Seminar Report
on
Web Caching

Presented by
Sai Rahul Reddy P
04IT6011

School of Information Technology
Indian Institute of Technology- Kharagpur
# Table of Contents

Abstract ............................................................................................................................................. 3
1. Introduction ........................................................................................................................................ 3
   1.1 Advantages of Caching ..................................................................................................................... 3
   1.2 Disadvantages of using a caching: ................................................................................................... 4
2. Proxy Caching ...................................................................................................................................... 4
   2.1 Reverse Proxy Caching: ..................................................................................................................... 5
   2.2 Transparent Caching: ........................................................................................................................ 5
3. Caching architectures ............................................................................................................................ 6
   3.1 Hierarchical caching architecture ..................................................................................................... 6
   3.2 Distributed caching architecture ..................................................................................................... 6
4. Prefetching ........................................................................................................................................... 6
   4.1 Between browser clients and Web servers ......................................................................................... 7
   4.2 Between proxies and Web servers ..................................................................................................... 7
   4.3 Between browser clients and proxies ................................................................................................. 7
5. Cache placement/replacement .............................................................................................................. 7
   5.1 Key-based replacement policies: ....................................................................................................... 8
   5.2 Cost-based replacement policies: ...................................................................................................... 8
6. Cache coherency .................................................................................................................................. 8
   6.1 Cache coherence mechanisms ............................................................................................................ 9
7. Dynamic Data Caching ........................................................................................................................ 10
Conclusion .............................................................................................................................................. 11
References .............................................................................................................................................. 11
Abstract

The World Wide Web can be considered as a large distributed information system that provides access to shared data objects. As one of the most popular applications currently running on the Internet, The World Wide Web is of an exponential growth in size, which results in network congestion and server overloading. Also, the WWW has documents that are of diverse nature and so everyone can find information according to their liking. But, this scorching rate of growth has put a heavy load on the Internet communication channels. This situation is likely to continue in the foreseeable future, as more and more information services move onto web. The result of all this is increased access latency for the users. Access latency, which is the time interval between the user giving out a request and its actual completion, could result from many reasons. Servers can get flooded with more requests than they can optimally handle. The network path between the user and the server could become congested due to increased traffic on any or some of the constituent links.

Caching popular objects close to the users provides an opportunity to combat this latency by allowing users to fetch data from a nearby cache rather than from a distant server. Web caching has been recognized as one of the effective schemes to alleviate the service bottleneck and reduce the network traffic, thereby minimize the user access latency.

In this seminar I would like to give about why we need web caching, desirable properties of the web cache, cache resolution, cache replacement policies, cache coherency and about caching of dynamic data.

1. Introduction

The World Wide Web can be considered as a large distributed information system that provides access to shared data objects. As one of the most popular applications currently running on the Internet, The World Wide Web is of an exponential growth in size, which results in network congestion and server overloading. Web caching has been recognized as one of the effective schemes to alleviate the service bottleneck and reduce the network traffic, thereby minimize the user access latency.

1.1 Advantages of Caching

Documents can be cached on the clients, the proxies, and the servers. The effects of Web caching are two-fold. First, it has been shown that caching documents can improve Web performance significantly. There are several advantages of using Web caching.
1. Web caching reduces bandwidth consumption, thereby decreases network traffic and lessens network congestion.

2. Web caching reduces access latency due to two reasons:
   a) Frequently accessed documents are fetched from nearby proxy caches instead of remote data servers, the transmission delay is minimized.
   b) Because of the reduction in network traffic, those documents not cached can also be retrieved relatively faster than without caching due to less congestion along the path and less workload at the server.

3. Web caching reduces the workload of the remote Web server by disseminating data among the proxy caches over the wide area network.

4. If the remote server is not available due to the remote server's crash or network partitioning, the client can obtain a cached copy at the proxy. Thus, the robustness of the Web service is enhanced.

5. A side effect of Web caching is that it provides us a chance to analyze an organization's usage patterns.

1.2 Disadvantages of using a caching:

1. The main disadvantage is that a client might be looking at stale data due to the lack of proper proxy updating.

2. The access latency may increase in the case of a cache miss due to the extra proxy processing. Hence, cache hit rate should be maximized and the cost of a cache miss should be minimized when designing a caching system.

3. A single proxy cache is always a bottleneck. A limit has to be set for the number of clients a proxy can serve. An efficiency lower bound (i.e. the proxy system is ought to be at least as efficient as using direct contact with the remote servers) should also be enforced.

