Graph based Prediction Model to Improve Web Prefetching

P.Venketesh
Assistant Professor (SG)
Department of CIS
PSG College of Technology
Coimbatore, India

R.Venkatesan
Professor and Head
Department of CSE
PSG College of Technology
Coimbatore, India

ABSTRACT

Web prefetching is an effective technique used to mitigate the user perceived latency by making predictions about the user's future requests and prefetching them before the user actually demands them. In this paper, we present an algorithm that learns from user access patterns and builds a Precedence Graph (PG) that is used to generate the predictions. The difference in the relationship between objects of the same web page and the objects of different web pages are reflected in the graph implementation. It uses simple data structure to implement the graph, which is cost effective and consumes less computational resources. The proposed approach significantly improves the performance of web prefetching by utilizing limited amount of resources as compared to other existing algorithms used for prefetching.

General Terms

Web Mining, Prefetching

Keywords

Web prefetching, web prediction, access latency, precedence graph

1. INTRODUCTION

The enormous growth of World Wide Web and the development of new applications and services have dramatically increased the number of users, resulting in increased global traffic that damages the quality of service (availability, reliability, security) and the latency perceived by the users. Web prefetching is a technique that is widely used to reduce the user perceived latency. It exploits the spatial locality inherent in the user's accesses to the web objects. It uses prediction techniques to process the past user requests and generate useful hints, which are used to download the web objects and store it in cache before they are actually requested by the users. The task of predicting user's navigational behavior has been used in various applications such as web caching, web page recommendation, web search engines and personalization. Prefetch systems are designed to generate the web predictions based on different criteria such as access patterns, object popularity and structure of accessed web documents. Based on the type of content used for gathering the web information, they are categorized as: web content, web structure and web usage mining techniques.

In the basic web architecture that comprises of server, proxy and client, the prediction and prefetching engine can be located in any component to deliver the prefetch service to the users. In most cases [4, 11, 23, 24], the web server holds the prediction engine and is responsible for generating the predictions and providing it to the clients. The client holds the prefetching engine that uses the predictions to decide the web objects to be prefetched during the browser idle time. When user selects a hyperlink in the currently viewed web page or types a URL for a new web page, and if the requested web page is available in the cache due to prefetching, then it reduces the web page access time observed by the user.

In this paper, we present a prediction model that builds a Precedence Graph (PG) from user access patterns to predict the future accesses of users. The precedence relation is observed by considering the object URI and referrer in each user request; i.e. referrer representing the source of link precedes the requested web object. The algorithm also considers the characteristics of current websites when constructing the graph by differentiating the dependencies between objects of the same web page from objects of different web pages. The graph uses simple data structure for its implementation and provides good prefetching performance with minimal resource consumption.

The remaining of this paper is organized as follows. Section 2 presents the previous related work. Section 3 describes the methodology for building the Precedence Graph that is used for generating the predictions. Section 4 discusses the experimental environment and section 5 analyzes the experimental results. Finally, section 6 presents conclusion of the paper.

2. RELATED WORK

Several research activities has been carried out in the concept of web prefetching, which aimed to minimize the user perceived latency by prefetching web objects before the user requested them. The important task in web prefetching is to build an effective prediction model and the data structure for storing selective historical information. A server side prefetching approach that used dependency graph to represent the access patterns of users and make predictions based on it was presented in [1]. The server dynamically updated the dependency graph based on the client access patterns. To characterize the user behavior and predict user's next actions, two models were presented in [2] based on random walk approximation and digital signal processing techniques. In [3] a methodology was presented to build sequence prefix tree (path profile) based on the HTTP requests from the server logs. It used the longest matched most-frequent sequences to predict the user's next requests. A top-10 prefetching approach was proposed in [4] that allowed servers to push popular documents to the proxies at regular intervals based on the client's aggregated access profiles. In [5] a proxy-initiated prefetching technique

was proposed that used Prediction-by-Partial-Matching (PPM) algorithm to generate the predictions.

