in a stack according to their last reference times, the most recently referenced page being on the top of the stack, then the stack depths of the references form a sequence of i.i.d. random variables. The LRU model has the important property of locality: a recently referenced page is more likely to be referenced again in the near future. On the other hand, in this model, all the pages are statistically indistinguishable.

The third arrival model, renewal model, has not been studied as extensively as the two earlier ones. It tries to combine the features of the other models: locality and page differentiation. The renewal model assumes that interreference times to a page i are i.i.d. variables (different for different pages) and that the renewal processes for all the pages are independent. The IRM is obtained as a special case with exponentially distributed intervals.

Because of the complexity of the renewal model no optimal replacement algorithm is known for it. In this paper we use simulation to examine the question of the best replacement algorithm for reference strings created by the renewal model.

The next two sections present the central notations and concepts. The following sections give a short review of the most important arrival models and some basic results related to these. Finally we address the problem of good replacement algorithm, assuming renewal arrival process. Specifically, the performance of a set of algorithms in studied by simulation.

2 Definitions

The arrival of requests is expressed as a request string $R_t = r_1, r_2, r_3, \dots, r_t$, where r_t denotes the page requested at time t. There exists a given set of pages $X = 1, 2, \dots, n$ and a cache memory able to accommodate m pages. If $r_t = i$, it is understood that page i is requested at time t. A cache miss occurs if at the time of the request the page is not in the cache.

A locality set is defined as a subset of pages which has the property that if they are kept in the cache the frequency of misses remains relatively low. The locality set varies in time, and it is therefore desirable for a replacement algorithm to have the following two properties, which are in part contradictory: The replacement algorithm should retain in the cache the pages of the locality set, and it should rapidly update memory when the locality set is changed.

A sequence of LRU (least recently used) stacks $s_0, s_1, \dots, s_t$ can be associated with the reference string $r_1, r_2, \dots$, [5]. The stack s_t is the n-tuple $(D_1, D_2, \dots, D_n)$ in which D_i is the ith most recently referenced page at time t. If d_t is the position of the requested page r_t in the stack s_{t-1}, then, associated with the request string, there exists the distance string $\delta_t = d_1, d_2, \dots, d_t$.

3 Replacement algorithms

In addition to the parameter of cache memory size, an important design factor is the replacement algorithm. A replacement algorithm defines the set of pages in the cache. In particular, it determines which pages are to be replaced, when necessary, in order to make room for an incoming page. The efficiency of the cache is determined by the number of requests it can satisfy and depends thus heavily on the chosen replacement algorithm.

With a fixed size cache, the minimum number of cache misses is achieved by an algorithm that always replaces that page in the cache which will not be requested for the longest period of time [2]. However, such an algorithm is unrealizable, since advance information is normally not available on future page requests. So, the problem is the identification of algorithms that will give an acceptably high frequency of references to the cache for the greatest variety of user behavior.

A demand replacement algorithm brings pages into the cache only when they are requested. Once in memory, a page remains there until the replacement algorithm decides to remove it. Once removed, it will not reenter the cache until it is next requested.

A wide variety of possible demand replacement algorithms exists. The physically realizable ones use the observed historical information, either the history of the most recent

use or the history of their absence and presence in memory. There are also algorithms that do not use any information about memory usage.

Optimal replacement algorithm can be defined as the one that minimizes the expected frequency of requests to the original sources. A heuristic criterion, the expected forward distance criterion, is a rule which chooses for replacement the page in the cache having the longest expected time until the next request. Using this rule does not necessarily constitute an optimal replacement algorithm but for some basic models it does.

4   Independent Reference Model (IRM)

The independent reference model [1] is the simplest arrival model. It describes the request string $R_t = r_1, r_2, r_3, \ldots$, as a sequence of independent, identically distributed random variables with the probability distribution

$$P(r_t = i) = \beta_i, \quad \text{for } i = 1, 2, \ldots n, \quad \text{with } \sum_{i=1}^{n} \beta_i = 1, \tag{1}$$

where the pages are indexed $1, 2, \ldots, n$.

As the request string is a discrete parameter independent process, it is clear that the interval between two successive requests to page i, or interreference interval X of page i, is geometrically distributed with parameter β_i and mean $1/\beta_i$,

$$P(X = k) = \beta_i (1 - \beta_i)^{k-1} \quad \text{for } k = 1, 2, \ldots \tag{2}$$

The optimal replacement policy for IRM is to keep in the cache the m pages with the highest referencing probabilities [1]. This is the same set of pages as the one with the shortest expected times until next reference.

