Hierarchical Image Caching in Content Distribution Networks

Zhou SU, Teruyoshi WASHIZAWA, Jiro KATTO and Yasuhiko YASUDA
(Graduate School of Science and Engineering, Waseda University,
3-4-1, Ohkubo, Shinjuku-ku, Tokyo 169-0072, JAPAN.)
Contact E-mail: suzhou@yasuda.comm.waseda.ac.jp

***Abstract*—**

The efficient distribution of stored information has become a major concern in the Internet. Since the web workload characteristics shows that more than 60% of network traffic is caused by image documents, how to efficiently distribute image documents from servers to end clients is an important issue. Proxy cache is an efficient solution to reduce network traffic. And it has been shown that an image caching method (Graceful Caching) based on hierarchical coding format showed better performance than conventional caching schemes in recent years. However, as the capacity of the cache is limited, how to efficiently allocate the cache memory to achieve a minimum expected delay time is still a problem to be resolved.

This paper presents an integrated caching algorithm to deal with the above problem in the Internet. By analyzing the web request distribution of Graceful Caching, both replacement and pre-fetching algorithms are proposed. We also show that our proposal can be carried out based on information readily available in the proxy server and it flexibly adapts its parameters to the hit rates and access pattern of users' requesting documents in the Graceful Caching. We finally verify the performance of this algorithm by simulations.

***Index*— Graceful Caching, Network Traffic, Web Caching, Web Performance, Caching Algorithm**

I. INTRODUCTION

As the scale and use of the Internet increase, Web content providers find that it is difficult to serve all users with low response time, especially in the face of high loads. In recent years, how to set up content distribution networks to efficiently distribute stored information has become a major concern in the Internet [1] [2].

Since the web workload characteristics shows that almost 67% of network traffic is caused by image documents [7], how to efficiently distribute image documents from servers to end users is a very critical issue. Proxy caches are commonly introduced between servers and clients to reduce data traffic and improve response time in the Internet, because data is cached temporarily in the proxy cache and utilized many times.

II. GRACEFUL CACHING

However, current system employs a "hard" caching strategy: an image is stored in a cache or not even if its data is quite big.

To overcome this problem, a caching scheme called Graceful Caching [5][6], in which a progressive image format is used and a low resolution version of the original image can be kept in the cache, is proposed. By doing this, variable amounts of cache memory can be assigned to each image according to clients' demand. This hierarchical image caching system has been shown to achieve substantial performance improvement [6].

The advantage of this approach is that, in many cases, users will not need to download full resolution images from the server if an acceptable quality of image is retrieved. Furthermore, at the same time, more kinds of images can be kept in the cache for user's re-utilization by using hierarchical image format.

III. CACHING ALGORITHM

As the cache capacity is limited, how to efficiently allocate cache memory plays an important role in cache performance. This so –called caching algorithm has attracted many researches. However, most of the current available algorithms are for the conventional (hard) caching schemes. How to determine which images and which layers in the images should be cached and replaced are not mentioned [9][10][12]. Besides, most of the common algorithms simply concern themselves with just how to replace the object from the cache. They show little consideration for how to fetch the object from the server to the cache [3][8][13]. In this paper, we therefore

try to present an integrated caching algorithm in which both replacement and pre-fetching for this Graceful Caching (soft cache) system will be considered.

IV. THEORETICAL ANALYSIS

We firstly carry out a theoretical analysis of the web access of Graceful Caching.

Web request probability in the Graceful Caching

We define the probability space on a set of N images with L layers, by introducing parameters $Q=\{A, B, P\}$ as follows:

$A=\{ a_{i,j}$; the j th layer of image $i\}$,

$i=\{1,\ldots,N\}$, $j=\{1,\ldots, L\}$,

$B=\{ b_{i,j} =\{ a_{i,1} , a_{i,2} , \ldots . , a_{i,j} \}\}$,

$P: B \rightarrow [0,1]$

Here, B is a Borel field, the family of partial subsets of A which is strictly regulated by Progressive JPEG encoding format [14]. For example, the user can't request for the (j+1)-th layer of one image unless he has got the j-th layer before. P is the probability measure that maps B into a [0,1] value, $P(b= b_{i,j}) = P(i, j)$.

