Prefetching on the Web Through Merger of Client and Server Profiles

Craig E. Wills and Joel Sommers

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

Abstract

In this work we look at the idea of merging client and server profiles for enhanced performance through prefetching. Previous work has focused on each of these types of profiles, but in isolation of each other. Our work builds on previous server profile work, but goes much further in defining client profiles, showing how to combine them with server profiles and evaluating the effectiveness of such combined-profile prefetching using a diversity of Web server logs. Results of trace-driven simulations suggest that such an approach to prefetching is effective in reducing access latency without significantly increasing network and server load.

1 Introduction

Prefetching is risky venture. If done in a haphazard way, cache space on the client is wasted, network bandwidth is squandered, and server load is needlessly increased. If prefetching is to be of use we believe it must be done through cooperation of the client and server. In this work we look at the idea of merging client and server profiles for enhanced performance through prefetching [7].

Profiles can describe many aspects of a client or server. For this work, a client profile captures the browsing behavior of the client's user while the server profile includes information about which resources are accessed most frequently at that site.

Previous work has focused on each of these types of profiles, but in isolation of each other. The client may go through some effort to monitor and predict user access patterns, but without knowledge of a specific server it works with incomplete information. Similarly, assuming that all clients take the same approach for accessing the resources for a server reduces the potential gain from that server maintaining a resource usage profile.

The result of our work has been an investigation on the merit of merging client and server profiles for prefetching. The work builds on previous server profile work, but goes much further in defining how to determine client profiles, showing how to combine them with server profiles and evaluating the effectiveness of such combined-profile prefetching using simulation with a diversity of Web server logs. The following presents a condensed version of our results.

2 Related Work

Related to server profiles, Bestavros [1, 2] has implemented a system in which the server responds to a resource request with, in addition to the requested resource, a series of popular resources as monitored by the server. The assumption is that popular resources are globally popular, and not just to a small set of users. Mogul describes a similar method of prefetching using server suggestions [6].

Recent work by Lee [5] uses access statistics on the server to predict client accesses. The server does not automatically send popular documents, but rather the client must explicitly request prefetch prediction information. These prefetch predictions are cached in tables at the client for future reference with a fixed cache of 100 entries and a LRU cache removal policy. This approach makes it difficult to separate effects of prefetching from effects of cache removal.

Related to a client profile, Wachsberg et al. [8] developed a prefetching scheme based on past user behavior. The user's resource history list, frequently accessed pages ("hot pages"), usage of hyperlinks (e.g. whether links at the top of a page are accessed more frequently than those lower on a page), and a keyword match were all used in determining which resources should be prefetched. Also on the client side, work has been done to characterize a browsing strategy as either directed search, general purpose, or serendipitous [3]. Directed search is a closed task—

there is a specific topic the user is investigating. Serendipitous browsing is an open task—the user has no fixed goal in mind; hyperlinks are followed somewhat whimsically. 'General purpose' browsing does not fit easily into either category.

3 Approach

The approach used in our work is to compute both a server and client profile for access to Web resources. How we determine and use these profiles is given in the following.

3.1 Server Profile

The work of Bestavros [2] forms the basis for much of our work on server profiles. In our work, we divide the accesses from a single client to a server into *sessions*. We use a value of one hour to limit what should be considered related accesses. Within any session, two sequential accesses for resources R_i and R_j are assumed to be related in one of two ways. Either the relationship is an *embedded dependency*, meaning resource R_j is embedded (or inlined) in resource R_i, or the relationship is a *traversal dependency*, meaning either there is a hyperlink from R_i to R_j, or R_j was accessed in some other way after accessing R_i (e.g. a URL was typed directly into the browser, a bookmark was selected, etc.). The reason for distinguishing between the two types of dependencies is that embedded dependencies are not good candidates for prefetching—inlined images will *always* be automatically retrieved by the browser upon parsing the page in which they are referenced (unless already cached or image retrieval is disabled in the browser). In examining the real benefits of prefetching we believe it is most appropriate to consider traversal dependencies because it is here that potential for prediction would have the most gain.

