Utilizing Group Prediction by Users' Interests to Improve the Performance of Web Proxy Servers

Tsozen Yeh
Department of CSIE
Fu Jen Catholic University
New Taipei City, Taiwan
yeh@csie.fju.edu.tw

Liangtzu Chang
Department of CSIE
Fu Jen Catholic University
New Taipei City, Taiwan
tzu98@csie.fju.edu.tw

***Abstract*—Companies and institutions often use web proxy servers to service the multiple requests of the same web pages (or web objects) from users therein to save the network bandwidth and reduce the Internet latency. Web proxy servers usually are geographically close to their clients (users). If web proxy serves have cached valid copies of requested web objects, they can be directly delivered to users. Otherwise, users need to spend a long time on getting web objects from their hosting web servers. Both cases will require users to wait for their requests. This period of latency can be largely reduced if web proxy servers could predict what web objects users may need in the near future, and send those predicted web objects to the client sites before their actual usage. Various predicting algorithms, such as those based on temporal locality and data mining, had been proposed to enable proxy servers to make prediction of what web objects users may access. However, they often need to constantly update and maintain their models in realtime to make their schemes effective. As a result, for proxy servers servicing a large number of users, schemes demanding for complicated and realtime calculation will not be as useful as expected in practice. Based on websites and web pages commonly visited by users, we proposed a new model with an offline learning algorithm to help web proxy servers make prediction about upcoming requests of web objects from users. Compared with the hit ratio achieved by the original environment without prediction, our model can improve the caching performance of users' web browsers by up to 51.37%.**

***Keywords*-web performance; web proxy server**

I. INTRODUCTION

The amount of traffic on World Wide Web has greatly increased in recent years. Consequently, this could increase the Internet latency that users experience, in particular for large companies or institutions with many users inside. Web proxy servers (proxy servers) could shorten the Internet latency by caching web objects geographically close to their clients [1]. Often a proxy server is assigned for the same group of users. Users make requests of web objects through their designated web proxy server if they cannot find them cached in their local web browsers. Unfortunately, the limited memory space allocated for individual web browsers can only cache a small number of web objects accessed recently. If a proxy server has a valid copy (not missing nor stale) of the requested web object, then it will be sent to the requesting client's site directly. Otherwise, the proxy server will get a new copy of the web object from the web server hosting that web object. After that, the proxy server will send the web object to the client making the request.

Even if the proxy server holds the required web object, the requesting user still needs to wait for its delivery. The waiting issue gets worse if the proxy server needs to contact the hosting web server to obtain a fresh copy of that web object. Researchers had proposed different ways to enable proxy servers to make prediction of web objects likely to be accessed by users in the future [2]–[4]. If proxy servers can make correct predictions, those predicted web objects can be sent to their corresponding clients before they are actually needed later. The period of latency users experienced could therefore be largely reduced. Many predicting algorithms used techniques such as temporal and spacial localities [5], [6], data mining [7], as well as Markov model [8], [9] to make prediction. The effectiveness of predicting models mainly lie in three parts. The first one is how accurately the proxy server can make correct prediction. Obviously this is the key factor affecting the overall outcome. The proxy server will not be very helpful if it fails to make correct prediction most of the time. The second part relates to the number of web objects predicted each time. Theoretically, the more web objects predicted and prefetched to users each time, the better chances that users can get what they want from those predicted web objects. Nevertheless, we could easily clog the network if we predict and prefetch an extremely large number of web objects to the client's site each time. This will eventually prolong the client's waiting time. The number of web objects predicted should be limited to a degree not consuming too much of the network bandwidth. The last one concerns the cost of maintaining those predicting algorithms. Almost all previous predicting algorithms require online learning, which means they need to constantly update their learning model after receiving every user's request. In other words, depending on the realtime learning cost, the real results of their methods could vary significantly.

