

Improving the performance of client Web object retrieval

Alexander P. Pons *

Computer Information Systems, University of Miami, 421 Jenkins Building, Coral Gables, FL 33146, USA

Received 25 April 2003; received in revised form 27 February 2004; accepted 29 February 2004

Available online 30 April 2004

Abstract

The growth of the Internet has generated Web pages that are rich in media and that incur significant rendering latency when accessed through slow communication channels. The technique of Web-object prefetching can potentially expedite the presentation of Web pages by utilizing the current Web page's view time to acquire the Web objects of likely future Web pages. The performance of the Web object prefetcher is contingent on the predictability of future Web pages and quickly determining which Web objects to prefetch during the limited view time interval of the current Web page. The proposed Markov–Knapsack method uses an approach that combines a Multi-Markov Web-application centric prefetch model with a Knapsack Web object selector to enhance Web page rendering performance. The Markov Web page model ascertains the most likely next Web page set based on the current Web page and the Web object Knapsack selector determines the premium Web objects to request from these Web pages. The results presented in the paper show that the proposed methods can be effective in improving a Web browser cache-hit percentage while significantly lowering Web page rendering latency.

© 2004 Elsevier Inc. All rights reserved.

Keywords: Web prefetching; Markov model; Knapsack selector; Web application

1. Introduction

The majority of the Internet population accesses the World Wide Web via dial-up modem connections. Studies have shown that the limited modem bandwidth is the main contributor to latency perceived by end users. Today's Web browsers employ a demand-fetch technique in which Web page objects (graphics, pictures, audio and/or video) are acquired only when the user initiates a request for a page. When a browser navigates to a Web Page, it checks to see whether the page and its compositional objects are in the browser's cache. If not, these objects must then be retrieved from the origin or proxy server. Since browser caches sizes are limited and maintain the most recently accessed objects, it is likely that the current page's objects will have to be requested from the origin server. This demand-fetching approach typically increases a Web page's rendering latency,

depending on the number and sizes of these objects and the speed of the communication channel.

The concept of Web object prefetching has the potential to expedite Web page rendering and increase utilize the communication channel, since the process makes use of the bandwidth that would otherwise be idle. The operation of object prefetching is complementary to the technique of browser object caching. For example, if a browser requests an object that is not in the cache, but the object is in the prefetch area, the browser avoids having to make a server request. Anticipating which objects will be referenced in the near future has been the focus of much recent research. In Pons (2002), we introduced a Multi-Markov Web application centric prefetching approach. In this paper we take our basic model and augment it with an object selection technique that extends our previous work to further improve performance. We define and subsequently elaborate on the Markov–Knapsack Prefetcher (MKP), which has the following properties:

- An Initial Web-application Markov Model (IWAM) is generated and downloaded to a client's machine.

* Tel.: +1-305-284-1960; fax: +1-305-284-5161.

E-mail address: apons@miami.edu (A.P. Pons).

The IWAM is a complete Web-application hyperlink domain bounded model in which the hyperlinks are the nodes and the edges are the transition frequency.

- During subsequent Web-application access the IWAM is personalized into a Web-application Markov Model (WAM) for the client.
- An IWAM exists for each Web application and is customized to a client's Web-application access patterns.
- The WAM predicts based the most likely next Web page based on the currently viewed Web page and acquires its objects during the idle communication channel time.
- The Knapsack technique identifies from among these probable Web page objects which one should be retrieved to minimize latency.

The remainder of this paper is organized as follows: Section 2 reviews some important related work concerning Web prefetching techniques. The Web-application centric Markov–Knapsack prefetcher is discussed in Section 3, and an example that highlights the approach is illustrated in Section 4. In Section 5, the experiment design is introduced with performance comparison results explained in Section 6. Finally, Section 7 provides a conclusion and suggests future research directions.

2. Related work

In the literature many prefetching methods have been proposed to predict the user's next action, from subsequence matching that use past sequences of actions to identify a possible next action for the current sequence, to single and multi-step Markov and Markov-like models that associate the possible next actions as states with probabilities from the current state. Other approaches use utilize user characteristics such as bookmarks and history to determine the user's behavior, while some use Web page content and semantic link information to make predictions for what will be requested next. These proposed methods consist of the following works.

