iJET EXPLORE International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 2 No. 1, January, 2012

# Design and Implementation of Web Prefetching Solution

[1]G.N.K. Suresh Babu , [2]S.K. Srivatsa
[1]Sri Chandrasekarendra Viswa Mahavidyalaya, Enathur, Kanchipuram-531 602, Tamil Nadu, India
[2]St. Joseph College of Engineering, Chennai-600 119, Tamil Nadu, India

## ABSTRACT

The enormous potential of locality based strategies like caching and prefetching to improve web performance motivates us to propose a new algorithm for performance evaluation in scenarios where different parts of the web architecture interact. Web prefetching is a technique focused on web latency reduction based on predicting the next future web object to be accessed by the user and prefetching it in idle times. So, if finally the user requests it, the object will be already at the client's cache. This technique takes advantage of the spatial locality shown by the web objects. The basics of web prefetching techniques preprocess the user requests, before they are actually demanded. Therefore, the time that the user must wait for the requested documents can be reduced by hiding the request latencies.

**Keywords:** *Web cache, Web prefetch, Latency, Access time.*

## 1. INTRODUCTION

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 .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. The prefetching technique has two main components: The prediction engine and the prefetching engine. The prediction engine runs a prediction algorithm to predict the next user's request and provide these predictions as hints to the prefetching engine. The prefetching engine handles the hints and decides to prefetch them or not depending on some conditions like available bandwidth or idle time. Wcol introduced by Chinen & Yamaguchi [2001] is a prefetching proxy server for WWW. This program prefetches referred pages (first n embedded images and m linked documents to be more precise) from a user-retrieved page. This scheme suffers from the drawback that it is a deterministic approach and necessarily prefetches a specified number of web objects linked to the requested web page. This could lead to network congestion and could result in prefetching large number of useless objects. The scheme has the advantage of being very simple to implement.
The goal of this article is to design an efficient popularity-based prefetching algorithm that could be deployed at the proxy level of network architecture. The focus of the work is to make use of the popularity of the web documents requested by the clients. To improve cache performance, researches have introduced web prefetching to work in conjunction with web caching, which means prefetching web documents from web servers, even before the user requests them. Prefetching techniques rely on predictive approaches to speculatively retrieve and store web objects into the cache for future use. Predictions on what to prefetch are made based on different criteria such as history, popularity and content.

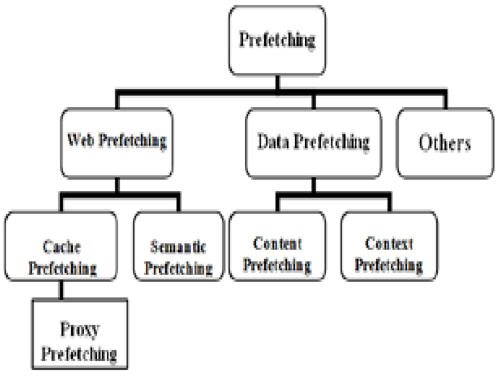
## 2. PREFETCHING

This paper is concerned with client-activated preloading of content. We are less concerned here about how the client chooses what to prefetch (e.g., Markov model of client history, bookmarks on startup, user-specified pages, the use of proxy or server hints), with the understanding that a hints-based model has the potential to avoid some of the concerns discussed here but may also introduce other problems and likewise lacks specification as a standard.
Prefetching is a method for reducing Latencies. This area of research gains importance since, an user always expects an interactive response, better satisfaction and quality of output. The prefetching can be defined as "A reference that misses in the object cache is prefetchable for a peak period if the object:

- is cacheable
- exists at least since the beginning of the previous off-peak period
- is not modified since the beginning of the previous off-peak period

From the above definition we can see that prefetching requires the use of a cache and prefetchability requires cacheability. Prefetching requires at least idempotence, since resources may end up being fetched multiple times. Speculative prefetchability requires safety. Unless we can guarantee that the client will actually use this resource, a prefetching system should not perform any action that has unwanted (from either the client or server's perspective) side-effects at the server.

Now the prefetching can be defined according to World Wide Web as “A Web resource is prefetchable if and only if it is cacheable and its retrieval is safe”. A number of commercial systems used today implement some form of prefetching. There are also a number of browser extensions for Netscape and Microsoft Internet Explorer as well as some personal proxies that perform prefetching of links of the current page. Many research papers have been published on the use of prefetching as a mechanism to improve the latencies provided by caching systems



**Prefetching Tree**

The Figure shows a Prefetching tree. There are various types of Prefetching techniques exist namely, Web Prefetching, Data Prefetching and Other issues, Web Prefetching is further classified into Cache Prefetching, Proxy Prefetching and Semantic Prefetching, Data Prefetching is classified into Content Prefetching and Context Prefetching.

## 3. ISSUES OF PREFETCHING

There are various two types of prefetching exist namely Web Prefetching and Data Prefetching. Further web prefetching can be classified into cache prefetching, proxy prefetching and semantic prefetching. Whereas, data prefetching is classified into content prefetching and context prefetching. The issues of prefetching depend on type of prefetching method used. In general there are two issues exist namely the user access latency and the hit ratio.

