An Unifying Replacement Approach for Caching Systems

Kai-Hau Yeung, Kwan-Wai Ng, and Kin-Yeung Wong

Abstract: **A cache replacement algorithm called probability based replacement (PBR) is proposed in this paper. The algorithm makes replacement decision based on the byte access probabilities of documents. This concept can be applied to both small conventional web documents and large video documents.**

The performance of PBR algorithm is studied by both analysis and simulation. By comparing cache hit probability, hit rate and average time spent in three systems, it is shown that the proposed algorithm outperforms the commonly used LRU and LFU algorithms. Simulation results show that, when large video documents are considered, the PBR algorithm provides up to 120% improvement in cache hit rate when comparing to that of conventional algorithms.

The uniqueness of this work is that, unlike previous studies that propose different solutions for different types of documents separately, the proposed PBR algorithm provides a simple and unified approach to serve different types of documents in a single system.

Index terms: **caching systems, cache replacement, video**

I. Introduction

Cache is widely viewed as an effective way to improve Internet performance. The use of it reduces not only retrieval latency, but also networks traffic and server load. Cache performance is mainly determined by the effectiveness of cache replacement algorithms. Among them, LRU (Least Recently Used) and LFU (Least Frequently Used) algorithms are most popular. LRU evicts the document which was requested the least recently, whereas LFU evicts the document which is accessed least frequently. However, while they work well in traditional computer systems, they do not provide best performance in the Internet environment due to the difference in access characteristics. Therefore, cache replacement algorithms exclusively designed for Internet systems are needed. In this paper, we propose a new replacement algorithm called probability based replacement (PBR).

PBR algorithm is particularly suited for systems that have to handle a large number of continuous media streams (i.e., video and audio). Since the conventional Web caching algorithms are designed for Web documents such as text and image only, they are not appropriate for those large media. Storing these large document entirely will consume huge cache space and hence seriously affect the overall performance of the cache. In later part of this paper, we shall discuss how the PBR algorithm can be extended to handle these large documents.

Many cache replacement algorithms for conventional Web documents have been proposed. LRU-threshold [1] performs as same as LRU, except documents with sizes larger than a certain threshold are never cached. Hyper-G [2] is a refinement of LFU with last access time and size being considered. Site-based LRU [3] makes replacement decisions based on the site information instead of object information. It is similar to LRU but it purges all documents belonging to the same site instead of a single document when replacement is needed. On the other hand, Low Inter-reference Recency Set (LIRS) [4] is a general and efficient replacement algorithm. It was proposed to address the limitations of LRU by using recency to evaluate Inter-Reference Recency (IRR) of accessed blocks for making a replacement decision. Besides, detection-based adaptive replacement (DEAR) [5], a probabilistic replacement algorithm, was proposed for buffer cache management. It has been proved to effectively enhance the performance of caching of disk blocks in operating systems. Other algorithms can be seen in [6].

Caching schemes for video documents have also been studied widely. In [7], resource-based algorithm which considers cache disk bandwidth and space, is proposed. In [8], a partial video sequence caching scheme addressing issues of heterogeneous clients is proposed. Similar to the idea of [8], prefix caching scheme is proposed in [9] to store the prefix of video documents so as to reduce client latency and perform work-ahead smoothing. Prefix caching has also been studies in [10], where the optimal prefix placement scheme for video cache was analytically determined. On the other hand, [11] present a caching architecture and associated cache management algorithms to accelerate the streaming media delivery. By using the technique of partial caching, it is able to reduce service delay and improve overall stream quality.

As seen from the above discussions, most previous works on Web caching systems focus either on conventional Web documents or on large documents. The uniqueness of our study is, however, on proposing a PBR algorithm that

Manuscript received June 13, 2006 and revised November 11, 2007.

This research was supported by Macau Science and Technology Development Fund numbered 099/2005/A and City University Strategic Research Grant numbered 7001941.

K. H. Yeung and K. W. Ng are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (email: eeayeung@cityu.edu.hk).

K. Y. Wong is with the Computer Studies Program, Macao Polytechnic Institute, Macao (email: kywong@ipm.edu.mo)

elegantly provides a simple and unified approach to serve both types of documents in a single system. The algorithm is also well studied by simulation and analysis. Results show that it outperforms the LRU and LFU algorithms by giving 85% and 50% improvement in cache hit rate respectively. The rest of the paper is organized as follows: In section II, we shall discuss the proposed PBR algorithm and its performance study. In Section III, we first discuss the problems of caching video documents. We then discuss how the PBR algorithm can be extended for caching these documents. At last, we conclude the paper in Section IV.

II. PBR Algorithm for conventional web objects

In this section, we propose a probability based replacement (PBR) algorithm. The mechanism and the performance analysis on it will also be discussed.

A. Mechanism

Consider a generalized Internet system which contains N unique documents. The documents have different sizes and we denote the size of document i in bytes as s_i. Let C be the total cache size available in the server, and $C > s_i$, for all i. Let p_i be the probability that a user will access document i, and

$$\sum_{i=1}^{N} p_i = 1. \tag{1}$$

Since p_i are usually unknown to the system, they can be estimated from the past history of document accesses. Let h_i be the cache hit probability for document i. The hit rate of the system can then be obtained by

$$HR = \sum_{i=1}^{N} h_i p_i . \tag{2}$$

Denote b_i be the *byte access probability* of document i. It is defined as

$$b_i = \frac{\text{access probability of document } i}{\text{size of document } i} = \frac{p_i}{s_i} . \tag{3}$$

The proposed (PBR) algorithm is based on byte access probability. It is to always cache documents with higher byte access probabilities b_i first. The design rational behind is explained as follows. Consider a particular situation that Z highest b_i's documents, namely d_1, ..., d_Z, are already stored in the cache as shown in Fig. 1. A set of documents with a total size s in the cache is going to be replaced by a new set of incoming documents having the same total size s. Since the summation of the b_i's for the original set in the cache must be larger than that for the new set, the overall hit rate of the cache after the replacement will be lowered. Therefore, the hit rate of the cache will be maximum if no replacement such as the one mentioned above is made. In this case, the cache always stores the documents with highest b_i's. Note that LRU and LFU algorithm will always replace documents in the cache when a new document access arrived. This will not ensure the documents with higher byte access probabilities in the cache.

