

Determining WWW User's Next Access and Its Application to Pre-fetching*

Carlos R. Cunha[†]
Computer Science Department
Boston University
111 Cummington St
Boston, MA 02215, USA
carro@cs.bu.edu

Carlos F. B. Jaccoud
Division of Quality Control
Embratel
R. Senador Pompeu 27
Rio de Janeiro - RJ - Brazil
jaccoud@embratel.com.br

March 26, 1997

Abstract

World-Wide Web (WWW) services have grown to levels where significant delays are expected to happen. Techniques like pre-fetching are likely to help users to personalize their needs, reducing their waiting times. However, pre-fetching is only effective if the right documents are identified and if user's move is correctly predicted. Otherwise, pre-fetching will only waste bandwidth. Therefore, it is productive to determine whether a revisit will occur or not, before starting pre-fetching.

In this paper we develop two user models that help determining user's next move. One model uses Random Walk approximation and the other is based on Digital Signal Processing techniques. We also give hints on how to use such models with a simple pre-fetching technique that we are developing.

1 Introduction

User traces have shown applicability in the understanding of characteristics of network connections and Web file systems, as reported in [CBC95, CB95]. But, they can also

*This is an extended version of the article with the same title presented in the *International Symposium on Computers and Communication'97*, Alexandria, Egypt, 1-3 July, 1997.

[†]Partially sponsored by CNPq.

answer questions about user's behavior. For instance, a user's inter-request time, which corresponds to the user's *thinking time*, is modeled as a Pareto distribution [Den96], what has implications in network capacity dimensioning.

User's behavior also plays an important role in system performance. Users expect a prompt response from systems. The Web became a complex system, with a vast variety of available information. Also, the number of Web users grows everyday through on-line service providers, like America Online, Prodigy, Compuserve, etc. This growing load imposes an increasing demand on networks and servers, which have limited capacity. This conflicting situation requires some individualized solution in a *speculative* fashion [BC95]. One possible way to reduce the noticed response time is to anticipate user's requests and to order highly likely objects accordingly. This is known in the literature as *pre-fetching*.

The first problem with pre-fetching is to predict what users will need. A good hint is user's past history. If we capture noticeable access patterns from it, we will be able to predict the objects to be accessed with certain confidence. Although simple analysis of user's traces reveals enough information about which documents are the most probable to be accessed, pre-fetching based only on the observed access patterns may overburden the network even more than it already is. Therefore, we also have to predict what the next access will be, i.e., whether the user will access a previously visited document or move to a new object. If these actions can be devised, an efficient pre-fetching technique may be designed [Cun97].

In this paper, we present two user models. The first, based on *Random Walk* [Pap91] and its similarities to software caching, is intended for long range predictions. The second model is derived by directly applying *Digital Signal Processing* techniques [RS78] to the curve describing a user's access behavior. As we are going to see, the models introduced here capture a significant part of the user's access patterns (85% on average).

2 Data Collection

Two years ago, the *Oceans* research group¹ at Boston University started to study the World-Wide Web [BLCcGP92]. As part of this research, a database of user traces was established [CBC95]. This database constitutes a rich source of material to study user's behavior and network traffic.

A series of studies was performed concerning network traffic [BCC⁺95, CB95, CBC95]. However, the same cannot be said in relation to user's profile determination. For instance, [CP95] characterizes browsing strategies by looking at the longest common access sequences. In [Cun97], we show how trace analysis can play a significant role in enhancing system performance, both on client and server sides.

The data we use here was obtained between November/94 to May/95 in the Boston University Computer Science Department laboratories. The users were students employing a modified version [CBC95] of Mosaic V2.4 [NCS] to collect information about which documents were accessed, when they were accessed, and how long each object took to be fetched.

3 User's Profiles

Users were involved in multiple sessions. Each session contained multiple requests, either to distinct or to repeated URLs². For each user, we arranged the sessions and requests in chronological order and created a mapping between the URLs and some unique id. The chronological plotting of reference versus id constitutes the user profiles.

Depending on the navigation strategy, the user can either return to the same objects over and over, or always visit a new object. These two extreme types of user's behavior have their profile diagram represented in Figure 1. The left diagram is due

¹<http://www.cs.bu.edu/groups/oceans/Home.html>

²URL = Uniform Resource Locator.