Padmanabhan and Mogul (1996), Bestavros (1995, 1996) and Albrecht et al. (1999) proposed server-initiated approaches where the Web server maintains a Markov model of page request interdependency. These approaches vary in the type of statistics and use of the model to make predictions. In Padmanabhan, when a client requests a page, the server sends along with the page the names of the most likely subsequent accessed pages, leaving the initiative for prefetching to the client. Using trace-driven simulations, they achieved a 36% reduction in network latency at a cost of 40% increase in network traffic. They constructed the Markov model

using the number of client page accesses to compute the weights. In Bestavros, a Time Markov model is used to present pages to the client based on the currently requested page. The model is constructed from the behavior patterns of the general user population and selecting the present page with the highest probability of being requested next. Bestavros' trace-driven simulations showed that with an increase of 10% in bandwidth, a 23% reduction in page miss rate is possible. Albrecht builds on the approach taken by Bestavros and constructs a hybrid prediction model, which combines four Markov models and uses a decision-theoretic model for presenting pages.

Markatos and Chronaki (1998) combines a server's knowledge of its most popular pages, a Top-10 list of pages with client access profiles. A client determines how much and when items from the list are prefetched. Page prefetching occurs off-line, with experimental results suggesting that it manages to prefetch up to 60% of future requests, with less than a 20% increase in traffic. Hine et al. (1998) also combines client and server information to perform page prefetching. They extend the work done by Bestavros and define various client-browsing modes based on the number of client accesses to a server within a single client browsing session to arrive at a prefetching scheme.

Fan et al. (1999) proposes a prediction algorithm based on the Prediction by Partial Matching (PPM) studied by Vitter and Krishnan (1996) that demonstrates a relationship between data compression (Mogul et al., 1997) and prediction. They construct an m -order Markov and consider a prediction depth of more than one. The model predicts not only the next page, but also which pages will be requested after that. The m -order predictor uses the context of the past m references to predict the next set of prefetch pages for the client. Results show that their technique reduces client latency up to 23.4%.

Cunha and Jaccoud (1997) use a prefetch model that focuses on the client being active in gathering usage information and making prefetch decisions. The client machine utilizes a Markov model with three types of links that indicate the manner in which objects were accessed. These links distinguish between objects accessed within a time window, objects embedded within other objects, and objects that are likely to be accessed close in time. Their approach uses a mathematical model that combines link categorization and a Markov model, which exceeds a hit rate of over 80% when the user's behavior fits the model. The experiments conducted by Dunchamp (1999) indicate that a collaborative effort between the client and server works best for accurate prefetching. The server uses a Markov model built from client data to dispense information to clients, allowing them to perform prefetching according to their own needs using different algorithms.

Some current research has focused on using the content of the current Web page to decide which links on the page to consider for prefetching. While the previous techniques look at past actions as a basis to predict the future, context-oriented techniques base their link selection on the hypertext characteristics of the current Web page. Since prefetching all of the links of the current page is often not viable Ibrahim and Xu (2000) describe a neural net approach where predictions are weighted according to what “anchor text” has already been clicked on the page. The link ranking of a page is determined as a score computed by an artificial neuron. Davison (2002) proposes predicting a user’s next action based on analysis of the content of the pages requested recently by the user. Predictions are made by analyzing the user’s perceived interest of the text in and around the hypertext anchors of recently requested Web pages. Zhuge (2003) proposes the use of semantic links with associated heuristics to determine the semantic association of the next Web pages based on the current Web page. This method provides a more precise approach, since it does not have to infer the significance of the links from its surrounding anchor text, but rather utilizes assigned link semantics. These approaches can make predictions of actions that have never been taken by the user and can potentially make predictions that reflect current user interests. Effectively evaluating the performance of these approaches is difficult at best, since their results are tightly coupled to the set of Web pages employed during their testing.

The underlining concept of these previous studies consists of attempting to predict a user’s next Web page, were their distinguishing characteristics embody specific operational details. The proposed Markov–Knapsack method uses an approach that combines a Multi-Markov Web-application centric prefetch model with a Knapsack Web object selector, which is distinct in the manner in which the prediction model builds, personalizes to individual clients and determines which objects to prefetch. In order to establish a minimum performance value, we use the simplest 1-history Markov model in our empirical analysis to focus on the concept of Web-application centric prefetching. In addition, we investigate the selection strategy of prefetching objects from a set of likely hyperlink Web pages. The proposed Knapsack object selector extends the classical 0–1 Knapsack problem and outperforms other object selection techniques in trace-driven simulations. The combination of these two components is shown to be an effective means of improving Web page rendering and using trace data measured at the client-side for trace-driven simulations. Another difference is that most models attempts to predict the next page for the general case; our model is for a bounded domain of pages and hyperlinks logically composing the structure of a Web application, which limits the possible choices of Web objects.