The details of replacement (PBR) algorithm is as follow. This algorithm first requires the documents in the cache are sorted in ascending order according to their byte access probabilities to form a probability list (see Fig. 2). As a result, the documents with largest b_i is put at the bottom of the list. The byte access probabilities of the documents can be updated periodically, and the list will be a rather static one if the b_i's of the documents change slowly. Let $\overline{b_i}$ and $\overline{s_i}$ be the byte access probability and the size of the ith document in the list, respectively. Obviously, $\overline{b_i} > \overline{b_j}$ if $i>j$. when a replacement decision has to be made, says a new document (not in the cache) with byte access probability $\overline{b_{new}}$ and size $\overline{s_{new}}$ is accessed, the new document will only replace the first X documents at the top of the cache if and only if

$$\overline{b_{new}} \geq \sum_{k=1}^{X} \overline{b_k} \tag{4}$$

where X is the smallest number of documents which have a total size greater than $\overline{s_{new}}$ or

$$\sum_{k=1}^{X} \overline{s_k} \leq \overline{s_{new}} \leq \sum_{k=1}^{X-1} \overline{s_k} . \tag{5}$$

By using this algorithm, the documents with relatively higher byte access probabilities will therefore always be cached.

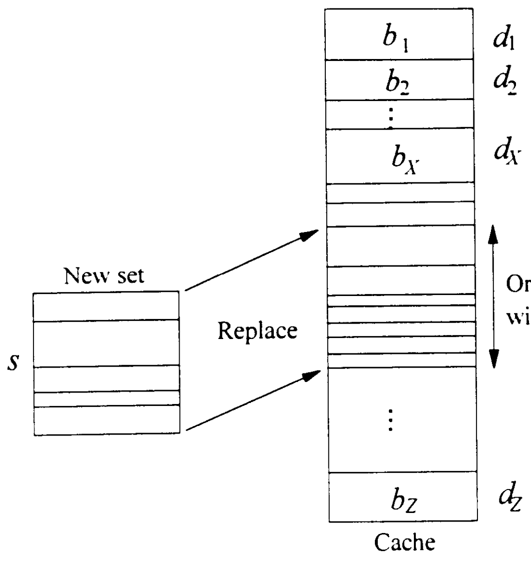


Fig. 1. Cache Optimality.

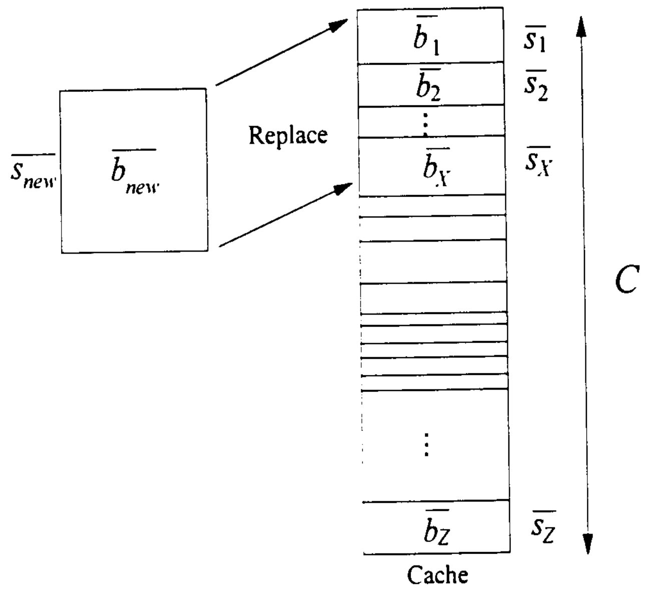


Fig. 2. Byte access probability list.

B. Performance Analysis

B.1 Hit Rate

Assume that the cache is holding Z documents with highest b_i's as shown in Fig. 2. Maximum number of Z is always chosen under the condition that the total size of the documents do not exceed the size of the cache. That is,

$$\sum_{i=1}^{Z} \overline{s}_i \leq C\,. \tag{6}$$

As the documents in the cache are sorted by their byte access probabilities, the cache hit probability of document i can be easily obtain as

$$h_i = \begin{cases} 1 & b_i \geq \overline{b}_1 \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

The hit rate of PBR algorithm can then be obtained by substituting Equation (7) into Equation (2).

B.2 Average Document Access Time

An Internet system can be modeled by a M/G/1 queuing system. The arrival process of this queuing system can be assumed to be Poisson with rate λ as the number of users accessing an Internet system is usually large. Arrived document requests are served one by one in a FCFS manner with a mean service rate μ (see Fig. 3). We denote the mean service time by T_s, where $T_s = 1/\mu$. Let μ_i be the service rate for document i. It is reasonable to assume that the service time of a document is proportional to its size, i.e., $1/\mu_i \propto s_i$. Suppose the average service time for the hit documents can be shortened by a speed-up factor ϕ when compared to that of the miss documents. Then we have,

$$T_s = \sum_{i=1}^{N} p_i \left[h_i \frac{\phi}{\mu_i} + (1-h_i)\frac{1}{\mu_i} \right]. \tag{8}$$

Having obtained the mean service time T_s, the average queuing time of document requests, denoted as T_W, can be obtained by the P-K formula [12]. The average sojourn time of the document requests T is equal to

$$T = T_w + T_s\,. \tag{9}$$

C. Simulation and Numerical Examples

To study the performance of PBR algorithm, a trace-driven simulation was run. In the simulation, a 10-day log from the WWW Server of the Department of Electronic Engineering, City University of Hong Kong was used. The log contains 95031 accesses and the total access size is about 1GB. The server has totally 14366 unique documents and the accesses of these documents were ranged from once to 2198 times. The total size of these 14366 documents is 298MB. TABLE I shows the top 30 documents accessed in this period, and their corresponding number of accesses and sizes. Although the total size of these 30 documents only contributes 0.04% of the total size of the WWW database, more than 17% of the accesses were targeted on them in the 10-day period. This implies that even a small cache can significantly improve the performance of a WWW server.

