A Most Popular Approach of Predictive Prefetching on a WAN to Efficiently Improve WWW Response Times

Christos Bouras[1,2], Agisilaos Konidaris[1,2], and Dionysios Kostoulas[1]

[1] Computer Engineering and Informatics Department, University of Patras, GR-26500, Patras, Greece
kostoula@ceid.upatras.gr
[2] Computer Technology Institute-CTI, Riga Feraiou 61, GR- 26221, Patras, Greece
{bouras, konidari}@cti.gr

Abstract. This paper studies Predictive Prefetching on a Wide Area Network with two levels of caching. The WAN that we refer to is the GRNET academic network in Greece. We rely on log files collected at the network's Transparent cache (primary caching point), located at GRNET's edge connection to the Internet. Our prefetching model bases its predictions on popularity ranking of passed requests. We present a "n-next most popular" approach used for prefetching on GRNET's architecture and provide preliminary results of our experimental study, quantifying the benefits of prefetching on the WAN.

1 Introduction

Web Prefetching has been proposed mainly as a complementary procedure to caching, due to limitations in the performance of caching [1]. The work in [2], [3] present useful overviews of caching and prefetching. The benefits of prefetching have been explored in various Internet configurations, including client/server [4], client/proxy/server [1], [5], [6] and client/mediator/server [7], [8] systems. In this paper we present a study on how prefetching can be performed on a Wide Area Network with three levels in its caching hierarchy. A Transparent cache on the edge of the WAN to the Internet and local Proxy servers on the edge of the backbone.

A prediction algorithm is at the heart of any prefetching system. The Prediction by Partial Match (PPM) algorithm, which originates in the data compression community, has been explored in depth. In [4], [7] PPM is used to create branches from historical URLs. Moreover, Data mining techniques and algorithms with Markov models have been proposed for predictions [9], [10], [11]. In [12] Padmanabhan and Mogul propose a method in which the server makes predictions while individual clients initiate pre-fetching. Finally [6] presents a popularity-based Top-10 approach to prefetching, which combines the servers' active knowledge of their most popular documents (their Top-10) with client access profiles.

Our work is based on a "n-next most popular" approach that uses access log data to predict future requests, based on most popular requests following a specified request.

This scheme uses a popularity-based algorithm quite similar to the one proposed in [6] by Markatos and Chronaki. However, the knowledge of the most popular documents from those found to have been requested after a given server's document is used, limiting predictions only to those pages that appear to be highly related to the currently displayed page. Furthermore, a page dependency threshold is used in a similar way to [12] in order to keep the amount of prefetched documents low, in case of bandwidth limitations. Finally, the computational and storage complexity of our algorithm is much lower than that of the more complex Markov, PPM or Data mining techniques, making it possible to serve our primary goal of applying prefetching which is to reduce the response times on the WWW.

In this paper we look at the case of several inter-connected LANs with the use of a broadband backbone that provides access to the Internet through a main access point. This is the case of the Greek Research Network, GRNET [13].

2 The n-Next Most Popular Approach

To predict a future request made by a client, we first need to build the access profile for this client. The Transparent cache log data is used for that reason. Analysis of log data focuses on the frequency of visits and the sequence of requests. These specify the preferences of users and imply their future behavior. Log data processing includes popularity ranking of requested pages, frequency of content change estimation and page dependency examination. All these procedures are carried out for every client separately and result in the construction of different popularity lists for each client. These popularity lists are then used by a prediction algorithm to compute which pages are most likely to be requested next and by an additional decision algorithm that decides whether prefetching will be applied or not, and how many pages are going to be prefetched, based on bandwidth limitations determined by available resources. If an overall approach is followed, both the prediction algorithm and the decision algorithm use general popularity lists extracted by adding up popularity data from all the separate client-based popularity lists.

