A Popularity-Based Prediction Model for Web Prefetching

The popularity-based PPM model uses grades to rank URL access patterns and builds these patterns into a predictor tree to aid Web prefetching. In a trace-based simulation, this model was more accurate and used less space than other prediction models.

Xin Chen
Xiaodong Zhang
College of William and Mary

A challenging problem for Internet computing is how to reduce Web access latencies. With the increase in both server and client types, the number of unique file objects cached in client browsers continues to multiply, and Web access behavior is becoming more erratic. Approaches that rely solely on caching offer limited performance improvement[1] because it is difficult for caching to handle the large number of increasingly diverse files.

A promising solution to Web access latencies is to combine caching with Web prefetching—obtaining the Web data a client is expected to need on the basis of data about that client's past surfing activity. Prefetching adds efficiency because it actively preloads the data for two kinds of clients. For group clients, it preloads commonly shared data objects. For individual clients, it loads each of their popular objects. A study shows that the performance improvement with caching and prefetching can be twice that of caching alone.[2]

An important prefetching task is to build an effective prediction model and data structure for storing highly selective historical information. The prediction by partial match (PPM) model,[3] for example, which is widely used for Web prefetching, makes prefetching decisions by reviewing the URLs that clients on a particular server have accessed over some period. The model structures these URLs in a Markov predictor tree that the server dynamically maintains. (A Markov tree is an m-order context tree that uses m preceding symbols to determine the probability of the next one.)

For a Web server that supports millions of Web pages, however, this kind of prefetching takes too much memory, or storage space. Some variations of the PPM model attempt to avoid this overhead by having the servers collect access information for specified documents in near real time,[4] but they sacrifice prediction accuracy because there is less historical information.

We propose a variation of the PPM model that builds common surfing patterns and regularities into the Markov predictor tree. The model assigns long branches to popular URLs—ones that clients access frequently—and shorter branches to less popular URLs. The server dynamically updates the tree with a maximum height for each branch type. Because the root nodes are the most popular URLs—not *all* URLs, as in the standard PPM model—our model effectively uses the space allocated for storing nodes. It also performs two optimizations: It directly links a root node to duplicated popular nodes in a surfing path to give popular URLs more consideration, and it goes back to the completed tree to remove less popular branches.

Because the tree in our model has varied branch lengths, it effectively negotiates the tradeoff between predictive accuracy and memory. By limiting the number of less popular documents (short branches), the tree uses less memory, yet it preserves accuracy because it includes the access information that is most likely to result in a prediction hit.

To verify our model's memory efficiency, predictive accuracy, and general performance, we conducted trace-driven simulations using the data set

0018-9162/03/$17.00 © 2003 IEEE

Simulation Environment

Our simulation environment consisted of traces, a simulated server in which different PPM models make prefetching decisions, and multiple clients that send requests to the server and receive requested and prefetched data from the server.

The traces, the WorldCup98 data set (available from Lawrence Berkeley National Laboratory, http://ita.ee.lbl.gov), consisted of all the requests made to the 1998 World Cup Web site between 26 April 1998 and 26 July 1998, which represents 92 days of access. During this time, the site received 1,352,804,107 requests. The access events were recorded in 1-second intervals.

The simulated server dynamically maintained and updated three PPM models—a standard model, the longest repeating sequence PPM model, and our popularity-based model—according to these traces. The server assumed that both proxies and browsers were connected to it. The predictor in the simulator assumes that if an address (or IP) sends more than 1,000 requests per day, it is a proxy; otherwise, it is a browser. The predictor assumes that the proxy has a disk cache of 16 Gbytes and that a browser has a cache of 10 Mbytes. The proxy and user clients use a standard least recently used cache replacement alogorithm.

User identification provides useful information for constructing the prediction structure. Unfortunately, obtaining the HTTP log files that identify users was difficult. Some logs have unique user IDs for clients—for example, HTTP cookies—but this type of log file is not available in the public domain. Thus, we used IP addresses, which may represent proxy servers. We recognize that using IP addresses could introduce some inaccuracy in our simulation, but we do not believe it affects our evaluation of the different prediction models.