We proposed a new technique, which makes prediction of web objects for possible future access by users. The main idea is that users with the same interests are likely to access certain popular websites related to their common interests, somewhat related to concept of nearest neighbors [10]. Through processing web traces collected from proxy servers offline, our model can establish certain number of groups (called interest groups), one for each different kind of interest, for users under these proxy servers. Each interest group will keep information (such as domain names) of websites belonging to that group and popular web objects from those websites as well as users with the interest of that group. For example, lovers of basketball would often visit the official NBA (National Basketball Association) website (www.nba.com), and other popular basketball related websites such as ESPN (www.espn.com). Our model will record popular basketball websites (such as www.nba.com and www.espn.com) for the basketball group, and keeps a list of popular web objects from those websites.

For each web request, we check if the requesting user belongs to any interest group (or groups). If so, our model will predict and prefetch up to a certain number of popular web objects from websites within all interest groups that user had joined. Since most people do not change their interests overnight, our model does not have to do online learning to maintain effective prediction. We tested our design on the proxy server traces over a period of three months collected from our institution. Through simulation, compared with the hit ratio achieved by local web browsers without prediction, we can improve the caching performance of web browsers by up to 51.37%.

The remainder of this paper is organized as follows. Section II reviews previous works related to proxy servers and proxy prediction. Section III describes the design of our scheme. Section IV presents experimental results. Section V concludes this paper, and the future work is discussed in Section VI.

II. Related Work

As stated, proxy servers could reduce the Internet latency users perceived. Not surprisingly, how to better the performance of proxy servers is an import issue, in particular for busy environments where many users access Internet simultaneously. A predictive proxy server can aid in the amelioration of its performance. The idea of prediction and prefetching has long been adopted in operating systems. The huge speed difference between memory and disk noticeably slows down the performance of filesystems. Early research works had enabled operating systems to predict what files running programs will access later and prefetch them from disk to memory [11], [12]. Similarly, enabling proxy servers to make future access prediction of web objects that users may require could remedy the accompanying latency [1]–[3], [5], [6], [13]. Some schemes used more complex techniques such as Markov models and data mining to find out potential access patterns of users, and then make predictions based on those findings [7]–[9], [14]. In the meanwhile, it is common that individual web pages contains multiple web objects (such as hyperlinks). Consequently, web objects embedded in web pages are good candidates for the proxy server to prefetch them to the client requesting those web pages [2], [5], [15].

It is desirable for proxy servers to keep hot web objects cached in memory instead of in disk just like what regular filesystems would do. Intelligent cache replacement algorithms can help proxy servers realize this goal [16]–[19]. Some organizations set up multiple proxy servers working together. As a result, sharing the contents among affiliated proxy servers also helps their performance as well [20].

III. Grouping by Users' Interests

In this section, we will explain how our model establishes user interest groups based on offline learning and how to make prediction accordingly. People often do not change their interests within a short period of time. Generally speaking, for a group of users with the same interest, it is logical to infer that they will visit some popular websites related to their common interest as the NBA example mentioned earlier. Thus, web objects with high popularity in those related popular websites will have a fairly good possibility to be accessed by individual users within that interest group.

A. Grouping Scheme

As described, we do not need to train our model online. We use web traces collected to train our model to establish "interest groups" during the learning process. Our model does not impose any minimum length on web traces, even though a longer period of training trace could provide better learning results as our experiments revealed.

The learning process consists of two steps. The first step is, for every user (identified by IP) in the trace, we create an interest group to record users sharing common interests with that user. We define two users sharing the same interests if both have visited at least twenty common websites. After the first step, for each user, we will have one interest group to record all other users sharing common interests with that user. For convenience, we refer to each user as the "host" for his or her interest group, and view the other users in the group as "member users". Of course, a user may not have common interests with other users. When the first step is done, for any given group, not all users in that interest group share common interests among themselves. They only share common interests with the "host" of the group.

For the second step, for each group, we examine if all users (except for the host of the group) in it share common interests among one another. For instance, assume that there are thirty-one users in a group hosted by a user A. For each of the thirty member users in this group headed by the

user A, we will check how many other member users share common interests with that member user. If the number is less than 10% of total member users, that member user will be removed from this group. Therefore, in this example, a member user will be eliminated if that member user shares common interests with fewer than three other member users in this group. The main purpose of the second step is to make sure that all users in the same group share common interests with at least 10% of other member users.