Clearly, any detectable dependencies are influenced by the size of the window. In our work we explored a number of sizes ranging from 5 to 120 seconds, with a size of 30 used as a baseline. For a given window size, we considered three methods for computing dependencies: single-pair, closure, and multiple-pair. Single-pair and closure are the same as defined in [2] where the single-pair method indicates a dependency between R_i and R_j if the percentage of access of R_j directly after R_i is accessed exceeds a given threshold. The closure method indicates a dependency between two resources if there is a transitive closure of the single-pair dependencies. To take into account non-linear dependencies we also explored the multiple-pair method, which indicates a dependency between R_i and all resources that are accessed within the given window regardless of specific order. Our premise was that the traversal of resources could be similar between different clients without being linearly the same. However in our testing we found this approach generated too much "noise." In the results shown we always use the single-pair method.

3.2 Client Profile

If a user is "bouncing" from server to server, it is futile to prefetch because the user will probably not remain at any server long enough to make prefetching worthwhile. In contrast, a user who spends more time viewing resources at a single server is probably a better candidate for prefetching. We term a user as operating in different browsing modes[1] as *wanderer*, *sojourner*, and *resident*:

- Wanderer: few requests made to any one server. The user might be looking through search engine results.
- Sojourner: many requests are not concentrated to any particular site, but the user is not quickly bouncing.
- Resident: many requests are made to a server. The user might be reading material at a digital library.

These modes may be compared with Pitkow and Cateledge's classification of users as either browsing serendipitously, browsing in a general-purpose way, or engaged in a directed search [3].

Dynamically deciding which browsing mode the user is in is not directly addressed in our work. Rather, the log for a server is divided into three types of clients. Wanderers form the lower quartile of client requests per session for a server, residents form the upper quartile, and sojourners comprise the interquartile range.

This implementation allows us to explore the potential of combining and client and server profiles. Actual implementing the determination of the browsing mode by a browser might be done through measuring how many requests the user makes to a particular site, and using a history of these measurements to predict how long the user will stay at future sites.

3.3 Use of Profiles

The method we adopt is for using these profiles is that the client asks for suggestions and the server replies with a list of URLs. This strategy allows a client to selectively prefetch from a list of server-supplied suggestions, and also for the client to decide when and if it wants to act on suggestions. The client can initiate prefetching by using an extension HTTP header, such as `Suggest-accept`, to which the client attaches a percent value P between 0 and 1 indicating that it is willing to accept suggestions with a server-computed likelihood of access greater than or equal to P. In response, the server returns the requested URL and *piggybacks* [4] a header `Suggest` with a list of suggestions. The following shows how a server might reply with suggested URLs for the resource `/index.html` with a suggestion threshold

[1]Describing user behavior is problematic because terms such as 'strategies' and 'styles' give either the impression that users deliberately shift between a discrete set of behaviors, or that users are aware of how they are browsing. Neither of these impressions is intended nor should be inferred.

of 0.3. The server includes the probability of access and size for each suggested resource.

```
HTTP/1.0 200 OK
Content-type:  text/html
Suggest:  /People/index.html (0.32) (2432)
Suggest:  /Programs/index.html (0.50) (7441)
Suggest:  /Help/index.html (0.61) (4246)
```

4 Test Results

Server logs were used to drive a simulation study. A diverse set of logs from the Worcester Polytechnic Institute computer science department (cs.wpi.edu), the National Center for Supercomputing Applications (NCSA) (www.ncsa.uiuc.edu), and from Sun Microsystems (www.sun.com) were obtained [more details on log characteristics in final draft]. Testing with these logs was done using the first half of each log used to determine resource dependencies at a server and the second half of each log used to simulate client access to the server.

The primary goal of prefetching is to reduce client access latency. However, if resources are prefetched and never eventually requested, network load and server load are both increased, and cache space on the client is wasted. Thus the results are presented in terms of the percentage of prefetches that are used and the percentage of prefetches that are wasted (never used).

