Replacement Algorithm Selection Mechanism Considering File Size for Web Cache Server

Tanasun Sontisiri*, Pawin Sopechoke*, Sakchai Thipchaksurat* , and Ruttikorn Varakulsiripunth**

*Department of Computer Engineering, **Department of Electronics Engineering
Faculty of Engineering and Research Center for Communications and Information Technology (ReCCIT),
King Mongkut's Institute of Technology Ladkrabang,
Chalongkrung Road, Ladkrabang, Bangkok, Thailand 10520
E-mail: {s6063003, s2061088, ktsakcha and kvruttik}@kmitl.ac.th

Abstract: This paper describes the improvement of web cache server by scoping in replacement algorithm of data which are collected from the clients. We have found that each replacement algorithm is suitable for each type of data in the web pages. Therefore, we introduce the mechanism to select the replacement algorithm depending on the size of data called the Replacement Algorithm Selection Mechanism (RASM). RASM allows the web cache server to have the suitable replacement algorithm for each type of data. As the result, the byte hit ratio of web cache server can be increased and the congestion in the network can be alleviated.

Keywords: Replacement Algorithm, %Hit, %Byte-Hit ,and Threshold

1. INTRODUCTION

At present, the growth of World Wide Web (WWW) has been rapidly increased. The model of communication is client-server, which clients would contact website servers to request data for web pages. The growth rate mentioned above causes the congestion in the network. As a result, clients receive data quite slowly. To solve this problem web cache server has been developed. In the Internet, web cache server is located between a client and a web server. The client who needs the web page has to request to the web server. The web server then transfers the web page to the client. Therefore, all data will also be buffered in the web cache server. When the cache in the web cache server is nearly full, the replacement algorithm [1] will be called to select data which will be removed from the cache. Each of the replacement algorithm models has its own characteristics and will give the value of %Hit and %Byte-Hit differently.

According to the characteristics of the replacement algorithm, we introduced threshold variable into replacement algorithm of web cache server. It uses for selecting replacement algorithm between Greedy-Dual-Size with Frequency (GDSF) [2, 4] and Least Frequency Used with Dynamic Aging (LFUDA) [3, 4] to generate priority keys of data in the cache of web cache server. And web cache server will use priority keys to decide data out from cache of web cache sever.

The remainder of this paper is organized as follows. Section 2 refers to some related works. Section 3 describes Replacement Algorithm Selection Mechanism (RASM). Section 4 presents the performance of RASM. Section 5, we describe the suitable threshold for RASM. Finally the conclusion of this paper is given in Section 6.

2. RELATED WORKS

Until now, many types of replacement algorithm have been introduced in order to increase the performance of web cache server. Each type of replacement algorithm considers variables that deal with rules to choose data that will be removed from cache of web cache server to increase the value of %Hit and %Byte-Hit. It is found that there are 2 variables from replacement algorithm which are important to consider data kept in cache of web cache server - frequency of requests and age of data that is kept in cache. Replacement algorithms which consider both variables mentioned are the Greedy-Dual-Size with Frequency (GDSF) and the Least Frequency Used with Dynamic Aging (LFUDA).

2.1 Greedy-Dual-Size with Frequency (GDSF)

The GDSF algorithm was developed by Cao and Irani based on GD-Size algorithm [5]. The GD-Size algorithm performs well, but does have one significant shortcoming: it does not take into account how many times the data was accessed in the past. For example, consider how GD-Size handles two different data of the same size. If they are requested at the same time, they are inserted into a priority queue with the same priority key value (K_i). The data f_1, which was accessed n times in the past will get the same K_i value as the data f_2 accessed for the first time. In the worst case scenario, f_1 will be replaced instead of f_2.

The GD-Size algorithm can be improved to reflect data access patterns by incorporating a frequency count F_i in the computation of K_i as denoted in Eq. (1).

$$K_i = \frac{F_i * C_i}{S_i} + L \quad (1)$$

Where C_i is the cost associated with bringing data i into the cache. S_i is the data size and L is a running age factor that starts at 0 and is updated for each replaced data f to the priority key of this data in the priority queue.: i.e., $L=K_f$. This algorithm achieves the best hit rate when C_i =1.

