5s-07 Replacing Objects in Proxy Server's Cache Based on Two Criteria

Qusai Abuein *Susumu Shibusawa*
Department of Computer and Information Sciences
Faculty of Engineering, Ibaraki University

1 Introduction

In this paper a replacment algorithm for cache objects based on two criteria is presented. It is important to choose the object to be replaced with a new object that does not reduce the hit ratio or has the probability to be requested in the near future so as not to refetch it again, while keeping an object with less probability to be requested in the near future. Previous work proposed an algorithm to choose a set of objects to replace a new object based on some criteria such as age or popularity[1].

By providing documents from its local disk cache without actually connecting to the remote host, the proxy server is an effective place to cache web objects which otherwise experience long network latencies. It also helps reducing the network traffic and distributing the server load since the object only needs to be transferred once to the proxy, and subsequent requests can be handled from the proxy[2].

Since the proxy has limited amount of storage space we should only keep the objects that will be requested frequently in the future[1], this is also going to be our aim in this paper since that also increases the hit ratio.

An object is any item retrieved by a URL, such as a textfile, image, or an audio file, ... etc. A hit is serving the client from the proxy cache. Miss is not serving the client from the proxy cache[3].

In this paper we are going to compare the replacement policies by simulating various replacement policies, such as FIFO, Least Recently Used (LRU), Least Frequently Used (LFU), then compare it to our algorithm that takes in consideration more than one criterion in replacing the object, such as size and number of requests, or size and last time accessed.

2 Defenition of the algorithm

Lets take into consideration the size of the document and number of requests. If the objects in the cache are stored according to a criterion, let it be the size, and a new object is comming, consider the following example: $o_1, o_2, \ldots, o_5$ are stored in a cache according to that criterion, where o_1 is object1, o_2 is object2, and so on, and a new object o_{new} arrived to the full cache, then

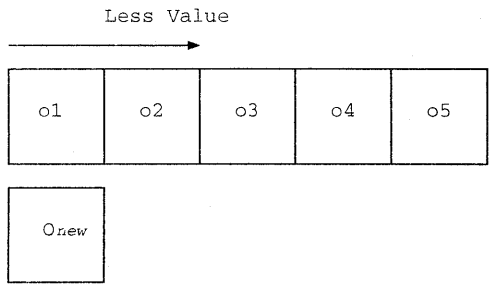

Figure 1: An example of a replacement object.

NO.	Size	NO. of requests
o1	100	200
o2	100	100
o3	200	50
o4	250	300
o5	500	50

Figure 2: An example of a replacement object based on two criteria.

it must be replaced, in the traditional caching scheme o_1 will be selected to make room for the new object, as shown in Figure 1. Consider the same example in addition to another criterion, let it be the number of requests, as shown in Figure 2. When chosing o_1 to replace o_{new} then we are removing the object that is more popular which has a No. of requests equals to 200, which is the higher probability to be requested in the future, so taking into consideration another criterion (No. of requests), o_2 will be removed which it has less number of requests (200), and that might increase the hit ratio, since the probability of o_1 to be requested in the future is more than the probability of o_2.

When using the BACK button of any browser to browse a previous page, if it is not in the cache either the proxy's or the user's local cache, it must be fetched again and it consumes time, which makes the user to wait and that what he does not like, so it is important

to keep the object that will be frequently requested in the future.

2.1 Replacement algorithm based on two criteria

We devided the cache into n parts according to the size of the objects. The first part is for the objects with size less than s_1 byte, the second part is for the objects with size between s_1 byte and s_2 byte, the last part is for the objects with size more than s_{n-1} byte. One advantage of deviding the cache is to make the competition for replacement between the objects of the same size, another advantage is to make the sorting mechanism easier and faster to perform as will be shown in the algorithm's steps. Of course we need to replace an object only when the cache is full.

In order to implement a fully functional web proxy cache, a cache architecture requires several components:

- A storage mechanism for storing the cache data.
- A mapping mechanism to establish relationship between the URLs to their respective cached copies.
- Format of the cached object content and its metadata[4].

In our paper we are going to concentrate on the storage mechanism and sort the objects in the cache based on the two criteria, the first one is considered as the primary key, the second is the secondary key.

So the example shown in Figure 2 becomes as shown in Figure 3. After sorting it based on size(primary key), and No. of requests(secondary key). If the request is a

NO.	Size	NO. of requests
o2	100	100
o1	100	200
o3	200	50
o4	250	300
o5	500	50

Figure 3: objects stored in the cache based on size and No. of requests.

hit then that object's No. of requests is increased, so we need to resort the objects of the part of the cache again. As mentioned before the resorting is easier and faster when dealing with cache as more than one part.

step 1 If hit then resort objects in that part of the cache.

step 2 Find an object equal to or greater than the new object o_{new} in the cache part.

step 3 replace the o_{new} with the found object (since the found object has the least No. of requests value between the objects in that part of the cache, and its size is equal to o_{new} then o_{new} is replaced in its place and no need to resort the objects again).

We define the hit ratio as follows:

$$hit\ ratio\ = \frac{\#\ of\ requests\ served\ from\ cache}{total\ \#\ of\ requests}$$

The hit ratio is then increased while the objects requested are still in the cache, and that the removed object is not requested more than the existed ones, taking into consideration the object is still up to date, means that the property If-modified-since is less than the date in which the request is done.

From the example showed, we can notice that if there are objects equal in size, differ in No. of requests, our algorithm can choose the proper object to replace.

3 Conclusion

By this algorithm it can be sure that the object with the appropriate size and with the least number of requests (least popular) is removed and that the objects which are more popular remained, and when requested it can be served to increase the hit ratio.

We are going to make a simulation program to test our algorithm, compare the result to other replacement schemes by analysing logfiles.

References

[1] Ilhwan Kim, Heon Y.Yeom, Joonwon Lee, Analysis of Buffer Replacement Policies for WWW Proxy, Proc. of 13th Int. Conf. Information Networking (ICOIN 98).

[2] A. Chankhunthod and P.B.Danzing, A hierarchial Internet Object Cache, Harvest Cache Project, Nov. 1995.

[3] Stephen Williams, Marc Abrams, Charles R.Standridge, Ghaleb Abdulla, Edward A.Fox, Remote Policies in Network Cache for World-Wide Web Documents, Computer Communication Review, ACM SIGACOMM, Vol.26, No.4, Oct. 1996.

[4] Ari Luotonen, Web Proxy Server, Prentice-Hall, 1998.