Popularity ranking: The basic goal of log data analysis is finding the most frequently requested pages after each page. We look for pages that were accessed within n accesses after a specified page. The parameter n is called lookahead window size. Any page requested within n accesses after a specified page was requested is considered to be a n-next page of it. In order for a page to be counted as a n-next page of an examined page, it also needs to have been requested within the same user session that the examined page has been requested. For every page in the log data we find the frequency of visits within the lookahead window size, of all other pages found to be a n-next page to it.

The value of the lookahead window size needs to be large enough to extend the applicability of the prefetching algorithm and small enough to avoid abusing the system and network resources available. Our study of Transparent cache log data reveals that pages accessed more than 5 accesses after a specified page are not highly related to this page, we choose the lookahead window size value to be equal to 5.

Initially, the process of n-next popularity ranking is carried out for every client separately. For each web page that has been requested by the client, the pages that had been requested within n accesses after it are stored. The instances of any of these pages as n-next of the examined page are calculated and the pages are ranked according to their popularity as n-next of the examined page. Thus, a n-next popularity list is created for every page requested by the client (client-based n-next popularity ranking). Putting the results from all clients together, we build a general n-next popularity list for every page logged, which maps the web behavior of the general population for that page (overall n-next popularity ranking).

Frequency of change: As in the case of n-next popularity ranking, the process of finding a page's frequency of change as n-next of a specified web page is carried out both for every client separately and overall. In the first case, frequency of change is estimated only for those times the page has been requested by the specific client within n accesses after the examined page. In the second case, all occurrences of the page as n-next of the examined page are taken into account. Frequency of change values are kept for every page in the proportional field of the appropriate n-next popularity list of the specified page.

Page dependency: The accuracy of prediction of the next request to be made by the client is affected by the extent of relation between pages. If a page that is a candidate for prefetching, is highly dependent of the currently displayed page, then its prediction as a client's next request, has a high probability of being correct. Dependency is defined as the ratio of the number of requests for a page as n-next of a specified page (stored in a n-next popularity list), to the total number of requests for the specified page (stored in a page popularity list).

Prediction algorithm: To predict a future request of a client, we use a simple algorithm that is based on n-next popularity data. Suppose a client is currently displaying a page. To predict his next request we rely on the client's request history for the currently displayed page. The pages that have the best chance of being requested after the currently visited page are those that were most frequently visited as n-next of it in the past. Thus, the pages, which are at the top of the n-next popularity list of the currently displayed page, appear to be candidates for prefetching.

The number of pages that are going to be predicted as candidates for prefetching is specified by the parameter m, which is called prefetching window size. The prediction algorithm suggests the m first web pages of the n-next popularity list of the currently displayed page as the most probable pages to be accessed next. The prefetching window size is a parameter of the prefetching scheme. A large value of m results in many pages being prefetched, which increases the number of successful prefetching actions. However, more bandwidth is required to perform prefetching then, resulting in a considerable increase of network traffic.

Decision algorithm: The "n-next most popular" prefetching model that is proposed in this paper uses a decision process, which determines whether or not prefetching will be applied and how many pages will be prefetched. Prefetching is decided for any page suggested by the prediction algorithm. We characterize this decision policy as an aggressive prefetching policy. However, when available bandwidth is limited we need to restrict our model and perform prefetching only for those predicted pages that appear to have a high chance of being actually requested. Those are pages that are highly dependent to the currently displayed page or pages whose content seems not to change frequently. This decision policy is called strict prefetching policy. In this case, the decision algorithm checks the following parameters for any page proposed by the prediction algorithm: its size, its dependency to the current page and its rate of content change, and decides, regarding bandwidth limitations, whether prefetching of that page would be advantageous or not. If the size of the page is larger than the average size of all visited pages, then a possible unsuccessful prefetching action performed for this page would affect network traffic dramatically. So prefetching is decided for such a page only if the dependency and the frequency of change values estimated for it satisfy the thresholds used to assure that prefetching of this page is very probable to be successful.

