Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 1947–1958
computer communications
www.elsevier.com/locate/comcom

Coordinated data prefetching for web contents

Xin Chen[a,*], Xiaodong Zhang[b]

[a]Ask Jeeves Co., 1551 S. Washington Ave., Suite 400, Piscataway, NJ 08854, USA
[b]Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA

Received 6 April 2005; accepted 7 April 2005
Available online 10 May 2005

Abstract

With the development of active proxy, the functions of a proxy have been enhanced beyond simply storing Web contents. Web prefetching activity in proxy is such an example to reduce client-perceived latency. In this paper, we propose a coordinated *proxy–server prefetching technique* that adaptively utilizes the access information and coordinates prefetching activities at both proxy and Web servers. In our design, the access information stored in proxies will be the main source serving data prefetching for groups of clients sharing common surfing interests. The access information in the Web server will be used to serve data prefetching only for data objects that are not qualified for proxy-based prefetching. Conducting trace-driven simulations, we show that both hit ratios and byte hit ratios contributed from coordinated proxy–server prefetching are 30–75% higher than other prefetching schemes, and they are comparable to the ratios from a proxyless server-based prefetching that is able to observe every access to the server.

Keywords: Proxy caching; Web prefetching; Pretetching thresholds; Cache hit ratios

1. Introduction

With the popularity of World Wide Web, latency perceived by the clients becomes an important factor of the quality of Web services. As an effective way to improve the Web access latency, Web prefetching attempts to preload to-be-used Web data based on the historical information of surfing activities. Combined with caching, it can provide the latency reduction as much as twice of caching alone [10]. In recent years, Web prefetching gained many attentions for mobile devices, which can effectively utilize the spared bandwidth between two continuous accesses. Till now, Web prefetching has been proposed to improve many kinds of Web services, such as static Web objects [12,13], dynamically generated Web objects [15], Web searching engines [11] and CDNs [4]. In recent years, the efforts to reduce prefetching overheads [8] and facilitate its deployment [9,17] further push Web prefetching toward a practical technique.

As an important application in active networks, active proxy has attracted a lot of attentions to extend its functions beyond temporarily storing Web contents. An active proxy may invoke a specific function (e.g. an applet in the object entity header) when serving an object. Supported by an active proxy, prefetching can be conducted between the clients and the proxy and it is recognized as one effective solution to reduce the client-perceived latency [7]. However, with more and more proxies being used by organizations and ISPs, the effects of coordination between the proxies and the Web servers should be paid a special attention in active proxy studies.

Existing prefetching models are either server-based [6] or proxy-based [7]. In a server-based environment, prefetching decisions are based on the reference access information provided by Web servers, while the decisions are based on the reference access information in proxies in a proxy-based environment. If proxies are not involved for Web accesses (we call this a proxyless environment), Web servers can provide accurate access information because a server can observe every client access. There are two limits for the server-based prefetching model. First, in a proxyless environment, frequent communication between clients and servers is required adding additional burdens to servers and networks. Second, if proxies are involved in Web accesses,

* Corresponding author. Tel.: +1 732 907 3089; fax: +1 732 907 3090.
E-mail addresses: xchen@ask.com (X. Chen), zhang@cs.wm.edu (X. Zhang).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.04.003

the accuracy of reference access information in Web servers decreases because some accesses may be directly served by proxies. Regarding proxy-based prefetching, it is attractive since it does not require additional bandwidth between the proxy and the server. However, although the burdens of servers and network can be offloaded, prediction accuracy can also be significantly limited if we only rely on this model, because proxies can only observe commonly shared data files among a limited number of clients. In summary, the access information held in proxies or Web servers is used independently of the current Web prefetching environment, and may not be able to provide accurate and cost-effective predictions.

In this study, we will first quantitatively answer two questions: (1) What are the capability limits of proxy-based prefetching? (2) What are the capability limits of server-based prefetching? Motivated by our study to address these two questions, we continue to address a third question: can we effectively integrate both server-based and proxy-based techniques to achieve comparable performance of proxyless Web server-based prefetching, and to minimize the load burden to Web servers?

In this paper, we propose a *coordinated proxy–server prefetching* technique that adaptively utilizes the access information and coordinates prefetching activities at both proxies and Web servers. In our design, the access information stored in proxies will be the main source serving data prefetching for groups of clients sharing common surfing interests. The access information in the Web server will be used to serve data prefetching only when the predictions cannot be made by proxy-based prefetching. Conducting trace-driven simulations, we show that both hit ratios and byte hit ratios contributed from coordinated proxy–server prefetching are 30–75% higher than existing prefetching schemes, and they are comparable to the ratios from the proxyless server-based prefetching.

2. Preliminaries

2.1. A PPM structure for data prefetching

In the data compression community, researchers have developed several context models to use m preceding symbols to determine the probability of next symbol. Prediction by partial match (PPM) (see e.g. [5,16]) is such a context model that has been actively used in Web predictions. Researchers have also made efforts to improve the performance of PPM trees in terms of prediction accuracy and efficiency of data structures (see e.g. [3,14]).

The PPM model structure is represented by a set of trees, each of which is rooted by the first accessed URL of a sequence of accesses. There are three variables to build a PPM prediction tree. The first one is the order of m that is the number of previous accesses used to predict future accesses. The second one is l, the number of accesses the PPM tree tries to predict. The height of a PPM tree is $m+l$. The last variable is p, the access probability that is defined as the ratio between the access frequency to an URL node and the frequency to its parent URL node in a PPM tree. We also set an access probability threshold for making prediction decisions so that only the URL nodes whose access probability is higher than the threshold are considered for prefetching. The threshold can also be a variable to adjust the prediction accuracy in both proxy and Web servers. In our study, a PPM tree is built based on $m=2$, $l=1$. The access probability threshold is set as 25% in our prefetching algorithms in most experiments.

2.2. Trace-driven simulations

Standard prefetching PPM models [5] are built in both proxies and servers. The PPM models are dynamically maintained and updated based on historical data in a time interval of 12 h. The basic prefetching operations in the simulation are as follows. While serving requests from clients/proxies, proxies/servers piggyback predicted results in responding messages. Upon receiving these results, clients/proxies conduct local searches to decide whether to further send requests to proxies/servers.

Two performance metrics are mainly used in our study. *Hit ratio* is the ratio between the number of requests that hit in browser caches and the total number of requests. *Byte hit ratio* is the ratio between the number of bytes that hit in browser caches and the total number of bytes requested. We also define relative hit ratios in order to compare the prefetching performance with the server-based prefetching scheme in a proxyless environment that is considered as the ideal reference.

We assume each client has a browser cache of 10 MB. Our sensitivity study shows that the performance in our simulated environment is independent of this size for a very large range of thresholds without observing cache replacement. The disk cache size of the proxy is 16 GB, and LRU replacement policy is used. We only prefetch objects whose data sizes are less than or equal to 50 KB, in order to reduce overhead caused by inaccurate predictions.

