中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-05-015

Design and Implementation of Domain-Based Proxy Prefetching

Ray-I Chang, Jan-Ming Ho

September 2005 || Technical Report No. TR-IIS-05-015

http://www.iis.sinica.edu.tw/LIB/TechReport/tr2005/tr05.html

Design and Implementation of Domain-Based Proxy Prefetching

Ray-I Chang

Department of Engineering Science

National Taiwan University

Jan-Ming Ho

Institute of Information Science

Academia Sinica

Abstract

Users are usually interested in some specific domains while surfing the Internet. Based on such *domain-preferential* browse-behavior, the Domain-Top (DT) proxy prefetching method is proposed. DT uses the popular pages in the same popular domain to model users' future demands. If there is a request for any one of the pages in the popular domain, the popular pages in the same domain are considered as its future demands and will be prefetched. The development of DT prefetching is based on a hypothesis that the browse-behavior is always domain-preferential. However, clients may explore the Internet aimlessly and will aceess different domains in the near future. Analyzing proxy logs without considering diverse browse-behavior may acquire wrong anticipation in prefetching. This paper proposes the DTC (DT prefetching with Classification) method that tries to improve DT prefetching by removing unreliable logs. DTC adopts the concept of entropy to discriminate the browse-behavior from "domain mode" and "exploratory mode". Only access logs in domain mode are considered in calculating the popular domains. Different from DT that considers a constant number of popular pages in prefetching, we ssign each domain a suitable number of popular pages. Experiments on real traces show that

the proposed DTC method can achieve higher hit ratio than that of the DT method. As DTC utilizes only the historical logs to offline decide the popular pages and the popular domains for prefetching, only few function modules on the present proxy need to be revised. It imposes small burden and can be easily implemented in Squid -- the most famous open source proxy server.

Keywords:
proxy caching, web prefetching, open source software

2005/9/14

1. INTRODUCTION

Due to the exponential growth of network traffic, users bear significant latency in surfing the Internet. A lot of time is wasted to wait the requested web pages to come up on screen. The way to improve the performance for accessing request pages become one of the hottest research topics in the Internet community. In past years, researchers have dedicated considerable efforts to address this challenging problem [2], [16]. Among these techniques proposed, proxy caching is the best-known one. Caching pages on local proxies can significantly decrease the number of requests sent to servers [14]. Therefore, both the Internet traffic and the access latency are reduced. Define the hit ratio, a common performance measurement of cache, as the fraction of all data reads which are satisfied from the cache. The hit ratio of a proxy is increasing as its cache size increases. Since the cache size is limited in a proxy, the hit ratio obtained depends on the request repetitiveness of pages cached. Traditional proxy caches the pages that are just demanded only. As the trend of browse-behavior is not well thought-out, the caching scheme may not perform as well as desired [1]. To resolve this drawback, the technique of prefetching is investigated to predict the browse-behavior of clients and to pre-cache their request pages.

Proxy prefetching deduces clients' future requests and gets those pages into the cache before an explicit request is made for them [12]. Usually, users' future demands were predicted by their historical access pattern. An additional data structure (tree or link-list) is applied to model the access frequency of pages [11], [13]. When a client request is sent, its future demands are predicted and prefetched by following this data structure. Then, if these pages are requested in the near future, they will not need to be accessed from the remote server. In conventional approaches (either the Top-10 approach [11] or the Path Profiles approach [13]), only the access frequencies of individual pages are considered. However, in real cases, users are usually interested in some specific domains and will browse more pages on the same domains. Therefore, users' demands in the near future can be successfully predicted by the popular pages in the same popular domain. To consider such *domain-preferential* attribute of browse-behavior, the Domain-Top (DT) method [14]

is proposed to consider the domain a page belongs to, rather than the page itself. By accumulating clients' access domains and pages, the DT method first identifies the popular domains that are requested most frequently. Then, in each popular domain, it selects a *constant number* of popular pages that are demanded most frequently. The collections of domains and their pages are applied to make prediction. If one of the popular pages is requested, the related popular pages in the same popular domain are prefetched. Otherwise, the general caching mechanism is applied.

