Improving the Performance of the Web Proxy Server through Group Prefetching

Tsozen Yeh[*]
Department of Computer Science and Information Engineering
Fu Jen Catholic University
No.510, Zhongzheng Rd. Xinzhuang Dist.
New Taipei City 24205, Taiwan
yeh@csie.fju.edu.tw

Yenlin Pan[†]
Department of Computer Science and Information Engineering
Fu Jen Catholic University
No.510, Zhongzheng Rd. Xinzhuang Dist.
New Taipei City 24205, Taiwan
pan97@csie.fju.edu.tw

ABSTRACT

The web proxy server has widely been used to reduce the internet latency that the client perceived. With the help of the proxy server, the client may receive requested web objects from it rather than from the web server hosting them. Nevertheless, the client still needs to spend the time waiting for the web object being transferred from the proxy server when the local web browser does not have a valid copy of the requested object. This period of latency could be further reduced by predicting what web objects the client may need in the near future and then prefetching them to the cache of the local web browser. Different approaches had been proposed to help the proxy server capable of making prediction and prefetching to further reduce the Internet latency. Most of them use techniques including temporal locality, data mining, or more complicated mathematic models such as the Markov model to devise different predicting algorithms. We propose a new model, which dynamically combines temporal and spacial locality to help the proxy server make web object prediction and prefetching. Through the simulation against real traces, our model shows that with our design, collectively, the local web browser can lift its caching performance by an increase of 30% to 40%.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance Attributes; D.4.8 [**Operating Systems**]: Performance—*Modeling and Prediction*

[*]Dr. Yeh is the director of the Operating System and Storage Laboratory of the Department of Computer Science and Information Engineering.

[†]Mr. Pan is a member of the Operating System and Storage Laboratory of the Department of Computer Science and Information Engineering.

General Terms

Performance

Keywords

proxy server, operating system

1. INTRODUCTION

As the traffic of World Wide Web dramatically increases over the past few years, the demand of reducing Internet latency users experience has become an important issue. The latency mainly comes from retrieving web objects from their host web servers. Consequently, one way to mitigate the latency is to keep cacheable web objects at the local client's site for possible future reference. For those web objects dynamically created, it is not worthy to cache them since their contents vary largely each time they are requested. Often they are referred as non-cacheable objects. Modern web browsers cache web objects for future reference to save time. Our simulation results against real traces we collected reveal that, in some best cases, the web browser by itself can satisfy up to 42% of local client's cacheable requests. However, for the rest of 58%, requests of valid copies will be sent out for them.

Proxy servers can be classified into two categories, server-side and client-side, which are geographically close to the web server or the client respectively [20]. The proxy sever has been broadly used to shorten the time awaiting the requested web objects from their sources. A client-side proxy server is often designated to the same group of users within the same institution. Generally speaking, all users within the same group are configured to one or few predefined proxy servers. When the local web browser does not have a valid copy of the web object that the client requested, the request will be sent to the designated proxy server. If the proxy server has a valid copy, then the copy will be sent to the client directly. If the proxy server does not have a valid copy, either missing or stale, it will ask the source of the web object for a new copy. After receiving the new copy of the web object requested, the proxy server will send it to the client.

Regardless if the proxy server has a valid copy of the requested web object nor not, the client has to wait until the needed web object arrives. However, if we can make prediction about what the client may need shortly, and prefetch them to the cache of the web browser on the client site, then this period of waiting could be further reduced. Previous researchers had proposed various methods to make proxy server aim at this goal [3, 10, 12, 14]. The effectiveness of those methods largely depends on how accurately they can make future request prediction. The temporal locality [4, 6, 7], data mining [15], and Markov model [9, 17, 18] are techniques among those commonly adopted. Besides the predictive accuracy, the number of web objects prefetched to a client each time is a practical issue we need to consider. Ideally, it would be nice to prefetch as many web objects as possible to the client to satisfy the client's future request. However, doing excessive prefetching could likely clog the network and eventually it will prolong the client's waiting time. We propose a new technique, which dynamically combines temporal and spacial locality to make the client-side proxy server capable of making future request prediction whenever it receives a web object request from the client. For brevity, unless specified otherwise, we will use the term "proxy server" instead of "client-side proxy server" from now. Our model is based on the idea of grouping. We argue that the access to one web object is often accompanied with the access to certain other relevant web objects. By compiling web objects likely to be accessed together into groups, valid prediction and prefetching can be done accordingly. In our model, for each web object request the proxy server receives, it will examine if that object belongs to any existing groups. If so, up to a certain number of web objects in the group (or groups) will be prefetched to the client making the request. If not, then the proxy server will create a new group for this web object and no prefetching will be performed thereafter.