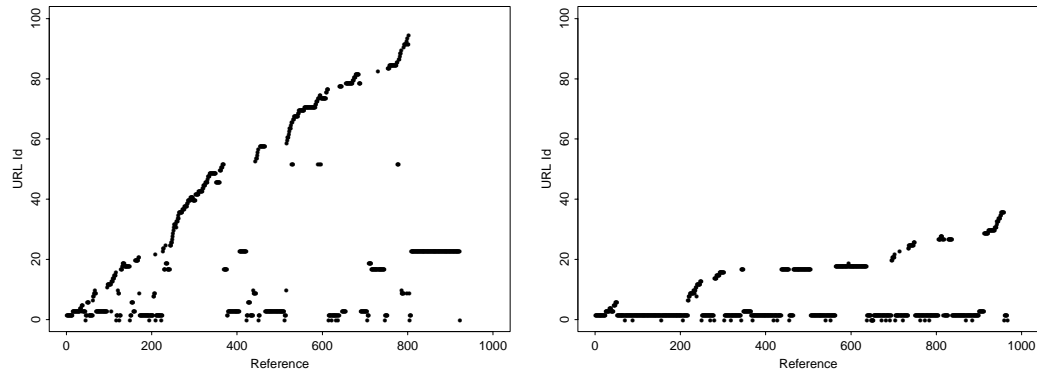


Figure 1: User profiles.

to a user who is more interested in exploring the cyberspace than to explore what the document can offer him. Let us call this user a “*net surfer*.” The user on the right is more concerned with exploring the contents of the objects in a certain site. (S)He has lots of repeated references, identified in the diagram by horizontal lines. Let us call this user “*conservative*”.

These two users pose different challenges to pre-fetching procedures. On one hand, it is reasonably easy to guess which documents a conservative user will access next. A simple pre-fetching model as the one described in Section 5 is enough to capture the accessing probabilities. The pre-fetching tables are well-behaved, i.e., the transitions are smooth and as (s)he repeatedly revisits documents, the prediction is easier. Furthermore, the extra bandwidth necessary to get some improvement in the response time is well paid off. On the other hand, the net surfer does not have a transition table which is able to capture any particular standard behavior and all documents have equal probability of being accessed. Consequently, the price to be paid in terms of extra bandwidth is too high. Therefore, it is advantageous to detect the current user’s strategy, and stop (or not) pre-fetching depending on it.

4 User Characteristics

As we noted in Section 3, user's profile performs an important role in the effectiveness of pre-fetching algorithms. In particular, diagrams such as the ones in Figure 1 can supply information necessary to make a decision when applying pre-fetching is worth, and when it is not. In this section, we construct two empirical user models exploring user profiles.

4.1 Random Walk User Model

To a certain extent, a parallel can be drawn between user's new requests and program misses in a cache. A program miss requires the CPU to fetch the new set of instructions from the main memory. Similarly, a new request obliges the browser to fetch the document from the server instead of from the cache. Consequently, if we understand the nature of program cache misses, we will be able to use similar ideas to predict when a new request will occur and thus start/stop pre-fetching. In both cases, we consider an infinite size cache for the sake of analysis.

Voldman *et al.* [VMH⁺83] noticed that program intermiss distances follow a hyperbolic distribution, characterizing a fractal behavior. Their analysis shows that the distribution of the distances characterizes its behavior, following the property [Man83]:

$$Pr[U > u] = (u/u_0)^{-\theta},$$

where U is the random variable describing the distances, u_0 is a constant, and θ is the fractal dimension. The value of θ indicates what is expected from the walk. If $\theta > 1$, the walk constantly visits new positions. This case is called *transient*. If $\theta < 1$, the walk is *recurrent*, revisiting many of the already accessed positions.

A good reference on the study of the nature of cache misses is Thiébaud's paper [Thi89]. In this paper, Thiébaud constructs a cache model based on that by Gillis and Weiss [GW70] for the accumulated number of unique cells visited in an r -step walk.

In Thiébaud's model, the accumulated number of cache misses corresponds to the program's random walk range. Thus, the fractal model helps explain the asymptotic number of accumulated cache misses in an infinite cache as a function of the number of references made by a program. The expression explaining such a relationship is:

$$N(r) = A r^{1/\theta}, r \gg 1, \theta \geq 1$$

where r is the the number of references, $N(r)$ is the accumulated number of misses, θ greater than one³ sets the curve growth pace, and A is a constant. An example of one such curve is given in Figure 2.

