Volume 2, Issue 4, April 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Advance Testimony for Weblog Prefetching Data Mining

Indla kasthuri [1], M.A. Ranjit Kumar [2]
K. Sudheer Babu[3],Dr.S.Sai Satyanarayana Reddy [4]
[1]M.tech 2nd year
[2,3] Asst. Professor
[4] Professor and Head
Dept of CSE,LBRCE
Mylavaram, Krishna Dist.,A.P
ikast huri28@gmail.com

***Abstract*— Web pre-fetching is mechanism by which web server/clients can pre-fetch web pages well in advance before a request is actually received by a server or send by a client. The idea is, give a request, how accurately can you predict the next consequent request. This paper is about designing and implementing a pre-fetching model in the form of a Markov model and association rule mining. The Markov property states that, given the current state, future states are independent of the past states. The Web prefetching strategy also proposed association rule mining algorithm to discover the prefetched documents. It discovers dependencies between pairs of documents. A web server can cache the most probable next request reducing the time taken to respond to a request considerably that helps us to make up the web latency that we face on the Internet today.The vigilant implementation of these methods can reduce access time and latency making optimal usage of the servers computing power and the network bandwidth.**

***Keywords*— Web mining, Association Rule, Pre fetching, Data mining**

I. INTRODUCTION

1.Web Mining

Web mining, looked upon in data mining terms, can be said to have three operations of interests.

- **Clustering** – ex: finding natural grouping of users, pages etc.,
- **Association** – ex: which URLs tends to be requested together.
- **Sequential Analysis** – ex: the order in which URLs tend to be accessed

With huge amount of information available online, the World Wide Web is a fertile area for Data mining reach. Web mining research is at crossroads of research from several research communities, such as Database, information retrieval, and written Ai, especially the sub areas of machine learning and Natural Language Processing. Web mining is use of data mining techniques to automatically discover and extract information from web documents and services. This area of research is so huge today partly due to interest of various communities, the tremendous growth of information sources available on the web and recent interest in e-commerce when we ask what constitutes Web mining.

Web mining can be broken into following sub tasks:

- **Resource Finding:** Retrieving documents intended for web.
- **Information Selection and Processing:** Automatically selecting and pre processing specific information from resources retrieved from web.
- **Generalization:** To automatically discover general patterns at individual websites as well as across multiple sites.
- **Analysis:** Validation and/or interpretation of minded patterns.

There are several ways to make use of the following purposes:

Finding Relevant Information – To find specific information on web. Usually specify a simple keyword query and the response from a web search engine is a list of pages, ranked based on their similarities to query.

Discovering new knowledge from the web – Here, the user have a data triggered process that presumes which already have a collection of data and extract potentially useful knowledge out of it.

Personalized Web Page Synthesis – The user may wish to synthesize a web page for different individuals from the available set of web pages. Individuals have their own preferences in style of contents and presentations while interacting with the web. The information providers like to create a system which responds to user queried by potentially aggregating information from several sources, in manner which is dependent on user.

Learning About Individual Users – It is about knowing what the customers do and want. Inside this problem, there are sub problems, such as miss customizing the information to the intended customers or even personalizing it to the individual user. Problems related to effective website design and management, problems related to marketing etc.

There are several major challenges for Web Mining research:

1. Most web documents are in HTML format and contain many markup tags, mainly used for formatting. While web mining applications must parse HTML documents to deal with these markup tags, the tags also can provide additional information about the document.

For example, a bold type face markup (<b>) may indicate that a term is more important than other terms that appear in normal typeface. Such formatting cues have been widely used to determine the relevance of terms.

2. While traditional IR systems often contain structured and well written documents, this is not the case on the web. Web documents are much more diverse in terms of length, document structure, writing style and many web pages contain many grammatical and spelling errors. Web pages are also very diverse in terms of languages and domains; one can find almost any language and any topic on web. In addition, the web has many different types of content, including text, images, audios, videos and executables. There are numerous formats such as HTML,XML,PDF,MS Word,mp3,wav.ra,rm,avi,just to name a few. Web applications have to deal with these different formats and retrieve the desired information.

3. While most documents in traditional IR systems tend to remain static overtime, Web pages are much more dynamic; they can be updated everyday, every hour or every minute. Some web pages do not have a static form; they are dynamically generated on request, with content varying according to the user and the time of request. Such dynamics make it much more difficult for retrieval systems such as search engines to keep an up-to-date search index of the web.

