An Integration Approach of Data Mining with Web Cache Pre-Fetching

Yingjie Fu[1], Haohuan Fu[2], and Puion Au[2]

[1]Department of Computer Science
City University of Hong Kong, Hong Kong SAR
fuyingjie@tsinghua.org.cn
[2]Department of Computer Engineering and Information Technology
City University of Hong Kong, Hong Kong SAR
fu.haohuan@student.cityu.edu.hk; poau@it.cityu.edu.hk

Abstract. Web caching plays a very important role for improving the performance of many Web-Based systems. As web cache capacity is limited, most web cache systems are using replacement algorithm to wash out the outdated data. Our web cache prediction method is based on the fact that, many clients usually have some kinds of regular procedures to access web files, such that the regular-procedure knowledge can be mined or learned by web cache system and files can be pre-fetched accordingly .

1 Introduction

As networks become the basic infrastructure for data sharing and communication, web server response time becomes a very important measurement factor of the network and server performance. It is known that a server side web cache is placed between a web server and the clients; and the web cache buffers the copies of files requested by the clients. The usage of web cache is to improve the server responding time, decrease server load, and reduce latency to minimize "World Wide Waiting" problem.

In traditional web cache systems, file replacement policies are used to handle the limited cache size. There exist numerous replacement policies, such as LRU (Least Recently Used), LFU (Least Frequently Used), SLRU [1], LRU-MIN [2] and SIZE [3], etc. However, these pure replacement-policy strategies do not associate with any

predication approach, which can enable the servers to find out what the client's access-profiles are, and the use this knowledge to predict the files to be accessed. In order to enhance web caching performance, not only the replacement policies can be applied, web cache pre-fetching technologies are also frequently used. Web cache pre-fetching needs the server-side cache to predict what kind of information would be used in the near future. In this paper, we present a new approach for web cache prediction to achieve better performance of file hit rate and byte hit rate.

2 Algorithm

At the beginning of data mining process, the web log records are loaded into a user-request pool p, which includes user record lists $l_1, l_2 \ldots l_n$. Users are identified by IP address. In each user record list l, it contains the user-accessed file *URL*s in a time ascending order.

As shown in Fig.1, history request pool is a table of user-request lists ls, and each l is a list of accessed file *URL*s. For better performance, user-request pool is designed to be a hash table with user IP as primary-key. When a request comes, according to the request source IP address, new record will be added to the user-request pool p.

User-req pool p	User request list l_1	Accessed file URLs of l_1
	User request list l_2	Accessed file URLs of l_2
	…	…
	User request list l_n	Accessed file URLs of l_n

Fig. 1. User-request Pool

There are numbers of accessed file *URL*s in each user-request list. Let T_{min} be the minimum threshold of *URL* number. In data mining process, the user-request lists with more than T_{min} *URL*s will be marked. Since minor *URL*s do not reflect user-access profile, we believe only these marked user-request lists will contribute to the data mining process. On the other hand, the *URL* number in each l should not grow unboundedly. When it grows to an upper threshold T_{max}, old *URL*s can be treated as outdated and will be washed out (deleted) from the list. It is obvious that, very old data source will not be useful for new data mining because both the user-accessing profile and web server files may have been changed already.

A new data structure is used to record user access orientation for each file *URL*. As shown in Fig.2, access-orientation is an integer to measure user accessing orientation for a file. It is also a hash table with file *URL* as primary-key.

URL_1	URL_2	…	URL_k
access-orientation$_1$	*access-orientation*$_2$	…	*access-orientation*$_k$