Suppose twenty member users were removed from the interest group headed by user A in the second step. After the two processing steps, there are eleven (the user A and other ten member users) users left in this group. Each of the ten member users and the user A have a common set of websites visited as observed in the first step. All websites from the ten sets are collected and referred as the "member websites" of this interest group. After that, within the training traces, all web objects from those "member websites" are gathered and ranked by their popularity (times of access). We refer to them as the "member web objects" of the interest group hosted by the user A, and they will be the candidates of web objects being predicted subsequently. Besides being the host of his or her interest group, each user can also be the "member user" of other interest groups. In fact, after the two processing steps, one user can exist in multiple interest groups simultaneously. The role of "host" or "member" does not matter then.

The reason for us to use the number, 20, to decide if two users share common interests is as follows. We assume that users will visit the same websites related to their interests at least once a week. A recent Nelson survey revealed that, on average, a person visits 97 websites within a month [21]. We divided the number 97 by 4 and obtained a quotient 24.25, we then chose the number, 20, to evaluate if two users have common interests. We also conducted experiments with value of 30, instead of 20. The results for the outcome of using 30 were only slightly worse than those of using 20. Using the value of 30 means employing a more stringent condition to check if two users share common interests. Therefore, we would establish fewer interest groups as well as fewer "member websites" and "member web objects" in each interest group. As to the number, 10%, we did not have strong theories behind that percentage. We used that number as a minimum standard for a user to stay in an interest group. Interestingly, we also tried to use 20%, and we found that the performance of our model was not very sensitive to that number either. We believed this was because our model did not actually predict many member web objects for each prediction as discussed later. Consequently, those popular web objects predicted will not vary noticeably under such circumstances.

B. Grouping Timing

Since our model performs the two learning steps offline, we can do the training at anytime. Our experience showed that it is not necessary to frequently retrain our model in practice unless an excessive number of new users join the service coverage of the proxy server. Our experiments indicated that the proxy server could maintain its comparable performance even it was trained weeks ago. Therefore, the actual cost of keeping our model working appropriately is much lower than that of predicting models requiring online training.

IV. Performance Evaluation

To evaluate our model, we used real-life proxy traces collected between October of 2010 and December of 2010 from one of the two main proxy servers at our university. The proxy server was running Squid, a popular tool installed on many proxy servers. We conducted simulation experiments for each of the three months to examine how our model performed for different sections of web traces. It would be desirable if a predicting model has stable performance across different periods of time. For each month, the traces of its first ten days were used to train our model, while data for the rest of that month were used to evaluate the performance of our model. All non-cacheable records were filtered out from the Squid log beforehand since the contents of non-cacheable web objects, such as those dynamically generated, would vary each time. The numbers of IPs recorded in each of the three months were 2713, 1612, and 1606 correspondingly. The numbers of cacheable web objects therein were around 9.4 million, 11 million, and 10 million respectively. Our goal is to examine how much better our predicting model can help the proxy server do its job. Therefore, instead of listing the performance improvement for thousands of individual users (IPs), we report the global performance difference in terms of total numbers of hits collected from browsers on all users' sites for cases with and without our prediction scheme.

A. Experimental Design

We mentioned earlier that a predicting model should not predict and prefetch too many web objects at a time to clog the network. The average size of a web object from web traces between September of 2010 and February of 2011 at our institution was about 24.5 KB. It is common for an environment with a bandwidth of 1 Gbit per second at present time. This means a regular environment can handle about 5350 (1 Gb / 24.5 KB) web objects per second. Practically, not all users will stay online twenty-four hours a day at the same time. To be conservative, let's assume that had been the case in our traces. Hence, we can afford about two (5350 / 2713) web objects per person for every second, translating into 120 (2 * 60) web objects per minute for everyone. A study showed that the number of web objects in an average web page was about 100 in November 2012 [22].

In other words, a user can request about 1.2 (120 / 100) web pages per minute in our environment, which approximately matches a recent Nelson survey reporting that the duration of a web page viewed was about 65 seconds in May 2012 [21]. We want to point out again that the number, 1.2, was calculated under the stringent assumption that the maximum number of users, 2713, in our traces had simultaneously stayed online twenty-four hours a day.