The prefetching of web documents based on the estimated round-trip retrieval time was presented in [6]. The technique was limited to prefetch only the static web pages. In [7] a model based on data mining concept was proposed to capture the user navigation behavior patterns, where the high probability strings represented the user's preferred trails. Use of Markov chains to build a probabilistic sequence generation model was proposed in [8], which predicted next requests based on the history of access requests from clients. Based on the popularity of URL access patterns a PPM model was designed in [9] that generated accurate predictions and efficiently managed the storage space than the standard PPM model. In [10] a framework was presented to demonstrate the web prefetching algorithms based on Markov predictors. A novel non-intrusive web prefetching system was developed in [11] that utilized only the spare resources to perform prefetching and thereby avoided interference with the demand requests from user.

The use of prefetching in a wide area network (WAN) was presented in [12], which suggested that if applied at the edge network connection it provided maximum efficiency. In [13] several techniques were proposed to select parts of different order Markov models to create a new model with reduced state complexity and improved prediction accuracy. New models based on Markov probabilistic techniques were built in [14] to examine the issues when predicting the web requests. It considered the information from user access history and web page content to accurately predict the user's next request.

The impact of web prefetching architecture in reducing the user perceived latency was analyzed in [15]. It addressed two main issues: a) Identifying best architecture to perform prefetch and b) providing insight into the efficiency of system. In [16,17] a prediction algorithm based on Double Dependency Graph (DDG) was proposed that considered the characteristics of current web sites to improve the web prefetching performance. It was based on the Dependency Graph (DG) algorithm [1], but able to differentiate the dependencies between objects of the same page and objects of different pages. A new web prediction algorithm based on Referrer Graph (RG) was proposed in [18] as a low-cost solution to predict user's next access. The graph was built using the user's access requests by considering the Object URI and the referrer in each request. It minimized the number of arcs used in the graph compared to DG and DDG mechanisms.

In [19] a cost-benefit analysis was carried out to compare the prefetching algorithms from the user's view point. Performance difference among prediction algorithms were mainly due to the size of predicted objects. A new approach called Prediction at Prefetch was proposed in [20] that allowed prediction algorithm to provide hints for both standard object requests and the prefetching requests. In [21] a novel PPM prediction model was developed based on stochastic gradient descent to perform web prefetching. It considered various factors such as page access frequency, prediction feedback, context length and conditional probability.

3. PREDICTION MODEL

The proposed prediction model builds a Precedence Graph (PG) to represent the user access patterns by creating arcs (links) from one node (web page) to one or more other nodes (web pages). Each arc in the graph has a transition weight associated with it that represents the association between the predecessor and successor nodes. When user requests for a particular web page, it is used to update the graph by adding a new node or arc; else increment the transition weight of existing arc and occurrence count of the nodes. The graph is then used to provide predictions (hints) for a user request by analyzing the transition weights of arcs associated with this request. It considers the arcs with transition weights greater than the threshold value to supply the predictions. The prediction (hint) list is then provided to the prefetching engine residing at the client machine to prefetch the web objects from the server during browser idle time.

3.1 Precedence Graph

The graph structure is built using algorithm (Figure 1) that learns from user access patterns to predict the future accesses.

Input:

- Precedence Graph (PG)
- Requested object & Referrer in the request
- Object MIME type

Output:

- PG with updated information

Step 1:

Adding new node (or) Updating existing node
x→ node
Find *x* Є graph that matches with the requested object.
If *x* available, then
    x occurrence ← *x* occurrence + 1
else
    x ← new node representing the requested object
    x occurrence =1
end if

Step 2:

Adding new arc (or) updating existing arc
y→ node
Find *y* Є graph that matches with the referrer in user request.
If *y* available, then
    Find arc *yx*, transition from node *y* to *x*
    If *yx* available, then
        yx occurrence ← *yx* occurrence + 1
    else
        yx ← new arc from y to x
        yx occurrence = 1
    end if
else
    no arc added or updated in the graph
end if

Compute the transition confidence of all arcs from node *y*
arc transition confidence ← arc occurrence / *y* occurrence