3. Web-application Markov–Knapsack prefetcher

Web applications that are associated with intranet or extranet have the most to benefit from prefetching, since their client populations and tendencies are better known than those of general Internet applications. Internet Web applications can still profit from prefetching for membership or community sites, since the amount of dynamic content is limited. According to these user populations, a clustering algorithm identifies IWAMs associated with different types of users. Therefore, a Web application has several IWAMs defined and based on a new user’s characteristics an IWAM can be selected and installed on their machine. During subsequent user accesses to the Web application the IWAM is personalized to more closely match the user’s access pattern, becoming a WAM. It is possible that a user machine can have several WAMs used during the access of different Web applications that support the proposed prefetching scheme. The prefetcher uses the WAM of each Web application to determine the next Web page set based on the currently viewed Web page. The Knapsack technique chooses and retrieves compositional objects that provide the greatest profit from among these Web pages. The profit of an object is computed from how often it appears in these Web pages and the likelihood of a hyperlink to that Web page.

3.1. Building and distributing IWAM

One type of Web application that has much to gain from our approach is Application Service Providers (ASP), such as document and office management applications. The application we selected for our study attempts to straddle the range of both static and dynamic content types. It consists of an Internet community that offers its members current and past information. The content of such a Web application is mostly static, but does contain client-specific dynamic content and unqualified dynamic content. Those Web applications that are member based, such as ASP and Communities, to name a couple, have the most to gain from our approach and form the basis to construct and issue IWAMs to new members. When a new Web application is brought online, several IWAMs are defined on a functional basis. As existing members access the Web application, their interactions are recorded and analyzed by a clustering algorithm of access patterns. Should a new pattern emerge that is dissimilar to existing IWAMs, a new IWAM is defined. This new IWAM will either augment the list of IWAMs or replace a least used IWAM, preventing unbounded production. Therefore, the production of IWAMs is an ongoing process that continues to evolve as members interact with the Web application. For non-member Web applications, an IWAM could be selected on the basis of

the client's initial access, extending the prefetcher functionality to a larger Internet population. Therefore, clients are given an IWAM upon access to the Web application, which is customized into a WAM on subsequent client Web-application accesses. Most Web applications can benefit from the proposed technique, though specific types of applications stand to benefit the most. All that is required is a downloadable client software component that accepts and maintains the WAM, and a server-side software component to construct and deploy new IWAMs to said clients.

3.2. Web page Markov model prefetcher

The prefetcher utilizes a Web page 1-history Markov model consisting of the all the Web pages of a Web application as its nodes. These nodes are connected by arcs associated with the probabilities of the user to hyperlink to Web page y when viewing the current Web page x . The following definitions and notations are used to define the Markov–Knapsack Prefetcher:

Definition 3.1. Let S be the set of all Web pages composing a Web application.

Definition 3.2. The user's average view time for Web page “ X ” is X_{avt} , where $X \in S$.

Definition 3.3. Let T_x be the set of all the Web pages that can be hyperlink from the current Web page x , where $x \in S$ and $T_x \subseteq S$.

Definition 3.4. $|A|$ is the number of set members in A .

Definition 3.5. F_x is the set of probabilities f_y associated with all possible hyperlink from Web page x to Web page y , where $y \in T_x$ and $0.0 < f_y \leq 1.0$.

Definition 3.6. O_y is the set of $o_{m,y(\text{size})}$ Web objects composing Web page y , $y \in T_x$, where m is the name of the object $1 \leq m \leq |O_y|$ and the size is in kilobytes.

Definition 3.7. Q_x set is a super set of O_y , $y \in T_x$ and x is the current Web page.

Definition 3.8. $P(F_x, Q_x)$ computes as $F_y * O_y$, $o_{ky(\text{size})(f_y)}$, $y \in T_x$ and $1 \leq k \leq |O_y|$, if there exists any object with the same name, aggregate their f_y producing one with that name.

