Vol. 39 No. 3 Transactions of Information Processing Society of Japan Mar. 1998

Regular Paper

An Evaluation of Generational Replacement Schemes Based on WWW Caching Proxy Server Logs

Noritaka Osawa,[†] Toshitsugu Yuba[†] and Katsuya Hakozaki[†]

This paper proposes and evaluates generational replacement schemes suitable for patterns of access to World Wide Web (WWW) proxy server caches. The schemes partition a cache into generations and put frequently accessed data into older generations where entries are less likely to be replaced. With our schemes, the hit rate per page is about 5.2 percentage points higher than with the Least Recently Used (LRU) algorithm, using logs of more than 8 million accesses. This improvement reduces the number of cache misses by about 10.8 percent with respect to the LRU — roughly twice as good as the LRU's improvement over the First-In First-Out (FIFO) algorithm.

1. Introduction

As the popularity of the World Wide Web (WWW) on the Internet continues to increase rapidly, WWW accesses have come to account for a large part of network traffic. User access latency and server load have also become problem areas. One effective way of solving these problems is to use a caching technique. Caching the frequently accessed data at sites near a client reduces the network traffic and access latency; it also reduces the WWW server load by reducing the number of accesses to a server.

Proxy servers were introduced to permit accesses beyond a network firewall, and subsequently enhanced to include caching facilities[10]. Although there have been several studies of WWW caching[1),6),7),13),15)], new caching schemes for WWW access patterns have not been fully researched, and the sizes of access logs (or trace data) used in previous studies were not sufficient.

This paper proposes generational replacement schemes suitable for patterns of access to WWW proxy server caches. Caching schemes are evaluated using the logs of the proxy server at the Information Processing Center (IPC) of the University of Electro-communications (UEC) in Japan. Our analysis of caching performance is based on logs of more than 8 million accesses[14]. Analysis of the logs shows that many pages are accessed only once, thus it is more effective to keep pages that have been accessed two or more times than to keep pages that have been accessed only once. This characteristic is utilized in our generational replacement schemes. The hit rates of the cache, which are indices of the reduction in network traffic and access latency, are evaluated and discussed in this paper.

We explain the hit rates and our evaluation method in Section 2. Section 3 describes access logs used for evaluations and explains the basic characteristics of the logs. The proposed generational replacement schemes are explained in Section 4, and our proposed schemes are evaluated in Section 5. Section 6 discusses how our schemes differ from other proposed schemes. Finally, we summarize our findings and outline our plans for future work.

2. Evaluation Method and Hit Rates

We use trace-driven simulation to evaluate caching schemes. Trace logs are collected at a proxy server, and are used as input data to programs that simulate caching schemes with cache size parameters. The simulation programs output the hit rates.

The entity that a Universal Resource Locator (URL)[5] refers to is called a *page* in this paper. The *page* is usually implemented as a file in a server. The *page* does not represent one file block but the entire data contained in a file.

Since the size of a WWW page is not constant, two types of hit rate should be considered: the hit rate per page (*Hp*), that is, the hit rate per entry or URL, and the hit rate per byte (*Hb*), that is, the hit rate per data unit (byte). These will be defined and described in the following subsections. They were not discussed in

† Graduate School of Information Systems, the University of Electro-Communications

previous studies[1),6),15)] of WWW caching.

2.1 Hit Rate Per Page

Let H/N be the hit rate per page (Hp) where H and N are the number of page hit accesses and the total number of page accesses, respectively. Hp effects the effective access latency: as Hp increases, the effective access latency decreases. The effective access latency $D(Hp)$ is represented by

$$\begin{aligned} D(Hp) &= HpD_h + (1 - Hp)D_m \\ &= D_m - Hp(D_m - D_h), \end{aligned}$$

where D_h and D_m are the access latencies in the case of a cache hit (local access) and a cache miss (remote access), respectively. Therefore, the reduction ratio $Rp_{j,i}$ of $D(Hp_j)$ to $D(Hp_i)$ is expressed by the following equation:

$$\begin{aligned} Rp_{j,i} &= \frac{D(Hp_j) - D(Hp_i)}{D(Hp_i)} \\ &= \frac{(Hp_i - Hp_j)(D_m - D_h)}{D_m - Hp_i(D_m - D_h)} \\ &= \frac{Hp_i - Hp_j}{\frac{D_m}{D_m - D_h} - Hp_i}. \end{aligned}$$

2.2 Hit Rate Per Byte

The hit rate per byte (Hb) is defined as B/M where B and M are the number of hit data and the total number of accessed data, respectively. A cache hit does not increase network traffic, but a cache miss does. Thus Hb influences the network traffic load. As Hb increases, the external network traffic decreases. When Hb is improved to Hb', the amount of network traffic T decreases to T'. The reduction ratio Rb of T' to T is expressed as follows:

$$\begin{aligned} Rb &= \frac{T - T'}{M} = \frac{M(1 - Hb) - M(1 - Hb')}{M(1 - Hb)} \\ &= \frac{Hb' - Hb}{1 - Hb}. \end{aligned}$$

3. Log Data

This section describes the proxy server in which logs were gathered and explains basic characteristics of the logs.