After analysis, in this paper we obtain a web request probability for the j-th layer of image i by the following equation:

$$P(b_{i,j}) = \prod_{m=0}^{j-1} d_{k,m} \cdot P(b_{i,1}) = \cdot \frac{\Omega \cdot \prod_{m=0}^{j-1} d_{k,m}}{r_i^{\alpha}} \qquad (1)$$

Here, Ω, α are parameters of Zipf distribution, and r_i is the ranking of the image i. $d_{k,j}$ (Greedy Degree) means a probability that a user will request the (j+1)-th layer after getting the j-th layer. k means the number of layers in the cache; when a user begins to request an image, there are k layers of that image stored in the cache at that time.

After analysis, we also obtain the Greedy Degree by the following equation:

$$d_{k,j} = \begin{cases} p_k & (j = k) \\ 1 & (j \neq k) \end{cases} \quad k=\{0,\ldots,L\}, j=\{0,\ldots,L\text{-}1\} \qquad (2)$$

We also prove that, if the original image is divided into 4 layers ($L=4$) as a simple scenario, a following matrix can be defined

$$D = [d_{k,j}] = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} = \begin{bmatrix} P_1 & 1 & 1 \\ 1 & P_2 & 1 \\ 1 & 1 & P_3 \end{bmatrix} \quad k=\{1,2,3\}, j=\{1,2,3\} \qquad (3)$$

where $d_{4,j}=1$, $d_{0,j}=1$, and $d_{k,0}=1$.

From Equation (3), we can find that we only need to calculate three parameters (P_1, P_2, P_3) if images are divided into four layers in total.

V. PROPOSED ALGORITHM

In this section, the proposed integrated caching algorithm will be introduced. Our proposed algorithm consists of two parts. One is a replacement algorithm presented in 1. Another is a pre-fetching algorithm presented in 2.

1 *Replacement Algorithm*

Assume that k layers of image i are in the cache, if we remove its k-th layer (the largest size layer in the cache), there will be only k-1 layers in the cache. According to Equations 2 and 3, the vector of $\boldsymbol{D}$ will be $\boldsymbol{D}_{k\text{-}1}=\{1,\ldots,1, p_{k\text{-}1},1,\ldots,1\}$.

If a user requests image i next time, the probability that he is not satisfied with the quality of k-1 layers, which means a miss hit, can be got from Equation.1.

As there are k-1 layers in the cache and a user wants to get the *k-th* layer after getting k-1 layers, the probability that miss hit happens will be as follows:

$$P(i,k) = \frac{\Omega \cdot \prod_{m=0}^{k-1} d_{k-1,m}}{r_i^{\alpha}} \qquad (4)$$

Since $\boldsymbol{D}_{k\text{-}1}=\{1,\ldots,1, p_{k\text{-}1},1,\ldots,1\}$, we can get,

$$P(i,k) = \frac{\Omega \cdot p_{k-1}}{r_i^{\alpha}} \qquad (5)$$

As a result, when the cache is full and we need to make room for the new coming object, the question is simplified to calculate the index as the Equation 5 for the objects in the cache. The data whose index is the lowest (the probability of miss hit is the lowest) will be first removed from the cache.

2 Pre-fetching Algorithm

It has been proved that most of web requests to a server are for a very small set of objects[10][15]. Recent studies suggest that prefetching those objects can further increase cache hit ratio as long as future client requests are tactfully anticipated [16][17][19]. When one prediction interval is over, we can consider the contents in the cache could also be adjusted at the same time. Here we will compare the access probability of the objects in the cache and objects in the server. If an object

in the server, which has not been cached yet, has the higher probability than some object in the cache, it will be pre-fetched in the cache.

