Web Caching & Web Prefetching Integration

A. Nalini
Lecturer (Sr.Gr)
Department of Computer Engineering
Natchimuthu Polytechnic College, Pollachi, India

Abstract—Two important techniques used to reduce the response time are web caching and web prefetching. By integrating Web caching and Web prefetching, these two techniques can complement each other since the Web caching technique exploits the temporal locality, whereas Web prefetching technique utilizes the spatial locality of Web objects. However, without proper design, the integration of these two techniques might cause significant performance degradation to each other. In view of this, an innovative cache replacement algorithm is proposed, which not only considers the caching effect.

Key words: Web Proxy, Caching, Prefetching

I. INTRODUCTION

The growth of the World Wide Web over the past few years has imposed a significant traffic burden upon the Internet. A significant amount of research effort has been elaborated upon investigating effective and scalable solutions to reduce the noticeable response time perceived by users. Among others, Web caching and Web prefetching are two important techniques to this end.

A. Web Caching & Prefetching

According to the locations where objects are cached, Web caching technology can be classified into three categories, i.e., client’s browser caching, client-side proxy caching, and server-side proxy caching. In client’s browser caching, Web objects are cached in the client’s local disk. If the user accesses the same object more than once in a short time, the browser can fetch the object directly from the local disk, eliminating the repeated network latency. However, users are likely to access many in the Web environment, but also evaluates the prefetching rules provided by various prefetching schemes. Specifically, a normalized profit function (including a new metric called the relevance of the web page) to evaluate the profit from caching an object is formulated. Based on the normalized profit function devised, a Web cache replacement algorithm is also devised. Similarly a filtering algorithm is used before caching an object to cache only the popular objects.

Sites, each for a short period of time. Thus, the hit ratios of per-user caches tend to be low. In client side proxy caching, objects are cached in the proxy near the clients to avoid repeated round-trip delays between the clients and the origin Web servers. To effectively utilize the limited capacity of the proxy cache, several cache replacement algorithms are proposed to maximize the delay savings obtained from cache hits. Such advanced caching algorithms differ from the conventional ones (e.g., LRU or LFU algorithms) in their consideration of newly identified parameters in the Web environment, i.e., the size, fetching delay, reference rate, invalidation cost, and invalidation frequency of a Web object. Incorporating these parameters into their designs, these cache replacement algorithms show significant performance improvement over the conventional ones. On the other hand, server-side Web caching and content distribution network (CDN) are recently attracting an increasing amount of attention. It is noted that, as the Web traffic grows exponentially, overloaded Web servers become the sources of the prolonged response time.

Unlike Web caching which exploits the temporal locality of the Web objects, Web prefetching technique takes advantage of the spatial locality of them. Generally, a user’s browsing sequence will follow the hyperlinks between Web objects. That is, if object A has a hyperlink to object B, the probability that B will be accessed, given A has been accessed already, will increase significantly. Hence, if we prefetch those objects which are very likely to be referenced in the client’s subsequent requests, part of the network latency can be hidden within the time between client’s consecutive requests. From this aspect, it is noted that the performance of a Web prefetching scheme mainly relies on the accuracy of the employed prediction algorithm. Markov model is one useful tool for modeling and predicting a user’s browsing behavior over the Web. The prefetching schemes take advantage of Markov models to derive prefetching rules from a Web server’s access log. The work proposed an approach of predicting the users’ next actions based on the contents of Web pages. In summary, various techniques have been utilized for improving the accuracy of predicting user access patterns, making the prefetching of Web objects more effectively.

To maximize the performance improvement by integrating Web caching and Web prefetching, we propose in this paper an innovative cache replacement algorithm, not only considering the caching effect in the Web environment, but also evaluating the prefetching rules provided by various prefetching schemes. Explicitly, rather than using a static confidence threshold to control the prefetching level, we design a normalized profit function to dynamically evaluate the profit from caching an object (either a nonimplified object or an implied object according to some prefetching rule). This normalized profit function considers not only such factors as the size, fetching delay, reference rate, invalidation cost, and invalidation frequency of a Web object, but also the prefetching effect caused by various Web prefetching schemes. Based on the normalized profit function devised, we devise an innovative Web cache replacement Algorithm.

II. EXISTING SYSTEM

A. System Model

This paper studies the integration of Web caching and Web prefetching in client-side proxies. A typical system model is depicted in the Figure. The proxy is located near the Web clients to avoid repeated round-trip delays between the clients and the origin Web servers. The origin Web server in our model is an enhanced Web server which employs a prediction
engine to derive prefetching rules from the server’s access log periodically. These derived rules are assumed to be frequent. That is, only rules with supports larger than the minimum support are derived and provided by Web servers. The derived prefetching rules are stored in the prefetching rule depository of the Web server. As shown in above Fig., the proxy serves the requests sent from the Web clients. In the case that a cache miss occurs, the proxy will forward the request to the origin Web server for resolution. Upon receiving the request, the origin server will log this request into record, fetch the requested object form the Web object depository, and check the prefetch rule depository at the same time. If this request triggers some prefetching rules in the prefetching rule depository, the objects implied by these prefetching rules and their corresponding confidences will be piggybacked to the responding message as hints and returned to the proxy.