Fig. 4 plots the size distribution of the documents. The figure shows only the statistics for document sizes smaller than 100 kilobytes (it represents almost 98% of the documents). From the figure, we observe that most of the documents are small, and 74% of the documents have sizes smaller than 10 kilobytes. Fig. 5 plots the access frequencies of the documents with sizes smaller than 50KB. From the figure, we observe that most of the document accesses target on small documents with sizes smaller than 10 Kbytes. Another observation is that the access frequency of a document is roughly proportional to the inverse of its size ($1/s_i$). Similar observation was made by Bestavros as reported in [13].

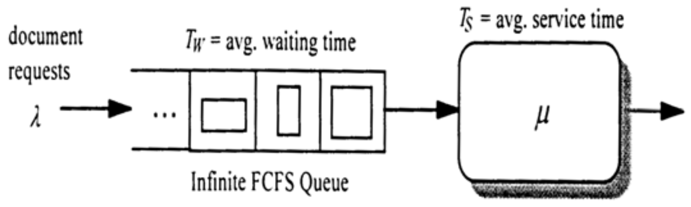


Fig. 3. M/G/1 queuing model for the Internet system.

TABLE I
TOP 30 DOCUMENTS FROM CITYU-EE WEB SERVER IN THE 10-DAY PERIOD.

Rank	Document (/=http://www.ee.cityu.edu.hk)	Document size	No. of access
1	/	3788	2198
2	/panel.gif	7677	1343
3	/tit1e20.gif	6942	1303
4	/gif/newtiny.gif	141	1121
5	/gif/eetitle.gif	1658	845
6	/people/people.html	1422	760
7	/gif/s green.gif	104	753
8	/gif/s_red.gif	326	743
9	/~95473199/f50front.jpg	29542	659
10	/gif/return.gif	1082	621
11	/gif/eetitle06.gif	1658	609
12	/gif/ hinese.gif	110	551
13	/~edap045/cgi-bin/video3.html	402	533
14	/people/people.gif	488	426
15	/image/bar/eebar06.gif	1690	320
16	/people/academic.html	6275	275
17	/people/all_student.html	14825	271
18	/~edap045/leung.html	2759	246
19	/image/icon/s green.gif	104	244
20	/~eelmpo/	3462	243
21	/gif/cyancube.gif	346	240
22	/gif/purcube.gif	374	240
23	/gif/redcube.gif	235	237
24	/gif/grencube.gif	241	237
25	/~eelmpo/ee2235/ee2235.html	5207	232
26	/~edap064/mpeg/beginnerguide.jpg	16137	?30
27	/~edap064/mpeg/videobitstream.gif	5625	225
28	/~edap064/mpeg/ videostreamdatahierarchy.gif	3857	222
29	/~edap064/mpeg/ Macroblockcomposition.gif	1088	221
30	/~uedap045/cgi-bin/video2.htm1	403	220

Computer simulation on LRU, LFU and PBR algorithms using the 10-day log was performed. ϕ is assumed to be 0.01 in our simulation. Fig. 6 plots the hit rates of the algorithms against the cache size. It can be observed from the figure that the analytical results for PBR, when assuming that $p_i \propto 1/s_i$, differ from the simulation results by at most 13%. Another observation is that when the cache size is small, PBR algorithm has 85% and 50% improvement in cache hit rate when compared to that of LRU and LFU respectively. When the cache size is large, PBR algorithm still maintains at least 26% and 18% improvement in cache hit rate over that of LRU ad LFU.

Fig. 7 plots the average document access time T against the server utilization when the cache size is 10% of the total sizes of the documents. From the figure, we observe that PBR algorithm again outperforms LRU and LFU algorithms under all server utilization. The maximum server utilization for LRU and LFU is only about 0.85 while that for PBR algorithm is over 0.95.

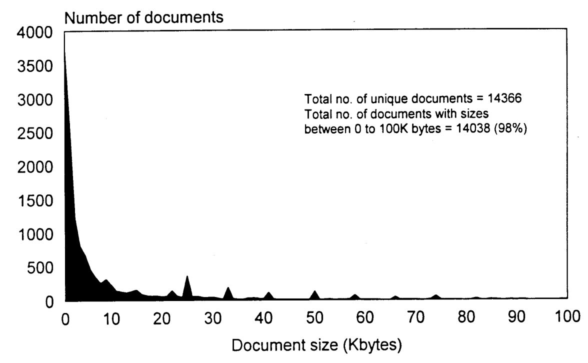


Fig. 4. Size distribution of the documents in the access log.

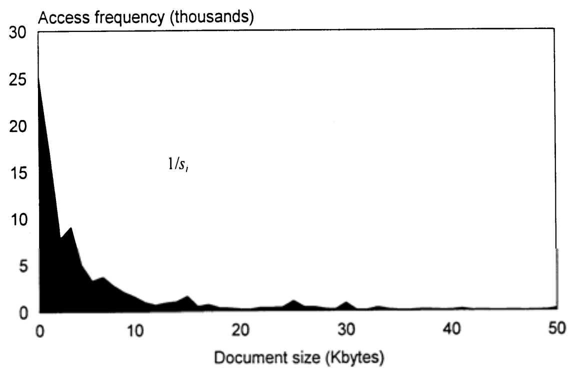


Fig. 5. Access frequency against document size.

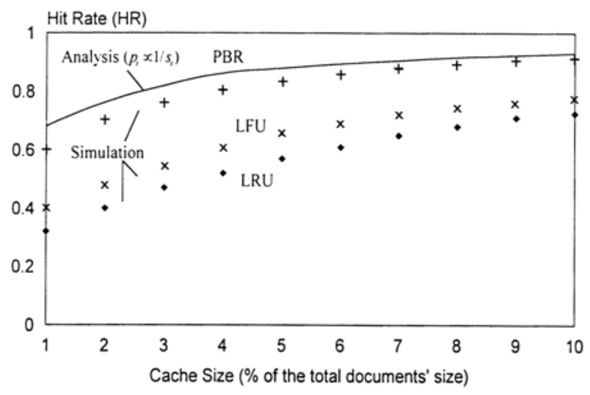


Fig. 6. Hit rate against cache sizes.