Finally, in practice, an HTML document can contain embedded image files. Thus, when a client accessed an HTML file and then accessed an image file within the next 10 seconds, we considered the image file to be embedded in the HTML file. We recorded these embedded files as a part of the HTML files.

from 92 days of requests to the 1998 World Cup site. Our simulations demonstrate that the tree structure with variable heights in different branches improves prediction accuracy in both long and short branches while keeping the storage requirement low. The "Simulation Environment" sidebar describes more about our server and evaluation method.

A study comparing our model's performance with the standard PPM model and the longest repeating sequences (LRS) PPM model[5] demonstrates that our model not only is significantly more space efficient, but also provides the most accurate predictions.

SURFING PATTERNS

We observed several popularity patterns during trace analysis. A URL's *popularity* is the number of times users access it in a given period. To calculate the popularities for all the URLs in the trace file, we used *relative popularity*—the individual URL's popularity divided by the highest popularity in the trace. Thus, if URL *A* had the highest number of accesses, its RP would be 100 percent. If URL *B* had 10 percent of URL *A*'s accesses, its RP would be 10 percent.

We further ranked URL popularity by four grades:

- grade 3, 10 percent < RP ≤ 100 percent;
- grade 2, 1 percent < RP ≤ 10 percent;
- grade 1, 0.1 percent < RP ≤ 1 percent; and
- grade 0, RP ≤ 0.1 percent.

We characterized each client's surfing behavior as an *access session*—the sequence of Web URLs that the client continuously visited. If a client was idle for more than 10 minutes, the next request from that client started a new session.

Each access session is composed of the steps the client takes to complete the session, and each step has a sequence number, starting with step 0, the first URL visited in the session. The total number of steps determines the session length, the number of steps per session varies, and each trace file consists of numerous access sessions.

Thus, a trace file with P sessions is a two-dimensional array RP (i,j), where index i represents the ith session in the trace ($i = 1, ..., m$), and index j represents the jth step in the session ($j = 1, ...$). We sorted sessions by length, which decreased as the index increased.

Our trace analysis revealed two important recurring patterns, or regularities. All the evaluation days exhibited these regularities to some degree, but for illustration, we randomly chose results from 13 July, the 79th day.

Distribution of popularity grades

To examine the relationship between URL popularity and access session, we first divided each trace into four session groups, which differed in the popularity grade of their starting URLs (grades 3, 2, 1, and 0, respectively).

The first regularity we observed is that most URLs in a server are not popular files, but most clients start an access session from a popular URL. Figure 1 shows these popularity patterns.

These results offer both good and bad news for building prefetching models. The good news is that paying special attention to popular URLs that are only a small percentage of total URLs can be effective. The bad news is that the accumulated number of accesses to less popular URLs can be large, so focusing on only a small percentage of them in prefetching could result in low hit rates.

