***Chapter 1***

**INTRODUCTION**

**1.1 General Introduction:**

The Internet is considered as a giant distributed system that provides access to the shared web objects like web pages, images and other types of digital objects. Due to the exponential growth of the internet, it creates problems too, like network congestion, server overloading etc. Web caching has been recognised as one of the best technique to reduce this network congestion. It also improves the performance of web by minimize the user access latency. Prefetching is another powerful technique, that has been using along with web caching for better performance. Web caching along with prefetching doubles the performance when compared to the web caching alone.

**1.2 Web caching:**

Web caching is a technique where web objects stored in cache when they are accessed from the main server for the first time and subsequent requests for those objects are served from the cache directly. With this technique performance of web increases since the latency caused by the request decreases. For every request first it will check the cache for a valid copy. If it is not a valid one or the copy is staled one then it directly goes to the main server. Since web objects are stored in cache, it even reduces the workload at the main server. A user can access the web objects from the cache itself even though the origin server is down.

Prefetching is a technique that predicts the future requests based on the past access patterns or based on the content of current page and fetches them into the cache during the network idle time. Sometimes if the prediction goes wrong the prefetched objects will no longer be useful to the user. In those cases the network traffic will increase because of these unused objects. But overall it improves the performance. Web caching along with prefetching improves the performance greatly. [1] Shows combining the web caching along with prefetching doubles the performance when compared to web caching alone.

**1.2.1 Types of web cache**: Web caching can be deployed at three levels: Client side, proxy side and server side.

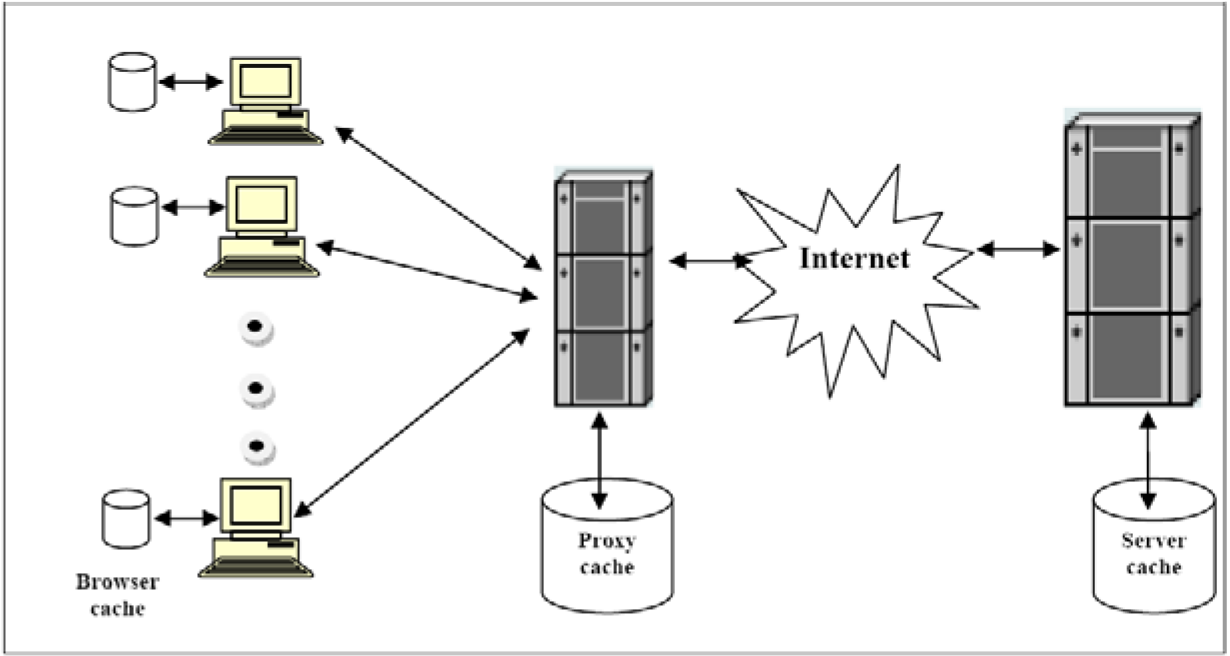


Figure 1.1: Types of web cache

**Browser cache**: Here the cache is located on the client machine. It deals with single client and multiple servers. That is, web objects pertaining to various servers will be stored. The size of browser cache is small. Examples are internet explorer, Mozilla firefox, google chrome etc., where we can notice cache setting of any of these browser.