In the DT method, given a constant size of cache, the request repetitiveness of domains and pages prefetched would affect its hit ratio. By tracking the cache replacement policy, a domain or page that is good for prefetching should be requested repetitively within *a certain period*. However, in the DT method, domains and pages of the entire access patterns (within the whole day, not a certain period) are accumulated directly. Notably, the development of DT prefetching is based on the hypothesis that the clients considered are orderly and domain-preferential. But some clients in real cases may explore the Internet aimlessly and there is no rule to guess which domains or even pages they will access in the near future. To analyze these information for prefetching may acquire wrong anticipation. As the DT method does not take the reliability of browse-behavior into consideration, it may not attain the effective results but wastes the extra bandwidth. In this paper, we consider the classification of browse-behavior in DT prefetching to resolve this drawback. Moreover, our DTC (DT prefetching with Classification) method tries to provide suitable numbers of prefetching pages for different domains. For example, a popular domain with many popular pages may require more pages in prefetching. It is different from the DT method that uses a constant number of pages, always 8 pages, in prefetching.

DTC adopts the concept of entropy to differentiate the reliability of browse-behavior. Given a clickstream of browsing domains and pages, it uses a sliding window to calculate the entropy of access logs in a certain period of time. Then, based on its entropy, we discriminate this period of browse-behavior from "domain mode" and "exploratory mode". When a clickstream belongs to the domain mode, the client prefers the specific domains and it usually has high reliability in prefetching. However, when clients are in the exploratory mode, it is more difficult

to guess what will be accessed next. Therefore, the accuracy of prefetching is low and the price to be paid in terms of extra bandwidth is high. Different from the DT method, our DTC method considers only access logs with domain-preferential attribute to acquire more precise browse-behavior. It ignores unreliable logs in calculating the popular domains. Additionally, this browse-behavior model is applied to assign suitable numbers of prefetching pages for domains with different degrees of popularity. Experiments on real traces show that the proposed DTC method can help proxy to predict future requests efficiently. As DTC utilizes the historical logs to offline calculate the top domains and pages, only few functions need to be revised on the present proxy server. It imposes really small burden and can be easily added into Squid -- the most famous open source proxy server.

2. RELATED WORKS

The fast growth of Internet services introduces a huge amount of network traffic. It leads serious performance degradation in terms of user latency on the Internet. Several studies [1], [3], [7], [8], [10] have been conducted on the utilization of proxy servers to cache pages on local. Since the cache space is limited, the space usage should be suspiciously managed to improve its hit ratio. For example, when a new page arrives, the proxy server should decide which page(s) is purged to store the new one if the cache space is insufficient. Nowadays, a proxy server usually follows the LRU (Least Recently Used) replacement policy [18] to remove the cached pages that are unlikely to be accessed in the near future. If these cached pages are requested in the near future, they will not need to be accessed from the remote server. Therefore, the network resource consumed as well as the access latency perceived can be reduced. However, as traditional proxy servers cache only web pages that are just demanded, studies have showed that even an ideal cache with infinite space would achieve a hit ratio of 40%-50% only [8].

One way to further increase the cache hit ratio is to anticipate clients' future requests and prefetch them into a local cache. The technique relies on the prediction table that is made from the historical access patterns of clients. Then, proactively preloads web pages into the cache to facilitate near-future accesses [18]. In general, the prefetching scheme can be applied in various environments. The most common case is between proxies and web servers. Kroeger [8] has shown that a proxy with both caching and prefetching schemes could provide at best a 60% latency reduction. With the local proxy caching alone, it could reduce latency at best by 26% only.

