Abstract

This paper summarizes the various research efforts on prefetching schemes in WWW. We try to classify and rate the previous methods and results on this topic and point out the potential works. We will proceed from several perspectives: questions addressed in the past, methods that have been used, advantages and limitations, comparison of different results, open questions, new changes in Internet environments, modifications needed in protocol or algorithm for successful prefetching, prefetching effects and future directions.

keywords: prefetching, WWW, cache, latency.

1 Introduction

Web traffic is the predominant traffic in the current Internet, and this will be still true in the near future. Since Web provides an interactive interface, user is sensitive to the latency used in loading Web pages. To reduce latency, and to avoid World Wide Wait, many solutions have been proposed in the past few years. One of them is prefetching, which is based on web page access patterns to predict which page is to be accessed next or very soon, these pages will be prefetched into user’s cache. In this paper, we will discuss various aspects of prefetching. In section 2 we will give a big picture of latency reduction techniques in WWW; In section 3 we introduce previous prefetching methods and their results; In section 4, we present some effects of prefetching; In section 5, we analyze some theoretical results and system model of global prefetching (including adaptive prefetching as we proposed); In section 6, we give out some other voices about prefetching, present the reality of successful implementation of prefetch; We conclude in Section 7 and point out future research directions in Section 8.

After exhaustive literal research on this topic, we found that many things have already been done, and many ways from different perspectives have been tried for this topic. Our previous proposal had been well studied in [1]. We strongly feel that it’s worth to write a survey to summarize the past work on prefetching, this is also a time-saving way for the future works.

2 Overview of the latency reduction techniques in WWW

According to the introduction part of [4], ”the rapid growth of the WWW inspired numerous techniques to reduce web latency”. In this section, we first describe the possible WWW environments and then proceed to describe the different techniques in latency reduction.
2.1 Caching

Caching in the WWW can be divided into three categories[4] based on its location: proxy, clients, and server side caching.

In proxy caching, a computer (proxy server) serves as a cache of objects for a set of WWW clients. Clearly, it will be faster than the content provider (WWW server). Early studies showed that a significant latency reduction if proxy caching is employed. [12] showed that proxy cache hit ratios can approach 60% for a pool of 8000 clients. It was shown that the hit ratio is a function of the number of clients the proxy server services.

In client side caching, the objects (i.e. HTML, GIF and JPEG files) are stored for a limited period of time on the local hard drive of the client. Only request made by the client, are cached. [4] speculated that a large portion of the benefit from client side caching is limited to the use of the BACK button.

Server side caching caches the content of web pages, which are generated dynamically. This method reduces the large object creation latency in constructing web pages. Since temporal locality exists in access pattern to the web server (i.e. if a page is accessed by one client, it is likely to be accessed by other clients in the near future), thus caching those pages on the server is beneficial. With server-side caching very few clients will incur the latency associated with generating the dynamic web pages. This technique was successfully used by IBM in the 1996 Atlanta Olympic games’ official web site [13].

For caching, each cached file has a TTL, the refreshment policy of TTL is also an interesting topic, which is discussed by [5]. This method is useful only when the cached file is valid and the TTL expires. But this can greatly reduce overhead of a new request to the server. Actually, the TTL refreshment policy is more important for the proxy server, because they may have more unexpired TTL than the cache of a single client.

2.2 Prefetching

Prefetching on proxy server has been well explored. The knowledge whether to fetch a page or not can come from the prefetching side or the server side. Since the server has more information regarding request patterns, it is beneficial to use server hints to determine what to prefetch. [6] gives a very good summary of various aspects and methods in prefetching. [6] use Markov model as prediction algorithm and get a relatively high accuracy of about 50% which means half of the prefetched files are eventually accessed. Prefetching from the proxy side is a well-studies concept. While a reduction latency of up to 50% has been observed, it only came at the expense of significantly increasing the amount of traffic on the network. Our proposal is to explore the effect of a clogged network on latency. While we are not aware of any literature on server side prefetching, it should be possible to reduce latency by prefetching a dynamic web page on the server. This scheme is better combined with server caching with dynamic content together. [4] also studied the client prefetching method, which is requested by each client manually.

