An Efficient Approach For Optimal Prefetching To Reduce Web Access Latency.

Dinesh Kumar, Reena Patel

ABSTRACT: The exponential growth and popularity of WWW increases the amount of traffic which results in major congestion problems over the available bandwidth for the retrieval of data. This results in the increase of user perceived latency. Prefetching of web pages is a potential area that can significantly reduce the web access latency. It refers to the mechanism of deducing the forthcoming page accesses of a client. Prefetching reduces the user's perceived latency but on the contrary it increases the traffic that may result in further congestion,. So the major concern of the prefetching is to device an algorithm that could efficiently and optimally prefetch the pages so that the traffic load is minimized. In this dissertation an optimal prefetching algorithm is proposed which gives the optimal number of web documents to be prefetched to reduce latency. The algorithm is based on the current content of the web documents so there is no requirement of maintaining past history of the users and is also beneficial for first retrieval of access of web resources.

Keywords: Latency, Cache, Prefetching, Apriori, Hyperlinks, Congestion, Data Retrieval, Optimal Prefetching

1. INTRODUCTION

Data Mining and WWW are two important and active areas of current researches. WWW is a large distributed hypertext repository of information where people navigate through links and view pages through browsers and Data Mining is used to discover the hidden facts contained in databases. A natural combination of the two areas i.e. Data Mining and WWW is referred as Web Mining. Web mining is the application of data mining techniques to automatically discover and extract information from web documents and services. It has been the focus of several recent research projects and papers.

2. MOTIVATION FOR PREFETCHING

Prefetching complements the research in caching, since all benefits of prefetching are in addition to those of caching. While caching attempts to provide fast access to a document for the second time it is accessed, prefetching provides fast access to a document even at the first time it is being accessed. Moreover, the already existing infrastructure for caching can be exploited for prefetching.

3. METHODOLOGY USED IN THE STUDY

In order to accomplish the proposed work, the following methodology is used:

- *Dinesh Kumar, Reena Patel*
- *MTech Scholar of Computer Science & Engineering, DITMR*
- *Department of Computer Science & Engineering, DITMR*
- *Email: dinesh.sorout@gmail.com, rinapatel27@gmail.com*

3.1 Association Rule

Mining for associations among items in a large database is an important database mining function. Association rules are used to show relationship between data items. For example the purchasing of one product when another product is purchased represents an association rule. The terminology used for Association Rule:

- Itemset or dataset: It is a combination of items e.g. {Mouse, keyboard}.
- Support: It is the number of transactions in a dataset where itemset appears. Let A and B are itemsets. If A $\Rightarrow$ B Then

$$\text{Support } (A \Rightarrow B) = \frac{\text{no. of tuples containing both A and B}}{\text{total no. of tuples}}$$

- Confidence**:** It is the percentage of transactions containing A itemset that also contain B itemset. If A $\Rightarrow$ B Then

$$\text{Confidence } (A \Rightarrow B) = \frac{\text{no. of tuples containing both A and B}}{\text{no. of tuples containing A}}$$

- Minimum support and Minimum confidence: It is some predefined minimum value for support and confidence.
- Large or frequent itemset: Itemsets that have a support exceeding the threshold i.e. minimum support

3.2 Apriori Algorithm

- Apriori [3][5] is a breadth-first or generate and test type of algorithm. Apriori is an association rule-based data mining technique that finds the association between the items of the database. The Apriori algorithm iterates over the database in multiple passes. However the itemsets having support equal to or greater than Minsup are selected for the next pass and this process continues until an item set with

maximum number of items along with Minsup is achieved. Minsup is a user-specified parameter.

- Apriori algorithm starts from each item and then checks its support and rejects the item with support less than the minimum support, then add one more item with previous item one by one and again checks the support and this process continues until we find largest item set with support greater than the minimum support.

Association Rule [5] mining involves finding all the rules that satisfy user-defined constraints on minimum support and confidence with respect to a given dataset. For example, a particular supermarket may find that out of the 1000 customers shopping on a Thursday night, 200 bought diapers, and of those 200 who bought diapers, 50 bought beer. Thus, the association rule would be "If buy diapers, then buy beer" with a support of 200/1000 = 20% and a confidence of 50/200 = 25% The common approach to finding association rules is to break the problem into two parts:

- Find large itemsets that have a support exceeding the threshold i.e. minimum support.
- Generate rules from frequent itemsets.

