Proceeding of the International Conference on
Internet and Multimedia Systems and Applications, IMSA 2000
November 20-23, 2000 Las Vegas, USA, pp 280-285.

DYNAMIC PARTIAL PREFETCH RANKING IN HYPERMEDIA NEIGHBORHOOD

JAVED I. KHAN

Media Communications and Networking Research Laboratory
Department of Math & Computer Science, Kent State University
233 MSB, Kent, OH 44242
javed@kent.edu

Abstract

This paper investigates how to rank the candidate pages for potential pre-fetch to accelerate responsiveness of the web access. Given the estimates of the hyperlink transition probabilities, it seems that instead of prefetching in order of maximum likely hood path, a ranking order that also considers the loading time yields much better performance both in terms of reduction in overall response lag and the wasted prefetch.

Keywords: prefetch, streaming, cache.

1 Introduction

Caching and prefetching are the two principal techniques for improving responsiveness for systems that involve data communication. Basic caching helps the case of repetitive references, prefetching reduces response lag for newer resources. Both have been extensively used in hardware systems to offset memory access latency. In Internet systems, caches have been integrated within the infrastructure [1,2], and recent focus has been shifted to prefetching.

Modern prefetch enabled caching for hardware systems have shown more than 90% hit rate. However, the Web is yet to match the success [3,4,5]. Web prefetching faces some additional complexities. Decision points mostly bifurcate the control flow tree in hardware due to the *if-then* programming construct that drives branching. In contrast, the degree of branching in the web is higher since there is no limit on the number of links in a page. In hardware, quite often all the parallel branches are prefetched. In some cases conditions can be pre-evaluated to compute the prefetch path. Neither seems to be very practical for web systems.

The secondary/tertiary memory access times are mostly uniform for hardware pages. However, it widely varies between web pages due to wide differences in their sizes, link speeds and proxies in access path. There is also concern that excessive web prefetch can have adverse effect since the network here is highly shared [6,7,8]. Clearly, optimization at the branch point seems to be very crucial for the case of Web. In this research we focus on the issue-- on the selection and ordering of transition paths in the case of high branching factor and varying access time.

1.1 Related Works

The research in prefetching has gained momentum more recently [5,9,10]. Kroeger et. al. demonstrated that with ample knowledge of future reference a combined caching and prefetching can reduce access latency as much as 60% [3,11]. Jacobson and Cao [10] proposed a prefetching method based on partial context matching for low bandwidth clients and proxies. Palpanas and Mendelzon [4] demonstrated that a k-order Markovian prediction engine can improve response time by a factor of up to 2. Both these methods studied context matching (past sequence of accessed references) for prediction of future web references. Pitkow and Pirolli [12] investigated various methods that have evolved to predict surfer's path from log traces such as session time, frequency of clicks, Levenshtein Distance analyses and compared the accuracy of various construction methods. This Markov model based study noted that although information is gained by studying longer paths, but conditional probability estimate, given the surf path, is more stable over time for shorter paths and can be estimated reliably with less data.

Also of interest is the work by Cohen and Kaplan [8] who cited bandwidth waste in prefetch, and as opposed to document prefetch, suggested pre-staging only the communication session- such as pre-resolving DNS, pre-establishing of TCP connection and pre-warming by sending dummy HTTP HEAD request. RealPlayer (release 7 onward, 1999) already pre-stages streaming connections linked from a page by pre-extracting and readying individual codec associated with each.

While considerable empirical studies have been performed to study the methods for predicting path transition probabilities. Many of the previous studies, except for Cohen and Kaplan [8], seems to have two assumptions in ordering and prefetch stage—where the results of the predictions are actually used. First is the full

Proceeding of the International Conference on
Internet and Multimedia Systems and Applications, IMSA 2000
November 20-23, 2000 Las Vegas, USA, pp 280-285.

document prefetch. Second is prefetching documents in order of maximum likely hood path.

This research, presents a prefetch mechanism that differs on these two assumptions. Earlier, in [7] we proposed that instead of full documents (or none), a middle approach is possible. A **partial prefetch** scheme has been presented in [7] where only an optimum estimated lead segment is prefetched. It can significantly reduce the wasted prefetch without compromising the responsiveness gained in the first place.