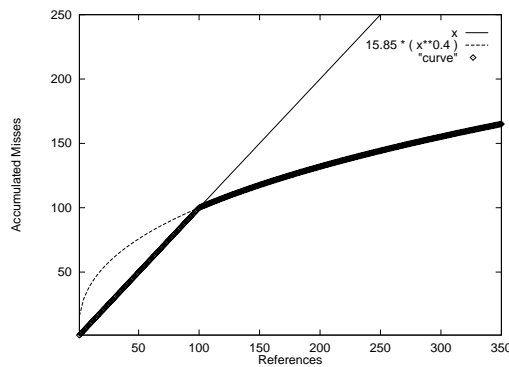


Figure 2: Program cache misses behavior.

We can interpret the envelope of each plot in Figure 1 as the number of accumulated misses in an infinite browser's cache. Therefore, we can use Thiébaud's model to characterize user's strategies. Figure 3 shows such envelopes (heavy dotted lines). Also in this figure, we see the least square fits performed on a log-log plot, assuming such a random walk model. The models (the light solid lines) for these two users are: $N(r) = 1.0 \cdot r^{0.8}$, with $R^2 = 0.995$, and $N(r) = 1.6 \cdot r^{0.6}$, with $R^2 = 0.9767$, respectively. Indeed, for the majority of the tested users, this model applied quite well.

³In the extreme case, every access is to a new position.

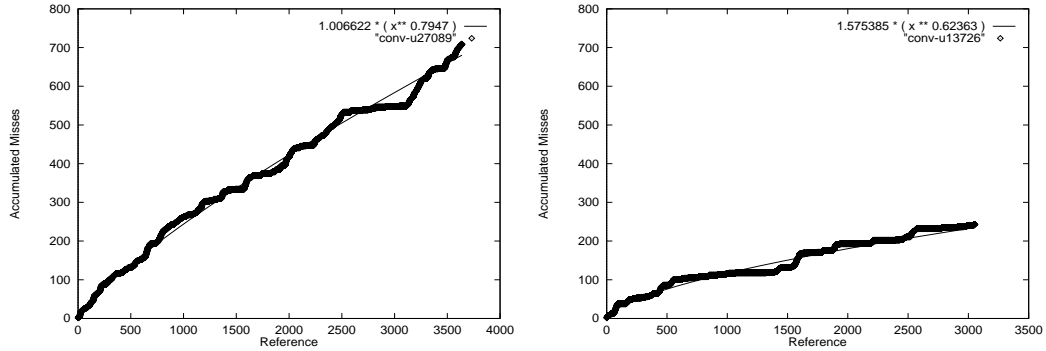


Figure 3: User accumulated number of new locations.

An interesting application of the model is the capture the behavior within a window. This local behavior reveals the user's current strategy. Figure 4 shows a particular user who displays various behaviors. For the period between 1000 to 2200 references, for instance, this user progresses in a very slow fashion. For the period between 2800 to 3900, on the other hand, the user becomes very aggressive and the curve climbs very steeply.

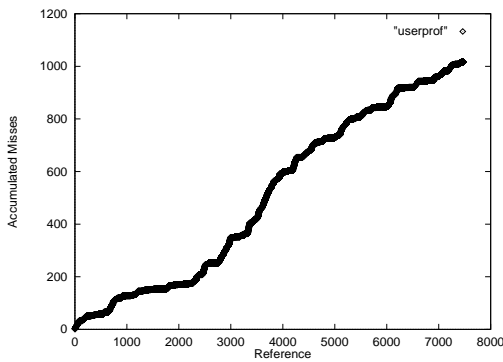


Figure 4: User accumulated number of new locations.

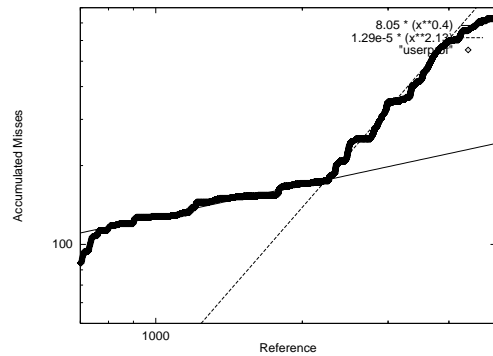


Figure 5: Fitting of user behavior.