**Proxy cache**: Here cache is maintained between the main server and the clients. Most of the organizations maintains it near to the clients in order to save bandwidth latency. Unlike browser cache it supports more cache objects since the size of cache is big compared to browser cache. It deals with many clients as well as many servers.

**Reverse proxy cache**: This technique is also called gateway cache. In this caches are deployed at the end server. It deals with single server and many clients. It helps in reducing the server load, since we are maintaining cache at end server. If a requested object is not present in the cache then it redirects to the server and updates the cache based on the corresponding cache algorithm.

**1.3 Caching architectures**:

The efficiency of web caching increases with number of clients. Since increasing the number of users will also increases the probability of accessing the same object. As a result cache hit ratio increases. Based on how a group of caches are interacting, cache architecture is divided into three ways.

**1.3.1 Hierarchical caching architecture:** In this approach caches can be placed at multiple levels of the network like level 1, level 2, and so on. If the object is not present in lower lever cache then it will go for next higher level cache. If an object is not present in any of these levels then it will fetches the object directly from the main server and it leaves a separate copy at all caching levels. Hierarchical caching architecture is bandwidth efficient. However, it suffers from problems like delay at each cache level, same web object copy might present in all cache levels etc.

**1.3.2 Distributed caching architecture:** In this approach there is no intermediate cache levels like hierarchical caching architecture. Here a group of cache servers scattered across the network will communicate with each other. In this, all cache servers maintain a cluster data information which contains information regarding other cache servers. If an object is not present in the local cache then based on the cluster data information it sends a request to the neighbour cache server. In this approach most of the network flows through lower network levels. The Harvest group designed the Internet Cache Protocol (ICP) [79], which supports discovery and retrieval of documents from neighbouring caches as well as parent caches.

**1.3.2 Hybrid architecture:** In this caches will work together corporately. Object will fetch from either parent cache that is higher level cache (hierarchical architecture) or from neighbour cache that is cache at same level (distributed architecture) depending on round trip delay.It is preferable to fetch the object from origin server if it takes more time to fetch from neighbour cache.

P. Rodriguez et. al in [32] shows that hierarchical caching has shorter connection times than distributed caching, and hence, placing additional copies at intermediate levels reduces the retrieval latency for small web objects. It’s also shown that distributed caching has shorter transmission times and higher bandwidth usage than hierarchical caching. A well configured hybrid scheme can combine the advantages of both hierarchical and distributed caching, reducing the connection time as well as the transmission time.

**1.4 Desirable properties of web caching system**

**Fast access:** From users’ point of view, access latency is an important measurement of the quality of Web service. A desirable caching system should aim at reducing Web access latency. In particular, it should provide user a lower latency on average than those without employing a caching system.

**Robustness:** From users’ prospect, the robustness means availability, which is another important measurement of quality of Web service. Users desire to have Web service available whenever they want. The robustness has three aspects. First, it’s desirable that a few proxies crash wouldn’t tear the entire system down. The caching system should eliminate the single point failure as much as possible. Second, the caching system should fall back gracefully in case of failures. Third, the caching system would be design in such a way that it’s easy to recover from a failure.

**Transparency:** A Web caching system should be transparent for the user, the only results user should notice are faster response and higher availability.

**Scalability:** We have seen an explosive growth in network size and density in last decades and is facing a more rapid increasing growth in near future. The key to success in such an environment is the scalability. We would like a caching scheme to scale well along the increasing size and density of network. This requires all protocols employed in the caching system to be as lightweight as possible.