2.2 Least Frequency Used with Dynamic Aging (LFUDA)

The LFUDA algorithm was developed from LFU and LFU-Aging algorithm [6] that can achieve the highest of %Byte-Hit. These algorithms accomplish by retaining the most popular data, regardless of data size. Unfortunately LFU-Aging requires parameterization in order to perform well. Parameter less algorithms are preferable as they are generally less complex and easier to manage. Thus, they then replace the tunable aging mechanism. They call this algorithm as Least Frequency Used with Dynamic Aging (LFUDA). LFUDA calculates the priority key value K_i for data i using Eq. (2).

$$K_i = (C_i * F_i) + L \tag{2}$$

By setting C_i to 1, this equation uses only the frequency count and the inflation factor to determine the priority key of a data. The LFUDA algorithm may be useful in other caching environments where frequency is an important characteristic but where LFU has not been utilized due to cache pollution concerns.

3. REPLACEMENT ALGORITHM SELECTION MECHANISM (RASM)

According to the characteristics of GDSF replacement algorithm and LFUDA replacement algorithm [7]. We notice the following:

- Data that have small size and have more of frequency access should be kept in cache of web cache server.
- Data that have large size and have access more than 2 times in a short time should be kept in cache of web cache server.

Thus, we can introduce the selection method for replacement algorithm for web cache server called Replacement Algorithm Selection Mechanism (RASM) as the following steps:

1. Determine the threshold that is used to compare with the size of data which will be kept in cache of web cache server.

2. If web cache server detects data that have equal to size or over threshold, it will call LFUDA algorithm to make priority key of data. Since large sized data are accessed only one time, they must have low priority key. But data that have large size and are accessed more than 2 times in a short time must have high priority key.

3. If web cache server detects data that have size under threshold, web cache server will call GDSF algorithm to make priority key of data. Since small and medium sized data are accessed many times, they must have high priority key. As for data that have small or medium size and have a few times of access, they must have low priority key.

In this paper, the threshold is determined to compare with the size value of data. It is utilized as the value to select replacement algorithm. Furthermore, the values of the parameters in priority key (K_i) of each type of replacement algorithm is modified to be able to be considered together as show in Eq. (3).

$$K_i = \begin{cases} \left(\frac{F_i}{S_i}\right) + L & \text{If } S_i < \text{Threshold} \\ F_i + L & \text{If } S_i \geq \text{Threshold} \end{cases} \tag{3}$$

Where S_i is the data size. L is a running age factor that starts from 0 and is updated for each replaced data f to the priority key of this data in the priority queue: i.e., $L=K_f$ and F_i is frequency of each data i that start from 1 when S_i < threshold. Otherwise, it will start from 0.

4. PERFORMANCE EVALUATION

We evaluate the performance of RASM to confirm its effect by computer simulation. We compare RASM with LFUDA and GDSF replacement algorithm.

4.1 Data Collection

The access log file which used for simulation RASM was collected from web cache server of the Department of Computer Engineering, King Mongkut's Institute of Technology Ladkrabang. These log file contain information of all client requests in one week. Each entry in an access log contains information on a single request received by the web cache server. The recorded information included the client IP address; the time of the request; the client's request such as method, URL, and HTTP version; the response status from the web cache server and the origin server; the amount of header and content data transfer; and the time required for the web cache server to complete its response. In this paper, we will not consider the following URLs:

- Dynamic URLs such as cgi, php, asp, etc.
- URL that contain the special symbol such as ?, =, %, +, *, &, @, $, [ ], ( ).

For considering hit rate and byte hit rate, we focus only on the lines of requests which have words such as TCP_HIT, TCP_MEM_HIT, TCP_REFRESH_HIT, TCP_REF_FAIL_HIT and TCP_IMS_HIT.