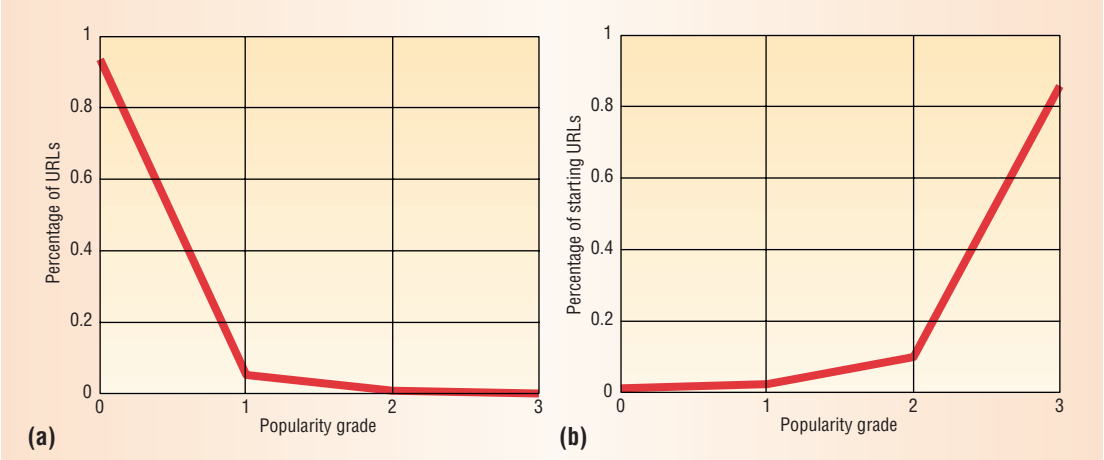

Figure 1. Popularity patterns in Web access sessions during day 79 of the WorldCup98 data set. (a) The percentage of URLs of each grade and (b) the percentage of starting URLs in all sessions for each popularity grade. Fewer than one percent of URLs are of grade 3 popularity (10 percent < RP ≤ 100 percent). In contrast, less popular URLs dominate, with grade 0 URLs representing about 95 percent of the total URLs in that day's traces. Results for this day were fairly consistent with the rest of the evaluation period.

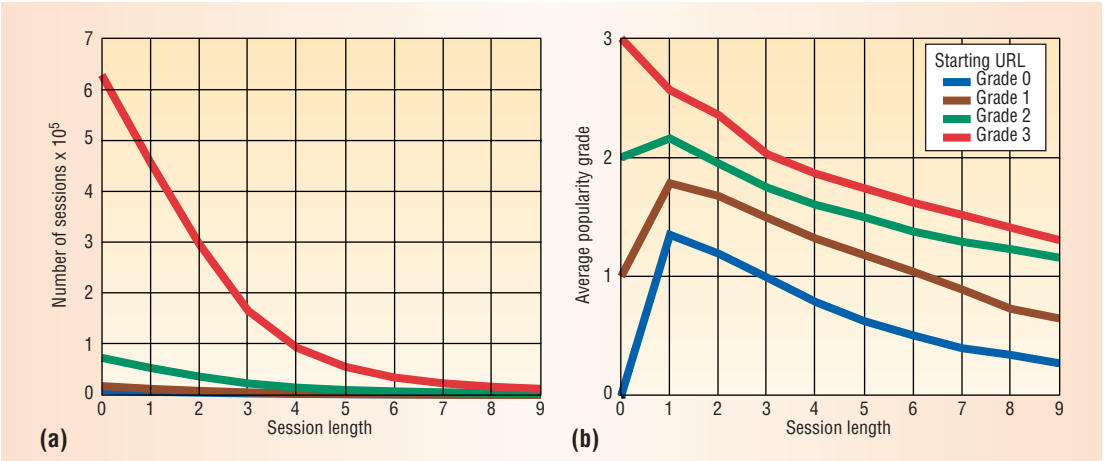

Figure 2. URL popularity and access-session length. (a) The number of sessions remaining as session length increases, and (b) the average popularity grade across session length, which shows that the average popularity grade decreases as the session length increases.

Popularity and session length

We then plotted the changing curves using the number of access sessions as a function of session length. Figure 2 shows the results for day 79's traces. Fully 86 percent of the access sessions started from popular URLs, moved to less popular URLs, and exited from the least popular ones. Only 1.3 percent of the sessions started from less popular URLs, remained in the same type of URLs, and exited from the least popular ones.

Thus, the second regularity we observed is that most URL sessions start with a popular URL, move to less popular URLs, and exit from the least popular ones. The least sessions start from less popular URLs, remain in less popular URLs, and exit from the least popular ones.

Our data analysis consistently shows that the starting URLs determined the number of access sessions for a given length. In Figure 2a, 628,232 sessions started with URLs of grade 3 popularity. As session length increased, the remaining sessions decreased proportionally. For example, 13,293 of the remaining sessions had length 9. In contrast, the remaining sessions with length 9 for sessions starting with URLs of grades 2, 1, and 0 popularity were 4,757, 1,237, and 1,740.

