# Proxy-Side Web Prefetching Scheme for Efficient Bandwidth Usage: A Probabilistic Method

Swapnil S. Chaudhari
Research Scholar: Department of Computer Engineering
G.H. Raisoni College of engineering & Management
Pune, India

Prof. Poonam Gupta
Asst. Professor: Department of Computer Engineering
G.H. Raisoni College of engineering & Management
Pune, India

***Abstract*—The expansion of the World Wide Web on internet has emphasize the need for upgrading in user latency. One of the methods that are used for enhancing user latency is Web Prefetching followed by web caching. Web prefetching is one of the methods to condense user's latencies in the World Wide Web efficiently. User's accesses makes it possible to predict future accesses based on the previous objects and previous sites. A prefetching engine makes use of these predictions to prefetch the web objects and performed site before the user demands them on the behalf of user. Web prefetching is becoming important and demanding, even though the Web caching system has been recover because of bandwidth usage. Web prefetching is a helpful implement for upgrading the access to the World Wide Web and it also diminish the bandwidth usage at the time. Prefetching can be perform at the client side or at the server side and in this paper we are going through Proxy side web prefetching scheme. We Propose Proxy side web prefetching scheme with probabilistic method which improves cache hit rate with tiny amount of supplementary storage space.**

***Keywords— Improvement of web caching followed by web prefetching, Probabilistic method for web prefetching, Proxy side web prefetching, Web prefetching objects,***

## I. INTRODUCTION

As the number of internet(World Wide Web) users forms, web jamming enforces to enhance at an exponential pace of WWW or web access users.[1] At present, Web traffic is one of the chief parts of Internet traffic which we must have to cut. Web traffic bandwidth usage in student cluster for 24 hours at Organization or college, elevated bandwidth is required in peak periods that is at college timings, while leaving bandwidth at rest during off-peak periods(at night). It is enforced to equilibrium the bandwidth usage between peak periods and off-peak periods to shrink bandwidth usage in peak periods. This will reduce Web access time and make more efficient use of HTTP links and use of repeated objects. [1]

One of the solutions to play down Web traffic and speed up Web access is in the course of Web caching [2]. However, Web caching cannot tolerate network bandwidth usage during peak periods [1, 4, and 5]. In this Paper we are going to focus on the use of prefetching followed by web caching, based on a caching server, for reducing bandwidth during peak periods by using off-peak period bandwidth. We have proposed a unique, proxy-side prefetching scheme on the basis of probabilistic method that progress cache hit rate. This prefetching scheme uses Web access patterns of users. this scheme may increase total bandwidth usage in comparison with particular Web caching. However, the proposed scheme can efficiently reduce Web traffic bandwidth usage during peak periods by prefetching the objects during off-peak periods. Web prefetching is a technique that makes efforts to resolve the problem on access latencies. Specially, global caching methods that include crosswise users work depend on their access. Prefetching is used as an attempt to place data close to the processor before it is required as the concept of cache, removing as many cache misses as possible. Caching offers the following benefits: Latency decline, Less Bandwidth consumption, Lessens Web Server load and the most important is faster work. Prefetching is the means to anticipate probable future objects and to fetch the most probable objects, patterns, before they are actually requested. It is the retrieval of a resource into a cache in the anticipation that it can be served from the cache in the near future era. [1, 2, 3]

As the increased usage of internet services, the performance of internet servers slows down and load increases which affect user satisfaction. A not expensive and effective way to advance the routine is Web caching only. It is achieved by storing copies of documents, pages, objects (HTML pages and images) in a cache to reduce web site access times. A properly designed web cache can save network bandwidth, reduce server load and perceived cover on user waiting times. Assume that web documents are prepared as pages. In common, the cache is smaller and contains a lesser amount of number of pages than the origin server because it has less memory and less efficient. All pages are requested and served from the cache to memory and memory to cache. When the requested page is not present in the cache, it is requested from the origin server where it is located originally. The requested page is then transferred into the cache and served to the client as per request. When the cache is full, a page must be aloof to