We set our predictive proxy server to predict and prefetch as most 50 web objects each time during our simulation. This means that when the proxy server received a webpage request from a user, it predicted and prefetched totally up to 50 most popular "member web objects" from all interest groups that user had joined, to the cache of the web browser on that client's site. As a reminder, the proxy server did not always predict 50 web objects at a time. The number, 50, just served as the upper limit. We were also interested in exploring how the length of training period would affect the performance of our model. We tested cases with different lengths of training periods.

B. Experimental Results

The first set of experiments was conducted against the trace data for October of 2010. We trained our model with different learning periods all starting from October 1. Five training periods with the lengths of two, four, six, eight, and ten days were conducted separately. We wanted to examine how our model reacted to different length of training.

After the training, we tested our model against individual testing days including October 11 to October 15, October 20, and October 30 to observe how our model worked in weeks after its training. A client's request of a web object can be either a hit or a miss from his or her local web browser. The "hit" requests from browsers of every client site were added up to obtain the overall number of hit. Similarly, the numbers of web objects requested for every user were summed up to get the overall number of request. We then divided the overall number of hit by the overall number of request to acquire the overall hit ratio. Table I lists overall hit ratios for different cases. The leftmost column shows the testing days. The next column, NP (No Prediction), represents the original situation where the proxy server did not make any prediction. The next five columns are for those with different training periods. Each testing day holds two rows. The upper one shows the hit ratios for corresponding training periods. The lower one represents the relative performance improvement over the original situation with no prediction, which was obtained by dividing the hit-ratio difference (between cases with prediction and without prediction) by the hit ratio without prediction.

For the training of a short 2-day trace, compared with the original case with no prediction, our model improved between 10.38% (testing day 10/11) and 29.51% (testing day 10/12). For a given testing day, except for October 30, our model did better with a longer training period. Take the October 12 for example, the improvement began with 29.51% for a 2-day training, and ended with 46.72% for a 10-day training. The high hit ratios for the October 11 draw our attention. We examined the data collected for that day and found that the number of requests was about only one-tenth of data volume for other testing days. We did not know the exact reason. Our model performed better with a longer period of training in general. However, the performance difference among different training periods is indeed not that much. In other words, our model can do comparably well even if the training trace covers only a short period of time. Figure 1 displays the results of Table I. The X-axis lists periods with different training days. The Y-axis represents the hit ratios from all testing days.

The second set of experiments was done like the first set. The testing trace was for November of 2010, and we trained our model with learning periods all starting from November 1. Again, after the training, we tested our model against individual testing days including November 11 to November 15, November 20, and November 30. Table II details the results. Basically Table II resembles what we saw in Table I. Overall speaking, our model delivered higher hit ratios when it was trained with periods covering more days. Nevertheless, even with a training of 2-day trace, our model can learn quickly and produced comparable results from a

Table I
HIT RATIOS AND IMPROVEMENT FOR TRACES OF OCTOBER 2010

testing days	NP	2-day training 10/1-2	4-day training 10/1-4	6-day training 10/1-6	8-day training 10/1-8	10-day training 10/1-10
10/11	49.76%	54.92%	55.73%	56.87%	57.43%	57.96%
(imp.)		(10.38%)	(12.01%)	(14.29%)	(15.42%)	(16.48%)
10/12	27.76%	35.95%	36.25%	38.75%	40.18%	40.73%
(imp.)		(29.51%)	(30.59%)	(39.61%)	(44.76%)	(46.72%)
10/13	28.49%	36.38%	36.69%	38.86%	40.10%	40.68%
(imp.)		(27.70%)	(28.77%)	(36.39%)	(40.75%)	(42.77%)
10/14	28.52%	36.09%	36.35%	38.09%	39.29%	39.81%
(imp.)		(26.55%)	(27.48%)	(33.57%)	(37.76%)	(39.59%)
10/15	27.03%	34.17%	34.43%	36.25%	37.23%	37.62%
(imp.)		(26.45%)	(27.40%)	(34.13%)	(37.76%)	(39.18%)
10/20	28.87%	35.49%	35.76%	37.80%	38.76%	39.11%
(imp.)		(22.92%)	(23.86%)	(30.92%)	(34.22%)	(35.45%)
10/30	27.86%	34.09%	33.03%	35.54%	36.18%	36.37%
(imp.)		(22.37%)	(23.00%)	(27.57%)	(29.89%)	(30.56%)

