Push-Pull Caching

Vimal Kumar, Manoj Patel, Sandesh Meda & Praveen Mahadevanna
Department of Computer Science,
University of Texas at Arlington
Arlington, TX 76019

Abstract

The explosive growth in Internet traffic has made it critical to look for ways to accommodate the increasing number of users while preventing excessive delays, congestion, and widespread blackouts. For the past several years web browsing and most of the web related traffic have generated a large amount of Internet traffic [1]. Insufficient bandwidth causing high latency is one of the problems faced by the web users [2]. Several techniques have been developed to meet the requirements for a faster and efficient Internet. Push Pull caching is one of them.

1 Introduction

User requests on the Web are often served by a single machine, which can easily become a bottleneck. Even worse, that server represents a single point of failure—if it is down, access to the information is lost.

To preserve the usability of the World Wide Web, the following issues need to be addressed at the server level:

1. Document retrieval latency times must be decreased.
2. Document availability must be increased, by distributing resources among several servers.
3. The amount of duplicate data transfer must be reduced—certainly an important issue for anyone paying for network usage.
4. Network access must be redistributed to avoid peak hours.

A popular and widely accepted approach to address at least some of these problems is the use of caching proxies. Others can be solved by an approach that combines caching and replication.

2 Traditional Caching techniques

The first approach to caching on the Web was the establishment of a client local disk and/or in-memory cache in Web browsers. It was soon realized that the notion of memory hierarchy could be extended to consider Web servers as another external level of memory. Because the effectiveness of caching is dependent on the number of times the same document is requested, it was clear that the gains realized would noticeably increase if the cache were shared among several users.

These facts led to the development of a second approach of caching now in use on the Web; that of caching proxies. A proxy acts as a middleware between the users' machine and the outside world. From the users' point of view, the proxy acts like a Web server: each request is sent to and answered by the proxy. From the servers' point of view the proxy acts like a client: it forwards requests to the original web server. The proxy is therefore an ideal place to a second level of cache. It is shared among several users, so there is an increased probability of data being accessed more than once. It also confers the significant benefit of being able to act as a firewall between the local machines and the outside world [2].

3 Push/Pull Caching

It is necessary to differentiate between Push and pull caching before we go any further. Pull caching is client initiated whereas the Push caching is Web Server Initiated.

The push caching has rapidly gained considerable popularity since its emergence in April 1996; at the time PointCast announced its '*PointCast Network'*, which soon became extremely popular. The push caching was created to alleviate problems facing users of the Internet, e.g. information overload and low bandwidth.

As shown in figure 1, in the push-based data delivery, the server tracks all proxies that have requested objects. If a web page has been modified, it notifies each proxy and when the client requests for the file, it is served from the proxy's cache instead of the request going directly to the server. This improves network utilization.

There are 2 kinds of notifications between the server and the proxy [3]

1. Indicate that the object has been changed (Invalidate)
2. Send new version of the object (Update)

The kind of notification that has to be selected by the Server depends on the rate at which the modifications occur. The Update notification can be used for frequently changing objects whereas invalidation can be used for rest.

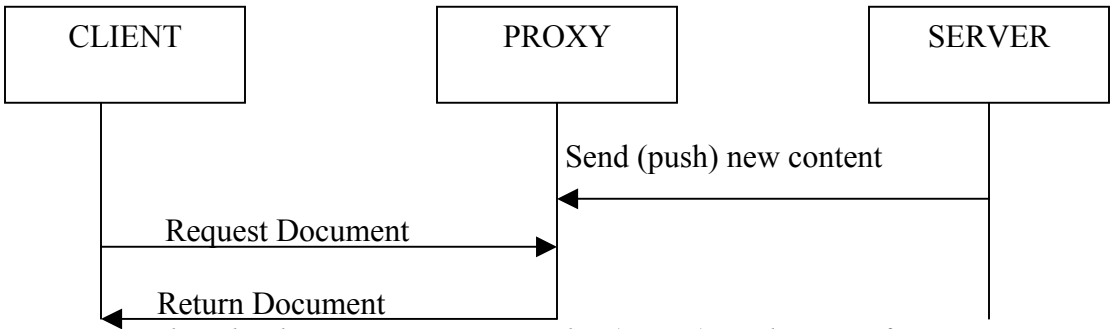

Figure 1: Push technology - A content provider (server) sends new information to proxy and client is notified about it.