To evaluate our model, we collected the proxy server traces over a period of six months from our institution. Among those traces, we pick several representative sections lasting from one to three days to conduct experiments. All our experiments demonstrate the effectiveness of our model. Without prefetching, the caching done by web browser can approximately fulfill between 20% and 40% of the client's request. With the lift of prefetching in our design, the web browser can satisfy about extra 30% to 40% of requests by the client in all cases.

The remainder of this paper is organized as follows. Section 2 discusses previous works related to proxy prefetching. Section 3 describes the scheme of group prefetching. Section 4 presents the results from our experiments. Section 5 concludes this paper, and the future work is discussed in Section 6.

2. RELATED WORK

The proxy server can affect the response time that users experience. Amelioration from different aspects have been proposed to improve its performance. Web object prefetching is among one of them. The idea of prefetching is not only for the proxy server, also for some other areas. Many computer filesystems suffer the performance loss caused by the speed gap between memory and hard drive. Prefetching files from hard drive to memory before their subsequent accesses is a proven way to reduce the delay [13, 23]. The time delay between the client's sending and obtaining the requested web object is essentially similar to the issue just mentioned in the regular computer filesystems. Likewise, having the proxy server prefetch web objects possibly requested later to the client in advance could remedy the latency perceived by users [4, 6, 7, 10, 12, 14, 3, 20, 2]. Some prefetching schemes employ Markov models to make prefetching guidelines [9, 17, 18, 22]. Techniques based on data mining exploring possible user access patterns can also make prediction [15]. The hyperlinks embedded within web objects are often good candidates to get prefetched with their hosting web objects [6, 10]. Besides prefeching, smart cache replacement algorithms could help the proxy server function better by keeping more useful web objects in its cache [1, 5, 19]. Sharing contents among different proxy servers could extends their hit ratio [11]. In addition, the contents of web objects can indicate what the user may request next [8]. The proxy server needs to interact with its hard drive, reducing the I/O activity will also improve its performance [16].

3. GROUP PREFETCHING

Since the proxy server is meant for users in the same group, it accepts all requests of web objects within the same group of users. Consequently the proxy server has a better chance to identify the relationship among those requested web objects. In other words, the proxy server could observe web objects which tend to be accessed together within the same period of time and then make them belong to the same group. We can dynamically build and maintain those groups in the proxy server. Whenever a request sent to the proxy server, based on the grouping information, up to a certain number of related web objects will be prefetched to the client making that request accordingly.

3.1 Grouping Process

The grouping information updates quickly. Each time the proxy server receives a request of web object, the proxy server will check if it has a valid copy of that object. If not, the proxy server will request a new copy of that web object from its hosting web server. Otherwise, the proxy server will send the valid copy to the requesting client directly. In both cases, the corresponding grouping information of that web object will be updated as stated in section 3.2. After that, up to a certain number of web objects in the same group with the requested web object will be sent to the requesting client to complete the process of prefetching.

3.2 Grouping Scheme

Our design unites temporal and spacial localities to build and maintain grouping information. Based on temporal and spacial localities between individual web objects and existing groups, our scheme assigns individual web objects to one or few groups. If a close relationship between a web object and existing groups cannot be established, then that web object will form a new group by itself. The web objects contained in the same group are considered closely related. In other words, an access to one web object of a given group is possibly accompanied by the access to some other web objects in the same group. The way we use temporal and spacial localities to build groups is discussed as follows.

