A STUDY ON WEB PREFETCHING TECHNIQUES

K.Ramu
Research Scholar,
Department of CSE,
Bharath University,
Chennai, India
k.ramu147@gmail.com

Dr.R.Sugumar
Asst.professor,
Department of IT,
R.M.D.Engineering College,
Chennai, India.
sugu16@gmail.com

B.Shanmugasundaram
Asst. professor,
Department of CSE,
Gojan School of Business &Tech,
Chennai, India.
en1107@gmail.com

Abstract:
The growth of the World Wide Web has emphasized the need for improvement in user latency. One of the techniques that are used for improving user latency is Web Prefetching. Web prefetching is one of the techniques proposed to reduce user’s perceived latencies in the World Wide Web. The spatial locality shown by user’s accesses makes it possible to predict future accesses based on the previous ones. A prefetching engine uses these predictions to prefetch the web objects before the user demands them. The existing prediction algorithms achieved an acceptable performance when they were proposed but the high increase in the amount of embedded objects per page has reduced their effectiveness in the current web. Web prefetching is becoming important and demanding, even though the Web caching technique has been improved. Web prefetching is an effective tool for improving the access to the World Wide Web. Prefetching can be initiated either at the client side or at the server side.

Keywords: Web prefetching, cache, latency

I. INTRODUCTION

Web prefetching is a technique that made efforts to solve the problem of these access latencies. Specially, global caching methods that straddle across users work quite well. However, the increasing trend of generating dynamic pages in response to HTTP requests from users has rendered them quite ineffective. Prefetching is used as an attempt to place data close to the processor before it is required, eliminating as many cache misses as possible. Caching offers the following benefits: Latency reduction, Less Bandwidth consumption, Lessens Web Server load. Prefetching is the means to anticipate probable future requests and to fetch the most probable documents, before they are actually requested. It is the speculative
retrieval of a resource into a cache in the anticipa-
tion that it can be served from the cache in the near
future, thereby decreases the load time of the ob-
ject.

The earlier research on web prefetching
focused on using the history of client access pat-
terns to make predictions. The access patterns were
represented using URL graph and based on graph
traversal done by search algorithm the prefetching
predictions are computed. These schemes suffer
from the drawback of not able to prefetch web
objects
that are newly created or never visited before. To
overcome these limitations, keyword based seman-
tic prefetching approach [6, 16] was introduced. It
could predict the web objects to satisfy the future
requests based on semantic preferences of past
retrieved web objects. Neural networks was trained
using the keyword set to make predictions, and the
research was motivated by the fact that the web
user’s surfing behavior was often guided by the
keywords in anchor texts of URL that refer to a
web object. The anchor link
(URL) represents relationship between two web
documents or two parts of the same web docu-
ment. Anchor text provides relevant descriptive or
contextual information to users
about the contents related to a particular anchor
link.

Web prefetching can be applied in a web
environment as between clients and web server,
between proxy servers and web server and be-
 tween clients and proxy server [8]. If it is applied
between clients and web server, it is helpful in
reducing user perceived latency, but the problem is
it will increases network traffic. If it is applied
between proxy server and web server, can reduce
 the bandwidth usage by prefetching only a specific
number of hyper links. If it is applied between
clients and proxy server, the proxy starts feeds pre-
fetched web objects from its cache to the clients so
there won’t be extra internet traffic. This paper
describes about the various prefetching techniques,
how they predict the web object to be pre-fetched

and what are the issues involved in these tech-
niques.

The rest of the paper is organized as fol-
low: Section 2 discusses classification of web
prefetching techniques.
Section 3 describes the different types of web
prefetching techniques. Section 4 deals with other
research directions and section 5 provide some
conclusion remarks.

II. Classification of Web Pre-Fetching Tech-
niques

The Web prefetching approaches can
be characterized according to its short-term and
long-term benefits. In this context, the existing
prefetching policies as short term prefetching poli-
cies and long term prefetching policies.

A. Short-term prefetching policies

In traditional short-term prefetching,
caches use recent access history to predict and
prefetch objects likely to be referenced in the near
future. In the short-term policy, objects that are
likely to be referenced in the near future are
prefetched based on the client’s recent access his-
tory. Future requests are predicted to the cache’s
recent access history. Based on these predictions,
clusters of Web objects are prefetched. In this con-
text, the short-term prefetching schemes use De-
pendency Graph (DG), where the patterns of ac-
tesses are held by a graph and Prediction by Par-
tial Match (PPM), where a scheme is used, adopt-
ed from the text compression domain. In addition,
several short-term prefetching policies are exist
namely Predictive Web prefetching, semantic web
prefetching and proxy web prefetching. These
prefetching methods are based on Markov models,
which are used for modeling and predicting user’s
browsing behavior over the Web.