3.1 Proxy Server and Its Users

The proxy server and its users at the IPC of the UEC are described here. Since educational workstations (WS's) for students at the IPC cannot communicate directly with sites outside the university, users of the educational WS's have to use a proxy server to access pages outside the university. The proxy server is believed to be used by all users of the educational WS's at the IPC. The proxy server at the IPC is also used as a cache by departments and laboratories that do not have their own caching proxy servers. The IPC used the CERN httpd[10)] as its proxy server when the logs were collected.

At that time, NCSA Mosaic[2)] was the WWW client on educational WS's at the IPC. Mosaic did not hold cached data beyond one session. On computers other than educational WS's at the IPC, other WWW clients were used in addition to Mosaic. Some of them, such as Netscape Navigator, held cached data beyond one session.

3.2 Periods and Access Amounts

This study focuses on successful accesses with a `Status-Code` of 200 through the Hypertext Transfer Protocol (HTTP)[4)]. Successful accesses will be simply referred to as accesses in this paper. Log data were gathered between 1 pm on November 7, 1995 and 1 pm on June 25, 1996. The total number of accesses was 8,362,840, with an average of 36,202.8 accesses/day. The hit rates during different log periods, where the cache size was infinite and no invalidation was performed, are shown in **Table 1**. The period of a log can be seen as an expiration period, and thus we can estimate the effect of expiration of data from the hit rates in Table 1.

3.3 Access Frequencies

The number of pages that have the same access frequency is shown in **Fig. 1**. The X-axis shows the access frequency of a page, while the Y-axis shows the number of pages whose access frequencies are the same. Requesting hosts are not distinguished in our analyses; thus, the access frequency of a page is the total number of accesses to the page, regardless of the hosts requesting the accesses. The pages that have the highest access frequencies are WWW home pages related to educational computers for students. Mosaic on an educational WS usually accesses the home pages at startup time, because it does not hold cached data beyond one session.

Next, we will show that our log data is not special but usual on the basis of Zipf's law[8)]. Glassman[6)] states that the access frequencies of pages follow Zipf's law. The line in Fig. 1 shows Zipf's law. In the figure, Zipf's law is applicable to the lower range of access frequencies, but there is a small difference between the law and the log data.

Table 1 Hit rates during different log periods. N is the total number of accesses. $\overline{N}$ is the average number of accesses per day. M is the total number of accessed data. $\overline{M}$ is the average number of accessed data per day. Hp and Hb are the hit rate per page and the hit rate per byte, respectively, where the cache size is unlimited.

Period (Starting from Nov. 07, '95)	N	$\overline{N}$	M	$\overline{M}$	Hp	Hb
Nov. 14, '95	288,518	41,216.9	2,165.96 MB	309.42 MB	69.6%	53.0%
Dec. 05, '95	1,085,342	38,762.2	9,274.40 MB	331.23 MB	78.1%	58.7%
Jun. 25, '96	8,362,840	36,202.8	77,203.66 MB	334.25 MB	83.5%	67.8%

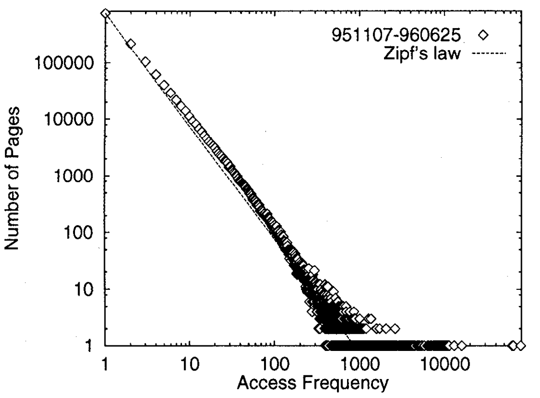


Fig. 1 Number of pages that have the same access frequency. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)

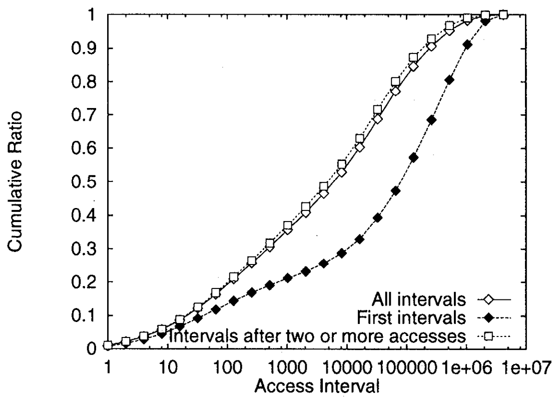


Fig. 2 Cumulative ratio of access intervals. (Based on log data collected between Nov. 7, 1995 and June 25, 1996).

