Proceedings of 2002 International Conference on Parallel Processing, (ICPP'2002), Vancouver, Canada, August 18-21, 2002.

Popularity-Based PPM: An Effective Web Prefetching Technique for High Accuracy and Low Storage *

Xin Chen and Xiaodong Zhang
Department of Computer Science
College of William and Mary
Williamsburg, Virginia 23187-8795, USA

Abstract

Prediction by Partial Match (PPM) is a commonly used technique in Web prefetching, where prefetching decisions are made based on historical URLs in a dynamically maintained Markov prediction tree. Existing approaches either widely store the URL nodes by building the tree with a fixed height in each branch, or only store the branches with frequently accessed URLs. Building the popularity information into the Markov prediction tree, we propose a new prefetching model, called popularity-based PPM. In this model, the tree is dynamically updated with a variable height in each set of branches where a popular URL can lead a set of long branches, and a less popular document leads a set of short ones. Since majority root nodes are popular URLs in our approach, the space allocation for storing nodes are effectively utilized. We have also included two additional optimizations in this model: (1) directly linking a root node to duplicated popular nodes in a surfing path to give popular URLs more considerations for prefetching; and (2) making a space optimization after the tree is built to further remove less popular nodes. Our trace-driven simulation results comparatively show a significant space reduction and an improved prediction accuracy of the proposed prefetching technique.

1 Introduction

Accurately predicting Web surfing hyperlink paths based on the historical information of the surfing paths, we are able to reduce the Web access latencies by prefetching to-be-used Web data. Web prefetching is becoming important and demanding, even though the Web caching technique has been improved.

*This work is supported in part by the National Science Foundation under grants CCR-9812187, EIA-9977030, and CCR-0098055.

An important task for prefetching is to build an effective prediction model and its data structure for storing highly selective historical information. Prediction by Partial Match (PPM) is a commonly used technique in Web prefetching, where prefetching decisions are made based on historical URLs in a dynamically maintained Markov prediction tree. For a Web server supporting a huge amount of Web pages, a prefetching system built by the traditional PPM model occupies a significant portion of the memory space. We have conducted a comprehensive study to evaluate common Web surfing patterns and regularities related to popularities of URLs [6]. We characterize the surfing behavior of each individual client as an *access session* which consists of a sequence of Web URLs continuously visited by the same client. If a client has been idle for more than 30 minutes, we assume that the next request from the client starts a new access session. We have analyzed the surfing patterns of multiple trace files (to be discussed in the next section), and observed that high percentage of access sessions follow three strong regularities.

- **Regularity 1:** *Majority clients start their access sessions from popular URLs of a server. However, majority of URLs in a server are not popular files.*
- **Regularity 2:** *Majority long access sessions are headed by popular URLs.*
- **Regularity 3:** *The accessing paths in majority access sessions start from popular URLs, move to less popular URLs, and exit from the least URLs. The accessing paths in minority access sessions start from less popular URLs, and remain in the same type of URLs, and exit from the least URLs.*

The popularity information is ignored in existing prediction models, which is their major and common weakness. Building the popularity information into the Markov prediction tree, we propose a new prefetching model, called

popularity-based PPM, to achieve both high prediction accuracy and low storage requirement.

Conducting trace-driven simulations, we show that our technique effectively combines the advantages of the existing PPM prefetching approaches, and addresses their limits. Compared with two existing PPM techniques, we show the popularity-based PPM significantly reduces the required space for storing the nodes by 1 to several dozen times. Furthermore, we show that our technique outperforms these techniques by 5 to 10% of hit ratios in most cases. We also demonstrate that the proposed prediction method is highly effective to be used between Web servers and Web proxies.

Our proposed prefetching technique not only reduces storage space significantly and improves the prediction accuracy, but also has the following advantages for Web servers. First, with the efficient data structure of compacted trees, the proposed technique significantly reduces the Web server processing time for prefetching, downloading the server burden. Second, the storage space requirement increases slightly as the number of days for URLs increases. In contrast, the storage requirement for existing PPM techniques increases in a much faster pace, limiting its scalability. Finally, since the popularity of Web files is normally stable over a long period, a prefetching system focusing on popular URLs that are frequently accessed would be relatively reliable. The study in [22] has a similar observation.

2 Evaluation Methodology

The popularity-based PPM model and related performance issues are evaluated by trace-driven simulations. The evaluation environment consists of different Web traces, a simulated server where different PPM models are built to make prefetching decisions, and multiple clients sending requests to the server and receiving requested and prefetched data from the server.

