

# WWW Cache Replacement Algorithm Based on the Network-distance

Masaru KAMIZATO†, Tomokazu NAGATA†, Yuji TANIGUCHI†, Shiro TAMAKI†

†Department of Information Engineering,  
Univ. of the Ryukyus

1 Senbaru, Nishihara, Okinawa, 903-0213, JAPAN

Tel.: +91-98-895-8715, Fax.: +91-98-895-8727

‡Computing and Networking Center,  
Univ. of the Ryukyus

1 Senbaru, Nishihara, Okinawa, 903-0213, JAPAN

E-mail: dai@ads.ie.u-ryukyu.ac.jp

**Abstract:** With the popularity of utilization of the Internet among people, the amount of data in the network rapidly increased. So that, the fall of response time from WWW server, which is caused by the network traffic and the burden on WWW server, has become more of an issue. This problem is encouraged the research by redundancy of requesting the same pages by many people, even though they browse the same the ones.

To reduce these redundancy, WWW cache server is used commonly in order to store WWW page data and reuse them. However, the technical uses of WWW cache that different from CPU and Disk cache, is known for its difficulty of improving the cache hit rate. Consequently, it is difficult to choose effective WWW data to be stored from all data flowing through the WWW cache server. On the other hand, there are room for improvement in commonly used cache replacement algorithms by WWW cache server.

In our study, we try to realize a WWW cache server that stresses on the improvement of the stresses of response time. To this end, we propose the new cache replacement algorithm by focusing on the utilizable information of network distance from the WWW cache server to WWW server that possessing the page data of the user requesting.

## 1. Introduction

In recent years, information services on wide area network using WWW has been widely spread. As a result, the delay of the response time of WWW services, because of the network traffic and the burden on the WWW server, has become a big problem. The network traffic and burden on the WWW server are encouraged by redundancy of requesting the same pages by many people, even though they browse the same ones.

One of the solutions for these problems is utilization of WWW cache server. A WWW cache server is what gives the WWW data caching function to the proxy server. There are a lot of programs working as the WWW cache server. For instance, squid, apache http server, CERN httpd server and Delegate server are known as proxy server.

The outline of the function of the WWW cash server is as follows. When a user demands page data from WWW and WWW cache server holds the object that the client required in the cache, the cache server trans-

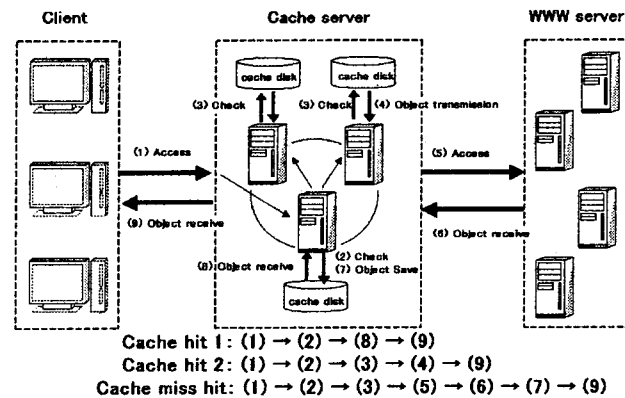


Figure 1. Web Cache System

mits the object to the client instead of requesting it to WWW server. And when the required object is not in the cache of the WWW cache server, while acquiring an object from the original WWW, it stores the object also in own cache (Figure 1) (conceptual diagram of the WWW cache server). By realizing reuse of the same object with the cache function, the amount of communications data between the WWW server and the WWW cache server are cut down, and average response time is also shortened.

On the other hand, there is no useful cache replacement algorithm, which is used commonly by the WWW server, taking the information of network-distance into account to store and manage the page data. Though, some existing cache replacement algorithms consider the size of the page data and the access frequency of the page. Therefore, from the view of improvement of the response time, existing algorithms are not sufficient.

The purpose of the study is to realize a WWW server that reduction of the stress on the improvement of the response time. To achieve the end, we propose the new cache replacement algorithm by focusing on the utilizable information of network distance from the WWW cache server to WWW server that possessing the page data of the user requesting.

## 2. Existing Cache Replacement Algorithms

Cache replacement algorithm is the method that determines page data to be evicted if there is little space for storing new ones. In this chapter, we explain three major cache replacement algorithms, Least Recently Used (LRU), Least Frequently Used with Dynamic Aging (LFUDA) and Greedy-Dual Size-Frequency (GDSF). Moreover, we sort out the problems of each algorithms.