The most natural counterpart of the IRM in continuos time is given by the superposition of n independent Poisson processes, one for each page, with parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$. The arrival rate λ_i corresponds to the request probability

$$\beta_i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}. \tag{3}$$

Notably, one would find it logical to assume that the referencing of a page becomes less probable as the time since last reference grows. Since exponential distribution is memoryless, this property can not be described with this model.

5   Least Recently Used (LRU) Stack Model

The second commonly used arrival model is the LRU stack model. This section presents the model and some analytical results for it.

A sequence of LRU stacks $s_0, s_1, \ldots, s_t$ and the distance string $\delta_t = d_1, d_2, \ldots, d_t, \ldots$ can be associated with a reference string $r_1, r_2, \ldots,$. The LRU-stack model [8] assumes that the distance string is a sequence of independent identically distributed random variables with the probability mass function $\{a_i\}$

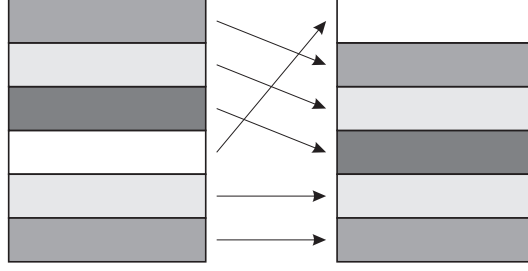$$P(d_t = i) = a_i, \quad \text{for } i = 1, 2, \ldots, n \quad \text{with } \sum_{i=1}^{n} a_i = 1. \tag{4}$$

Figure 1: A LRU stack update: the page at distance 4 from the top is referenced and brought to the topmost position.

The distribution function is then

$$P(d_t \leq i) = A_i = \sum_{j=1}^{i} a_i, \quad \text{for } i = 1, 2, \ldots, n. \tag{5}$$

While the distance string is an independent process in the LRU stack model, the corresponding request string is a dependent process. In some cases a locality condition $a_1 \geq a_2 \geq \ldots \geq a_n$ has been added to the model. When this condition is in force, the recently referenced pages have a higher probability of being referenced again.

The movement of a tagged page through the LRU stack is a discrete parameter Markov chain. The position of the page in stack s_{t+1} is determined by the next request r_{t+1} and the position of the page in stack s_t, but not its position in previous stacks. The distribution of the time X between successive requests for the tagged page can be calculated based on the transition probabilities. One way is to use generating functions in an iterative manner on the Markov chain (the details are omitted here). The probability distribution is

$$P(X = 1) = a_1, \tag{6}$$

$$P(X = i) = \sum_{j=1}^{i-1} a_{j+1} \prod_{w=1}^{j} (1 - A_w) \sum_{i=1}^{j} \frac{A_i^{i-2}}{\prod_{w=1, w \neq i}^{j} (A_i - A_w)} \tag{7}$$

$$= \sum_{j=1}^{i-1} a_{j+1} \prod_{w=1}^{j} (1 - A_w) \sum_{b_1 + \cdots + b_j = i-j-1} A_1^{b_1} \cdots A_j^{b_j}. \tag{8}$$

The first form, Eq. (7), is computationally impractical (and equivalent to the convolution form presented in [7]) since the denominator consists of a product of many small numbers. The second form, Eq. (8), is intuitively understandable and represents the sum over all possible paths such that the page on the top of the stack at time $t=0$ is referenced next at time $t = i$. The first sum is over those positions which are possible returning points to the position 1. The second part is the probability of reaching the return point and the third part is the probability of the necessary loops, that bring the length of path to i.