Based on the information applied in the prediction tables, conventional prefetching schemes can be classified into two kinds. One is to consider the individual pages that accessed very frequently. The other is to prefetch the interrelated pages that have high correlation. A sequence of requests can be viewed as a path. In the Path-Profiles (PP) approach [13], the profiles of clients' access paths are applied to predict their future requests. It bases on the complex probability to measure the correlation of web pages. Then, a tree structure is applied to model this access relation for every client. To predict the future path, a client's current path

is passed and is matched with the tree. Based on a tree of the relationships of web pages, the PP approach can predict the future requests more accurately. However, it needs a large storage space and more computation time to maintain the tree structure. In addition, it was designed to update the tree structure for each coming request. It is so expensive and so difficult for implementing in a real proxy system. The Top-10 (TT) approach [11], which adopted popularity-based predictions, uses historical access patterns to measure the access frequency of every individual page. It calculates a list of 10 most popular pages as the prediction table. When a request is sent to a server, the proxy will prefetch the top-10 popular pages of server. The TT approach is simple and easy-to-calculated. However, it is too simple to have a significant improvement. Moreover, current servers are not designed to calculate any list for popular pages.

Notably, in real cases, users usually access the pages in a specific domain. For example, when the period of Olympic Games, user may browse many pages in the same domain within a certain time and will request more pages in the future. If the proxy prefetches these pages in advance, the prefetching schema can further reduce the user access latency. To deal with this domain-preferential situation, Wong and Yeung [17] propose the Site-Based approach. They collect the top-domain (called hot-site in [17]) list from a modified browser which repeatly checks the request of current user. If the requested domain appears in the list, the browser will forward this request to the proxy. Otherwise, it bypasses the proxy and retrieves the page from the corresponding remote Web server directly. Only the requests of the top-domain can use the proxy. Based on the similar concept, the Domain-Top (DT) method [14] has been proposed for prefetching. Usually, clients are grouped to have similar characteristics in surfing webs and may take place in some specific domains. The DT algorithm classifies the domains by clients' historical profiles and finds the domains that are requested frequently, called top-domains. Then, proxy selects the constant numbers of top-pages (called top-documents in [14]) that are demanded most frequently in each top-domain. Finally, top-domains and top-pages are collected to make a simple prediction table.

A trace-driven simulation with logs gathered from a proxy server in KAIST between Feb. 1999 and Apr. 1999 (about 460000 requests) is applied to examine the

DT method. Comparing by the hit ratio obtained with Squid, the improvements of the DT method are ranged from 20% to 450% for different days. However, it considers only 40 clients in the same department. They may have the same navigation strategy. After analyzing millions of access patterns, Cunha and Jaccoud [5] found two different kinds of navigation strategies of clients. One is the 'surfering user' who is more interested in exploring the cyberspace and always visits the fresh and new pages. The other is the 'conservative user' who is more concerned with exploring pages in a certain domain. The prefetching scheme is effective only if the navigation strategies of clients are correctly predicted. In the DT method, the entire access patterns are directly accumulated to decide the popular domains and pages. As the incredibility of clients' browse-behavior is not handled, it may make wrong prefetching and waste extra network bandwidth and cache storage. In this paper, we propose the DTC (DT prefetching with Classification) method, which considers the classification of browse-behavior in DT prefetching, to resolve these drawbacks. Moreover, different from the constant number of top-pages applied in DT, we base on the popularity of different domains to dynamically determine the number of top-pages. The algorithm is simplicity and imposes small burden on the proxy.

3. PREFETCHING BY OUR DTC METHOD

In a general web environment, the characteristic of clients' access patterns are varying. This variation must be considered in prefetching. How to accurately distinguish the browse-behavior from input requests is a very important problem. Like DT, our DTC method calculates clients' requests to decide popular domains and popular pages. Figure 1 shows the architecture of the system. The preprocessing module and the analyzing module cope with the situations mentioned above for deciding popular domains and popular pages. They are off-line processes and only work while the proxy server is in leisure periods. The third component, the on-line prefetching module, decides whether a page should be prefetched or not.