### 2.1 LRU

LRU is one of the most commonly used cache replacement policies. LRU deletes the object that has been used (read or written) less recently than any other object. This simple algorithm uses only reference history of FIFO to evaluate the value of the data. However, there are rooms for improvement in the byte hit rate and the hit rate in this algorithm, because LRU does not consider the frequency of the reference and size of the data (Figure 2 ).

$$Key = (Last\ referred\ time)$$

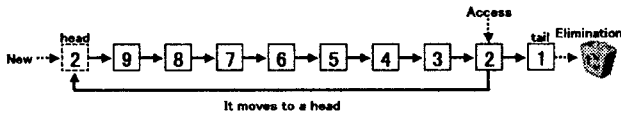


Figure 2. List of LRU

### 2.2 LFUDA

LFUDA is a variant of Least Frequently Used that uses a dynamic aging policy to accommodate shifts in the set of popular data. In the dynamic aging policy, the cache age factor is added to the reference count when a data is added to the cache or an existing data modified. LFUDA uses reference history and the frequency of the reference to evaluate the value of the data. With the use of frequency of the reference, LFUDA is expected to cache the data having possibilities of being accessed compared with LRU. However there are rooms for improvement in byte-hit rate, because LFUDA takes no account of the data size (Figure 3 ).

$$\begin{aligned} Key = & (Cache\ retention\ period) \\ & + (reference\ count) \\ & - (formula\ for\ adjustment) \end{aligned}$$

### 2.3 GDSF

GDSF is a variant of the Greedy Dual-Size policy that considers frequency of the reference. GDSF is optimized for caching more popular and smaller objects in order to maximize object hit rate. GDSF policy assigns

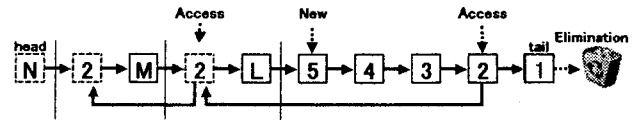


Figure 3. List of LFUDA

a value to each object computed as the object's reference count divided by its size, plus the cache age factor. By adding the cache age factor, we limit the influence of previously popular documents, as described above in the part of LFUDA. Since GDSF considers the size of the data besides the reference history and the frequency of the reference, LFUDA is expected to achieve the improved hit rate compared with LFUDA. However, there still are rooms for improvement in byte-hit rate, because GDSF caches smaller-sized data (Figure 4 ).

$$\begin{aligned} Key = & (Cache\ retention\ period) \\ & + (reference\ count)/size \\ & - (formula\ for\ adjustment) \end{aligned}$$

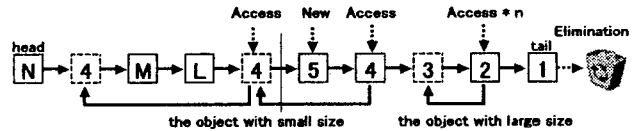


Figure 4. List of GDSF

All the algorithms introduced above does not utilize the information of the network-distance to evaluate the page data, therefore there still are rooms for improvement in the response time.

## 3. Cache replacement algorithm considering the network distance

As we mentioned in chapter 2, existing cache replacement algorithms mainly focus on the access frequency of each data, and no cache replacement algorithm considers the network-distance between the WWW cache server and the WWW server. Therefore the data that is accessed fewer and placed further network tends to be evicted rather than the data accessed more often and placed nearer. This is one of the reasons of the fall of the response time. In the other words, the existing cache replacement algorithms are not designed for improving the response time. Thus we can improve the response time by considering the information of the network-distance to the WWW server.

In our study, we suggest a new cache replacement algorithm, which is stressed on the improvement of the response time, considering the network-distance besides other factors. To prove the effectiveness of it, we examined the relationship between the network-distance and the response time.

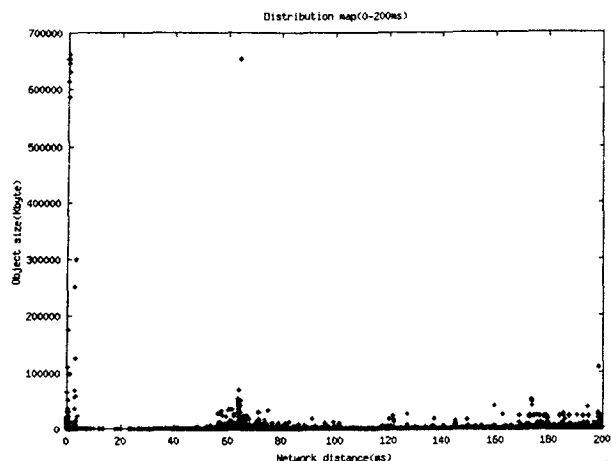


Figure 5. Relationship between the network-distance and the data size (1).