Figure 2b shows that the average popularity grade always decreased as the session length increased. For example, the average popularity of the access sessions starting from URLs of grades 3 and 2 popularity decreased proportionally as session length increased. When the session length increased from 0 to 9, the average popularity grade of the URLs decreased from 3 to 1.24 and from 2 to 1.1. For access sessions starting from URLs of grades 1 and 0 popularity, the average popularity

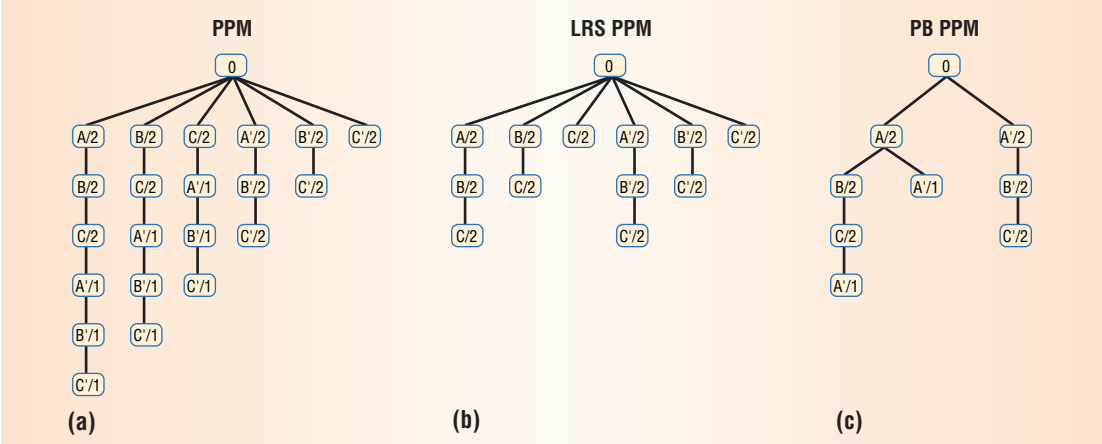

Figure 3. Three prediction models for three access sessions—{ABCA′B′C′}, {ABC}, and {A′B′C′}—which the server in the trace simulation used to make predictions for prefetching: (a) the standard PPM model, (b) the LRS PPM model, and (c) the popularity-based-PPM.

changed only slightly as session length increased. Thus, we can infer that clients starting with less popular URLs tend to surf among URLs with the same popularity grade.

THREE PREDICTION MODELS

To evaluate our popularity-based PPM model against other PPM models, we built three PPM models, shown in Figure 3, into the server as the basis for its prefetching decisions. The server dynamically maintained and updated the PPM models according to traces over the 92-day evaluation period.

Standard model

The first model is the standard PPM model, which Figure 3a shows for three access sequences: {*ABCA′B′C′*}, {*ABC*}, and {*A′B′C′*}. The standard PPM model uses multiple Markov models to store historical URL paths. Each Markov model partially represents a client session. The model structure is a tree, and each branch is a Markov model with multiple URL predictors. Variable orders in each branch can make predictions.

Node 0 represents the root of the forest. When a client accesses URL *A*, the server builds a new tree with root *A* and sets the access counter to 1. When *B* comes, it creates another tree with root *B*. Because *B* follows *A* in the same session, the server must generate another node for *B* as a child node of *A*. The process completes until the server has scanned all the URLs accessed in the three sessions.

Several prefetching prototypes and systems[6-8] use this standard model, which follows three main structural rules:

- It uses any URL for a root node and records every subsequent URL in the tree rooted by the first URL.
- Every node represents an accessed URL in the server. A counter records the number of times the URL occurs in the path from the root node. For example, the notation *A*/2 indicates that URL *A* was accessed twice.
- Every path from every root node to a leaf node represents the URL sequence for at least one client.

The advantage of the standard PPM model is that it is not very complex and is thus easy to build. Because the tree records every accessed URL, however, it takes up too much space in the server. Entropy analysis and empirical studies have shown that as the prediction order in each branch increases, so does the space that stores the PPM model's predictors, and prediction accuracy improves.[5]

Some variations of the standard model attempt to fix the tree height (put a ceiling on the number of accessed URLs that can become nodes). This saves storage, but the tree no longer matches common surfing patterns. Also, prediction accuracy can be low with a short tree, and even a small height increase can rapidly increase the storage requirement.