In this paper we present the analysis on the other distinguishing aspect-- **path ordering**. Instead of prefetching in order of maximum probable transition paths, we propose a new ranking order-- which optimizes the expected response time with respect to all probable transition paths.

In the following sections, we first present the model and the analytical considerations supporting the prefetch scheme. Section 3 then outlines key systems issues. Finally, section 4 presents the performance of the scheme under various workload supported by rigorous statistical simulation. The deployment of any prefetch system, including ours, will require a much more detailed design work at the protocol level. However, discussions about these issues cannot be accommodated in this paper.

2 Analysis

2.1 Hyperspace Model

The reader moves through a sequence of nodes in the hyperspace. Lets call this the *anchor sequence*. When a client (reader) program is active, the idea is to track the *anchor nodes*, monitor its neighboring regions, and prefetch selected parts from a subset of these nodes in client's *prefetch cache* with high likely-hood of traversal based on some optimization objective. Generally, information about only a subpart of the hyper-space is *visibl*e to any practical cache system. For tractability, a further pruned graph should be used for pre-fetch decision. We call this graph the *roaming- sphere* and will denote it by $G(V_G,E_G)$. A subset of its' nodes is eventually preloaded in the prefetch cache. We also assume that nodes in $G(V_G,E_G)$ has some node 'statistics' (such as size or load time). Also, each link in it has a *transition probability* p(i,j) associated with it.

2.2 Streamed Transport Model

According to the partial prefetch scheme the transport model also divides the transport into two probable phases. Each node thus has two transport parts-- the *lead segment* and the *stream segment*. The available bandwidth is correspondingly separated into two sub-channels; *feed channel* for loading the streaming segment of the current anchor, the *lead channel* to proactively load the lead segments of resources. Fig-4 shows the event sequence for the transport model. We assume that D_{total} is the size of an elementary resource, D_{lead} is the bytes in

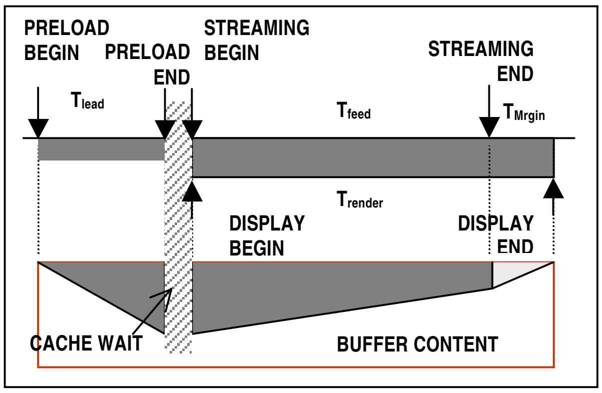

Fig-1 Transport event model used in the analysis. The top diagram shows event sequence and various time quantities and the bottom diagram shows the buffer occupancy under constant rate operation.

lead segment and D_{feed} is the bytes to be streamed. We use β to denote the ratio of total bandwidth to that allocated to the lead sub-channel.

2.3 Prefetch Node Ranking

We now address the question-- what is the best prefetching sequence of the nodes that will minimize the expected cumulative *read-time lag* for a given network bandwidth?

First we define the optimization criterion. In a hyper-graph G, Lets $U=(a_1,a_2,a_3\ldots a_l)$, where $u_i \in G$, is the *anchor sequence*-- the sequence of nodes followed by a user. Let's Γ is the *prefetch sequence* in which the nodes are loaded in the cache (Clearly, $U \subseteq \Gamma \subseteq$ {nodes in G}). Let p_i is the estimated probability that a user traverses a node n_i in roaming sphere G, and $T_{L,i}$ is the time the node a_i is fetched and $T_{P,i}$ is the time spent by the user in that node.

Thus, we define an overall penalty function-- the expected cumulative *read-time lag*:

$$T(\Gamma \mid U) = \sum_{i}^{U} p_i \max\{[T_{L,i} - (T_{L,i-1} + T_{P,i-1})],0\} \qquad ..(1)$$

The objective is to find the *prefetch sequence* Γ that will minimize the expected penalty $E\{T(\Gamma|U)\}$.