The temporal locality has been used to explore file access patterns in regular file systems. Not surprisingly, some of previous research also adopted the idea of temporal locality among sequential requests of web objects seen in the proxy server [4, 6, 7]. Our grouping scheme uses the idea of temporal locality in a different way. Instead of focusing on the instant temporal sequence among individual web objects, we maintain the temporal locality between groups and individual web objects. Each group has a timestamp and often consists of multiple web objects. The timestamp of a group, namely group time, is set to the requesting time of the web object that latest joins the group. The group time will be used as a part of the condition evaluating whether a given web object should join the group nor not.

Since the proxy server is often designated for a large number of users, it could receive many requests from different users within a short period of time. Consequently, methods relying only on using temporal locality to make prediction can easily misidentify the relevancy among unrelated web objects. To avoid this drawback, we combine the temporal locality explained above with the spacial locality discussed later to mitigate this problem. The spacial locality originally refers to the situation that if a particular memory or storage location is referenced, then its nearby locations will likely to be referenced as well. When a user is visiting a website, multiple web objects on the same website also could be requested within a short period. Early research had identified this phenomenon [21]. Some studies suggested to store web objects within the same URL path closely in the proxy server [21]. Web objects stored in the same web server often share the same URL (Uniform Resource Locator) prefix, which is somewhat similar to the concept of the spacial locality. Our scheme explores this feature when establishing groups holding closely related web objects. It is not practical to group all, even a vast portion of, web objects on a website into a single group. The path of each URL consists of multiple layers (directories) separated by the symbol "/". The length of common layers, starting from the domain name, between the URL paths of web objects could reveal the relevancy among them. The longer the common length is, the more relevant those web objects are. For example, a URL *http://a/b/c/d.html* is more relevant to a URL *http://a/b/c/e.html* than a URL *http://a/b/x/y.html*. As a result, the common length between different URL paths is used to make the overall grouping decisions as well.

Our model integrates both temporal and spacial localities to build groups in the following way. For two web objects being able to join the same group, their temporal and spacial localities will be evaluated jointly. Both the short difference between their requested time and the long common length of their URL path prefix reveal their relevant tendency. To be able to join the same group, if the difference between their requested time is longer (tending to be less relevant), then the common length of their URL path prefix also needs to be longer (tending to be more relevant). Similarly, if the difference between their requested time is shorter (tending to be more relevant), then the common length of their URL path prefix can be shorter (tending to be less relevant). Of course, two web objects can join the same group if their temporal and spacial localities both tend to be more relevant to each other.

It is common for a user to visit different parts of a website within the same period of time. Among those web objects on the same website, their URL paths could vary largely except for the domain name. Under such circumstances, if they are not requested within a relatively short period of time, then neither spacial locality nor temporal locality may be able to identify the relevancy among those web objects. To remedy this disadvantage, our scheme applies a time threshold used to ease the grouping condition. As long as the requested web objects are from the same website and the difference of their requested time is smaller than or equal to the time threshold, they can join the same group.

Collectively, our scheme uses temporal locality, spacial locality, and a time threshold to build groups of relevant web objects. Equation 1 is the formula we use to examine if two web objects are relevant enough to stay in the same group.

$$\frac{t}{T} \leq L \tag{1}$$

Whenever the proxy server receives a web object request (say for web object X), it uses the equation 1 to check if that web object X can join any existing group (or groups). The latest member of each existing group is used to compare with the web object X in both temporal and spacial localities to decide if the web object X is relevant enough to join its group. The numerator t represents the time difference between the group time and the requested time of the web object X. As stated earlier, each group uses the requested time of its latest member as its group time. We also compare the URL path of the requested web object X with the URL path of the latest member in the group to obtain the value of L, which denotes the the length of common layers (starting from the domain name) between two URL paths. The denominator T stands for the time threshold explained above. We will have more discussion on it in our experiments later.