In this short term analysis θ may be smaller than one. Indeed, the least square

expressions that best describe the behavior in these two regions are: $N(r) = 8.05 * r^{0.4}$ and $N(r) = 1.29 * 10^{-5} * r^{2.13}$ (See Figure 5). In these two cases, the values of θ are 2.5 and 0.47, respectively. That means, in the first case, the walk is recurrent and the user tends to revisit locations. In the second case, the walk is transient, i.e., the user is in an exploratory mode.

4.2 DSP User Model

Although the model described in the previous section provides a good approximation for user behavior in the long term, we would like to have a more accurate method to predict the next user access based on his past history. *Time-series analysis* and *digital signal processing* (DSP) techniques are candidates for this prediction.

Voldman and Hoevel [VH81] studied the software-cache connection using DSP techniques. In particular, they were interested in long-range prediction of cache misses. They also intended to show that software related events are responsible for fluctuations in the buffer miss ratio. In this paper, Voldman and Hoevel used *spectral analysis* to identify loops and cache line sizes.

Motivated by Voldman and Hoevel's idea of using DSP to analyze the cache miss rate, we decided to use other DSP techniques to build an alternative model to predict user's access behavior. First, we notice that the important feature in the original signal $N(r)$ is whether a miss occurred or not. Therefore, the first order difference, $N'(r)$, is important because it carries this feature more explicitly, and transmits the same information as $N(r)$:

$$N'(r) = N(r) - N(r - 1).$$

It is easy to see that

$$N'(r) = \begin{cases} 1, & \text{if } N(r) \text{ was a miss} \\ 0, & \text{if } N(r) \text{ was a hit.} \end{cases}$$

In other words, $N'(r)$ is equal to 1 if the user accesses a different URL and it is equal to 0 if (s)he accesses a previously visited URLs.

We need to detect changes in user's behavior trends. These trend changes are represented by a sequence of pulses 1 or 0. A sequence of 1's indicates the user is surfing. A sequence of 0's translates a conservative mode. The longer the sequence is, the stronger is the new trend. This is detected by calculating the density of 1-pulses within a given interval (window) over $N'(r)$. Figure 6 shows the results for a window of 20 past accesses for the curve in Figure 4.

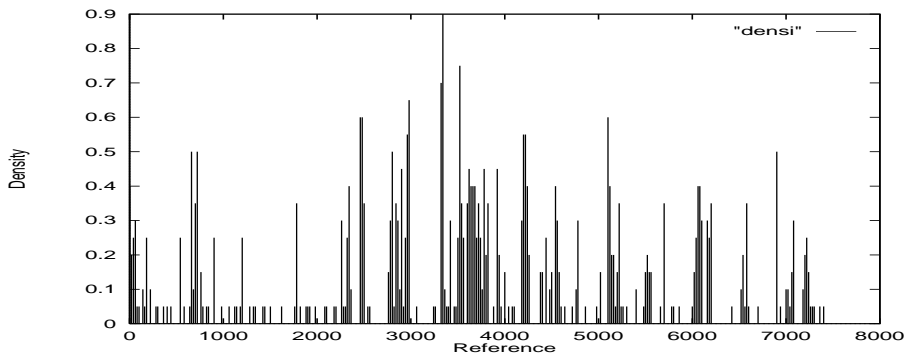


Figure 6: Pulse density for a window of 20 references.

Comparing Figures 4 and 6 we see that the points that we have mentioned as starting points of a new behavior produce very high peaks in the density graph. That is, high peaks one indicates a change of behavior. Of course, there are other secondary peaks that can lead to a false conclusion if we consider the whole graph. It is necessary to set a threshold to eliminate false detection.

To conform the signal $N'(r)$ to a suitable processing form by the specialized predicting procedures, a small displacement by 0.5 is performed. With values between -0.5 and 0.5 it is easy to detect a change in the behavior: $N'(r) + N'(r - 1) = 0$, if such a change occurs.