2.1 Traces

The Web traces we have used for performance evaluation are most recently available in public domains:

1. NASA [17]: There are two trace files available in the Internet Traffic Achieve site, which contain HTTP requests of two months to the NASA Kennedy Space Center WWW server in Florida. We have used the trace file of July, collected from 00:00:00 July 1, 1995 through 23:59:59 July 31, 1995, a total of 31 days. The interval of the timestamp is one second.

2. UCB-CS [9]: This trace contains HTTP requests of one week to the Web site (Apache httpd server) of the Department of Computer Science at the University of California at Berkeley. The recorded requests were from July 1, 2000 (Saturday), 00:00 am to July 11, 2000 (Monday) 00:00 am. The internal of the timestamp is one second.

2.2 Simulation

We have built a simulator to construct Web servers connected to clients. The requests to servers can be directly sent from the clients or from proxy caches. Several prefetching PPM models which will be discussed in next sections are built in the server. The server makes prefetching decisions based on the PPM models by sending both requested and prefetched data to the targeted clients. The PPM models are dynamically maintained and updated based on historical data during a period of time.

The simulator consists of two function parts: (1) analyzing traces and building PPM models, and (2) making predictions.

In practice, image files can be embedded in an HTML document. The embedded image files can be in types of ".gif", ".xbm", ".jpg", ".jpeg", ".gif89", ".tif", ".tiff", ".bmp", ".ief", ".jpe", ".ras", ".pnm", ".pgm", ".ppm", ".rgb", ".xpm", ".xwd", ".pcx", ".pbm", and ".pic". An HTML document can be in types of ".html", ".htm", ".shtml". If an HTML file of the same client is followed by image files in 10 seconds, we consider the image file as an embedded file in the HTML file. For these embedded files, we record them with the HTML files.

The simulator assumes both proxies and browsers exist, and are connected to the server. Identifications of users are useful information to construct the prediction structure. Unfortunately, limited HTTP log files identify the users making requests. Although some logs have unique user IDs for clients (for example, by storing HTTP cookies), this type of log files has only been available recently, but not available in the public domain. We have to use IP address which may represent proxy servers. We recognize that this may introduce some inaccuracy in our simulation, but it will not affect evaluation of the principles of different prefetching models.

The simulator makes the following assumption. If an address (or IP) sends requests more than 1000 per day, it is considered as a proxy, otherwise it is a browser. The proxy is assumed to have a disk cache size of 16 GB and a browser is assumed to have a cache of 10 MB. The cache replacement algorithm used in our simulator is LRU.

2.3 Performance Metrics

- *Hit ratio* is the ratio between the number of requests that hit in browser or proxy caches and the total number of requests.

- *Latency reduction* is the average access latency time reduction per request.
- *space* is the required memory allocation measured by the number of nodes for building a PPM model in the Web server for prefetching.
- *traffic increment* is the ratio between the total number of transferred bytes and the total number useful bytes for the clients minus 1.

3 Popularity-based PPM Model

In data compression community, researchers have developed several context models to use m proceeding symbols to determine the probability of next symbol (an m order Markov model). Prediction by Partial Match (PPM) (see e.g. [7]) is one of such context models, which has been actively used in Web predictions. Entropy analysis and empirical studies have shown that the prediction accuracy increases as the increase of the prediction order in each branch, so does the space storing the URL nodes of the PPM model (see e.g. [23]). In this section, we first review two representative methods to build PPM models for prefetching, and then propose our popularity-based PPM model.

3.1 Relative Popularity

The popularity of a Web URL is normally characterized by the number of accesses to it in a given period of time. The popularities of different URLs can be ranked by a server dynamically from time to time. We have calculated the popularities for all the URLs in each trace file collected from different servers by its Relative Popularity, RP. For a given URL, we have

$$RP = \frac{the\ number\ of\ accesses\ to\ the\ URL}{the\ highest\ popularity\ in\ the\ trace}, \quad (1)$$

where the $highest\ popularity$ is the number of accesses to the most popular URL in the trace. This metric compares each URL with the most popular URL in the trace. For example, an URL with $RP = 10\%$ has received 10% accesses relative to the most popular URLs.

We have further ranked the popularities of URLs by 4 grades in a $\log_{10}$ base: Grade 3: $RP = (10\%, 100\%]$, Grade 2: $RP = (1\%, 10\%]$, Grade 1: $RP = (0.1\%, 1\%]$, and Grade 0: $RP \leq 0.1\%$.

The requests from individual client could be divided as *access session* which consists of a sequence of Web URLs continuously visited by the same client. If a client has been idle for more than 30 minutes, we assume that the next request from the client starts a new access session.