There are various Association Rule discovery algorithms available that efficiently discover the frequent itemsets. One such algorithm is Apriori algorithm.

4. PROBLEMS WITH EXISTING PREFETCHING TECHNIIQUES

Several researchers have designed various prefetching schemes. There are some limitations in the existing approaches for prefetching. The some existing prefetching schemes prefetches the web documents that may further result in huge increases in network traffic that may leads to network congestion. Also it may decrease the bandwidth available that again increases the time required to access documents that were not prefetched. This makes the situation worsen. It means prefetching is always at danger of crossing the borderline from making things better to making things worse. Thus it is required that prefetching must be done in a controlled way. Also the some existing approaches of prefetching are history based and history of the clients has to be maintained at client side, proxy side or server side. Also for history based approaches in order to achieve high hit-rates, a long observation time is required. Also it is difficult to react to new trends in user behaviour immediately using these approaches. In this dissertation an optimal prefetching algorithm is proposed to limit the number of documents to be prefetched based on the relevancy of the keywords supplied by user that may not require past history. It is a non-statistical prefetching method which uses an algorithm based on data mining techniques to optimally prefetch the web documents. It is based on the current content of the web documents so there is no requirement of maintaining past history of the users and is also beneficial for even first retrieval of access of web resources. The algorithm predicts the next user request and prefetched those referenced pages whose words in the anchor text are found in the user's supplied keyword list. Unlike previous techniques, this approach can also prefetch web documents who are never accessed before and does not require history access of users that makes it good adaptively to the change of user surfing interests.

5. PROPOSED ARCHITECTURE FOR OPTIMAL PREFETCHING SYSTEM

A client side architecture is proposed that addresses the prefetching of hyperlinks of current web document. This approach includes criteria which are determined by the keywords supplied by user. In this the collection of the user-specific criteria is accomplished by the keywords supplied by user. The keywords are collected from the keywords supplied by user and added to user-specific keywords list. As soon as a web document is retrieved, the prediction engine (Fig. 4.1) parses the document, builds a list of current hyperlinks. The keywords in this list are compared to a user-specific keywords list to optimally prefetch the hyperlinks from the current web page or document. Each time a request is made by the client as well as keywords are supplied by the client, same process is applied to optimally prefetch the hyperlinks.

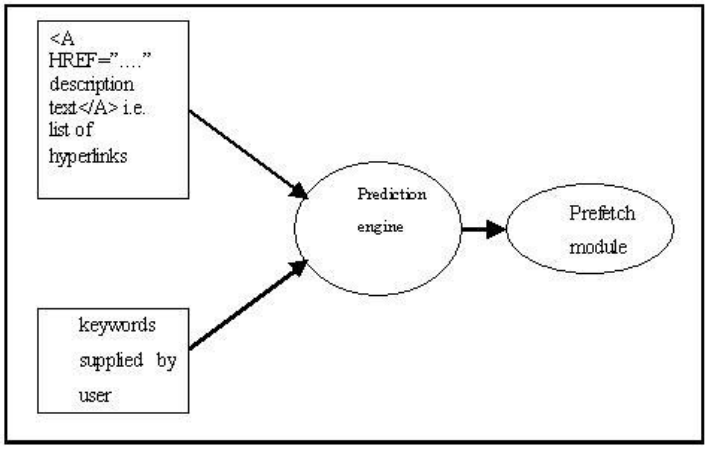

Fig. 5.1 The Prediction Engine compares Keywords from Hyperlinks with the User Specified Keyword List

5.1 MODULES OF THE PROPOSED ARCHITECTURE

The basic architecture of optimal prefetching system is shown in Fig. 5.2. It contains the following modules:

1) Prefetch module

The prefetch module is responsible for receiving request from client, serving request to the client and prefetching the hyperlinks according to the list of hyperlinks supplied by the prediction module and puts them into the prefetch cache. This module also sends the details of the keywords that are supplied by the user and the requested web document to the Extraction module.