Deduction of futures references on the basis of predictive prefetching can be implemented by having an engine, which after processing the past references derives the probability of future access for the documents accessed so far. The prediction engine can reside either in the client or in the server side. In the former case, it uses the set of past references to find correlations and initiates Prefetching. No modifications need to be made to the current Web infrastructure(ex: HTTP protocol, web servers) or to the Web browsers if the prefetcher module runs as a proxy in browser.

The main limitation of this approach is that the clients, in general, lack of sufficient information to discover the correlations between the documents since their requests cover a broad range of Web servers and an even broader range of documents. On the other hand, Web servers in better position to make predictions about future references since they log a significant part of request by all internet clients for the resources they own.

The main drawback of the latter approach is that additional communication between the server and the client is needed in order to realize the Prefetching scheme. This scheme can be implemented by either the dissemination of predicted resources to the client or exchange of messages between server and clients, having the server piggybacking information about the predicted resources on the regular response messages, avoiding establishment of any new TCP connections. Such a mechanism has been implemented and seems the most appropriate, since it requires relatively few enhancements to the current request-response protocol and no changes to the HTTP 1.1 protocol.

In what follows in this article , we assume that their is system implementing a server based predictive prefecther, which piggybacks its predictions as hints to its clients.

The way users navigate in the Website depends not only on their interest, but also on the site structure. More precisely , a user , who has currently selected document D , chooses the document to visit next mainly among the set of the links contain in D. This choice, in general is based on the previously visited documents (i.e., D and any other documents visited before D).

Otherwise, the user is randomly exploring the site, not seeking for particular information. The former case induces dependencies between the visited documents of the site. If these dependencies correlate only pairs of documents, then they are called **first ordered dependencies**, otherwise they are called **higher ordered dependencies**. This model of user navigation describes a Markovian process over the graph whose nodes are the document of the site and arcs are the links between the documents.

In contrast, large sites, i.e., with a large number of documents and fairly high connectivity (that resemble the traditional hypertext databases), present navigational alternatives, hence the impact of these parameters is significant. These kinds of sites is expected to become more popular in the years to come automated by using tools like Areneus and Strudel that automatically generates site based on the contents of underlying databases and find applications in service providing, like e-commerce. Since the performance

requirements for this type of Web sites significantly increased, prefecthing can be very beneficial for them.

Web mining is a very hot research topic, which combines two of the activated research areas:

- Data Mining
- World Wide Web

The Web mining research relates to several researches communicates such as Database, Information Retrieval and Artificial Intelligence.

Although Web mining puts down the roots deeply in data mining, it is not equivalent to data mining. The unstructured feature of Web data triggers more complexity of Web mining. Web mining research is actually a converging area from research communities, such as Database, Information Retrieval, Artificial Intelligence, and also Psychology and statistics as well.

1.1 Web Usage Mining-Preprocessing

- techniques are always necessary to eliminate the impact of the irrelevant items to the analysis result.
- The input data may include Web server logs, referral logs, registration files, index server logs, and optionally usage statistics topology and page classification.

1.2 Client/Server Architecture

The original PC networks were based on file sharing architectures, when the server downloads files from the shared location to the desktop environment. The requested user job is then run(including logic and data) in the desktop environment. File sharing architectures work if shared usage is low, update contention. File sharing architectures work if shared usage is low. PC LAN (local area network) computing changed because the capacity of the file sharing was strained as the number of online user grew (it can only satisfy about 12 users simultaneously) and graphical user interfaces (GUIs) became popular (making mainframes and terminal displays appear out of date). PCs are now being used in Client/Server architectures.

As a result of the limitations of file sharing architectures, the Client/Server architecture emerged. This approach introduced a database server to replace the file server. Using a Relational DataBase Management Systems (DBMS), user queries could be answered directly. The client/Server architecture reduced network traffic by providing a query response rather than total file transfer. It improves multi-user updating through a GUI front end to a shared database. In Client/Server architectures, Remote Procedure Calls (RPCs) or Standard query language (SQL) statements are typically used to communicate between the client and server.

Alternatives to Client/Server architectures would be mainframe or file sharing architectures. Complementary technologies for Client/Server architectures are computer-aided software engineering (CASE) tools because they facilitate Client/Server architectural development and open systems because they facilitate development of architectures that improve scalability and flexibility.

II. LITERATURE SURVEY

The long-term success of the World Wide Web depends on fast response time. People use the Web to access information from remote sites, but do not like to wait long for their results. The latency of retrieving a Web document depends on several factors such as the network bandwidth, propagation time and the speed of the server and client computers. Although several proposals have been made for reducing this latency, it is difficult to push it to the point where it becomes insignificant. This motivates our work, where we investigate a scheme for reducing the latency perceived by users by predicting and prefetching files that are likely to be requested soon, while the user is browsing through the currently displayed page. In our scheme the server, which gets to see requests from several clients, makes predictions while individual clients initiate prefetching. Our results indicate that prefetching is quite beneficial in both cases, resulting in a significant reduction in the average access time at the cost of an increase in network traffic by a similar fraction. We expect prefetching to be particularly profitable over non-shared (dialup) links and high-bandwidth, high-latency (satellite) links.

