Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 3673–3692

Computer Networks

www.elsevier.com/locate/comnet

Performance analysis of a client-side caching/prefetching system for Web traffic ☆

Abdullah Balamash [a], Marwan Krunz [a,*], Philippe Nain [b]

[a] *Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA*
[b] *INRIA, 06902 Sophia Antipolis, France*

Received 6 May 2006; received in revised form 7 January 2007; accepted 15 March 2007
Available online 2 April 2007

Responsible Editor: N. Bartolini

Abstract

Network congestion remains one of the main barriers to the continuing success of the Internet. For Web users, congestion manifests itself in unacceptably long response times. One possible remedy to the latency problem is to use caching at the client, at the proxy server, or within the Internet. However, Web documents are becoming increasingly dynamic (i.e., have short lifetimes), which limits the potential benefit of caching. The performance of a Web caching system can be dramatically increased by integrating document prefetching (a.k.a. "proactive caching") into its design. Although prefetching reduces the response time of a requested document, it also increases the network load, as some documents will be unnecessarily prefetched (due to the imprecision in the prediction algorithm). In this study, we analyze the confluence of the two effects through a tractable mathematical model that enables us to establish the conditions under which prefetching reduces the *average* response time of a requested document. The model accommodates both passive client and proxy caching along with prefetching. Our analysis is used to dynamically compute the "optimal" number of documents to prefetch in the subsequent client's idle (think) period. In general, this optimal number is determined through a simple numerical procedure. Closed-form expressions for this optimal number are obtained for special yet important cases. We discuss how our analytical results can be used to optimally adapt the parameters of an actual prefetching system. Simulations are used to validate our analysis and study the interactions among various system parameters.

Keywords: Web modeling; Caching; Proxy; Prefetching; Multi-fractal traffic

1. Introduction

1.1. Motivation and related work

Web users can experience response times in the order of several seconds. Such response times are often unacceptable, causing some users to request the delayed documents again. This, in turn,

☆ This work was supported by the National Science Foundation through Grant ANI-0095626. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

* Corresponding author.
E-mail addresses: balamash@ece.arizona.edu (A. Balamash), krunz@ece.arizona.edu (M. Krunz), nain@sophia.inria.fr (P. Nain).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.03.004

aggravates the situation and further increases the load and the perceived latency. Caching is considered an effective approach for reducing the response time by storing copies of popular Web documents in a local cache, a proxy server cache close to the end user, or even within the Internet. However, the benefit of caching diminishes as Web documents become more dynamic [21]. A cached document may be stale at the time of its request, given that most Web caching systems in use today are *passive* (i.e., documents are fetched or validated only when requested).

Prefetching (or proactive caching) aims at overcoming the limitations of passive caching by proactively fetching documents in anticipation of subsequent demand requests.[1] Several studies have demonstrated the effectiveness of prefetching in addressing the limitations of passive caching (e.g. [14,17,22,23,27,31,32,35,42,46,49]). Prefetched documents may include hyperlinked documents that have not been requested yet as well as dynamic objects [42,37]. Stale cached documents may also be updated through prefetching. In principle, a prefetching scheme requires predicting the documents that are most likely to be accessed in the near future and determining how many documents to prefetch. Most research on Web prefetching focused on the prediction aspect. In many of these studies (e.g. [14,35]), a *fixed-threshold-based approach* is used, whereby a set of candidate files and their access probabilities are first determined. Among these candidate files, those whose access probabilities exceed a certain prefetching threshold are prefetched. Other prefetching schemes involve prefetching a fixed number of popular documents [32]. Teng et al. [43] proposed an integrated Web caching and prefetching (IWCP) policy, which considers both demand requests and prefetched documents for caching based on a normalized profit function. The work in [30] focuses on prefetching pages of query results of search engines. In [47], the authors proposed three prefetching algorithms to be implemented at the proxy server: (1) the *hit-rate-greedy algorithm*, which greedily prefetches files so as to optimize the hit rate; (2) the *bandwidth-greedy algorithm*, which optimizes bandwidth consumption; and (3) the *H/B-greedy algorithm*, which optimizes the ratio between the hit rate and bandwidth consumption. The negative impact of prefetching on the average access time was not considered.

Most of the above works rely on prediction algorithms that compute the likelihood of accessing a given file. Such computation can be done by employing Markovian models [20,35,41,36]. Other works rely on data mining for prediction of popular documents [38,48,29,34].

Numerous tools and products that support Web prefetching have been developed [1–4,6,7,9,10]. Wcol [3] prefetches embedded hyperlinks and images, with a configurable maximum number of prefetched objects. PeakJet2000 [10] is similar to Wcol with the difference that it prefetches objects only if the client has accessed the object before. NetAccelerator [9] works as PeakJet2000, but does not use a separate cache for prefetching as in PeakJet2000. Google's Web accelerator [4] collects user statistics, and based on these statistics it decides on what links to prefetch. It also can take a prefetching action based on the user's mouse movements. Web browsers based on Mozilla Version 1.2 and higher also support link prefetching [1]. These include Firefox [6], FasterFox [2], and Netscape 7.01+ [7]. In these browsers, Web developers need to include html link tags or html meta-tags that give hints on what to prefetch.

In terms of protocol support for prefetching, Davison et al. [19] proposed a prefetching scheme that uses a connectionless protocol. They assumed that prefetched data are carried by low-priority datagrams that are treated differently at intermediate routers. Although such prioritization is possible in both IPv6 and IPv4, it is not yet widely deployed. Kokku et al. [26] proposed the use of the TCP-Nice congestion control protocol [45] for low-priority transfers to reduce network interference. They used an end-to-end monitor to measure the server's spare capacity. The reported results show that careful prefetching is beneficial, but the scheme seems to be conservative because it uses an additive increase (increase by 1), multiplicative decrease policy to decide on the amount of data to prefetch. Crovella et al. [17] showed that a rate-control strategy for prefetching can help reduce traffic burstiness and queuing delays.

Most previous prefetching designs relied on a *static* approach for determining the documents to prefetch. More specifically, such designs do not consider the state of the network (e.g., traffic load) in deciding how many documents to prefetch. For example, in threshold-based schemes, *all* documents

[1] The term *demand request* is used throughput the paper to refer to a user's request for a document that needs to be displayed right away.