B. Long-term prefetching policies

The long-term prefetching uses long-term
steady-state object access rates and update fre-
frequencies to identify objects to replicate to content
distribution locations. Compared to demand cach-
ing, long-term prefetching increases network bandwidth and disk space costs but may benefit a system by improving hit rates. In the long term policy, objects are prefetched and updated based on long-term global access and update patterns. Global object access pattern statistics (e.g., objects’ popularity, objects’ consistency) are used to identify valuable objects for prefetching. In this type of scheme, the objects with higher access frequencies and no longer update time intervals are more likely to be prefetched.

There are various types of long term prefetching exist namely frequency based prefetching, greedy-dual-size prefetching and popularity based prefetching.

III. Web Prefetching Techniques

Web pre-fetching is a mechanism by which web server or clients can pre-fetch web pages well in advance before a request is actually received by a server or send by a client. The Web pre-fetching can be classified into different types are described below.

A. Prediction by Partial Match Web Prefetching

The Prediction by Partial Match is a commonly used technique in Web prefetching, where prefetching decisions are made based on historical URLs in a dynamically maintained Markov prediction tree. Existing approaches either widely store the URL nodes by building the tree with a fixed height in each branch, or only store the branches with frequently accessed URLs. Building the popularity information into the Markov prediction tree, has proposed a new prefetching model, called popularity-based PPM. In this model, the tree is dynamically updated with a variable height in each set of branches where a popular URL can lead a set of long branches, and a less popular document leads a set of short one.

B. Predictive Web Prefetching

Predictive Web Prefetching is a mechanism of deducing the forthcoming page accesses of a client based on its past accesses. In this research work, a new context was presented for the interpretation of Web prefetching algorithms as Markov predictors. This method has identified the factors that affect the performance of Web prefetching algorithms. Predictive prefetching algorithms examined in database, file systems, and recently on the Web can be categorized into two families: 1) those that use a graph, called Dependency Graph (DG), to hold the patterns of accesses 2) those that use a scheme adopted from the text compression domain called Prediction by Partial Match (PPM). Existing Web prefetching schemes differ from the predictive web prefetching in the context of file systems only because it use techniques for the identification of user sessions. For the core issue in prefetching, i.e., prediction of requests, existing algorithms from the context of file systems have been utilized. Consequently, existing Web prefetching algorithms do not recognize the specialized characteristics of the Web Wang et.al (2005).

C. Model based Predictive Pre-fetching

Yang (2001) proposed a model based predictive pre-fetching, in which an integrated web-caching and web-pre-fetching model is used [9]. The prediction model used in this is based on the statistical correlation between web objects. The prediction model is time based, prediction window represents some specific time period than number. The algorithm constructs a logical graph called correlation graph, which shows the correlation between web objects and prefetch prefetch web objects that are highly correlated to a currently requested object. They developed an integrated caching and prefetching algorithm, Pre-GDF. This algorithm is based on the algorithms GD-Size[10] and its enhancement GDSF[11]. The key components in the algorithm are replacement manager, pre-fetching agent, prediction queue and cache.

D. Semantic Web Prefetching

The Semantic Web Prefetching is responsible for making efficient predictions of web objects to be prefetched for satisfying the user’s future requests with low latency. It is based on the concept of client-side prefetching, where the client directly prefetches web documents from the server and stores it in local cache to service user requests. It significantly reduces the latency when servicing
user requests, since there is no network latency for retrieving locally cached documents. The prefetching scheme consists of the following components: Tokenizer, Prediction unit, Prefetching unit and the Prefetching cache. When user views a web page, the anchor links in that page form a pool of URLs that the user might visit next. Semantic prefetching system is designed to efficiently identify a set of anchor links in the web page that reflects user’s interests.

When user is viewing a web page, Tokenizer parses the web page to extract anchor links (URL) and its associated anchor text. It then identifies the tokens (keywords) from each anchor text of a link. A token is considered as the meaningful word within anchor text of a URL. When a user clicks anchor link in a web page, then tokenizer moves the tokens of that particular anchor text into user token repository. The repository has collection of tokens with their frequencies, where token frequency indicates the number of times a particular token is seen in the anchor text of links selected by the user. When a token occurs for the first time, new entry is created in the repository with initial count value as 1. For the existing tokens its count value gets incremented. The user token repository is used by the prediction unit to compute probability values of anchor links that are not accessed by the user in a web page.

E. Link Pre-fetching

This mechanism, utilizes browser idle time to download documents that the user might visit in the future. A web page provides a set of pre-fetching hints to the browser and after the browser finishes loading the page, it starts prefetching specified documents and stores them in its cache. When the user visits one of the pre-fetched documents, it can be served up quickly out of the browser's cache. Fisher et. al proposed a server driven approach for ink prefetching. In this approach a browser follows special directives from the web server or proxy server that instructs it to pre-fetch specific documents. This mechanism allows servers to control the contents to be prefetched by the browser.