If we describe the time series $N'(r)$ as an auto-regressive (AR) process, we say

that

$$N'(r) = \sum_{k=1}^p \alpha_k N'(r-k) + u(r)$$

is the p^{th} order expression for the process, where $u(r)$ is a white noise factor with zero mean. Therefore, we may assume that the following *forward linear predictor* [SLM87] exists with coefficients α_k and output $\tilde{N}'(r)$:

$$\tilde{N}'(r) = \sum_{k=1}^p \alpha_k N'(r-k).$$

Thus, the prediction error at virtual time r , $e(r)$, is:

$$e(r) = N'(r) - \tilde{N}'(r) = N'(r) - \sum_{k=1}^p \alpha_k N'(r-k)$$

Because user's behavior changes, the coefficients are not constant through the whole sequence. They vary as a function of the virtual time r . Consequently, the problem is to find a set of coefficients $\{\alpha_k\}$ which minimizes the short-time prediction error, E_r , around a vicinity of size n for a sample at virtual time r . E_r is given by [RS78]:

$$\begin{aligned} E_r &= \sum_m e_r^2(m) & (1) \\ &= \sum_m \left(N'_r(m) - \sum_{k=1}^p \alpha_k N'_r(m-k) \right)^2 \end{aligned}$$

The summation ranges in the above equations were left open since they are dependent on the current user's behavior. If the user is in the conservative mode, not many samples are necessary to predict his(her) behavior. On the other hand, if the user is in the surfing mode, we need more history to correctly predict his(her) next move. Therefore, we need to combine the density of pulses with the above prediction procedure to determine the ranges of the summations.

The solution for $\{\alpha_k\}$ in Equation 1 is known as the determination of the *linear prediction coefficients* (LPC) in the DSP jargon. The solution of these system of equations has a efficient method: the *Durbin's Method* [RS78]. This method is iterative, approximating the coefficients in each step. One characteristic of this method is that the solution for step i are the coefficients for the i^{th} polynomial approximation.

We have to determine the minimum number of coefficients necessary to give a good approximation. We also need to establish a measure that indicates how good our approximation is. Let

$$D(r) = N'(r) - \tilde{N}'(r)$$

be the difference between the first order differences of the actual signal and the predicted signal. This difference is zero either when both signals are congruent, or when they are parallel. If the signals are congruent, we guessed correctly the user's action. If the signals are parallel, there was an error somewhere in the past, but the prediction caught up and is following the original signal again. In either case, $D(r) = 0$ is an indication that the guess was correct.

Now, we compute the percentage of zeros, Υ , in $D(r)$ from the first sample to the virtual time r . The higher Υ is, the better is the quality of the prediction. Hence, Υ can be used as a metric for determining the number of coefficients. The smaller the number of coefficients the better because less computation is required.

Figure 7 describes the algorithm used to predict the user's next action. It combines the ideas and observations above. The algorithm is divided into two phases: *preparation* and *prediction* phases. The preparation phase computes the first order difference of the envelope of the user's profile curve, displaced by 0.5 as aforementioned.

The prediction phase has four steps. Initially, we determine in the last t accesses how conservative the user was. This is done by counting the number of zero crosses existing in the displaced curve SEQ , $ntrans$. The second step gets this count and determines how much history, $base$, will be used to compute the desired coefficients. The next step calls a routine based on the Durbin's method to calculate the LPC coefficients. Finally, the predicted value is computed as a linear combination of the past $NCOEF$ terms.

Figure 8 compares the user's profile of Figure 4 and the output of the above algorithm with four coefficients and an analysis window of the last $t = 10$ accesses. As we can see the difference between the two curves is almost indistinguishable. In

1. Preparation Phase.

- (a) Get user's history with n references. Transform each URL in an unique id.
- (b) Generate a chronological sequence, call it SEQ , the following way:

```

for each URL  $i$  in chronological order
  if URL  $i$  appeared before then  $SEQ(i) \leftarrow -0.5$ 
  if URL  $i$  is new then  $SEQ(i) \leftarrow +0.5$ 

```

2. Prediction Phase.

- (a) Compute number of the last t transitions.

```

 $ntrans \leftarrow 0$ 
for  $i \leftarrow n - t$  to  $n$ 
  if (  $SEQ(i) + SEQ(i - 1) == 0$  )
    then  $ntrans \leftarrow ntrans + 1$ 

```

- (b) Use a proportional history.

```

 $back \leftarrow \lfloor \frac{ntrans+1}{2} \rfloor \cdot 2 \cdot t$ 
 $base \leftarrow SEQ(n - back : n)$ 

```

- (c) Compute the LPC coefficients.

```

 $coefs \leftarrow LPC(base, back, NCOEF)$ 

```

- (d) Compute predicted value.

```

for  $i \leftarrow 1$  to  $NCOEF$ 
   $predict \leftarrow predict - coefs(i) \cdot SEQ(n - i)$ 

```

Figure 7: Prediction Algorithm.

fact, Υ for this experiment was 82.63%.