3 Results

In order to evaluate the performance benefits of our prefetching scheme we use trace-driven simulation. Access logs from the GRNET Transparent cache are used to drive the simulations. The results presented in this paper are based on logs of web page requests recorded over a 7-day period. In all experiments, 80% of the log data is used for training (training data) and 20% for testing (testing data) to evaluate predictions. Furthermore, all traces are preprocessed.

The performance metrics used in our experimental study:

- Prefetching Hit Ratio is the ratio of prefetched pages that the users requested (useful prefetched pages) to all prefetched pages. It represents the accuracy of predictions.
- Usefulness of Predictions is the ratio of prefetched pages that the users requested (useful prefetched pages) to all requested pages. It represents the coverage (recall) of predictions.
- Prefetch Effectiveness is the ratio of requests that are serviced from prefetched documents to the total number of requests for which prefetching is performed. This value is different from that of Usefulness of Predictions as only requests for which prefetching is applied are taken into account.
- Network Traffic Increase is the increase in network traffic due to unsuccessful prefetching. It represents the bandwidth overhead added, when prefetching is employed, to the network traffic of the non-prefetching case.
- Average Rank is the average rank of prefetch hits (in cases of successful prefetching action) in the set of predicted pages (or in the n-next popularity list of the active request).

The experimental scenarios for the evaluation of the “n-next most popular” prefetching scheme’s performance, as these derive from the alternative values of the parameters mentioned above, are:

1. Aggressive policy, m = 5, client-based and overall prediction
2. Aggressive policy, m = 3, client-based and overall prediction
3. Strict policy, m = 5, client-based and overall prediction

For every request examined the actual request that was made just after it (by the client) is checked from the simulation log data. This request is compared to the page suggested for prefetching by the "n-next most popular" algorithm proposed. If the actual request was the one predicted, then this request is counted as a prefetch hit. Otherwise a prefetch miss is logged and traffic overhead for the unsuccessful prediction is estimated. In the first case the rank of the successfully predicted page is also found in the n-next popularity list in order to calculate the average rank for all useful prefetched pages.

Table 1 shows that when the n-next popularity data is obtained from the general population, instead of client log data, prefetch effectiveness is a bit higher (54%). However, this requires an 18% traffic increase. This is expected, since in the case of overall prediction there is greater availability of n-next popularity data. Therefore, prefetching is performed for more requests. As a result, the cost in bandwidth is greater, but prefetch effectiveness is higher. If traffic increase is limited to 8%, then prefetch effectiveness is found equal to 50%. It is clear that for the same traffic increase the performance results of client-based prediction are better since client data implies more accurately the future web behavior of the user connected to this client than data extracted from all clients does.

It is clear that a small value of the prefetching window size provides better accuracy of predictions. Actually, the less documents a client is allowed to prefetch, the higher its prefetching hit ratio will be, as only highly probable objects are going to be prefetched. When practicing simulation for a smaller value of the prefetching window size (m = 3) and client-based prediction, we experience a significant increase of hit ratio (58%). In addition, less bandwidth is required. However, usability of predictions is lower (25%), as less prefetching actions are performed. Table 1 shows results taken for all three cases of client-based prediction.

Table 1. Performance results for client-based prediction

	Aggressive policy, m = 5	**Aggressive policy, m = 3**	**Strict policy, m =5**
Hit Ratio	48%	58%	51%
Usefulness of predictions	27,5%	25%	27%
Prefetch Effectiveness	52%	42%	51%
Network Traffic Increase	6%	4%	4%
Average Rank	2	1	2

A comparison of the performance results for the different prefetching policies in the case of client-based prefetching is also depicted in Figure 1. Figure 2 (a,b) compares performance results of client-based and overall prefetching applied at the Transparent cache. As we saw earlier network traffic overhead is much more in the case of prefetching based on the general population than in the client-based scenario.