The growth of the World Wide Web has emphasized the need for improved user latency. Increasing use of dynamic pages, frequent changes in the site structure, and user access patterns on the internet have limited the efficiency of caching techniques and emphasized the need for prefetching. Since prefecthing increases bandwidth, it is important that the prediction model is highly accurate and computationally feasible. It has been observed that in a web environment, certain sets of pages exhibit stronger correlations than others, a fact which can be used to predict future requests. Previous studies on predictive models are mainly based on pair interactions of pages and TOP-N approaches. In this paper we study a model based on page interactions of first order where we exploit set relationships among the pages of a web site. We find that the model based on first order page interaction is more robust and gives competitive performance in a variety of situations.

Benefit:

Due to the fast development of internet services and a huge amount of network traffic, it is becoming an essential issue to reduce World Wide Web user-perceived latency. Although web performance is improved by caching, the benefit of caches is limited. To further reduce the retrieval latency, web prefetching becomes an attractive solution to this problem. Prefetching reduces user access time, but at the same time, it requires more bandwidth and increases traffic. Performance measurement of prefetching techniques is primarily in terms of hit ratio and bandwidth usage. A significant factor for a prefetching algorithm in its ability to reduce latency is deciding which objects to prefetch in advance. This paper presents a solution space of prefetching

according to various object-selecting criteria and a comparison of their performance is provided.

With the background information furnished above we now proceed to the technical part of this project which begins with the definition of the problem and in-depth analysis which is followed by the design and implementation and finally concluded this through testing of the coded application.

III. BACK GROUND

To reduce the user web accesses latency by the concept of the Web Prefetching.

A challenging problem for Internet computing is how to reduce the access latencies. With the increase in both server and client types, the number of unique file objects cached in client browsers continues to multiply, and the web accesses behavior is becoming more erratic.

Approaches that relay solely on caching offer limited performance improvement because it is difficult for caching to handle the large number of increasing diverse files. A promising solution to the Web access latencies is to combine caching with Web Prefetching – obtaining the Web data, which a client is expected on the basis of data about that client's past surfing activity. The study shows that performance improvement with caching and Prefetching can be twice that of caching alone.Web Prefetching, makes Prefetching decisions by reviewing the URLs that client's on a particular server have accessed over some period. The ultimate goal of Web Prefetching is to reduce what is called User Perceived Latency on the Web. User Perceived Latency is the delay that an end user (client) actually experiences when requesting a Web resource.

The first solution that was investigated toward the reduction of latency was the caching of Web documents at various points in the network (client, proxy and server). Caching capitalizes on the temporal locality. Effective client and proxy caches reduce the client perceived latency, the server load, and the number of traveling packets, thus increasing the available bandwidth.

Several caching policies have been proposed during the previous years, especially for proxy servers. Nevertheless, the benefits reaped due to caching can be limited when Web resources tend to change very frequently, resources cannot be cached (dynamically generated Web documents), they contain cookies (this issue matters only caching proxies), and when request streams do not exhibit high temporal locality.

The negative effects of the first problem can be partially alleviated by employing some, though costly, cache consistency mechanism.

The second problem could be addressed by enhancing the cache with some of the respective server's query processing capabilities, so as to perform the necessary processing on data. The third and fourth problems seem to not be tackled by caching at all.

As a further improvement in the situation, the technique of prefetching has been investigated. Prefetching refers to the process of deducing a client's future requests for Web objects and getting those objects into the cache, in the background, before an explicit request is made for them. Prefetching capitalizes on the spatial locality, that is,
Correlated references for different documents present in request streams [1], and exploits the client's idle time, i.e., the time between successive requests.

The main advantages of employing prefetching are that it prevents bandwidth underutilization and hides part of the latency. On the other hand, an overaggressive scheme may cause excessive network traffic. Additionally, without a carefully designed prefetching scheme, several transferred documents may not be used by the client at all, thus wasting bandwidth. Nevertheless, an effective prefetching scheme, combined with a transport rate control mechanism, can shape the network traffic, reducing significantly its burstiness and, thus, can improve the network performance.