We experimented with a number of users varying the number of coefficients and the analysis window t to verify the effect of these two parameters. We judged the quality of the prediction based on the Υ that each combination produced. Table 1 reports the results for the same user whose profile is in Figure 4. Surprisingly, a small number of coefficients, four, is enough to issue a high value of Υ . The analysis time window is dependent of the number of coefficients, but a default value equals to the number of coefficient seems to be enough.

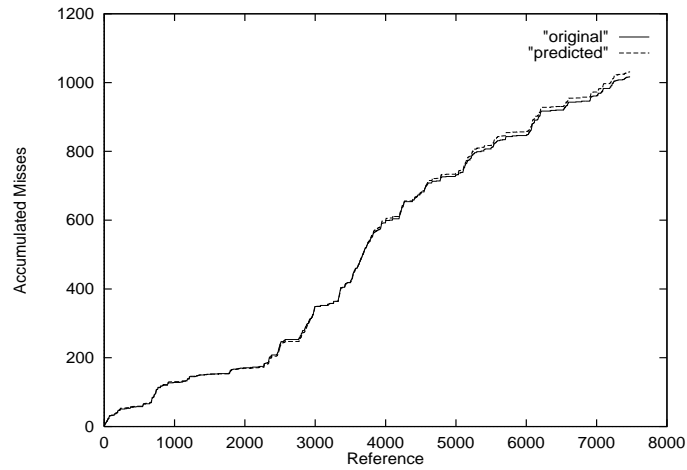


Figure 8: Comparison of an actual user’s profile and a predicted profile with four coefficients and window analysis of last 10 accesses.

For a mix of users, selected randomly from the users with the most number of sessions/references, we passed their profiles through the above procedure and the results were encouraging: Υ ranges from 80% to 95%.

Number of coefficients	Analysis Time Window									
	1	2	3	4	5	6	7	8	9	10
1	81.67	81.75	81.67	81.79	81.79	81.89	81.84	82.00	82.01	82.07
2		84.85	84.50	84.73	84.61	84.94	85.02	85.08	85.14	85.17
3			87.98	86.89	86.75	86.28	86.27	86.16	86.40	86.28
4				88.75	89.42	89.44	89.48	89.41	89.48	89.58
5					84.84	84.41	84.42	84.61	84.62	84.72
6						82.98	82.96	83.12	82.95	83.11
7							83.39	83.27	83.10	83.22
8								82.42	82.32	82.13
9									82.59	82.31
10										81.60

Table 1: Prediction quality, Υ (%), as a function of the number of coefficients and of the analysis time window.

5 Pre-fetching Method

Pre-fetching is a standard technique that has been used in many other areas, which requires some knowledge of what is to be pre-fetched. The problem is deciding what to pre-fetch. We succinctly describe here our pre-fetching method.

5.1 Model Description

Our method constructs relationships between documents according to the transitions the user made while navigating the Web. The transitions may be viewed as a graph. An example of such graph is shown in Figure 9, where each D_i , for $i = 1, 2, 3, 4, 5$, is an unique document. Two steps are performed in the construction process. First, related documents are identified. Second, the relationship weight is determined.

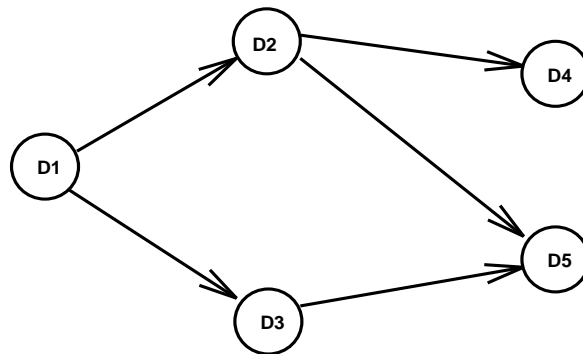


Figure 9: Example of a transition network.

In the construction process of the pre-fetch table, which contains the relationship between objects, the transitions between objects are separated in three categories: *embedded objects*, *traversal objects*, and *not related objects*. The embedded objects are determined by scanning the base object and extracting its components. Embedded objects are always pre-fetched. Traversal objects are determined by setting a *reasonable* threshold, empirically derived from trace observation. Finally, not related

objects are all the other that have transition time bigger than the used threshold. These categories are illustrated in Figure 10.

The need of establishing a *reasonable* threshold to accept the documents as related can be justified as follows. We could consider all documents accessed by the user, constructing a huge graph. However, such a graph would contain more than the user preferences, and the number of pre-fetched objects would be large, degrading the technique efficiency (see [Cun97]).