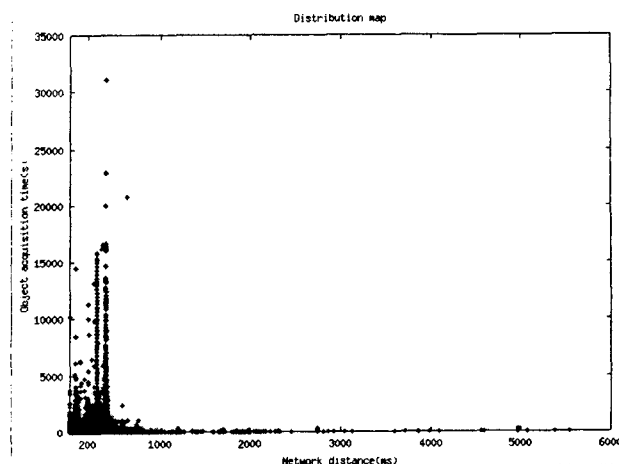


Figure 7. Relationship between the network-distance and the data acquisition time (1).

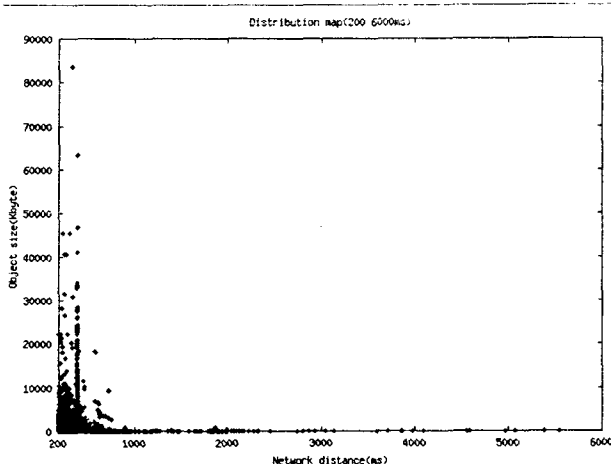


Figure 6. Relationship between the network-distance and the data size (2).

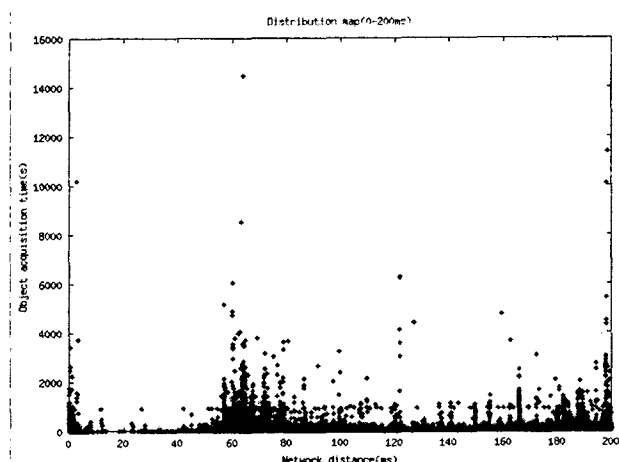


Figure 8. Relationship between the network-distance and the data acquisition time (2).

### 3.1 Analysis of log in a WWW cache server

We calculated network-distance of all WWW servers, which are picked from an access log on a WWW cache server running in the University of the Ryukyus. To this end, we measured the average response time of a transmitted ICMP echo packet to the WWW servers. As for some servers that did not permit the use of the ICMP echo packets, we used approximate value of the average response time to measurable place by traceroute command as the network-distance instead. As a result, the average of network-distance to the WWW server, which had the demand from the user, was approximately 120.2ms. The request whose network-distance was less than 67ms occupied about 50% of the whole, and the request whose network distance was less than 200ms occupied about 80%, and other 20% was dispersed between 200ms and 5545.5ms (Figure 5, Figure 6). As for the ratio of the completion time of acquisition of the data from

the WWW server, the request whose network-distance was less than 67ms accounted for 16%, and even the request to the WWW server placed nearer than 200ms was only 28% of the whole (Figure 7, Figure 8).

Judging from the result above, the request to the WWW server whose network-distance is more than 200ms, though it held only 20% of the number of the total access, accounted for no less than 72% of the overall completion time of acquisition of the object. Therefore, we can say that most of the access to the WWW server are accounted by the request whose network-distance is more than 200ms. Meanwhile, the response time become worse rapidly when the request to a WWW server is further than 200ms. Thus, we can improve the whole response time by caching the object whose network-distance is more than 200ms preferentially.