4. PERFORMANCE EVALUATION

To justify our design, we collected real-life proxy traces from our university over a period of six months. Among those proxy traces, we chose several representative sections of data covering from one to three days. Experiments based on simulation were conducted against the selected traces. We examined the performance improvement for web browsers using our prefetching model over the cases using web browsers without our model.

4.1 Experiment Data

There are six proxy servers in the campus computer center. Two of the six are responsible for the vast majority of the network traffic. We used the traces collected between September 1, 2010 and February 28, 2011 from one of the two proxy servers. For our simulation, we prepared two sets of data. The first set consists of three one-day traces, while the second set includes three three-day traces. Table 1 lists the characteristics of data in the first set. The column *IP number* represents the number of different IPs (users) seen in corresponding traces. The column *total web objects* is the number of web objects requested, including both cacheable and non-cacheable, observed during a given date. The column *cacheable web objects* shows the number of cacheable web objects requested. Since it does not make too much sense for the web browser to keep the non-cacheable web objects, so we only utilize the cacheable web objects of the traces in our simulation. One interesting observation is that the amount of cacheable web objects occupies about one-third of the entire traces.

4.2 Experiment Design

We conducted two sets of experiments based on simulation against traces covering one and three consecutive days. The equation 1 is used for establishing and maintaining grouping information. As a reminder, to decide whether a web object

Table 1: trace data characteristics

date	IP number	total web objects	cacheable web objects
09/07/2010	1141	1839158	573854
10/08/2010	1715	3626388	1217578
11/09/2010	956	3344036	1101976

Table 2: time between consecutive requests

time (in minutes)	percentage
0 - 5	57.81%
5 - 10	12.36%
10 - 30	11.88%
30 - 60	6.12%
over 60	11.82%

X can join an existing group, we apply the equation 1 to both the latest member of that group and the web object X, if it passes, then the web object X will join that existing group as its latest member. If the web object X cannot join any existing groups after applying the equation 1 to all of them, then it will form a new group by itself.

As stated earlier, the denominator T in the equation 1 stands for the time threshold easing the grouping condition for web objects coming from the same website. As long as two web objects with the same domain name, the value of L on the right side of the equation will be equal to or greater than one. For the left side of the equation, as long as the numerator t (requested time difference between web objects) is not longer than the threshold T, the value of $\frac{t}{T}$ will be equal to or smaller than one. Hence, two web objects with the same domain name can be grouped together if the difference of their requested time is not larger than the time threshold T.

Selecting a time threshold T is an interesting question. An inappropriately small number will make web objects from the same website hard to join the same group. However, making the time threshold T excessively large will bring nearly all web objects from the same website into the same group, which is also improper. We analyzed a portion of the trace covering one full month (outside our testing periods) to see how often a given cacheable web object is repeatedly requested. Table 2 reveals the average time between consecutive requests for individual web objects. It shows that about 57.81% of them have their next request within the period of five minutes, while 12.36% of them wait five to ten minutes before they are requested again. Overall, for individual web objects, the majority of them, 70.17% (57.81% + 12.36%), will have their next requests within the period of ten minutes. As a result, we set the value of the time threshold T to five and ten minutes accordingly. This reason we use this estimation is that we view the consecutive request for the same web object as a cycle, and requests of relevant web objects from the same website would possibly show up thereinto.

Ideally, the more web objects the proxy server prefetches for the client, the more future requests the client can possibly save. However, doing prefetching overly could consume too much bandwidth and clog up the network, which will end up with hurting the overall performance of the proxy server. Consequently we need to limit the number of web objects prefetched each time to avoid this network congestion. The average size of the web objects in the six-month trace we collected is 24.4 Kbytes. To be conservative, we limit the bandwidth to 10 Mbits, which confines the number of prefetched web objects up to 52. In our experiments, we evaluate the situations with the limit of 20, 30, 40 prefetched web objects respectively. Besides, we also conducted all experiments under different time threshold T, five and ten minutes accordingly. The cache size of individual web browsers is configured to 50 Mbytes, which is a default setting for Firefox, and a common number for most popular web browsers.