Table 1 Summary of access log file

Access Log Duration	One Week
Total Requests	338,822 Bytes
Unique Requests	78,580 Bytes
Total Content Bytes	2.61 GB

4.2 Simulation Model and Configuration

We evaluate the performance of RASM by modelling the simulation model as shown in Fig. 1

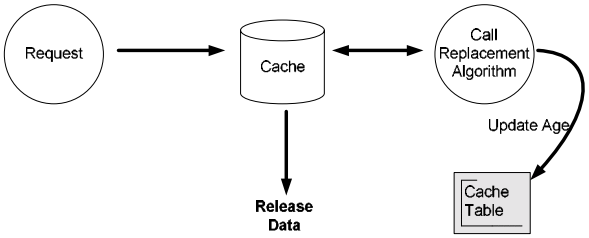


Fig. 1 Simulation Model

The simulation process can be explained as follows:

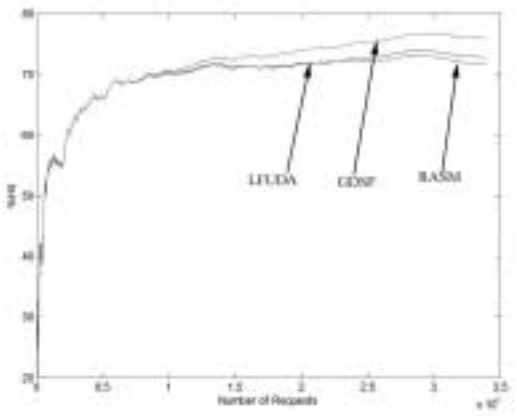
1. Define parameters of web cache server such as size of cache, type of replacement algorithm, high limit, low limit of cache, and threshold as follow:
   - The size of cache is set to 128 MB.
   - The low limit and high limit are equal to 90% and 95%, respectively.
   - Thresholds are calculated from the mode and mean of the data which have the value of 933 Bytes and 32 Kbytes, respectively.
2. Send requests of test data to web cache server one by one.
3. Check files in the cache. If a request file is a hit, its priority key will be updated and wait for a next request. Otherwise, the process will check the size of cache. If it is higher than a high limit, the replacement algorithm will start to remove files from a cache until the size of cache decreases equal a low limit. Then the replacement algorithm will stop and the new file is kept in the cache and waits for a next request.

In the real system, when we measure value of %Hit and %Byte-Hit, its value will increase more than simulation because the replacement algorithm can be divided into two processes. The first one is Memory Cache process and the second one is Disk Cache process. In the memory cache process, web cache server will check the using areas in memory cache. If they are higher than high limit of memory cache, a web cache server will use a replacement algorithm to move files from a memory cache to a disk cache until using areas come to a low limit of memory cache. In the disk cache process, web cache server also checks using areas like memory cache process. If the using areas are higher than the high limit of disk cache, a web cache server must remove files out until using areas come to a low limit of disk cache.

4.3 Simulation Results

This section provides the simulation results of the web cache server replacement algorithm. The threshold is equal to 933 Bytes and 32 Kbytes, respectively. We compare the performance of RASM with GDSF and LFUDA, respectively.

Fig.2 (a) and (b) show %Hit and %Byte-Hit for threshold value equal to 933 Bytes.. In Fig.2 (a) %Hit of RASM drop and close to %Hit of LFUDA algorithm. Fig.2 (b) %Byte-Hit of RASM get %Byte-Hit between %Byte-Hit of GDSF and LFUDA algorithm. We find that this threshold give us low performance because %Hit of RASM drops from that of GDSF algorithm a lot. The problem occurs when cache of web cache server keeps large data size which is larger than 933 Bytes. So, it has little space to keep data. Web cache server must call replacement algorithm to release data from cache many times. Consequently, %Hit will have lower value.