Figure 1 shows that the recall of the algorithm is greater when a larger prefetching window size is used or a more aggressive prefetching policy is carried out, as in both cases more prefetching actions are being performed and therefore more useful pages are pre-sent. The use of a smaller prefetching window size appears to limit the coverage of prefetching method more than the use of a more strict policy, but this results in a significant increase of the accuracy of predictions (58% for the aggressive policy with m=3 compared to 51% for the strict policy with m=5).

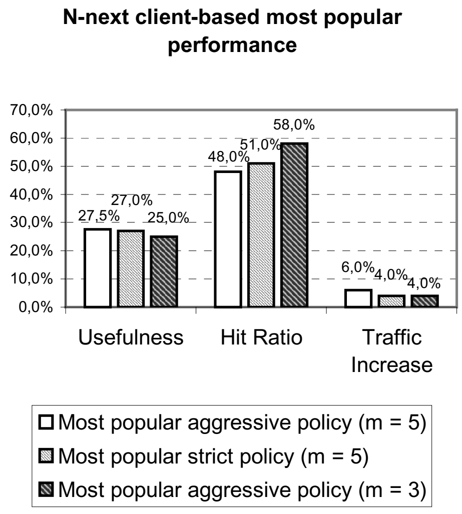

Fig. 1. Comparison of client-based policies for different performance metrics.

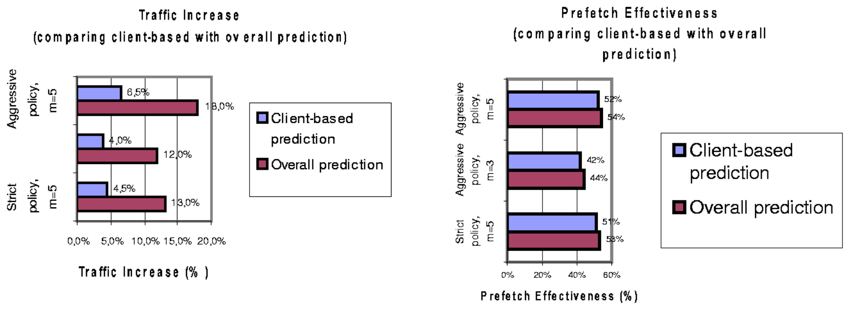

Fig. 2. Comparison of client-based and overall prediction scenarios for all policies (graphs should be read in pairs of bar charts)

As we mentioned earlier in this paper, a basic motivation for applying a prefetching scheme is to reduce the delay that an end user (client) experiences when requesting a Web resource. The computational cost of the "n-next most popular" algorithm presented in this paper is very low, since a simple algorithm, with no special operational or storage requirements, is used for the construction of the "n-next most popular" prediction model. This results in an efficient improvement of response times experienced by users as no significant time is needed to perform predictions about which pages to prefetch.

The application of prefetching on a higher level, the level of a Wide Area access point, offers the opportunity to use additional amount of bandwidth. Therefore, more predictions can be made, resulting in the increase of the number of useful predictions. The impact of this increase on the accuracy of predictions and the network traffic increase is not so significant in the case of a WAN due to greater bandwidth availability. The "n-next most popular" approach appears to have high usefulness of predictions, taking advantage of the available bandwidth when performing prefetching on a WAN. For the case of the "most popular" aggressive policy with prefetching window size equal to 5, for example, the usefulness of predictions is equal to 27,5%. It is found that if a more complex prefetching algorithm, a PPM algorithm with proportional values for its parameters with the case of the "n-next most popular" algorithm mentioned above, was used on the same Wide Area architecture the usefulness of predictions would be no more than 24%. Furthermore, the difference in the traffic increase between the two methods is insignificant for a WAN. The "n-next most popular" algorithm appears to add only 2% more traffic increase than the PPM algorithm does. This shows the advantage of applying the "n-next most popular" prefetching algorithm in the Wide Area, since it manages to profit a lot from prefetching by effective use of the extra bandwidth that is available on a WAN.