Push based systems are used in applications where information feeds as stock quotes and sport tickets, electronic newsletters, mailing lists, and weather information systems, cable TV on the Internet [4]

Pull-based data delivery is also known as demand data delivery. As shown in Figure 2, a proxy explicitly requests data items from the server [4] when the client requests for a document. The proxy instead of the request going directly to the server form the client can service subsequent requests for the same file by clients. In the pull-based approach, the proxy is entirely responsible for maintaining consistency. The proxy maintains data constancy by setting a TTL on the document cached and this copy is served until the TTL expires [3].

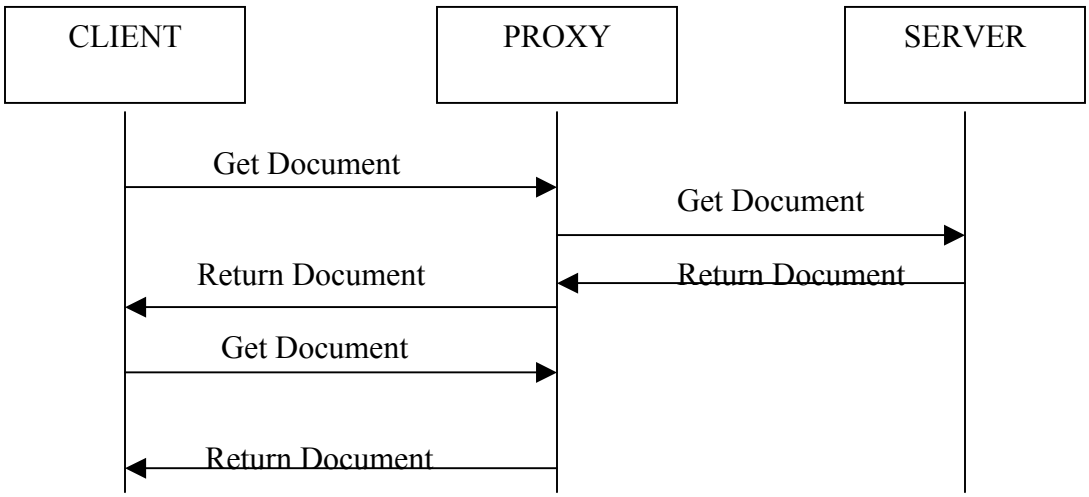

Figure 2: Pull Technology [1]

3.1 Advantages of the Push approach

1. The push technology can reduce the burden of acquiring data for tasks in which there is a large information flow. Push technologies improve efficiency by downloading information to the users' system in a scheduled fashion so it can be rapidly viewed, and thereby eliminating the risk of the user being served stale data.

2. Pushing alerts to the user (e.g. in the form of e-mail or the change itself), improves the efficiency of Web-based time-sensitive information distribution (such as stock quotes or trouble tickets in a technical support system).

3. Automatic downloading of software upgrades and fixes is a way to deliver software faster, and at the same time, reduce costs.

4. The push technology enables intelligent information filtering based on personalized user profiles describing required information needs.

3.2 Push Vs Pull

In the push approach, the server repetitively transmits (multicasts) the data to the proxy services. In such a system, data items are periodically sent from the server to the proxy service without requiring a specific request from the clients. In contrast, the pull-based approach explicitly requests data items by sending messages to the web server [6]. Pull based access has the advantage of allowing proxy service to play a more active role in obtaining the data, rather than relying solely on push enabled web servers.

4 Prototype Design

The objective of our project is to develop a middleware that resides between conventional browsers and caching enabled servers. In order to achieve this, we have simulated the Internet. The browser was build to receive all kinds of files, including htm, and non-htm, from the servers. We introduce the proxy service that resides between client and the web servers. The assumption here is that the proxy service, residing within the LAN, is serving large number of local clients and all the traffic for clients pass through this service. The unique feature of our proxy service is that it transforms into a web server when it caches data in its local web space, hence, reducing traffic in WAN and greatly improves QoS on the Internet. Thus, our design of this prototype revolves largely around reducing network traffic while maintaining data consistency.

