ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

A Data Mining Algorithm for Long Term Web Prefetching

C.Anuradha,

Assistant Professor, Department of CSE, Bharath University, Chennai, TN, India.

ABSTRACT: The World Wide Web is the Internet's most widely used tool for information access and dissemination, but today's users often experience long access latency due to network congestion—particularly during peak hours and big events, such as the Olympic Games. Caching frequently used data at proxies close to clients is an effective way to alleviate these problems. Specifically, caching can reduce load on both the network and servers (by localizing the traffic) and improve access latency (by satisfying user requests from local storage rather than remote servers).

Cache Prefetching is related to replacement, but unlike data caching, which waits on object requests, Prefetching proactively preloads data from the server into the cache to facilitate near-future accesses. However, a cache Prefetching policy must be carefully designed: if it fails to predict a user's future accesses, it wastes network bandwidth and cache space. The prediction mechanism thus plays an important role. We are proposing the system where predicting the user's future access is done by analyzing the two or three pages of his access. The pages found in accordance with the analyzed pages will be shown to the user. The user can select his required pages from that list.

I. INTRODUCTION

A Web cache sits between Web servers (or origin servers) and a client or many clients, and watches requests for HTML pages, images and files (collectively known as objects) come by, saving a copy for itself. Then, if there is another request for the same object, it will use the copy that it has, instead of asking the origin server for it again.

There are two main reasons that Web caches are used:

1. To reduce latency - Because the request is satisfied from the cache (which is closer to the client) instead of the origin server, it takes less time for the client to get the object and display it. This makes Web sites seem more responsive.
2. To reduce traffic - Because each object is only gotten from the server once, it reduces the amount of bandwidth used by a client. This saves money if the client is paying by traffic, and keeps their bandwidth requirements lower and more manageable.

If you examine the preferences dialog of any modern browser (like Internet Explorer or Netscape), you'll probably notice a 'cache' setting. This lets you set aside a section of your computer's hard disk to store objects that you've seen, just for you. The browser cache works according to fairly simple rules. It will check to make sure that the objects are fresh, usually once a session (that is, the once in the current invocation of the browser). This cache is useful when a client hits the 'back' button to go to a page they've already seen. Also, if you use the same navigation images throughout your site, they'll be served from the browser cache almost instantaneously.

Web proxy caches work on the same principle, but a much larger scale. Proxies serve hundreds or thousands of users in the same way; large corporations and ISP's often set them up on their firewalls . Because proxy caches usually have a large number of users behind them, they are very good at reducing latency and traffic. That's because popular objects are requested only once, and served to a large number of clients.

A server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.

Proxy servers have two main purposes:

1. Improve Performance**:** Proxy servers can dramatically improve performance for groups of users. This is because it saves the results of all requests for a certain amount of time. Consider the case where both user X and user Y access the World Wide Web through a proxy server. First user X requests a certain Web page, which we'll call Page 1. Sometime later, user Y requests the same page. Instead of forwarding the request to the Web server where Page 1 resides, which can be a time-consuming operation, the proxy server simply returns the Page 1 that it already fetched for user X. Since the proxy server is often on the same network as the user, this is a much faster operation. Real proxy servers support hundreds or thousands of users. The major online services such as America Online, MSN and Yahoo, for example, employ an array of proxy servers.
2. Filter Requests**:** Proxy servers can also be used to filter requests. For example, a company might use a proxy server to prevent its employees from accessing a specific set of Web sites.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