4.3 Experiments for the One-Day Period

The first set of our experiments contains three one-day traces dated September 07, 2010, October 08, 2010, and November 09, 2010. Table 3 lists the results obtained for the trace of September 07, 2010. The "limit" column is the maximum number of web objects prefetched each time. So the number 20 means the proxy server will prefetch up to 20 web objects from the group (or groups) relevant (according to the equation 1) to the web object being requested. In the case where there are totally more than 20 relevant web objects existing in the relevant group (or groups), our model will pick 20 of them with the latest 20 requested time. The next column, WOR (web objects requested), denotes the total number of requested web objects that the proxy server received for the date. The third column, BH-WO-P (browser hit without prefeching), represents the number of client's requests satisfied by the web browser cache itself without our prefetching scheme. The numbers in the second and the third columns remain unchanged due to their irrelevancy to the prefeching number. The next column, BH-W-P (browser hit with prefetching), is the counterpart of BH-WO-P with our prefetching scheme. The last column counts the total number of web objects sent out by the proxy server, which includes the requested and prefetched ones.

Table 4 is derived from the table 3. The second column shows, without prefetching, the percentage of all requests that can be satisfied by the web browser. The third column reveals the performance of the web browser with our prefetching scheme. These numbers demonstrate that, without our prefetching scheme, the web browser can satisfy 34.05% of the client's request. With the help of our prefetching model, the web browser can double its performance to about 70%, which is a noticeable improvement. The last column designates, with prefetching, the number of web objects sent out by the proxy server is between 4.35 and 6.35 times as many as the number when no prefetching is applied. In other words, on average, not counting the requested one, the proxy server will prefetch between 3.35 and 5.35 web objects to the client for each request it received, which is much smaller than the number of prefetching limit (20,30, and 40). This also explains the close performance among various prefetching limits.

As stated earlier, we also conducted experiments with the time threshold of ten minutes. For brevity, we only exhibit the comparison of improvement percentage between the five-minute and the ten-minute thresholds as seen in the table 5. Compared with the numbers of five-minute threshold, the performance under ten-minute threshold ameliorates slightly, while the web object sent-out rates (WOS / WOR) are also slimly larger than its counterparts in the five-minute threshold.

Table 3: trace date: 09/07/2010 numbers, with the 5-min time threshold

WOR: web object requested
BH-WO-P: browser hit without prefetching
BH-W-P: browser hit with prefetching
WOS: web objects sent by the proxy server

limit	WOR	BH-WO-P	BH-W-P	WOS
20	573874	195376	386302	2495817
30	573874	195376	399059	3174574
40	573874	195376	399189	3644072

Table 4: trace date: 09/07/2010 improvement, with the 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	34.05%	67.32%	4.35
30	34.05%	69.54%	5.53
40	34.05%	69.56%	6.35

The table 6 and the table 7 list the result obtained for the trace of October 08, 2010. It shows that, without our prefetching scheme, the web browser can satisfy 23.61% of the client's request. With the help of our prefetching model, the web browser can increase its performance up to 62.8%. On average, excluding the requested one, the proxy server prefetched between 5.41 and 9.70 web objects to the client for each request it received. Besides, there is also no significant performance among respective prefetching limits. The table 8 compares the results with two various time thresholds for this experiment. It shows that, as seen in the table 5, both time thresholds contribute commensurate performance and cost.

The table 9 and the table 10 display the outcome for the trace dated November 09, 2010. In the case of no prefetching, the web browser can fulfill 29.05% of the client's request. When applying our prefetching method, its performance is lifted up to 64.07%. The proxy server prefetched between 5.06 and 9.35 web objects per request on average. Moreover, no obvious performance difference observed in three prefetching limits. Similarly, as seen in the previous two cases, the table 11 shows close numbers for both time thresholds as well.