For images *i* with the predicted ranking r_i, we assume there are *k* layers in the cache now. Based on Equation 2, we can derive:

$$d_{k,j} = \begin{cases} p_k & (k = j) \\ 1 & (0 \le j \le k-1 \text{ Y } k+1 \le j \le L-1) \end{cases} \quad (6)$$

There are two kinds of access probability for this image. One is for the cached data:

$$P(i,j) = \frac{\Omega \cdot \prod_{m=0}^{j-1} d_{k,m}}{r_i^{\alpha}} = \frac{\Omega}{r_i^{\alpha}} \qquad (j=[1, k]) \quad (7)$$

Another is for the non-cached data still in the server.

$$P(i,j) = \frac{\Omega \cdot \prod_{m=0}^{j-1} d_{k,m}}{r_i^{\alpha}} = \frac{\Omega \cdot p_k}{r_i^{\alpha}} \quad (j=[k+1,L]) \quad (8)$$

The cache system can then compare the access probability of all parts of all images. A threshold of access probability can be decided according to the cache capacity. If the access probability of one object in the cache is lower than this threshold, it will be removed from the cache. On the other hand, if the access probability of one object in the server is higher than this threshold, it will be pre-fetched from the server.

VI. Evaluation of Algorithms

A. Simulation Conditions

We assume that there are 5000 different images in the server. Images are supposed to be accessed according to Zipf-like frequency distributions [11][12]. The Zipf-like parameter α is 0.8.

The original images will be encoded into a hierarchical image format with four layers. The data size of different layers is proportional to 5:13:22:59 according to the Progressive JPEG format [14], in which the fourth layer has the largest size. Cache size is 5% of the total size of all images in the server.

In Equation5, 7 and 8, Ω, α are parameters of Zipf distribution. As a result, as long as we get p_k and r_i (image ranking), the algorithm can be carried out.

As for p_k , from Equation 3 we can find that we only need to calculate three parameters (P_1, P_2, P_3) if images are divided into four layers in total. And p_k can be directly obtained from the local cache information (miss hit ratio) when the cache sends all stored layers (1~*k*) to the user.

In our simulation, the values of *pj*, (*j=1,2,3*), they are firstly evaluated by using a group of real data as shown in [4]. We then experimentally compare our proposed algorithm with other three previous caching algorithms: LRU, OPT-SOFT, TTLLRU.

We analyze the performance of the algorithm with different access patterns. In this simulation, two extreme access situations are evaluated. In one case, an access pattern is static, where the image ranking (r_i) is not changed at all during the simulation. In the other case, an access pattern (r_i) is intensively changed temporarily in order to investigate the robustness of every algorithm under dynamic access pattern changes. We assume that the simulation time is 30 days and the ranking of all images is changed about 5% percent within every 8 days.

B. Simulation Results

Hit-ratio has been popularly used to evaluate the cache performance by most researchers [7][9][10][13][19]. Hit-ratio is defined as the ratio of request times that the data requested by the user can be directly found in the cache to the total number of requests from the user.

Fig.1 shows the hit ratio when four different replacement algorithms are used under the completely static access pattern. Not only OPT-SOFT but also TTL-LRU and the proposed algorithm outperform the commonly used algorithm, LRU, OPT-SOFT and TTL-LRU provide a modest improvement while the proposed algorithm achieves the best result.

Next, we also evaluate their performances when the clients' access pattern is intensively changed. From Fig. 2 we can see that the proposed algorithm is much better than the others, although its performance drops slightly compared with the result under the static access pattern. However, OPT-SOFT drops most among the four algorithms. The reason is that the access frequency in the last period is used to decide images' priorities. And it can't reflect the current image popularity. Although LRU showed poor results in Fig 1, its performance is very stable under the dynamic access pattern. This is because it always keeps the most recently used image in the cache. Performance of TTL-LRU is close to LRU as shown in Fig.2. The proposed algorithm works well under the dynamic access pattern, too. This is because, on one hand, prediction of popular images can prevent the replacement algorithm from removing current popular objects from the cache. On the other hand,

pre-fetching can also help the cache to keep user's favorite images in the cache.