In the rest of this paper, we assume all prefetched objects always fresh after they are fully prefetched. It is reasonable since our simulated duration lasts only 12–24 h and only small amount of Web objects may change in such a short period. Furthermore, our study is not dependent on specific prediction algorithms. When the consistency cannot be ignored, any other algorithms, which take considerations the object update frequencies (e.g. [18]), can be used in the deployment.

2.3. Simulated global web

Our study and performance evaluation are based on trace-driven simulations. In order to evaluate prefetching techniques in an Internet environment where clients,

proxies, and Web servers co-exist, we need proxy traces to construct the proxy-based prefetching mechanism, and the traces of every Web server accessed by clients through the proxy to support the server-based prefetching. One challenge of our study is the lack of correlated server traces and proxy traces. It is not unusual that thousands of Web servers are accessed by a moderate size proxy (e.g. a proxy serving 500 clients). In practice, the available traces are either proxy-based or server-based. Individual server traces are not suitable because we are not able to distinguish clients' accesses from proxies' accesses. On the other hand, proxy traces do not include access information observed by servers. Instead of using existing server/proxy traces directly, we create a simulated global Web environment from large proxy traces, which is adequate to evaluate different kinds of prefetching techniques. The simulated global Web environment includes: (a) six scaled-down proxy traces with number of clients from 50 to 10,000, (b) six global pseudo server traces correspondent to the proxy traces, and (c) six individual pseudo Web server traces. For more detailed information about the simulation environment, see Appendix A.

3. Investigation of current prefetching methods

3.1. Limits of proxy-based prefetching

A relative hit ratio for a given proxy is defined as the ratio between a hit ratio obtained by proxy-based prefetching and a hit ratio obtained by proxyless server-based prefetching. The relative hit ratio presents the relative prefetching ability of proxy-based prefetching compared with proxyless server-based prefetching.

Making prefetching predictions at the proxy for the six individual pseudo servers, we have observed the changes of relative hit ratios by increasing the number of clients to access the proxy (see Fig. 1(a)). Randomly selecting different groups of clients, we repeated each experiment with an individual Web server 20 times, and presented the average results. This study indicates that proxy-based prefetching ability, measured by the relative hit ratios, increases as the number of clients served by the proxy for commonly accessing a particular server increases. For example, as the number of clients accessing server 1 increases to 16 and 64, the relative hit ratios for the proxy increase to 60 and 75%, respectively.

Our trace-driven simulation results show that the predictions made by the proxy-based prefetching scheme is reasonably reliable only when the number of clients accessing the same Web server is larger than 16. The next quantity we want to determine is that what is the percentage of total accesses shared by less than or equal to 16 clients going to the same server.

The distribution curves in Fig. 1(b) are presented by the Cumulative Density Functions (CDF) for the six proxy traces with 50, 200, 500, 1000, 5000, and 10,000 clients, respectively. Each point in a proxy CDF curve represents the percentage of client requests (marked in the vertical axis) going to the same Web server with the equal or a less number of shared clients through the proxy (marked in the horizontal axis). For example, for the 50 client proxies (proxy 1), 97% of the requests to a particular Web server comes from 16 or fewer clients. On the other hand, for the 10,000 client proxy (proxy 6), 44% of the requests to a particular Web server comes from 16 or fewer clients.

In practice, the number of clients served by proxies 1 or 6 is either too small or too large. Thus, we use the 1000 client proxy (proxy 4) as a regular case, and find that more than 62% of the requests to a particular Web server come from 16 or fewer clients. This example shows that a proxy-based prefetching can reasonably satisfy less than 40% of the requests by achieving approximately a 60% relative hit ratio. However, for more than 60% of the requests, proxy-based prefetching may not be sufficiently effective. In order to improve the access hit ratios, we should adaptively rely on server-based prefetching for those requests that do not have a sufficient number of shared clients.

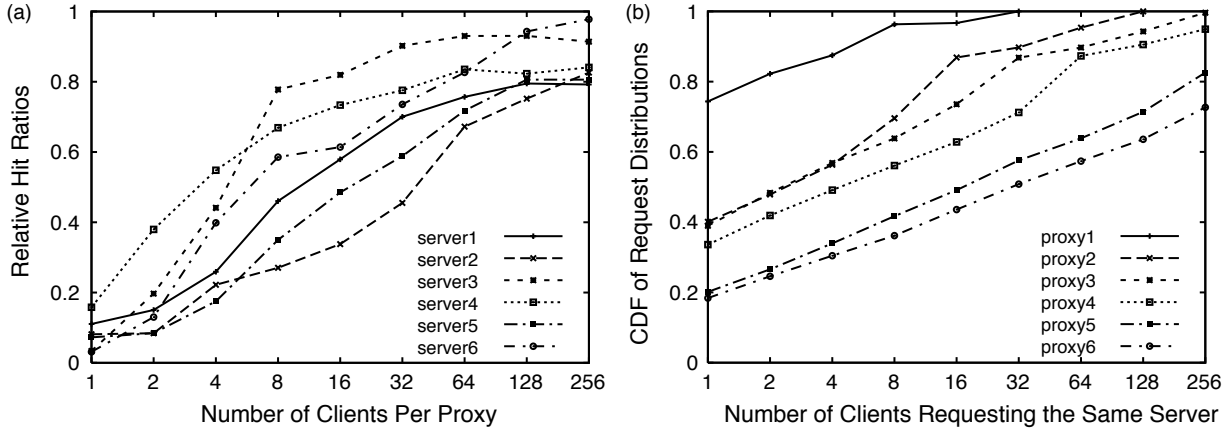

Fig. 1. (a) Relative hit ratios for accessing each of six servers through the proxy where the prefetching is made in the proxy. (b) Cumulative density functions (CDF) for the six proxy traces with 50 (proxy 1), 200 (proxy 2), 500 (proxy 3), 1000 (proxy 4), 5000 (proxy 5), and 10,000 (proxy 6) clients, respectively.

In summary, the purpose of this part of the study is to show two Web access effects. First, as the number of clients targeting the same server going through a proxy increases, the client hit ratios increases proportionally with a support of the proxy-based prefetching. The effectiveness of proxy-based prefetching is based on this. Second, although increasing the number of clients going through a proxy can improve the effectiveness of proxy-based prefetching, the percentage of requests to the same server shared by a certain number of clients (e.g. larger than 16) is moderate. Thus, proxy-based prefetching may not serve well to other clients with a smaller number of sharing partners.

3.2. Limits of server-based prefetching

With an increasing number of proxies being installed in the Internet to provide caching storage for clients, the burden of a huge number of direct requests to Web servers can be lightened. On the other hand, the accesses directly observed by the Web servers also decrease, which will inevitably weaken server-based prefetching capability, because this prefetching mainly relies on the access information obtained in the Web servers. A practical server-based prefetching can only rely on the access information practically available in the server, which is based on the accesses not satisfied by proxies. This can be simulated by building proxies between clients and servers. When clients request documents that are not cached in the proxies, the requests could be received and used for predictions by the server.