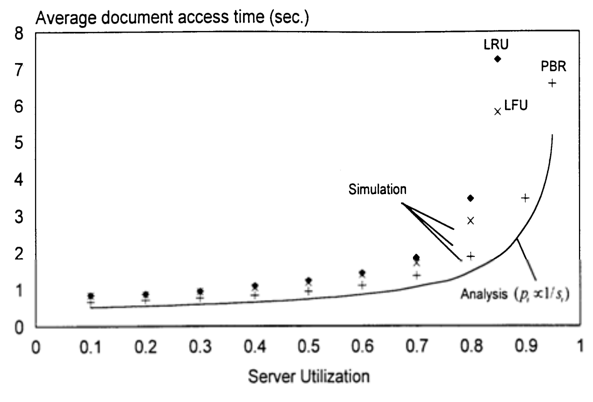


Fig. 7. Average document access time against server utilization (cache size is 10% of the total document size.).

III. A Unifying Approach with consideration of large objects

In section II, the concept of *byte access probability* and the basic of PBR algorithm for conventional web objects are introduced. In this section, the algorithm is extended to become a unifying approach with the consideration of large video objects.

A. Challenging of Caching Video Documents

As mentioned earlier, the basic accessing unit in the Internet environment is a document. When conventional replacement algorithms such as LRU or LFU are used, only the access frequency of each document is considered when making replacement decisions and the document size is not considered. Furthermore, LRU and LFU will always replace documents in the cache when new documents arrive. Due to these two reasons, a currently cold video document, which is usually large in size, may occupy the whole cache when it is accessed. When this happens, small but active documents are replaced from the cache and the overall cache hit rate will be lowered significantly. We shall observe the same problem when other types of large documents such as high-quality audio documents are accessed. When PBR algorithm is used, the replacement decision is controlled by the byte access probability b_i of each document. Unlike LRU and LFU algorithms, a newly accessed document may not replace documents in the cache if its b_i is not high enough. Since video documents will usually have smaller b_i's due to their large sizes, the chance of caching video documents is relatively small. The problem mentioned above is therefore solved when PBR algorithm is used.

In previous research works, the access probability p_i is usually estimated by one single constant for each document. However, this is often not the case for video documents. Web users may drop off a video at any time while watching it. They may also choose their favorite pop music in some music web pages by listening to just the first part of each music song. So the probability of access for the starting part of each video (or audio) may be higher than the ending part of the document. In general, *p_i is not uniform within a continuous media stream*. If the PBR algorithm discussed above is used to with caching those video documents, the byte access probability of each video document will be represented by one single b_i. This is not enough to reflect the byte access probability of each part of the video. Therefore, the PBR algorithm discussed in section II should then be extended for the cache systems considering video documents.

B. PBR Algorithm for Frequent Video Accesses

Consider a multimedia Internet system which contains text, image, audio and video documents. In this system, we assume that interactive features for video and audio accesses such as "Pause", "Fast Forward", "Fast Reverse" and "Fast Search" are not supported. Web users can only play an audio or video document from the beginning of the document, but can stop playing at anytime. Since the user can stop accessing a document at any time, p_i is not uniform within a video document. As discussed above, the cache will operate in far from optimal condition if only one single p_i, is used for each document. The solution to this problem is to segment video document into blocks, and use different bytes access probabilities (b_i) for these blocks. If we do this, a natural question arises for the right choice of block size. This issue is addressed in the following analysis.

Consider a video document i with size s, as shown in Fig. 8. The size of a block is assumed to be k bytes, and there are totally $[s_i/k]$ blocks. Let the byte access probability of the j^{th} byte of the document be $b_{i,j}$. The probability $b_{i,j}$ will be a decreasing function of j. This is because a user must play a video document from the beginning of the document but can stop playing at any time. Let $b_{i,m}$ be the byte access probability of the starting byte of the r^{th} block where $m = (r-1)k+1$. If we take this $b_{i,m}$ as the overall byte access probability of this block, the mean-square error for this block r when the block size is k is equal to

$$E_{k,r} = \frac{1}{k}\sum_{l=0}^{k-1}\left(b_{i,m+l} - b_{i,m}\right)^2 . \tag{10}$$

Consider the situation when the block is enlarged by one byte by including byte $m+k$.

The mean-square error for this enlarged block will be changed to

$$E_{k+1,r} = \frac{1}{k+1}\sum_{l=0}^{k}\left(b_{i,m+l} - b_{i,m}\right)^2 \tag{11}$$

or

$$E_{k+1,r} = \frac{1}{k+1}\left[kE_{k,r} + \left(b_{i,m+k} - b_{i,m}\right)^2\right]. \tag{12}$$

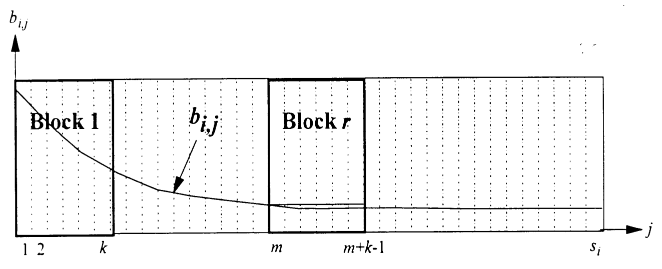


Fig. 8. A segmented video document.

Since $b_{i,j}$ is a decreasing function for any number n where $m \le n \le m+k-1$

$$\left(b_{i,m+k}-b_{i,m}\right)^2 > \left(b_{i,n}-b_{i,m}\right)^2 \quad (13)$$

This implies,

$$k\left(b_{i,m+k}-b_{i,m}\right)^2 > \sum_{n=m}^{m+k-1}\left(b_{i,n}-b_{i,m}\right)^2 \quad (14)$$

or

$$\left(b_{i,m+k}-b_{i,m}\right)^2 > E_{k,r}\,. \quad (15)$$

From Equations (12) and (15), we can thus conclude that

$$E_{k+1,r} > \frac{1}{k+1}\left[kE_{k,r}+E_{k,r}\right] \quad (16)$$

or

$$E_{k+1,r} > E_{k,r} \quad (17)$$

Notes that for any blocks starting from any byte m, the same result stated in Equation (17) is obtained. Therefore we can conclude that the mean-square error of a block increases as k increases. In other words, minimum mean-square error is obtained when the block size is as small as possible. However, when the block size decreases, the database in holding the byte access probability information will increase. Therefore, the minimum block size is limited by physical factors such as memory available and the processing requirement of database management.