While the form presented in Eq. (8) represents a practical way to calculate the distribution, the easiest way of calculating the interreference distribution is with a matrix formulation. Consider the following partition of the transition probability matrix P of
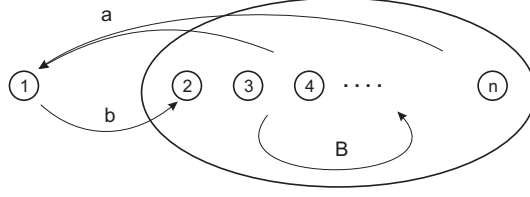
Figure 2: Figurative description of equation (13).

the position of the tagged page

$$P = \begin{bmatrix} a_1 & b^T \\ a & B \end{bmatrix}, \qquad (9)$$

with

$$b^T = \begin{bmatrix} (1 - A_1) & 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad (10)$$

$$B = \begin{bmatrix} A_1 & 1 - A_2 & 0 & \cdots & 0 \\ 0 & A_2 & 1 - A_3 & \cdots & 0 \\ 0 & 0 & A_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_{n-1} \end{bmatrix}, \qquad (11)$$

$$a^T = \begin{bmatrix} a_2 & a_3 & a_4 & \cdots & a_n \end{bmatrix}, \qquad (12)$$

with the aid of these the solution for $i > 1$ can be written as

$$\mathrm{P}(X = i) = b^T B^{i-2} a, \quad i > 1. \qquad (13)$$

In Eq. (13) b^T represents the step of exiting position 1, B^{i-2} represents the $i-2$ steps between the other positions and a the return to position 1 (see Figure 2).

In [9] a simple expression for the expected time until the next request for a page currently in a given stack position i is given as

$$T(i) = \frac{n - i + 1}{1 - A_{i-1}}, \quad \text{for } 1 \leq i \leq n, \quad \text{with } A_0 = 0. \qquad (14)$$

Thus the expectation for the interreference interval is $T(1) = n$.

Generally the interreference distribution is not geometric. The locality condition $a_1 \geq a_2 \geq \ldots \geq a_n$ is reflected in a sharper peak for short intervals and heavier tail for longer intervals. A notable feature of the LRU model is that all pages are statistically indistinguishable. The interreference time distribution is the same for all the pages and the probability distribution of each page in the stack is uniform. Indeed, the IRM with identical reference probabilities, $\beta_i = \beta$, is equivalent to a LRU model with uniform distance distribution.

The optimal replacement algorithm for the LRU stack model is LRU itself [3]. The algorithm keeps the set of pages cached that have the highest reference probabilities but also implements the heuristic of the expected forward distance criterion when the locality condition is satisfied. That is to say, this is also the set of pages with the shortest expected times until next reference.

6 Renewal model

In the previous sections we saw that the LRU stack model has the desirable property of locality but, unrealistically, the model does not make any distinction between different pages., i.e. in the long run each page is referenced with equal frequency. On the other hand, in the IRM different pages have different reference probabilities but there is no locality effect. As an alternative to these models, the renewal model, was introduced in [6] and suggested in [9]). The renewal model attempts to combine best features of the IRM and LRU stack model: the locality property and the different behavior of different pages.

The renewal model is best introduced in the context of continuous time. It can be viewed as a simple generalization of the IRM obtained by replacing the Poisson arrival processes for each page by a more general independent renewal processes. For page i the interreference time is assumed to obey distribution $F_i(t)$ (with pdf $f_i(t)$) and to be independent of other interreference times of page i or those of other pages.

For the continuos IRM, the probability of referencing some page i during the small time interval Δt is constant and equal to $\beta_i \Delta t$. In renewal theory β_i is known as the instantaneous renewal rate, in the present case the term immediate reference density (ird) is logical. With exponentially distributed interreference intervals the ird β_i is independent of the current backward distance of page i, that is, the time interval between the last reference to page i and the current time. The principle of locality implies that the larger the current backward distance of page i, the smaller its reference probability. The ird β_i should not be constant but a decreasing function of the backward distance.

By defining the form of $\beta_i(t)$ the probability density function $f_i(t)$ and the cumulative distribution $F_i(t)$ are uniquely determined.

$$
\begin{aligned}
\beta_i(t) &= \lim_{\Delta t \to 0} \frac{P(T < t + \Delta t \mid T > t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{F(t + \Delta t) - F(t)}{\Delta t R(t)} \\
&= \frac{f(t)}{R(t)} = -\frac{R'(t)}{R(t)} = -\frac{d}{dt} \ln R(t),
\end{aligned}
\tag{15}
$$

where $R(t) = 1 - F(t)$. Thus defining

$$
B(t) = \int_0^t \beta_i(x) dx,
\tag{16}
$$

we have

$$
f_i(t) = \beta_i(t) e^{-B(t)}, \quad \text{and} \quad F_i(t) = 1 - e^{-B(t)}.
\tag{17}
$$