Collectively, table 5, table 8, and table 11 manifest the effectiveness of our prefetching model. In general, the web

Table 6: trace date: 10/08/2010 numbers, with the 5-min threshold

limit	WOR	BH-WO-P	BH-W-P	WOS
20	1217578	287440	720530	7798744
30	1217578	287440	748553	10514703
40	1217578	287440	764612	13033405

Table 7: trace date: 10/08/2010 improvement, with the 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	23.61%	59.18%	6.41
30	23.61%	61.48%	8.64
40	23.61%	62.80%	10.70

browser could do between doubling and nearly tripling its performance when our prefetching scheme is applied. The average number of web objects prefetched remains small. In the meanwhile, there is no apparent performance difference when using different values for the time threshold in these experiments.

4.4 Experiments for the Three-Day Period

Besides the experiments conducted for one-day traces, we also want to know if our prefetching model can further better its potency as the grouping information evolves under a longer period of time. The second set of our experiments targets traces of three three-day periods starting from the same dates specified in the first set of our experiments. Table 12 reports the results obtained for the three-day period starting from September 07, 2010.

Table 13 comes of the table 12. The web browser can meet 41.83% of the client's request in the case of no prefetching. The numbers in this table certify that the web browser can grow its performance up to 72.88% with our prefetching design. Compared with table 4, a few percent improvement is obtained . We believe this is because the grouping scheme can gradually aggregate web objects more relevant to each other into the same group (or groups) as it operates for a longer period of time. Interestingly, the web browser without prefetching also did better (about six percent better) in this experiment, which is more than what our model achieves in this case. Not surprisingly, generally it is more difficult to make improvement on top of a higher ground than a lower base. Besides, the numbers of web objects prefetched are also comparable to those in table 4, which indicates the sta-

Table 5: trace date: 09/07/2010 improvement compared between 5-min and 10-min thresholds

limit	BH-W-P / WOR 5-min	BH-W-P / WOR 10-min	WOS / WOR 5-min	WOS / WOR 10-min
20	67.32%	67.45%	4.35	4.42
30	69.54%	69.73%	5.53	5.63
40	69.56%	71.01%	6.35	6.66

Table 8: trace date: 10/08/2010 improvement compared between 5-min and 10-min thresholds

limit	BH-W-P / WOR 5-min	BH-W-P / WOR 10-min	WOS / WOR 5-min	WOS / WOR 10-min
20	59.18%	59.33%	6.41	6.42
30	61.48%	61.65%	8.64	8.66
40	62.80%	62.99%	10.70	10.68

Table 9: trace date: 11/09/2010 numbers, with the 5-min threshold

limit	WOR	BH-WO-P	BH-W-P	WOS
20	1101976	320134	666884	6675222
30	1101976	320134	693381	9165394
40	1101976	320134	706053	11410201

Table 10: trace date: 11/09/2010 improvement, with the 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	29.05%	60.52%	6.06
30	29.05%	62.92%	8.32
40	29.05%	64.07%	10.35

bility of our model.

Table 14 displays the result obtained for the three-day period starting from October 08, 2010. Table 15 shows the performance comparison. Without prefetching, the web browser accomplishes 29.84% of the client's request, while its performance can reach up to 64.56% with our prefetching design. By examining table 15 and table 7, a few percent improvement is gained for the three-day experiment over the one-day experiment, which is similar to the case in the previous three-day period.

Table 16 exhibits the results for the three-day period starting from November 09, 2010. Table 17 shows the performance comparison. Without prefetching, the web browser satisfies 38.61% of the client's request, while its performance can attain up to 66.84% with our prefetching technique. Again, a few percent improvement is gained for the three-day experiment over the one-day experiment

All experiments in the second set were also conducted with ten-minute time threshold as done in the first set of experiments. The performance improvement of those with ten-minute time threshold over those with five-minute threshold considerably resembles what we observed in the first set of experiments. Therefore, we do not present the duplicated information in this section.