Fig. 9 shows an example of cache replacement using PBR algorithm. Consider a particular situation that video document i, which is segmented into ten blocks with block size k, is being accessed by a user. The byte access probability of each block of this video is denoted as $b_{i,1}$, $b_{i,k+1}$, …, $b_{i,9k+1}$ as shown in the figure. Suppose that the first three blocks of the video are cached because their byte access probabilities are high enough. Note that the caching of this document rarely happens when it is not segmented because the byte access probability of the whole document is relatively much smaller. Segmenting videos will therefore in effect allow the beginning blocks of popular videos to be cached. This will certainly improve the overall caching performance.

Note that though video objects will be segmented, after segmentation, the same replacement approach based on the PBR algorithm will be applied for all objects. That is why it is called unifying approach. Also note that, thought large objects such as ZIP files exist in the Internet, they have to be downloaded as a whole. Therefore, segmentation technique should not be applied to them. This exceptional type of large file should be handled as a conventional web object. That is, if they are unpopular, they will be replaced with a smaller or a more popular object.

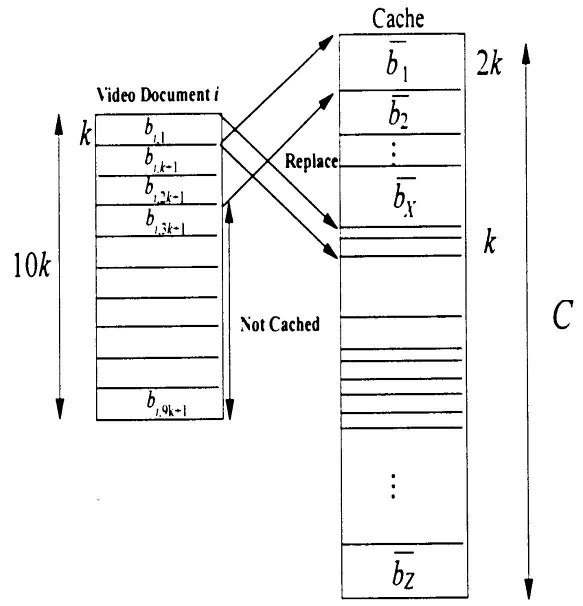


Fig. 9. Byte access probability list with segmented videos.

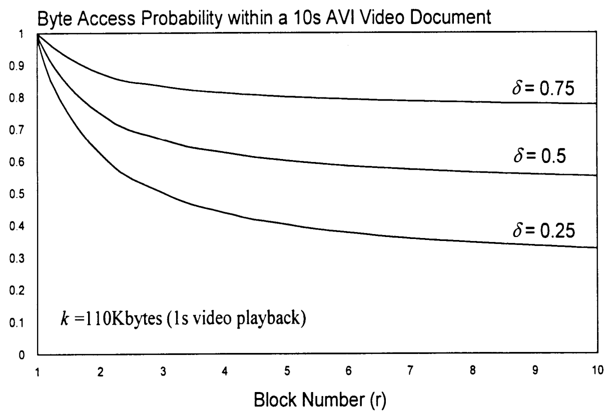


Fig. 10. Byte access probabilities of the blocks in a 10s AVI video document with variable δ's.

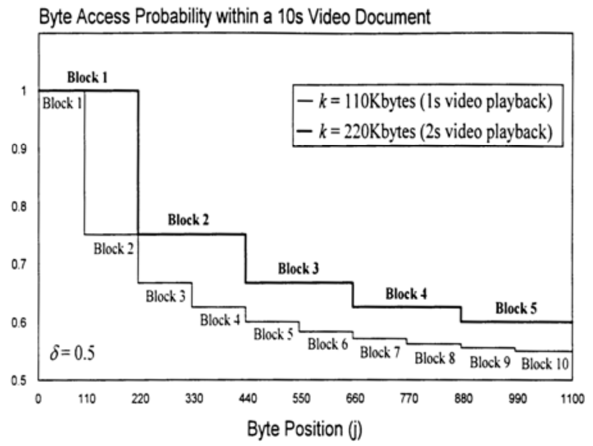


Fig. 11. Byte access probabilities of the blocks in a 10s AVI video document with variable k's.

TABLE II
VIDEO DOCUEMTNS GENERATION WITH DIFFERENT SIZE DISTRIBUTION.

Group	Size range (MB)	Number of video documents		
		Video proportion=10%		Video proportion=20%
		Size distribution =uniform	Size distribution $\propto 1/s_i$	Size distribution =uniform
1	0.5-1	270	660	610
2	1-2	270	330	610
3	2-3	270	220	610
4	3-4	270	165	610
5	4-5	270	132	610
6	5-6	270	110	610
	Total	1620	1617	3660

C. Simulation and Numerical Examples

To study the performance of the PBR algorithm with segmented videos, we assume that $b_{i,j}$ is a decreasing step function with the block number r, that is

$$b_{i,j} = \frac{p_i}{k}\left[\frac{1}{r}(1-\delta)+\delta\right] \tag{18}$$

where $r = [j/k]$ and δ is a factor reflecting the drop-off rate of users. When δ equals 0.75, 25% of the users will not watch the video until the end. In our study, we assume that (δ is constant for all video documents. We also assume that the minimum block size is 110 KB (which is about one second long of a 160x120 256K color AVI video). Fig. 10 and Fig. 11 plot the byte access probability function of a 10-second AVI video document with variable δ's and k's, respectively.