From Fig.1 and Fig.2, we can find that our proposed algorithm is the best and the most adaptive one since it achieves the best results under both of the two extreme access patterns.

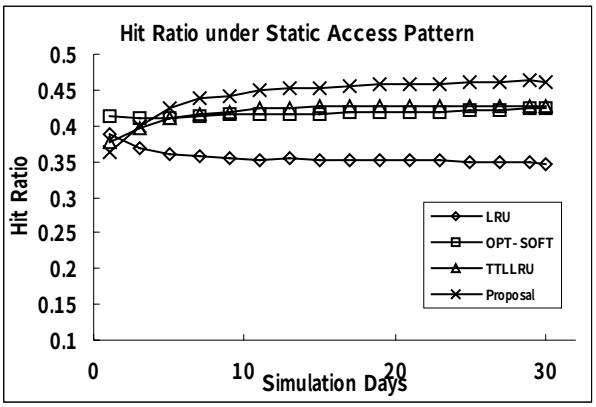

Fig 1: Hit Ratio under Static Access Pattern

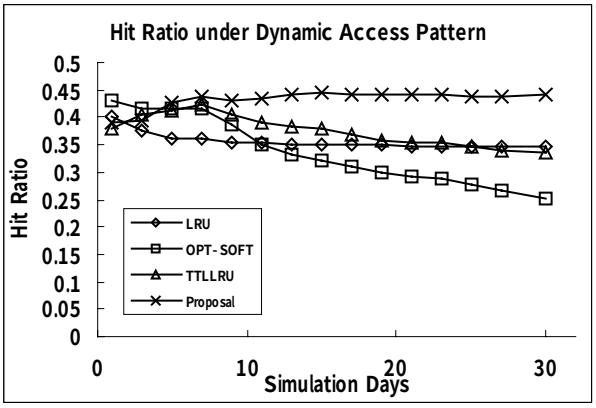

Fig 2: Hit Ratio under Dynamic Access Pattern

Fig.3 and Fig.4 compare relative response times with respect to the commonly used algorithm, LRU. According to the investigation on the range of users' access bandwidths [7], we assume that the bandwidth between clients and a proxy to be 1Mbps and the bandwidth between servers and a proxy to be 64Kbps. Fig.3 shows that the proposed algorithm has the shortest relative response time under completely a static access pattern. It can reduce the response time up to 20 %, which is almost 5 % more than the reduction that OPT-SOFT has achieved.

Fig.4 shows that the proposed algorithm can also reduce the response time around 20 % compared with LRU, when the clients' access pattern is intensively changed. Compared with the result in Fig 3, our proposed algorithm is the most stable one since its performance under the different access patterns doesn't change very much. Note that the other algorithms' performances drop drastically when the popularity ranking of images is changed. The results in Fig.1~ Fig.4 also verifies that the algorithm that gets a high hit ratio also obtains a shorter response time.