All results studied in the above paragraphs clearly show that the application of prefetching in the Wide Area can be quite beneficial. Even with the use of a simple prediction algorithm, as the "n-next most popular" algorithm proposed in this paper, the accuracy of predictions can reach 58% (case of aggressive, user-based policy with prefetching window size equal to 3) with an insignificant for a WAN increase of traffic network equal to 4%.

In fact, performance results are better, if we take into account that many of the prefetched requests will be used by more than a single end user, as prefetching in many cases is performed for ICP requests made by Proxy servers, which in turn serve many individual clients connected to them.

4 Future Work and Conclusions

In this work we did not study the prefetching of dynamically constructed resources such as search engine results or parameterized pages. The study of whether dynamic content may be included in prefetching, is very "attractive". This study must include the Web resource frequency of change problem in order to provide adequate results. Another important open issue is the creation of an algorithm that would prioritize ICP requests to the Transparent cache over direct TCP requests. This idea is based on the observation that ICP requests originate from Proxy servers and TCP requests originate from single users. The prioritization of ICP requests in prefetching intuitively means that the resulting prefetched resource could potentially be useful to more than one clients since it would reside on a proxy server. We also intend to study the usefulness of prefetched objects in the case of ICP requests

Prefetching can be highly beneficial to the reduction of User Perceived Latency. In this paper we argue that prefetching can be more efficient if it is applied at the edge network connection of a WAN. This approach can be more efficient than applying client initiated prefetching because of the more efficient use of available bandwidth and because prefetching at this point may be useful to many clients.

In this work we have shown that if we employ an "n-next most popular" approach and find that prefetching can be potentially beneficial to the GRNET WAN. Of course many further issues must be explored, before deploying prefetching on the edge of GRNET. Preliminary results provide a clear indication that response times would be significantly improved in GRNET if a simple "most popular" prefetching policy was performed at the Transparent cache.

References

1. Kroeger, T., M., Long, D., D., E., Mogul, J., C.: Exploring the Bounds of Web Latency Reduction from Caching and Pre-fetching: Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS), Monterey, CA (1997) 13-22
2. Wang, J. : A survey of web caching schemes for the Internet: ACM Computer Communication Review, 29(5), (1999) 36-46
3. Wang, Z., Crowcroft, J.: Prefetching in World Wide Web: Proceedings of the IEEE Global Internet 96, London, (1996) 28-32
4. Palpanas, T., Mendelzon, A.: Web Prefetching Using Partial Match Prediction: Proceedings of the Web Caching Workshop, San Diego, CA, USA, (1999)
5. Chen, X., Zhang, X.: Coordinated data prefetching by utilizing reference information at both proxy and Web servers: ACM SIGMETRICS Performance Evaluation Review, Volume 29, Issue 2, (2001) 32-38
6. Markatos, E., P., Chronaki, C., E.: A Top-10 Approach to Pre-fetching the Web: Proceedings of INET '98 (The Internet Summit), Geneva, Switzerland, (1998)
7. Fan, L., Cao, P., Jacobson, Q.: Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '99), Atlanta, GA, (1999) 178-187
8. Loon, T., S., Bharghavan, V.: Alleviating the latency and bandwidth problems in WWW browsing: Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97), Monterey, California, USA, (1997) 219-230
9. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: Effective Prediction of Web-user Accesses: A Data Mining Approach: Proceedings of the Workshop WEBKDD, San Francisco, CA, (2001)
10. Bestavros, A.: Using speculation to reduce server load and service time on the WWW: Proceedings of the 4th ACM International Conference on Information and Knowledge Management, Baltimore, Maryland, (1995) 403-410
11. Zukerman, I., Albrecht, D., W., Nicholson, A., E.: Predicting Users' Requests on the WWW: Proceedings of the 7th International Conference on User Modeling, Banff, Canada, (1999) 275-284
12. Padmanabhan, V., Mogul, J.: Using Predictive Prefetching to Improve World Wide Web Latency: Computer Communication Rev., 26(3), (1996) 22-36
13. GRNET, Web Site: http://www.grnet.gr/