2) Prediction Engine

It consists of two modules that are Extraction module and Prediction module:

a) Extraction module

The main task of this module is to collect the keywords that are supplied by the user and to put them into a user specified keyword list. The requested web document is also scanned by this module to prepare the list of hyperlinks that contains the keywords from the user specified keyword list. The list of all hyperlinks that contains any keyword from the user specified keyword list is prepared by this module.

b) Prediction module

This module compares the keywords in the user specified keyword-list with the list of keywords describing the hyperlinks on the actual HTML page using the optimal perfecting algorithm and sends the list of the hyperlinks to the prefetch module.

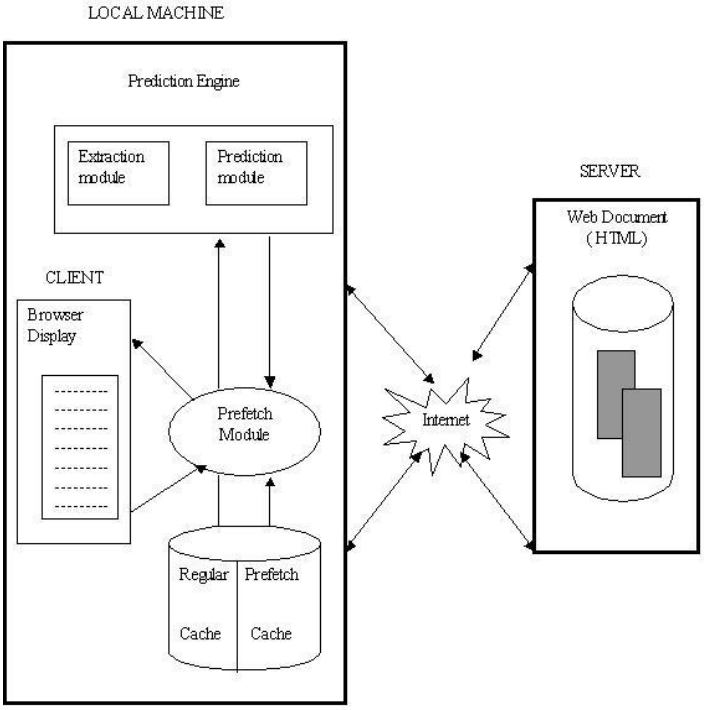

Fig. 5.2 Proposed Architecture of Optimal Prefetching System

5.2 PROPOSED ALGORITHM FOR OPTIMAL ALGORITHM

Input:

1) The list of 'm' hyperlinks of the current web page.
2) Minimum support i.e. 'minsup' which is used to control the number of hyperlinks to be prefetch.
3) User-specified keyword list

Output:

1) The optimal list of hyperlinks that are to be prefetched i.e. 'L' which is large set of hyperlinks when support >= minsup

Variables:

1) 'n': integer which contains total number of unique keywords
2) $i_1, i_2 \text{---------} i_n$ are set of unique keywords from list of hyperlinks
3) 'k': integer
4) 'termination': Boolean
5) 'support[]' which contains the support value for the keywords which is calculated with the help of formula
6) 'C' which contains the set of keywords that are candidate to be prefetched and $C_1, C_2 \text{----------} C_k$ are the candidate keyword set which contains set of keywords of length k with their support value.

Functions used :

1) count() function which takes the keyword set as input and gives the total number of hyperlinks in which the particular keyword set appeared from the list of hyperlinks.

Algorithm optimal_prefetching ()

{

1. Count unique individual keywords from list of hyperlinks that matches with the user-specified list by scanning all the hyperlinks once and say it as 'n'.
2. for j=1 to n do
   {
   Compute support[i_j]=count(i_j)/m by scanning all the hyperlinks once and
   counting the number of hyperlinks that keyword i_j appears in (i.e.
   count(i_j)).
   }
3. Now, create the candidate1 keyword set i.e. C_1which will be the set of keywords $i_1, i_2 \text{---------} i_n$ with their support value.
4. for j=1 to n do
   {
   Compute the L_1 which contains the subset of keywords from C_1
   where support(i_j)>=minsup.
   }
5. Let k=1 and termination = false
6. Repeat steps (a) to (e) until termination=true
   {
   a. Let L_{k+1}=empty.
   b. Create the candidate (k+1) keyword set i.e. C_{k+1} by combining members of L_k by selecting and extending k-keyword set by one more keyword so that set of keywords will be unique and no keyword will repeat in each set.
   c. In addition, only consider as keywords of C_{k+1} those k+1 keywords such that every subset of size k appears in L_k.
   d. Scan the hyperlinks once and compute the support for each member of C_{k+1}. If the support for a member of C_{k+1}>=minsup then add that member to L_{k+1}
   e. If L_{k+1} is empty then
   termination=true
   else
   k=k+1
   }
7. Now L_k contains the list of hyperlinks that are to be prefetch
8. End.