2.3 Preopening

In some papers we reviewed, some also discussed how to speculatively preopen network connections. Instead of prefetching all files, client only makes the DNS lookup and sets up the three-way handshake with server. This can reduce latency to some extent and have a small risk. This is discussed in [7].
3 Prefetching schemes and results

Prefetching can be divided into 3 categories: client, proxy and server side prefetching. The client side prefetching consists of prediction algorithm and decision-making. There are two basic methods for prediction algorithm. One is structure-based prediction, which constructs a graph of the web sites’ content. Each page or file is represented as a node in the graph, and the weight of links between nodes describes the access pattern of each file based on the analysis of hyperlink relations. Suppose A, B are two files in the web site. Once A is accessed, it makes sense to prefetch B if the link $A \rightarrow B$ has a higher weight. The weight of links may be dynamic, and it could change depending on clients. The other commonly employed method is the Markov algorithm, which makes no use of structure information. It seeks patterns within a ”window” of prior references, where the window is typically measured by time or number of references[6]. Although the Markov algorithm may discover non-intuitive patterns, it is not good for long tail distribution, and the computational cost is also relatively high. In the decision-making, the client part may be more active or reactive. In the scheme described by Dan Duchamp, at the beginning of a connection, client negotiates with server to determine the parameters of prefetching such as whether it will happen and how much information they will exchange to support it[6]. Server is limited to a data provider, which collects acknowledgements from clients and provides the statistic information. On the other hand, V. Padmanabhan and J. Mogul [14] give another scheme, where server plays a more active role by suggesting clients to prefetch certain links. Client may make the final decision of prefetching based on whether the file is already in the local cache or other criterias. One important difference between the two models is that Duchamp’s model considers dynamic objects such as pages generated dynamically, by query URLs or those having cookies are prefetchable. While in Padmanabhan and Mogul’s model dynamic objects are not included into the simulation. Both schemes claim significant improvements. Duchamp’s scheme has a lower client latency (by 52.3%) and less wasted network bandwidth (24.0%). The other scheme has doubled the hit rate. It is hard to compare the two results directly, because different assumptions and measurements are used. Duchamp’s scheme may have a better hit rate by using local access pattern.

The prefetching from proxy side is a well-studied concept. [4] However the latency reduction causes significant increase in the amount of traffic on the network. The server side prefetching may reduce the construction latency of dynamic objects, but this objective can be achieved using simple server side caching technique. Therefore this kind of prefetching has no better result than the caching scheme on the server with the only exception when a rapid change of data, on which page construction depends, is present. [4]

4 Network effects of prefetching

Although prefetching may reduce user’s perceivable latency, it also increases network traffic[4] [6] [14]. Even when prefetching adds no extra traffic to network, straightforward approaches to prefetching increase the burstiness of individual sources, leading to increased average queue sizes in network switches[10]. The reason behind this is that Web traffic is shown to be bursty at a wide range of scales (“self-similar”). The aggregation of many ON/OFF behavioral individual source lead to self-similar traffic. Crovella and Barford’s simulations show that besides increasing bursti-
ness, prefetching keeps routers buffers full at a higher average level. Thus when a burst of packets arrives, instead of being stored, packets must be dropped. Crovella and Barford proposed a transport rate control mechanism to solve the problem.

Some other negative network effects due to prefetching include unknown cacheability, server overhead, side effects of retrieval and user activity conflation[8]. When the prefetching is set up, before loading objects, one cannot know whether the object is cacheable or uncacheable. In the point of view of users, prefetching may reduce access latency, however, it may not be so interesting for content provider, because it consume some portion of the server’s resources including bandwidth and cpu. Moreover, some retrievals may bring undesirable effects to content provider and content recipient (i.e. placing items corresponding to those links into a shopping cart without the knowledge and consent of the user). Finally, prefetching scheme must distinguish true requests from prefetching requests, otherwise the statistics may lead to incorrect prefetching.