**Efficiency:** There are two aspects to efficiency. First, how much overhead does the Web caching system impose on network? We would like a caching system to impose a minimal additional burden on the network. This includes both control packets and extra data packets incurred by using a caching system. Second, the caching system shouldn’t adopt any scheme which leads to under-utilization of critical resources in network.

**Load balancing:** It’s desirable that the caching scheme distributes the load evenly through the entire network. A single proxy/server shouldn’t be a bottleneck (or hot spot) and thereby degrades the performance of a portion of the network or even slow down the entire service system.

**Ability to deal with heterogeneity:** As networks grow in scale and coverage, they span a range of hardware and software architectures. The Web caching scheme need adapt to a range of network architectures.

**Adaptivity:** It’s desirable to make the caching system adapt to the dynamic changing of the user demand and the network environment. The adaptivity involves several aspects: cache management, cache routing, proxy placement, etc. This is essential to achieve optimal performance.

**Stability:** The schemes used in Web caching system shouldn’t introduce instabilities into the network. For example, naïve cache routing based on dynamic network information will result in oscillation. Such an oscillation is not desirable since the network is under-utilization and the variance of the access latency to a proxy or server would be very high.

**1.5 Dynamic data caching:** Web caching scheme is limited by cacheable data. That is, it is not possible to store non-cacheable data (i.e. personal data, authenticated data and server dynamically generated data etc.) at proxy levels. And one more disadvantage is many web sites disable caching of documents. The reason behind that is many sites contain ads and they collect money from them based on number of hits of the document. If the document is cached then it is not possible to do so and other reason might be to collect user information that who is using the content. Dynamic data caching approaches can be classified into two categories: active cache and server accelerator.

Active cache migrates parts of server processing on each user request to the caching proxy in a flexible, on demand fashion via cache applets. A cache applet is a server supplied code that is attached with a universal resource locator (URL). When a user requests a hits on the cached copy and the proxy would like to service the request, the proxy must invoke the cache applet with the user request. The applet then decides what the proxy will send back to the user, either giving the proxy a new web object to send back to the user, or allowing the proxy to use the cached copy. Cache applets allow servers to obtain the benefit of proxy caching without losing the capability to track user accesses.

Web server accelerator resides in front of one or more web servers to speed up user accesses. It provides an API which allows application programs to explicitly add, delete and update the cached data. The API allows the accelerator to cache dynamic as well as static data.

**1.6 Cache replacement mechanism:**

Cache replacement algorithm plays a vital role in web caching. Due to the limited size of the cache, if the cache is full and a new web object need to be accommodated , then this algorithm will decide which objects is to be evicted based on some criteria like object size or lifetime to make space for new object.If the contents of cache are not so frequently used in the near future then it leads to **cache pollution**. Because of these unused objects, the cache miss rate increases. In order to avoid this, better caching algorithm required.

**Traditional mechanisms**: Least recently used (LRU) and Least frequently used (LFU) are most commonly used caching algorithms for selecting a victim. In LFU, object with least frequency is select for replacement. But it suffers from hot cache pollution. Objects present in the cache are due to their access frequency in the past and currently those are not accessing then it is called hot cache pollution.

**Cache pollution:** If the cache is almost filled with unused objects which are never going to use in the near future, then it is termed as cache pollution. It is of two types: 1. Cold cache pollution 2. Hot cache pollution.

If the cache is completely filled with unpopular objects then it is termed as cold cache problem. LRU based algorithms are mostly suffers due to this problem. Because the recently used object is placed at the top of the stack even though it is not popular. It will take some time to move this object to the bottom. If the cache is filled with the previously popular (hot) web objects and currently no longer to the user then it is termed as hot cache pollution. LFU based algorithms are mostly suffers because of this problem. There, for every object certain count value is maintained. Even though the web object is no longer useful to the user in the near future if contains maximum count value because of the previous accesses it remains in the cache. In [5] they provided solutions to these problems.

**1.7 Metrics for Comparing web caching Techniques**

The standard metrics for analysing the performance of web caching are hit ratio (HR), byte hit ratio (BHR) and latency saving ratio (LSR). Each of these terms are defined as follows.