Web caches are widely used in the current Internet environment to reduce the user-perceived latency of object requests. One example is a proxy server that intercepts the requests from the clients and serves the clients with the requested objects if it has the objects stored in it; if the proxy server does not have the requested objects, it then fetches those objects from the Web server and caches them and serves the clients from its cache. Another example is local caching that is implemented in Web browsers. In the simplest cases, these caching techniques may store the most recently accessed objects in the cache and generally use an LRU replacement algorithm that does not take into account the object size and object download cost. Cao and Irani developed a Greedy Dual-Size algorithm that is a generalization of the LRU replacement algorithm to deal with variable object sizes and download times and it was shown to achieve better performance than most other Web cache evicting algorithms in terms of hit rate, latency reduction, and network traffic reduction. However, all these techniques use on-demand caching or short-term caching and the objects to be cached are determined by the recent request patterns of the clients. Long-term prefetching, on the other hand, is a mechanism that allows clients to subscribe to Web objects to increase the cache hit rates and thus reduce user latency. The Web servers or Web object hosts proactively "push" fresh copies of the subscribed objects into Web caches or proxy servers whenever such objects are updated. This makes the user hit rates for these objects always 1. The selection of the prefetched objects is based on the long-term (statistical) characteristics of the Web objects, such as their (average) access frequencies, update intervals, and sizes, rather than the short-term (recent) access patterns at individual caches The long-term characteristics are obtained and maintained by collaboration among content distribution servers. The statistics may be collected and published within a specific domain, such as a news Website and its subscribed customers. A prefetching mechanism may be applied in this domain to increase the performance. Intuitively, to increase the hit rate, we want to prefetch those objects that are accessed most frequently; to minimize the bandwidth consumption, we want to choose those objects with longer update intervals. We assume unlimited cache sizes for both on-demand and prefetching .

Our current research focus is to apply similar machine learning mechanisms to the problem of action prediction on the web. In particular, we wish to be able to predict the next web page that a user will select. If one were able to build such a user model, a system using it could anticipate each page retrieval and fetch that page ahead of time into a local cache so that the user experiences very little retrieval latency, and so reduce widespread complaints about the ``World-Wide Wait''. Thus, performance measurement of such a system is primarily in terms of user-perceived latency. However, since a perfect prediction system is impossible, we must also consider side-effects of prefetching incorrectly, such as increased server loads and bandwidth usage

Due to the fast development of internet services and a huge amount of network traffic, it is becoming an essential issue to reduce World Wide Web user-perceived latency. Although web performance is improved by caching, the benefit of caches is limited. To further reduce the retrieval latency, web prefetching becomes an attractive solution to this problem. Prefetching reduces user access time, but at the same time, it requires more bandwidth and increases traffic. Performance measurement of prefetching techniques is primarily in terms of hit ratio and bandwidth usage. A significant factor for a prefetching algorithm in its ability to reduce latency is deciding which objects to prefetch in advance. In today's common web configurations, the proxy server exists between clients and web servers. The proxy server intercepts the requests from clients, serves with the requested object if it is present in proxies, or retrieves new objects from the appropriate web server, then caches the new objects or updates the existing objects if it has been modified since the last reference. 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. Among them, bandwidth is likely to be the primary limiting factor. So, we add moderate bandwidth requirements as an important performance measure for a good web prefetching model. Our solution space for web prefetching spreads according to these two performance evaluation factors: hit ratio and bandwidth consumption.

1. This project uses a family of prefetching algorithms, Objective-Greedy prefetching, that are directed to improve the performance in terms of the various objectives that each algorithm is aimed at —hit rate, bandwidth, or H/B metric. The Objective-Greedy prefetching algorithm, H/B-Greedy has lineartime complexity and is easy to implement. The H/B-Greedy prefetching aims to improve the H/B metric, which combines the effect of increasing hit rate (H) and limiting the extra bandwidth (B) consumed by prefetching. This criterion was first, as a performance metric in terms of both hit rate and bandwidth usage.. We hypothesize that our H/B-Greedy prefetching achieves close-to optimal H/B-performance and this is justified by our simulation results..
2. The project shows the results of a simulation analysis of the above algorithm in terms of the hit rate, bandwidth, and H/B metrics.

Each of the proposed prefetching algorithms is seen to provide better performance than any existing algorithms based on the respective prefetching objective. In particular, the best trade-off between increasing hit rate and reducing extra bandwidth usage is obtained when the H/B metric is used.