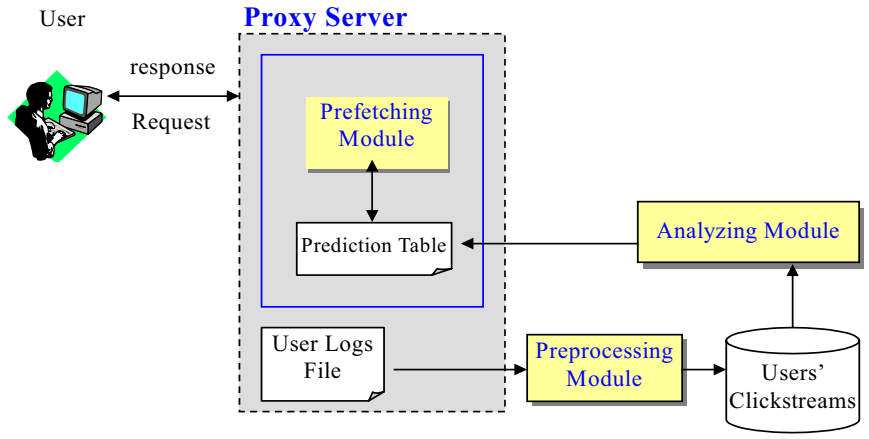

Figure 1. System Architecture.

When a client accesses web pages through a proxy server, the request information will be stored in the proxy log trace. The log provides, for each HTTP request, the time of the request, the user's IP address, the requested URL and Web server, and the referring URL. It is a historical trace of browse-behavior and can be applied to predict client's future requests. Notably, there may have two categories of web pages, static pages and dynamic pages. A dynamic page created by a web server when the page is requested (e.g., .cgi, .asp and .php scripts) is not suitable to be cached or prefetched. Therefore, this kind of information must be filtered out before predicting client's future requests.

Observing the log data, we need to know which client requested which page on what time called clickstreams. Table 1 presents a sample clickstream. The alphabet "T" indicates the request time. "W" means the request domain (website), and "P" is the request page. The suffix number shows the sequence of request.

Table 1. Users' Clickstreams.

Users	Clickstreams
U_1	$\{T_1,W_2,P_3\}\{T_2,W_3,P_1\}\{T_5,W_1,P_1\}\{T_{11},W_1,P_3\}$
U_2	$\{T_3,W_2,P_2\}\{T_4,W_2,P_3\}$
U_3	$\{T_8,W_7,P_3\}\{T_9,W_1,P_3\}\{T_{13},W_2,P_1\}$

The task of analyzing module is to make the Top-List. The list is composed of constant numbers of Top-Domains and variable numbers of Top-Pages. Top-Domains list store the most popular *n* domains. The number *n* is a constant number depend on situation of proxy. There are variable numbers of popular pages in each domain rely on the degree of popularity of domain. Accordingly, the first step is to calculate the ranking of domain. Early research [14] calculates the popularity of domains using whole log trace. However, as mentioned above, there are two kinds of browse-behavior on Internet. One group navigate the web pages aimlessly, called e*xploratory mode.* The other, called d*omain mode*, explores the pages in a certain domain. Browse-behavior in *exploratory mode* is incredible and difficult to predict. It violates the prefetching assumption that most browse-behavior has rules and could be forecasted. Hence, the clients' requests in exploratory mode should be removed to raise the accuracy of predictable table.

We propose the concept of entropy to differentiate the reliability of browse-behavior. In another word, the distribution of domains on each client's clickstream must be considered. We use the following entropy formula:

$$E(X) = -\sum_{i=1}^{n} P_i \log\left(P_i\right) \Big/ \log\left(\mathrm{n}\right)$$

where *n* is the number of domains that the user browsed in a certain period, variable *i*

indicates i^{th} domain and P_i shows the probability that client navigated the domain i. The higher entropy means the higher scatter of domains. We suggest a threshold E_{max}, if the entropy higher than the threshold these clickstreams in the certain period will be withdraw.