Figure 1: Algorithm for building Precedence Graph

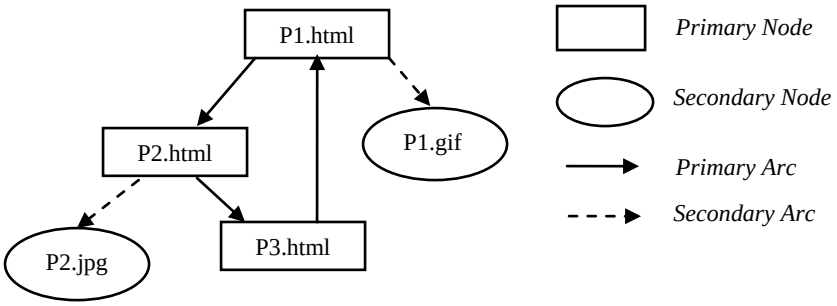

Figure 2: Precedence Graph

Each user requested web object is represented as a node in the graph and its occurrence count get incremented each time the user requests the same web object. The arcs between the nodes represent the transition from one web object to another. A web object (B) will be requested by the user from a source web object (A) i.e. B depends on A (A → B, A precedes B). The HTTP referrer recorded in the access log file for each user request represents the source of the web object being requested. Based on the referrer information available for each web request, an arc is created from the source node (referrer) to the requested node (successor). The arc weight represents the transition confidence of moving from the predecessor node to the successor node.

A web page consists of a main object (demand requested by the user) referred to as the primary and many embedded objects called as secondary objects. The graph shown in Figure 2 contains two types of node: primary and secondary. The primary nodes are used to represent the web objects that are demand requested by the users, while the secondary nodes are used to represent the embedded web objects that are requested by the web browser. The arcs between two primary nodes are termed as primary arcs and those between the primary and secondary nodes are termed as secondary arcs.

The graph is initially empty and is built and updated through a learning process. Each node in the graph maintains information such as: object URI, node type, occurrence count, list of primary and secondary arcs. Each arc maintains information such as: destination URI, arc type, occurrence count and transition confidence. The node occurrence indicates the number of requests to the represented web object and the arc occurrence indicates the number of requests to the successor node from the predecessor node. The arc transition confidence is computed by dividing the arc occurrence with predecessor node occurrence.

The graph is dynamically updated whenever new user requests are received and it involves the following steps:

a) Increment the node occurrence if the user requested web object finds a matching node; else a new node representing the web object is added with its occurrence count initialized to one.

b) Increment the arc occurrence if it already exists in the graph; else a new arc is added with its occurrence count initialized to one.

The graph size is restricted by removing the nodes that are least representing the user access sequence and not important to the prediction process.

3.2 Prediction process

When user demands a web page identified by its URI, the web browser first requests the primary object of that page and then gets the secondary objects from the server or from the local cache. The perfect prediction algorithm [22] needs to report three main types of hints for each web page: primary object of the next web page to be requested by the user, secondary objects present in that next page, and the next pages.

In our model, the predictions (hints) are generated for a user requested web object if it finds a matching node in the graph; else no hints are generated. The steps for generating the hint list are:

a) Find primary node in the graph that matches with the user request.

b) If primary node exists, analyze all the primary arcs associated with that node. Arcs having transition confidence greater than or equal to the threshold value are selected.

c) Object URI's stored in the primary nodes linked to the arcs selected in step (b) are added to the hint list and arranged based on their confidence value (high to low).

d) Analyze the secondary arcs associated with the primary nodes used in step (c). Arcs having transition confidence greater than or equal to the threshold value are selected.

e) Object URI's stored in the secondary nodes linked to the arcs selected in step (d) are added to the hint list and arranged based on their confidence value (high to low).

f) Hint list now comprises of the object URI's of both primary and secondary nodes. It is given as input to the prefetching engine for downloading the web objects.

3.3 Prefetching

When prefetching engine receives the hint list, it automatically starts downloading the hinted web objects during the browser idle time to avoid interfering with regular user requests. It determines whether a web object is eligible for caching by checking its MIME type in the HTTP response header. The prefetched objects are stored separately in a prefetching cache without disturbing the existing objects in regular cache, which helps to improve the hit rate.