In order to demonstrate the advantages of video segmentation, simulation using the 10-day access log of the CITYLU-EE Web Server was run. To simulate an environment with frequent video accesses, additional video documents are generated to the log so that the number of video documents accounts for about 10% or 20% of the total number of documents in the server. The video documents are generated according to six size groups and two different size distributions (uniform and inversely proportional) in each group, as shown in TABLE II. We also use these two typical distributions to model the access probabilities of the video documents. Thus, there are all together four different conditions in our simulation, denoted as S1, S2, S3 and S4 as shown below:

S1 : Distribution of video size=uniform; p_i=1/N

S2 : Distribution of video size=uniform; $p_i \propto 1/s_i$

S3 : Distribution of video size$\propto 1/s$; $p_i \propto 1/N$

S4 : Distribution of video size$\propto 1/s$; $p_i \propto 1/s_i$

Having obtained the p_i's for all documents, the b_i's can be calculated by using Equation (3). For video documents, the $b_{i,j}$'s of each blocks can be obtained by using Equation (18). With a particular cache size, the byte access probability list is generated. In our simulations, the same M/G/1 queuing model described in Section II B is used. There are at least 5×10^4 arrivals in each simulation run. ϕ is assumed to be 0.01.

Note that the aim of our paper is not to design another algorithm which outperforms all the previous works in caching video objects. Instead, our aim is to provide a unified approach to handle the two types of objects (conventional and video) consistently. This could simplify the implementation of cache servers. Today, most implementations of cache servers use a single policy (typically, LRU and LFU) to handle all kinds of web object. Since our proposal, PBR, is also aimed at handling all kinds of web object, we compare the performance of PBR to that of LRU and LFU.

Fig. 12 plots the hit rate against the cache size for S1, S2, S3 and S4 with δ = 0.5, k=110KB and video proportion is 10% of all documents. From the figure, we have a number observations. The first one is that PBR with segmented videos provides the best throughput among the four algorithms. Taking S4 as an example, when the cache size is small, PBR with segmented videos outperforms PBR without video segmentation, LFU and LRU by providing 16%, 90% and 120% improvement in cache hit rate respectively. When the cache size increases, PBR with segmented videos still maintains at least 8%, 34% and 40% improvement for the other three algorithms respectively. Better improvements by PBR with segmented videos are observed under conditions S1, S2 and S3. The second observation is that results of S1 are very close to those of S3. Similar case occurs for the results of S2 and S4. This implies that the size distribution of the video documents is not sensitive to the caching performance. Finally, we observe that, when the distribution of p_i is changed (i.e. uniform vs. 1/s_i), the performance of all algorithms are significantly improved. This means that the distribution of p_i is an important factor to the caching performance.

As discussed above, the size distribution of the video documents is insensitive to system performance, we shall only use the uniform distribution for the document size in our following studies. Fig. 13 plots the hit rate against the cache size with different values of k's under S1 and S2. From the figure, we observe that the distribution of p_i's affects the performance of LRU and LFU, but not PBR. This may be due to the fact that PBR already provides nearly optimal performance, as discussed earlier in this paper. Another observation is that the hit rate increases as k decreases. This verifies our earlier conclusion that the block size should be as small as possible.

Fig. 14 plots the hit rate against the cache size with different values of δ under S1 and S2. From the figure, we observe that δ will not affect the caching performance of PBR very much.

Fig. 15 shows that caching performance of all algorithms under two different video proportions of 10% and 20%. From the figure, we observe that the performance of PBR once again is not so affected by the video proportion of documents, which are not the cases for LRU and LFU. From this observation we can conclude that LRU and LFU provide even worse performance when video accesses become more dominant in the future Internet systems. In contrast, PBR is a better choice for both text based and video based systems.

Fig. 16 plots the average document access time against server utilization under S1 and S2 respectively. Under situation S1, the maximum server utilization for LRU is only about 0.5 (0.55 for LFU) while that for PBR with segmented videos is about 0.8. Under situation S2, the maximum server utilization for LRU is about 0.6 (0.64 for LFU) while that for PBR with segmented videos is over 0.85.