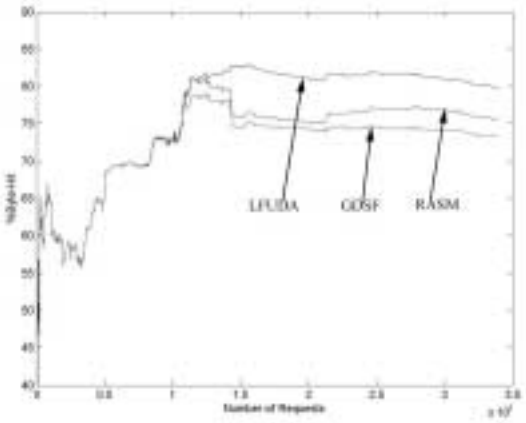
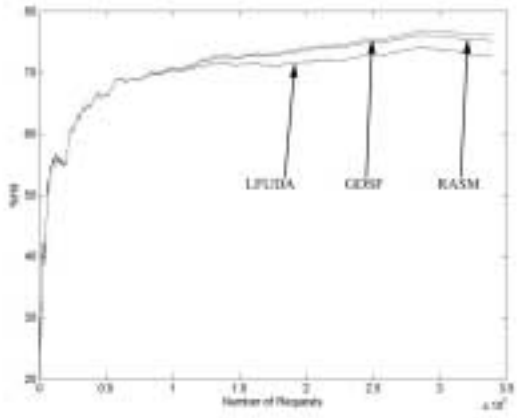


Fig.2 (a) %Hit versus number of requests for threshold 933 Bytes (b) %Byte-Hit versus number of requests for threshold 933 Bytes



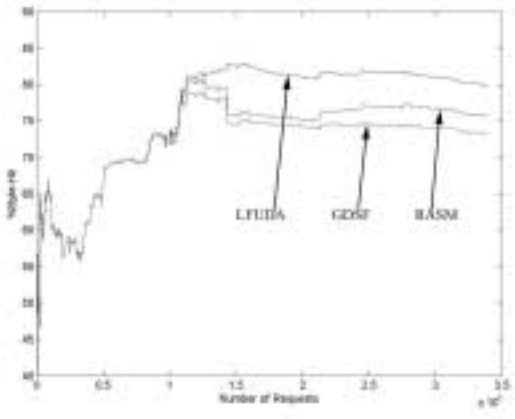


Fig.3 (a) %Hit versus number of requests for threshold 32 Kbytes (b) %Byte-Hit versus number of requests for threshold 32 Kbytes

Next, we change threshold to 32 Kbytes. The results show in Fig.3 (a) and (b), respectively. In Fig.3 (a) %Hit of RASM is close to %Hit of GDSF algorithm because when the threshold is increased, data that have larger size than the threshold are kept in cache of web cache server less than the threshold of 933 Bytes. Therefore, the cache of web cache server has space to keep data more than threshold of 933 Bytes. In Fig.3 (b) %Byte-Hit of RASM is shown between %Byte-Hit of GDSF and LFUDA algorithm. However, if we compare %Byte-Hit with %Byte-Hit of RASM for threshold equal to 933 Bytes, %Byte-Hit of RASM for threshold equal to 32 Kbytes is a little higher ,so it means that threshold of 32 Kbytes has better performance than threshold of 933 Bytes.

5. DISCUSSION

Since %Hit and %Byte-Hit of RASM cannot be proven into mathematical equation, they depend on the environment of requests, the size of data, and the frequency of requests in each data. In order to consider the suitable threshold, we do a simulating by changing the threshold from 10 Kbytes to 5 Mbytes.

We can explained as follows:

1. %Hit of RASM must decrease from GDSF algorithm at the lowest degree denoted as $\overline{\Delta H_i}(gdsf)\min$ and must increase from LFUDA algorithm highest denoted as $\overline{\Delta H_i}(lfuda)\max$.

2. %Byte-Hit of RASM must decrease from LFUDA algorithm lowest denoted as $\overline{\Delta H_b}(lfuda)\min$ and must increase from GDSF algorithm highest denoted as $\overline{\Delta H_b}(gdsf)\max$.