}

The optimal_prefetching algorithm takes hyperlinks of the current web page and user-specified keyword list as input as well as minsup to control the number of hyperlinks to be prefetch. Firstly it will count unique individual keywords from list of hyperlinks that matches with the user-specified list by scanning all the hyperlinks once Then for each unique individual keyword it will compute support by scanning all the hyperlinks once and counting the number of hyperlinks that keyword appears in. Now it will create the candidate1 keyword set i.e. C_1which will contain set of keywords of length 1 with their support value. Again for each individual keyword it will compute the L_1 i.e. Large set of hyperlinks which contains the subset of keywords from C_1 where support is greater than equal to minsup. Now by letting k=1 , termination = false and L_{k+1}=empty it will create the candidate (k+1) keyword set i.e. C_{k+1} by combining members of L_k by selecting and extending k-keyword set by one more keyword so that set of keywords will be unique and no keyword will repeat in each set. In addition, only consider as keywords of C_{k+1} those k+1 keywords such that every subset of size k appears in $L_{k..}$ It scan the hyperlinks once and compute the support for each member of $C_{k+1.}$ If the support for a member of C_{k+1}>=minsup then add that member to $L_{k+1.}$ If L_{k+1} is empty then termination=true else k=k+1. Same steps are going to repeat if termination is false otherwise it will give L_k as output which contains set of keywords for the list of hyperlinks that are to be prefetch.

4.3 EXAMPLE

List of hyperlinks :

1) keyword-based semantic prefetching approach
2) keyword-based semantic prefetching
3) keyword-based semantic prefetching approach
4) keyword-based semantic prefetching approach
5) semantic prefetching approach
6) semantic prefetching
7) keyword-based prefetching
8) semantic approach
9) keyword-based semantic prefetching approach
10) keyword-based prefetching approach

Total no of hyperlinks i.e. m=10

Unique keywords from hyperlink list that matches with the user-specified list are

1) keyword-based ie. k
2) semantic i.e. s
3) prefetching i.e. p
4) approach i.e. a

Total no of Unique individual keywords i.e. n=4

Minimum support i.e. minsup = 4/10 i.e. .4

Step 1 :

C1

Keyword set	Support value
k	.7
s	.8
p	.9
a	.7

Compare support value with minsup

L1

Keyword set	Support value
k	.7
s	.8
p	.9
a	.7

Step 2:

C2

Keyword set	Support value
ks	.5
kp	.7
ka	.5
sp	.7
sa	.6
pa	.6

Compare support value with minsup

L2

Keyword set	Support value
ks	.5
kp	.7
ka	.5
sp	.7
sa	.6
pa	.6

Step 3:

C3

Keyword set	Support value
ksp	.5
ksa	.4
kpa	.5
spa	.5

Compare support value with minsup

L3

Keyword set	Support value
ksp	.5
ksa	.4
kpa	.5
spa	.5

Step 4:

C4

Keyword set	Support value
kspa	.4

Compare support value with minsup

L4

Keyword set	Support value
kspa	.4

Step5:

C5

Keyword set	Support value

Compare support value with minsup

L5

Keyword set	Support value

Now L5 is empty so L4 contains the set of keywords for the list of hyperlinks that are going to be prefetch.