4. A single proxy is a single point of failure.

5. Using a proxy cache will reduce the hits on the original remote server which might disappoint a lot of information providers, since they cannot maintain a true log of the hits to their pages. Hence, they might decide not to allow their documents to be cacheable.

2. Proxy Caching

Proxy servers are generally used to allow access to the Internet from users within a firewall. For security reasons, companies run a special type of HTTP servers called "proxy" on their firewall machines. A Proxy server typically processes requests from within a firewall by forwarding them to the remote servers, intercepting the responses, and sending the replies back to the clients. Since the same proxy servers are typically shared by all clients inside of the firewall, naturally this leads to the question of the effectiveness of using these
proxies to cache documents. Clients within the same firewall usually belong to the same organization and likely share common interests. They would probably access the same set of documents and each client tends to browse back and forth within a short period of time. Therefore on the proxy server a previously requested and cached document would likely result in future hits. Web caching at proxy server can not only save network bandwidth but also lower access latency for the clients.

2.1 Reverse Proxy Caching:
An interesting twist to the proxy cache approach is the notion of reverse proxy caching, in which caches are deployed near the origin of the content instead of near clients. This is an attractive solution for servers that expect a high number of requests and want to ensure a high level of quality of service. Reverse proxy caching is also a useful mechanism when supporting Web hosting farms (virtual domains mapped to a single physical site), an increasingly common service for many Internet service providers (ISPs).

2.2 Transparent Caching:
Transparent proxy caching eliminates one of the big drawbacks of the proxy server approach: the requirement to configure Web browsers. Transparent caches work by intercepting HTTP requests and redirecting them to Web cache servers or cache clusters. This style of caching establishes a point at which different kinds of administrative control are possible; for example, deciding how to load balance requests across multiple caches. There are two ways to deploy transparent proxy caching: at the switch level and at the router level.

Router-based transparent proxy caching uses policy-based routing to direct requests to the appropriate cache(s). For example, requests from certain clients can be associated with a particular cache.

In switch-based transparent proxy caching, the switch acts as a dedicated load balancer. This approach is attractive because it reduces the overhead normally incurred by policy-based routing. Although it adds extra cost to the deployment, switches are generally less expensive than routers.
3. Caching architectures

3.1 Hierarchical caching architecture

One approach to coordinate caches in the same system is to set up a caching hierarchy. With hierarchical caching, caches are placed at multiple levels of the network. For the sake of simplicity, we assume that there are four levels of caches: bottom, institutional, regional, and national levels. At the bottom level of the hierarchy there are the client/browser caches. When a request is not satisfied by the client cache, the request is redirected to the institutional cache. If the document is not found at the institutional level, the request is then forwarded to the regional level cache which in turn forwards unsatisfied requests to the national level cache. If the document is not found at any cache level, the national level cache contacts directly the original server. When the document is found, either at a cache or at the original server, it travels down the hierarchy, leaving a copy at each of the intermediate caches along its path. Further requests for the same document travel up the caching hierarchy until the document is hit at some cache level.

3.2 Distributed caching architecture

In distributed Web caching systems, there are no other intermediate cache levels than the institutional caches, which serve each others' misses. In order to decide from which institutional cache to retrieve a miss document, all institutional caches keep meta-data information about the content of every other institutional cache. With distributed caching, most of the traffic flows through low network levels, which are less congested. In addition, distributed caching allows better load sharing and are more fault tolerant. Nevertheless, a large-scale deployment of distributed caching may encounter several problems such as high connection times, higher bandwidth usage, administrative issues, etc.

There are several approaches to the distributed caching. Internet Cache Protocol (ICP), which supports discovery and retrieval of documents from neighboring caches as well as parent caches. Another approach to distributed caching is the Cache Array Routing protocol (CARP), which divides the URL-space among an array of loosely coupled caches and lets each cache store only the documents whose URL are hashed to it.

4. Prefetching

Although Web performance is improved by caching documents at proxies, the benefit from this technique is limited. Previous research has shown that the maximum cache hit rate can be achieved by any caching algorithm is usually no more than 40 % to 50%. One way to further increase the cache hit rate is to anticipate future document requests and preload or prefetch these documents in a local cache. Prefetching can be applied in three ways in the Web contexts.
4.1 Between browser clients and Web servers

In this scheme, the server computes the likelihood that a particular web page will be accessed next and conveys the information to the clients. The client program then decides whether or not to prefetch the page. The prediction is done by a prediction by Prediction by Partial Match (PPM) modal. A dependency graph is constructed that depicts the pattern of accesses to different files stored at the server. The graph has a node for every file that has ever been accessed. One disadvantage of this approach is that it takes too much bandwidth which clients normally did not have.