For our performance comparison, we used a standard PPM model with a maximum branch height of 7. In practice, the branches in a standard PPM model should have a fixed height, but our experiments show that prediction accuracy will degrade if the branches are too short. We also fixed the height at 7 to make the tree height in the standard model equal to that in our model to provide a more reasonable basis for comparison.

LRS model

The other representative approach to building a PPM model is the LRS PPM model, which keeps the longest repeating subsequences and stores only long branches with frequently accessed URL predictors.[5] A longest sequence is a frequently repeating sequence in which at least one occurrence of one subsequence belongs to an independent access session. Thus, a longest sequence covers many independent access sessions. As in the standard model, the tree height is not fixed. A sequence of URLs that a client accesses more than once is considered a repeated sequence. Figure 3b shows the predictor

tree structure of the LRS PPM model for the three access sequences. The server builds the tree the same way as in Figure 3a, but it then scans each branch to identify and eliminate nonrepeating sequences, such as {*A*′1, *B*′1 and *C*′1}.

Relative to the standard PPM model, the LRS PPM model offers a lower storage requirement and higher prediction accuracy. Clients access most objects in Web servers infrequently, so keeping only frequently accessed paths does not noticeably affect overall performance, but it does significantly reduce the storage requirement. The high prediction accuracy comes from the model's use of high-order Markov models in a limited number of branches.

The LRS PPM model also has limitations. Because the tree keeps only a small number of frequently accessed branches, it ignores prefetching for many less frequently accessed URLs, so overall prefetching hit rates can be low. Also, to find the longest matching sequence, the server must have all the previous URLs of the current session, which means the server must maintain sessions and update them. This process can be expensive.

For our performance comparison, we used the original LRS design.[5] If clients accessed a URL sequence more than once, we considered it to be frequently repeating.

Popularity-based model

The third approach is our model, which uses only the most popular URLs as root nodes. Figure 3c shows the tree structure for the three access sequences, where URLs *A* and *A*′ have grade 3 popularity, URLs *B* and *B*′ have grade 2 popularity, and URLs *C* and *C*′ have grade 1 popularity. In this example, the maximum branch height is 4. The server creates a root node only for the starting node and when it detects URLs with grade 3 popularity. Thus, for *A* it creates a root node, but for *B*, it creates only a child node of *A*. It does the same for C. When it detects *A*′, it generates another root node and a child node of C. It also links child node *A*′ to root node *A*.

Our popularity-based model builds surfing regularities into the standard PPM model's Markov predictor tree using four rules:

- *Rule 1.* Set the maximum height initially for branches starting from the most popular URLs. The heights of other branches starting from less popular URLs decrease proportionally. Adjust the proportional differences among different branches to adapt to access pattern changes.
- *Rule 2.* Set the initial maximum height by considering the available memory space for the PPM model and access session lengths. The session length reflects the demand for data prefetching. If the lengths of most access sessions are short, building long branches may not be necessary. The maximum height is a moderate number in practice. Our experiments show that more than 95 percent of the access sessions have nine or fewer URLs (or clicks). This is consistent with the results reported elsewhere[9] for a trace file of 3,247,054 Web page requests from 23,692 AOL users on 5 December 1997.
- *Rule 3.* In most case, add each URL in a sequence only once to the tree. Create a special link between the heading URL and a duplicated node of this URL only if a URL not immediately following the heading URL has a popularity grade higher than the heading URL's grade or has the highest or second highest grade. If this popular node leads to a sequence of URLs, add the sequence to that root or build a new tree rooted by the node. This approach gives popular URLs more considerations for prefetching, aiming at increasing prediction accuracy and access hit ratios.
- *Rule 4.* Periodically build the model on the basis of log files for a previous time interval to predict the surfing patterns in the coming time interval. The interval can be a day, a week, or a month. Dynamically build or adjust the model as each URL request arrives.

The popularity-based model uses only the most popular URLs as root nodes.

Our model also makes space optimizations to the completed tree. The first is based on the relative access probability of nodes (URLs) that are not root nodes, which is the ratio between the number of accesses to a URL and the number of accesses to its parent URL. The server examines each nonroot node, and if the node's relative access probability is less than a certain percentage (predetermined), it removes the node and the branches to its children nodes and all the connected nodes of younger generations. It also removes each node representing a URL that clients accessed only once.