When we calculate the entropy, the other important issue is how to proper identify the individual user. It is inadequate to use IP address alone. Our research combines the sliding windows with IP address to estimate the entropy of clickstreams in a certain period. We divide every client's clickstreams into several sessions depending on sliding windows. To calculate the entropy of every session and eliminate sessions that their entropy higher than threshold. Finally, we obtain the precise ranking of domains by its access times. After the Top-Domains were created, we determine the popular pages (Top-Pages) in each domain. In the process of making the Top-List, the number of domains and pages in each domain must be considered. The previous research [14] used the constant number of pages in each domain. However, the degree of popularity of each domain is different, the constant number lacks the flexibility and cannot reflect the real clients' interests. Therefore, the DTC method adopts the variable number depending on the popularity of domain. The number of pages in each domain is defined by:

$$N_x = \left(DN_{size} \times \frac{PG_{size}}{2} \right) \times \left(S_x \Big/ \sum_{x=1}^{DN_{size}} S_x \right) + \frac{PG_{size}}{2}$$

where DN_{size} is the number of Top-Domains. PG_{size} is a constant number and S_x is the frequency that domain X was requested by client in a certain period. Top-List is a two-dimension list as shown in Figure 2.

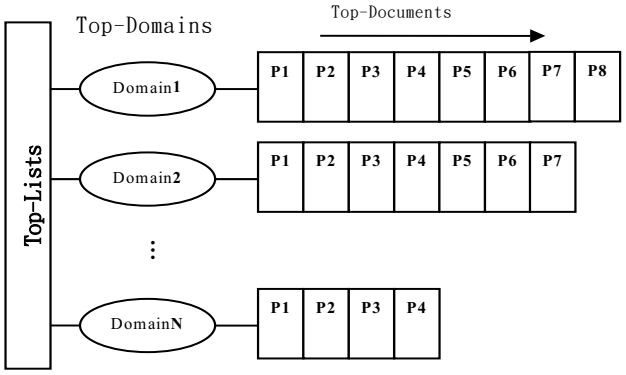

Figure 2. The structure of Top-List.

Finally, we investigate two questions. First, how often a new Top-List should be released. The time interval should neither be too long nor too short. We dissect several log files and find that for most log traces the best interval is one day. And second question is when to rebuild the Top-List. In measuring access numbers of requests by time line, we can find that in the morning, there are few requests to the server. We use this period for making the predictable table. We generate the Top-List when proxy is not busy. After Top-List is completed, prefetching engine can determine what to prefetch. Though additional burden imposes on the proxy, it is so trivial in this period.

Prefetching module is an on-line process. The function is very simple. While client requests a page belongs to a Top-Domain, then the module prefetch all pages in same domain from remote server automatically in advance. The premise of prefetching is most clients access one page in a certain domain, i.e. there is high possibility to request other pages in same domain. And popularity domains and pages have the more high possibility to be request. Hence, Top-List store the most popular domains and pages in each popular domain. It can reduce the user latency while client requests other pages in the Top-List. Figure 3 shows the prefetching flow.