Let the number of pages whose frequency is f be $P(f)$. The total number of accesses to pages whose access frequency is f is $A(f) = fP(f)$. If Zipf's law holds, $A(f) = M/f$ where M is a constant. By transforming equations, we obtain $P(f) = M/f^2$. $P(f)$ is plotted in Fig. 1, where Zipf's law holds. Moreover, the hit rate $\gamma(F) = H(F)/N(F) = \sum_{f=1}^{F} \frac{f-1}{f^2} / \sum_{f=1}^{F} \frac{1}{f}$ where pages are accessed according to Zipf's law, the cache size is infinite, and no invalidation is performed. Let F be the maximum access frequency, $N(F)$ the total number of accesses, and $H(F)$ the number of hit accesses.

The hit rate per page is 83.5% where the cache size is infinite and no invalidation is performed, as shown in Table 1. The hit rate is 70.1% where the access frequency is limited to between 1 and 1000 in Fig. 1. This hit rate is larger than in other studies 1),6),7), because the amount of log data, cache size and the invalidation policy influence the hit rate. According to Zipf's law, the hit rate is reasonable because it is less than $\gamma(1000) \approx 78.0\%$.

3.4 Access Intervals

The relationship between access intervals and their cumulative ratios of frequency are shown in **Fig. 2**. The access interval of a page represents the interval between an access to the page and the next access to the same page, no matter what hosts request the accesses. Let $A_i(P)$ be the i-th access to a page P, regardless of the requesting hosts, and let $G(A)$ be the sequence number of an access A through the proxy server, also regardless of the requesting hosts. The access interval between $A_i(P)$ and $A_{i+1}(P)$ is defined as $G(A_{i+1}(P)) - G(A_i(P))$.

The cumulative ratio of intervals after two or more accesses increases more rapidly than the cumulative ratio of the first intervals, which are intervals between the first access and the second access to a page, in Fig. 2.

Figure 1 shows that there are many pages that are accessed only once. It is useless to retain these pages in a cache: their retention decreases the effective cache size in a finite cache and degrades the hit rate of the cache. Figure 2 also shows that it is more effective to keep pages that have been accessed two or more times than to keep pages that have been accessed only once. To improve the hit rate, we propose algorithms that use generations and replace pages according to the possibility of future accesses.

4. Replacement Algorithms

First, we propose generational replacement algorithms that are suitable for WWW data caching, and then we describe conventional (basic) replacement algorithms.

4.1 Generational Caching

The proposed algorithms use generations and replace pages according to the possibility of future accesses. A page that has been accessed only once is less likely to be accessed than a page that has been accessed two or more times within a certain period. The algorithm replaces the entry that is least likely to be accessed. We call the caching scheme using this algorithm *generational caching* and a cache using this scheme *a generational cache.*

A conceptual diagram of generational caching is shown in **Fig. 3**. Entries in a cache are sequentially ordered. A cache is divided into generations. Any number of generations are possible; however, this paper will deal mainly with evaluations of the simplest case, that is, two generations.

If an entry is not found in the cache, it is inserted into the youngest generation. When an entry is inserted, every entry in the youngest generation shifts to the next position. An entry at the end of the youngest generation overflows and is removed from the cache.

If an entry is found (hit) in the cache, it is moved to the head of the next-older generation (or the oldest generation). Entries between the original position and the destination position of the entry are shifted. Incidentally, unless the original position is in the oldest generation, the entry at the end of the next-older generation moves to the head of the next-younger generation, that is, it is not removed from the cache.

If the size of the youngest generation is too large, pages accessed only once reduce the effective cache size. On the other hand, when the size is too small, transition of a page to an older generation will hardly ever occur, because the page will be removed before it is accessed again. Consequently, the hit rate is not improved when the size is too small. Therefore it is desirable that the size of the youngest generation should be larger than the interval between multiple accesses.

Figure 4 shows a conceptual diagram of generational caching using an additional history list. This generational caching uses an additional history list (a special generation), which does not hold data but only entry information. In other words, an entry in the list has only URL information without the page data that the URL refers to. Even if an entry is found in the additional history, the access to the found entry cannot be considered a hit, because its data cannot be accessed in the cache. However, the insertion point is different from that when the entry is not found, as shown in Fig. 4. An entry found in the additional history is inserted into an older generation than an entry that is not found.