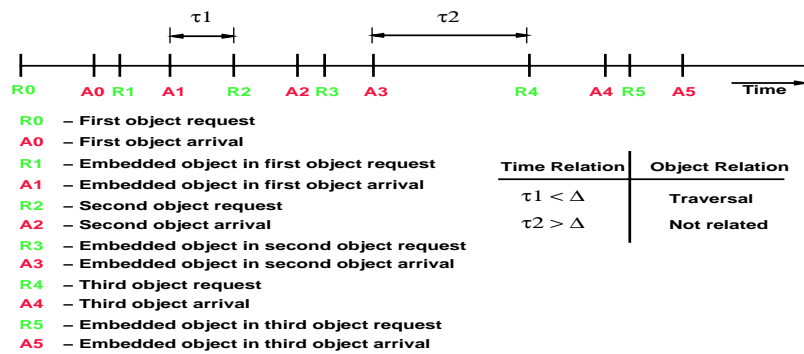


Figure 10: Relationships categorization.

With this procedure, the individualized pre-fetch table is updated after each user session, with the session's history guiding the construction/update. The number of steps ahead may be set as one argument of the procedure.

6 Using User Model in Pre-fetching

As we mentioned above pre-fetching is a speculative process. If the guess is correct, a score is made. If the guess is wrong, a high price is paid. Hence, we want to be as accurate as possible, suspending pre-fetching if there is any chance that the user will not revisit a document.

Therefore, we suggest that, coupled with the pre-fetching procedure, we have a

routine that will guess the user's next move. This guess may be done by either method. With the first method, compute the least square fit and determine the parameters A and θ . If $\theta > 2$, stop pre-fetching. If $\theta < 2$, start pre-fetching.

For the second method, the same general principle applies. Before a pre-fetching request is issued, verify with the oracle if the user will revisit a document. If a positive answer comes out, then issue the pre-fetching request. This oracle works as the procedure described above.

7 Related Work

Pre-fetching has a long list of references [Smi82, AK94, LS84, PZ91, Kri95, Mar96, Bes95, Pad95]. Each of these references deals with a different situation. Krishnan's approach [Kri95] is the closest to ours, except that he considers all transitions between pages accessed in a database environment.

Markatos [Mar96], Bestavros [Bes95], and Padmanabhan [Pad95] consider pre-fetching from the server side. The server is able to provide good hints about what other user visit in the site, this only gives the user a localized idea, but not a global idea.

Catledge and Pitkow [CP95] analyze client logs with the view of user's interface. They categorize users according to size of their common sequences. The categorization is based on the slope of the plot of common sequence size against sequence frequency.

Bolot and Hoschka [BH96] analyze server's log using seasonal ARIMA. They targeted the number of connections that their web server receives, with the intention of dimensioning the work a web site would receive. Their predictions are within 10% to 30% range of error.

8 Conclusion and Future Work

In this paper, we saw two user models suitable to work in conjunction with pre-fetching techniques. The first model using Random Walk approximation captures the long term trend. The second model applies to the short term behavior. Both models are able to track user's behaviors.

We are now in the process of incorporating the models inside the pre-fetching method briefly presented in Section 5 (for more details, see [Cun97]). We expect that we will be able to reduce the extra bandwidth used by pre-fetching and still obtain good levels of latency reduction. We also expect that this new pre-fetching procedure will achieve this objective by issuing requests only when a document revisit is guessed to occur.

As we have seen, the two models have a high degree of accuracy (around 85%). Our next step is to verify which model presents a better prediction when coupled with pre-fetching techniques.