Our current research focus is to apply similar machine learning mechanisms to the problem of action prediction on the web. In particular, we wish to be able to predict the next web page that a user will select. If one were able to build such a user model, a system using it could anticipate each page retrieval and fetch that page ahead of time into a local cache so that the user experiences very little retrieval latency, and so reduce widespread complaints about the ``World-Wide Wait''. Thus, performance measurement of such a system is primarily in terms of user-perceived latency. However, since a perfect prediction system is impossible, we must also consider side-effects of prefetching incorrectly, such as increased server loads and bandwidth usage.

3.1 EXISTING SYSTEM:

- ❖ Additional communication between the server and the client is needed in order to realize the Prefetching scheme.
- ❖ Fast recovery is not possible if there is any node or link failure then it has to fetch from the server.
- ❖ Congestion is unavoidable and traffic is more
- ❖ Time consuming is very large
- ❖ Flexibility is less

3.2 PROPOSED SYSTEM:

- ❖ As the Frequently accessed files are stored in the Cache, Fast recovery is possible.
- ❖ Accuracy in prediction with quite low overhead in network traffic.
- ❖ Traffic is reduced and bottlenecks are avoided.
- ❖ As retrieving is done fast ,time is saved
- ❖ Flexibility is more
- ❖ We predict the future access by analyzing the current usage of the user. It is the way of dynamic prediction.

ALGORITHM

This project uses H/B greedy algorithm as the web prefetching algorithm. It proposes a prefetching algorithm which does an objective-Greedy prefetching, which is directed to improve the performance in terms of the various objectives that this algorithm is aimed at —hit rate, bandwidth, or H/B metric. H/B-Greedy algorithm is easy to implement. The H/B-Greedy prefetching algorithm aims to improve the H/B metric, which combines the effect of increasing hit rate (H) and limiting the extra bandwidth (B)
consumed by prefetching.

The web prefetching algorithm can be better analyzed by using the H/B model. It uses the H/B metric . It is defined by

$$H/B = \frac{Hitpref / Hitdemand}{BWpref / BWdemand}$$

Here, Hitpref and Hitdemand are the overall hit rate, with and without prefetching, respectively; BWpref and BWdemand are the total bandwidth with and without prefetching. The H/B metric expresses the ratio of hit rate improvement over the bandwidth increase. It is a quantitative evaluation of the hit rate improvement a prefetching algorithm can bring relative to excessive bandwidth consumption. In addition, a more generalized form, Hk=B can be used to give relative importance to hit rate or bandwidth by varying the value of k.

With this form of the H/B metric, $k > 1$ indicates an environment with abundant available bandwidth and hardware resources and the improvement of hit rate is
more favored than the economy on bandwidth. When the network bandwidth is limited, a smaller k (possibly $k < 1$) is used instead. The access pattern of an object i is assumed to follow the Poisson distribution with the average access rate being api, and the update interval is assumed to follow the exponential distribution with the average interval being li . For an object i that is not prefetched, a current access is a hit if the last access occurred after the last update and the probability of an access being a hit is given as follows:
Phit(i) = aPili / (aPili + 1)

If the object i is not prefetched , the hit rate of that object is Phit(i).If it is already prefetched ,then the hit rate of that object is equal to 1. Thus the steady state hit rate of the objects can be given by the above. Similarly, the steady state bandwidth consumption can be given by the following. Let si be the size of object i. If object i is not prefetched, then only when an access results in a cache miss would this object be retrieved from the Web server. Thus, the bandwidth for this object is api(1 – f(i))si. If object i is prefetched, then this object is downloaded from its Web server to the cache each time it is updated in the server and the bandwidth is si/ li.