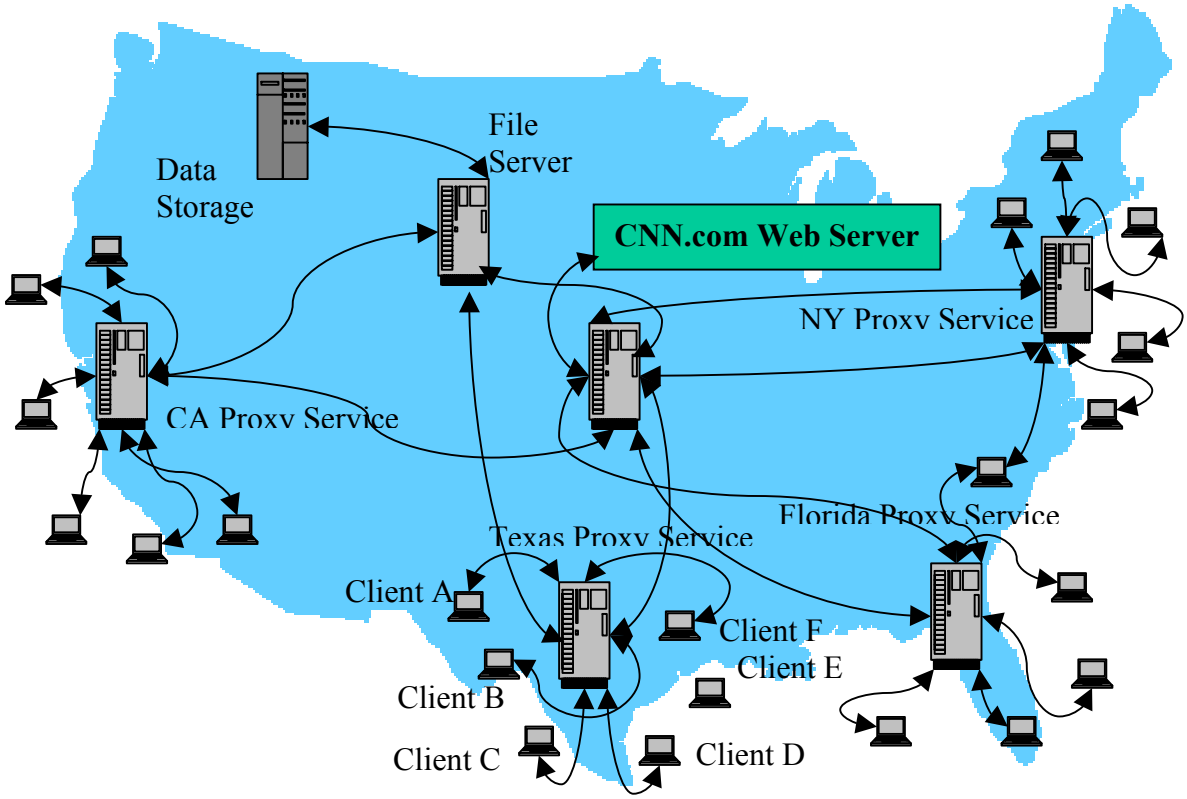

Figure 3:Push Pull Model

4.1 Architectural Overview

This architecture is based on push and pull model of caching, which allows server to store data in proxy services and allow clients and proxy services to request for data from the web servers. Figure 3 gives an overview of push and pull model. There are three basic components to our prototype design: Client, Proxy Service, and Web Server. Figure 3 depicts many clients accessing www.cnn.com from CNN's web server through different state proxy services. Clients in the figure simulate individual users in the LAN. Proxy Services provides services to the requests made by LAN Clients. Web Servers are the servers where the file being requested resides, for ex. www.cnn.com resides at CNN Web Server; this server can also be an ftp web server with large data storage. As illustrated, no clients directly access the file from web servers, instead they are routed through proxy services, which can provide them with the cache copy of the requested page or a fresh copy of the page from the web servers.

4.2 Components of the Prototype

Figure 4 describes components of our pull-push prototype model. These are divided into three levels. Following is a detail description of each of the levels.