References

- [AK94] Thomas Alexander and Gershon Kedem. Design and Evaluation of a Distributed Cache Architecture with Prediction. Technical Report CS-1994-05, Department of Computer Science, Duke University, Durham, NC 27708-0129, March 9, 1994.
- [BC95] Azer Bestavros and Carlos R. Cunha. A Prefetching Protocol Using Client Speculation for the WWW. Technical Report TR-95-011, Computer Science Dept., Boston University, 111 Cummington St, Boston, MA 02215, May 1995.
- [BCC⁺95] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-Level Document Caching in the Internet. Technical Report BU-CS-95-002, Boston University, Boston, MA 02215, February 1995. Also in Proc. SDNE'95 (2nd. International Workshop on Services in Distributed and Networked Environments).
- [Bes95] Azer Bestavros. Using Speculation to Reduce Server Load and Service Time on the WWW. Technical Report TR-95-006, Computer Science Dept., Boston University, 111 Cummington St, Boston, MA 02215, February 1995.
- [BH96] Jean-Chrysostome Bolot and Philipp Hoschka. Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design. In *Electronic Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 6-10, 1996. The World Wide Web Consortium.
- [BLCcGP92] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2), Spring 1992.
- [CB95] Mark Crovella and Azer Bestavros. Explaining World Wide Web Traffic Self-Similarity. Technical Report TR-95-015, Computer Science Department, Boston University, 111 Cummington St, Boston, MA 02215, August 1995. Also in Proc. of SIGMETRICS'96.

- [CBC95] Carlos R. Cunha, Azer Bestavros, and Mark Crovella. Characteristics of WWW Client-based Traces. Technical Report TR-95-010, Computer Science Dept., Boston University, 111 Cummington St, Boston, MA 02215, April 1995.
- [CP95] Lara D. Catledge and James E. Pitkow. Characterizing Browsing Strategies in the World-Wide Web. In *Electronic Proceedings of the Third International World Wide Web Conference*, Darmstadt, Germany, April 10-13, 1995. The World Wide Web Consortium.
- [Cun97] Carlos R. Cunha. *Trace Analysis and Its Applications to Performance Enhancements of Distributed Systems*. PhD Thesis in preparation, Computer Science Department, Boston University, 111 Cummington St, Boston, MA 02215, USA, 1997.
- [Den96] Shuang Deng. Empirical Model of WWW Document Arrivals at Access Link. In *Proc. International Communications Conference - ICC'96*, Dallas, Texas, USA, 23-17 June 1996. IEEE Press.
- [GW70] J.E. Gillis and G.H. Weiss. Expected Number of Distinct Sites Visited by a Random Walk with an Infinite Variance. *J. Math. Phys.*, 11:1307-1312, 1970.
- [Kri95] P. Krishnan. *Online Prediction Algorithms for Databases and Operating Systems*. PhD thesis, Department of Computer Science, Brown University, Providence, Rhode Island 02912, USA, August 1995. Also as Technical Report CS-95-24.
- [LS84] Johnny K.F. Lee and Alan Jay Smith. Branch Prediction Strategies and Branch Target Buffer Design. *IEEE Computer*, 17(1):6-22, January 1984.
- [Man83] B. Mandelbrot. *The Fractal Geometry of Nature*. Freeman, San Francisco, CA, 1983.
- [Mar96] Evangelos P. Markatos. Main Memory Caching of Web Documents. In *Electronic Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 6-10, 1996. The World Wide Web Consortium.
- [NCS] NCSA. Mosaic browser. Available via anonymous ftp from ftp.ncsa.uiuc.edu.
- [Pad95] Venkata N. Padmanabhan. Improving World Wide Web Latency. Report No. UCB/CSD-95-875, Computer Science Division, University of California at Berkeley, Chicago, USA, May 1995.
- [Pap91] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, New York, NY, USA, 3rd edition, 1991.
- [PZ91] Mark L. Palmer and Stanley B. Zdonik. Fido: A Cache that Learns to Fetch. Technical Report CS-91-15, Department of Computer Science, Brown University, Providence, RI 02912, February 1991.
- [RS78] Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall Signal Processing Series. Prentice-Hall, 1978.
- [SLM87] Jr. S. Lawrence Marple. *Digital Spectral Analysis with Applications*. Prentice-Hall Signal Processing Series. Prentice-Hall, 1987.
- [Smi82] Alan Jay Smith. Cache memories. *ACM Computing Surveys*, 14(3):473-530, September 1982.
- [Thi89] Dominique Thiébaud. On the Fractal Dimension of Computer Programs and Its Applications to the Prediction of the Cache Miss Ratio. *IEEE Transactions on Computers*, 38(7):1012-1026, July 1989.
- [VH81] J. Voldman and Lee W. Hoewel. The Software-Cache Connection. *IBM Journal of Research and Development*, 25(6):877-893, November 1981.
- [VMH⁺83] Jean Voldman, Benoit Mandelbrot, Lee W. Hoewel, Joshua Knight, and Philip L. Rosenfeld. Fractal Nature of Software-Cache Interaction. *IBM Journal of Research and Development*, 27(2):164-170, 1983.