5 Prefetching model analysis

In this field, there are not too much theoretical analysis, we only see [1],[2],[3]. In [3], they mainly discussed client initiated prefetching and implemented it through caching proxies. They used M/M/1 model which is very simple and got a relation between bandwidth and delay tradeoff.

\[ E = P/((R_x - R_0)/R_0) > R_0/(1 - R_0) \]  (1)

Formula (1) determines the relationship between the efficiency of prefetching \( E \) and the link utilization with and without prefetching \( R_x \) and \( R_0 \) respectively. In this analysis, they didn’t consider the effect of global prefetching.

In [2], they presented a well studied system model for prefetching. Their prefetching scheme is divided into two parts, prefetching algorithm and threshold algorithm. They made some simplifications by considering network link and server as one server in the model. In this system multiuser can prefetch, and they use M/G/1 and Round-Robin processor-sharing. They also model the normal requests and prefetching requests as Possion process. They define the total cost of a user request as the sum of the system resource cost of transmitting the file and the delay cost of waiting for the file to arrive. Therefore, the cost of a normal request is:

\[ c_1 = \alpha_B \cdot s + \alpha_T \cdot t \]  (2)

where \( s \) is file size , \( t \) is transmission time. If the user requests for a file that had previously been prefetched and saved on the local cache, the cost associated with this request is only

\[ c_2 = \alpha_B \cdot s \]  (3)

Let \( \lambda \) be the arrival rate of user requests when no prefetching is applied. After prefetching, the normal requests rate is \( \lambda_1 \), prefetching requests are \( \lambda_2 \). The likelihood of a page is accessed is \( p \), then we have equation:

\[ \lambda_1 + \lambda_2 \cdot p = \lambda \]  (4)

\[ \lambda_1 + \lambda_2 = \lambda + (1 - p)\lambda_2 \]  (5)

In Round-Robin system, the average system response time for a request requiring \( x \) time units of processing is

\[ t = \frac{x}{1 - \rho} = \frac{s}{b(1 - \rho)} \]  (6)

where \( \rho = \lambda x \) is the system load, \( s \) is the average file size, and \( b \) is the system capacity. Plug (6) into (2) we get the cost of a normal request is

\[ c_1 = \alpha_B \cdot s + \alpha_T \cdot \frac{s}{b - (\lambda_1 + \lambda_2)s} \]  (7)

Since users issue requests with rate, and some of them \((p\lambda_2)\) are satisfied by prefetched files, the
rest ($\lambda_1$) are sent to the WWW server as normal requests. By equation (3),(7) we get

$$C = \frac{\lambda_1 \cdot c_1 + \lambda_2 \cdot c_2}{\lambda}$$  \hspace{1cm} (8)

$$C = \frac{s}{\lambda}((\lambda + (1-p)\lambda_2)\alpha_B + \frac{(\lambda - p\lambda_2)\alpha_T}{b - (\lambda + (1-p)\lambda_2)s})$$  \hspace{1cm} (9)

Derive equation (9), let $\frac{dC}{d\lambda} = 0$ we can get the critical value $\lambda'_{2}$

$$\lambda'_{2} = \frac{1}{(1-p)s}[b - \lambda s - \sqrt{(pb - \lambda s)\alpha_T}}{(1-p)\alpha_B}]$$  \hspace{1cm} (10)

if $\lambda'_{2} \leq 0$, prefetching files with access probability $p$ minimizes the cost. Thus we can get the prefetch threshold $H$

$$p \geq H = 1 - \frac{(1-\rho)\alpha_T}{(1-\rho)^2b + \frac{\alpha_T}{\alpha_B}}$$  \hspace{1cm} (11)

The simulation results of them show that this method works. If we change the cost function (2), like below:

$$c_1 = \alpha_B \cdot s + \alpha_T \cdot t^2$$  \hspace{1cm} (12)

The result $\lambda'_{2}$ will be very complicated (solution of a quartic equation). We suspect it is too complicated and may be not viable in a real environment (because [2] simplify the network link and server as a single server, a too complicated cost function may not work in this condition).