## 4. MINIMIZE THE USER ACCESS LATENCY

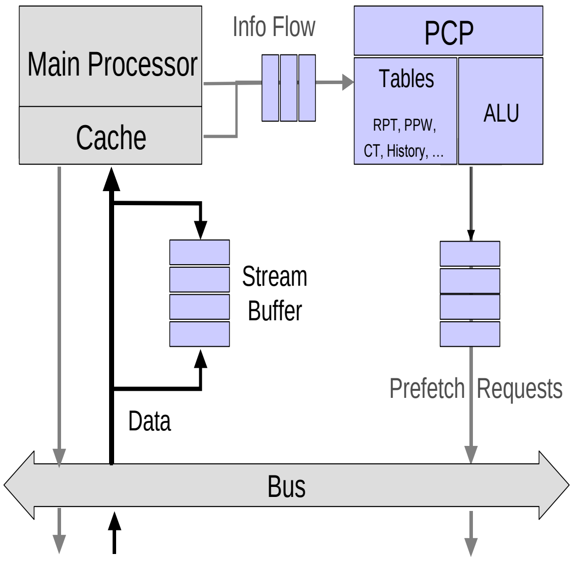
Delays in access to Web based Information continues to be a serious problem even with higher network bandwidth, due to overhead web latency has increased due to which web performance has decreased. User perceived latency from several sources such as bandwidth, speed, overhead, accessing the web page etc. In accordance of “Eight Second Rule”, it can be observed that web latency affects the user work and lot of efforts is taken to minimize the latency perceived by the user. Caching of web documents has been developed to reduce the latency but it has the drawback that it stores the pages without any prior knowledge i.e. the hit ratio is less. Web prefetching is an effective technique to minimize user’s web access latency.

## 5. MAXIMIZE THE HIT RATIO

The Hit Ratio is a ratio of the requests that are serviced from prefetched cache to the total number of requests. Higher Hit Ratio can contribute more to the improvement of the efficiency of cache. 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. Web prefetching has been recognized as an effective solution to maximize the hit ratio.

## 6. DESIGN OF POPULARITY BASED PREFETCHING ALGORITHM

The major objective of this research work is to design and implement a Popularity Based Prefetching (PBP) algorithm which combines both the top 10 and next-n prefetching approaches. In addition to using access-frequency as the criteria for prefetching, the proposed system also use the time of access of web documents to generate the top 10 list. This approach of using access-frequency and time of access is known as the Greedy Dual Size Policy approach, which has been used in cache management. Instead of generating next-n list for all the documents accessed by the users, the system log the next-n documents for the top 10 documents only, thus reducing complexity and overhead.



**Architecture of Web Prefetching**

The architecture of web prefetching is shown in figure. The proposed architecture is different from the design introduced by davison (2008), as it does not generate next-n lists for each of the requests that arrive at the proxy. It is much simpler than the existing approach and it requires much less processing time. The architecture uses the fact that popularity of web documents is one of the best measures for prefetching very extensively as only those documents are prefetched that fall into the look-ahead window of the most popular documents.

The idea of the PBP algorithm is to keep the top ten popular documents for each web server, by this means, clients or proxy servers can prefetch only these popular documents without significantly increasing network traffic. The proposed result shows that this approach expects more than 30% of client requests and achieves close to a 65% hit ratio at the cost of increasing network traffic by no more than 15% in most cases. This experiment is close to the prefetching by Popularity algorithm. The algorithm keeps copies of n most popular objects in the cache and updates them immediately whenever these objects are modified. From the Zipf-like distribution, we know that popular objects are responsible for majority of requests from users. If we keep in our cache copies of popular objects which are most likely to be requested, this will definitely achieve the highest possible hit ratio. On the other hand, its bandwidth consumption is high.

The proposed PBP algorithm uses as input such global access statistics as (1) estimates of object reference frequencies and (2) estimates of object lifetimes. This estimation can be well maintained by content distribution servers if they can collaborate between each other. Content servers collect user access statistics and publish information of objects popularity and patterns of usage; gather usage reports from their users, aggregate and analyze them and make them available to each other. They can also send prefetching hints to each other or actively push to each other objects that are likely to request in the near future. Whenever the replicated object is updated in the original server, the new version of it will be sent immediately to any cache that has subscribed.