Therefore, we derive the following equations:

$$\overline{\Delta H_i}(gdsf) = \frac{\sum_{n=1}^{N} \frac{\%Hit_{gdsf} - \%Hit_{\mathrm{mod}\,ified}}{\%Hit_{gdsf}}}{N}, \quad (4)$$

$$\overline{\Delta H_i}(lfuda) = \frac{\sum_{n=1}^{N} \frac{\%Hit_{\mathrm{mod}\,ified} - \%Hit_{lfuda}}{\%Hit_{lfuda}}}{N}, \quad (5)$$

$$\overline{\Delta H_b}(gdsf) = \frac{\sum_{n=1}^{N} \frac{\%Byte_{\mathrm{mod}\,ified} - \%Byte_{gdsf}}{\%Byte_{gdsf}}}{N}, \quad (6)$$

$$\overline{\Delta H_b}(lfuda) = \frac{\sum_{n=1}^{N} \frac{\%Byte_{lfuda} - \%Byte_{\mathrm{mod}\,ified}}{\%Byte_{lfuda}}}{N}. \quad (7)$$

Where $\%Hit_{gdsf}$ is the %Hit of GDSF algorithm. $\%Hit_{lfuda}$ is the %Hit of LFUDA algorithm. $\%Byte_{gdsf}$ is the %Byte-Hit of GDSF algorithm. $\%Byte_{lfuda}$ is the %Byte-Hit of LFUDA algorithm and N is the number of requests.

The suitable threshold can be obtained according to the following conditions.

1. For GDSF algorithm, we calculate the different between $\overline{\Delta H_b}(gdsf)$ and $\overline{\Delta H_i}(gdsf)$ denoted as H_{gdsf}. Therefore, we get.

$$H_{gdsf} = \overline{\Delta H_b}(gdsf) - \overline{\Delta H_i}(gdsf) \quad (8)$$

2. For LFUDA algorithm, we calculate the different between $\overline{\Delta H_b}(lfuda)$ and $\overline{\Delta H_i}(lfuda)$ denoted as H_{lfuda}. So, we get

$$H_{lfuda} = \overline{\Delta H_b}(lfuda) - \overline{\Delta H_i}(lfuda) \quad (9)$$