**1.7.1 Hit ratio (HR):** It is defined as the percentage of requests the cache satisfy. Mathematically it is the number of requests that are served from the cache to the total number of requests. Let N be the total number of requests and Si =1 if the request i is served from cache else 0.

HR= =∑ni=1 Si/N

In this, hit ratio is high if the cache is filled with more number of frequently used objects with less size. It doesn’t bother about how much latency has been saved.

**1.7.2 Byte hit ratio (BHR):** BHR is the number of bytes satisfied from the cache as a fraction of the total bytes requested by user.Mathematically it is the number of bytes present in the cache to the number of bytes requested.

BHR=∑ni=1bi\* Si /=∑ni=1bi

Where bi is the size of web object returned for the ith request.

In this case, size of objects only matters not number of objects. That is, less number of objects where every object has more size and more number of objects in which every object is of small size are treated like same.

**1.7.3 Latency saving ratio (LSR):** LSR is defined as the ratio of the sum of download time of objects satisfied by the cache over the sum of all downloading time.

LSR=∑ni=1ti\*Si/=∑ni=1\*ti

Where ti is the time to download the ith referenced object from its server to the cache.

Size of cache plays crucial role on performance. If the cache size is high it can accommodate more number of web objects. So accuracy of hit ratio (HR) and byte hit ratio (BHR) depends on cache size.

Cache object life time is also important for all these metrics. If the cache is filled with frequently used objects with more object life time then accuracy is always high.

**1.8 Web prefetching:**

Web prefetching is a technique that predicts the next user request based on some criteria like users past history, structure of current web page etc. and fetches them in to the cache during network idle time from clients perspective. If the prediction goes wrong then it effects the network traffic. It is possible to set low priority to the prefetching requests when current pending requests are high at server side. With the help of prefetching user experiences less amount of time to access a web object. In [10] they shown that combination of web caching with prefetching improves performance up to 60%, whereas caching alone improves the performance up to 30%.

Web prefetching mainly consist of two modules -1. Prediction module and 2. Prefetching module. Prediction module runs an algorithm to predict the next user requests. Based on some criteria like present web page structure or history of previously requested pages this algorithm predicts the subsequent requests. These requests will be forwarded to prefetching engine and the prefetching engine will fetch those objects.

User Request

Prediction

Prefetching Request

X

Z

Y

Fig. 1.2: Web prefetching

From the above figure (1) Z indicates set of all prediction requests done by the prediction engine, and Y indicates set of fetching objects done by the prefetching engine. Out of all these requests (Y) only set X of objects are useful for the clients.

**Types of prefetching**

Prefetching can be implemented at three levels. 1. Client side, 2.proxy side, and 3.server side.

**Client side prefetching:** Here it concentrates on the navigation patterns of the single client across many servers. Here both prediction module and prefetching module runs on client side. Based on the client’s navigation patterns the prediction algorithm works.

**Proxy side prefetching:** Here it concentrates on the navigation patterns of many clients across many servers. The prediction module runs the algorithm based proxy log. In this, the prefetching objects can be shared by many clients.

**Server side prefetching:** It concentrates on the navigation patterns of many clients over single server. [8, 9] used prediction module runs on server side and prefetching module runs on clients side. When a request comes to the server, in addition to processing that request it also piggybacks prefetching information to the client. Prefetching engine at client prefetch those requests during the idle period.

Table 1: Comparison of different prefetching schemes at different locations

|  |  |  |  |
| --- | --- | --- | --- |
| **Prefetching location** | **data for prediction model** | **Advantages** | **Disadvantages** |
| **Client** | Historical and current user requests | Easy to partition user session and realize personalized prefetching. | Not share prefetching content among users.  Needs a lot of network bandwidth. |
| **Proxy** | Proxy log and current user requests | Reflects common interests for a group of users.  Shares prefetching content from different servers among users. | Not reflect common interests for a single Website from all users. |
| **Server** | Server log and current user requests | Records a single website access information from all users and better reflect all users’ common interests. | Not reflect users’ real browsing behaviour.  Difficult to partition user session.  Needs additional communications between clients and servers for deciding prefetching content. |

**1.8 Performance measures for web prefetching:** There are several metrics to analyse the efficiency and performance of web prefetching. The following metrics are frequently used for evaluating performance of web prefetching.