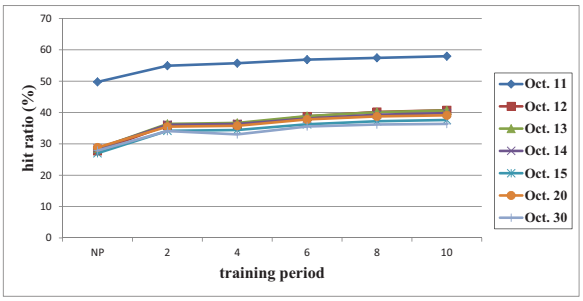

Figure 1. Test Traces: October 2010

Table II
HIT RATIOS AND IMPROVEMENT FOR TRACES OF NOVEMBER 2010

testing days	NP	2-day training 11/1-2	4-day training 11/1-4	6-day training 11/1-6	8-day training 11/1-8	10-day training 11/1-10
11/11 (imp.)	29.63%	36.74% (23.98%)	39.58% (33.56%)	41.42% (39.79%)	41.53% (40.15%)	42.31% (42.77%)
11/12 (imp.)	29.95%	36.30% (21.22%)	39.06% (30.40%)	40.65% (35.72%)	40.73% (36.00%)	41.33% (38.01%)
11/13 (imp.)	31.82%	37.76% (18.65%)	40.49% (27.24%)	41.93% (31.78%)	42.00% (32.00%)	42.49% (33.54%)
11/14 (imp.)	37.25%	44.26% (18.81%)	46.22% (24.07%)	47.63% (27.87%)	47.72% (28.11%)	48.05% (28.99%)
11/15 (imp.)	42.46%	48.36% (13.89%)	49.88% (17.47%)	51.43% (21.12%)	51.51% (21.33%)	51.83% (22.06%)
11/20 (imp.)	27.44%	32.54% (18.60%)	34.63% (26.23%)	35.59% (29.71%)	35.64% (29.91%)	35.97% (31.11%)
11/30 (imp.)	30.08%	35.04% (16.47%)	37.04% (23.12%)	37.98% (26.27%)	38.03% (26.41%)	38.31% (27.36%)

Table III
HIT RATIOS AND IMPROVEMENT FOR TRACES OF DECEMBER 2010

testing days	NP	2-day training 12/1-2	4-day training 12/1-4	6-day training 12/1-6	8-day training 12/1-8	10-day training 12/1-10
12/11 (imp.)	25.51%	32.69% (28.15%)	34.28% (34.37%)	34.46% (35.07%)	35.96% (40.97%)	38.61% (51.37%)
12/12 (imp.)	45.86%	51.63% (12.59%)	52.71% (14.94%)	52.79% (15.12%)	53.50% (16.67%)	55.62% (21.28%)
12/13 (imp.)	33.87%	40.70% (20.17%)	42.37% (25.11%)	42.64% (25.91%)	43.62% (28.80%)	46.87% (38.39%)
12/14 (imp.)	27.00%	33.78% (25.11%)	35.28% (30.63%)	35.40% (31.09%)	36.73% (36.02%)	39.30% (45.53%)
12/15 (imp.)	25.01%	32.08% (28.25%)	33.61% (34.35%)	33.75% (34.93%)	35.30% (41.14%)	37.61% (50.34%)
12/20 (imp.)	30.87%	36.36% (17.79%)	37.46% (21.34%)	37.61% (21.84%)	38.61% (25.08%)	40.10% (29.92%)
12/30 (imp.)	22.18%	26.88% (21.18%)	27.83% (25.49%)	27.89% (25.76%)	28.57% (28.80%)	29.68% (33.80%)