We also use a relative hit ratio to quantify the prefetching capability, which is defined as the ratio between the hit ratio obtained by a practical server-based prefetching and the hit ratio obtained by the proxyless server-based prefetching. The six individual pseudo Web server traces are used in our experiments. In order to build a predictor tree in a Web server, we limited the number of clients connected to the proxies to 256 clients, and changed the number of clients served per proxy during the experiments.

Fig. 2(a) presents the changes of relative hit ratios for the six servers as the number of clients per proxy increases. Our trace-driven simulation results show that server-based prefetching capability is weakened significantly as the number of clients per proxy increases. For example, as the number of clients per proxy increases from 1 to 16, and to 64, the relative hit ratios for server 1 decrease from 100 to 70%, and to 40%, respectively.

We have also measured the changes of the number of requests observed by Web servers as the number of clients per proxy increases. This evaluation quantitatively shows the trade-off between lightening the server burden and weakening the server-based prefetching capability. Fig. 2(b) presents the ratios of the numbers of requests to a server between practical server-based prefetching and proxyless server-based prefetching for the six servers as the number of clients per proxy increases.

By comparing the two figures in Fig. 2, it is interesting to observe that the ability of server-based prefetching measured by the relative hit ratio decreases faster than the decreasing pace of the number of requests received by the servers. For example, when the number of clients is 16, the ability of prefetching is about 65% while the number of requests is about 75%. Since the server could not observe all clients' requests due to proxy caching, successive requests from the same client received by the server may not be continuous. However, in order to increase accuracy, prediction algorithms are usually based on several past accesses. (For example, in our experiments, we use two previous accesses to make predictions for next accesses.)

3.3. Threshold sensitivity

3.3.1. Threshold vs. hit ratio

The access possibility threshold for prefetching of 0.25 is used in the previous experiments. We have observed the hit ratio changes for server-based and proxy-based prefetching

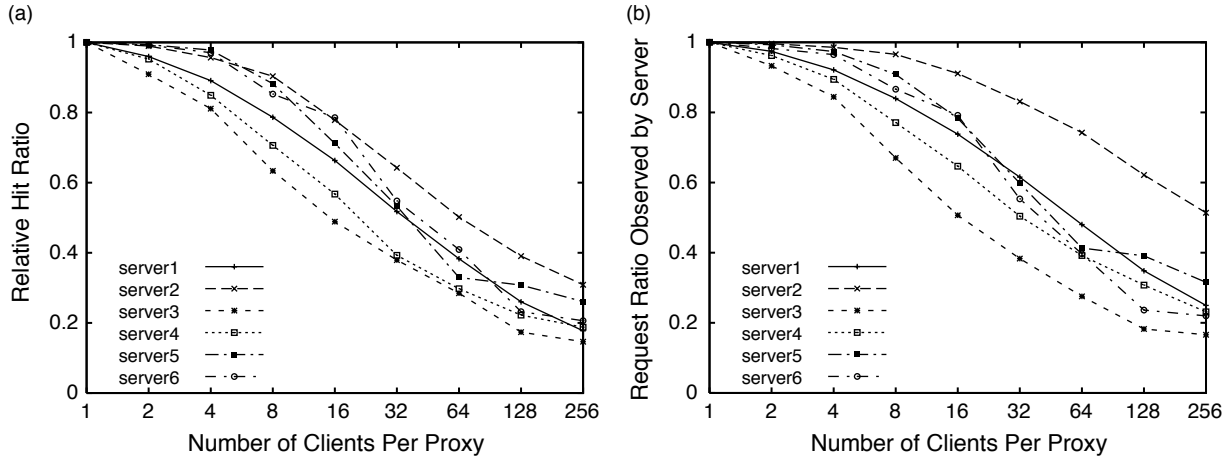

Fig. 2. (a) Relative hit ratios for accessing each of six servers through the proxy where the prefetching is made in the server. (b) Percentage of requests observed by the server through different size proxies.

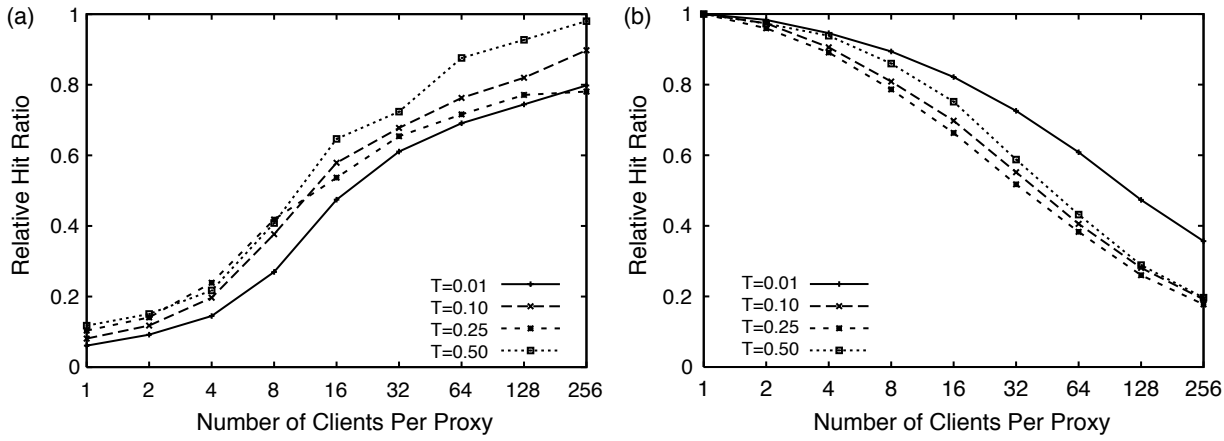

Fig. 3. (a) The relative hit ratios of proxy-based prefetching for different thresholds when various numbers of clients accessing server 1 through a proxy. (b) The relative hit ratios of server-based prefetching for different thresholds when various numbers of clients accessing server 1 through a proxy.

techniques by increasing the access probability threshold from 0.01 to 0.5. A lower threshold will make the system prefetch more files and increase the hit ratios. We found that the hit ratios increase 30% when we decrease the threshold value from 0.5 to 0.01. However, the prefetching ability, measured by the relative hit ratios, is varied in a similar trend with the number of clients per proxy, no matter which threshold is used. This statement is validated by all individual pseudo server traces. As an example, we use server 1 in our presentation. Four typical threshold values 0.01, 0.10, 0.25 and 0.50 are selected to show the changes of relative hit ratios when the threshold varies in the range between 0.01 and 0.50.

Fig. 3(a) displays the relative hit ratios of proxy-based prefetching scheme. Given the number of clients per proxy, the differences among all relative hit ratios are within 10% except the threshold 0.50, whose value is 20% higher than that of threshold 0.01 when the number of clients is larger than 32. When we set a threshold as high as 0.50, the prefetched objects are mainly most popular ones for all clients. From the PPM structure, the proxy can correctly identify those objects by accumulating the access history of a certain number of clients. Thus, in most cases, the proxy-based prefetching can achieve slightly better performance when a higher threshold is set.