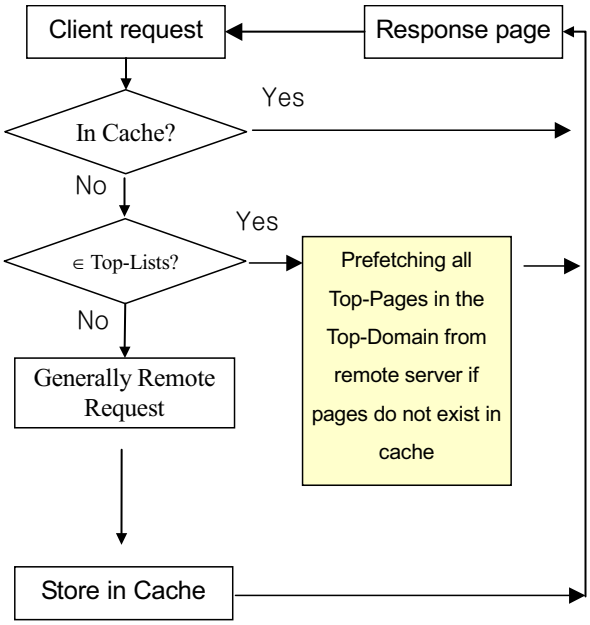

Figure 3. The prefetching flow.

4. EXPERIMENT RESULTS

This section presents trace-drive simulations based on real dataset to examine the performance obtained. Table 2 shows the detailed trace information. The traces are gathered from a proxy server of the computer center of NCU (National Central University) (http://proxy.ncu.edu.tw, port 3128). The time period is from May 10, 2004, to May 29, 2004. There are more than 27,502,000 requests and the total number of clients is about 36,100. We can find the amount of requests and clients in weekend is fewer than weekdays. And there are fewer information in the last week since it is the week of graduation examination in NCU.

Table 2. Traces Information from May. 10, 2004 to May. 29, 2004

Date	0510 Mon.	0511 Tues.	0512 Wed.	0513 Thur.	0514 Fri.	0515 Sat.	0516 Sun.	0517 Mon.	0518 Tues.	0519 Wed.
Clients	2468	2406	2429	2317	2101	1259	1383	2210	2234	2260
Requests	2181588	2099091	1740685	1959941	1625226	1182322	1020683	1885906	1862264	2034726
Date	0520 Thur.	0521 Fri.	0522 Sat.	0523 Sun.	0524 Mon.	0525 Tues.	0526 Wed.	0527 Thur.	0528 Fri.	0529 Sat.
Clients	2291	2072	1179	1007	1531	1363	1505	1609	1579	946
Requests	1940849	1159441	350383	721406	624886	875999	1203659	1291234	1116384	625507

Our comparison focuses on the DT method and our DTC method. The performance measures are the hit ratio and the prefetching effect [5], [11], [14]. The hit ratio is the bnumber of requests that hit in cache as related to the total number of requests.

$$\text{Hit ratio} = \frac{\text{the number of requests that hit in cache}}{\text{the total number of requests}}$$

The prefetching effect is the number of prefetching pages that hit in cache divided by the total number of requests that hit in cache.

$$\text{Prefetching Effect} = \frac{\text{the number of requests that are prefetched in cache}}{\text{the total number of requests that are in cache}}$$

Before evaluating the performance, some parameters must be determinated first. In this paper, we use several log files to analyze what values of these parameters would be more appropriate. As mentioned above, the entropy threshold influences the hit ratio and the prefetching effect. Observing the figure, a threshold higher than 0.6 may not filter the surfering clients. If the threshold is lower than 0.6, the creditable domains may be eliminated. In this paper, we set the value of threshold as 0.6 since it can obtain better hit ratio and prefetching effect. Another parameters are the sliding window size and the overlay size. From the obtained results, we set the window size as 8 and the overlay size as (1/4 * window size). Moreover, we consider the number of Top-Domains and Pages in each domain. Experiment shows when keeps 24 Top-Domains can obtain best performance. In order to evaluate fairly, the number of Top-Domain is set 20. The number of Top-Pages is 8 in the DT method. Thus the total amount of Top-Pages no more than max number of Top-Domains multiplies by eight in our DTC method. In this case, for example, is 20 * 8, and the number of Top-Pages in each domain is four least. In addition, we must take account of the time interval for the Top-List calculation. It should be neither too large, nor too small. Previous researches show the one day is a better period [5], [14]. Accordingly, we calculate the Top-List in the morning, the spare time of proxy server, every day.