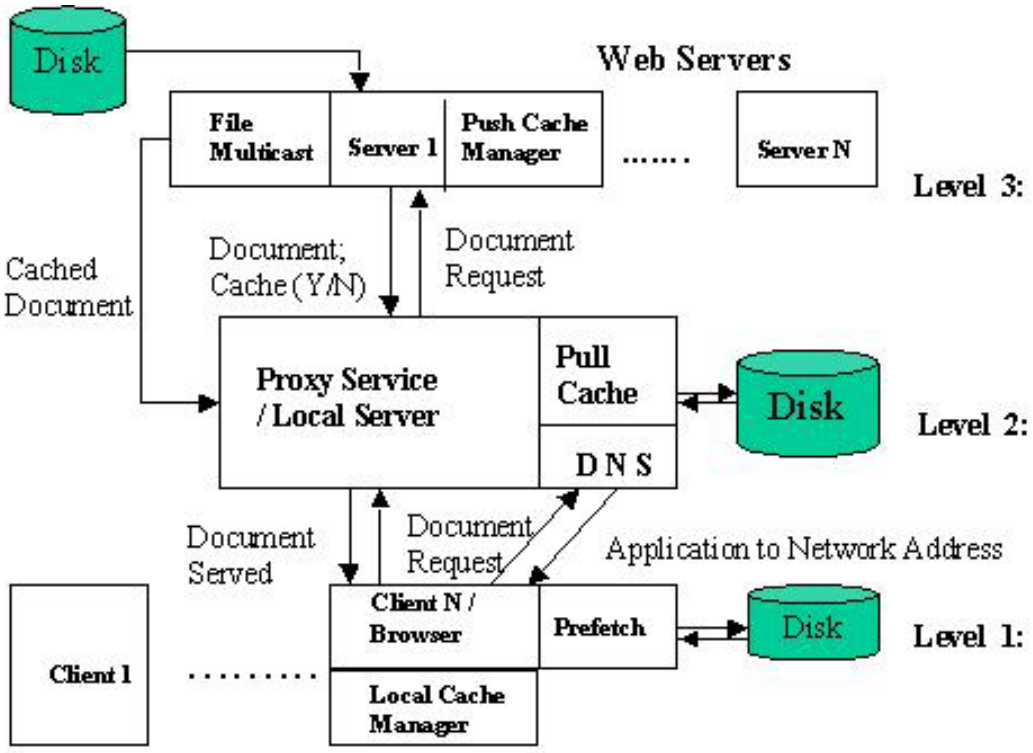

Figure 4: Component Model of Pull-push prototype

Level 1:

This level is where the local user is surfing the web or accessing database from the network. There can be 1 to n number of such clients accessing files from the network. There is also a fixed storage space allocated to individual clients for caching. In our implementation, we have made this dynamically configurable. Clients use pull-based model to cache its data from the web servers. We have implemented cache manager at this level for caching and removing unused cached files from the storage. Least Recently Used (LRU) algorithm is used to maintain cache size and remove last used files from the storage space. In addition to cache manager, we have prefetch algorithm that allows Clients to cache files in advance based on user access record.

Level 2:

This level acts as a middleware between clients and web servers. There can be 1 to n number of such services serving its LAN clients. Proxy service in level 2 processes requests made by clients at level 1. Proxy service also act as a client to the level 3 Web Servers, which is discussed shortly. Proxy services also incorporate DNS, which provide clients with IP address for the requested server address. As in client cache, there is also a fixed storage space allocated for caching at this level. Unlike level 1 cache, which is pull-based, cache in proxy services can be either a push or pull-based. Cache model is push-based if the web server initiates the cache at this level. Pull-based model is used if the cache is initiated and maintained by the proxy service. To maintain cache consistency, cache size, and remove the least used files from the storage space, proxy service uses pull cache manager. We implemented Least Frequently Used (LFU) algorithm at this level for removal of files that are accessed the least number of times. In summary, at this level the page requests from the clients are handled and the requested file(s) are sent to the clients. The file send to the client can be either cache copy or file from the web server depending on the availability of the page requested and cache validity if the page/document was available in the cache.

Level 3:

Web Server at this level is the main site where the pages requested by the client reside. There can also be 1 to n number of web servers each with a number of pages/files. In the World Wide Web, the clients directly access this web server. In our prototype, the proxy services requests the page on behalf of the clients. Hence, proxy service in level 2 becomes client which requests pages from the web server. Web server uses the push-based model for caching. As its name implies, push-caching is the pushing of the requested page to the proxy service in our implementation. Web Server makes the decision on when to push-cache and where to push-cache based on the history and frequency of the page requested. After initial push-caching, the cache copy at individual proxy services are maintained by periodically sending the updated copy of the page to all the services that have been push-cached. Multicasting is the technique used for updating the cache from the web server to reduce bandwidth requirements. Multicasting is to send updated page to a group of proxy services that web server originally pushed its cached copy to. This requires web server to keep track of where it did push-cached and maintain the table for multicasting. Multicasting has less load on the bandwidth but more computation in part of the web server.

In the following section, we give example for pull-caching and push-caching which shows the role of each level in page request.

5 Operational Procedures

We have the following class files that simulate our pull-push cache strategy:

Browse.java : simulates client interaction at level 1§.
WebService.java : simulates proxy service at level 2§.
Server.java : simulates Web Server at level 3§.

§ Levels here corresponds to the levels in figure 4.

Start up procedure:

1. Web Server – "Java Server 90" will start Web Server at port 90.
2. Proxy Services – "Java WebServer" will start proxy service at default port 80.
3. Client – "Java Browse" will start the browser that will allow the user to request for pages/files.