make space for the page to be transferred from the origin server. [1] A page can be removed at casual or one of the page replacement policies can be used to choose which page should be removed. A normally used page-replacement algorithm is the least-recently used (LRU) approach. It remove a page from the cache based upon its "age" clear as the time onwards since it was last accessed. The "very oldest" page is "thrown out" to make space for latest pages. An LRU-based cache performs better than a cache that selects a fatality page at random. The rationale behind an LRU page-replacement policy is that a page that has not been requested for a long time is less likely to be requested another time. This idea has gained much interest from the researchers and has been termed Prefetching. Web prefetching is a helpful tool for upgrading the admission to the World Wide Web and it also decrease the bandwidth usage at the time. Prefetching can be done at the client side or at the server side and in this paper we are going through Proxy side web prefetching scheme. We Propose Proxy side web prefetching scheme with probabilistic method which improves cache hit rate with tiny amount of supplementary storage space. [1, 4, 10]

One of the goals in advance cache routine is to achieve superior hit rates. When a requested object found in the cache, it is called a cache hit. Hit rate is the ratio of "hits" over total demand. Higher hit rate implies less contact to the web server. Therefore, both network traffic and client waiting time are condensed. Although hit rate is important for cache performance, other performance metrics have to be taken into account in the evaluation. [4, 10]

### *A. Introduction to Proxy Servers*

In computer networks, a proxy server is a server (a computer system or an application) that take action as an mediator for requests from clients asking for resources from added servers. A client fix to the proxy server, requesting some service, such as connection, web page, file or other resource existing from a unlike server. The proxy server estimate the request as a way to make simpler and control their density. Today, most proxies are web proxies, help access to content on the World Wide Web. [5, 6]

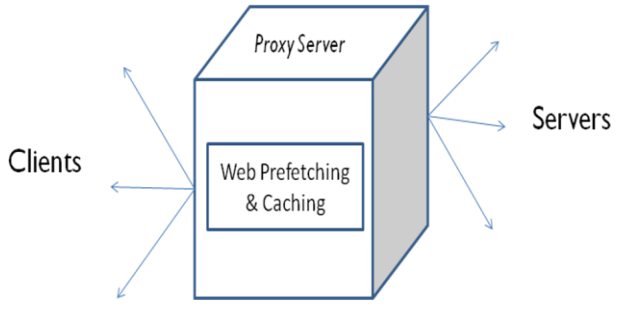


Fig.1. Concept of Proxy server

### *B. Classification of Web Prefetching Techniques.*

#### *1. Short term Prefetching*

In usual short-term prefetching, store use recent access past to predict and prefetched objects and subjects likely to be referenced in the next to future. In the short-term policy, objects that are possible to be referenced in the next to future are prefetched depends on the client's up to date access history. Future requests are predicted to the cache's new access history. Based on these predictions, clusters of Web objects and subjects are prefetched. In this framework, the short-term prefetching schemes use Dependency Graph (DG), where the patterns of contact are held by a graph and Prediction by Partial Match (PPM), [4,8] where a method is used, approve from the text density domain. In addition, several short-term prefetching policies are stay alive namely semantic web prefetching, Predictive Web prefetching and proxy surface web prefetching. [2]

#### *2. Long-term prefetching policies*

The long-term prefetching uses long-term steady-state object access rates and update frequencies to identify objects to duplicate to content sharing locations. Compared to demand caching, long-term prefetching boost network bandwidth and disk room costs but may benefit a system by recover hit rates. In the long term policy, objects are prefetched and reorganized based on long-term global access and modernize patterns. Global object access pattern data (e.g., objects' status, objects' reliability) are used to identify costly objects for prefetching. In this type of method, the objects with advanced access frequencies and no longer modernize time period are more likely to be prefetched. There are a mixture of types of long term prefetching exist namely frequency based prefetching, hungry dual size prefetching and popularity based prefetching. Prefetching is a method to anticipate future Web object requests and prefetch objects in a local cache. A prefetching approach can be classify by following three aspects. [2]

### *C. prefetching approaches*

#### *1. Server-Side, Client-Side and Proxy-Side Prefetching*

Prefetching can be initiated by Web servers, by clients or by cache servers (also called proxies). When a client demands a Web object, a Web server may anticipate the next request and immediately preload the corresponding Web objects to the client [2]. A client can also initiate the prefetching of Web objects by changing the user's configuration. A client can also use a Web access pattern monitoring system, which observes the past access patterns for particular Web objects and prefetches. Web objects on the behalf on the user [16]. Also, a proxy can initiate the prefetching of Web objects. A proxy can analyze the Web access patterns of clients, anticipate future requests and prefetch them [11, 12, and 15].