6 Prefetching: pros, cons and reality

After diving into this topic, we found a lot of interesting discussions about this topic. A latest paper [8] even has an assertion that prefetching with GET is not good. The main idea is that without extension of HTTP, prefetching will have side effects like placing another linked item to shopper’s shopping list. After reading this we recognize that we should reflect this topic again. Why do we have so many fancy schemes but not usable? The older version Netscape Navigator even have some prefetching code present, but disabled [9]). We may see the following pros and cons of prefetching:

**Pros:**

- Improve the WWW latency by utilizing the idle time of a user

**Cons:** (also discussed in [8])

- Waste bandwidth for unnecessary data
- The potential increase in queue length and delays in router from prefetching
- The negative effect on the Internet (added traffic and server load) if all clients were to implement prefetching [10]

Given these facts, we think that client and server prefetching have dangerous and effects and limited benefits. Client initiated prefetching need to interact with client to disable or enable prefetching, which seems awkward. Server initiated prefetching can only reduce the objects creation time and couldn’t reduce the internal latency between client and proxy server and external latency between proxy server and content provider. The most viable way is to implement prefetching without changes in protocols is the proxy initiated prefetching. Because of widely used proxy server or content distributor like Akamai[11], the proxy server can have special contract with content provider (For example, after a client fetch a clip of a video from an Akamai server, the Akamai server may prefetch next clip of the video from content provider). It seems that no existing research has been done on this topic: how to make viable
prefetching in real world and how to implement an effective prefetching scheme in proxy server?

7 Conclusion

Caching and prefetching, which make use of temporal and spatial locality respectively, have been carefully studied long before World Wide Web was born, in processor design, operating systems and many other areas. But people have never encountered so huge a distributed information system as WWW before, they kept trying migrating caching and prefetching techniques in other areas into WWW in the mid-90’s, in the hope to reduce network traffic, and user-perceived latency. In general, Web cache has made a big success. But prefetch, on the other hand, has very limited application in WWW, if at all. Someone even tried push (or pre-send) technology, which has been proven to be a failure. Now, it’s time to calm down and think about the frenzy in retrospect to learn some lessons. In this paper, we carefully examined caching and prefetching literature in WWW setting, with emphasis on prefetching, studied and compared various schemes for prefetching(structure-based, Markov model based, etc), at various locations(client/proxy/server), their potential effect on the network as a whole, its potential side effect. We also surveyed different methodologies used in studying prefetching, including analysis, simulation. We found no papers on prefetching using network measurement yet, very few uses analysis, most uses simulation.

8 Direction of future works

Too much work has been done on prefetching in WWW setting, too many attempts have been tried. Thus, in our humble opinion, unless there’s significant change in network architecture, say, network bandwidth becomes infinite, or WWW has undergone a radical restructuring, it’s not worth to try working on prefetching any more, though we do have some ideas in the process of the literature survey. For example, Markov model is good at capturing short-term dependency, but Web traffic has been proven to have long-term dependency, what if find a new model which could capture the long-term dependency, perhaps the new model could also help in prefetching prediction. Another example, we know the key thing in prefetching is to have a precise understanding of access patterns, could we use some results in the field of machine learning to learn the access patterns based on access history? Maybe.

References

[2] Zhimei Jiang, Lenorad Kleinrock, Prefetching Links on the WWW.
[7] Edith Cohen, Haim Kaplan, Reducing user perceived latency by prefetching connec-