For our performance comparison, we set the maximum branch height to 7 for grade 3 URLs, to 5 for grade 2 URLs, to 3 for grade 1 URLs, and to 1 for grade 0 URLs. We had the server cut each branch with a 5 percent or lower relative access probability.

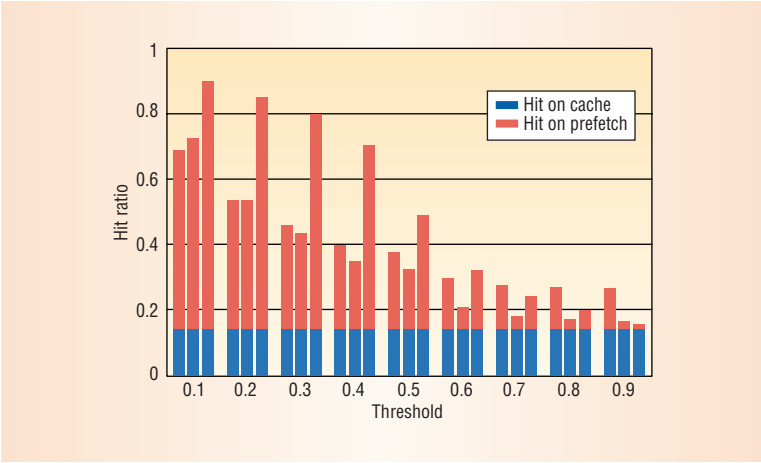

Figure 4. The hit ratio for three PPM models with different thresholds. Each set of three bars for each threshold represents hit ratios for the standard PPM model (left bar in each group), the LRS PPM model (middle bar in each group), and our popularity-based PPM model (right bar in each group). The hit ratio is the number of requests that hit in a browser or proxy cache relative to the total number of requests. The thresholds represent the probability that clients may access that document.

COMPARATIVE PERFORMANCE

All the PPM models use a longest matching method, which matches as many previous URLs as possible to make a prediction. If the model does not find a prefix match, it will not make a prediction. We set a document selection threshold between 0.1 and 0.9, which meant that the server would prefetch only documents with a relative access probability equal to or greater than the threshold.

We used three performance metrics:

- The hit ratio is the number of requests that hit in a browser or proxy cache as related to the total number of requests.
- The traffic increment is the ratio between the total number of transferred bytes and the total number of bytes that clients find useful minus 1. The traffic increment is 0 percent if clients find every transferred byte useful.
- Space is the required memory allocation measured by the number of nodes available to build a PPM model in the Web server.

The maximum size of prefetched files affects both hit ratios and the traffic increment. A large value lets the server prefetch more data, which helps the hit ratios but may increase traffic. We set the maximum prefetched file size to 20 Kbytes for all three models in all experiments. We also selected day 46 from the WorldCup98 traces, which was one of the busiest days of the evaluation period. We used the day 45 traces as the training data to build the tree structures for the three models.

Hit ratios

Figure 4 shows the changes in the hit ratios using the day 45 traces versus the threshold used for predicting access on day 46. The threshold 0.3, for example, contains the hit ratios (vertical bars) when the server prefetches documents with an access probability of 0.3 or higher.

As the figure shows, the hit ratios were consistently higher with our popularity-based PPM model than with the standard PPM and LRS PPM models when the threshold is equal to or less than 0.6. This was generally true of the entire evaluation period (not just day 45). For example, with a 0.3 threshold, the hit ratio of our model is 81 percent—78 percent higher than the LRS PPM or the standard PPM model. With a larger threshold, however, the hit ratio of our model was the lowest because at the higher thresholds, unpopular files (files that clients have accessed only once) become dominant and available for prefetching. Thus, the standard PPM model achieves almost the same hit ratio at thresholds of 0.6 to 0.9. The access probability of unpopular files is 1.0 when clients access two unpopular files continuously.