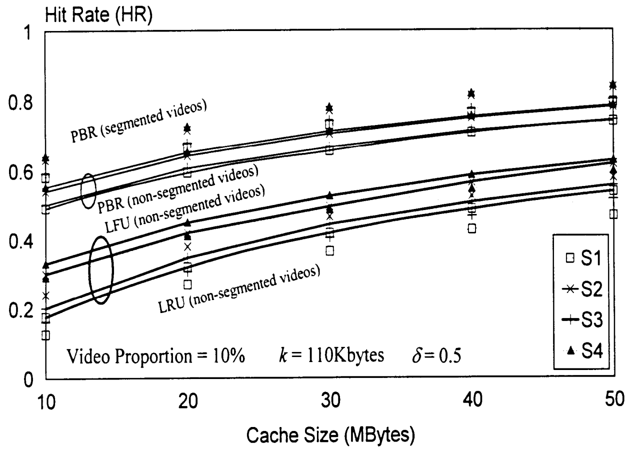
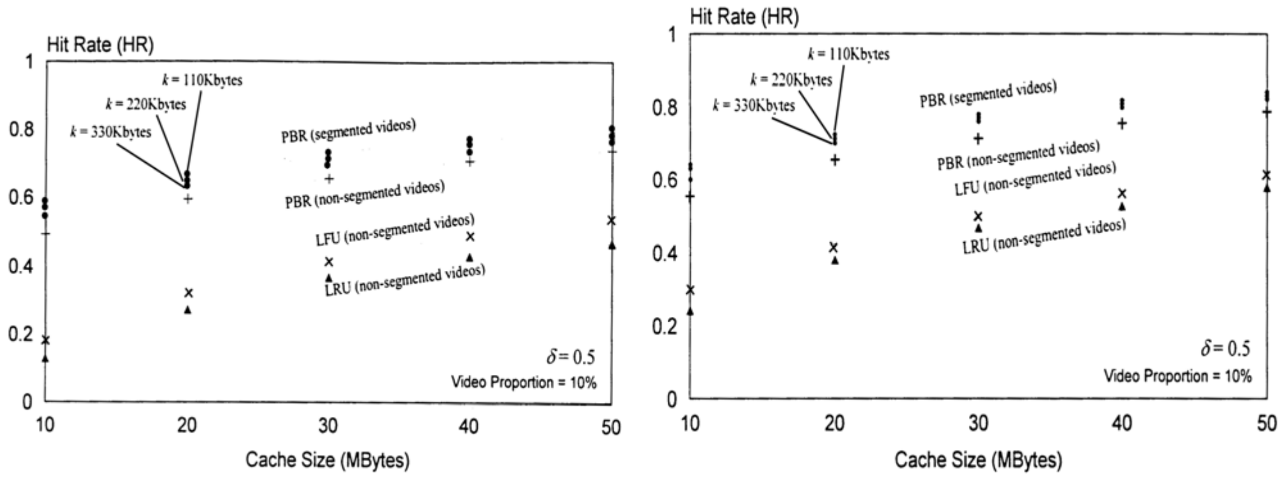
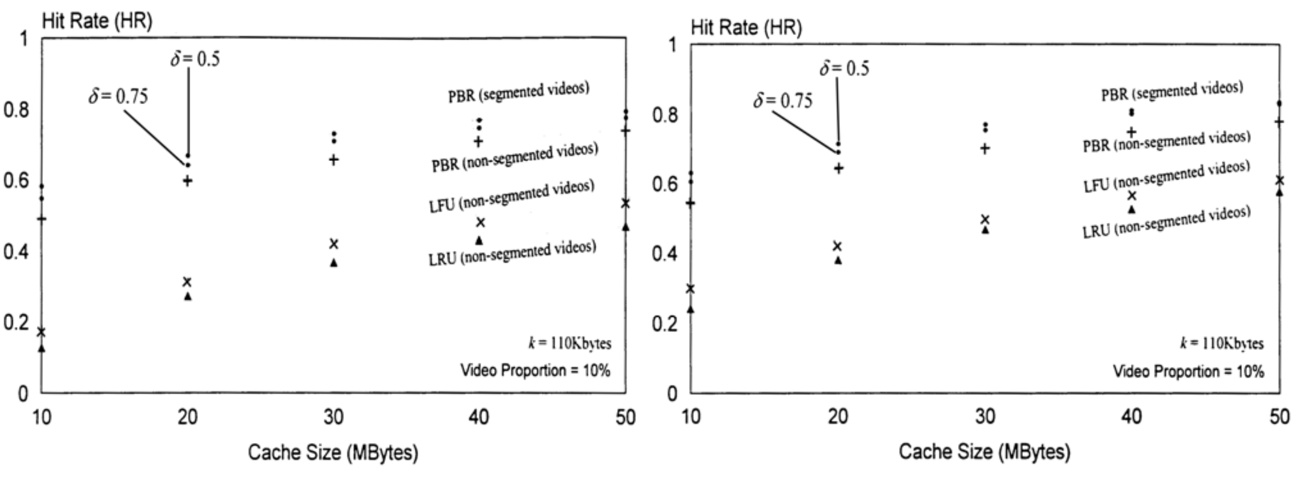


Fig. 12. Hit rate against cache size for different type of traffic (S1, S2, S3 and S4) with δ=0.5, k=110Kbytes and video proportion is 10% of all documents.



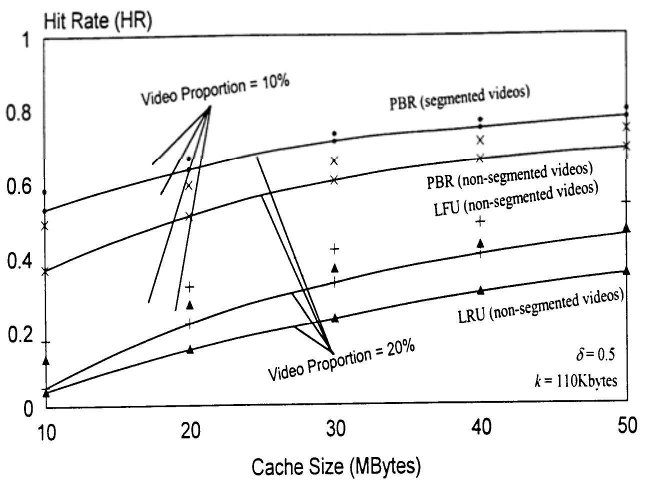
(a) (b)

Fig. 13. Hit rate against cache size with values of k. (a) For situation S1. (b) For situation S2.

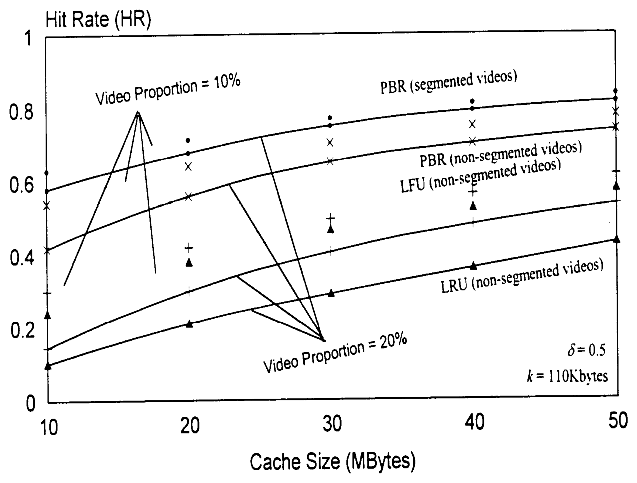


(a) (b)

Fig. 14. Hit rate against cache size with different values of δ. (a) For situation S1. (b) For situation S2.