### 3.2 Cache Replacement Algorithm by consideration of Network-distance

From the above suggestions, we utilize the information of the network-distance in addition to access frequency and the size of the object. We set higher priority to the object whose data size is larger and network-distance from the WWW cache server is further, and set same degree of priority to the larger-sized object that is in the further the WWW server and the smaller-sized object that is in the nearer one. When the object is referred by the client, we arrange the page data in the priority in the reference order list to adjust the length of the stay of the page data. Moreover, we use the distance time of 200ms as the basis of the priority value, because it is thought to be effective judging from the log analysis.

Though, we use the ICMP echo packet to measure the network-distance, it is inefficient if we send it to all the servers in the log list. Therefore, we only send the ICMP echo packet to the newly accessed the WWW server, and as for the WWW server accessed in the past, we does not send it if it is accessed in 5 minutes. The reasons why we set up the period of measure to 5 minutes are that there are little difference of the response time of the ICMP echo packet sent within 5minutes, and the result of taking the load of the WWW cache server caused by sending the ICMP echo packets into consideration. As the basic value of performance evaluation, the hit rate, which is calculated by dividing the number of the pages by the number of requested object, and the byte hit rate, which is calculated by dividing the data total size of whole page that is hit by the number of total requested object are used under normal conditions.

suggesting method 1 :

$$\begin{aligned} \text{Key} = & (\text{Cache retention period}) \\ & + (\text{The network distance}) \\ & - (\text{formula for adjustment}) \end{aligned}$$

suggesting method 2 :

$$\begin{aligned} \text{Key} = & (\text{Cache retention period}) \\ & + (\text{reference count})/\text{size} \\ & + (\text{The network distance}) \\ & - (\text{formula for adjustment}) \end{aligned}$$

### 3.3 Implementation

To implement suggesting method and verify the effectiveness of the cache replacement algorithm, we used squid, a powerfull and high-efficiency WWW cache server program that is distributed under GPL(GNU General Public Liscence) and the current major release version is Squid-2.4. In the current stable release of squid, squid-2.4 Stable 6, the part of cache replacement algorithm is separated as module. Therefore we can mount the new algorithm and verify it easily.

### 4. Future works

In suggesting method, we give priory to the information of the network-distance to determine the priority of the cache object. However, there is no effective mean to measure the network-distance in real time, therefor we use the ICMP echo packet experimentally. Sending the ICMP echo packets to all the servers that is requested by the user is not realistic because the number of request per unit time is untold millions. Moreover, so far, the implementation is still experimental, therefore the evaluation formula for calculating the priority value is still quite simple. In the future, we need to run a simulation to collect data in addition to real data to calculate more sufficient valuation formula from more detailed data.

### 5. Conclusion

In this paper, we prove the effectiveness of the information of the network distance from the WWW cache server to the WWW server based on the experimental analysis of a log in WWW cache data running in a real network. Then, we propose the new cache replacement algorithm and experimentally mounted this algorithm that utilizes the information of the network-distance. To be more precise, we mounted the cache replacement algorithms, which utilizes the network-distance of the cache object besides the data size and access frequency as indication of its efficiency, and then validate the effectiveness by system operation data.

In the future, we improve the valuation formula that determine the priority of the cache object, and suggest new algorithms, which is more sufficient to improve response time , and verify it.

### References

- [1] Katsumi KISHIMOTO, Tomokazu NAGATA, Yuji TANIGUCHI, Shinji KONO, Shiro TAMAKI "Mobile Agent based Web Cache Proxy System", THE FIRST IEEE ELECTRO/INFORMATION TECHNOLOGY CONFERENCE, 2000
- [2] "Squid Web Proxy Cache", <http://www.squid-cache.org/>
- [3] "The Apache Software Foundation", <http://www.apache.org/>
- [4] "Netcraft Web Server Survey", <http://www.netcraft.co.uk/survey/>
- [5] "Status of the W3C httpd", <http://www.w3.org/Daemon/Status.html>
- [6] "DelGate Home Page", <http://www.delegate.org/delegate/>
- [7] Noritaka OSAWA, Fumitaka HAYANO, Toshitsugu YUBA, Katsuya HAKOZAKI "Evaluation of Replacement Policies in WWW Cache Based on Logs from Proxy Server", IPSJ 96-DPS-74, pp.191-196, Jan. 1996
- [8] N.Nishikawa, T.Hosokawa, H.Tsuji, Y.Mori, K.Yoshida "WWW Traffic Analysis and Distributed Cache System", IPSJ DSM Report May. 19997