3.2 Two Existing PPM Models

There are two representative approaches to build a PPM prediction model. The first approach is to widely create branches from the historical URL files. The PPM tree is given a fixed height so that the length of each branch is not allowed to grow beyond the height. The left figure in Figure 1 shows the prediction tree structure of the standard PPM model for the access sequence of $\{ABCA'B'C'\}$. The advantage of this approach is that the model can be easily and regularly built with a low complexity. There are two major limits associated with this approach. First, the fixed height Markov prediction tree of PPM does not well match the common surfing patterns presented in the previous section. Second, the prediction accuracy can be low if the tree is short in order to save space; and the storage requirement can grow rapidly for a small height increment of the tree. This approach has been used in several prefetching prototypes and systems (see e.g. [26], [12]).

The other approach is to reduce the space requirement by only storing long branches with frequently accessed URLs, which is called LRS (Longest Repeating Subsequences) [23]. Since majority objects in Web servers are accessed infrequently, keeping only frequently accessed paths would not noticeably affect overall performance, but significantly reduce space requirement. Besides the advantage of space reduction, this approach also has high prediction accuracy from the high-order Markov models in a limited number of branches.

There are two limits for this approach. First, since the LRS-PPM model tree only keeps a small number of frequently accessed branches, prefetching for many less frequently accessed URLs is ignored, thus, the overall prefetching hit rates can be low. Second, in order to find the longest matching sequence, the server must have all the previous URLs of the current session, which is maintained by the server and updated based on the accessing information of clients. This process can be expensive.

3.3 Observations from the Two Models

Using popularity grades, we further examine the effectiveness and prefetching behavior of the standard PPM and LRS-PPM models. We have two following observations.

1. *Majority hit documents from the prefetched files are popular documents.*

Similar to a practical configuration, the height of the prediction tree of the standard PPM model is set to 3 in our experiments (simplified as 3-PPM). We only present the NASA trace here since the experiments with other traces show similar results. The left figure in Figure 2 presents comparisons of the percentage of the hit documents from

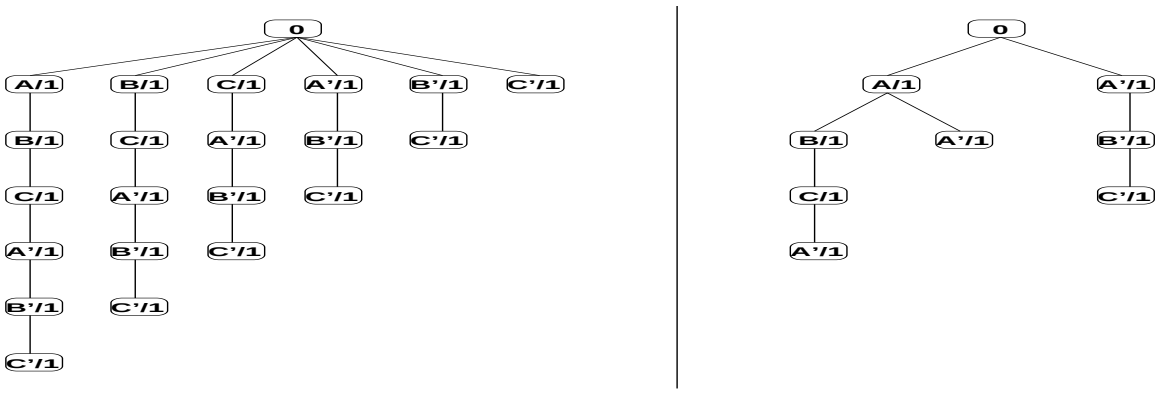

Figure 1. Tree structures of the standard PPM (left) and PB-PPM (right) models for the access sequence of $\{ABCA'B'C'\}$.

the prefetched files using the standard PPM model (3-PPM), the LRS-PPM model (LRS), and our proposed popularity-based PPM (PB-PPM) to be discussed in the next section. Although the percentage of prefetched popular files decreases as the historical record accumulated by multiple days increases, the percentage is at least 60%. The standard PPM model has the lowest percentage. Our experiments also show that the prediction accuracy on popular documents is higher than that on less popular documents in both 3-PPM and PB-PPM models.

2. *A significant amount of predicted branches are unused in the two models.*

In the standard PPM and LRS-PPM models, an URL in a session may be recorded multiple times. For example, for a repeated sequence of URL $\{ABCDEF\}$, C will be recorded twice and F will be recorded 5 times if we do not limit the length of prediction trees in both models. Although a large amount of information for prediction is available, the usage of different paths is not equal. If all accesses to C are after AB, the path BC will never be used. This is common due to the hierarchical structure of Web pages. Since only a small amount of paths are used, the tree structures in both models can occupy large memory space for many unused or infrequently used branches. We also evaluate the tree utilization of 3-PPM model and LRS model. We define a path as a URL sequence from the root to an ending leaf. If this path has been used, we mark it useful. The unmarked paths are unused paths. The right figure in Figure 2 presents the path utilization rates in the three models. When we increase the number of days for predictions, the utilization rates of both 3-PPM and LRS structures decrease rapidly. For example, the utilization rate of the 3-PPM model decrease to less than 20%, and LRS decreases to 40% when 7 days are used.