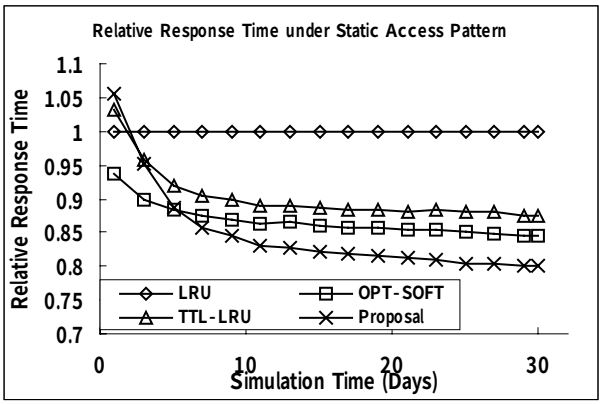

Fig 3: Relative Response Time under Static Access Pattern

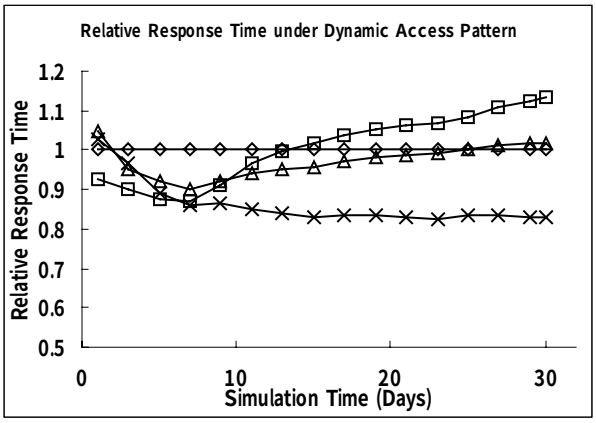

Fig 4: Relative Response Time under Dynamic Access Pattern

If web objects' access pattern can be predicted, it will be helpful to improve the cache performance. However, as there are numerous images in the server, it is unrealistic to predict all images' access frequency. Moreover, since references to images accessed only once account for a large fraction, it is not necessary to predict these once-accessed images in every prediction interval.

Previous researches showed that the distribution of web requests from a fixed group of users follows a Zipf-like distribution. It has been proved that most web requests to a server are for a very small set of objects, for example top 10 %[10]. Because of this property, it is enough to only predict that aforesaid set of images. That

is to say: it is not necessary to predict the access counts for all images because web access web follows a Zipf distribution.

In the simulation in Fig.1~ Fig.4, only top 20 % popular images' access frequency was predicted. In Fig. 5 we will test the performance of our proposal when the predicted percentage is changed from 0.0 to 1.0. Form Figure 5 we can find the hit ratio is relative low when the predicted percentage is lower than 10%. And the performance becomes stable after predicting top 20% popular images.

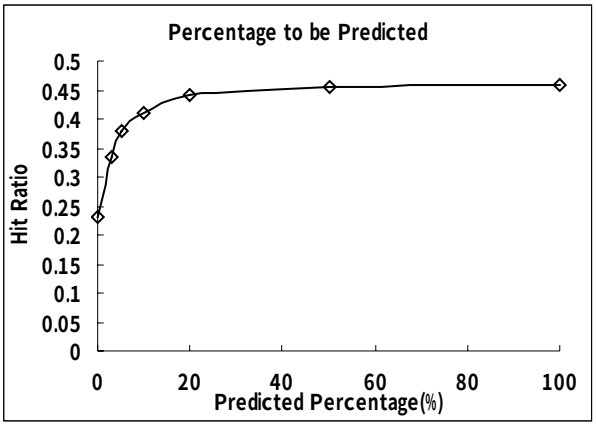

Fig.5: Performance of the Proposal vs. Predicted Percentage

The reason can be found from web access distribution. According to Zipf distribution, most of the requests are for a very small set of objects. Top 20% of the images account for about 70% of all requests seen by the cache, and top 50% of the images account for 85% of all requests [12]. When more than top 20% images' access counts are predicted, the cache can know that most of all requests are to the cached images with high probability. That is to say: the access pattern is almost predicted. As a result, the performance becomes stable. The result in Figure 5 proves the conclusion that only predicting part of popular images' access count is enough for predicting the access pattern of web access.

VII. Conclusion and Further Work

In this paper, based on the analysis we proposed an integrated caching algorithm in which both cache replacement and image pre-fetching were carried out. We quantitatively evaluated our proposal and compared our proposal with other previous ones. The results showed that the proposed algorithm outperforms all of the previous ones. Both cache hit ratio and user response time have been efficiently improved.