The relative hit ratios of server-based prefetching schemes are presented by Fig. 3(b). Similarly, for any number of clients per proxy, the relative hit ratios for all thresholds are close (within 5% in most cases) except the threshold 0.01, whose value is always higher than any other thresholds. When a low threshold is used, a large amount of objects are selected for prefetching. Since many of them may be selected several times, they still can be preloaded if a prefetching procedure is not initiated when the client access is satisfied by the proxy. For this reason, a lower threshold can achieve slightly higher performance.

3.3.2. Threshold vs. prediction accuracy

In a proxyless environment, the prediction accuracy is determined by the selected threshold. When a proxy is used, the prediction accuracies of both proxy-based and server-based prefetching are affected by the number of clients accessing a server through the proxy. Similar to relative hit ratios, we define relative accuracies in order to compare the proxy/server-based prefetching with the proxyless server-based prefetching scheme. In the same sets of experiments in the previous section, the relative accuracies are measured with the change of number of clients accessing a server through a proxy. Fig. 4(a) displays the relative accuracies of proxy-based scheme. Given the number of clients per proxy, the differences among all relative accuracies are within 10% except the threshold 0.01, whose value is always above 1. When both the number of clients and the threshold are small, in the proxy-based prefetching, the number of prefetched objects at each access is far below the proxyless server-based scheme. However, the most popular objects are likely prefetched even with a small amount of access information accumulated by the proxy, which makes the proxy-based prefetching get higher accuracy than the proxyless server-based scheme. On the contrary, when the threshold is large, in the proxy-based prefetching, the number of prefetched objects at each access is likely more than the proxyless server-based scheme, since any infrequent access path has high access probability with a small amount of access information accumulated by the proxy. For these thresholds, the relative accuracies are always less than 1, which makes the proxy-based prefetching require more traffic consumption than proxyless server-based schemes for the same hit ratios.

The relative accuracies of server-based prefetching schemes are presented by Fig. 4(b). Different from proxy-based schemes, given any number of clients, the relative accuracies of all thresholds are within 5% from 1. With more number of clients per proxy, the prediction for the next

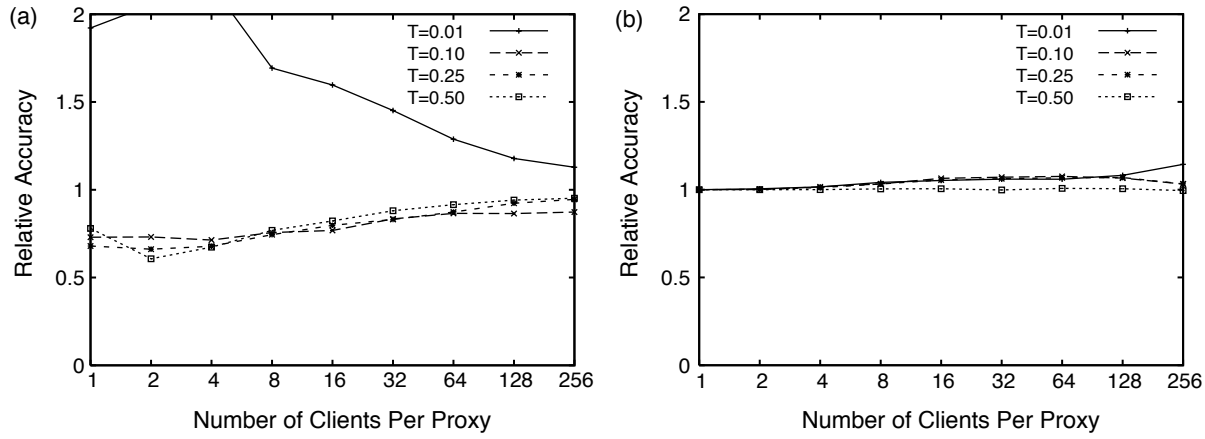

Fig. 4. (a) The relative accuracies of proxy-based prefetching for different thresholds when various numbers of clients accessing server 1 through a proxy. (b) The relative accuracies of server-based prefetching for different thresholds when various numbers of clients accessing server 1 through a proxy.

accesses becomes less accurate due to the change of access patterns by using proxy caching. However, the hit ratio of each prefetched object is increased due to more clients sharing the cache. Thus, when no consistency issues are considered, in a long term, the server-based prefetching scheme is not affected by the number of clients behind a proxy.

For both proxy-based and server-based prefetching schemes, the hit ratio is increased with the decrease of the threshold while the increase of file allocations and network traffic are also significant. This should not be a major concern because disk space is becoming very cheap, and increasingly more network bandwidth will be available. A recent study [18] argues that aggressive prefetching should be conducted. This can be achieved by significantly lowering the access probability threshold in the PPM model.

Since the comparative performance in our study is not sensitive to the threshold, we choose a moderately high threshold value of 0.25 in all experiments in the rest of the paper. Since the only increased traffic is between the clients and the proxy, the communication overhead is ignored in our performance evaluation.

4. Coordinated proxy–server prefetching

One effective approach to overcome the limits of both server-based and proxy-based prefetching techniques is to develop a coordinated proxy–server prefetching technique. Since proxy servers may be located between servers and clients, Web server traces may not exactly reflect client access behavior. On the other hand, the proxy servers have limited prefetching capacities because of the diverse interests of the connected clients. A good coordinated prefetching technique should effectively address these concerns.

4.1. Prefetching without coordination

A simple way to overcome the limits of proxy-based and server-based prefetching is to let proxy and server make predictions independently. In this method, the proxy makes predictions for the requests from clients and the server makes predictions for those requests forwarded by the proxy. Since there is no cooperation information needed for Web prefetching between the proxy and the server, we call this method non-coordinated proxy–server prefetching technique. We tested the performance of this scheme by using the six individual server traces.

Now we define a relative hit ratio as the ratio between a hit ratio obtained by the non-coordinated proxy–server prefetching and the proxyless server-based prefetching schemes. The results are shown by Fig. 5(a). The relative hit ratio is decreased as the number of clients per proxy increases first, but it becomes stable when the number of clients is large enough. We observed that when there are more than four clients per proxy, the relative hit ratio varies in the range between 60 and 90%, with an average of 80%, which indicates the low performance of the non-coordinated proxy–server prefetching.