4.2 Between proxies and Web servers

In this scheme, the Web servers push the pages to the proxies regularly. Without prefetching the performance of the proxies is limited. This scheme further reduces the latency. Geographical Push-Caching where a Web server a Web server selectively sends it documents to the caches that are closest to its clients. Here the important issues is to maintain the accurate network topology.

4.3 Between browser clients and proxies

Prefetching can also be done between browser clients and proxies. One approach is to predict which cached documents a user might reference next (based on PPM) and take the advantage of idle time between user requests to push the documents to the users.

The first two approaches run the risk of increasing wide area network traffic, while the last one only affects the traffic over the modems or the LANs. All of these approaches attempt to prefetch either documents that are considered as popular at servers or documents that are predicted to be accessed by user in the near future based on the access pattern.

5. Cache placement/replacement

The key aspect of the effectiveness of proxy caches is a document placement/replacement algorithm that can yield high hit rate. In addition to traditional replacements policies like LRU, LFU there are other type of algorithms which try to minimize various cost metrics, such as hit rate, byte hit rate, average latency, and total cost. Current cache replacement strategies can be divided into 3 types.

5.1 Traditional replacement policies and its direct extensions:
Least Recently Used (LRU): LRU evicts the object which was requested the least recently.
Least Frequently used (LFU): LFU evicts the object which is accessed least frequently.
Pitkow/Recker Strategy: This uses LRU. It evicts objects in LRU order, except if all objects are accessed within the same day, in which case the largest one is removed.
5.2 Key-based replacement policies:

**LRU-MIN**: If there are any objects in the cache which have size being at least S, LRU-MIN evicts the least recently used such object from the cache. If there are no objects with size being at least S, then LRU-MIN starts evicting objects in LRU order of size being at least S/2. That is, the object who has the largest \( \log(size) \) and is the least recently used object among all objects with the same \( \log(size) \) will be evicted first.

**LRU-Threshold**: It is the same as LRU, but objects larger than a certain threshold size are never cached.

**Lowest Latency First**: It minimizes average latency by evicting the document with the lowest download latency first.

5.3 Cost-based replacement policies:

**Greedy Dual Size (GD-Size)**: It associates a cost with each object and evicts objects with least cost/size. This is a generalization of the LRU algorithm to the case where each object has a different fetch cost. The motivation behind the GD-Size scheme is that objects with large fetch costs should stay in the cache for longer time. The algorithm maintains a value for each object that is currently stored in the cache. When an object is fetched into the cache, its value is set to its fetch cost. When a cache miss occurs, the object with the minimum value is evicted from the cache, and the values of all other objects in the cache are reduced by this minimum value. And if an object in the cache is accessed, then its value is restored to its fetch cost. A further implementation optimization is to note that it is only the relative value that matters in this algorithm. So, instead of deleting a fixed quantity from the value of each cached entry, the fixed quantity could be added to the value of the new object, and the effect would remain the same.

**Hierarchical Greedy Dual (Hierarchical GD)**: This algorithm does object placement and replacement cooperatively in a hierarchy. Cooperative placement helps to utilize a nearby idle cache and the hit rates in the cache hierarchy are increased by placement of unique objects. It is the same as GD-size with the added advantage of cooperative caching. In this scheme, when an object is evicted from one of the child clusters, it is sent to its parent cluster. The parent first checks whether it has another copy of the object among its caches. If not, it picks the minimum valued object among all its cached objects. Out of the two objects, it retains the one that was used more recently. It propagates the other object recursively to its parent.

6. Cache coherency

Caches provide lower access latency with a side effect: every cache sometimes provide users with stale pages - pages which are out of date with respect to their master copies on the Web servers where the pages originated. Every Web cache must update pages in its cache so that it can give users pages which are as fresh as possible.
6.1 Cache coherence mechanisms

Current cache coherence schemes providing two types of consistency. Strong cache consistency and weak cache consistency have been proposed and investigated for caches on the World Wide Web.

1. Strong cache consistency

   a) Client validation: This approach is also called polling-every-time. The proxy treats cached resources as potentially out-of-date on each access and sends an If-Modified-Since header with each access of the resources. This approach can lead to many 304 responses (HTTP response code for "Not Modified") by server if the resource does not actually change.

   b) Server invalidation: Upon detecting a resource change, the server sends invalidation messages to all clients that have recently accessed and potentially cached the resource. This approach requires a server to keep track of lists of clients to use for invalidating cached copies of changed resources and can become unwieldy for a server when the number of clients is large. In addition, the lists themselves can become out-of-date causing the server to send invalidation messages to clients who are no longer caching the resource.