**1.8.1 Precision:** It is defined as number of prefetch hits to the total number of objects prefeched.

Precision=number of prefetch hits/Total prefetches.

**1.8.2 Byte precision:** Byte precision measures the percentage of prefetched bytes that are subsequently requested. It can be calculated by replacing the number of prefetched objects with their size in the equation above.

**1.8.3 Recall:** It is defined as number of prefetch hits to the total number of requests.

Recall=number of prefetch hits/Total number of requests.

**1.8.4 Traffic Increase**: It is defined as the bytes transferred through the network when prefetching is done to the bytes transferred in the non-prefetching case.

**1.9 Integrating Web caching and prefetching**

Web caching exploits temporal locality, whereas prefetching technique utilizes the spatial locality of web objects. Combining these two features will doubles the performance when compared to the single feature alone.

In [8] they integrated these two techniques for better performance. At client side proxy they placed a cache where cached and prefetching objects will store. They implemented an algorithm called ICWP for cache replacement and used the existing markov model for prefetching. And for prefetching they used the existing markov model. Upon receiving a user request the proxy will check for a valid copy. If it is a hit then directly it serves the request. If it is a miss, then it will redirect it to the main server. Upon receiving a request the server logs the request in to its records and fetches the requested object. Meanwhile the prefetching engine works. These two operations will be done simultaneously. Then the server sends the requested object to the proxy. During idle times it also sends prefetching data to the proxy. If the cache size is full then, based on the ICWP algorithm it selects victim for object replacement.

**1.10 Challenges in web caching**

1. **Latency caused by cache miss** If the web object is not present in the cache then it causes an extra delay due to the search time in cache.
2. **Stale data** Some servers frequently updates the data. Because of these
3. **Non-cacheable data** It is not possible to cache the all content. Example Personalized data, authenticated data and server dynamically generated data.

**1.11 Challenges in prefetching**

1. **Prefetch miss** if the prediction goes wrong then the prefetched object is no longer useful to the user. Because of these unused objects the cache is polluted.
2. **Band width consumption** Since we are prefetching the web objects and storing them in to the cache it effects the efficiency of costly use of bandwidth. If those prefetched objects are not useful then the efficiency of the bandwidth decreases.

**1.12 Motivation of the work**

Even though, a lot of web caching and prediction algorithms have been proposed, there is a lot of scope to propose and develop new ideas in this area. Even a small change in web caching and prefetching algorithms, improves the user experience greatly. Because of caching the web object, retrieval of the web object is possible, even though the origin server is down. Web caching even reduces the costly use of bandwidth since retrieving the web object is from cache server itself instead of going to the origin server. Prefetching saves the user access latency, because the next subsequent user request is already present at cache server. The main motivation to this thesis is saving the costly use of bandwidth by caching the objects at proxy level server and reducing the user access latency by implementing an efficient prediction algorithm.

**1.13 Problem statement and objectives**

To analyse and improve web caching from existing web caching techniques by exploiting efficient ways of prefetching based on history based and cost function based approaches.

## Milestones

1. Compare and analyze existing caching and prefetching schemes for observing efficiencies.
2. To propose a modified prefetching scheme based on past history of clients.
3. Simulate and evaluate the scheme as well as compare with existing schemes.

**1.14 Thesis outline:**

The Thesis will start by a theoretical study of web caching and prefetching methods, presenting a state of the art of different technical solutions and selecting the most promising algorithms.

The rest of the thesis is organized as follows.

Chapter 2 consists of related work done previously in web caching and prefetching. This chapter critically reviews the previous work done in web caching and prefetching.

In chapter 3, propose methods on web caching and prefetching

In chapter 4, simulation and results of web caching along with prefetching compared to the results of existing approaches.

Chapter 5 gives the conclusion and future scope of the web caching along with prefetching.