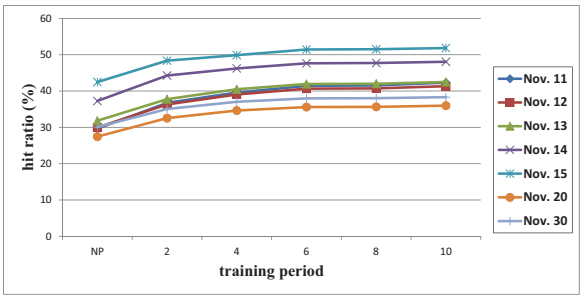

Figure 2. Test Traces: November 2010

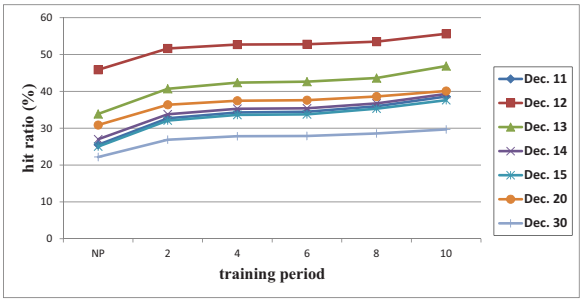

Figure 3. Test Traces: December 2010

training of 10-day trace. Figure 2 shows the results of Table II.

The third set of experiments was conducted in the same way. The testing trace was for December of 2010, and we trained our model with learning periods all starting from December 1. Testing days included December 11 to December 15, December 20, and December 30. Table III manifests the outcome. The highest improvement rate was 51.37% (testing day 12/11, 10-day training). Figure 3 plots the numbers of this experiment. The results from all three sets of experiments clearly demonstrate that our model can help the proxy server make prediction of web objects possibly accessed in the future to improve its performance. Moreover, its performance did not seriously degrade in weeks after its latest training. As we expected, our model could learn more and does better from longer training. Nonetheless, it can still do relatively well even when its training trace covers as short as two days.

Logically, even for offline learning models should get retraining once a while to update its knowledge. We were interested in finding how long the effectiveness of the training could last. For the October test case seen in Table I, except for October 11, the performance actually did not largely degrade even long after the training was done. Take the 2-day training for example. The hit ratio of October 12 (10 days after the training) was 35.95%, while the ratio was 34.09% for October 30 (28 days after the training). The numbers for the 10-day training were 40.73% (October 12) and 36.37% (October 30) respectively. Similar results were also observed in the November numbers seen in Table II. For the 2-day training, the hit ratios were 36.74% (November 11) and 35.04% (November 30). For the 10-day training, the numbers were 42.31% (November 11) and 38.31% (November 30). The December test showed the trend as well. The small performance variances among different number of days away from the last training validate our observation that users often do not alter their interests overnight.

We mentioned earlier that our model predicted at most 50 web objects for each prediction. However, this does not mean our models always predicted and prefetched 50 web objects for each prediction during our experiments. We picked nine testing days to calculate the ratio between the number of web objects sent out (including requested and predicted) and the number of web objects requested. The results are presented in Table IV. It shows that, for each web object requested, our model actually sent out between 7.91 (testing day 12/20, 2-day training) and 25.31 (testing day 12/11, 10-day training) web objects, much fewer than 50. This means that our model would consume less network bandwidth than expected in reality, which is a desirable outcome.

Table IV
THE RATIO BETWEEN THE NUMBER OF WEB OBJECTS SENT AND THE NUMBER OF WEB OBJECTS REQUESTED

testing days	2-day training	4-day training	6-day training	8-day training	10-day training
10/11	10.67	13.33	15.24	15.50	15.89
10/15	17.94	18.42	22.29	23.52	23.91
10/20	16.26	16.85	21.39	22.82	23.28
11/11	14.62	20.87	22.17	22.26	22.68
11/15	11.15	15.18	16.24	16.41	16.78
11/20	12.81	18.76	19.81	19.91	20.29
12/11	15.84	19.39	19.47	21.71	25.31
12/15	14.78	18.00	18.12	20.52	24.00
12/20	7.91	10.93	11.05	12.72	14.82