The expected length of the interreference interval is

$$
m_i = \int_0^\infty (1 - F_i(t)) dt = \int_0^\infty e^{-B(t)} dt.
\tag{18}
$$

The expected time until next reference when time t has passed since the last reference is (see, e.g. [4])

$$
m_i(1 + B(t)) - t.
\tag{19}
$$

If page i is assumed to have time t_i since the last reference and a known form of immediate reference density $\beta_i(t_i)$ or its integral $B_i(t_i)$ the expected time until next reference can be calculated. The pages can be ordered accordingly. Alternatively, the pages

could be ordered by their immediate reference density. Both orders depend on the form of the immediate reference densities as well as on the updated times, so the order of the unreferenced pages is not necessarily conserved. This makes it difficult to determine an optimal replacement policy.

Using the first ranking of expected time until next reference emulates the popular heuristic of replacing the page with the longest time till next reference. The second ranking, ordering the pages according to their reference probabilities, is analogous to the optimal policy of the IRM model. In [1] this was recognized as a reasonable approximation to optimality when the respective order of the page reference probabilities is slowly varying.

7   Replacement algorithm and simulation

In the following we will construct a possible way of utilizing the renewal model. After constructing a replacement algorithm its efficiency is tested by a simple simulation. In particular, we assume that the interreference times obey the Weibull distribution

$$F_i(t) = 1 - e^{-\alpha_i t^{\gamma_i}} \tag{20}$$

with mean

$$m_i = \frac{\Gamma(1 - \frac{1}{\gamma_i})}{\alpha_i^{\frac{1}{\gamma_i}}}. \tag{21}$$

The corresponding probability density function $f_i(t)$ and immediate reference density $\beta_i(t)$ are

$$f_i(t) = \alpha_i \gamma_i t^{\gamma_i - 1} e^{-\alpha_i t^{\gamma_i}}, \quad \text{and} \quad \beta_i(t) = \alpha_i \gamma_i t^{\gamma_i - 1}. \tag{22}$$

The exponential distribution can be viewed as a special case of the Weibull distribution for which $\beta_i = 1/m_i$ and $\gamma_i = 1$.

By assuming that γ_i has the same value for all pages, $\gamma_i = \gamma$, α_i can be estimated with the maximal likelihood method as

$$\hat{\alpha}_i = \frac{k}{\sum_{i=1}^{k} t_i^{\gamma}}, \tag{23}$$

where the page i has been referenced k times with the observed interreference times $t_1, t_2, \ldots, t_k$. With this estimate and an estimate of the mean (the average of interreference times) we can calculate the reference probability and expected time until next reference needed for the implementation of the two stack algorithms suggested earlier.

---

0. INITIALIZATION

Select universal value for γ

Initialize for each page
    - number of references so far $k^{(i)} = 0$,
    - $\alpha_0^{(i)}$,
    - the average of interreference intervals so far $T_0^{(i)}$
    - time of last reference $t_0^{(i)} = 0$

Upon a reference for page i at time t

1. UPDATE

- $\alpha_{k+1}^{(i)} = \frac{k+1}{\frac{k}{\alpha_k^{(i)}}+(t-t_{k+1}^{(i)})^\gamma}$

- $T_{k+1}^{(i)} = \frac{kT_k^{(i)}+(t-t_{k+1}^{(i)})}{k+1}$

- $t_k{}^{(i)} = t$

- $k^{(i)} = k^{(i)} + 1$

2. REARRANGE

Rearrange list with the chosen key

    A: expected time until next reference

    B: reference probability

3. REMOVE

If necessary remove the page [1] with

    A: the longest expected time until next reference

    B: the smallest reference probability

In order to examine the question of the near optimal stack algorithm a simulated cache with a request string corresponding to the renewal model was created. In the simulated case each of the n pages had a Weibull distributed interreference times with a common parameter γ. The means were chosen from a uniform distribution (since γ is a constant, this also sets the values of the α_i). The initial values $\alpha_0{}^{(i)}$ and $T_0{}^{(i)}$ were chosen as the average of the parameter values of the created page population. The times of first references were generated from the forward recurrence time distribution.