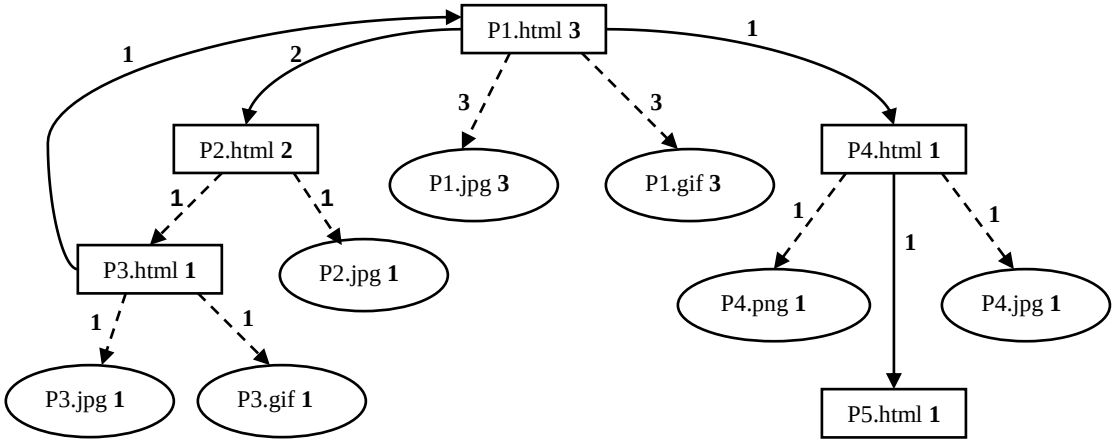

Figure 3: Precedence Graph for the user requests in Table 1

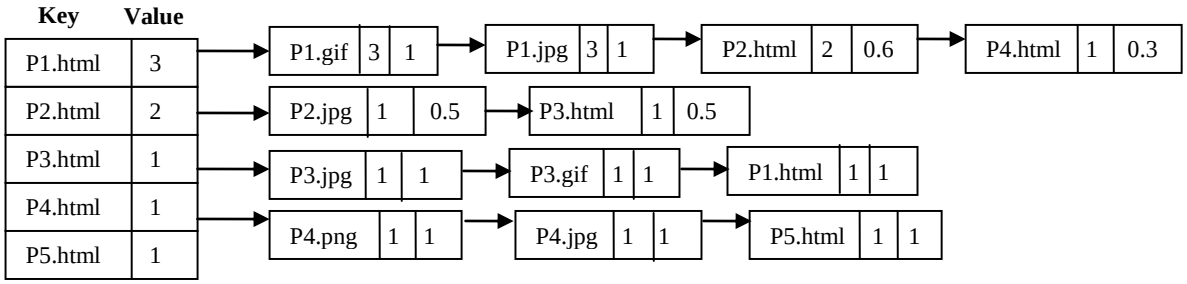

Figure 4: Precedence Graph implemented as Adjacency Map

If a hint is already available in the regular or prefetch cache, then it will not be prefetched. When a new web page is demand requested by the user, then prefetching of the hints get terminated. When client does not use prefetching, then the requested web object will be served from the local (regular) cache or proxy cache or from the web server.

3.4 Working Example

The working of proposed prediction model is illustrated using sample web requests in a client session as shown in Table 1. The precedence graph generated using the web requests of Table 1 is shown in Figure 3. The primary and secondary nodes are represented with Object URI and their occurrence count. The primary and secondary arcs are represented with their occurrence count.

The graph shown in Figure 3 is implemented using the adjacency map as shown in Figure 4, with primary nodes of the graph stored as keys in the map and the edges (arcs) starting from each node stored as a list associated with appropriate key in the map. The secondary nodes of the graph are not stored as keys in the map, since in most cases they don't act as the source for a new web link. The elements of the list are represented with three fields: Object URI is represented in the first field, occurrence count of the element is represented in the second field and the transition confidence of the list element is represented in the third field.

$$\text{Transition Confidence} = \frac{\text{arc occurrence count}}{\text{node occurrence count}}$$

i.e. p1.gif = 3/3 =1, p2.html = 2/3 = 0.6, p4.html = 1/3 = 0.3

Table 1: sample web requests in a client session

Requested URI	Referrer of URI
/P1.html	-
/P1.gif	/P1.html
/P1.jpg	/P1.html
/P2.html	/P1.html
/P2.jpg	/P2.html
/P3.html	/P2.html
/P3.jpg	/P3.html
/P3.gif	/P3.html
/P1.html	/P3.html
/P1.gif	/P1.html
/P1.jpg	/P1.html
/P4.html	/P1.html
/P4.png	/P4.html
/P4.jpg	/P4.html
/P5.html	/P4.html
/P1.html	-
/P1.gif	/P1.html
/P1.jpg	/P1.html
/P2.html	/P1.html

The hints generated for the user requests based on the available information in the graph is shown in Table 2.