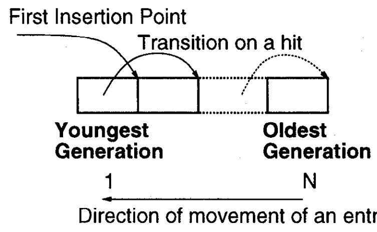


Fig. 3 A conceptual diagram of generational caching.

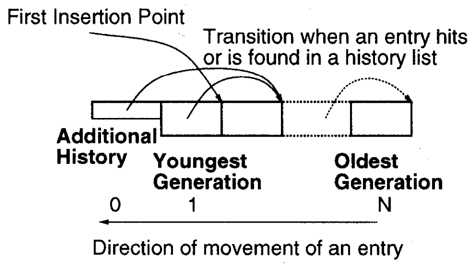

First Insertion Point
Transition when an entry hits or is found in a history list
Additional History
Youngest Generation
Oldest Generation
0
1
N
Direction of movement of an entry

Fig. 4 A conceptual diagram of generational caching using an additional history list.

4.2 Conventional Algorithms

Conventional replacement algorithms[11] for a cache are explained and abbreviations for these algorithms are also given, because they will be used in a performance comparison with our proposed algorithms. In the following, LRU, FIFO, LFU and OPT are referred to as conventional algorithms.

LRU Least Recently Used algorithm. This algorithm replaces the least recently used entry with a new entry. LRU and simplified LRU algorithms are widely used in cache replacement.

FIFO First-In First-Out algorithm. This algorithm is used when a simple mechanism is preferred for the hardware cache.

LFU Least Frequently Used algorithm. This algorithm replaces the least frequently used entry with a new entry.

OPT Optimal algorithm. OPT replaces the entry that will be accessed in the furthest future. OPT cannot be implemented unless the future accesses are known. Therefore OPT is not a practical algorithm, but it shows the limitation of hit rates in a fixed-size cache.

5. Evaluation of Replacement Algorithms

This section evaluates replacement algorithms in a cache with a fixed entry size and in a cache with a fixed data capacity size. In this paper, replacement algorithms choose a victim not only among the pages accessed by the host that caused the miss, but among all the pages in the cache. In other words, they are global algorithms.

5.1 Entry Size Limitation

The results of evaluation of replacement algorithms in a cache with a fixed number of entries are now given. The hit rates per page for conventional algorithms are shown in **Fig. 5**, where the entry size of a cache (the number of entries in a cache) is limited. Figure 5 shows that *Hp* for LRU is higher than that for FIFO, while *Hp* for LFU is lower than those for FIFO and LRU when the cache entry size is small. This is because the number of entries that have no hits becomes smaller and becomes 1 if the cache entry size is insufficient. In such situations, entries that have more than one hit are rarely replaced. In brief, the newest entry is almost always chosen for replacement. Therefore the hit rate for LFU is low when the cache is small.

We evaluated two generational algorithms: **gp2** and **gp2a**. Both have two generations but **gp2a** uses an additional history list. The replacement algorithm within a generation is LRU. The hit rates for **gp2** and **gp2a** are shown in **Fig. 6** and **Fig. 7**, respectively. If the entry size of the younger generation is too small, only pages that are accessed consecutively remain in the cache. Hence the hit rate is bad if the