4.1 Baseline Results

The baseline parameters we used were the single-pair dependency method and with a window size of 30 seconds. Initially we considered only using server suggestions without a client profile. In order to try and capture traversal and not embedded dependencies, we used only references to HTML and text resources in order to remove accesses of inline images. The results for the baseline parameters are shown in Figure 1 for each of the server logs. The results are shown for bytes with similar results for resources [results on resource basis will be shown in final draft].

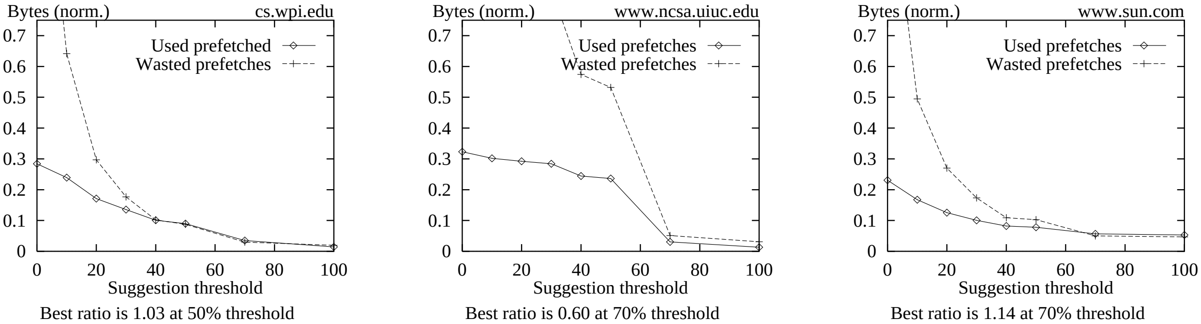

Figure 1: Baseline Results (Bytes)

Data points of Figure 1 (and similar graphs to follow) are normalized against the number of bytes (or resources) required for a session. The best ratio for each graph attempts to capture the point in each graph where the ratio between used and wasted prefetches is maximized. A ratio of 1 indicates that prefetching is equally beneficial as it is wasteful. For ratios less than 1, more prefetches are wasted than are used, and for ratios greater than 1, more prefetches are used than are not.

Looking at the graphs of Figure 1, one notices that the plots for WPI and Sun are similar while the NCSA plots are somewhat aberrant. One explanation is that a high percentage of client sessions to NCSA involve only a single request, such as is done by the Mosaic browser when it defaults to loading the NCSA home page on startup. Since benefits from prefetching can only be realized if more than one access is made over a session, any prefetching done for a session of just one request will be wasted. For the NCSA log, 66% of all sessions are composed of just 1 request (13054 out of 19726 total sessions). For the WPI and Sun logs, the percentage of single-request sessions is about 20% lower, at 47% and 41%, respectively. The average size of NCSA resources is also larger, which can have a significant negative impact if incorrectly prefetched.

4.2 Consideration of All Resources

If simulations are performed without filtering images and other non-HTML resources from the logs, results improve dramatically. However, this increase is due to embedded dependencies. For brevity, just the Sun log is shown in Figure 2 [more details on based on variation of parameters in final draft]. At a 50% threshold, 88.6% of the total bytes prefetched are images and other non-HTML and non-text resources. 91.9% of all used prefetches are of this same resource category. Including images, applets, etc. clearly inflates the potential for prefetching.

As a comparison, Bestavros considers all resource types while using a 5 second window, and computing closure [1]. His results suggest a 23% reduction in service time while prefetching conservatively using only server suggestions. He does not indicate how much of prefetched data are HTML resources or images.