Table 2: Hints generated for the user requests

Requested URI	Hints generated
/P1.html	/P2.html, /P2.jpg, /P4.html, /P4.png, /P4.jpg
/P2.html	/P3.html, /P3.jpg, /P3.gif
/P3.html	/P1.html, /P1.gif, /P1.jpg
/P4.html	/P5.html

4. EXPERIMENTAL ENVIRONMENT

The framework used for evaluating the proposed model and the workload used for constructing the graph and generating the predictions are discussed in this section.

4.1 Evaluation Framework

The framework consists of web server with the prediction engine and client with the prefetching engine. Web server builds the graph using web access log data and then generates the predictions for user requests. Client gets the predictions from web server and uses it to download the web objects during browser idle time. To simulate the set of users accessing the web server, real or synthetically generated web traces are fed to the client that uses prefetching enabled web browser. The time interval between two successive web requests obtained from the timestamp values recorded in the log file is used to mimic the actual client behavior. During simulation, basic information related to each web request and its response is recorded in a log file. The performance metrics (precision, recall) are computed by analyzing the log file after completing the simulation.

4.2 Workload Description

Web users exhibiting different access patterns during website navigation are stored in web access log files. These log files can be maintained at three different points in the web architecture: server, proxy and client. The log files maintained at the web server is used as the main data source in most of the research activities. We collected log file from our institutional web server that reflect the users accessing the institutional web site to obtain various information like academics, administration, examination details and news articles.

The web logs are preprocessed to reformat them for effectively identifying the web access sessions and use it for building the graph and generating the predictions. The first task of preprocessing is to perform data cleaning by removing redundant and useless records from web log file and retain only valid information related to the visited web pages. The entries removed from log file during the data cleaning operation are: a) Requests executed by automated programs such as web robots, spiders and crawlers b) Requests with unsuccessful HTTP status codes and c) Entries with request methods except “Get”. The second task is to perform session identification by segmenting the long sequence of web requests into individual user access sessions. Each user session consists of sequence of web pages visited over a period of time. If the user remains idle for more than 30 minutes without making any request, then the next request from the user is considered as the start of new access session.

5. RESULTS

The performance of the proposed Precedence Graph based predictions is discussed in this section by considering metrics such as recall and precision. When the user requested web object is served from the prefetching cache, it indicates prefetch hit else it is prefetch miss. The prediction algorithm must provide meaningful hints for each web request to reduce the user access latency in a higher rate.

Prefetch hit ratio (Recall) indicates the ratio of total number of prefetch hits to the total number of user requests. It measures the usefulness of predictions. Prefetch accuracy (Precision) indicates the ratio of total number of prefetched pages requested by the user from the prefetch cache to the total prefetched pages.