Because the predicted documents are selected by predetermined thresholds, not all predictions would have results. We define a useful prediction as one giving prediction results. We further define a relative prediction ratio as the ratio between the number of useful predictions of non-coordinated proxy–server prefetching and the proxyless server-based prefetching schemes. When counting the useful predictions of the non-coordinated scheme, we include the predictions made by both proxies and servers. Fig. 5(b) shows the relative prediction ratios for the same set of experiments. Compared with Fig. 5(a), the relative prediction ratios are approximately proportional to the relative hit ratios. When the number of clients changes from 1 to 4, the relative hit ratio (and the useful prediction rate) is decreased fast, then increased, and become stable as the number of

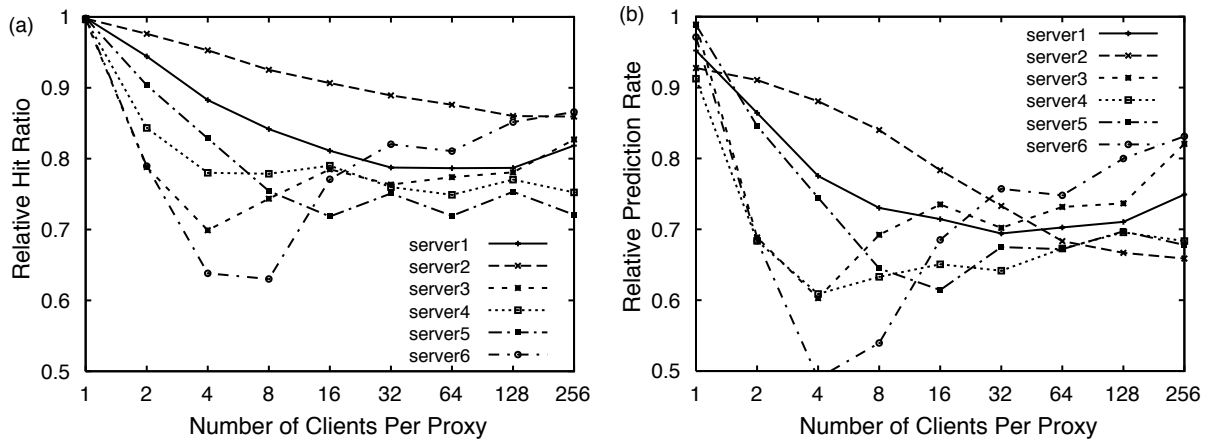

Fig. 5. (a) Relative hit ratios of the non-coordinated proxy–server prefetching scheme for six servers. (b) Relative prediction ratios of the non-coordinated proxy–server scheme.

clients continues to increase. This change can be explained by the different effects of caching and prefetching with the increase of number of clients. If only a few clients exist, prefetching procedure may not make predictions effectively because its threshold filters all objects with access probabilities below it. It is especially obvious when the number of clients is less than 1/threshold. In Fig. 5, all relative hit ratios and relative prediction rates are decreased very fast when the number of clients is increased to four. On the contrary, when few clients use the proxy, the hit ratio on the proxy is increased very fast with the number of clients increased. Because hits on a proxy stop a server to make predictions even when the prediction cannot be made by the proxy, the fast-increased hit ratio and slow-increased prediction ability exacerbate the performance of the non-coordinated prefetching scheme.

4.2. Algorithm of coordinated prefetching

In order to overcome the limits of both server-based and proxy-based prefetching techniques and further improve the performance, we propose a coordinated proxy–server prefetching technique that coordinates a proxy and a server by exchanging prefetching statuses with each other, and by distinguishing the prefetching duty of each. Specifically, prefetching in proxy is mainly used for serving groups of clients sharing common Web pages, while prefetching in Web servers is for the rest of individual clients.

In our algorithm, we still include the input from the clients even when a hit happens in the browser cache. Upon a client request of an object file, if the request hits in the browser, the object will be accessed locally. The client will also inform the proxy about the access to the object file, and initiate the coordinated proxy–server prefetching process. If the request misses in the browser, the request will be forwarded to the proxy, and the coordinated proxy–server prefetching process is initiated. Starting in the proxy, the coordinated proxy–server prefetching process will handle the following four different cases in the proxy:

1. the object *exists* and a prediction *can* be made in the proxy;
2. the object *does not exist* and a prediction *can* be made in the proxy;
3. the object *exists* and a prediction *cannot* be made in the proxy; and
4. the object *does not exist* and a prediction *cannot* be made in the proxy.

A proxy-based prefetching procedure will handle case 1, while a server-based prefetching procedure will handle case 2. We propose the proxy–server-based prefetching procedures (I) and (II), defined below, to handle cases 3 and 4, respectively. In both cases, when no prediction results are given by a proxy (no prefix matched or no access above the threshold), the server will be requested to make predictions. The four procedures involved in the coordinated proxy–server prefetching process are shown in Fig. 6.

Proxy-based prefetching procedure:

(a) The proxy sends the requested object to the client, along with a list of URLs of predicted objects.
(b) After a local searching, the client sends a selected list of URLs of predicted objects to the proxy.
(c) The proxy sends the selected objects to the client.

Server-based prefetching procedure:

(a) The proxy forwards the client request to the server, and asks the server to make predictions based on its PPM model.
(b) The server sends the requested object to the proxy along with a list of URLs of predicted objects.
(c) The proxy stores the object, and sends it to the client along with the list of URLs of predicted objects.
(d) After a local searching, the proxy sends a selected list of URLs of predicted objects to the server.

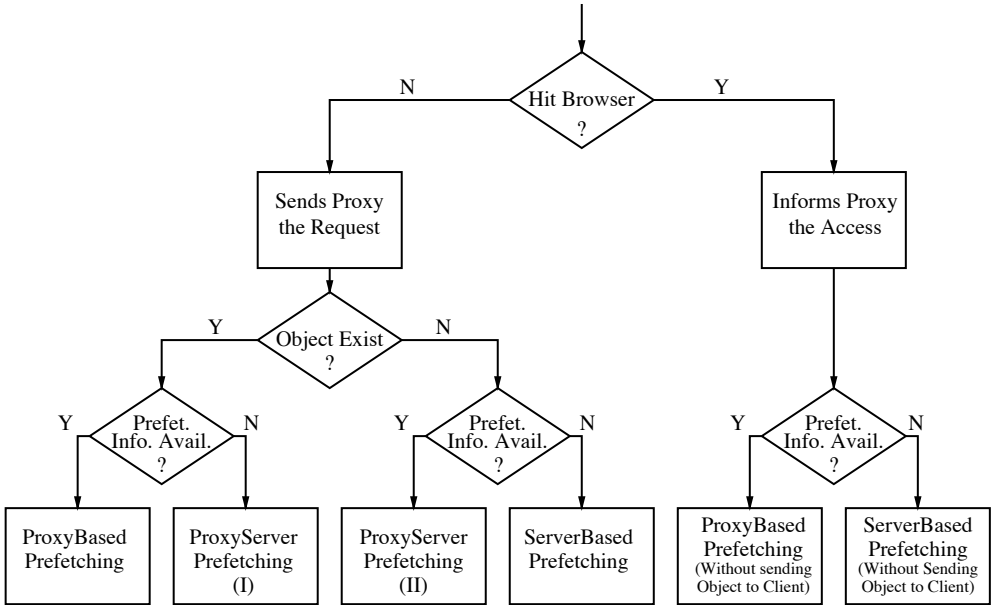

Fig. 6. The coordinated proxy–server prefetching system design.