Figure 4 presents the comparison of hit ratio between DT and DTC. The statistics in Figure 4 report that the hit ratio of DTC method exceed the hit ratio of DT method in most traces. However, Figure shows that Date 15 get very high hit ratio and Date 16 the hit ratio of DTC method is lower than DT method. Why did the phenomenon occur? We explain as follows.

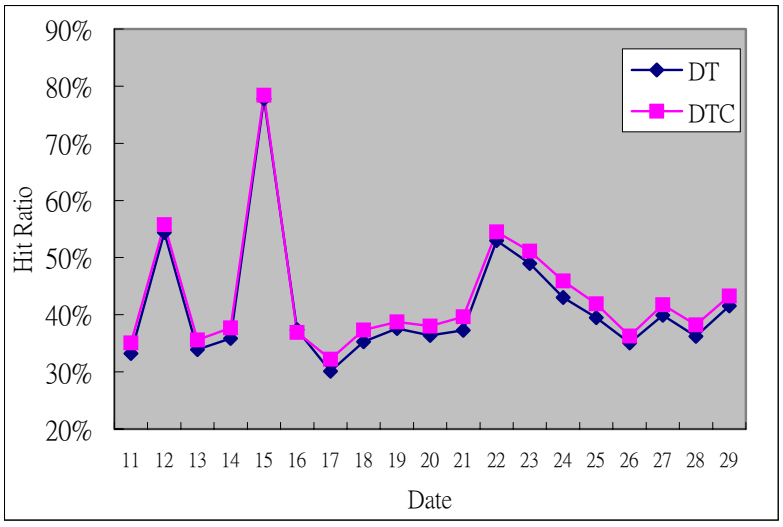

Figure 4. Comparison of hit ratio between DT and DTC.

The Date 15 is Saturday and Date 16 is Sunday. The possible reason is "weekend symptom." Generally, there are fewer students in the school while weekend. Hence, the number of requests is relative low and the browse-behavior differs from weekdays. Using the trace of one day ago to predict the user-behavior may observe dissimilar circumstances. However, why the second weekend (Date 22 and Date 23) shows more normal than first weekend? We think the cause is the week is examination week of graduation. In examination period, student usually search similar domain. Since the diversity of browse-behavior on Web, various policies should be development in the future.

In addition to hit ratio, another performance metrics we use in our experiment is "Prefetching Effect." It represents the "predictable ability" of one prefetching algorithm. The higher this ratio is, the lower the client latency and the server load. Figure 5 compares this metric for the DT method and our algorithm. The result shows that the DTC policies outperform DT. Because our model removes requests that clients browse the Web in exploratory mode, these requests are incredibility and difficult to predict the browse-behavior. Therefore, discriminating the browse-behavior accurately is important in prefetching algorithm. And we can find the reselble "weekend symptom" in this evaluation.

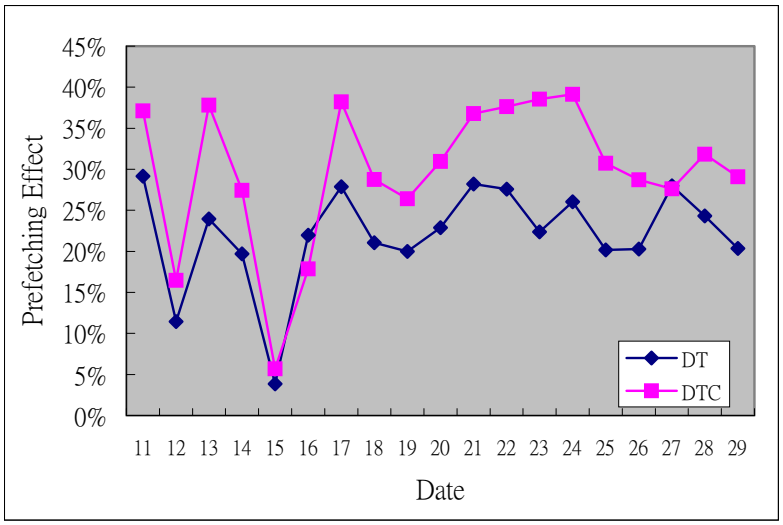

Figure 5. Comparison of prefetching effect between DT and DTC.