#### *2. Statistical and Deterministic Prefetching*

The decision whether a Web object should be prefetched can be either statistical [5] or deterministic [1]. In statistical prefetching, the prefetching system uses the Web access patterns of users. In a deterministic scheme, prefetching can be configured statically by clients or by proxies. For example, a user may organize Web objects such as the home page of a popular broadsheet with the deterministic prefetching manner. These Web objects are always prefetched all time the prefetching is enabled.

### 3. *Prediction and Batch Prefetching*

The criteria for fixing when a Web object should be prefetched by prediction or by batch. In the prediction proposal, when a client requisite a Web object, a Web server, proxy server or client may predict the subsequently request and immediately preload the predicted Web subjects to the client [2, 5]. The purpose of most prediction prefetching is to diminish delay in Web access moment. However, if the prediction is not correct, network possessions are wasted and network bottlenecks are created during peak periods. In the batch scheme, Web objects that the user is accesses in the future are prefetched during off-peak periods [1, 9]. Prefetching can be applied in a mixture of domains to recover the system performance. A location awake prefetching mechanism is introduced that is autonomous of any additional infrastructure and that assemble information exclusively over deployed, low bandwidth wireless links. Location-awareness becomes more and more important for delivering significant data. The concluding is a challenging duty when facing with unstable and low-bandwidth wireless networks, especially in domain that are not or only feebly covered. Prefetching is an well-designed technique to handle information provisioning for users under these conditions. In this paper, new tactic for prefetching in a location-aware surrounding are explored. The method, as well as the fundamental location model, could however also be relocate to other application areas such as airline networks and rail networks. The vibrant service precedence allocation and the expansion of location aware cache termination algorithms are the focus of research in this domain. To defeat the problem of long retrieval latency caused by the volatile user behaviors during multimedia arrangement, a prefetching scheme using the association rules from the data mining procedure was proposed.

## II. METHODOLOGY

### A. *Problem Definition :*

We describe several prefetching routines in the previous section. In the server side prefetching scheme, the client has to aware of prefetching and both the client and web server system has to be maintained. Client side prefetching cannot contribute their web access blueprint of all users in the intranet, because this plan uses only one users web access blueprint. Deterministic Prefetching has partial expansion to all web objects in the prediction prefetching rule, Network traffic may be generated during peak periods so this method is not proper for reducing peak bandwidth usage. Therefore we decided to develop a statistical web Proxy-Side Prefetching scheme for resourceful usage of bandwidth.

The problem Definition is to design and implement probabilistic cache prediction procedure to improve cache hit ratio in web proxy servers.

To design Prefetch List Generator, Cache Request Generator and Refreshing Algorithm of Proxy server for freshness of objects.