2. Weak cache consistency

   a) Adaptive TTL: The adaptive TTL handles the problem by adjusting a document's time-to-live based on observations of its lifetime. Adaptive TIL takes advantage of the fact that file lifetime distribution tends to be bimodal; if a file has not been modified for a long time, it tends to stay unchanged. Thus, the time-to-live attribute to a document is assigned to be a percentage of the document's current "age", which is the current time minus the last modified time of the document. Studies have shown that adaptive TTL can keep the probability of stale documents within reasonable bounds (< 5%). Most proxy servers (e.g. CERN httpd) use this mechanism. However, there are several problems with this expiration-based coherence. First, user must wait for expiration checks to occur even though they are tolerant to the staleness of the requested page. Second, if a user is not satisfied with the staleness of a returned document, they have no choice but to use a Pragma:No-Cache request to load the entire document from its home site. Third, the mechanism provides no strong guarantee towards document staleness. Forth, users can not specify the degree of staleness they are willing to tolerate. Finally, when the user aborts a document load, caches often abort a document load as well.

   b) Piggyback Invalidation: There are two types of algorithms in Piggyback Invalidation one that works on the proxy and other on the server. The Piggyback Cache Validation (PCV) capitalizes on requests sent from the proxy cache to the server to improve coherency. In the simplest case, whenever a proxy cache has a reason to communicate with a server it piggybacks a list of cached, but potentially stale, resources from
that server for validation. The basic idea of the Piggyback Server Invalidation (PSI) mechanism is for servers to piggyback on a reply to a proxy, the list of resources that have changed since the last access by this proxy. The proxy invalidates cached entries on the list and can extend the lifetime of entries not on the list. To achieve better performance hybrid approach which combines the PSI and the PCV techniques can be used. The choice of the mechanism depends on the time since the proxy last requested invalidation for the volume. If the time is small, then the PSI mechanism is used, while for longer gaps the PCV mechanism is used to explicitly validate cache contents. The rationale is that for short gaps, the number of invalidations sent with PSI is relatively small, but as the time grows longer the overhead for sending invalidation will be larger than the overhead for requesting validations.

7. Dynamic Data Caching

As we mentioned before, the benefit of current Web caching schemes is limited by the fact that only a fraction of web data is cacheable. Non-cacheable data (i.e. personalized data, authenticated data, server dynamically generated data, etc.) is of a significant percentage of the total data. For example, measurement results show that 30% of user requests contain cookies. Many web sites disable caching of documents. The main reason behind that is many sites contain banner adds and they collect money from them based on the no 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 the user information that who is using the content. For these reasons all documents can’t be cached. Current 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) or a collection of URLs. The code is typically written in a platform-independent programming language such as Java. When caching a document, the proxy also fetches the corresponding cache applet. When a user request 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 and other information as arguments. The cache applet then decides what the proxy will send back to the user, either giving the proxy a new document to send back to the user, or allowing the proxy to use the cached copy, or instructing the proxy to send the request to the Web server. Furthermore, the applet can deposit information in a log object, which is sent back to the server periodically and when the applet or the document is purged from the cache. It's shown that Active cache scheme can result in significant network bandwidth saving at the expense of CPU cost. However, due to the significant CPU overhead, the user access latency is much larger than that without caching dynamic objects.
Cache applets allow servers to obtain the benefit of proxy caching without losing the capability to track user accesses and tailor the content presentation. They can perform a variety of functions, for example, logging user accesses, rotating advertising banners, checking access permissions, constructing client-specific Web pages, etc.

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 cached data. The API allows the accelerator to cache dynamic as well as static data. Invalidating and updating cached data is facilitated by the Data Update Propagation (DUP) algorithm which maintains data dependence information between cached data and underlying data in a graph.

Conclusion

As Web services becomes more and more popular, users will suffer from network congestion and server overloading. Web caching is recognized to be one of the effective techniques to alleviate server bottleneck and reduce network traffic, thereby minimize the user access latency. The main challenges in Web caching are proxy placement, dynamic data caching, cache routing. The research frontier in Web performance improvement lies in developing efficient, scalable, robust, adaptive and stable Web caching scheme that can be easily deployed in current and future network.
References


2. P. Cao, J. Zhang, and K. Beach, "Active Cache: Caching dynamic contents on the web"