In general, there exist two prefetching approaches. Either the client will inform the system about its future requirements or, in a more automated manner and transparently to the client, the system will make predictions based on the sequence of the client's past references. The first approach is characterized as informed prefetching and concentrates around the ideas developed as part of the so-called Transparent Informed Prefetching, where the application discloses its exact future requests and the system is responsible for bringing the respective objects into the buffer. In the design of a prefetching scheme for the Web, its specialties must be taken into account.

Two characteristics seem to heavily affect such a design:

1) The client server paradigm of computing the Web implements
2) Its hyper textual nature.

Therefore, informed prefetching seems inapplicable in the Web since a user does not know in advance its future requirements, due to the "navigation" from page to page by following the hypertext links. Source anticipation i.e., the prefetching of all (or some part thereof) of the embedded links of the document, may work in some cases, but seems inappropriate in general, because there is no a prior information about which of a large set of embedded links the client is likely to request.

On the other hand, the second approach, called predictive prefetching, is more viable, especially under the assumption that there is sufficient spatial locality in client

requests because such a prefetching method could use the history of requests to make predictions.

The secret of authority hides in Web page linkages. These hyperlinks contain an enormous amount of latent human annotation that can help automatically infer the notion of authority. When a Web page's author creates a hyperlink pointing to another Web page, this action can be considered as an endorsement of that page .The collective endorsement of a given page by different authors on the Web can indicate the importance of the page and lead naturally to the discovery of authoritative Web pages. Thus the Web's linkages data provides a rich Web mining source.

This idea has routes in traditional publishing as well: In the 1970's researchers in information retrieval proposed methods for using journal article citations to evaluate the quality of research papers. The Web linkage structure has several features that differ from general citations, however.

- First, not every hyperlink represents the endorsement a search is seeking Web page author creates some links for other purposes, such as navigation or to serve as paid advertisement. Overall, though if most hyperlinks function as endorsements, the collective opinion will still dominate.
- Second, an authority belonging to commercial or competitive interest will seldom have its Web page point to rival authority's pages.

IV. RELATED WORK

In this paper, we focus on predictive prefetching. First, we identify two factors, i.e., the order of dependencies between page accesses and the noise which affects the method for calculating the appearance frequencies of user access sequences that characterize the performance of predictive Web prefetching algorithms. According to these factors, we present a framework which is used to describe Prefetching algorithms in terms of Markov predictors.

This framework allows for the formal examination of existing Web prefetching algorithms and the identification of equivalences or differences among them. Additionally, we develop a new algorithm that is formally proven to be a generalization of existing ones. An extensive analytical and experimental comparison of all algorithms, for the first time (the performance of existing algorithms have been examined only independently), indicates that the proposed algorithm outperforms existing ones by combining their advantages without presenting their deficiencies. Hence, the main contributions of this paper can be summarized as:

- A novel framework for a formal and unified description of web prefetching algorithms.
- A new Web prefetching algorithm that generalizes existing ones, and
- A detailed (analytical and experimental) comparison of all algorithms.

In today's common Web configurations, the proxy server exists between clients and Web servers. The proxy server intercepts the requests from clients, servers with the requested object if it is present in proxies, or retrieves new objects from the appropriate Web server, then caches the new objects or updates the existing objects if it has been modified since the last reference.

One way to further increase the cache hit ratio is to anticipate future requests and prefetch these objects into a local cache. On the other hand, prefetching consumes more network bandwidth. In this paper, by considering that network bandwidth usage is cost and hit ratio is the performance of prefetching, we introduce a concept which defines our nation of the value of prefetching by measuring the balance between the responses time improvement and the extra amount of system resources consumes by prefetching compared to demand caches.

We use threshold-based algorithms and prefetch those objects with values that exceed the threshold. The object selection criterion of prefetching determines the threshold. The object selection criterion of prefetching determines which object to replicate in advance. We have several options of criteria such as object popularity and the lifetime. Based on the analytical statistics model, we examine these prefetching criteria and obtain a solution space.

These analytical results provide a proof of concept for deterrent prefetching criteria, e.g., the criterion of lifetime favoring the long-lived objects results in low bandwidth cost but low hit ratio. The key contribution of our work is anew prefetching algorithm that balances object frequency and object update frequency. The algorithm can also be tuned to emphasis fetching long-lived popular objects rather than always maximizing benefit/cost balancing object lifetime vs. popularity.

4.1 Prefetching

Prefetching is a technique used to enhance overall computer system functions. It has been used to enhance operating systems, file systems and of course Web based systems. It is always useful to be able to foresee a request in such systems, in order to be able to service it before it is actually placed, since this world boost system performance. Prefetching systems always run the risk of misusing or abusing system resources in order to execute their functions.