Our observations further confirm the important roles of popular documents in prefetching, and the lack of the utilization in the two existing models.

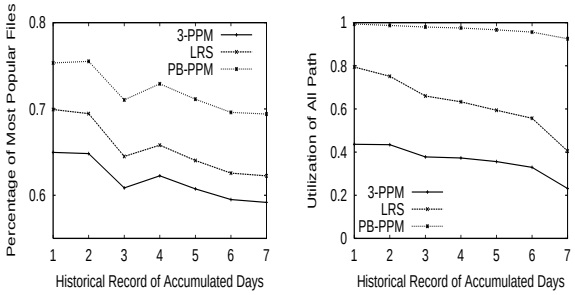

Figure 2. The left figure gives the percentages of popular documents in the total prefetched files. The right figure gives the utilization rates of paths for predictions.

3.4 Constructing a Popularity-based PPM Model

Building the popularity information into the Markov prediction tree, we propose a new prefetching model, called *popularity-based PPM*. In this model, the Markov prediction tree dynamically grows with a variable height in each branch where a popular URL can lead a set of long branches and a less popular document leads a set of short ones. This tree structure combines the advantages of the existing PPM prefetching approaches, and addresses their limits by improving the prediction accuracy from both long branches with popular URLs, while keeping the storage requirement low from only storing less popular documents in a large group of short branches. The popularity-based PPM is built with the following rules.

1. The heights for branches starting from different URLs

are proportional to the relative popularity of the URLs. The proportional differences among different branches can be adjusted to adapt the changes of access patterns.

2. The initial maximum height is set by considering the available memory space for PPM model and access session lengths. If the lengths of most access sessions are short, building long branches in the PPM model may not be necessary. The maximum height is a moderate number in practice. Our experiments show that more than 95% of the access sessions have 9 or less URLs (or clicks). This is consistent with the results reported in [13] for a trace file of 3,247,054 Web page requests from 23,692 AOL users on December 5, 1997.

3. If the popularity grade of an URL not immediately following the heading URL in a branch is higher than the heading URL's grade, or is the highest grade, we will create a special link between the heading URL and a duplicated node of this URL. This approach gives popular URLs more considerations for prefetching, to increase the prediction accuracy and access hit ratios.

4. Each URL in a sequence is added only once to the tree unless the the URLs' popularity grade is higher than the node ahead of it. This approach limits the number of root nodes, effectively reducing the space requirement.

If symbols A, B, and C are three different URLs, notation ABC represents an access sequence to the three URLs by one or more clients. The right figure in Figure 1 gives an example on how a popularity-based PPM model is built based on access sequence $ABCA'B'C'$, where URLs A and A' have a predetermined popularity grade of 3, URLs B and B' have a predetermined popularity grade of 2, and URLs C and C' have a predetermined popularity grade of 1. In this example, the maximum height is 4.

We have further made space optimization after the popularity-based PPM tree is built by combining the following two alternatives. The first optimization is based on the relative access probability of non-root nodes in the tree, which is defined as the ratio between the number of accesses to a URL node and the number of accesses to its parent node. We examine each non-root node, and if the relative access probability is lower than a certain level (ranging 5% to 10% in our experiments), we will remove the node and its linked branches. The second space optimization alternative we have used is to remove each node to which the absolute number of accesses is only one. We will show that these two optimization alternatives can significantly reduce space in next section.

The left figure in 2 shows that the popularity-based PPM model significantly increases the usage of popular documents from the prefetched files, ranging from 70% to 75%. The right figure in 2 shows that the method significantly increases the path utilization rate, ranging from 92% to 100%.

4 Comparative Performance Evaluation

4.1 Simulation Parameters

We have implemented the three PPM prediction models for prefetching: (1) the standard PPM model without limiting the height for each branch, (In some experiments, we had to limit the heights due to limited memory space.) (2) the LRS model keeping the longest repeating subsequences in the PPM tree, and (3) our popularity-based PPM model with the space optimization.

In practice, the standard PPM model should be built with a fixed height for each branch. In order to maximize the prediction accuracy of the standard PPM model, we did not limit the height of each branch while building the model. This will give an upper bound of prediction accuracy for the PPM model.