### B. *System Architecture*

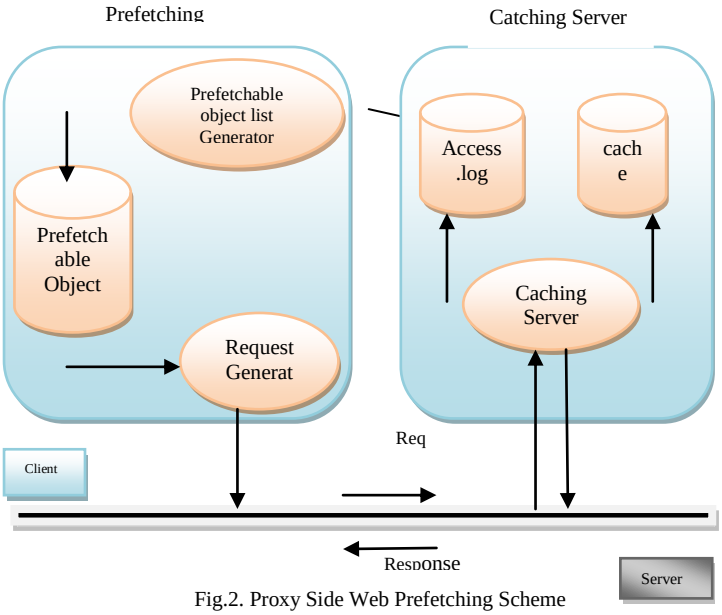


Fig.2. Proxy Side Web Prefetching Scheme

### 1. *Prefetchable Object List Generator*

The prefetch list generator is used to make your mind up whether a Web object should be prefetched. It bring into play the access log that reports the entrance information of HTTP requests, url requests from clients, and the store log that reports the information of Web objects in the cache. The access log stay records of Web cache requests. For each requested Web object, time of request, size of the object, and URL of the object are pull out to understand access model of Web objects in the cache. The store log stay records of category, class of cached Web objects. OBJ_DATE is a value of the HTTP or url date [1] described by the date and time at which the message originated. OBJ_LASTMOD is a value of the HTTP, url(TCP,UDP) Last- Modified, indicating the date and time when the sender believes the object was last modified. By monitoring the above points in the log files, the prefetch list generator forms a list of Prefetchable objects with calculated reference counts. [1]

### 2. *Cache Refreshing Algorithm of Proxy Server*

Web objects need to be detached from the cache when they die. When a cached Web object is requested, proxy server in our case windows proxy server or squid proxy server checks the freshness of the object, and then records the incident in access log and store log format. If an object is 'FRESH,' it is obtainable to clients. If an object is 'STALE,' it wants to be fetched from the original or master server. The refresh algorithm is checked in the order listed and the first matching entrance is used. If none of the entries matches, then the default will be used. [6, 8, 17]

The CLIENTS_MAX_AGE is the maximum object age the client will accept from the HTTP/URL Cache Control request header. The OBJ_AGE described as how much the object and subject has aged since it was retrieved. The LM_AGE is how old the object was when it was retrieved. The LM_FACTOR is the fraction of OBJ_AGE to LM_AGE. The NOW is current time. AGE is how much the object has aged since it was retrieved: OBJ_AGE =

NOW - OBJ_DATE. LM_AGE is how old the object was when it was retrieved: LM_AGE = OBJ_DATE - OBJ_LASTMOD. MS_FACTOR is the ratio of AGE to LM_AGE: LM_FACTOR = OBJ_AGE / LM_AGE. MAX_AGE, MIN_AGE and CONF_PERCENT are compared with the parameters of the refresh pattern rules. These values are used for the evasion value of Expires. [1]

*3. Access Log Format & Store Log Format of Proxy Server[14]*

| Time | Elapsed | Remote | Code/status | Bytes | Method | URL |
|---|---|---|---|---|---|---|

Fig.3. Access Log Format of Proxy server

| Time | Action | Status | Date | Mod | Exp | Type | Len | Method | URL |
|---|---|---|---|---|---|---|---|---|---|

Fig.4. Store Log Format of Proxy server

*4. Request Generator*

The request generator utility runs as a Web client. The request generator will generate HTTP/URL or web objects requests from the prefetch inventory and send the HTTP requests to the Web server during off-peak periods. We have developed the request generator using "wget," which is a command-line command in Unix for gathering web from server without use of browser Web client that supports HTTP [1]. Implementation of the request generator is undemanding and the most vital thing is that this request is generated by itself on behalf of client itself. The prefetched objects are amassed in client's cache [1].

*5. Algorithm of Coordinated Prefetching On The Basis Of Probability*

In our algorithm, we still include the input from the clients even when a hit come to pass in the browser cache. Upon a client request of an object file, if the request hits in the browser, the object will be accessed locally from cache itself. The client will also inform the proxy about the access to the object file, and begin the coordinated proxy–server prefetching process [3]. If the request is not in the browser, the request will be forwarded to the proxy, and the coordinated proxy–server prefetching process is open. Starting in the proxy, the coordinated proxy–server prefetching process will handle the following four parameter class in the proxy:

- The pattern object exists and a prediction can be made in the proxy server;
- The pattern object does not exist and a prediction can be made in the proxy;
- The pattern object exists and a prediction cannot be made in the proxy;
- The pattern object does not exist and a prediction cannot be made in the proxy;

And Proxy based web Prefetching Includes:-

- The proxy launches the requested object to the client, beside with a list of URLs of predicted objects itself;
- After a local searching, the client launches a selected list of URLs of predicted objects to the proxy;
- The proxy launches the selected objects to the original server.