Definition 3.9. I_x is the set of $o_{j(\text{size})(p)}$, according to $P(F_x, Q_x)$. It contains all the possible prefetch Web objects from Web page x , where j is the name of the object $1 \leq j \leq |I_x|$, size is in kilobytes and p is the profit of prefetching object o_j .

3.3. Web object Knapsack selector

The classical 0–1 Knapsack problem Martello and Toth (1990) is defined as having n items each with profit p_i and a weight w_i . The objective is to select a subset from among these items to maximize the profit, such that the total weight of the items selected is less than or equal to the capacity of the Knapsack c . Formally, the objective is

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^n p_j x_j \\ &\text{subject to} && \sum_{j=1}^n w_j x_j \leq c, x_j \in \{0, 1\}, \quad j = 1, \dots, n, \end{aligned}$$

where p_j is the profit of the item j , w_j is its weight and c is the Knapsack capacity.

The classical Knapsack is adapted to select from a set of Web objects with respect to the currently viewed Web page time, using these definitions:

Definition 4.1. The Knapsack capacity c is defined as X_{avt} in milliseconds.

Definition 4.2. Let wt_j be the time weight of a Web object, computed as the object size divided by the transfer rate of the communication channel in milliseconds.

Definition 4.3. The f_j of an object is its prefetch profit.

Definition 4.4. The delay set D_x is the conversion of set I_x of Web page x applying Definition 4.2, where the transfer delay time of each object is computed according to the channel transfer rate, such that the size subscript of I_x members are converted to milliseconds.

Definition 4.5. The prefetch objects for Web page x is specified in the set Pf_x composed of objects that meet the objective:

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^n f_j x_j \\ &\text{subject to} && \sum_{j=1}^n wt_j x_j \leq X_{\text{avt}}, \quad x_j \in \{0, 1\}, \quad j = 1, \dots, n. \end{aligned}$$

4. Web application prefetching example

The concepts covered in Section 3 are illustrated through an example Web application consisting of six Web pages and numerous composing Web objects. We first construct an IWAM from three Web-application users that demonstrate similar Web-application interest, which becomes a personalized WAM for each user during subsequent accesses. Then, we assume a currently

viewed Web page and determine from the WAM the next Web page probabilities in order to prefetch objects according to the Knapsack object selector.

The Web application is composed of a set of Web pages denoted as $S = \{A, B, C, D, E, F\}$ and each of these Web pages references a set of objects, indicated by $A(o_{1(3)}, o_{2(5)}, o_{3(1)})$ for Web page “A.” The subscripted objects $o_{1(3)}$, $o_{2(5)}$ and $o_{3(1)}$ are identified with a numerical value (object name), followed with a parenthesized number depicting the size of the object in kilobytes. In Fig. 1, the access patterns for User1, User2 and User3 comprise the sequence of Web page accesses for each user, indicating that Web page “A” is followed by a reference to Web page “B”, and so on.

Using these users’ reference, in Fig. 2 a Markov model is built that approximates the references with transition frequencies. All transitions from node X to node Y in the diagram are assigned a value that represents the fraction of transactions where a reference to X is followed by a reference to Y. In addition, each node is associated with an average view time (avt) value, which is recalculated on every Web-application hyperlink click.

Consider node “A” from the diagram ($T_A = \{B, C, D\}$) which has three transitions that commence from node “A”, with $F_A = \{0.30_B, 0.25_C, 0.45_D\}$ as hyperlink probabilities from Web page “A.” The pattern “AB”, “AC”, and “AD” occur 0.30%, 0.25%, and 0.45% of the time, respectively. Using one previous Web page reference to predict the next Web page reference, a user that re-accesses the Web application and follows the same hyperlink sequence, allows for the Markov model to predict the subsequent Web page. When the user views Web page “A”, the model would predict that Web page “B”, “C” or “D” would be the next referenced Web page and issue prefetch requests for the objects associated with each of these Web pages.

User 1 { A, B, C, E, F, A, B, C, A, ... }

User 2 { A, D, B, C, E, A, D, ... }

User 3 { A, D, B, E, F, A, ... }

Fig. 1. User transition probabilities.