L4

Keyword set	Support value
kspa	.4

4.4 Result

So hyperlinks that are going to be prefetch are given in bold and underlined i.e. only 4 hyperlinks are to be prefetch form 10 hyperlinks:

1) keyword-based semantic prefetching approach
2) keyword-based semantic prefetching
3) keyword-based semantic prefetching approach
4) keyword-based semantic prefetching approach
5) semantic prefetching approach
6) semantic prefetching
7) keyword-based prefetching
8) semantic approach
9) keyword-based semantic prefetching approach
10) keyword-based prefetching approach

4.5 ADVANTAGES OF THE PROPOSED ALGORITHM

The Algorithm optimal_prefetching leads to the following benefits:

1. It reduces perceived latency time by integrating prefetching with caching.
2. It reduces network traffic while using prefetching as optimal number of web documents are prefetched using this algorithm.
3. This algorithm for prefetching uses well defined threshold to make sure that only a small number of useful documents is prefetched. Thus, it cannot get out of control and lead to traffic chaos.
4. It prefetchs only those hyperlinks that contains the maximum keywords means the future documents client will probably want to access. Thus, the risk of bringing useless documents is minimized.
5. It is based on the current content of the web documents so there is no requirement of maintaining past history of the users. So there is no overhead for maintaining past history.
6. It can be used to catch new trends immediately as prefetching is based on the relevancy of the keywords supplied by user.
7. It is beneficial for prefetching of new documents that are not ever accessed by the client means useful for even first retrieval of access of web resources.

5. CONCLUSION AND FUTURE WORK

Prefetching is used to anticipate probable future requests and to fetch the most probable documents before they are actually requested. The goal of prefetching is to reduce user perceived latency on the Web. But if prefetching is not done cautiously it may further result in increases in traffic that also leads to network congestion. Thus it may decrease the bandwidth available that again increases the time required to access documents that were not prefetched. This makes the situation worsen. It means prefetching is always at danger of crossing the borderline from making things better to making things worse. Thus it is required that prefetching must be done in a controlled way. In this dissertation an optimal prefetching algorithm is proposed to reduce user perceived latency time without the extraneous penalty in network load because optimal number of documents are prefetched. It relies on the keywords supplied by the user to predict user's future requests. It considers that the hyperlinks appropriate for prefetching come from the current user page. The algorithm predicts the next user request and prefetched those referenced pages whose words in the anchor text are found in the user's supplied keyword list. Unlike previous techniques, this approach can also prefetch web documents who are never accessed before and does not require history access of users that makes it good adaptively to the change of user surfing interests. The proposed algorithm for optimal prefetching leads to the various benefits. It reduces perceived latency time by integrating prefetching with caching. It reduces network traffic as this algorithm prefetches optimal numbers of web documents. This algorithm for prefetching uses well defined threshold to make sure that only a small number of useful documents is prefetched. Thus, it cannot get out of control that leads to traffic chaos.It prefetchs only those hyperlinks that contains the maximum keywords means the future documents client will probably want to access. Thus, the risk of bringing useless documents is minimized. It is based on the current content of the web documents so there is no requirement of maintaining past history of the users. Thus there is no overhead for maintaining past history. It can be used to catch new trends immediately as prefetching is based on the relevancy of the keywords supplied by user. It is beneficial for prefetching of new documents that are not ever accessed by the client means useful for even first retrieval of access of web resources. Future work can be done to incorporate thesauri inside extraction module to increase the hit rate and to have better decision about keywords which may be entered in the user specified keyword list.

REFERENCES

[1]. T. Berners-Lee1, R. Cailliau2, N. Pellow3: The World-Wide Web Initiative

[2]. Tim Berners-Lee, Robert Cailliau : World-Wide Web

[3]. Sule Gunduz (2003): Recommendation Model for Web Users: User Interest Model and Click Stream Tree.

[4]. Daniel T. Larose : Discovering Knowledge in Data: An Introduction to Data Mining

[5]. Osmar R. Zaïane, 1999 : Principles of Knowledge Discovery in Databases

[6]. Agrawal R. and Srikant R. (1994): Fast Algorithm for Mining Association Rules

[7]. Rakesh Agarwal , Manish Mehta : The quest data mining system

[8]. :Navathe : Fundamental of database system

[9]. G. Piatetsky-Shapiro, U. M. Fayyad, and P. Smyth. From data mining to knowledge discovery: An overview.

[10]. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM

[11]. M. S. Chen, J. Han, and P. S. Yu. Data mining: An overview from a database perspective., 1996.

[12]. G. Piatetsky-Shapiro and W. J. Frawley. Knowledge Discovery in Databases.

[13]. J. Han and M. Kamber. Data Mining: Concepts and Techniques

[14]. Azizul Azhar Bin Ramli: Web Usage Mining using Apriori Algorithm: UUM Learning Care Portal Case.