4.2 Web Prefetching

Predictive Web Prefetching refers to the mechanism of deducting the forthcoming page access of a client based on its past accesses.

Prefetching refers to the process of deducing client's future requests for Web objects and getting that objects into the cache, in the background, before an explicit request is made for them. Prefetching capitalizes on the spatial locality present in request streams, that is, correlated references for different documents, and exploits the client's idle time, i.e., the time between successive requests. The main advantage of employing prefetching is that it prevents bandwidth underutilization and hides part of latency.

On the other hand, an overaggressive scheme may cause excessive network traffic. Additionally, without a carefully designed prefetching scheme, several transferred documents may not be used by the client at all, thus wasting bandwidth. Nevertheless, an attractive prefetching scheme, combined with a transport rate control mechanism, can shape the network traffic, reducing significant its burstiness and, thus can improve the network performance.

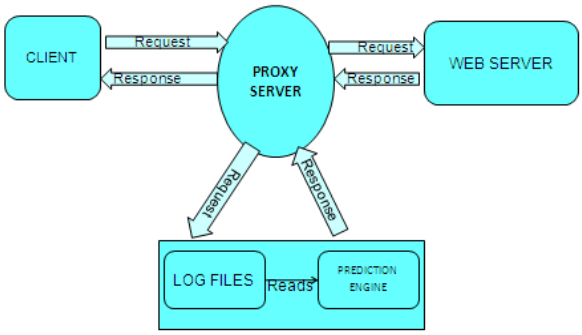


Fig 1: Architecture of web Server

4.3 Importance of Web Prefetching

The problem of predicting a user's behavior on a web site gained importance due to the rapid growth of the World-Wide-Web and the need to personalize and influence a user's browsing experience.

4.4 Web Prefetching as an Application of Data Mining

Data mining technology has emerged as a means of identifying patterns and trends from large quantities of data.

Data mining and data warehousing go hand-in-hand: most tools operate by gathering all data into a central site, then running an algorithm against that data. However, privacy concerns can prevent building a centralized warehouse-data may be distributed among several custodians, none of which are allowed to transfer their data to another site.

The benefit of Web prefetching is to provide low retrieval latency for users, which can be explained as high hit ratio. Prefetching also increases system resource requirements in order to improve hit ratio. Resources consumed by prefetching include server CPU cycles, server disk I/O's, and network bandwidth. Among them, bandwidth is likely to be the primary limiting factor. So, we add moderate bandwidth requirements as an important performance measure for a good web-prefetching model.

4.5 Proxy Cache

Proxy caches have become a central mechanism for reducing the latency of Wed documents retrieval. While caching alone reduces latency for previously requested documents, web document prefetching could mask latency for previously unseen, but correctly predicted requests.

We describe a prefetching algorithm suitable for use in a network of hierarchical web caches. This algorithm observes requests to a cache and its ancestors, and initiates prefetching for predicted future requests if prefetching is likely to reduce the overall latency seen by the cache's clients.

Web caching is recognized as one of the effective techniques to alleviate the server bottleneck and reduce network traffic, thereby reducing network latency. The basic idea is to cache recent requested pages at the server so that they do not have to be fetched again.

It attempts to guess what the next requested page will be. For regular HTML file accesses, prefetching techniques try to predict the next set of files/pages that will be requested, and use this information to prefetch the files/pages into the server cache. This greatly speeds up access to those files, and improves the user's experience.

To be effective however, the Prefetching techniques must be able to reasonably predict (with minimum computational overheads) subsequent web accesses.

Clustered accesses are access to clearly related pages. For example access to the pages of a single company or research group or to pages associated with the chapters of a book. Clustered accesses are very common and accounted for over70% of the accesses in the server logs that we studied.

Web Prefetching builds on web caching to improve the files access time at web servers. The memory hierarchy made possible by caches helps to improve HTML page access time. However, cache misses can reduce the effectiveness of the cache and increase this average time. Prefetching attempts to transfer data to the cache before it is asked for, thus lowering the cache misses even further.

Prefetching techniques can only be useful if they can predict accesses with reasonable accuracy and if they do not represent a significant computational load at the server. Note that prefetching files that will not be requested not only wastes useful space in the cache but also results in wasted bandwidth and computational resources.

4.6 Use of Web Prefetching

The ultimate goal of Web prefetching is to reduce what is called User Perceived Latency on the web. UPL is the delay that an end user (client) actually experiences when requesting Web resources. The reduction of User Perceived Latency does not imply the reduction of actual network latency or the reduction of network traffic.