For the LRS model, if an URL sequence is accessed twice or more, the sequence is considered as a frequently repeating one. A longest sequence covers many independent access sessions which can be removed from the PPM tree by keeping the longest sequence. Our implementations follow the original LRS design [23].

For the popularity-based PPM model, the maximum height of branches is set to 7 for heading URLs of grade 3, 5 for grade 2, 3 for grade 1, and 1 for grade 0 in our experiments. After the popularity-based PPM tree is built, we have done space optimization by cutting each branch which has 10% or lower relative access probability. If the absolute number of accesses to a node is no more than 1, we have also removed the node in for some traces (e.g. the UCB-CS trace).

A longest matching method is used in both the standard PPM and the LRS-PPM models, which matches as many previous URLs as possible to make a prediction. In contrast, when the current clicked URL is a root in the tree, the popularity-based PPM model will make additional predictions with the linked nodes which are duplicated popular ones in the branch.

We set two thresholds for predictions. One is used for the possibility of next accesses, which is set to 0.25 in all the models. Another is used for the maximum size of files to be prefetched. The threshold size affects both hit ratios and the amount of traffic increase. A large threshold allows more data to be prefetched, which is beneficial to hit ratios, but may increase traffics. Since the popularity-based PPM model gives more prefetching considerations to popular nodes, we limit its threshold to 30 Kbytes. The threshold for the standard PPM and LRS-PPM is set to 100 Kbytes.

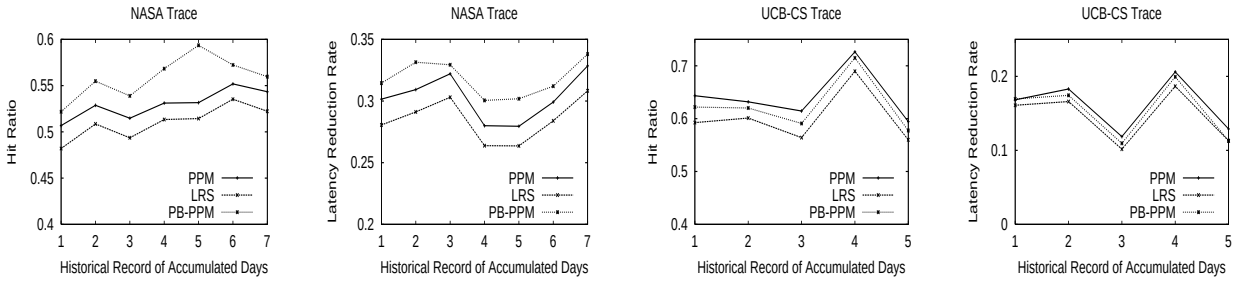

Figure 3. The first and the third figures (from left to right) give the hit ratios of two traces by the three different PPM prediction models. The second and the fourth figures give the comparisons of the latency reduction among the three models.

4.2 Hit Ratios and Latency Reductions

The leftmost figure in Figure 3 presents the changes of the hit ratios using the NASA trace versus the number of days used for predictions in prefetching. Our trace-driven simulation results show that hit ratios of the popularity-based PPM consistently higher than the other two prefetching models. Using historical data of five days to predict data accesses of the sixth day, the hit ratio of the popularity-based PPM is 20% and 13% higher than that of the LRS-PPM and the standard PPM models, respectively.

We estimated connection times and data transferring times by using the method presented in [16], where the connection time and the data transferring time are obtained by applying a least squares fit to measured latency in traces versus the size variations of documents fetched from different remote servers. The second figure in Figure 3 presents the comparisons of latency reductions using the NASA trace versus the number of days used for predictions in prefetching. We show that the popularity-based PPM outperforms both the standard PPM and the LRS-PPM models reducing 4% to 15% more average latency.

The right two figures in Figure 3 presents the performance results using the UCB-CS trace up to 5 days. The third figure in Figure 3 shows that the hit ratio of the popularity-based PPM is slightly lower than the the standard PPM model (a difference of about 2%), and is 6% higher than the LRS-PPM model. The rightmost figure in Figure 3 compares the latency reductions of the three models. The latency reductions of the popularity-based PPM model are slightly lower than the standard PPM (a difference of about 3%), and 4% higher than the LRS-PPM model. Compared with its significant space reduction, the popularity-based PPM model is still the most cost-effective one for the UCB-CS trace although the hit ratios and the latency reductions are not the highest.

4.3 Space and Traffic Overhead

Table 1 compares the number of URLs (nodes) stored by each of the three PPM models for predictions using the NASA trace. We show that number of nodes stored in the standard PPM model dramatically increases as the number files in days used for prediction increases. It is not fair to compare the standard PPM model with other models because we did not limit the height of each branch in the standard PPM model. However, the space requirement comparisons between the LRS-PPM and the popularity-based PPM are reasonable because both models optimize the space usage by different strategies.