$$\text{Recall} = \frac{\text{Prefetch Hits}}{\text{User Requests}}$$

$$\text{Precision} = \frac{\text{Prefetch Hits}}{\text{Prefetchs}}$$

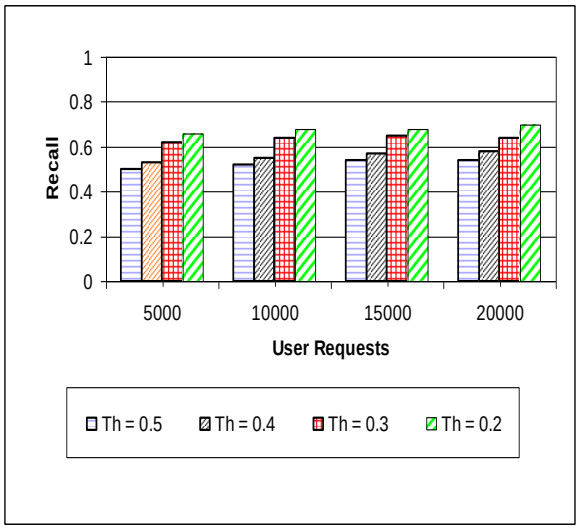

Figure 5(a): Recall

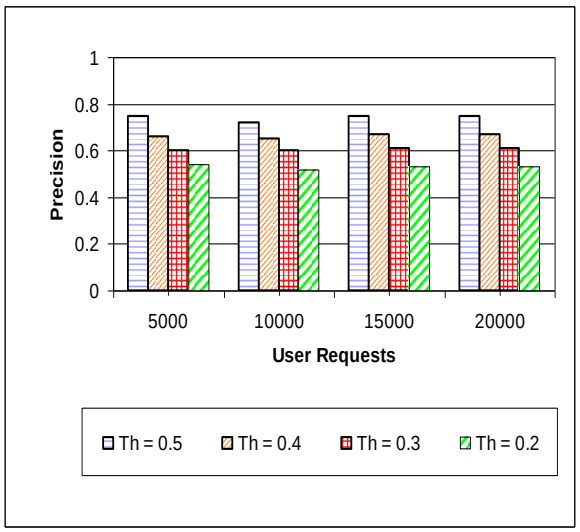

Figure 5(b): Precision

The recall and precision achieved for the user requests by varying the prediction threshold is shown in Figure 5 (a, b). Threshold value used for limiting the number of hints to be generated from the graph is varied from 0.5 to 0.2 for analyzing the prefetch performance. For threshold value of 0.2 the recall achieved is very high, since the graph generates more predictions resulting in more objects to be prefetched to satisfy the user requests. But the precision degrades due to the fact that some of the prefetched objects may not be requested by the user. For threshold value of 0.5, it achieves better precision with moderate recall performance.

The advantage of Precedence Graph (PG) is its ability to provide performance that matches with the existing algorithms such as DG [1] and DDG [16] with less computational requirements. It is due to the fact that the algorithm adds an arc between the nodes only based on its precedence relation i.e. successor node access depends on its predecessor node. This helps to reduce the number of arcs in the graph than DG and DDG algorithm. The number of arcs generated in the graph for different algorithms is shown in Figure 6, and it clearly indicates that the PG algorithm performs better than other algorithms. During implementation, only the primary objects are added to the adjacency map with key values. Secondary objects are only added as a link element to the respective key. It helps to avoid the wastage of key entry in the map for secondary objects that do not add any link elements.

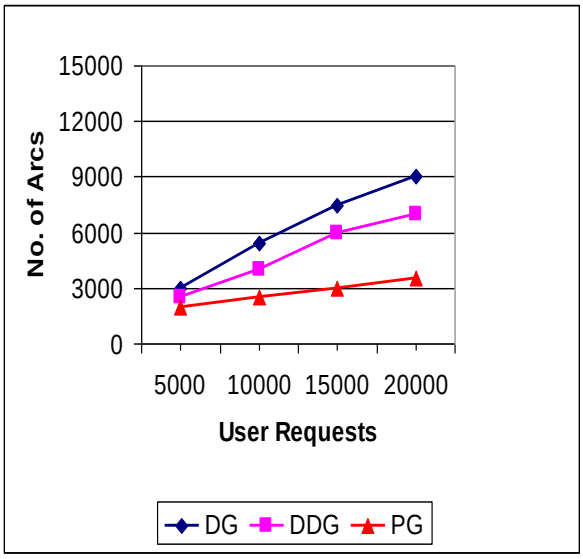

Figure 6: Number of Arcs in the Graph

The time taken by an algorithm to provide predictions for a user request depends on the total number of arcs in the graph. It is due to the fact that the algorithm requires time to analyze the arcs for generating the predictions. Since our graph structure has fewer arcs than the existing algorithms (DG and DDG), it is able to provide predictions in a shorter time duration.

6. CONCLUSION

In this paper we have presented a prediction model that built a Precedence Graph (PG) by considering the characteristics of current websites in order to predict the future user requests. The algorithm differentiated the relationship between the primary objects (HTML) and the secondary objects (e.g., images) when creating the prediction model. The graph structure was built with fewer arcs than the existing algorithms, because PG considered the precedence relation for each request instead of using the sequence of user accesses as recorded in a log file. Experimental results show that PG achieved good recall and precision values. It reduced user access latency with minimal resource consumption when compared to the existing algorithms.