**Table 7.1 Popularity Based Prefetching algorithm**

```
Procedure Prefetch (Array R, int M, float maxSize)
        //h, b are sequences of document ids
        begin
    1.  PrefetchSeq = Ø
    2.  for each rule h -> b such that h < R
    3.  for each dUb such that d.size <  maxSize
    4.  PrefetchSeq = prefetchSeq U d
    5.  end for
    6.  end for
    7.  Sort documents in prefetchSeq in decreasing
        order of the confidence of the corresponding
        rule and keep the first M ones.
    8.  Return prefetchSeq
        end
```

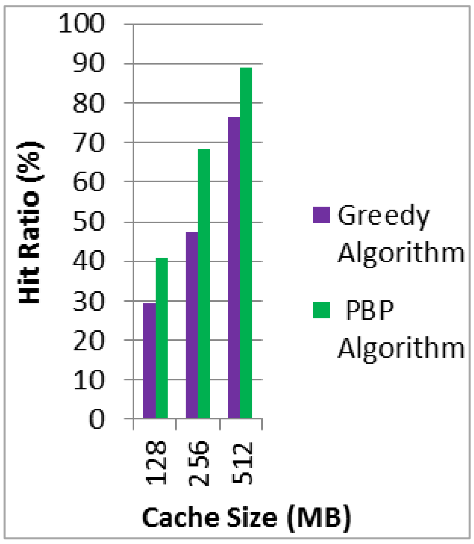
## 7. IMPROVING THE PREFETCH HIT RATIO USING PBP ALGORITHM

This article concentrates on the various performance metrics such as prefetch accuracy and prefetch hit ratio. The first metric used to describe the performance of PBP algorithm is prefetch hit ratio. The experiments are performed on different cache size of 128MB, 256MB and 512MB. Figure illustrates the graph of Prefetch Hit ratio and Cache size for greedy and PBP algorithm. It can be seen that hit ratio of the proposed PBP algorithm has improved compared to the greedy algorithm. Whenever the hit ratio of a prefetching algorithm is increased, then the specific document will be moved into top-10 cache.

**Table 7.2 Comparison of Hit Ratio and Cache size for Greedy and PBP algorithm**

| Cache Size (MB) | Hit Ratio of Greedy algorithm (%) | Hit Ratio of PBP algorithm (%) | % of Hit Ratio improved |
|---|---|---|---|
| 128 | 29.52 | 40.94 | 38.68 |
| 256 | 47.37 | 68.53 | 44.66 |
| 512 | 76.51 | 89.12 | 57.13 |

From the Table 7.2, it is clear that the hit ratio of the proposed algorithm is improved to 38.68%, 44.66%, and 57.13 for different cache size 128MB, 256MB and 512MB RAM respectively.



**Performance Comparison of Hit Ratio vs Cache size**

## 8. CONCLUSION

The proposed PBP algorithm is developed to design an efficient popularity-based prefetching algorithm that could be deployed at the proxy level of network architecture. The performance of PBP approach for 128MB, 256MB and 512 MB cache size compared with the existing greedy algorithm. The proposed algorithm used the time of access of web documents to generate the top 10 list. The results obtained from simulations, in terms of hit ratio and prefetch accuracy show the efficiency of the proposed algorithm as compared to other approaches. This research work focused on making use of the popularity of the web documents requested by the clients.

## REFERENCES

[1] Cao, P. and S. Irani: 1997, `Cost-Aware WWW Proxy Caching Algorithms'. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems. Monterey, CA.

[2] Cao, P. and C. Liu: 1998, `Maintaining Strong Cache Consistency in the World Wide Web'. IEEE Transactions on Computers 47(4), 445-457.

[3] Cate, V. 1992, Alex - A Global File System'. In: Proceedings of the USENIX File System Workshop. Ann Arbor, Michigan, pp. 1-11.

[4] Edwards, H. K., M. A. Bauer, H. Lut_yya, Y. Chan, M. Shields, and P. Woo: 2001, `A Methodology and Implementation for Analytic Modeling in Electronic Commerce Applications'. In: Proceedings of the International Symposium on Electronic Commerce. pp. 148-157.

[5] Gray, C. and D. Cheriton: 1989, `Leases: An E_cient Fault-Tolerant Mechanism for Distributed File Cache Consistency'. In: Proceedings of the 12th ACM Symposium on Operating Systems Principles. pp. 202- 210.

[6] Krishnamurthy, B. and C. E. Wills: 1997, `Study of Piggyback Cache Validation for Proxy Caches in the World Wide Web'. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems. Monterey, CA, pp. 1-12.

[7] Krishnamurthy, B. and C. E. Wills: 1998, `Piggyback Server Invalidation for Proxy Cache Coherency'. Computer Networks and ISDN Systems, 185-193.

[8] Wessels, D.: 1995, `Intelligent Caching for World-Wide Web Objects'. In: Proceedings of INET'95 Conference. Honolulu, Hawaii.

[9] Yin, J., L. Alvisi, M. Dahlin, and C. Lin: 1998, `Using Leases to Support Server- Driven Consistency in Large-Scale Systems'. In: Proceedings of the 18th IEEE International Conference on Distributed Computing Systems. pp. 285-294.

[10] Yin, J., L. Alvisi, M. Dahlin, and C. Lin: 1999, `Volume Leases for Consistency in Large-Scale Systems'. Knowledge and Data Engineering 11(4), 563-576.

[11] Yu, H., L. Breslau, and S. Shenker: 1999, `A Scalable Web Cache Consistency Architecture'. In: Proceedings of the ACM SIGCOMM'99. Boston, MA, pp. 163-174.