On the contrary in most cases even when UPL is reduced, network traffic increases. The basic effect of prefetching on a Web system is to "separate" the time when a resource is actually requested by a client from the time the client (user in general) chooses to see the resource.

4.6.1 Server

A computer or device on a network that manages network resources. For example, a file server is a computer and storage device dedicated to storing files. Any user on the network can store files on the server.

Servers are often dedicated, meaning that they perform no other tasks besides their server tasks. On multiprocessing operating systems, however, a single computer can execute several programs at once. A server in this case could refer to the program that is managing resources rather than the entire computer.

4.6.2 Proxy Server

A server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.

4.6.3 Purpose of Proxy Server

By definition, "proxy" refers to a person or agency that has authority to act for another. It's a computer program that acts as an intermediary between web browser and a web server. To give users rapid access to popular web destinations, Internet Service Providers use Proxy servers as "holding bins" to store frequently requested pages, rather than going out and fetching them repeatedly from the net.

They explain that **Proxy Servers** are programs that mediate between a work station user in a networked environment and the Internet beyond. They are associated with a "Gateway", which separates the internal network from the outside; a "Firewall", which screens all incoming traffic and protects the network from un welcome intruders; and a "caching" program, which looks locally for previously downloaded Web pages.

If the requested page is not found, The Proxy Server goes out to the net and retrieves it, almost invisibly, for the user. Apparently, this is a safe, efficient way of handling inbound and outbound traffic on a network.

Proxy Servers have two main purposes:

- Improve Performance
- Filter Requests.

4.6.3.1 Improve Performance

Proxy servers can dynamically improve performance for groups of users. This is because it saves the results of all requests for a certain amount of time.

Consider the case where both user X and user Y access the World Wide Web through a Proxy server. First user X requests a certain web page, which we will call Page1. onetime later, user Y requests the same page. Instead of forwarding the request to the Web server where Page1 resides, which can be a time consuming operation, the Proxy server simply returns the Page1 that it already fetched for user X. Since the Proxy server is often on the same network as the user Y, this is as much as faster operation.

Real Proxy server support hundreds or thousands of users. The major online services such as **CompuServe** and **America Online**, for example, employ an array of proxy servers.

4.6.3.2 Filter Requests

Proxy servers can also be used to filter requests. For example, a company might use a proxy sever to prevent its employees from accessing a specific set of websites.

4.7 Markov Predictors

If $S=<p_1,p_2,\ldots, p_n>$ is a sequence of accesses (called a transaction) made by a user, then the conditional probability that the next access will be pn+1 is $P(p_{n+1} / p_1,p_2,\ldots,p_n)$. Therefore, given asset of user transactions, rules of the form

$$p_1, p_2 \ldots p_n \Rightarrow p_{n+1} \quad \ldots\ldots\ldots\ldots (1)$$

Can be derived where $p_{n+1} / p_1,p_2,\ldots, p_n$ is equal to or larger than a user-defined cut-off value T_c. The left part of the rule is called head and the right part is called body. The body of the rule can also be an length larger than one. Thus the rules of the form:

$$p_1, p_2 \ldots p_n \Rightarrow p_{n+1},\ldots\ldots p_{n+m} \ldots\ldots\ldots\ldots (2)$$

can be formed. In this case, $P(p_{n+1},\ldots\ldots p_{n+m} / p_1,p_2,\ldots, p_n)$ has to be larger than Tc.

The dependency of the forth coming accesses on past accesses defines a **Markov chain.** The number of past access considered in each rule for the calculation of the corresponding conditional probability is called the order of the rule. For instance the order of the rule A,B=>C is 2.

Definition 1: An n-order Markov predictor is defined to be a scheme for the calculation of conditional probabilities $P(p_{n+1},\ldots\ldots p_{n+m} / p_1,p_2,\ldots, p_n)$ between document accesses and the determination of rules of the form (2). The head of each rule has a size equal to n and the body has a maximum size equal to m.

A predictive prefetching algorithm can be defined as a collection of 1,2,…n-order Markov predictors (if several orders are considered). Additionally, an activation mechanism for finding the prefetched pages from the corresponding rules is required. Thus, the objectives of a predictive Web prefetching algorithm can be summarized as:

- The calculation of conditional probabilities based on user accesses,
- The determination of rules of the form (2), and
- The activation of rules for which their head contains the accesses a user has done up to a time point and the prefetching of pages which are in the body of the rules.

We present below how existing algorithms are described inthis common context.

The Dependency Graph (DG) algorithm uses a first order (1-order) Markov predictor. It calculates conditional probabilities $P(p_i/p_j)$. It maintains a set of rules of the form $p_{i=>}p_j$. For a user who has accessed the sequence of documents $S< p_1, p_2 \ldots p_n >$, DG searches all rules with head pn and prefetches all documents $p_{n+1}$for which there exists a rule $p_n => p_{n+1}$.