(e) The server sends the selected objects to the proxy. Meanwhile, the client sends a selected list of URLs of predicted objects to the proxy after a local searching.
(f) The proxy sends the selected objects to the client.

Proxy–server-based prefetching procedure (I):

(a) The proxy sends the object to the client, and asks the server to make predictions based on its PPM model.
(b) The server sends a list of URLs of predicted objects.
(c) The proxy sends the list of URLs of predicted objects to the client.
(d) After a local searching, the proxy sends a selected list of URLs of predicted objects to the server.
(e) The server sends the selected objects to the proxy. Meanwhile, the client sends a selected list of URLs of predicted objects to the proxy after a local searching.
(f) The proxy sends the selected objects to the client.

Proxy–server-based prefetching procedure (II):

(a) The proxy forwards the client request to the server, along with a selected list of URLs of predicted objects based on the local prefetching information and a local search.
(b) The proxy sends the selected list of URLs of predicted objects to the client.
(c) The server sends the selected objects to the proxy. Meanwhile, the client sends a selected list of URLs of predicted objects to the proxy after a local searching.
(d) The proxy sends the selected objects to the client.

5. Performance evaluation

We use the simulated global Web environment to evaluate our proposed and existing prefetching schemes, which is constructed in Section 3. In this environment, hundreds of clients access thousands of Web servers through a proxy simultaneously, while the Web servers are also accessed by a huge number of clients directly. It provides an adequate test bed for comprehensive evaluations of Web prefetching techniques.

5.1. Simulation parameters

The basic simulation parameters are configured as the same as those in previous experiments. Additionally, in order to count the network latency in our experiments, the following network parameters are set in the simulation:

- A Local Area Network (LAN) is constructed between clients and the proxy server. It is a high bandwidth and low latency network with a round trip time, RTT = 1 ms.
- A wide area network (WAN) is constructed between the proxy server and Web servers. It is a high bandwidth and high latency network with a round trip time RTT = 90 ms.

In reality, the downloading time for a prefetched file is dependent on the available bandwidth and proxy or Web server processing speed, which are dynamically changed. In our simulation, we estimate the downloading time from the value of the field elapsed time in the proxy traces. This field records how many milliseconds between the time when a request is accepted and the time when the requested file

is fully transferred to a client by the proxy, including the server processing time, the transmission time between the proxy and the server, and the transmission time between the proxy and the client. The study in [10] shows that the external latency (between a proxy and a Web server) accounts for more than 80% of the response time. Based on this finding, we set the internal downloading time (between a client and a proxy) in our simulation to be 1/6 of the elapsed time obtained from the proxy trace.

Persistent connections and pipelining requests are also used in our experiments. We assume that only one persistent connection exists between a pair of a client/proxy and a proxy/server. When multiple requests are issued for predicted documents, they are sent to the server without waiting for the previous request to finish. Using default values of Apache HTTP server version 2.0, we set a fixed holding time of 15 s for each persistent connection, and the maximum number of requests of 100 per connection.

The CPU overhead is measured by the total number of prefetching decisions made in proxy and Web servers. The CPU cycles used for each decision are not counted in the simulation for two reasons. First, CPU cycles are becoming very cheap, and the prediction operations updating the PPM trees are not very computing intensive. Second, the predictions are normally conducted when proxy and Web servers are lightly loaded. In addition, according to the implementation experiences in [7], the PPM model consumes a very small amount of CPU operations.

5.2. Comparisons of hit and byte hit ratios

Fig. 7 presents hit ratio comparisons among proxy-based, server-based, non-coordinated proxy–server, coordinated proxy–server, and proxyless server-based prefetching techniques using the six proxy traces and their corresponding pseudo global server traces. Contributions to the hit ratios come from three sources: hits on cached files of a browser (simplified as browser hits), hits on fully prefetched files of a browser (simplified as f-prefetched hits) and hits on partially prefetched files of a browser (simplified as p-prefetched hits). The p-prefetched hits occur during the prefetching while the prefetched files are being transferred to a browser. Fig. 7(a) shows that the hit ratios of the coordinated prefetching are comparable to the hit ratios of the proxyless server-based prefetching, and are significantly higher than those of other prefetching techniques for all the six proxy traces. For example, the hit ratio of the coordinated proxy–server prefetching on proxy 4 with 1000 clients is 66.5, 84% higher than the proxy-based prefetching (36.1%), 46% higher than the practical server-based prefetching (45.5%), 21% higher than the non-coordinated proxy–server prefetching (54.7%), and almost the same as the proxyless server-based prefetching (67.5%).

Fig. 7(b) presents byte hit ratio comparisons, which are consistent with the hit ratios. We have observed that 1/3–1/2 of the prefetching hits are partial hits. This is because the time between a request to an HTML file and to its embedded image files is very short so that embedded image files could not be fully prefetched.

5.3. Reductions of web server loads

Using the same trace-driven simulation, we have compared the numbers of packets received by Web servers, and the numbers of prefetching decisions made by Web servers for all the five techniques (the proxy-based scheme is not involved in the latter comparison) in Fig. 8. The trace-driven simulations show that as the number of clients served by the proxy increases, the difference of the numbers of packets sent to Web servers between the coordinated prefetching technique and the proxyless server-based technique increases. For example, the packet differences are 434,393 representing a 48% reduction from the proxyless server-based prefetching, and 766,985

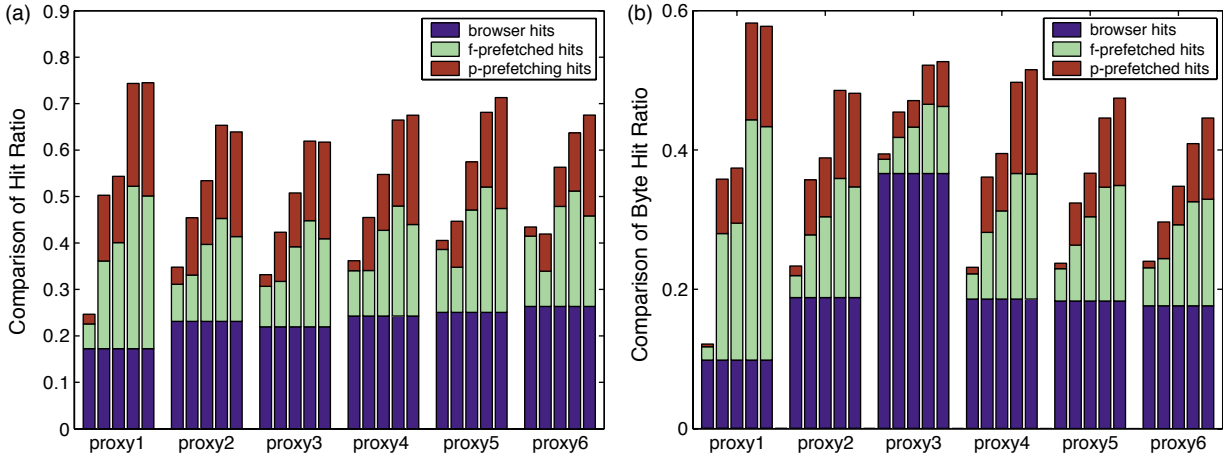

Fig. 7. (a) The comparisons of hit ratios among the proxy-based, server-based, non-coordinated proxy–server, coordinated proxy–server, and proxyless server-based prefetching techniques ordered from left to right in each proxy group. (b) The comparisons of byte hit ratios among the same five prefetching schemes in the same order from left to right in each proxy group.