The leftmost figure in Figure 4 plots the space requirement in number of nodes for the NASA trace as the number of days used for prediction increases. We show that the number of nodes required for LRS-PPM model proportionally and quickly increases as the number of days increases while the space requirement increases for the popularity-based PPM is in a much slower pace. The LRS quickly increases the space requirement by storing 1.73, 2.9, 3.8, 5.1, 6.4, and 6.9 times more nodes than the popularity-based PPM using 2, 3, 4, 5, 6, and 7 day files for predictions, respectively.

We have two major reasons to explain the quick increase of the number of nodes in LRS-PPM based on our experiments. First, each LRS branch in the model is further cut and paste into multiple sub-branches starting from different URLs. This process will produce node duplications. Second, as the number of day files increases, the number of the longest repeating sequences also proportionally increases, but the numbers of occurrences of subsequences which are also independent sequences decrease. In contrast, the popularity patterns are not changed a lot as the number of day files increases, thus, the popularity-based PPM model only moderately increases the number of nodes stored in the tree structure.

The second figure in Figure 4 compares the traffic in-

Days	1	2	3	4	5	6	7
PPM	424,387	1,008,905	1,674,680	2,588,131	3,115,732	3,575,437	4,133,146
LRS	9,715	19,567	33,233	44,325	56,635	70,247	82,525
PB-PPM	5,527	7,164	8,476	9,156	9,276	9,976	10,411

Table 1. Space size in number of nodes used for each PPM model for the NASA trace.

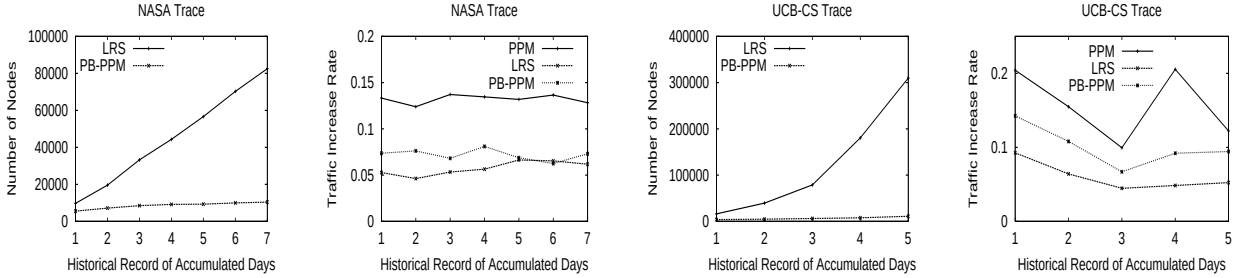

Figure 4. The first and the third figures (from left to right) give the number of nodes using two traces by the LRS-PPM and PB-PPM prediction models. The second and the fourth figures give the comparisons of the traffic increase percentage among the three models.

creases among the three PPM models for the NASA trace. The standard PPM model has the highest traffic increase (around 14%). The traffic increase of the popularity-based PPM model is about 7% while the LRS-PPM model is 6% when up to 4 day files are used. The traffic increases of the two models are quite close when more than 4 day files are used.

Table 2 compares the number of URLs (nodes) stored by each of the three PPM models for predictions using the UCB-CS trace. The two space optimization alternatives have been used in the popularity-based PPM model. Again, the number of nodes stored in the standard PPM model dramatically increases as the number files in days used for prediction increases. The third figure in Figure 4 plots the space requirement in number of nodes for the UCB-CS trace as the number of files in days used for predictions increases. The space reductions by the popularity-based PPM is 1 to several dozen times compared with the LRS-PPM model. The last figure in Figure 4 compares the traffic increases among the three PPM models for the UCB-CS trace. The PPM model has the highest traffic increase (up to 21%). The traffic increase percentage of the popularity-based PPM model is up to 14% which is higher than that the LRS-PPM model (up to 9%). The irregularity of the surfing patterns in the UCB-CS trace is a major reason for this difference. The popularity grades of the starting URLs are evenly distributed in the UCB-CS trace, and some of the popular entries may not lead to long sessions. Thus, some of the branches headed by less popular URLs in the popularity-based PPM model are given short heights, but may be accessed as frequently as the branches headed by popular URLs.

5 Prefetching between Servers and Proxies

Proxies have been widely used as caching storages in the Internet between Web servers and clients. In this section, we evaluate the effectiveness of the popularity-based PPM prefetching between Web servers and proxies.