F. Domain Top Prefetching

A domain top approach for web prefetching combines the proxy’s active knowledge of most popular domains and documents. In this approach proxy is responsible for calculating the most popular domains and most popular documents in those domains, and then prepares a rank list for prefetching. The objective of this method is to increase the hit ratio by the proxy prefetching and to put a little burden on the proxy and the network. In our scheme, proxy finds the popular domains using access profiles and searches the popular documents in each domain.

Based on these Top-Domain and Top-Documents, proxy makes the Rank list for prefetching, and client requests a file in a certain domain and proxy forwards to them their most popular documents in the Rank list. To find the popular domains and documents does not require heavy computation power to the proxy, but only needs a very small amount of Rank List that stores them at the proxy. In addition, this approach can be implemented without changes of server and client.

G. Data Prefetching

Data prefetching is a data access latency hiding technique, which decouples and overlaps data transfers and computation. In order to reduce CPU stalling on a cache miss, data prefetching predicts future data accesses, initiates a data fetch, and brings the data closer to the computing processor before it is requested. A data prefetching strategy has to consider various issues in order to mask data access latency efficiently. It should be able to predict future accesses accurately and to move the predicted data from its source to destination in time. There are two types of data prefetching exist namely Content Prefetching and Context based Prefetching.

H. Content Prefetching

The content sensitive data prefetching is a new hardware technique for data prefetching. This technique is simple, effective, and realizable content sensitive data prefetcher can be built as an on-
chip component of the memory system, and will scale well with future processor designs. One of the major findings is that the content prefetcher enhanced memory system is capable of delivering timely prefetches to fully suppress demand load misses for nearly all applications. The large percentage of full content prefetch hits fully validates the decision to place the content prefetcher on chip and positively indicates that the proposed memory system that utilizes prioritized memory traffic and prefetch reinforcement is well balanced.

I. Context based Prefetching

The Context based Prefetching technique is used for predicting useful prefetches when a navigational object-oriented interface was implemented on a relational DBMS. When implementing persistent objects on a relational database, a major performance issue is prefetching data to minimize the number of round-trips to the database. This paper proposes the use of the context in which an object is loaded as a predictor of future accesses, where a context can be a stored collection of relationships, a query result, or a complex object.

J. Proxy Cache Prefetching

Proxy Cache Prefetching is a major Web caching technique that attempts to serve user Web requests from one or a network of proxies located between the end user and Web servers hosting the original copies of the requested objects. Proxy caching effectively reduces the network resources that Web services consume, while minimizing user access latencies. Deploying Web caching proxies over the Internet is not easy, however, and presents several cache management challenges. A proxy is usually deployed at a network’s edge, such as at an enterprise network’s gateway or firewall. The proxy processes internal client requests either locally or by forwarding the requests to a remote server, intercepting the responses, and sending the replies back to the clients. Because this proxy is shared by internal clients who tend to have similar interests, it’s natural to cache commonly requested objects on the proxy.

A client-side browser typically retrieves a Web object by initiating an HTTP GET command with the object’s address. The browser first attempts to satisfy the request from its local cache; if it fails, it sends the unresolved request to its proxy. If the proxy finds the requested object in its cache, it returns the object to the client; otherwise, the request is forwarded to the object’s origin server, which as the authoritative source of the requested object—returns the object to the proxy. The proxy then relays the object to the client and, if needed, saves a copy in its cache. If a request is satisfied from the proxy cache, it is called a cache hit; otherwise, it’s a cache miss.

K. Dynamic web prefetching

In dynamic web pre-fetching technique, each user can keep a list of sites to access immediately called user’s preference list. The preference list is stored in proxy server’s database. Intelligent agents are used for parsing the web page, monitoring the bandwidth usage and maintaining hash table, preference list and cache consistency. It controls the web traffic by reducing pre-fetching at heavy traffic and increasing pre-fetching at light traffic. Thus it reduces the idle time of the existing network and makes the traffic almost constant.

A hash table is maintained for storing the list of accessed URLs and its weight information. Depending upon the bandwidth usage and weights in the hash table, the prediction engine decides the number of URLs to be pre-fetched and gives the list to pre-fetch engine for pre-fetching the predicted web pages. After pre-fetching, the proxy server keeps the pre-fetched web pages in a separate area called pre-fetch area. The major advantage of using this technique is that, number of links to be prefetched is based on the current bandwidth usage. This technique increases overhead to proxy or client.