Both the LRS PPM model and our model use space optimization to keep popular files, which means that the server will never prefetch unpopular files. However, our model's hit ratio decreases faster than the LRS PPM model's hit ratio because the LRS PPM model uses all URLs as roots, whereas our model uses only popular URLs as root nodes.

Because our model deletes URLs that clients seldom access, it has more flexibility than the other models. This is important when the server load and network conditions dynamically determine how aggressive prefetching can be.[10]

Traffic overhead

Figure 5 compares the traffic increment for the three PPM models for predictions of day 46 when the threshold varies from 0.1 to 0.9.

The three models have similar traffic increases when the threshold is less than 0.6. With a 0.2 threshold, for example, traffic increases 10.7, 9.9, and 10.9 percent. With a larger threshold, however, the traffic increase with the LRS PPM model and our model goes down rapidly and closes to 0 with a 0.9 threshold. Overall, the standard PPM model consumes more network bandwidth than the other models. Considering the hit ratios of the three models, our model is the most cost-effective.

Space overhead

Figure 6 compares the number of URLs (nodes) that each model stored for predictions about day 46 access. We used the varied number of clients in

the day 45 traces to build the prediction model for day 46 prefetching.

The number of nodes that the standard PPM model stores dramatically increases as the number of clients used for prediction increases. The nodes required for the LRS PPM model increases proportionally and quickly with more clients, while the space requirement for our model increases at a much slower pace. With 800 clients for prediction, for example, the LRS PPM stores 71 percent more nodes than our popularity-based model. Using 1,600, 3,200, 6,400, 12,800, and 25,600 clients for predictions, the LRS PPM model stores 1.9, 4.4, 8.7, 15.7, and 26.8 times more nodes than our model.

We see two main reasons for this quick increase. First, the LRS PPM model has many node duplications because it cuts and pastes each tree branch into multiple subbranches starting from different URLs. Second, as the clients increase, the number of the longest repeating sequences also proportionally increases, but the number of occurrences of subsequences that are also independent sequences decreases. In contrast, the popularity patterns do not change significantly as the client files increase, so our model only moderately increases the number of nodes in the tree structure.

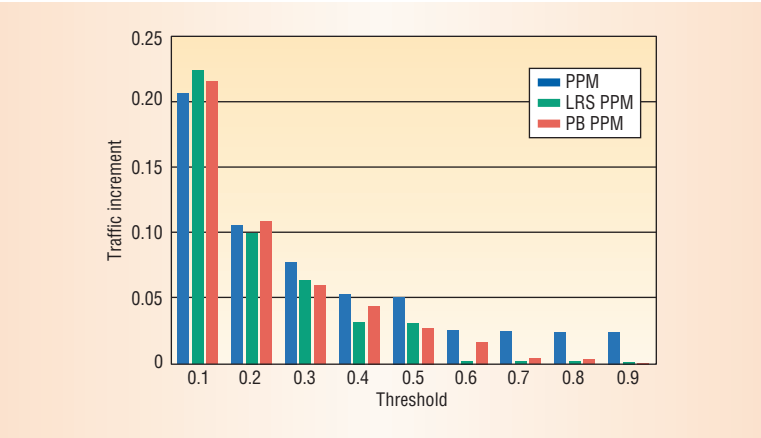

Figure 5. Traffic increment comparisons. Traffic overhead with the standard (PPM), longest repeating sequence (LRS), and popularity-based (PB PPM) models for access probability thresholds of 0.1 to 0.9. Considering the hit ratios of the three models, the PB PPM model is the most cost-effective.