To understand the number of groups established and the number of their associated web objects during our experiments. We tabulate those numbers in table 18 and table 19. Table 18 lists the average number of relevant web objects aggregated per group during the three-day experiments. The leftmost column specifies the starting date. The second to the fourth columns show the accumulated numbers recorded

Table 11: trace date: 11/09/2010 improvement compared between 5-min and 10-min thresholds

limit	BH-W-P / WOR 5-min	BH-W-P / WOR 10-min	WOS / WOR 5-min	WOS / WOR 10-min
20	60.52%	61.08%	6.06	6.15
30	62.92%	63.07%	8.32	8.36
40	64.07%	64.24%	10.35	10.41

Table 12: trace date: 09/07/2010 - 09/09/2010 numbers, with the 5-min time threshold

limit	WOR	BH-WO-P	BH-W-P	WOS
20	1322072	553029	936793	5168729
30	1322072	553029	951346	6689465
40	1322072	553029	963486	8016694

Table 13: trace date: 09/07/2010 - 09/09/2010 improvement, with the 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	41.83%	70.86%	3.91
30	41.83%	71.96%	5.06
40	41.83%	72.88%	6.06

at the end of each day respectively. On average, each group will accept from a few tens to one hundred plus new relevant objects per day. One might be concerned about the growing number on each day. However, this will not be an issue since we confine the maximum number (20, 30, and 40) of web objects prefetched each time, so in practice we can limit the number of relevant web objects in each group up to the maximum prefetching number, say 20, to avoid this problem.

Table 19 counts the average number of groups established at the end of each day. One way to deal with the growing group number is to expunge those inactive groups, which can be identified by either the last time they accepted new member or how long the proxy server has not prefetched web objects associated with them. As stated above, because we can confine the number of relevant web objects in each group, so it won't take long to establish deleted groups again if necessary.

Reviewing the results collected from both sets of experiments, we could draw several conclusions. Firstly, the grouping and prefetching model we proposed indeed can lift the performance of the web browser with an increase approximately between 30% and 40%. Secondly, on average, the number of web objects prefetched by the proxy server is somewhere between three and nine for each request it receives. These numbers (three to nine) are much smaller than the maximum prefetching limit (20,30,and 40) confined in our experiments. So we know that doing prefetching this way will not greedily consume bandwidth to congest the network with prefetched web objects. Lastly, the outcome with the ten-minute time threshold mildly outperforms that of the five-minute time threshold. This suggests that our model is not very sensitive to time threshold chosen in our experimental environment, which in a way implies the stability our model would possess.

Table 14: trace date: 10/08/2010 - 10/10/2010 numbers, with the 5-min time threshold

limit	WOR	BH-WO-P	BH-W-P	WOS
20	2721002	812076	1674325	16371737
30	2721002	812076	1727824	22200204
40	2721002	812076	1756693	27199877

Table 15: trace date: 10/08/2010 - 10/10/2010 improvement, with the 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	29.84%	61.54%	6.02
30	29.84%	63.50%	8.16
40	29.84%	64.56%	10.00

Table 16: trace date: 11/09/2010 - 11/11/2010 numbers, with the 5-min time threshold

limit	WOR	BH-WO-P	BH-W-P	WOS
20	2857540	1103416	1838362	16374648
30	2857540	1103416	1884691	22450113
40	2857540	1103416	1909983	27652004

Table 17: trace date: 11/09/2010 - 11/11/2010 improvement, the with 5-min threshold

limit	BH-WO-P / WOR	BH-W-P / WOR	WOS / WOR
20	38.61%	64.33%	5.73
30	38.61%	65.97%	7.86
40	38.61%	66.84%	9.68

Table 18: web object count

starting date	end of the first day	end of the second day	end of the third day
09/07/2010	138.62	231.40	292.82
10/08/2010	229.75	400.96	535.11
11/09/2010	202.23	357.46	499.82

Table 19: group count

starting date	end of the first day	end of the second day	end of the third day
09/07/2010	6169	7653	8911
10/08/2010	9659	11108	12421
11/09/2010	9689	11100	12206