The k-order PPM algorithm uses 1,2,……,k-order Markov predictors (k is a constant). These Markov predictors calculate conditional probabilities of the form:
$P(p_{n+1}/ p_n), P(p_{n+1}/ p_n, p_{n-1}),\ldots\ldots\ldots, P(p_{n+1}/ p_n,\ldots, p_{n-k+1})$

And determine the corresponding rules, which have headsizes equal to 1,2…….. k. Since rules of several orders are activated from one user sequence, the same documents can appear in the body of different rules, hence duplicate elimination is performed.

The prefetching scheme uses a 1-order Markov predictor. The rules are of the form $p_i=>p_j$. This is analogous to DG and 1-order PPM.

The scheme uses two constraints:

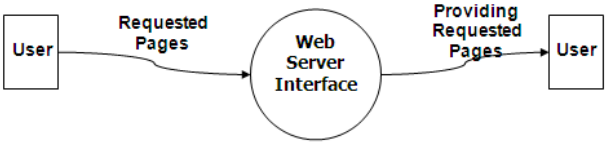
1) support-based pruning and
2) only rules with the maximum conditional probability (confidence)

are selected for each document which serves as head of a rule. These constraints do not affect the consideration as a Markov predictor.

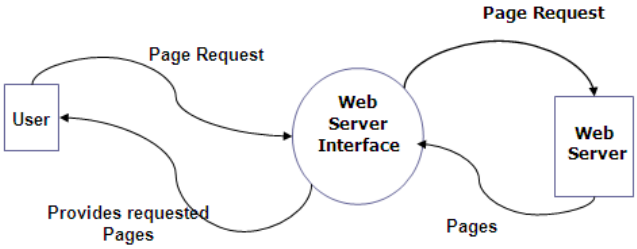
Finally, path traversals, from sequences of the form <p1,p2,……..pn>, derive rules of the form:

$p_1, p_2 \ldots p_k => p_{k+1} \ldots p_n$, by considering conditional probabilities $P(p_1, p_2 \ldots p_k / p_1, p_2 \ldots p_k)$.

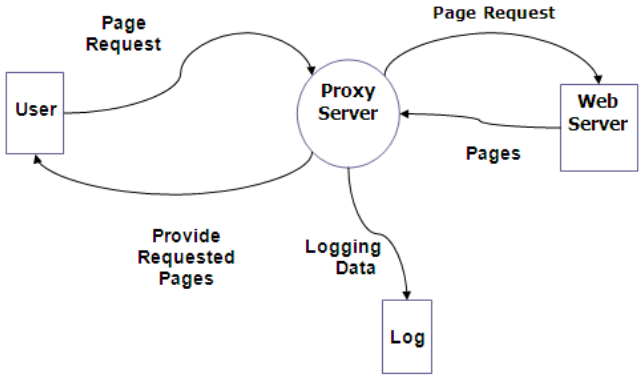
It is easy to see that path-traversals can use a set of 1,2,….,K-order Markov predictors and are equivalent to k-order PPM.



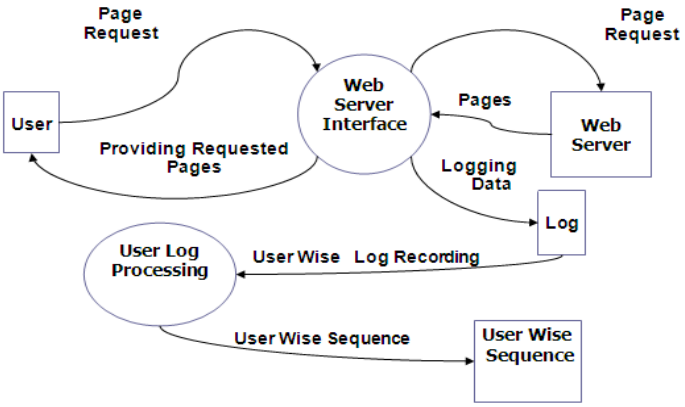
Here, User sends the request to the Web Server Interface for the web pages. The Web Server Interface provides the requested web pages to the user.