In our experiments, we randomly select 1 to 32 clients to connect with a proxy. The total document hits come from three sources: (1) hits on browsers, (2) hits on the cached documents in the proxy, and (3) hits on the prefetched documents in the proxy. Since our PB-PPM is sensitive to the threshold that is the maximum size of documents to be prefetched, we tested two thresholds of 40KB and 100KB in our experiments (simplified as PB-PPM-40KB, and PB-PPM-100KB, respectively). Due to the page limit, we only present the results from the NASA trace in this paper. We obtained similar results from other traces.

The left figure in Figure 5 presents comparative total hit ratios among the standard PPM (PPM), the LRS-PPM (LRS), PB-PPM-40KB, and PB-PPM-100KB as the number of the clients increases from 1 to 32. The hit ratio curve of the LRS is the lowest (from 42% to 71%), while the PB-PPM-100KB curve is the highest (from 61% to 78%). The PB-PPM-40KB and the standard PPM curves are in the middle. As the number of clients increases to 24 and more,

Days	1	2	3	4	5
PPM	3,339,315	8,872,552	10,674,669	21,579,994	43,365,678
LRS	16,200	39,437	78,816	180,521	309,916
PB-PPM	3,840	4,690	6,192	7,684	10,981

Table 2. Space size in number of nodes used for each PPM model for the UCB-CS trace.

the standard PPM hit ratio curve merges to that of the PB-PPM-40KB. This study indicates that the popularity-based PPM can achieve high hit ratios with a moderate threshold size for the prefetched documents, such as 100KB, in the environment of Web servers and proxies.

The right figure in Figure 5 presents the comparative the network traffic increments among the three models. As the number of clients increases, the traffic increment decreases for all the three models. We show that the standard PPM has the highest traffic increment (20% for 32 clients, while the PB-PPM-40KB has the lowest traffic increment (10% for 32 clients), although they have similar hit ratios. We also show that there is a tradeoff between increasing hit ratios and lowering traffic increment in the popularity-based PPM model. By adjusting the threshold size of prefetched documents, we are able to address the tradeoff.

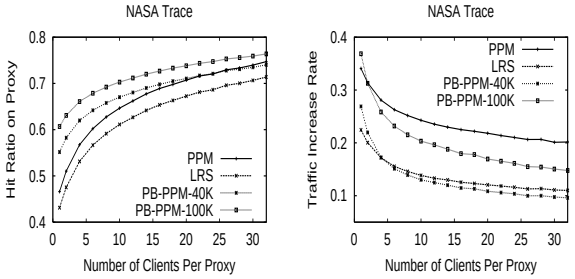

Figure 5. The left figure gives comparative proxy hit ratios among the three prefetching models, and the right figure gives the comparative network traffic increments between Web servers and the proxy among the three prefetching models.

6 Other Related Work

Prefetching methods can be divided into three types: server-initiated, client-initiated, and cooperative prefetching. In server-initiated prefetching, servers always push their popular documents to the clients. The work reported in [2] uses service proxies (agents) to serve for multiple home servers, and the mapping between home servers and service proxies can be many-to-many. Markatos and Chronaki [20] propose a Top-10 approach which combines the servers' active knowledge of their popular documents with client access profiles. Web servers regularly push their most popular documents to Web proxies, and proxies then push those documents to the active clients.

In client-initiated prefetching, the information observed by the client side are used to determine the prefetching contents. A simulation study on accesses to two popular UCLA sites is reported to present a client-initiated prefetching software [15]. Wcol [27] is a proxy software that parses HTML and prefetches documents, links and embedded images. Duchamp [11] presents a new implementation of client-based prefetching system which collects the knowledge of popular URLs for clients to initiate prefetching.

A cooperative prefetching between a server and a client/proxy uses the information of the two sides. Cohen et. al. [8] propose an end-to-end approach to improving Web performance by collectively examining servers and clients. Server groups the relative files together, and the proxy prefetches selected files. Server and client/proxy communicates with each other through data piggybacking. Another study in [5] proposes a coordinated proxy-server prefetching technique that adaptively utilizes the reference information and coordinates prefetching activities at both proxy and Web servers.

Using the first-order of Markov models and speculations based on statistical information of servers, other researchers have developed prefetching methods (see e.g. [2], [21], and [25]). Researchers have also proposed hint-based Web clients and servers which can load/store the predicted data objects for future use. (see e.g. [8] and [19]). The access distribution among all Web URLs operated by Google search engine is part a model of "random walks" [3]. These studies do not consider building the common surfing patterns and regularities into prediction models. High orders or variable orders of Markov models are also not concerned issues for prefetching. Related performance evaluation on caching and network effects has been reported in many studies (see e.g. [4, 10, 14]).