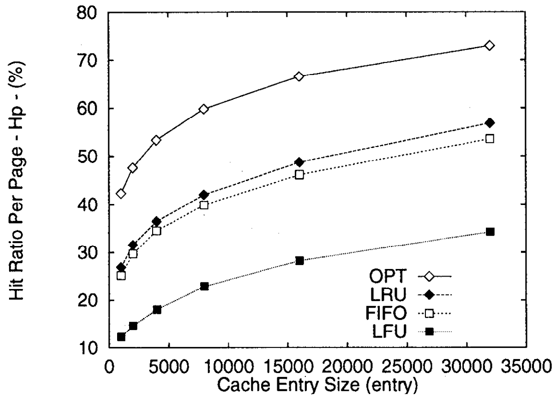
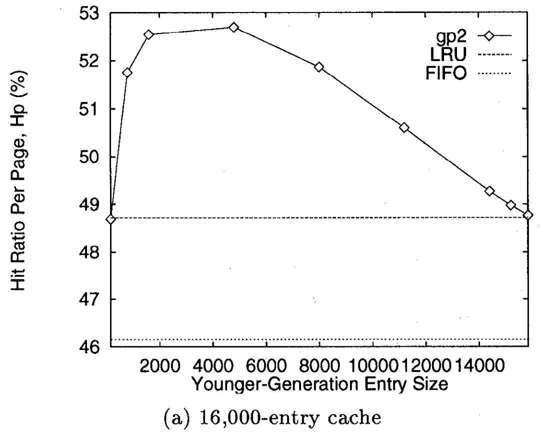
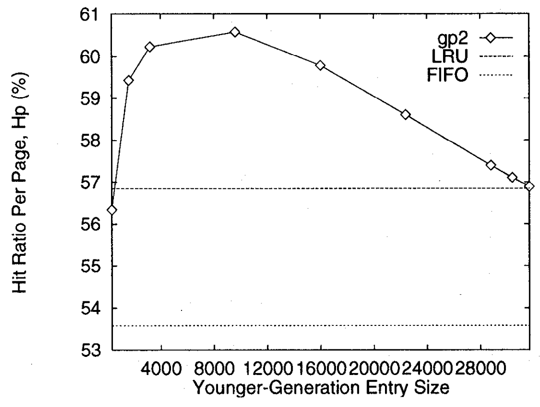


Fig. 5 Relationship between cache sizes and hit rates of pages in conventional algorithms. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)



(a) 16,000-entry cache



(b) 32,000-entry cache

Fig. 6 The hit rate per page for **gp2**. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)

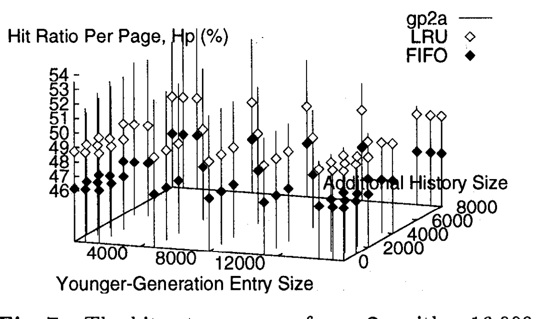


Fig. 7 The hit rate per page for **gp2a** with a 16,000-entry cache. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)

younger generation is too small. On the other hand, as the entry size of the younger generation approaches the total entry size of the cache, the older generation becomes smaller. As a result, the two-generation cache tends toward a single-generation cache, and therefore the hit rate of the cache approaches that for LRU.

Hp for **gp2** with a 4,800-entry younger generation (30% of the total size) is 52.7% as shown in Fig. 6 (a). The hit rates per page for LRU, FIFO and LFU are 48.7%, 46.2% and 28.2%, respectively, where the total entry size of the cache is 16,000. *Hp* for **gp2** with a 9,600-entry younger generation (30% of the total size) is 60.6% as shown in Fig. 6 (b). The hit rates per page for LRU, FIFO and LFU are 56.8%, 53.6% and 34.2%, respectively, where the total entry size of the cache is 32,000. Thus *Hp* for **gp2** is approximately 4% points higher than that for LRU in Fig. 6. The difference between the hit rates for **gp2** and LRU is larger than that between the hit rates for LRU and FIFO. That is to say, the improvement obtained by using **gp2** is twice as good as the improvement obtained by using LRU instead of FIFO in Fig. 6.

Hp for **gp2a** is about 4.8% points higher than that for LRU at its peak in Fig. 7. Figure 7 shows that the additional history list in **gp2a** can enhance the hit rate if the younger generation is small but *Hp* for **gp2a** does not significantly increase unless the younger generation is small.

5.2 Data Size Limitation

In the previous subsection, the size of a cache was limited by the number of pages or entries. However, the total amount of data should be considered as the real size limit of a cache. Therefore, this subsection describes evaluations in which the total amount of data is limited. FIFO-s, LRU-s, LFU-s and OPT-s represent FIFO, LRU, LFU and OPT algorithms, respectively, where the size of a cache is limited by the total amount of data.

sgr2 corresponds to the **gp2** algorithm when the size of a cache is limited by the total amount of data. The younger-generation entry size in **sgr2** is determined not by the number of entries but by the ratio of the younger-generation entry size to the total entry size.