It is important to note that this function optimizes with respect to all probable transitions of U, weighted by their transition probability.

Given the above optimization criterion, it can be shown that:

<u>Theorem-1 (Branch Decision)</u>*: Let $A=n_c$ is the current anchor point with direct transition paths to a set of candidate nodes n_1, n_2, $n_{3,\text{--}}$ n_n, such that T_i is the estimated loading times of node n_i, and $Pr[a_{n+1}=n_i \mid a_n=A]$ is the conditional link transition probability, then the*

Proceeding of the International Conference on
Internet and Multimedia Systems and Applications, IMSA 2000
November 20-23, 2000 Las Vegas, USA, pp 280-285.

average delay is minimum if the links are prefetched in-order of the highest priority Q_i, where:

$$\log Q_i = \log \frac{\Pr[a_{n+1} = n_i \mid a_n = A]}{T_i} \qquad ...(2)$$

For brevity, only the result is stated here. The proof can be found in the web retrievable reference [14].

This theorem states that at a simple branch point (roaming sphere of depth=1) embedded URLs should be prefetched in order of the conditional transition probability but in inverse order of their estimated load time.

It can be seen from (2) that in hardware prefetch, for each memory hierarchy the access time for all pages (from the stage immediate below) is uniform. Thus, a transition probability based hierarchy will still be optimum. However, the assumption of uniformity is not valid for the case of web access.

Unfortunately, the relative priority cannot be determined for a general graph for the optimization criterion defined by equation (2). In advanced work, under a slightly modified definition which ignores the credit due to cumulative read times along the paths ($T_{p,i-1}$=0 for all i in equation-2) it can be proven. See [14] for detail. However, equation (2) is sufficient for ranking immediately traversable hyperlinks. Seldom excessive prefetch bandwidth is available for deeper prefetch.

2.4 Lead Mass

The next question we address is how much of each node should be prefetched? This will also determine the pre-load time to be used in (2). Let us assume that the margin time=0. We use the constraint that the reading or rendering time should be at least equal to the streaming time for all components.

The idea is that instead of allocating full bandwidth to prefetch, we only allocate a part for prefetching and remaining is used for real time fetching of the remaining part of the current anchor. Given the bandwidth partition, and the estimated reading/rendering time of the document, it is possible to obtain how much data can be fetched in real time. We should prefetch the difference. Consequently, for a given consumption rate R^i_{render} for the media type=*i* the amount of data that has to be prefetched is:

$$D^i_{lead} \ge D^i_{total}\left(1 - \frac{B^i_{feed}}{R^i_{render}}\right) = D^i_{total}\left(1 - \frac{\beta^i \cdot B_{channel}}{R^i_{render}}\right) \qquad .(3)$$

If the pre-fetch mechanism is a proxy cache, instead of an end-client, the consumption rate will simply be the bandwidth between the requesting client and the proxy. Only the D_{lead} amount of data should be pre-loaded for minimum delay. It provides a lower bound and we call it *critical lead mass*. Note that, the other part left behind for fetching later, should not result in any increase in the perceivable delay to the client agent, because at any time client will have ample data to play. Corresponding pre-load time for the media is given by:

$$T_{lead} = \frac{\sum_i D^i_{lead}}{B_{preload}} \qquad ...(4)$$

3 Techniques

The prefetch mechanism can be implemented either as a client system or as a client proxy. Some system wide non-trivial mechanisms will be needed for procuring and propagating the link and document access statistics. In addition the prefetch mechanism will need techniques for partial document transfer and bandwidth partitioning.

3.1 Document Statistics and Partial Get

HTTP 1.1 Request-Header *Referrer* field can be used for generating access statistics [13]. A server plug-in then can tracks the intra-server link references from the referrer values. The list of embedded link, profile parameters and access statistics for hot documents can be stored in a *hot-prefetch database* by the origin server. Prefetch system can receive the statistics by a new *GET_statistics* request method implemented using HTTP reserve pool. The hot-prefetch database server should respond with the statistics for an URL.