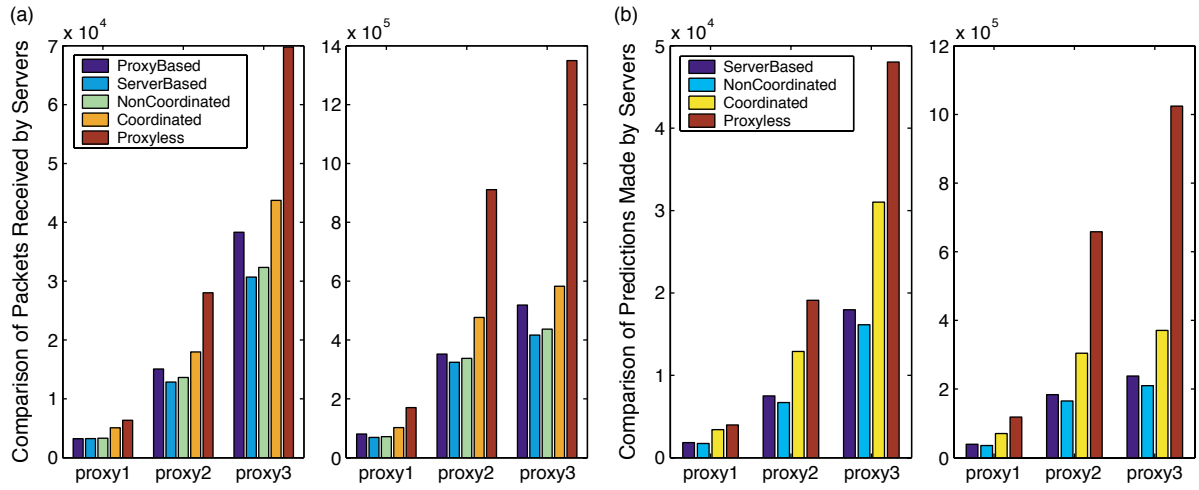

Fig. 8. (a) The packet reduction rates of the five possible prefetching schemes. (b) The comparison of predictions made on servers of the four prefetching schemes.

representing a 57% reduction from the proxyless server-based prefetching for proxies 5 and 6, respectively. This shows that the coordinated prefetching technique can effectively reduce the Web server load and network traffic.

The comparisons of the number of prefetching decisions made by servers follow similar patterns. For example, the number of prefetching decision differences is 353,840, representing a 54% reduction from the proxyless server-based prefetching, and 653,445, representing a 64% reduction from the proxyless server-based prefetching for proxies 5 and 6, respectively.

5.4. Implementation discussion

The proposed coordinated prefetching scheme can be easily implemented on existing and standard browsers, proxies and Web servers. For better performance, two kinds of requests are distinguished from each other by adding a new entry in the HTTP request header: regular and prefetch. The regular requests are explicitly requested by clients while the prefetch requests are implicitly sent by the browsers/proxies based on the prediction results from the proxies/servers.

Any requests without the entry in their headers will be considered as regular requests for the compatibility with existing deployment. Considering their different effects on the client-perceived latency, regular requests have higher priority than prefetch requests when they co-exist on the proxies/servers.

In our simulation experiments, we assume the browsers/proxies inform the proxies/servers when hits happen. In practice, it may introduce lots of messages and increase the overhead on proxies/servers to handle the incoming messages. However, if no predictions are made in these cases, the prefetching effectiveness may be largely reduced. In the implementation, the server/proxy makes predictions for all incoming requests. When browsers/proxies receive the prediction results with prefetched documents, they will cache them as well. When those prefetched documents are requested, the browser/proxy can use the cached prediction results to generate prefetch requests. Although it is not always accurate when more than one previous URL is used, our experiments demonstrate the hit ratio is only 5% lower while up to 40% messages are reduced, compared with the method using an additional message for a hit on cache.

In the implementation, the prefetching overhead can be further reduced by caching the prediction results of Web servers in the proxy. When the prediction results are returned by a server, the proxy records the previous access sequences and their prediction results. Before the proxy sends prediction requests to a server, it checks if the prediction results are cached. A cached result can be returned to the clients immediately without the involvement of the server. When a server has no prediction results, the proxy may also cache it for future references, which is called negative caching. Negative caching can eliminate the unnecessary prefetching overhead on the server. Our experiments show that caching prediction results on the proxy can reduce around 20% Web server prefetching overhead.

6. Conclusion

In this study, we have investigated the issues of coordinations between proxy-based prefetching and server-based prefetching schemes. We show that the access information of both proxies and servers is important to data prefetching, but should be utilized effectively. Our study shows that we can effectively integrate and coordinate both server-based and proxy-based techniques to achieve comparable performance of the proxyless server-based prefetching and to minimize the communication between proxies and Web servers.

Acknowledgements

We thank the anonymous referees for their constructive comments. We appreciate Bill Bynum for reading the paper and his comments. This work is supported in part by the National Science Foundation under grants CNS-0098055, CCF-0129883, and CNS-0405909.

Appendix A. Simulated global web

A.1. Web access traces

The Boeing Company has six proxy servers in its Puget Sound firewall. The access traces of these proxies consist of collected logs by using an anonymizer tool (log 2anon), which are available in [2]. The first five proxies are in a DNS round robin configuration for load balancing. The sixth is running a newer version of software than the others, and is used as a parent for internal proxy cache testing. Since we are interested in tracing accesses of each individual client, we have used the first five traces in our experiments. After sorting the five traces in time order, we have merged five traces into a large file for each day. The traces of March 4, 1999 are used for presentation in this paper, which include 22,187,580 requests from 66,358 clients to 142,338 servers.

In order to effectively evaluate prefetching techniques in a reasonably practical environment, we have created a pseudo Web access environment by using the given merged trace. In this environment, different sized groups of clients access a set of Web servers through a proxy (represented by *scaled-down proxy traces*), the same set of Web servers are also directly accessed by many other clients without going through proxies (represented by global pseudo server traces), and a set of popular Web servers are frequently accessed by many individual clients (represented by individual pseudo server traces).

There may be a limit for these created traces. Since all the clients are from the Boeing Company, common interests to certain Web servers are shared among them, which can favor both server- and proxy-based predictions. However, we believe this limit is minor, and should not affect our performance results because the number of clients (66,358) is huge, and access interests are sufficiently diverse.