Figure 8 shows the hit rates for conventional algorithms where the total amount of data in a cache is limited. In Fig. 8, LRU-s is superior to FIFO-s and LFU-s. The hit rates per page for LRU-s, FIFO-s and LFU-s are 52.4%,

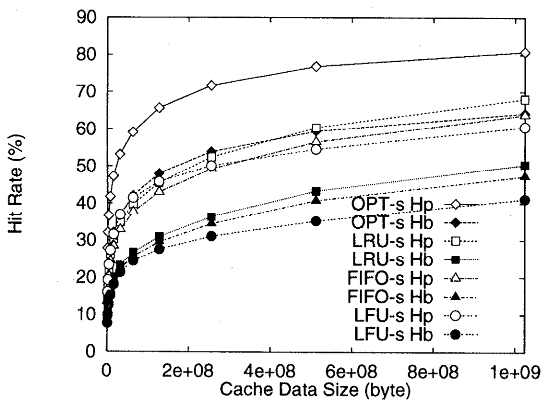


Fig. 8 *Hp* and *Hb* using conventional algorithms when the cache data size is limited. *Hp* and *Hb* represent the hit rate per page and the hit rate per byte respectively. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)

49.2% and 50.1%, respectively, where the total data size of the cache is 256M bytes, and the hit rates per byte for the same algorithms are 36.4%, 34.4% and 31.1%, respectively, as shown in Fig. 8.

Figure 9 shows *Hp* and *Hb* for **sgr2**. In Fig. 9 (a), *Hp* and *Hb* for **sgr2** are 57.6% and 39.7%, respectively, where the ratio of the younger-generation entry size to the total entry size is 10%. Therefore, **sgr2** has a hit rate per page about 5.2% points higher than LRU-s at its peak. In this case, 5.2% of the page accesses is equal to about 430,000. *Hb* for **sgr2** is approximately 3.3% points higher than *Hb* for LRU-s at its peak, and 3.3% of the total amount of accessed data is equal to about 2.5 G bytes. The difference between hit rates for **sgr2** and LRU-s is larger than the difference between those for LRU-s and FIFO-s when the cache data size is 256 M bytes and the younger-generation size is 10% of the total entry size in Fig. 9 (a). When the ratio of the younger generation entry size to the total entry size is 10%, the reduction ratio of network traffic (*Rb*) of **sgr2** with respect to LRU is approximately 5.15 %. Moreover, we obtain a reduction ratio of the effective access latency of approximately 10.8% when $D_m \gg D_h$. This value is equal to the reduction ratio of cache misses. In other words, **sgr2** reduces the number of cache misses by about 10.8% with respect to LRU.

The improvement of **sgr2** with a 1024 MB data cache is less than that with a 256 MB data cache, as shown in Fig. 9. However, generally speaking, **sgr2** is superior to LRU.

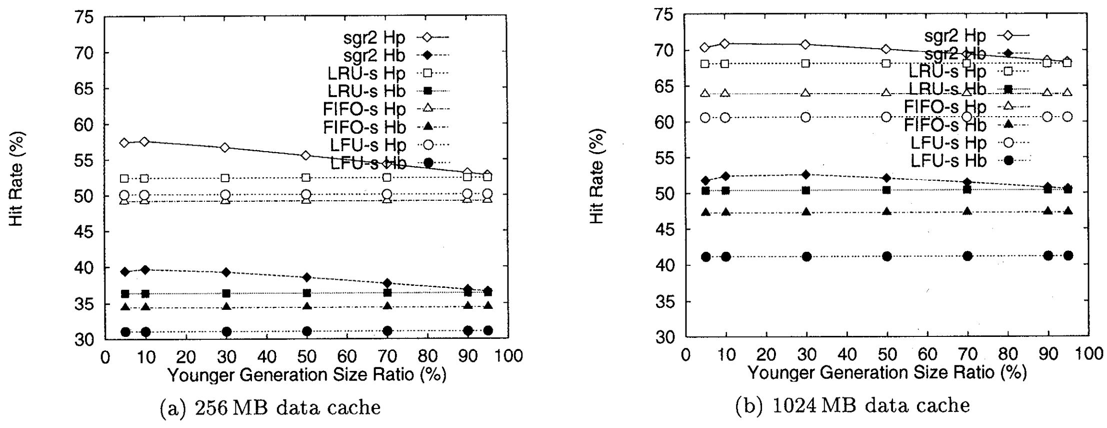


Fig. 9 *Hp* and *Hb* for **sgr2**. *Hp* and *Hb* represent the hit rate per page and the hit rate per byte, respectively. (Based on log data collected between Nov. 7, 1995 and June 25, 1996.)

6. Related Work

Studies have been made not only of caching in virtual memory systems[11] and databases[12] but also of WWW caching[1),6),15)]. This section explains the difference between our generational caching schemes and those described in related work.

Entities in virtual memory systems and disk management have fixed length. For examples, disk blocks managed in a disk buffer have the same length. Previous studies have investigated only a single hit rate. On the other hand, WWW entities have variable lengths. Therefore we investigated two types of hit rate: the hit rate per page and the hit rate per byte.