L. Greedy-Dual-Size Prefetching

Greedy-Dual-Size (GDS) prefetching is based on document size and has an elegant aging mechanism. Similarly, the Greedy-Dual-Frequency
(GDF) policy takes into account file frequency and exploits the aging mechanism to deal with cache pollution. The efficiency of a cache replacement policy can be evaluated along two popular metrics: file hit ratio and byte hit ratio. Using four different web server logs, it shows that GDS-like replacement policies emphasizing size yield the best file hit ratio but typically show poor byte hit ratio, while GDF-like replacement policies emphasizing frequency have better byte hit ratio but result in worse file hit ratio. The Greedy-Dual-Frequency-Size policy which allows to balance the emphasis on size and frequency.

Greedy- Dual-Size (GDS), as a solution for the web proxy replacement strategy. The GDS policy incorporates document size, cost, and an elegant aging mechanism in the decision process, and shows superior performance compared to previous caching policies. The GDS policy was extended, taking into consideration the document frequency, resulting in the Greedy-Dual-Frequency (GDF) policy, which considers a document’s frequency plus the aging mechanism, and the Greedy-Dual-Frequency-Size (GDFS), which also considers a document’s size.

Web proxy caches are used to improve performance of the WWW. Since the majority of Web documents are static documents, caching them at WWW proxies reduces both network traffic and request response time. One of the keys to better proxy cache performance is an efficient caching policy which keeps in the cache popular documents and replaces rarely used ones. Proposed caching strategy incorporates in a simple way the most important characterizations of the file and its accesses such as file size, file access frequency and recentness of the last access. GDFS is an improvement of Greedy-Dual-Size algorithm - current champion among the replacement strategies proposed for Web proxy caches. GDS does have one shortcoming: it does not take into account how many times a document has been accessed in the past.

III. Performance Metrics

There are several performance metrics exist to evaluate the performance of web prefetching techniques described below:

- **Hit Rate:** The hit rate (HR) is the ratio between the number of requests that hit in the cache and the total number of requests. Hit rate refers to the percentage of user access requests that are found in the prefetching cache. The byte-hit rate is an even more realistic measure of performance for web caching, it is the ratio between the number of bytes that hit in the proxy cache and the total number of bytes requested sent to its clients.

- **Byte Hit Rate:** Byte Hit Rate (BHR) is the percentage of the number of bytes that correspond to the requests served by the cache over the total number of bytes requested. A high HR indicates the user’s satisfaction and defines an increased user servicing. On the other hand, a high BHR improves the network performance and reduces the user-perceived latency (i.e., bandwidth savings, low congestion etc.).

- **Delay Saving Ratio:** It is a measure that takes into account the latency of fetching an object. It represents the reduced latency due to a cache hit over total latency when there is no cache present.

- **Precision:** It is the ratio of the number of correct predictions to the number of total predictions. If users in the subsequent requests access the predicted page that is in the prefetching cache, the prediction is considered to be correct, otherwise it is incorrect. The metric represents the fraction of predicted pages that are actually used.

IV. Other Research Directions

Prefetching can be applied in various domains to improve the system performance. A location-aware prefetching mechanism is introduced that is independent of any additional infrastructure and that gathers information solely over deployed,
lowbandwidth wireless links. Location-awareness becomes more and more important for delivering relevant data. The latter is a challenging task when dealing with unstable and low-bandwidth wireless networks, especially in areas that are not or only poorly covered.

Prefetching is an elegant technique to handle information provisioning for users under these circumstances. In this paper, new strategies for prefetching in a location-aware surrounding are investigated. The mechanism, as well as the underlying location model, could however also be transferred to other application areas such as railway networks and highways. The dynamic service priority allocation and the development of location-aware cache invalidation algorithms are the focus of research in this area. To overcome the problem of long retrieval latency caused by the unpredictable user behaviors during multimedia presentation, a prefetching scheme using the association rules from the data mining technique was proposed. The data mining technique can provide some priority information such as the support, confidence, and association rules which can be utilized for prefetching continuous media. The prefetching policy can predict user behaviors and evaluate segments that may be accessed in near future. The results show that the prefetching approach can get better latency reduction, even for small cache size.

V. CONCLUSIONS

This paper has given brief introduction about the different Web prefetching techniques namely Prediction by partial match, Predictive web prefetching Model based predictive pre-fetching, Semantic web prefetching, Link pre-fetching, Domain top approach, Data prefetching, Content prefetching, Context based prefetching, Proxy cache prefetching, Dynamic web pre-fetching and Greedy-Dual-Size pre-fetching are analyzed and discussed. The web prefetching scheme focus on the property spatial locality of web objects. These techniques are applied to reduce the network traffic and improve the user satisfaction. Web prefetching and caching can also be integrated to get better performance.

References


High-Performance Computing in the Asia-Pacific Region vol. 2, pp. 738-746.