5. CONCLUSION

In this paper, a new proxy method called DTC is proposed to improve the hit ratio and the prefetching effect. Different from the conventional DT method, DTC can handle highly changeable characteristics of web requests. Using the concept of entropy to discriminate browse-behavior, the incredible information is removed before making the prediction. It gets a more precise rules in proxy prefetching before pages are actually requested by clients. Comparing with DT, our DTC algorithm can achieve both high hit ratio and high prefetching effect. Additionally, the applied scheme is simple and easy-to-calculating. It is a lightweight design and can be implemented in real systems easily.

REFERENCES

[1] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams, and E.A. Fox, "Caching Proxies: Limitations and Potentials," In Proc. of 4th International World Wide Web Conference, pp.119-133, 1995.

[2] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," IEEE Communications Magazine, 2000.

[3] J. C. Bolot and P. Hoschka, “Performance Engineering of the World Wide Web,” In Proceedings of the Fifth International WWW Conference, 1996, Paris, France.

[4] L. Brunie, J. Pierson, and D. Coquil, "Semantic collaborative web caching," the Third International Conference on Web Information Systems Engineering (WISE), pp.30-39, 2002.

[5] X. Chen and X. Zhang, “A Popularity-Based Prediction Model for Web Prefetching,” IEEE Computer Society, Vol. 36(3), 2003.

[6] C. R. Cunha and C. F. B. Jaccoud, "Determining WWW user's next access and its application to prefetching," In Proceedings of Second IEEE Symposium on Computers and Communications (ISCC'97), 1997.

[7] S. Glassman, “ A Caching Relay for the World Wide Web,” In Proceedings of the First International WWW Conference, 1994.

[8] T. M. Kroeger, D. D. E. Long, and J. C. Mogul, “Exploring the bounds of web latency reduction from caching and prefetching,” in Proceedings of the USENIX Symposium on Internet Technology and Systems, pp. 13-22, 1997.

[9] R. Malpani, J. Lorch and D. Berge, “Making World Wide Web Caching Servers Cooperate,” In Proceedings of the Fourth International WWW Conference, 1995.

[10] E. P. Markatos, “Main Memory Caching of Web Documents,” In Proceedings of the Fifth International WWW Conference, 1996, Paris, France.

[11] E. P. Markatos and C. E. Chronaki, "A Top-10 Approach to Prefetching on the Web," Proceedings of INET’98, pp.276-290, 1998.

[12] Nanopoulos, D. Katsaros and Y. Manolopoulos, "A Data Mining Algorithm for Generalized Web Prefetching," IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 5, 2003.

[13] S. Schechter, M. Krishnan, and M. D. Smith, "Using Path Predict HTTP Request," Seventh International World Wide Web Conference, 1998.

[14] S. W. Shin, B. H. Seong, and D. Park, "Improving World-Wide-Web Performance Using Domain-Top approach to Prefetching," the Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, Vol. 2, pp.738-746, 2000.

[15] N. Swaminathan and S.V. Raghavan, "Intelligent Prefetch in WWW Using Client Behavior Characterization," International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp.13-19, 2000.

[16] J. Wang, "A survey of web caching schemes for the Internet," Computer Communication Review, Vol. 29, No. 5, pp. 36-46, 1999.

[17] K. Y. Wong and K. H. Yeung, "Site-Based Approach to Web Cache Design," IEEE Internet Computing, Vol. 5, pp.28-34, 2001.

[18] J. Xu, J. Liu, B. Li, and X. Jia, “Caching and Prefetching for Web Content Distribution,” IEEE Computing in Science and Engineering, Special Issue on Web Engineering, Vol. 6(4), pp. 54-59, 2004.