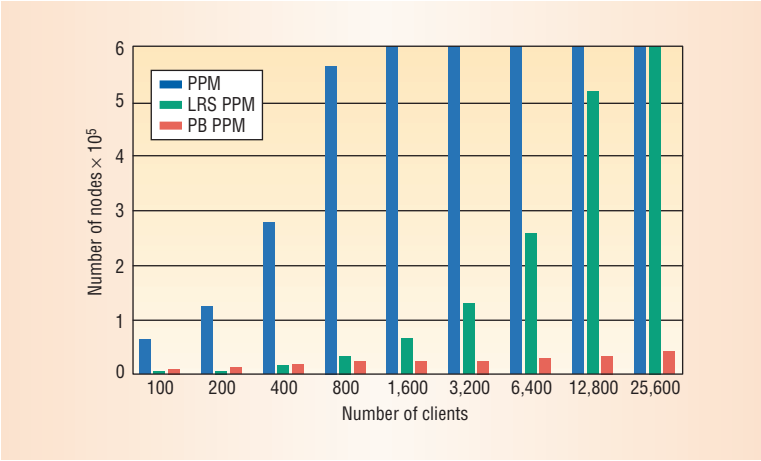

Figure 6. Node storage. The number of nodes the three different PPM predictor models generate is a function of the number of clients used for prediction in the day 45 traces. Because the popularity patterns do not change significantly as the client files increase, the PB PPM model only moderately increases the number of nodes in the tree structure.

A popularity-based prefetching technique is an effective Web management approach because Internet storage has become increasingly large and disorganized. Popularity information makes searching and prefetching highly objective and efficient. Our simulation and data analysis revealed two important popularity-related surfing regularities. By building these regularities into the PPM model, Web prefetching can have both high prediction accuracy and a low space requirement.

Our current work is to refine the model to address dynamic content. We are developing coordinated prefetching techniques within a Web server and between a proxy and a Web server to enable the global use of popularity information. ■

Acknowledgments

We thank the anonymous referees for their helpful and constructive comments.

This work is supported in part by the National Science Foundation under grants CCR-9812187 and CCR-0098055. In addition, this work is also part of an NSF-sponsored independent research project for Xiaodong Zhang.

References

1. L. Xiao and X. Zhang, "Exploiting Neglected Data Locality in Browsers," *Proc. 10th Int'l World Wide Web Conf.*, 2001; www10.org/cdrom/posters/frame.html.
2. T.M. Kroeger, D.D.E. Long, and J.C. Mogul, "Exploiting the Bounds of Web Latency Reduction from Caching and Prefetching," *Proc. Usenix Symp. Internet Technologies and Systems*, Usenix, 1997, pp. 13-22.
3. J.G. Cleary and I.H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Trans. Comm.*, vol. 32, no. 4, 1984, pp. 396-402.

4. D. Duchamp, "Prefetching Hyperlinks," *Proc. Usenix Symp. Internet Technologies and Systems*, Usenix, 1999, pp. 127-138.
5. J. Pitkow and P. Pirolli, "Mining Longest Repeating Subsequences to Predict World Wide Web Surfing," *Proc. Usenix Technical Conf.*, Usenix, 1999, pp. 139-150.
6. L. Fan et al., "Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance," *Proc. ACM SIGMetrics Conf. Measurement and Modeling of Computer Systems*, ACM Press, 1999, pp. 178-187.
7. J.I. Khan and Q. Tao, "Partial Prefetch for Faster Surfing in Composite Hypermedia," *Proc. Usenix Symp. Internet Technologies and Systems*, Usenix, 2001, pp. 13-24.
8. T. Palpanas, *Web Prefetching Using Partial Match Prediction*, master's thesis, Dept. Computer Science, Univ. Toronto, 1998; available as tech. report CSRG-376; www.cs.toronto.edu/~themis/publicationsp. html.
9. B. A. Huberman et al., "Strong Regularities in World Wide Web Surfing," *Science*, vol. 3, Apr. 1998, pp. 95-97.
10. Z. Jiang and L. Kleinrock, "An Adaptive Network Prefetch Scheme," *IEEE J. Selected Areas Comm.*, vol. 17, no. 4, 1998, pp. 358-368.

Xin Chen *is a PhD candidate in computer science at the College of William and Mary. His research interests are distributed systems, networking, and Internet computing. Chen received an MS in computer science from the University of Science and Technology of China. Contact him at xinchen@cs.wm.edu.*

Xiaodong Zhang *is a professor of computer science at the College of William and Mary and the program director of the Advanced Computational Research Program at the US National Science Foundation. His research interests include parallel and distributed systems architecture, and scientific computing. Zhang received a PhD in computer science from the University of Colorado at Boulder. He is a senior member of the IEEE. Contact him at zhang@cs.wm.edu.*