Fig. 1: System Model for Web Caching & Web Prefetching

After the proxy receives the response with the hints piggybacked from the origin Web server, the proxy will first send the requested object back to the client and then determine whether it is worth caching the piggybacked implied objects in the proxy. Our study is thus to devise such a cache replacement algorithm for the integration of Web caching and Web prefetching techniques. Note that, in the case that a cache hit is found (i.e., the client’s request can be satisfied directly with the proxy’s local cache), we assume that the proxy will still communicate with the origin Web server to obtain the prefetching hints related to that request after the proxy has sent the response to the client. As such, we are able to investigate each request the prefetching hints from the origin Web server to ensure that the discovered prefetching hints are always up-to-date. Note that the generation of prefetching rules can be also done on the proxy server. Specifically, it is known that the Web page access patterns can be discovered from proxy server logs. The traffic due to the delivery of accessed object index can thus be saved. However, since there are possibly a large number of Web servers that proxy clients may access, the resulted overhead for computing and storing the prefetching rules could be a great burden for the proxy server. In addition, it is more reliable to perform such a computation based on the Web server’s user access logs gathered from various clients.

B. Algorithm IWCP

The normalized profit function, PNf is formulated. Based on this normalized profit function, a cache replacement algorithm is designed. The main idea behind this algorithm is: Order the objects (either a non-implied object or an implied object) according to their values of the normalized

C. Cache Replacement Algorithm

With the background of the system environment described above, a cache replacement algorithm for integrating Web caching and Web prefetching in client-side proxies is proposed. The normalized profit function which will be used to determine the profit from caching either a non-implied object or an implied object is formulated. Utilizing this normalized profit function as the eviction function, an innovative cache replacement algorithm is devised.

D. Normalized Profit Function

Most cache replacement algorithms employ an eviction function to determine the priorities of objects to be replaced (or to be cached). A normalized profit function to determine the profit from caching an object will depend on the profits of caching individual objects. Explicitly, rather than using a static confidence threshold to control the prefetching level, a normalized profit function to dynamically evaluate the profit from caching an object (either a nonimplied object or an implied object according to some prefetching rule) is used. This normalized profit function considers not only such factors as the size, fetching delay, reference rate, invalidation cost, and invalidation frequency of a Web object, but also the prefetching effect caused by various Web prefetching schemes.

Profit function. Then, select the object with the highest value, one by one, until there is not enough space to hold any object more. The resulting set of selected caching an object our eviction policy will depend on the profits of caching individual objects.

E. Problems with Existing Replacement Policies

1) Requires data structure to be implemented.
2) Data structure requires a priority queue to be implemented.
3) Data structure needs to be constantly updated even when there is no eviction.

III. PROPOSED SYSTEM

The project aim is to replace the Least Recently Used document in web caches using Randomized web cache replacement algorithm.

A new filtering algorithm is incorporated before caching an object. (IE) Only popular object is cached.

A web cache sits between web server and a client and watches request for web pages. It caches web documents for serving previously retrieved pages when it receives a request for them.

A. Need for Web Cache

The various modules that are done as a hierarchical process are like

1) Frame Designing
2) Proxy Designing
3) Algorithm implementation
4) Web cache Management
5) Test and implementation
6) To reduce latency
7) To reduce traffic
8) To reduce load on web servers
B. Operations supported by Web Cache
- Access to web documents
- Eviction of document to make room for new document

C. Methodology
Project work is first started with the intention of overcoming the disadvantages of the existing system. The drawback is that there can be only few selected number of users, limited number of the web page access, if exceeds the slow retrieval timing.

To overcome these the randomized algorithm is applied so that the text frames (files) are stored in separate directory (folder) and the pictures are stored in the directory (folder). They are done with single internet connection and equally shared bandwidth. If the specific page is already seen or opened the time consumption is very less as the text frame and the pictures are already stored in the local folder, the work is to display the content. The upgrading of web-site is done automatically with the help of the header contents of the web page.

D. Implementation Outline
This project web cache (proxy server) is to develop a utility to share internet from single connection to a large network around 200 machines with different operating systems. The software, which should be developed uses the Java Language. Java applet applications are mostly used in the web pages, but we use JFC (swing) for developing the software. Sun Microsystems does not provide any kind of tool.

This project provides an intelligent environment containing a number of ready-made options like cache, log file, error checking, connection pooling, etc. These ready-made tools may be any of the GUI components that are available in the Java AWT package. By using this utility, Administrator can control and maintain the whole network. The Projects aim is to replace the Least Recently Used document in web caches using Randomized web cache replacement algorithm.

IV. SOFTWARE/HARDWARE REQUIREMENTS
Java is a technology that makes it easy to build distributed application over the network. Java allows the user to do the following Writing robust and reliable programs.

Build on application on almost any platform and run the application on any platform without recompiling the code.

Distribute your application over the network in a secured fashion.

A. Features
There are six features in Java that makes java as a power tool for Internet applications,
- Security.
- The core API.
- Open Standard.
- Distributed and Dynamic.
- Object oriented.
- Multithreaded.

B. Requirements
The minimum requirements for the project:
- CPU Type: Pentium IV
- CPU Clock: 1.4 GHz
- Hard Disk: 20 GB
- RAM Memory: 512 MB
- Operating System: Windows 9x/2000/ME/XP

C. Comparison with Existing System
1) Randomized algorithm is proposed to support the eviction decision.
2) Avoids the need for data structures.
3) The utility function assigns to each page a value based on -recentness of use -frequency of use -size of page - cost of fetching

V. CONCLUSION
In this paper, a cache replacement Algorithm is devised by integrating Web caching and Web prefetching in client-side proxies. A normalized profit function is formulated to evaluate the profit from caching an object (either a nonimplied object or an implied object by a certain prefetching rule). This normalized profit function not only considers such factors as the size, fetching cost, reference rate, invalidation cost, and invalidation frequency of a Web object, but also exploits the effect caused by various Web prefetching schemes. A new metric called relevance of a web page is also included in finding the profit of a web page. Utilizing this normalized profit function as the eviction function, a replacement algorithm is devised

Similarly before caching a prefetched web page, a filtering algorithm is used to cache only the popular objects. The various applications for this proxy are:
- Large networks
- Load Balancing

REFERENCES