Glassman[6] reported the results of evaluation of caching using LRU replacement at the System Research Center gateway of Digital Equipment Corporation, one of the first evaluations of caching at a proxy. The average number of accesses per day at the gateway was about 4,000 — about one tenth of our average.

Pitkow, et al.[15] proposed and evaluated a caching algorithm based on logs from a server, not a proxy. The period of logs was 3 months, whereas ours was more than 7 months.

Abrams, et al.[1] evaluated LRU and two of its variations, using logs of a proxy server at the Virginia Tech (Virginia Polytechnic Institute and State University) campus during one semester. The logs numbered fewer than 170,000 accesses, whereas ours cover about 8,300,000 accesses. Moreover, the hit rates over a long period could not be evaluated because of the limitations of their analysis tools. Therefore the hit rates which they reported are less than 50%. As stated before, two types of hit rate should be considered, because the size of a WWW page is not a constant. However the work mentioned above considers only one type of hit rate, whereas our study evaluates both the hit rate per byte and the hit rate per page.

Recent studies[16),17)] have focused on minimizing the cost of cache misses and taking bandwidth into consideration. We will investigate schemes that attempt to reduce the delay resulting from a cache miss, in addition to improving the hit rate.

It is known as an empirical law that many cells (or objects) have a short lifetime and cells that have lived for a while will live for a long time in a system that generates cells dynamically, such as a Lisp programming language system. A generational garbage collection method[9] that utilizes this empirical law has been proposed and used. This garbage collection scheme performs fewer garbage collections for older generations and more frequent garbage collections on younger generations. The main objective of the scheme is to reduce the overhead of garbage collections. There is a similarity between our generational caching and generational garbage collection. However, the main objective of our caching schemes, unlike that of generational garbage collection, is not to reduce the overhead but to improve the hit rate. Moreover, the statistical characteristics of WWW page accesses cannot be regarded as identical with those of the lifetime of cells.

Partitioning of a cache has been used for prefetching in the file buffer cache of a UNIX operating system. However, our schemes do not use look-ahead (prefetching) information, and our schemes' method of predicting (estimating) accesses is different from that of the buffer cache.

7. Concluding Remarks and Future Work

This paper has proposed generational caching schemes, evaluated them by using actual logs of more than 8 million accesses, and shown that the hit rate per page of our scheme is approximately 5.2% points more effective than that of LRU at its peak. The improvement in the hit rate per byte is about 3.3% points. In other words, the hit rate per page and the hit rate per byte of our scheme are approximately 10.8% and 5.15% more effective, respectively, than those of LRU.

If the cache size is sufficiently large, the hit rates of all replacement algorithms are close to the optimal hit rate, because replacement rarely occurs. In these circumstances, our generational caching algorithms are not always the best. However, the total amount of valid accessed data is growing as a result of the increase in the number of sound, image or movie data and the exponential growth in the number of WWW accesses. Therefore it will be difficult to keep all valid accessed data in a proxy server cache. Only some of the accessed data will be kept in the cache. Generational caching schemes are useful because they reduce cache misses (network traffic) and the effective latency (response time) in a fixed-size cache of WWW.

Moreover, in environments such as mobile or wireless WWW[3], resources such as disks and network bandwidth are limited. It is difficult to enhance resources, because portability is essential. We expect that generational schemes will be useful in enhancing hit rates in such environments as well as in the usual proxy servers.

For generational caching schemes to be effective, the size of generations has to be appropriate, and manual adjustment of the size of generations is not desirable. We will therefore also study automatic adjustment of the size of generations.

Acknowledgments We wish to thank Prof. Shuichi Itoh, Dr. Katushi Kobayashi and other members of the Information Processing Center at the University of Electro-Communications for helping us acquire the WWW access logs.

References

1) Abrams, M., Standridge, C.R., Abdulla, G., Williams, S. and Fox, E.A.: Caching Proxies: Limitations and Potentials, *Proc. Fourth Int'l World Wide Web Conference*, pp.119–133 (1995).

2) Andreessen, M.: NCSA Mosaic Technical Summary, Technical Report, National Center for Supercomputing Applications (1993).

3) Bartlett, J.F.: W4 – The Wireless World Wide Web, *Proc. Workshop on Mobile Computing Systems and Applications*, pp.176–178 (1994).

4) Berners-Lee, T., Fielding, R. and Frystyk, H.: Hypertext Transfer Protocol – HTTP/1.0, Http working group internet-draft, IETF (1996).

5) Berners-Lee, T., Masinter, L. and McCahill, M.: Uniform Resource Locators, RFC 1738, IETF (1994).