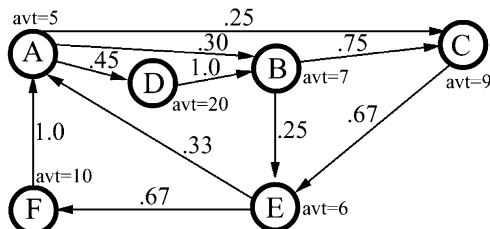


Fig. 2. 1-History Markov with average view time.

Assume that Web pages “B”, “C”, and “D” are composed of objects as $O_B = \{o_{1B(3)}, o_{2B(5)}, o_{3B(1)}\}$, $O_C = \{o_{1C(3)}, o_{4C(7)}, o_{3C(1)}, o_{5C(4)}\}$ and $O_D = \{o_{6D(15)}, o_{7D(20)}\}$. When the user views the current Web page “A” and the communication channel is available, the prefetcher will anticipate from the Markov model that, $Q_A = \{\{o_{1B(3)}, o_{2B(5)}, o_{3B(1)}\}, \{o_{1C(3)}, o_{4C(7)}, o_{3C(1)}, o_{5C(4)}\}, \{o_{6D(15)}, o_{7D(20)}\}\}$. Now, $P(F_A, Q_A) = \{o_{1B(3)}(0.30), o_{2B(5)}(0.30), o_{3B(1)}(0.30), o_{1C(3)}(0.25), o_{4C(7)}(0.25), o_{3C(1)}(0.25), o_{5C(4)}(0.25), o_{6D(15)}(0.45), o_{7D(20)}(0.45)\}$ generates an $I_A = \{o_{1(3)}(65), o_{2(5)}(30), o_{3(1)}(65), o_{4(7)}(25), o_{5(4)}(25), o_{6(15)}(45), o_{7(20)}(45)\}$ which are the objects to prefetch with respect to Web page “A”.

The Markov model generates the I_A set of Web Page “A”, which associates each object element with a prefetch profit. The I_A set maintains an object’s actual size, which must be converted to the object’s retrieval time, based on the client’s communication channel speed. Assume that a client is using a standard 28.8 k communication channel that transfers 3600 bytes/s and is accessing the Web application and spends 5 s viewing Web page “A” (A_{avt}). Then applying Definition 4.2 to I_A from Section 3.3, I_A is transformed into D_A , in which the object’s wt is calculated according to the communication channel speed, $o_{j(wt)(p)}$, $1 \leq j \leq |I_A|$, which is a set of candidate prefetch Web objects for Web Page “A”. The composition of the set D_A is $\{o_{1(834)}(65), o_{2(1389)}(30), o_{3(278)}(65), o_{4(1945)}(25), o_{5(1112)}(25), o_{6(4167)}(45), o_{7(5556)}(45)\}$.

According to Definition 4.5 the Pf_A of these objects consists of $\{o_1, o_2, o_3, o_5\}$, with a wasted capacity of 185 ms for an A_{avt} of 5000 ms. The Web Object Knapsack algorithm identified these objects to provide the greatest benefits within the time the client views the current Web page “A”, when anticipating a transition to one of its associated Web pages (B, C or D) based on the Markov predicative model of the Web application.

5. Markov–Knapsack analysis

Our prefetch analyses consist of evaluating the advantages of a Web-application centric scheme and appraise different alternatives for Web object selection. The goal is to determine which object selection technique achieves the highest cache-hit ratio. The approach undertaken requires the following activities:

- *Web-application identification.* A typical Web application was selected for the study normally encountered in e-business. The Web application consists of over 300 unique Web pages, nearly 6000 hyperlinks and in excess of 3000 unique objects. The Web application requires clients to be affiliated in order to access its capabilities. It consists of an Internet community that offers its members current and past information. The

content of such a Web application is mostly static, but does contain client-specific dynamic content and unqualified dynamic content.