The mechanism to obtain only a selected set of bytes can be implemented using the conditional R*ange GET* mechanism of HTTP 1.1 (If-Range header, Range and Content-range, Response Code 206 Partial Content).

3.2 Bandwidth Partition

Bandwidth partitioning is not available in the current QoS-less Internet. Instead a technique called **episodic byte proportioning** can be used effectively creating the same result. Here the *Range GET* requests are queued on the basis of small time episodes. The waiting *Range GET* requests are then proportionately interleaved in a queue within the episodes. The scheme works because the exact delivery times of the prefetch segments do not matter. Since, these are not immediate rendered. However the requests for real time feed segments should be at the top of the episode queues-- effectively giving them a little prefetch.

3.3 Method Prefetch

On the proxy side, when a new *user-agent* is detected, a new prefetch session is initialized for tracking its roaming sphere. The prefetch algorithm then finds $G(V_G,E_G)$ by recursive *GET-statistics* traversal using a cut-off threshold ε.

The prefetch mechanism then first computes the *critical lead mass* for probed hot nodes. It then determines the priority of the nodes according to the equation-2. The

Range GET requests are then formulated for appropriate lead byte segments based on small episodes.

The loading order remains valid until the current anchor node is read. Upon completion, a new node is traversed, and the episode queues are reorganized. The subsequent evaluation is incremental. A new anchor point changes only the conditional probabilities. Consequently, in this new graph, the conditional transition probabilities (and thus the priority) of the nodes downstream are upgraded by factor 1/p(i,j), while nodes branching out from upstream are downgraded by factor p(j,i).

The optimum ranking analysis presented here does not make any assumption about how to estimate the link transition probabilities. Any of the proposed methods [10,4,12] can potentially be used. For our simulation our design choice was the simplest one - the access frequency analysis based transition probability estimation.

4 Performance Results

It seems that the composition of web is changing significantly over time. For this particular work we found it more appropriate to generate broad range of hyperspace data with distinct and varying statistical properties and observe how the proposed method will perform in each of these conditions, rather than conducting a trace driven analysis (which is generally a single snapshot and offer little controllability of parameters).

In this experiment we generated a random set of nodes each with a parent HTML containing links to others and a set of embedded media and hyperlinks. We limited the maximum hyperlinks per page to be 21 (The underlying algorithm further pruned links with below ε low transition frequency and effectively considered much less links for prefetch). We generated the *document sizes* and the *link transition probabilities* for all links using normal distribution ($Pr_{SDIST}(x)$ and $Pr_{PDIST}(x)$, where x is the hyperlink index). Fig-1 plots a typical distribution pair. Here the link transition probability and the size distributions respectively have modes (index of maximum likelihood class) at paths 7 and 13. We have tested the system extensively for various distribution mean and standard deviations. Since, eq. (2) is a ratio of the two quantities, we ran the experiment for various mode separations.

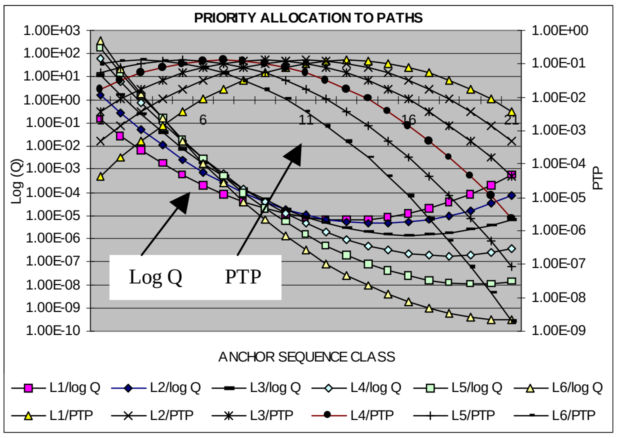

Fig-3

We evaluated their **Log Q** based priorities using equation (2). For comparison we also computed the