## III. MATHEMATICAL MODEL

*PAi*(*t*):-is defined as the probability *p* that the previous access to object *i* was *t* time units before the current time, and

*PBi*(*t*) :-is the probability *p* that no updates were done to object *i* since its last access *t* time units in the past.[13]

Then, the probability of a hit on a request to a demand cache is

$$Phitd = \sum_{i}^{n} PAi(t)PBi(t)dt \quad (1)$$

Here we provide a model *H/B* for the measurement of their balance, where the hit ratio and network resource consumption of demand cache serve as baseline for evaluation. This model defines the ratio of hit ratio upgrading over increased bandwidth cost.

$$\frac{H}{B} = \frac{Hitprefetching/Hitdemand}{BWprefetching/BWdemand} \quad (2)$$

***Hitprefetching/Hitdemand*** is the hit ratio step up of prefetching over *demand caching*.

***BWprefetching=BWdemand*** is network bandwidth boost over *demand caching*. [13]

The *H/B* ratio is a quantity that demonstrates how much possible advantage the prefetching algorithms can convey compared to *demand caching* with respect of their capability to balance hit ratio improvement next to increased bandwidth cost. It is always enviable that prefetching can attain the same amount of latency reduction at not as much of expense as possible, which acquiesce a higher *H/B* value. *H/B* model works as an appraisal metric for a given prefetching algorithm to find out, at which prefetching step, e.g., how many objects to duplicate in move on; it can obtain the most favorable value if *H/B* is fretful. Or it can also be considered as a metric for the assessment between prefetching algorithms. In later sectors, we will scrutinize what *H/B* values can be accomplish by the four prefetching algorithms, which prefetching approach can obtain the best balance of increased resource expenditure and superior response time when *H/B* is concerned and what it advocate.

$$\frac{H}{B} = \frac{\left(\frac{Hitprefetching}{Hitdemand}\right)k}{BWprefetching/BWdemand} \quad (3)$$

Compared with *H/B* ratio, *Hk/B* never does healthier in comparison between prefetching algorithms. However, *Hk/B* ratio is a enhanced metric for a given prefetching system to settle on at which prefetching point it achieves the unsurpassed benefit/cost balance. By increasing the power of hit ratio improvement (*k*), we increase the consequence of hit ratio improvement on evaluating the

benefit/cost balance. When measuring with *Hk/B*, we judge that even relatively small hit ratio perfection at cost of a relatively large amount of bandwidth can be defensible if there is copiousness of spare network resource. [13]

### A. *Performance Measures*

There are numerous metrics used to investigate the efficiency and the performance of Web pre-fetching techniques. The subsequent measures are frequently used for evaluating performance of web prefetching [13]

1. ***Precision*** *(Pc):* The proportion or ratio of prefetch hits to the full amount number of objects prefetched as shown in Eq.

$$Pc = \frac{\text{PrefetchHits}}{\text{Prefetches}} \quad (4)$$

2. ***Byte Precision*** *(BPc):* Byte precision determine by taking the fraction of prefetched bytes that are subsequently requested. It can be calculated by reinstate the number of prefetched objects with their size in bytes.

3. ***Byte Recall*** *(BRc):* Byte recall measures the percentage of demanded bytes that were beforehand prefetched and afterward requested.

4. ***Latency per page ratio****:* The latency per page ratio is the ratio of the latency that prefetching attain to the latency with no prefetching. The latency per page is calculated by measure up to the time between the browser initiation of an HTML page GET and the browser reception of the last byte of the most recent embedded image or object for that page. This metric represents the advantage perceived by the user, which will be finer as minor its value is.

## IV. EXPECTED RESULTS

### A. *Performance of Prefetching Strategies*

- The cache hit rate should have to increase by 65% on the average compared with non prefetching scheme. when the reference count is increased the request saving is decreased
- The cache hit rate should have to increase by 60% on the average compared with non prefetching scheme. When the reference count is increased the Bandwidth saving is decreased and wastage Bandwidth decreases.