7. REFERENCES

[1] V. Padmanabhan and J. Mogul, “Using Predictive Prefetching to Improve World Wide Web Latency,” Computer Communication Review, vol. 26, no. 3, pp. 22-36, July 1996

[2] C.R. Cunha and C.F.B. Jaccoud, “Determining WWW User’s Next Access and its Application to Prefetching”, Proceedings of International Symposium on Computers and Communication, pp. 6-11, July 1997

[3] S. Schechter, M. Krishnan, and M. Smith, “Using Path Profiles to Predict HTTP Requests,” Proceedings of 7th International World Wide Web Conference, also appeared in Computer Networks and ISDN Systems, vol. 20, pp. 457-467, 1998

[4] E.P. Markatos and C.E. Chronaki, “A Top-10 Approach to Prefetching on the Web”, Proceedings of INET ’98, July 1998

[5] L. Fan, P. Cao, W. Lin, and Q. Jacobson, “Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance”, Proceedings of SIGMETRICS ’99, pp. 178-187, May 1999

[6] R.P. Klemn, “Web Companion: A Friendly Client-Side Web Prefetching Agent”, IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 4, pp. 577-594, 1999

[7] J. Borges and M. Levene, “Data mining of user navigation patterns”, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1836, pp. 92–111, 1999

[8] R. Sarukkai, “Link Prediction and Path Analysis Using Markov Chains”, Proceedings of 9th International World Wide Web Conference, 2000

[9] X. Chen and X. Zhang, “A popularity-based prediction model for web prefetching”, IEEE Computer, march 2003

[10] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "A data mining algorithm for generalized web prefetching”, IEEE Transaction on Knowledge and Data Engineering, vol.15, no.5, pp.1 -16, 2003

[11] R. Kokku, P. Yalagandula, A. Venkataramani, and M. Dahlin,” NPS: A non-interfering deployable web prefetching system”, Proceedings of the USENIX Symposium on Internet Technologies and Systems, 2003

[12] C. Bouras, A. Konidaris, and D. Kostoulas, ”Predictive prefetching on the web and its potential impact in the wide area”, World Wide Web: Internet and Web Information Systems, vol.7, pp.143 –179, 2004

[13] M. Deshpande and G. Karypis, "Selective markov models for predicting web page accesses", ACM Transactions on Internet Technology, Vol.4, pp.163–184, 2004

[14] B. D. Davison, "Learning web request patterns", Web Dynamics: Adapting to Change in Content, Size, Topology and Use, Springer, pp. 435–460, 2004

[15] J. Domenech, J. Sahuquillo, J. A. Gil, and A. Pont, "The impact of the web prefetching architecture on the limits of reducing user's perceived latency", Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, 2006

[16] Josep Domenech, Jose A. Gil, Julio Sahuquillo, Ana Pont, "DDG: An Efficient Prefetching Algorithm for Current Web Generation", In Proceedings of the 1st IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), Boston, USA, 2006

[17] J. Domenech, J.A. Gil, J. Sahuquillo, A. Pont, "Using current web page structure to improve prefetching performance", Computer Networks, vol. 54, no. 9, pp. 1404 -1417, 2010

[18] B. de la Ossa, A. Pont, J. Sahuquillo and J. A. Gil, "Referrer Graph: a low-cost web prediction algorithm", in Proceedings of the 2010 ACM Symposium on Applied Computing, March 22-26, 2010

[19] J. Domenech, A. Pont, J. Sahuquillo, and J. A. Gil, "A user focused evaluation of web prefetching algorithms", Computer Communications, vol.30, no.10, pp. 2213-2224, 2007

[20] B. de la Ossa, J. A. Gil, J. Sahuquillo and A. Pont, "Improving Web Prefetching by making Predictions at Prefetch", proceedings of 3rd EuroNGI Conference on Next Generation Internet Networks, pp. 21-27, 2007

[21] Zhijie Ban, Zhimin GU, Yu Jin, "A PPM Prediction Model Based on Stochastic Gradient Descent for Web Prefetching", Proceedings of 22nd International Conference on Advanced Information Networking and Applications, pp.166-173, 2008

[22] B. de la Ossa, J. Sahuquillo, A. Pont, J. A. Gil, "An Empirical Study on Maximum Latency Saving in Web Prefetching," vol. 1, pp.556-559, IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, 2009

[23] Darin Fisher and Gagin Saksena, "Link prefetching in Mozilla: A server driven approach", in proceedings of the 8th International Workshop on Web Content Caching and Distribution (WCW 2003), New York, USA, 2003.

[24] Alexander P. Pons, "Improving the performance of client web object retrieval", Journal of Systems and Software, vol.74, issue.3, 2005.