Fig. 2. Access orientation hash table

At the beginning in the data mining process, the value of *access-orientation*s are set to zero. While the data mining processing carrying on, they will be increased respectively. Our data mining process is not complicated. Suppose the last request from the users is URL_{last}. For each marked user-request list *l*, find out all of the *URL*s are that equal to URL_{last}. If there are N_1 URL_1s behind URL_{last}, add N_1 to *access-orientation*$_1$; if there are N_2 URL_2s behind URL_{last}, add N_2 to *access-orientation*$_2$…until add N_k to *access-orientation*$_k$. Finally, the web cache prediction can be done according to the calculation of *access-orientation*s. Our data mining procedure and web cache pre-fetching procedure can be presented into the following pseudo code:

```
Load web log into user-request pool p;
Mark the user-request lists ls containing more than T_min records
FOR each l having more than T_max records
    Wash out outdated records;
END FOR
FOR each latest URL request
    FOR each URL_j in URL_1...URL_k
        Set access-orientation_j to 0;
    END FOR
    Get the latest requested URL URL_last;
    FOR each l_i in l_1… l_n
        Find all URL==URL_last;
        FOR each URL_j in URL_1...URL_k
            Set integer N_j=0;
            Count the number (N_j) of URL_js that follow URL_last;
            access-orientation_j= access-orientation_j+ N_j ;
            END FOR
        END FOR
    END FOR
```

```
Find out M URLs having the largest access-orientations from
 access orientation hash table, denoted as URL_p1, URL_p2…URL_pM;
FOR each URL_j in URL_p1, URL_p2…URL_pM
    IF URL_j NOT in web cache
        copy URL_j into web cache;
    END IF
END FOR
END FOR
```

Fig. 3. Algorithm Pseudo Code

In line 20, M can be assigned according to web cache size and whole system performance. Larger the web cache size is and better the system performance is, larger value is assigned to M. As for URL_{p1}, $URL_{p2}…URL_{pM}$ in line 22, there are two possible results. Either web cache will be full and cannot cache all of them, or web cache is large enough to hold all them. In the first case, the *URL*s that can not be cached will be discarded and web cache prediction process continues whereas in the second case, web cache prediction process will continue directly. In previous researches, there are a lot of contributions on replacement algorithm, e.g., FIFO, LRU, LFU, LRU-MIN, and SIZE etc. Web caching hit-rate and byte hit-rate can be better under the cooperation of web cache prediction method and web cache replacement algorithm. In our experiment, FIFO and LRU are used for the web cache replacement mechanism.

3 Simulation

We did the trace-driven simulation to compare the performance of our data mining approach with some well-known cache replacement algorithms. We choose three basic replacement algorithms, LFU, LRU and SIZE, which respectively consider three most basic factors, reference number, last access time and data size. We have done the trace-driven experiment for our data mining approach with 10 different cache sizes, compared with other five different cache replacement algorithms.

Fig. 4 shows the different hit rates for different algorithms. And among the different algorithms, the DM 600 means the data mining approach uses the latest 600 request-history records to do prediction, and DM 1200 means the approach uses 1200 the latest request-history records. In the web cache, LRU is used as wash-out ap-

proach. We could see that, from pure LRU to LRU plus web cache prediction, a lot achievement has been gotten. And from this figure we could see that the data mining approach has much better performance over any other pure replacement algorithms.

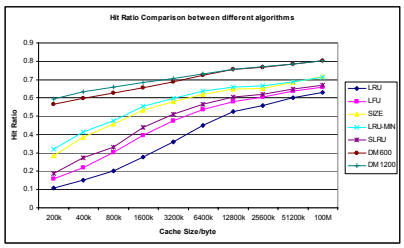

Fig. 4. Hit-rate comparison

4 Conclusion

There is a response time necessity in web accessing. In order to reduce to response time, web cache is used. Web cache prediction and pre-fetching is a very good method to improve the web response time. In this paper, we introduced a new data mining method to do prediction. The specification of the method approach and practical approach were discussed. Under the practical approach, we did experiment, and achieved satisfactory experiment results demonstrated the efficiency of integration which uses web cache prediction method and web cache replacement algorithm together in order to get better performance.

References

1. C. Aggarwal, Joel L. Wolf and P. S. Yu, *Caching on the World Wide Web*, IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, January/February 1999.
2. M. Abrams, C. R. Standridge, G. Abdulla, S. Williams and E. A. Fox, *Caching proxies: Limitations and potentials*, 4th International World-wide Web Conference, pages 119-133, Dec, 1995.
3. S. Williams, M. Abrams, C. R. Standridge, G. Abdulla and E. A. Fox, *Removal Policies in Network Caches for World-Wide Web Documents*, Proceedings of ACM SIGCOMM, pp. 293-305, 1996.
4. *Web_Log_2003_04.data*, City University of Hong Kong, April 2003
5. *Web Trace, uc.sanitized-access*, ftp. ircache.org, 18th, November 2003