7 Conclusion

Building popularity information into the PPM model, we have proposed and evaluated the popularity-based PPM model for Web prefetching with an objective of high accuracy and low space requirement. Conducting trace-driven simulations, we have shown its effectiveness measured by hit ratios, latency reductions, space requirement, and traffic increase percentage. We have also applied the proposed PPM model to the prefetching between servers and proxy caches, and showed that it is effective in this environment to improve the hit-ratios while reducing network traffics.

Acknowledgment: This work is a part of an independent research project sponsored by the US National Science Foundation for program directors and visiting scientists.

References

[1] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web client access patterns: characteristics and caching implications", *World Wide Web Journal*, 2(1):15-28, January,1999.

[2] A. Bestravos, "Using speculation to reduce server load and service time on the WWW", *Proceedings of the 4th ACM International Conference on Information and Knowledge Management*, (CIKM'95), Baltimore, Maryland, 1995.

[3] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine", *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[4] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich, "Web proxy caching: the devil is in the details", *Proceedings of the Workshop on Internet Server Performance*, Madison, Wisconsin, June 1998.

[5] X. Chen and X. Zhang, "Coordinated data prefetching by utilizing reference information at both proxy and Web servers", *Proceedings of 2nd Performance and Architecture of Web Servers*, PAWS'01, Boston, MA, June 2001.

[6] X. Chen and X. Zhang, "Popularity-based Web surfing patterns", Technical Report, Department of Computer Science, College of William and Mary, Ocotober 2001.

[7] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching", *IEEE Transactions on Communications*, Vol. 32, No. 4, pp. 396-402, 1984.

[8] E. Cohen, B. Krishnamurthy, and J. Rexford, "Improving end-to-end performance of the Web using server volumes and proxy filters", *Proceedings of the ACM SIGCOMM 1998*, 1998, pp. 241-253.

[9] Computer Science Department, University of California, Berkeley, URL: http://www.cs.berkeley.edu/logs/.

[10] M. Crovella and P. Barford, "The network effects of prefetching", *Proceedings of the IEEE INFOCOM'98 Conference*, San Francisco, California, April, 1998.

[11] D. Duchamp, "Prefetching hyperlinks", *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems* (USITS'99), October 1999.

[12] L. Fan, P. Cao, W. Lin, and 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*, May 1999, pp. 178-187.

[13] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose, "Strong regularities in world wide Web surfing", *Science*, Vol. 280, April 3, 1998, pp. 95-97.

[14] A. K. Iyengar, E. A. MacNair, M. S. Squillante, and L. Zhang, "A general methodology for characterizing access patterns and analyzing Web server performance", *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Montreal, Canada, July 1998.

[15] Z. Jiang and L. Kleinrock, "An adaptive network prefetch scheme", *IEEE Journal on Selected Areas of Communication*, Vol. 17, No. 4, 1998, pp. 358-368.

[16] S. Jin and A. Bestavros, "Popularity-aware greedyDual-size Web proxy caching algorithms", *Proceedings of 20th International Conference on Distributed Computing Systems*, (ICDCS'2000), April 2000.

[17] Lawrence Berkeley National Laboratory, URL: http://ita.ee.lbl.gov/

[18] J. I. Khan and Q. Tao, "Partial prefetch for faster surfing in Composite Hypermedia", *USENIX Symposium on Internet Technology and Systems*, San Francisco, California, USA , March 2001. USENIX Association.

[19] T. M. Kroeger, D. D. E. Long, and 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, California, April 1997.

[20] E. P. Markatos and C. E. Chronaki, "A top-10 approach to prefetching on the Web", Technical Report, No. 173, ICS-FORTH, Heraklion, Crete Greece, August 1996.

[21] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve World Wide Web latency", *Computer Communication Review*, 1996, pp. 22-36.

[22] V. N. Padmanabhan and L. Qiu, "The content and access dynamics of a busy Web site: findings and implications", *Proceedings of the ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[23] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict world wide Web surfing", *Proceedings of the 1999 USENIX Technical Conference*, April 1999.

[24] R. Sarukkai, "Link prediction and path analysis using markov chains", *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands , May 2000.

[25] S. Schechter, M. Krishnan, and M. D. Smith, "Using path profiles to predict HTTP requests", *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[26] T. Palpanas and A. Mendelzon, "Web prefetching using partial match prediction", *Proceedings of Web Caching Workshop*, San Diego, California, March 1999.

[27] Wcol Group, "Www collector: the prefetching proxy server for WWW", 1997. URL: http://shika.aist-nara.ac.jp/products/wcol/.