V. CONCLUSIONS

As the Internet traffic getting busier than ever before, it is imperative to use proxy servers to expedite the process of obtaining contents of required web pages for users. Researchers had proposed different ways to make proxy servers smarter in terms of making prediction for future requests of web objects that users may need. However, most schemes require online learning, which means they have to maintain and update their models with realtime calculation to keep their methods effective. Unfortunately, keeping online learning could degrade the performance of proxy servers in practice. We proposed a new model employing common interests among users to predict and prefetch web objects users may access in the future. Our model can do its learning process offline, so it does not impose heavy calculating cost on proxy servers like most previous models did. Through simulation on real-life web traces we collected at our institution, our model demonstrated that it can deliver good performance even it just had training from a trace covering only two days. Compared with the hit ratios that a regular proxy server generated, our predicting model could help the proxy server better its performance by up to 51.37%.

VI. FUTURE WORK

For our current design, the proxy server can predict and prefetch at most 50 web objects to a client for each prediction. One possible refinement we can do in the future is to dynamically adjust the number of web objects predicted each time according to the levels of traffic congestion in the network. As a result, more web objects could be predicted for users when the network is not busy, and vice versa. The training and testing traces were collected at our university. It will be great if we can conduct similar experiments on traces collected from other domains such as those from ISPs (Internet Service Providers) in the future.

REFERENCES

[1] W. Teng, C. Chang, and M. Chen, "Integrating web caching and web prefetching in client-side proxies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, pp. 444–455, 2005.

[2] D. Duchamp, "Prefetching hyperlinks," in *Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*. USENIX Association, 1999, pp. 12–12.

[3] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: potential and performance," in *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '99. ACM, 1999, pp. 178–187.

[4] T. Yeh and Y. Pan, "Improving the performance of the web proxy server through group prefetching," in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*. ACM, 2012, p. 81.

[5] K. Chinen and S. Yamaguchi, "An interactive prefetching proxy server for improvement of www latency," in *Proc. INET '97 Conference*, 1997.

[6] M. Crovella and P. Barford, "The network effects of prefetching," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, 1998, pp. 1232–1239.

[7] B. Lan, S. Bressan, B. C. Ooi, and K.-L. Tan, "Rule-assisted prefetching in web-server caching," in *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 2000, pp. 504–511.

[8] M. Deshpande and G. Karypis, "Selective markov models for predicting web page accesses," *ACM Trans. Internet Technol.*, vol. 4, pp. 163–184, May 2004.

[9] R. R. Sarukkai, "Link prediction and path analysis using markov chains," *Computer Networks*, vol. 33, no. 1, pp. 377–386, 2000.

[10] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *Computers, IEEE Transactions on*, vol. 100, no. 11, pp. 1025–1034, 1973.

[11] J. Griffioen and R. Appleton, "Reducing file system latency using a predictive approach." in *USENIX Summer*, 1994, pp. 197–207.

[12] 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 Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*. USENIX Association, 1997, pp. 13–22.

[13] A. Balamash, M. Krunz, and P. Nain, "Performance analysis of a client-side caching/prefetching system for web traffic," *Computer Networks*, vol. 51, no. 13, pp. 3673 – 3692, 2007.

[14] Q. Yang and H. H. Zhang, "Integrating web prefetching and caching using prediction models," *World Wide Web*, vol. 4, pp. 299–321, 2001.

[15] B. D. Davison, "Predicting web actions from html content," in *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*. ACM, 2002, pp. 159–168.

[16] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the world wide web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 94–107, 1999.

[17] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: design, implementation, and performance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 549–562, 1999.

[18] A. Balamash and M. Krunz, "An overview of web caching replacement algorithms," *Communications Surveys & Tutorials, IEEE*, vol. 6, no. 2, pp. 44–56, 2004.

[19] S. Jin and A. Bestavros, "Popularity-aware greedy dual-size web proxy caching algorithms," in *Distributed computing systems, 2000. Proceedings. 20th international conference on*. IEEE, 2000, pp. 254–261.

[20] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Networking.*, vol. 8, pp. 281–293, June 2000.

[21] "http://www.nielsen.com/us/en/newswire/2012/may-2012-top-u-s-web-brands-and-news-websites.html."

[22] "http://www.websiteoptimization.com/speed/tweak/average-web-page/."