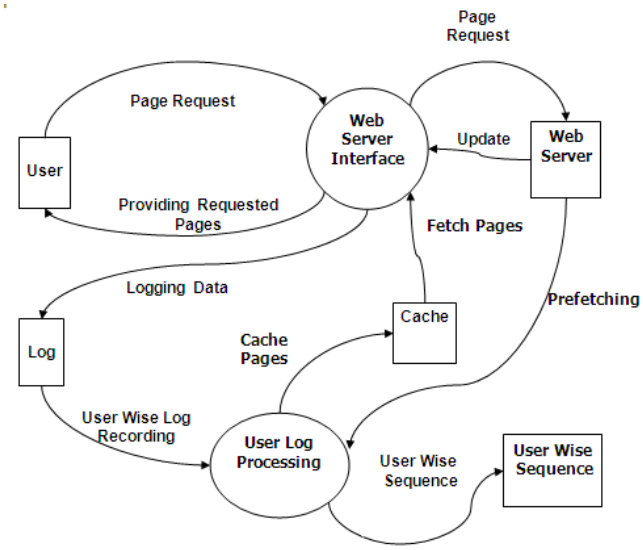
Here, User sends the request to the Web Server Interface which acts as Proxy Server, an interface between User and Server and if the requested page is present in the Proxy Server then it provides to the user. Otherwise, it fetches the pages from the Web Server and provides to the user.



Here, we maintain a Log which stores the details of each user and the pages accessed by him. If a page is requested by the user, first Proxy Server searches in the Log and if it is already stored, it provides to the user. Otherwise it fetches from the Web Server.



In this stage, From the Log, User wise Log is being processed and then User wise sequence is stored.



In the last stage, Every most frequently used Web page requested by the user is prefetched from the Web Server and Cached. From the Cache, Proxy Server provides the pages to the user.

V. CONCLUSIONS

Deduction of future references on the basis of predictive Prefetching, can be implemented by based on past references. The prediction engine can be residing either in the client/ server side. But in our context, prediction engine resides at client side. It uses the set of past references to find correlation and initiates Prefetching that is driving user's future requests for web documents based on previous requests.

Predictive Prefetching suits the web's hyper textual nature and significantly reduces the perceived latency. We present a new context for interpretation of web Prefetching algorithms as Markov Predictors. This context revealed the important factors that affect the performance of Web Prefetching algorithms. They are Order of dependencies between Web document accesses, interleaving of requests belonging to patterns with random ones and the ordering of requests. We proposed a new algorithm called WM_o, which has to be proven to be generalization of existing ones. By using this algorithm, we can efficiently address the increased number of candidates, increases the performance.

WM_o achieved large accuracy in prediction with quite low overhead in network traffic. In summary, WM_o is an effective and efficient predictive Web Prefetching algorithm.

VI. FUTURE WORK

In our paper, we ignore the nature of burstiness of web traffic by using a fixed average request arrival rate. This underestimates the hit ratio of demand cache since it ignores the temporal locality. Furthermore, burstiness of request traffic may hurt prefetching at a proxy point if there is no bandwidth available at the peak point of demand traffic. The trace-based simulations of real Proxy log trace will be used to measure the performance benefits that might gain from those approaches. The popularity of an object varies during its lifetime. Newly appeared objects exhibit typically a steeply rising start peak of user interest. And after these peaks, all objects share a general decrease of user interest in them. A popular approach for prefetching is to prefetch the contents that are related to the content that are recently accessed, e.g., pages whose hyperlink URLs are embedded in the pages that have been just referenced. In this way, the hit ratio can be increased. More study is to be done by including this approach into the algorithms presented in this paper.

VII. REFERENCES

[1] M.S. Chen, J.S. Park, and P.S. Yu, "Efficient Data Mining for Path Traversal Patterns," *IEEE Trans. Knowledge and Data Eng., vol. 10,no. 2, pp. 209-221, Apr. 1998.*

[2] M. Deshpande and G. Karypis, "Selective Markov Models For Predicting Web-Page Accesses," *Proc. SIAM Int'l Conf. Data Mining (SDM '01), Apr. 2001.*

[3] A. Nanopoulos and Y. Manolopoulos, "Finding Generalized Path Patterns for Web Log Data Mining," *Proc. East European Conf.Advances in Databases and Information Systems (ADBIS '00), pp. 215-228, Sept. 2000.*

[4] A. Nanopoulos and Y. Manolopoulos, "Mining atterns From Graph Traversals*," Data and Knowledge Eng. (DKE), vol. 37, no. 3,pp. 243-266, June 2001.*

[5] R. Sarukkai, "Link Prediction and Path Analysis Using Markov Chains," *Computer Networks, vol. 33, nos. 1-6, pp. 377-386, June 2000.*

[6] Z. Wang and J. Crowcroft, "Prefetching in World Wide Web," *Proc. IEEE Global Internet Conf., pp. 28-32, Nov. 1996*