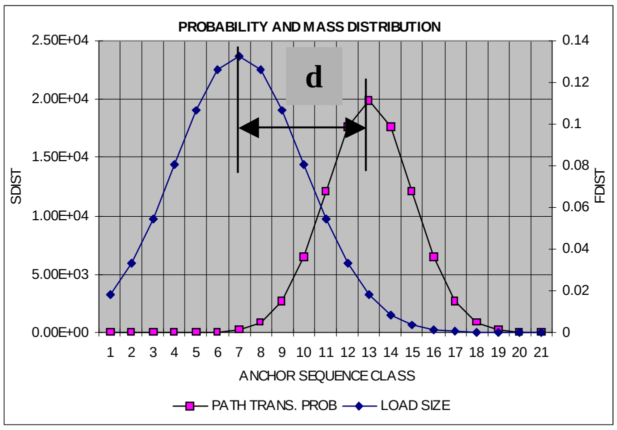

Fig-2

simple **path transition probability (PTP)** based priorities for all these cases. Fig-3 plots all the results. The Log Q and PTP priority values have different scales. Thus left y-axis and the first six curves show the path priorities assigned by Log Q method for six values of distribution mode separations (d). In these six cases we kept the mode of the size distribution $Pr_{SDIST}(x)$ to 3, and varied the mode of the $Pr_{PDIST}(x)$ respectively to L1=13, L2=11,L3=9, L4=7, L5=5, L6=3). The right y-axis and the upper six curves show the path priorities assigned by the PTP order for the same six cases. As can be seen from the plot these two methods followed quite different *prefetch sequence* Γ.

Now to evaluate the performance, we let the simulator fetch the pages in order of their priorities. The system determines the *critical lead mass* D_{lead} according to equation (3) and fetches this segments while the rendering progresses for current document. (At the same time it also fetches the feed segment of the current document). To estimate the expected delay we computed the estimated speedup experienced by all 21 probable anchor sequences for the same prefetch sequence U given by equation (2). Fig-4 plots the observed speedup in the responsiveness. It shows three quantities-- the maximum, minimum and the expected average. The later is obtained by weighing each class's speedup by the number of

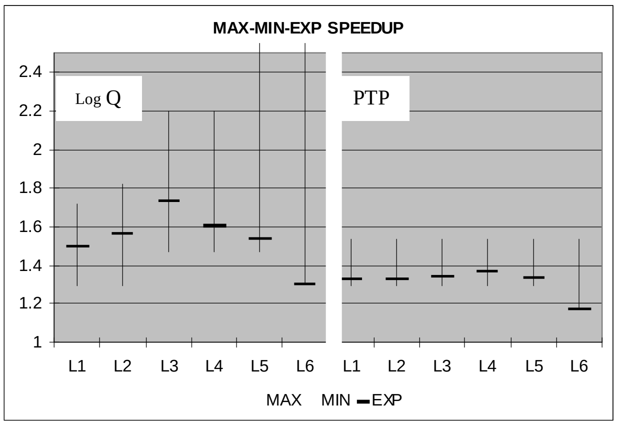

Fig-4

surfer's who follow the path. We assume this to be proportional to the estimated transition frequencies. The left L1-L6 set here plots the high-low-mean values of the observed performance for proposed log Q priority. For comparison the right L1-L6 set plots the observed performance for PTP priority.

The PTP priority demonstrates maximum speed-up in the range of about 1.2 to 1.5 (this is not far from what have been reported by other studies). It's mean seems to be less than 1.4. In contrast Log Q priority consistently demonstrated superior performance where mean ranged from 1.5-1.75. Also its maximum and minimum seems to be consistently much higher. Also, an interesting observation can be made— best-case improvement is more dramatic as the mode separation (d) becomes smaller (L5, and L6).

This gain in performance is not unexpected. The analytical model explains the optimum improvement of the expected average responsiveness over all the surfer's path classes. The major improvement in the best-case is due to the fact that the log Q ordering favors shorter jobs to be cleared first. It enables surfers' of the smaller documents to experience major improvement in system's responsiveness (from 1.6 to almost 2.2 and above).

5 Conclusions & Current Work

Prefetching seems to be an attractive method for improving responsiveness of web systems. However, prefetching need to be performed very selectively. Not only the branching factors are high, but also, (unlike the case of hardware) web is a highly shared resource. In this research a technique has been proposed for optimally ordering hyperlinks. The result suggests that significant performance improvement is possible by incorporating loading time in to the link priority evaluation.