As an increasing number of streaming media objects have been being distributed over the Internet, caching and distributing streaming media in Content Delivery Networks by using layered encoded method will be considered as an extension of our work in the future.

References

[1] J.Kangasharju and K.W. Ross," *Performance Evaluation of Redirection Schemes in Content Distribution Networks"* , the 5th International Web Caching and Content Delivery Workshop, May 2000.

[2] R.P. Doyle, J.S. Chase, S.Gadde, A. M. Vahdat "The Trickle-Down Effect: Web Caching and Server Request Distribution"," the 6th International Web Caching and Content Delivery Workshop, Jun 2001.

[3] Z. Su, T. Washizawa, J. Katto, Y. Yasuda; "A New and Robust Replacement Algorithm for Proxy Caching with Hierarchical Image Coding" 2001Image Media Processing Symposium (IMPS2001), Nov 2001.

[4] K Saito "Performance Improvement of Caching by Using Human's Access Degree for Images '' Graduate Thesis, Waseda University Mar 2001.

[5] Y. Yasuda, T. Yasuno, F. Katayama, T. Toida, and H. Sakata, ''Image database system featuring graceful oblivion", IEICE Trans., Commun., Vol.E79-B, No.8, pp.1015-1021, August 1996

[6] K. Kamogawa, D. Nakajima, and Y. Yasuda, "Proxy server systems with hierarchical image caching", IIEEJ, Vol.27 No.5, pp.548-556, Oct 1998

[7] A. Ortega, F. Carignano, S. Ayer, and M. Vetterli, "Soft Caching: Web cache management techniques for images", IEEE Signal Proc. Society Workshop on Multimedia Signal Processing, (Princeton, NJ), June 1997.

[8] J Kangasharju, Y. Kwon, A. Ortega, X. Yang and K. Ramchandran, ``Implementation of Optimized Cache Replenishment Algorithms in a Soft Caching System", In IEEE Signal Processing Society Workshop on Multimedia Signal Processing, Redondo Beach, CA, Dec 1998

[9] M. Arlitt, R. Friedrich, and T. Jin: "Performance evaluation of web proxy cache replacement policies'', Lect. Notes Computer Sience, Vol.1469, pp.193-206 1998

[10] E. Markatos and C. E. Chronaki: "A Top-10 approach to prefetching on the Web'' In Proceedings of INET' 98, Geneva, Switzerland, July 1998.

[11] V. A. F. Almeida, M. A. G. Gesrio et al" Analyzing the behavior of a proxy server in the light of regional and cultural issues"1998

[12] L. Breslao, P. Cao, L. Fan, G. Phillips, and S. Shenker "Web caching and Zip-like distributions: Evidence and implications" Proc. IEEE INFOCOM'99

[13] T. Ishikawa, W. B. Hui, H. Ohsawa, and Y. Yasuda "A method of improving response time in still database based on CD-ROM changers by Graceful Caching'' IIEEJ, 1999,Vol 28,No 5 pp.605-611

[14] W. Pennebaker and J. Mitchell, "JPEG Still Image Data Compression Standard''.Van Nostrand Reinhold, 1994

[15] J. Gwertzman. "Autonomous Replication in Wide-Area Networks". Technical Report 17-95, Harvard University, 1995

[16] L. Fan, Q. Jacobson, P. Cao and W. Lin "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance" in 1999 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems May, 1999

[17] K. M. Koeger, D. D. E.long, and J .C .Mogul, "Exploring the bounds of web latency reduction from caching and prefetching". in Proceedings USENIX Symposium on Internet Technology and System, Dec 1997

[18] Z. Su, T. Washizawa, J. Katto, Y. Yasuda, "A New Prefetching Algorithm for Graceful Caching System", 2001 General Conference of IEICE, March 2001.

[19] Z. Su, T. Washizawa, J. Katto, Y. Yasuda, "Performance Improvement of Graceful Caching by Using Request Frequency Based Prefetching Algorithm", IEEE TENCON, Singapore, Aug, 2001.