- *Trace-data collection.* A client-side measurement applet was developed in Java and integrated into each Web-application page to record a client's Web page view sequence and the time spent on each Web page. The Java applet records the current page's Universal Resource Locator (URL), the start time, when the page is requested, and the end time when the next page is selected. Once the end time is recorded, the applet submits the URL and the time spent on the page in milliseconds to the Web server that stores it in an ongoing database for customer trace-data.
- *Trace-driven Web browser simulator.* A trace-driven Web browser simulator (Smith, 1994) was developed that uses the collected trace-data to reproduce client–Web application interactions. The trace-data is used to determine the number of demand-cache object misses encountered with and without prefetching. The simulator aggregates for each trace-run the number of total objects requests, the number of cache misses, the number of bytes transferred and the time required to transfer the objects from the origin server utilizing a 28.8K baud communication channel. At the start of each demand-fetching and prefetching trace-run simulation the cache is cleared to establish a comparative baseline. The simulator uses the time obtained from each trace-run record to determine the amount of time a client spent on the current Web page, providing the available time to retrieve a set of predicted objects. Therefore, the simulator utilizes a client's Web-application session behavior to construct the prefetcher's object retrieval constraints.

5.1. Results metrics

The study uses the metrics of coverage and accuracy to characterize the effectiveness of the Web-application centric Markov–Knapsack prefetcher. The coverage is the fraction of cache object misses compared with both demand-fetching and object prefetching. The demand-fetching coverage is a measure of the number of objects requested and found in the client's system cache. The prefetcher coverage encompasses objects in the client's cache and prefetcher area during a client's session with

the Web application. The accuracy is the fraction of the total prefetched objects that were actually used to satisfy object requests preventing cache misses. The accuracy of the prefetcher is a measure of the wasted communication channel bandwidth and memory storage to handle unused objects. Additional factors are evaluated, such as the effects on the number of bytes transferred and Web page rendering time.

5.2. Simulation study

Evaluating the effectiveness of Web-application centric object prefetching requires simulation results that represent a client's actual usage of the Web application. Therefore, trace-run data is divided on an IP basis with each trace-run issued to the simulator in the order in which the Web application was accessed. The simulation results for each IP are accrued into several Web-application frequency access categories. These categories are established for the number of unique trace-runs associated with each client; they divide the number of client Web-application accesses into six bins: (1) 1–10, (2) 11–20, (3) 21–30, (4) 31–40, (5) 41–50, and (6) more than 50. Results over 50 accesses per client produced negligible differences and are therefore limited to a maximum of 50 access sessions per client. For example, a client with 15 trace-runs is in the second category. The simulation measurements of these 15 trace-runs are accumulated into the category's demand-fetching and prefetching miss counts and are also used to evaluate the object selection technique. For the simulation of each trace-run we assume that the browser's cache no longer maintains the objects for the Web-application's pages, requiring origin server object requests, focusing on the prefetcher's performance.

The results in Table 1 compare the effects of utilizing a Markov prefetcher and demand-fetching. The values are grouped according to the number of trace-runs per IP, independent of the length of the individual trace-runs. Each client is given an IWAM, which is personalized during subsequent client Web-application accesses.

In addition, Tables 2 and 3 show the results of using a Markov prefetcher with different strategies for speculative object retrieval. These include acquiring objects in a greedy manner based on the object size (largest first),

Table 1
Web-application client frequency access

Bins (IP trace-runs)	Hit increase (%)	Byte transfer decrease	Time saving (s)	Accuracy (%)
1–10	26.74	175656	32.72	77.94
11–20	63.96	223432	62.06	82.21
21–30	89.47	230234	61.98	99.16
31–40	90.72	214831	59.68	99.38
41–50	94.12	237196	65.89	99.82

Table 2
Hit increase percentage

Bins (IP trace-runs)	Size (%)	Priority (%)	Priority/size (%)	Knapsack (%)
1–10	24.29	42.98	27.50	95.99
11–20	50.00	70.69	57.19	99.10
21–30	89.47	94.12	89.47	100.00
31–40	84.89	86.28	84.89	100.00
41–50	77.43	88.19	83.71	100.00

Table 3
Accuracy percentage

Bins (IP trace-runs)	Size (%)	Priority (%)	Priority/size (%)	Knapsack (%)
1–10	79.7	81.9	77.8	75.0
11–20	98.7	98.8	98.6	98.6
21–30	99.7	99.5	99.4	98.7
31–40	99.9	99.9	99.9	99.8
41–50	99.9	99.8	99.8	99.7

priority (all objects of the highest predicted Web pages first), priority/size (sort objects on priority and then a minor sort on size) and Knapsack (using object profits) as defined in Section 3.3.