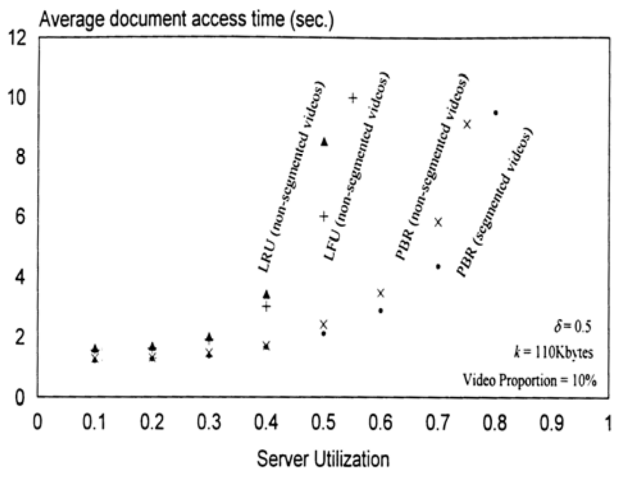


(a)

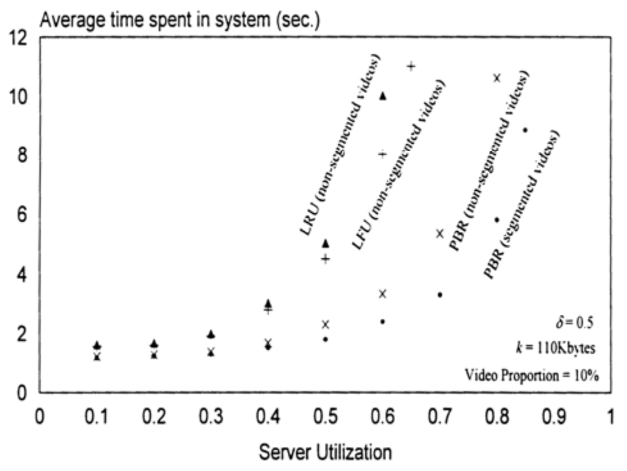


(b)

Fig. 15. Hit rate against cache size for different video proportion. (a) For situation S1. (b) For situation S2.



(a)



(b)

Fig. 16. Average time spent in the system against server utilization. (a) For situation S1. (b) For situation S2.

IV. CONCLUSIONS

In this paper, a novel cache replacement algorithm called Probability Based Replacement (PBR) for Internet systems is proposed. This algorithm replaces documents with the smallest byte access probability b_i and performs replacement only when the b_i of incoming document is larger than those documents with smallest b_i in the cache. Through both analysis and simulation, we observe that, for small caches, the PBR algorithm outperforms conventional LRU and LFU algorithms by giving 85% and 50% improvement respectively in cache hit rate. When the cache size is large, the algorithm still provides 26% and 18% improvement over that of LRU and LFU respectively. In the second half of this paper, we have discussed the challenging of caching large video documents. To solve the problems, the PBR algorithm is extended by the segmentation technique. That is, different parts of a video document will be treated differently when making cache replacement decisions. Through extensive simulations, it is shown that the PBR algorithm with segmented videos provides constantly much better performance than that of conventional replacement algorithms.

REFERENCES

[1] M. Abrams, C.R. Standbridge, G.Abdulla, S. Williams and E.A. Fox. “Caching Proxies: Limitations and Potentials,” *WWW-4*, Boston Conference, December, 1995.

[2] S. Williams, M. Abrams, C.R. Standbridge, G.Abdulla and E.A. Fox, "Removal Policies in Network Caches for World-Wide Web Documents," *In Proceedings of the ACM Sigcomm96*, August, 1996, Stanford University.
[3] K. Y. Wong and K. H. Yeung, "Site-Based Approach in Web Caching Design", *IEEE Internet Computing*, Vol.5, No.5, pp.28-34, September/October, 2001.
[4] S. Jiang and X. Zhang, "Making LRU friendly to weak locality workloads: a novel replacement algorithm to improve buffer cache performance," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 939 – 952, Aug., 2005.
[5] J. Choi, S. H. Noh, S. L. Min, E. Y. Ha, and Y. Cho, "Design, Implementation, and Performance Evaluation of a Detection-Based Adaptive Block Replacement Scheme," *IEEE Transactions on Computers*, vol. 51, no. 7, pp. 793-800, Jul., 2002
[6] K. Y. Wong, "Web Cache Replacement Policies: A Pragmatic Approach" IEEE Network, Jan-Feb. 2006, pp. 28-34.
[7] R. Tewari, H. M. Vin, A. Dan, D. Sitaram, "Resource-based caching for Web servers", *In Proceedings of SPIC/ACM conference on Multimedia Computing and Networking*, January 1998.
[8] Y. M. Chiu and K. H. Yeung, "Partial Video Sequence Caching Scheme for VOD Systems with Heterogeneous Clients", *IEEE Transactions on Industrial Electronics*, Vol.45, No.1, pp.44-51, Feb. 1998.
[9] S. Sen, J. Rexford and D. Towsley, "Proxy prefix caching for multimedia streams," *In Proceedings of IEEE INFOCOM'99*, vol. 3, pp. 1310-1319, 1999.
[10] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," *Proc. IEEE INFOCOM'02*, vol. 3, pp.1726-1735, 2002.
[11] S. Jin, A. Bestavros, and A. Iyenger, "Accelerating Internet streaming media delivery using network-aware partial caching," *Proc. IEEE ICDCS'02*, pp. 153-160, 2002.
[12] Mike Tanner, "Practical Queuing Analysis," *The IBM McGraw-Hill Series,* 1995.
[13] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya and Sulaiman A. Mirdad, "Application-Level Document Caching in the Internet," *Proc. 2nd Int'l Workshop on Services in Distributed and Networked Environment,* pp. 166-173, 1995.

Kai-Hau Yeung is an associate professor in the Department of Electronic Engineering, City University of Hong Kong. His research interests include Internet infrastructure security, mobile communication systems, and Internet caching systems. He is also an active industry consultant in the areas of computer networking and communication systems. From 1996 to 1998, he was involved in a project to develop a 900 MHz GSM mobile handset. The project team in City University successfully developed a handset prototype for a listed company in Hong Kong. He is a member of the IEEE, ACM, and BCS. He is also a Cisco Certified Network Professional, a Cisco Certified Academy Instructor and a Certified Ethical Hacker.

Kwan-Wai Ng received the B.E. degree in computer engineering and the M.E. degree in information technology from the City University of Hong Kong, Hong Kong, in 1995 and 1998, respectively.

Kin-Yeung Wong received his B.Sc. and Ph.D. degrees, both in information technology, from the City University of Hong Kong. He is currently an associate professor at Macao Polytechnic Institute. He is active in research activities, and has served as a reviewer and technical program committee member in various journals and conferences. His research interests include Internet caching systems, wireless communications, and network infrastructure security.