[15]. R.Cooley, B.Mobasher and J.Srivastava. Web Mining: Information and Pattern Discovery on the World Wide Web, 1997

[16]. Srivasta, J., Cooley, R. Deshpande, M., and Tan P. N. (2000). Web Usage Mining: Discovery and Application of Web Usage Pattern from Web Data.

[17]. Raymond Kosala, Hendrik Blockeel : Web Mining Research: A Survey

[18]. S. Schechter, M. Krishnan, M.D. smith : Using path profiles to predict HTTP requests.

[19]. C.M. Brown, P.B. Danzig : The Harvest Information Discovery and Access System

[20]. E. Spertus. Parasite : Mining structural information on the web

[21]. R.B. Doorenbos, O. Etzioni and D.S. Weld : A scalable comparision shopping agent for the WWW.

[22]. W.B. Fakes and R. Baeza-Yates : Information reterival data structures and algorithms

[23]. M. Pazzani,J. Muramatsu and D. Billsus : Syskill&Webert: Identifying interesting web sites.

[24]. S. Chakrabarti, B. Dom and P. Indyk : Enhanced hypertext categorization using hyperlinks

[25]. S. Brin and L. Page : the anatomy of a large-scale hypertextual web search engine

[26]. J.Borges and M. Levene. : Data Mining of User Navigation Pattern

[27]. L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web

[28]. Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns

[29]. Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a sow's ear: Extracting usable structures from the web.

[30]. R. Agrawal and R. Srikant. Fast algorithms for mining association rules, 1994

[31]. Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Creating adaptive web sites through usagebased clustering of urls.

[32]. Abdullah Balamash and Marwan Krunz (2004): A client Side WWW Prefetching Model.

[33]. J. Griffioen and R. Appleton, “Reducing file system latency using a predictive approach

[34]. christos bouras and agisilaos konidaris : Predictive Prefetching on the Web and its Potential Impact in theWide Area

[35]. T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web

[36]. D.S.W. Ngu and X. Wu. Sitehelper: A localized agent that helps incremental exploration of the world wide web.

[37]. H. Lieberman. Letizia: An agent that assists web browsing.

[38]. Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Creating adaptive web sites through usagebased clustering of urls.

[39]. T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking.

[40]. Olfa Nasraoui, Raghu Krishnapuram, and Anupam Joshi. Mining web access logs using a fuzzy relational clustering algorithm based on a robust estimator.

[41]. Evangelos Markatos, Main Memory Caching of Web Documents

[42]. Anawat Chankhunthod et al : A Hierarchical Internet Object Cache

[43]. Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the www.

[44]. S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests.

[45]. Charu C Aggarwal and Philip S Yu.: On disk caching of web objects in proxy servers.

[46]. Mike Perkowitz and Oren Etzioni.: Adaptive web sites:Automatically synthesizing web pages.

[47]. Mike Perkowitz and Oren Etzioni. Adaptive web sites: Conceptual cluster mining

[48]. Alex Buchner and Maurice D Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining.

[49]. Balaji Padmanabhan and Alexander Tuzhilin. A belief-driven method for discovering unexpected patterns.

[50]. L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web.

[51]. D. Duchamp, Prefetching hyperlinks, 1999.

[52]. T. Palpanas and A. Mendelzon. Web prefetching using partial match prediction.

[53]. R. Hugo Patterson : Informed prefetching and caching

[54]. Ken-ichi Chinen, “WWW Collector Home Page”

[55]. V.N. Padmanabhan, J.C. Mogul, “Using Predictive Prefetching to Improve World Wide Web Latency”

[56]. Azer Bestavros, Using Speculation to Reduce Server Load and Service Time on the WWW.

[57]. L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance.

[58]. E.P. Markatos and C.E. Chronaki. A Top-10 Approach to Prefetching on the Web

[59]. Carlos Cuncha, Carlos Jaccoud, Determining WWW User’s Next Access and its Application to prefetching

[60]. Catledge Pitkow, Characterizing Browsing Strategies in the World Wide Web

[61]. D. Menasce, V. Almeida, R. Fonseca, and M. Mendes, “A methodology for workload characterization of e-commerce sites”

[62]. B. D. Davison, “Predicting Web actions from HTML content”