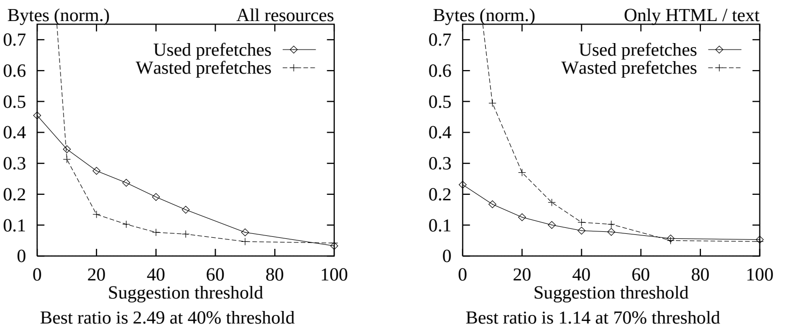

Figure 2: Including All Resources (www.sun.com)

4.3 Consideration of User Browsing Mode

In considering how users are browsing and applying this characterization to prefetching, it is important to understand is that in running tests, sessions are pre-classified as exhibiting one of the browsing modes as previously discussed. Since a mode is assumed at the outset of prefetching, results are best-case and show the potential of this approach. In examination of the logs, wanderers clearly comprise the majority of user types, especially in the NCSA log. The high number of wanderers in the NCSA log supports the suspicion that many single-request sessions are made to NCSA because the Mosaic browser defaults to loading the NCSA home page. Also for the NCSA log, residents make up only 2% of all session types. This is lower than for the WPI and Sun logs, where residents make up 11% and 9%, respectively.

Figure 3 gives results for different types of users accessing Sun. For wanderers and sojourners to Sun, at no point is prefetching being used more than not. For the other two logs, there is a point for sojourners where more is used from prefetching than is not used. For wanderers in all three logs, prefetching is always a losing effort (not shown). For Sun, only residents gain by prefetching. For a 70% threshold for residents, 8% of requests are satisfied from prefetched data with 4% wasted prefetches (ratio of 2.10).

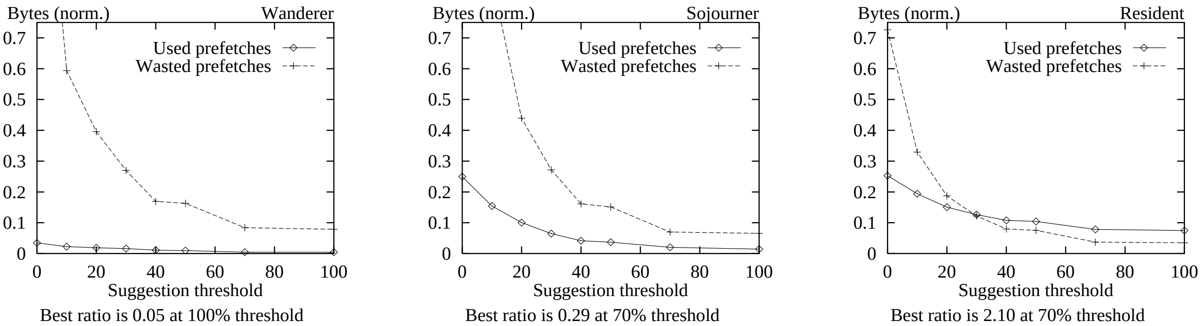

Figure 3: Comparison of Client Modes (www.sun.com)

4.4 Combination of User and Server Profiles

When an attempt is made to consider how a user is accessing the Web, prefetching is improved over a strategy where only server suggestions are considered. The degree of potential improvement varies from log to log. In the following, we compare the best server-only parameter set, which happens to be the baseline configuration, with a weighted combination of the best result for each user browsing mode[2]. The weighting considers the average number of resources or bytes requested for sessions in each browsing mode and the composition of a log by browsing mode.

[2]For the WPI and NCSA logs, the best results for wanderers, sojourners, and residents occur at 70%, 50%, and 50% thresholds, respectively. For the Sun log, these thresholds are 100%, 70%, and 70%.