6) Glassman, S.: A Caching Relay for the World Wide Web, *Proc. 1st Int'l Conf. on WWW*, Vol.27, Geneva, pp.165–173 (1994). (also appeared in Computer Networks and ISDN Systems).

7) Ichii, S. and Nakayama, M.: Effectiveness of WWW Caching in a Campus Network, SIG Notes DSM-9505033, IPSJ (1995). (in Japanese).

8) Knuth, D.: *Sorting and Searching*, The Art of Computer Programming, Vol.3, Addison-Wesley (1973).

9) Lieberman, H. and Hewitt, C.: A Real-Time Garbage Collector Based on the Lifetimes of Objects, *Comm. ACM*, Vol.26, No.6, pp.419–429 (1983).

10) Luotonen, A. and Altis, K.: World-Wide Web Proxies, *Proc. 1st Int'l Conf. on WWW* (also appeared in Computer Networks and ISDN Systems), Vol.27, Geneva, pp.147–154 (1994).

11) Maekawa, M., Oldehoeft, A.E. and Oldehoeft, R.R.: *Operating Systems: Advanced Concepts*, Benjamin/Cummings Pub., Menlo Park (1987).

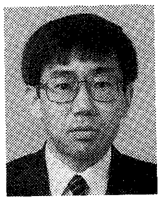
12) O'Neil, E.J., O'Neil, P.E. and Weikum, G.: The LRU-K Page Replacement Algorithm for Database Disk Buffering, *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pp.297–306 (1993).

13) Osawa, N., Hayano, F., Yuba, T. and Hakozaki, K.: Evaluation of Replacement Policies in a WWW Cache Based on Logs from a Proxy Server, SIG Notes DPS 74-33, IPSJ (1996). (in Japanese).

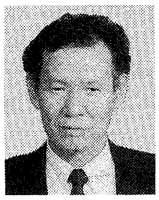
14) Osawa, N., Yuba, T. and Hakozaki, K.: Generational Caching Schemes for a WWW Caching

Proxy Server, *Proc. High-Performance Computing and Networking* (*HPCN97 Europe*), LNCS Vol.1225, pp.940–949 (1997).
15) Pitkow, J.E. and Recker, M.M.: A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns, *Proc. 2nd Int'l WWW Conf.*, pp.1039–1046 (1994).
16) Scheuermann, P., Shim, J. and Vingralek, R.: A Case for Delay-Conscious Caching of Web Documents, *Proc. 6th Int'l WWW Conf.*, pp.725–734 (1997).
17) Wooster, R.P. and Abrams, M.: Proxy Caching That Estimates Page Load Delays, *Proc. 6th Int'l WWW Conf.*, pp.325–334 (1997).

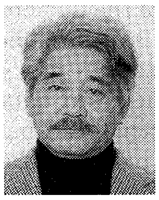
(Received May 16, 1997)
(Accepted November 5, 1997)



Noritaka Osawa was born in 1960. He received his B.S., M.S. and Ph.D. degrees in information science from the University of Tokyo in 1983, 1985 and 1988, respectively. Since 1993, he has been at the University of Electro-Communications, as a research associate. His current research interests focus on parallel and distributed system software. He is a member of IPSJ, ACM and IEEE-CS.



Toshitsugu Yuba was born in 1941. He received the B.E. and M.E. degrees in electrical engineering from Kobe University in 1964 and 1966, respectively, and the Ph.D. degree from the University of Tokyo in 1982. In 1966 he joined the Nomura Research Institute. From 1967 to 1993, he was with the Electrotechnical Laboratory (ETL) of the Agency of Industrial Science and Technology, the Ministry of International Trade and Industry. His final position at ETL was the director of Computer Science Division. He moved from ETL to the University of Electro-Communications in 1993. He is currently a professor of the Graduate School of Information Systems. His current interest is parallel and distributed computing in computer science; parallel computation models, parallel/distributed operating systems, parallel compilers, parallel architectures and parallel applications such as parallel discrete event simulation. Dr. Yuba is a member of IEICE, IPSJ, the Japan Society for Software Science and Technology, the Robotics Society of Japan, ACM, and IEEE-CS.



Katsuya Hakozaki is a professor at Graduate School of Information Systems, University of Electro-Communications. He received B.E. in electronics from Kyushu University in 1963 and D. Eng. from the same university in 1981. He joined in NEC Corporation in 1963 where he engaged in the research and development on performance evaluation of computer systems, language oriented computer architectures, databese machines and computer networks. He was assigned the Professor of University of Electro-Communications in 1994. His current research interests include applications of computer networks, digital libraries, network management and distributed information systems. He was awarded the Best Paper Prize from the Information Processing Society of Japan in 1982. He is a member of IPSJ, IEICE and IEEE.