6. Performance comparison

The benefits of a Web-application centric Markov prefetcher compared to demand-fetching of Web objects were investigated. The observed results reflect the advantages of prefetching approach over demand-fetching in all categories. These values were attributed to the personalization of the IWAM during subsequent client visits. The results in Table 1 are grouped according to the number of trace-runs per IP. The cache-hit percent values increase as the client frequents the Web application, since the IWAM becomes further representative of the client’s behavior.

A client that accesses the Web application between 1 and 10 times experiences an increased cache-hit of 27%, with an average reduction of demand byte prefetching of 175K and an average latency saving of 32.72 s per visit. The amplified IWAM personalization of the higher category ranges is evident from the increased cache-hits percentage, since a greater number of prefetched objects are correctly predicted. The cache-hit percentage reaches about 90% for clients that revisit the Web application above 21 times. The decrease in the average byte transfer and average time is produced through the aggregation of various length trace-runs per client visit. These values reveal the possible gains in Web page rendering associated with the frequency of client visits, unrelated to the number of pages viewed per visit. An important value is the accuracy percentage, which is high for all categories, but in particular for client Web-application accesses above 20 times. These high values are attributed to the benefits of increasing the accuracy of matching a client’s Web-application access patterns on an ongoing basis.

In addition, speculative object selection strategies coupled with the functionality of the Markov prefetcher have been investigated. The observed results in Tables 1 and 2 are presented in Figs. 3 and 4, respectively.

From Fig. 2, the Knapsack object selection technique outperforms the other methods of identifying objects to prefetch. The Knapsack’s higher cache-hit percentage is attributed to the manner in which an object’s profit is computed and the modifications associated with the Knapsack’s objective function. An object’s profit with

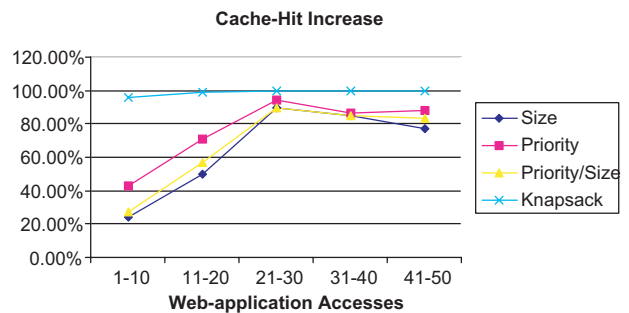


Fig. 3. Prefetch cache-hit percentage.

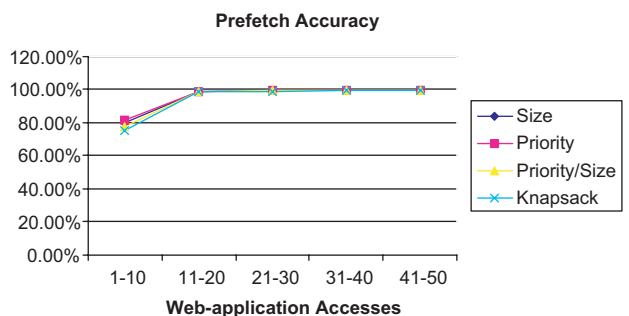


Fig. 4. Prefetch object accuracy.

respect to the currently viewed Web page is determined based on the probability frequency of the most likely hyperlink Web pages and the object's occurrences in these Web pages. This method of object selection enhances the Markov predictive model in increasing its object prefetch performance. All object selection methods are similar in their accuracy, with the accuracy improving for higher Web-application client accesses, since the Markov predictor converges closer to the client's behavior. The marginally lower Knapsack accuracy is due to the selection of additional, smaller unused objects that are fetched that fill available Knapsack capacity once all essential objects have been identified. These results indicate the potential improvement to Web applications utilizing a Web-application centric prefetching approach coupled with a Knapsack method for object selection.

7. Conclusion

The World-Wide-Web has established the infrastructure to deploy Web applications that provide a myriad of capabilities and services. An underlying feature of these sophisticated Web applications is the abundance of media objects that enhance the use of these Web applications. On the downside, incorporating extensive amounts of media objects (more and larger object retrievals) on a Web page can potentially prolong the rendering latency associated with that Web page. This will adversely impact the effectiveness of the Web application when accessed over slow communication channels.