For the NCSA log, results shown in Figure 4 dramatically show how much client information can help. If the number of resources (resources) is looked at, 36% of requests could be satisfied using browsing mode hints (used prefetches), compared with 11% at the best ratio for a server-suggestion strategy. The ratios of 3.60 and 1.14 for the respective schemes also support including browsing mode information. Using only server suggestions, more prefetched bytes are wasted than used. If we consider browsing modes, we could prefetch 16% of required bytes, with a ratio of 1.78.

Similar, although not as dramatic results are obtained for the WPI and Sun logs (not shown). For the WPI log 10% of resources and bytes could be retrieved from prefetching, with ratios of 1.42. For the Sun log, 7% of what is required could be prefetched, with a ratio of 1.75.

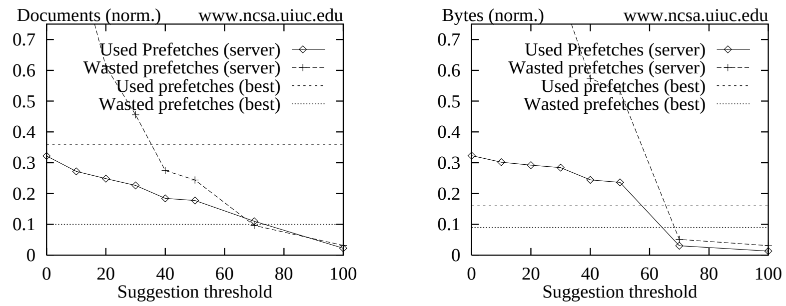

Figure 4: Comparison of Server-Only Prefetching with Constant Results of Considering User Browsing Mode (www.ncsa.uiuc.edu)

5 Summary

Suggestions of resources to prefetch are best made by the server, where general access frequencies are known. Clients, however, are in the best position to make decisions on what to prefetch. We have introduce and given an operational definition for three user browsing modes—wanderer, sojourner and resident—in terms of how many successive requests are made to a server. Our results show that server-suggested prefetching when the user browsing mode is considered is superior to using server suggestions alone. The degree of improvement varies between different logs we used to drive simulations. Future work includes further refinement of merging the profiles along with more work on client-side determination of the browsing mode.

References

[1] Azer Bestavros. Using speculation to reduce server load and service time on the WWW. In *Proceedings of the Fourth ACM Conference on Information and Knowledge Management*, Baltimore, MD, November 1995. ACM. URL: http://cs-www.bu.edu/best/res/papers/cikm95.ps.

[2] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time in distributed information systems. In *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, LA, March 1996. URL: http://cs-www.bu.edu/faculty/best/res/papers/icde96.ps.

[3] Lara D. Cateledge and James E. Pitkow. Characterizing browsing strategies in the World-Wide Web. In *Proceedings of the Third International Conference on the World Wide Web*, Darmstadt, 1995. URL: http://www.igd.fhg.de/www/www95/ proceedings/papers/80/ userpatterns/ UserPatterns.Paper4. formatted.html.

[4] Balachander Krishnamurthy and Craig E. Wills. Piggyback cache validation for proxy caches in the world wide web. In *Proceedings of the National Laboratory for Applied Network Research Web Cache Workshop*, June 1997. http://www.nlanr.net/Cache/Workshop97/Papers/Wills/wills.html.

[5] David Lee. Pre-fetch document caching to improve World-Wide Web user response time. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, March 1996. http://www.visc.vt.edu / dlee /thesis /thesis.pdf.

[6] Jeffrey C. Mogul. Hinted caching on the Web. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, Connemara, Ireland, September 1996. URL: http://www-sor.inria.fr/sigops96/papers/mogul.ps.

[7] Joel Sommers and Craig E. Wills. Prefetching on the web using client and server profiles. Technical Report WPI-CS-TR-97-2, Computer Science Department, Worcester Polytechnic Institute, June 1997.

[8] Stuart Wachsberg, Thomas Kunz, and Johnny Wong. Fast World Wide Web browsing over low-bandwidth links. URL: http://ccnga.uwaterloo.ca/ sbwachsb/paper.html, 1996.