The page population size used was 100 and the length of the request string 3000. Cache sizes ranged from 5 to 30 pages. Each of the algorithms compared used a different key to arrange the pages into a stack. The keys compared were

- number of references

- time since last reference (real time not virtual)

- mean of interreference times

- reference probability β_i calculated with the actual α_i

- expected time until next reference calculated with the mean and actual α_i

- average of interreference times

- reference probability β_i calculated with the estimated $\hat{\alpha}_i$

- expected time until next reference calculated with the average and estimated $\hat{\alpha}_i$

The results of the simulations were not quite what was expected. Figure 3 shows a representative example. Hit rates for the keys "time since last reference" and "reference probability" are so close to each other as to be indistinguishable. Next best is the key "expected time" and the order of the rest is dependent on the cache size.

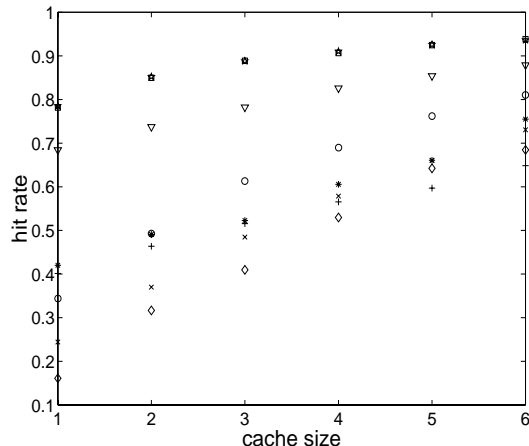[1] This can be the incoming page which is really not cached at all.

Figure 3: Results of a simulation with renewal model request strings and six different cache sizes and eight different keys: □ time since last reference, ⋆ reference probability β calculated with the actual α, ▽ expected time until next reference calculated with the mean and actual α, ○ reference probability β calculated with the estimated α, ∗ average of interreference times, × number of references, + expected time until next reference calculated with the average and estimated α, ◇ mean of interreference times

With the probabilistic keys that used the actual parameter values reference probability achieves clearly better results than the alternative expected time until next reference. The results obtained with the estimated reference probability are also fairly good. However time since last reference, which requires no estimation or prior knowledge performs better or as well as all of these.

If indeed the renewal model and Weibull distribution are descriptive of the traffic, these results imply that instead of using the specific keys, equally good results can be achieved with a simpler model.

In the simulation the data was updated and gathered for all the pages, whether they were in the cache or outside. This is not a option in reality. Notably the hit rates for the three best keys were high even for the smallest cache, indicating that references were concentrated to a small part of the pages.

8   Conclusions

Statistical arrival models provide the only way to examine the performance of caches analytically. In this paper we have reviewed three basic models. With simple models such as IRM and LRU stack model, some aspects of reality are not modeled but analytical results can be derived. The renewal model, which is more flexible, is too complicated for this. The simulation study presented in this paper shows that it does not follow the same rules that the more basic models do. The heuristic of keeping in the cache the pages with the highest reference probabilities performs better than the heuristic of longest expected times until next reference.

References

[1] A.V. Aho, P.J. Denning and J.D. Ullman. Principles of Optimal Page Replacement.

Journal of the ACM, vol. 18, no. 1, 1971, pp. 80-93.

[2] L.A. Belady. A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal, vol. 5, no. 2, 1966, pp. 78-101 .

[3] E.G. Coffman, Jr. and P.J. Denning. Operating Systems Theory. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1973.

[4] P.G. Harrison and N.M. Patel. Performance Modelling of Communication Networks and Computer Architectures. Addison-Wesley. 1992.

[5] R.L. Mattson, J. Gescei, D.R. Slutz and I.L. Traiger. Evaluation Techniques for Storage Hierarchies. IBM Systems Journal, vol. 9, no.2, 1990, pp. 79-117.

[6] H. Opderbeck and W.W. Chu. The Renewal Model for Program Behavior. Siam Journal on Computing, vol. 4, no. 3, 1975, pp. 356-374.

[7] A.J. Smith. Analysis of the optimal, look-ahead demand paging algorithms. Siam Journal on Computing, vol. 5, no. 4, 1976, pp. 743-757.

[8] J.R. Spirn and P.J. Denning. Experiments with Program Locality. AFIPS Conference Proceedings, vol. 41, 1972, pp. 611-621.

[9] J.R. Spirn. Program behavior: models and measurements. Elsevier 1977.