Prefetching techniques have demonstrated their effectiveness in reducing the latency associated with Web page rendering. We have defined a Web-application centric prefetching approach that improves performance by anticipating a client's hyperlink action. Our technique incorporates a Web-application centric Markov model with a Knapsack object selector that adapts to a client's Web-application access pattern. The main emphases of the technique are the definition of a Web-application specific Markov model that restricts the hyperlink domain of Web pages to the Web application, and the adaptive nature of the model to more accurately represent a client's behavior. In addition, the Knapsack object selector supplements the prefetch technique in identifying objects that provide the greatest profit, consequently increasing the object cache-hit percentage and prefetch accuracy. Trace-driven simulations have shown that the combination of these components is an effective means of reducing the latency encountered during Web page rendering. The MKP approach realizes these benefits through maintaining a number of personalized Markov prefetcher models for various compliant Web applications. Despite the MKP's per-

formance, improving its accuracy is an area of ongoing research. The focus is on the generation of an IWAM that is representative of the client's access pattern, using characteristics of the client (age, gender, interest, etc.) to select from among a set of similar interest IWAMs. The cooperation of Web applications and browsers in prefetching Web objects has the potential to improve a client's quality-of-service by reducing Web page rendering time as empirical results encourage adoption of the MKP approach.

References

- Albrecht, D.W., Zukerman, I., Nicholson, A.E., 1999. Pre-sending documents on the WWW: a comparative study. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI-99, Stockholm, Sweden, vol. 2. Morgan Kaufmann, Los Altos, CA, pp. 1274–1279.
- Bestavros, A., 1995. Using speculation to reduce server load and service time on the WWW. In: Proceedings of the 4th ACM International Conference on Information and Knowledge Management, CIKM'95, Baltimore, MD.
- Bestavros, A., 1996. Speculative data dissemination and service to reduce server load, network traffic and service time in distributed information systems. In: Proceedings of the International Conference on Data Engineering, ICDE'96.
- Cunha, C.R., Jaccoud, C.F.B., 1997. Determining WWW User's next access and its application to pre-fetching. In: Proceedings of the 2nd IEEE International Symposium on Computers and Communications, ISCC'97.
- Davison, B.D., 2002. Predicting Web actions from HTML content. In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia (HT'02), June 11–15, College Park, MD, pp. 159–168.
- Dunchamp, D., 1999. Prefetching hyperlinks. In: Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, USITS'99.
- Fan, L., et al., 1999. Web prefetching between low-bandwidth clients and proxies: potential and performance. In: Proceedings of the ACM SIGMET-RICS Conference.
- Hine, J., Wills, C., Martel, A., Sommers, J., 1998. Combining client knowledge and resource dependencies for improved World Wide Web performance. In: Proceedings of the 8th Annual Conference of the Internet Society, INET'98, Geneva, Switzerland.
- Ibrahim, T.I., Xu, C., 2000. Neural net based pre-fetching to tolerate WWW latency. In: Proceedings of the 20th International Conference on Distributed Computing Systems, ICDCS2000.
- Markatos, E., Chronaki, C., 1998. A Top-10 approach to prefetching the Web. In: Proceedings of the 8th Annual Conference of the Internet Society, INET'98, Geneva, Switzerland.
- Martello, S., Toth, P., 1990. Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester.
- Mogul, J. et al., 1997. Potential benefits of delta encoding and data compression for HTTP. In: Proceedings of ACM SIGCOMM, August, pp. 181–194.
- Padmanabhan, V., Mogul, J., 1996. Using predictive prefetching to improve World Wide Web latency. In: Proceedings of the ACM SIGCOMM'96 Conference on Communications Architectures and Protocols, July, pp. 22–36.
- Pons, A., 2002. Web-application centric object prefetching. *Journal of Software and Systems* 67 (3).
- Smith, A., 1994. Trace driven simulation in research on computer architecture and operating systems. In: Proceedings of New

Directions in Simulation for Manufacturing and Communications Conference.

Vitter, S., Krishnan, P., 1996. Optimal prefetching via data compression. *Journal of the ACM* 143 (5), 121–130.

Zhuge, H., 2003. Active e-Document framework ADF: model and platform. *Information and Management* 41, 87–97.

Alexander P. Pons is an Assistant Professor of Computer Information Systems at the University of Miami. He has a Ph.D. in Computer Engineering with extensive industry experience. His research interests include database design, object-oriented modeling, real-time systems and Internet technologies.