As indicated, this work is not about the estimation methods of link transition probability. Whether it is computed from conventional access log [4,10,12], or from explicit message exchange [9], the issue addressed here is how to use them.

Within the scope of this paper, only non-trivial steps- such as bandwidth partitioning on QoS less Internet have been outlined (section-3). However, the deployment of any prefetch system, including the one proposed here will require significant research on protocols. Few insightful references on these areas are in [15-18].

Within the scope of this paper, we deliberately switched off any caching. However, the impact of caching parameters (such as replacement policy, cache size, etc.) on an integrated scheme is an important area deserving further study.

The work is currently being funded by DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.

6 References:

[1] D. Wessels and K Claffy, ICP and the Squid Web Cache, IEEE Journal on Selected Areas in Communication, April 1998, Vol 16, #3, pp.345-357.

[2] Nancy Yeager & R. E. McGrath, Web Server Technology, Morgan Kaufmann, San Francisco, 1996.

[3] Z. Wang and J. Crowcroft, Prefetching in the World Wide Web. in procs. of IEEE Global Internet, London, UK, 1996.

[4] T. Palpanas and A. Mendelzon,, Web Prefetching Using Partial Match Prediction, WWW Caching Workshop, San Diego, CA, March 1999

[5] Li Fan, Pei Cao, and Quinn Jacobson. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. Procs. of the ACM SIGMETRICS' 99, Atlanta, Georgia, May 1999.

[6] M. Crovella, P. Barford, The Network Effects of prefetching, Proc. Of IEEE INFOCOM 1998, San Francisco, USA, 1998.

[7] Javed I. Khan, Active Streaming in Transport Delay Minimization, Workshop on Scalable Web Services, Int. Conf. on Parallel Processing, Toronto, August 2000, pp95-102.

[8] E. Cohen and H. Kaplan. Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency. Procs. of the IEEE INFOCOM 2000, Tel-Aviv, Israel, March 2000.

[9] D. Duchamp. Prefetching Hyperlinks. Proceedings of the USENIX Symposium on Internet Technologies and Systems, Colorado, USA, October 1999. [Http://www.usenix.org/events/usits99].

[10] Q. Jacobson, Pei Cao, Potential and Limits of Web Prefetching Between Low-Bandwidth Clients and Proxies, 3rd Int. WWW Caching Workshop, Manchester, England, June 15-17 1998.

[11] T. Kroeger, D. D. E. Long & J. Mogul, Exploring the Bounds of Web Latency Reduction from Caching and Prefetching, Proc. Of USENIX Symposium on Internet Technology and Systems, Monterey, December 1997, pp-319-328.

[12] P. Pirolli and J. E. Pitkow, Distributions of surfers' paths throug the World Wide Web: Empirical characterizations, Jounral of World Wide Web, 1999, v.1-2, pp29-45

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk & T. Berners-Lee, Hypertext Transfer Protocol HTTP/1.1, RFC 2068, January 1997.

[14] Javed I. Khan, Ordering Prefetch in Trees, Sequences and Graphs, Technical Report 1999-12-03, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports. html, also mirrored at http:// bristi.facnet.mcs.kent.edu/~javed /medianet]

[15] Grosso, Paul, Daniel Veillard, XML Fragment Interchange, W3C Working Draft 1999 June 30, [Retrieved from: http://www.w3.org/1999/06/WD-xml-fragment-19990630.html]

[16] A. Dan and D. Sitaram, Multimedia caching strategies for heterogeneous application and server environments, Multimedia Tools and Applications, vol. 4, pp.279-312, May 1997.

[16] S. Gruber, J. Rexford, and A. Basso, Design considerations for an RTSP-based prefix caching proxy service for multimedia streams, Tech. Rep. 990907-01, AT&T Labs - Research, September 1999.

[18] J. Jung, D. Lee, and K. Chon, Proactive Web Caching with Cumulative Prefetching for Large Multimedia Data. Procs. of the 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000.

Proceeding of the International Conference on
Internet and Multimedia Systems and Applications, IMSA 2000
November 20-23, 2000 Las Vegas, USA, pp 280-285.