## V. SUMMARY AND FUTURE WORK

Enhancement of cache performance in web proxy servers is tremendously important. This can be realizing through mixture of ways. In our critique we aim to achieve this performance expansion using Proxy side web prefetching scheme for the efficient use of bandwidth using web object prefetching probabilistic method. For that we use Statistical proxy side web prefetching proposal. Proxy side prefetching can show the way to significant reduction of peak bandwidth usage using extra network assets to prefetch web object during off-peak period

The work presented in this paper explores the use of the web in studious environment. We believe that we could achieve similar or better results in a business environment where the off-peak period is typically larger.

## REFERENCES

[1] Jae-young Kim "*A Sttistical Prefetching Web Caching Scheme for Efficient Internet Bandwidth* Usage*"* Distributed processing and network Managemrnt Lab Dept of computer Engg.San 31,Hyoja,Nam-gu Pohang,Korea 790-784,Jan,2005

[2] Bin Wu; Ajay D. Kshemakalyani. "Objective optimal algorithm for long term web prefetching", *Journal of IEEE Transactions on Computers*, Vol. 55, Is- sue 1, 2006.

[3] H.T. Chen, Pre-fetching and Re-fetching in Web caching systems: Algorithms and Simulation, Master Thesis, TRENT UNIVESITY, Peterborough, Ontario, Canada (2008).

[4] J. Cobb, and H. ElAarag, "Web proxy cache replacement scheme based on back-propagation neural network", *Journal of System and Software*, 81(9), (2008), pp. 1539-1558.

[5] P. Venketesh, R. Venkatesan, "A Survey on Applications of Neural Networks and Evolutionary Techniques in Web Caching", *IETE Tech Rev*, 26, (2009), pp.171-80 .

[6] T. Palpanas and A. Mendelzon, "Web prefetching using partial match prediction*", In Proceedings of the 4th International Web Caching Workshop*. San Diego, USA, (1999).

[7] D. Duchamp. Prefetching hyperlinks. In Proceedings of 2nd USENIX Symposium on Internet Technologies and Systems, pages 127–138,1999

[8] Farhan, Intelligent Web Caching Architecture. Master thesis. Faculty of Computer Science and Information System, UTM University, Johor, Malaysia, (2007).

[9] W. Tian, B. Choi, and V.V. Phoha,"An Adaptive Web Cache Access Predictor Using Neural Network". Proceedings of the 15th international conference on Industrial and engineering applications of artificial intelligence and expert systems: developments in applied artificial intelligence, Lecture Notes In Computer Science(LNCS), Springer-Verlag London, UK 2358, (2002).450-459

[10] K. Tirdad, F. Pakzad, and A. Abhari, "Cache replacement solutions by evolutionary computing technique", Proceedings of the 2009 Spring Simulation Multiconference, San Diego, California, Society for Computer Simulation International,(2009), pp. 1-4.

[11] B. Zhijie, G. Zhimin, and J. Yu, "A Survey of Web Prefetching*", Journal of computer research and development*, 46(2), (2009), pp. 202-210.

[12] J. Domenech, J. A. Gil, J. Sahuquillo, and A. Pont, "Using current web page structure to improve prefetching performance", *Computer Network Journal*, 54(9), (2010), 1404-1417.

[13] Y. Jiang, M.Y Wu, and W. Shu, "Web prefetching: Costs, benefits and performance", *Proceedings of the 11th International World Wide Web Conference*, New York, ACM, (2002).

[14] Q.Yang, H. Zhang, and T. Li, "Mining web logs for prediction models in WWW caching and prefetching", *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining*, (2001), pp. 473–478.

[15] J. Domenech, A. Pont-Sanju´an, J. Sahuquillo, and J. A. Gil, "Evaluation, Analysis and Adaptation of Web Prefetching Techniques in Current Web", *Web-based Support Systems, Springer, London*, (2010). 239-271.

[16] W. Teng, C. Chang, and M. Chen, "Integrating Web Caching and Web Prefetching in Client-Side Proxies", *IEEE Transaction on Parallel and Distributed Systems*, 16(5), (2005), pp 444-455.

[17] B. Jin, T. Sihua, C. Lin, X. Ren, and Yu. Huang, "An Integrated Prefetching and Caching Scheme for Mobile Web Caching System". *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, (2007).

[18] T.I.Ibrahim, C.Xu, "Neural net based predic- tive pre-fetching to tolerate WWW latency*", Pro- ceedings of the 20th International Conference on Distributed Computing Systems*, 2000.