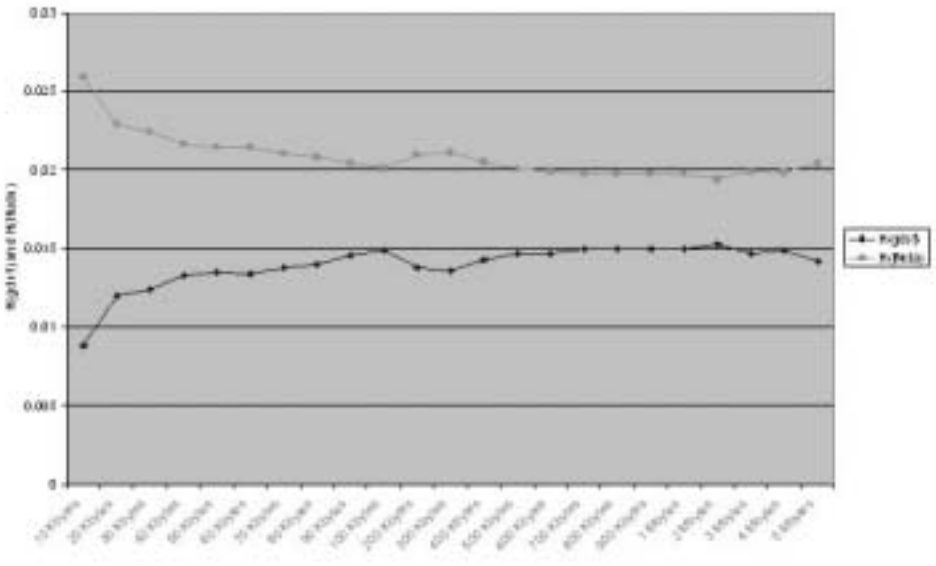


Fig. 4 H_{gdsf} and H_{lfuda} versus various value of threshold

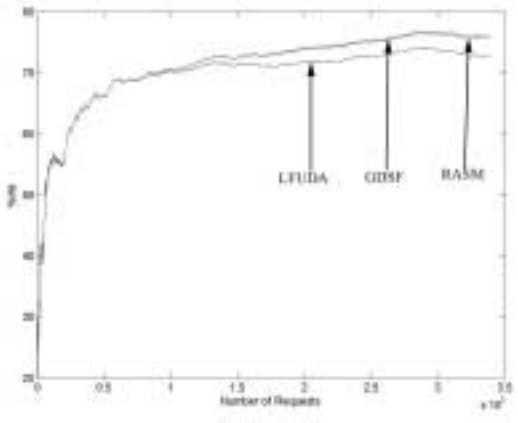
Hence we use both conditions to find suitable threshold value of data from web cache server of Department of Computer Engineering, King Mongkut's Institute of Technology Ladkrabang as shown in Fig. 4. and we get the suitable threshold equal to 2 Mbytes.

The simulation result for threshold of 2 Mbytes are shown in Fig. 5 (a) and Fig. 5 (b), respectively. In Fig. 5 (b), we can see that %Byte-Hit of RASM give the result nearly %Byte-Hit of RASM in Fig. 2 (b) and 3 (b). But in Fig. 5 (a), %Hit of our algorithm give the result better than graph %Hit of RASM in Fig. 2 (a) and Fig. 3 (a), respectively. Therefore, it may be suggested that our method for selecting may be applied for RASM.

6. CONCLUSION

In this paper, we have introduced the mechanism for selecting the suitable replacement algorithm based on the size of files contained in the web page called replacement algorithm selection mechanism (RASM). We have showed that RASM can provide the improvement over the GDSF replacement algorithm in the form of percentage of byte hit ratio and LFUDA replacement algorithm in the form of percentage of hit ratio.

For the future work, we will consider the dynamic threshold that can be changed according to the situation in the network in order to achieve higher byte hit and hit ratio of web cache server.



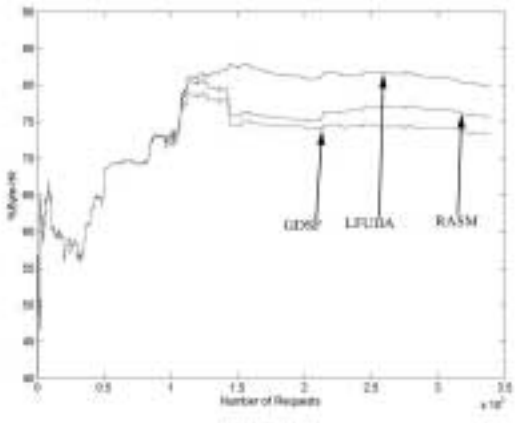


Fig.5 (a) %Hit versus number of requests for threshold 2 Mbytes (b) %Byte-Hit versus number of requests for threshold 2 Mbytes

REFERENCES

[1] J. Dilley, M. Arlitt and S. Perret, “Enhancement and Validation of Squid’s Cache Replacement Policy.” Technical Report HPL-1999-69, Hewlett-Packard Laboratories, May 1999.

[2] L. Cherkasova, “Improving WWW proxies Performance with Greedy-Dual-Size-Frequency Caching Policy.” Technical Report HPL-1998-69R1, Hewlett-Packard Laboratories, November 1998.

[3] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich and T. Jin, “Evaluating Content Management Techniques for Web Proxy Caches.” Technical Report HPL-98-173, Hewlett-Packard Laboratories, April 1999.

[4] S. Jin and A. Bestavros, “Popularity-aware greedy dual-size Web proxy caching algorithms.” Proc. 20th International Conference on Distributed Computing Systems, pp. 254-261, April 2000.

[5] P. Cao and S. Irani, “Cost Aware WWW Proxy Caching Algorithms.” Proc. USENIX Symposium on Internet Technology and Systems (USITS), pp. 193-206, December 1997.

[6] S. Podlipnig and L. Boszormenyi, “A Survey of Web Cache Replacement Strategies.” ACM Computing Surveys, vol. 35, no. 4, pp. 374-398, December 2003.

[7] P. Sopechoke and B. Piyatamrong, “Improvement Web Cache Server with Hierarchical Replacement Algorithm.” Proc. 7th National Computer Science and Engineering Conference 2003, pp. 140-144, October 2003.