5. CONCLUSIONS

The web proxy server plays a very important role in modern network environment. Many organizations and companies set up one or few designated proxy servers as a portal to make the Internet connection. Internet traffic in both directions (in and out) needs to go through the proxy server. In other words, the proxy server monitors the obtainment and transmission of all web objects on its associated network. Consequently, how well the proxy server functions will largely affect the Internet experience for all its corresponding client computers. Whenever a user needs to access a website, if a valid copy of the demanded web object is not available locally, then the request will first go to the preassigned proxy server for help. The proxy server will deliver a copy of that web object requested if it has a valid one. Otherwise, it will acquire a valid copy from the website hosting the requested web object, and then the proxy server will forward it to the client thereafter. In the case where the proxy server can fulfill the client's request by itself, the period of time sending the web object from the web server to the proxy server can be eliminated accordingly. However, the user still has to wait for the web object he or she needs from the proxy server. Ideally, if the proxy server could appropriately predict what web objects a client may access in the near future, then those predicted web objects can be transmitted to the client in advance to further save waiting time for the client.

We propose a new model integrating temporal and spacial locality to predict and prefetch what web objects a client may access in the future. With the assistance of our technique, the proxy server could make effective predictions to conduct the prefetching. To appraise our model, we conducted experiments based on real-life proxy server traces collected from our university. Among them, we picked several representative periods lasting for one day and three consecutive days. The outcome of our experiments clearly demonstrates the effectiveness of our design. Our experiments verify that, compared with no prefetching scheme applied, the web browser can satisfy about extra 30% to 40% of web objects requested by the client in all cases when the proxy server uses our design to make prefetching. In the meanwhile, the average number of web objects prefetched by the proxy server for each request is between three and nine, which is quite distant from the maximum number allowed according to the bandwidth analysis in the section 4.2.

6. FUTURE WORK

We adopt a universal prefetching limit throughout our experiments. One possible future extension of our model is to prefetch various numbers of web objects based on different situations. For example, important users with higher priority could receive a larger number of prefetched web objects from the proxy server. The other one is to make the prefetching limit adaptive to the congestion level of the network. The more congested the network is, the smaller the prefetching limit should be, and vice versa.

7. ACKNOWLEDGMENTS

We are grateful for the help from the director, Prof. Hong-Yen Lin, and the staff of the university computer center in providing us the traces used in our experiments.

8. REFERENCES

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

[2] A. Balamash, M. Krunz, and P. Nain. Performance analysis of a client-side caching/prefetching system for web traffic. *Computer Networks*, 51(13):3673 – 3692, 2007.

[3] A. Bestavros. Using speculation to reduce server load and service time on the www. In *Proceedings of the fourth international conference on Information and knowledge management*, CIKM '95, pages 403–410. ACM, 1995.

[4] P. Cao, E. W. Felten, A. R. Karlin, and K. Li. A study of integrated prefetching and caching strategies. In *Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '95/PERFORMANCE '95, pages 188–197, 1995.

[5] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, pages 18–18. USENIX Association, 1997.

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

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

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

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

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

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

[12] 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*, SIGMETRICS '99, pages 178–187. ACM, 1999.

[13] J. Griffioen and R. Appleton. Reducing file system latency using a predictive approach. In *USENIX Summer Technical Conference*. USENIX, 1994.

[14] 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*, pages 13–22. USENIX Association, 1997.

[15] 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*, pages 504–511. ACM, 2000.

[16] C. Maltzahn, K. J., and D. Grunwald. Reducing the disk i/o of web proxy server caches. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 17–17. USENIX Association, 1999.

[17] V. Padmanabhan and J. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26:22–36, July 1996.

[18] R. R. Sarukkai. Link prediction and path analysis using markov chains. In *Ninth International World Wide Web Conference*, 2000.

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

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

[21] J. Wang, R. Min, Y. Zhu, and Y. Hu. Ucfs - a novel user-space, high performance, customized file system for web proxy servers. *IEEE Transactions on Computers*, 51:1056–1073, 2002.

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

[23] T. Yeh, D. D. E. Long, and S. A. Brandt. Performing file prediction with a program-based successor model. In *Proceedings of the Ninth International Symposium on Modeling, Analysis, and Simulation on Computer and Telecommunication Systems*, pages 193–202. IEEE, 2001.