5.1 H/B GREEDY PREFETCHING

Note that a prefetching algorithm selects a subset of objects from the cache. When an object i is prefetched, the hit rate is increased from apili /(apili+1) to 1, which is 1/f(i) times that of ondemand caching; the bandwidth for object i is increased to si/li , which is also 1 /f(i) times the bandwidth for object i under on-demand caching. Prefetching an object leads to the same relative increase in its hit rate and bandwidth consumption. Recall the H/B metric that measures the balanced performance of a prefetching algorithm. We observe that this measure uses the hit rate and bandwidth of on-demand caching as a baseline for comparison.
In this section, we introduce a greedy algorithm, the H/BGreedy prefetching algorithm, as a first solution toward designing an optimal solution to the problem. It aims
to improve H/B by greedily choosing objects based on their individual characteristics. The H/B-Greedy algorithm attempts to select those objects that, if prefetched, would have
the most benefit for H/B. For example, suppose, initially, no object is prefetched and the H/B value is expressed as (H/B)demand; now, if we prefetch object j, the H/B value will be
updated to some other value. That increase in the value is given by the increase factor incr(j). Here, incr(j) is the factor that indicates the amount by which H/B can be increased if object j is prefetched. We call incr(j) the increase factor of object j.

Our H/B-Greedy prefetching algorithm thus uses increase factor as a selection criterion and chooses to prefetch those objects that have the greatest increase factors.
We hypothesize that this algorithm is an approximation to the optimal solution because, in real-life prefetching, due to the bandwidth constraints, the number of prefetched objects m is usually small. Furthermore, the increase factors of individual objects are generally very close to 1 due to the large number of objects in the system. Thus, previous
selection of prefetched objects is expected to have little effect on subsequent selections in terms of improving H/B.

Procedure H/B-Greedy takes the set of all objects in the Web servers—S, the number of objects to be prefetched—m, and the total access rate—a, as the inputs. Each object i is represented by a tuple <pi, li, si>, where pi, li, and si denote the access frequency, the lifetime, and size of the object, respectively.

- **FUTURE ENHANCEMENT:**
- 1.For the H/B-Greedy, the simulations showed that, when the number of prefetched objects is equal to some value nmax, the H/B metric attains the globally maximum value. This nmax is helpful in determining how many objects to prefetch in order to maximize the efficiency of network resources. It is a challenge to determine the nmax value for H/B-Greedy , given the total number of objects.
- 2.The principles of Web prefetching studied in this have potential applications in the fields of a) wireless networks where the power control and bandwidth control are of special importance and b) P2P networks and networks requiring dynamic Web access, where the data availability and load balancing are more important . The challenge is to adapt and extend the results of this paper to such networks.

REFERENCES

[1] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, “Characterizing Reference Locality in the WWW,” Proc. IEEE Conf. Parallel and Distributed Information Systems (IEEE PDIS ’96),pp. 92- 103, Dec. 1996.
[2] P. Atzeni, G. Mecca, and P. Merialdo, “To Weave the Web,” Proc. 23rd Conf. Very Large Data Bases (VLDB ’97), pp. 206-215, Aug. 997.
[3] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proc. 20th Conf. Very Large Data Bases (VLDB’94), pp. 487-499, Sept. 1994.
[4] R. Agrawal and R. Srikant, “Mining Sequential Patterns,” Proc.IEEE Conf. Data Eng. (IEEE ICDE ’95), pp. 3-14, Mar. 1995.
[5] C. Aggarwal, J. Wolf, and P. S. Yu, “Caching on the World Wide Web,” IEEE Trans. Knowledge and Data Eng., vol. 11, no. 1, pp. 95- 107, Jan./Feb. 1999.
[6] P. Barford and M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation,” Proc. ACM Conf. Measurement and Modeling of Computer Systems, (ACM SIGMETRICS ’98), pp. 151-160, June 1998.
[7] A. Bestavros, “Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time,” Proc. IEEE Conf. Data Eng. (IEEE ICDE ’96), pp. 180-189, Feb. 1996.
[8] B. Berendt and M. Spiliopoulou, “Analysis of Navigation Behavior in Web Sites Integrating Multiple Information Systems,” The VLDB J., vol. 9, no. 1, pp. 56-75, May 2000.
[9] M. Crovella and P. Barford, “The Network Effects of Prefetching,” Proc. IEEE Conf. Computer Comm. (IEEE INFOCOM ’98), pp. 1232-1240, Mar. 1998.

[10] P. Cao and S. Irani, “Cost-Aware WWW Proxy Caching Algorithms,” Proc. 1997 USENIX Symp. Internet Technologies and Systems (USITS ’97), pp. 193-206, Jan 1997.