A.2. Scaled-down proxy traces

Using the large merged trace file, we have created six proxy traces with different numbers of clients accessing the proxy, which are called scaled-down proxy traces. The numbers of clients we used are 50, 200, 500, 1000, 5000, and 10,000. The clients are randomly selected in the Boeing proxy traces for each scaled-down proxy trace. Since each scaled-down proxy trace is a portion of the original Boeing proxy traces, the trace contains realistic proxy access reference information.

Table A1 presents the characteristics of the six scaled-down proxy traces collected from the Boeing proxy traces.

A.3. Global pseudo server traces

While a scaled-down proxy trace contains access references to a number of Web servers, these servers are also accessed by many other clients who are not selected to form this scaled-down proxy trace. We collect the trace references to the same servers from the other clients to form a global pseudo server trace of the scaled-down proxy. Table A2 presents the characteristics of the six global pseudo servers.

Table A1
Selected scaled-down proxy traces

	Proxy 1	Proxy 2	Proxy 3	Proxy 4	Proxy 5	Proxy 6
Total clients	50	200	500	1000	5000	10,000
Total requests	13,566	68,985	154,527	341,619	1,859,045	3,068,912
Successful requests	12,275	60,613	133,517	302,856	1,607,145	2,653,288
Transferred Mbytes	123	360	1559	2080	13,353	20,745
Number of servers	647	1692	3482	7004	24,486	34,179
Number of files	9484	32,134	74,958	157,717	679,876	995,879

Table A2
Global pseudo server traces

	Global server 1	Global server 2	Global server 3	Global server 4	Global server 5	Global server 6
Total requests	7,951,627	11,270,833	13,462,240	14,865,900	18,126,906	18,902,854
Successful requests	6,850,784	9,866,199	11,718,589	12,897,176	15,689,774	16,414,926
Transferred Mbytes	38,886	59,204	76,752	82,450	114,371	119,795
Total clients	63,054	64,769	65,182	65,517	65,987	66,068
Number of files	1,505,796	1,969,177	2,464,367	2,794,933	3,818,797	4,136,209
Number of servers	647	1692	3482	7004	24,486	34,179

Table A3
Selected individual pseudo server traces

	Server 1	Server 2	Server 3	Server 4	Server 5	Server 6
Total requests	648,534	243,462	210,131	182,541	157,466	140,473
Successful requests	647,277	233,216	208,059	181,531	133,854	139,459
Transferred MBytes	1815	1204	1112	875	542	555
Total clients	21,769	15,193	3086	2149	1810	3582
Number of files	5810	4185	3003	3637	5130	1549
Distinct request ratio (%)	0.28	0.40	0.59	0.86	1.75	0.43
Distinct file access ratio (%)	30.98	22.32	41.00	43.11	45.79	39.12
Concentration ratio (%)	93.79	87.40	93.22	92.43	87.74	90.94

Using a scaled-down proxy trace and its corresponding global pseudo server trace, we can construct a Web access environment where some clients access a group of servers through a proxy (based on scaled-down proxy traces) and some clients access the same group of servers without going through the proxy (based on global pseudo server traces).

A.4. Individual pseudo server traces

In the merged Boeing proxy traces, we identified several popular servers that are frequently accessed by clients to form individual pseudo server traces. Such an individual pseudo server trace is created by collecting all access references to the identified server. We use three criteria to determine whether such a trace is sufficiently realistic to represent a server trace: (1) distinct request ratio (the percentage of distinct requests in the total number of requests), (2) distinct file access ratio (the percentage of distinct files in the total number of distinct server files), and (3) concentration ratio of references (the percentage of requests concentrating on 10% of popular files). In practice, a Web server's distinct request ratio, distinct file access ratio, and concentration ratio of references are below 3%, around 30%, and around 90%, respectively [1].

For our trace-driven simulation, we selected six of them, which are presented in Table A3.

References

[1] M. Arlitt, C.L. Williamson, Web server workload characterization, the search for invariants, Proceedings of ACM SIGMETRICS Conference, Philadelphia, PA, May, 1996.
[2] Boeing log files, ftp://researchsmp2.cc.vt.edu/pub/boeing/
[3] X. Chen, X. Zhang, A popularity-based prediction model for Web prefetching, IEEE Comput. 36 (3) (2003) 63–70.
[4] Y. Chen, L. Qiu, W. Chen, L. Nguyen, R.H. Katz, Clustering web content for efficient replication, Proceeding of the 10th IEEE International Conference on Network Protocols, Paris, France, November, 2002.
[5] J.G. Cleary, I.H. Witten, Data compression using adaptive coding and partial string matching, IEEE Trans. Commun. 32 (4) (1984) 396–402.
[6] D. Duchamp, Prefetching hyperlinks, Proceedings of the Second USENIX Symposium on Internet Technologies and Systems (USITS'99), Boulder, CO, October, 1999.
[7] L. Fan, P. Cao, W. Lin, Q. Jacobson, Web prefetching between low-bandwidth clients and proxies: potential and performance, Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Atlanta, GA, May, 1999.
[8] J.I. Khan, Q. Tao, Partial prefetch for faster surfing in composite hypermedia, USENIX Symposium on Internet Technology and Systems, San Francisco, CA, March, 2001.
[9] R. Kokku, P. Yalagandula, A. Venkataramani, M. Dahlin, A non-interfering deployable Web prefetching system, USENIX Symposium on Internet Technology and Systems, Seattle, Washington, March, 2003.
[10] T.M. Kroeger, D.D.E. Long, J.C. Mogul, Exploiting the bounds of Web latency reduction from caching and prefetching, Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, CA, April, 1997.
[11] R. Lempel, S. Moran, Optimizing result prefetching in web search engines with segmented indices, Proceedings of VLDB 2002, Hong Kong, China, August, 2002.
[12] V.N. Padmanabhan, J.C. Mogul, Using predictive prefetching to improve World Wide Web latency, Comput. Commun. Rev. 26 (3) (1996) 22–36.
[13] T. Palpanas, A. Mendelzon, Web prefetching using partial match prediction, Proceedings of Web Caching Workshop, San Diego, CA, March, 1999.
[14] J. Pitkow, P. Pirolli, Mining longest repeating subsequences to predict world wide Web surfing, Proceedings of the USENIX Technical Conference, Boulder, CO, April, 1999.
[15] S. Schechter, M. Krishnan, M.D. Smith, Using path profiles to predict HTTP requests, Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, 1998.
[16] R.R. Sarukkai, Link prediction and path analysis using Markov chains, Proceedings of the Ninth International World Wide Web Conference, Amsterdam, Netherlands, 2000.
[17] A. Venkataramani, R. Kokku, M. Dahlin, TCP nice: a mechanism for background transfers, Proceedings of the USENIX Operating Systems Design and Implementation Conference, Boston, MA, 2002.
[18] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, M. Dahlin, The potential costs and benefits of long-term prefetching for content distribution, Proceedings of the Sixth International Workshop on Web Content Caching and Distribution, Boston, MA, 2001.