Values for proxy service cache size, client cache size, and network bandwidth for the simulation are entered in the configuration GUI from the menu.

5.1 Pull-based cache example

Once start up procedure is complete, Web Server continuously listens to requests made by proxy services at port 90. Mean while, there are two threads running in the proxy service that listens for client requests. One of the threads is to handle DNS requests and one if for page requests. Following steps describes the flow of our program when client initiates the request for a page/file.

1. Client requests IP address for the web domain from DNS.
2. DNS receives the request from the client and returns the corresponding IP address for the requested domain name from its lookup table.
3. Client sends IP address and the requested file name to the proxy service.
4. The following scenario is handled by the proxy service:
    - If the file is not frequently used then, it will return the page from the Web Server to the client without caching it.
    - If the file is requested frequently, proxy service will cache the file to its storage space and return the cache copy to the client.
    - If the file is already in the cache and is valid then it will return the cache copy to the client.
    - If the file is already cache and not valid then it will update the cache by requesting updated page from the Web Server and then return the cache copy to the Client.

5. Client receives the page/file from the proxy service and caches the page every time to its storage space. Client keeps the record of all the files that user requested and prefetches the page [17] based on the previous request behavior.¶

¶ Due to limitation on the cache size, LFU and LRU algorithms are used to remove least frequently accessed cache files from the storage space in the Web Services and the Client respectively.

5.2 Push-based cache example

Once start up procedure is complete, Web Server continuously listens to requests made by proxy services at port 90. Mean while, there are two threads running in the proxy service that listens for client requests. One of the threads is to handle DNS requests and one if for page requests. Following steps describes the flow of our program when client initiates the request for a page/file.

1. Client requests IP address for the web domain from DNS.
2. DNS receives the request from the client and returns the corresponding IP address for the requested domain name from its lookup table.
3. Client sends IP address and the requested file name to the proxy service.
4. Proxy service send page request to the Web Server.¶
5. Web server receives the request and keeps the record of the request in the two lists that are file specific. One list is to keep tract of how many times that page/file is requested and by whom it was requested. Second list is for the push-caching when the hit counter goes above threshold value for the file. So,
    - If the hit count for the page requested is below its threshold value then it will send the page without doing push caching.
    - If the hit count goes above the threshold value, web server will add the file to the push list and sends first two bytes of the response to the proxy service that will instruct proxy service to cache the file to its storage space. Proxy service will send response back saying that it has cached the file to its storage space. Next time the client requests the cached page/file, proxy service will return the cached page.¶ After the file has been pushed to the proxy services, it will multicast periodically to the proxy services to update the cached copy. Web Server will send a delete cache request to the proxy service to delete its cached copy if that cache is not accessed.
6. Client receives the page requested from the proxy service which might be cache copy or non-cached copy.

¶ Due to limitation on the cache size, LFU and LRU algorithms are used to remove least frequently accessed cache files from the storage space in the Web Services and the Client respectively.

6 Features:

The final utility of the caching technique should be to impose fewer overheads on the network and take fewer resources, while at the same time making more data available to the users. Thus the caching system has many desirable features, which include transparency, scalability, efficiency, stability, and simplicity. A few of these features is displayed in our simulation model and we discuss them here:

Dynamic Adaptability: The nature of the Internet is dynamic and it is desirable to adapt to these changes. For efficient utilization of network resources only those contents that have been frequently accessed should be cached. This is essential to achieve optimal performance. With regard to the dynamic adaptability, caching is better adapted than replication because replication copies whole server data or parts of server data. However, most of the contents replicated are not frequently accessed: this leads to delays and wastes network resources.

Transparency: The web browser need not have knowledge about the existence of the proxy service and this is how proxies work. The only effect that is perceived by the user should be a faster access to data and higher data availability. This is a stumbling block for replication servers because they need explicit redirection mechanism to locate nearby servers. In fact, it has been shown many times that the replicated server chosen is farther away than the original server.

Document retrieval latency: Access latency is an important quality of techniques that are designed to improve document retrieval latencies. Research has shown that there is a definite pattern of browsing; latency is reduced by fetching frequently accessed documents from nearby proxy caches instead of the web server where the data resides. Once these documents are cached, the latency for clients being served from that proxy server drops dramatically.

Network Utilization: Cached copies that are served from the proxy service need not be downloaded from the web server while the copy is still valid. There are several ways in which such cache validation can be made without affecting retrieval latency. This not only improves document latency but also improves scalability for the web server, which has to handle fewer requests. In contrast, replication often times leads to wastage of network resources due to poor location of these replication servers. The inability of such servers to quickly adapt to the changing geographic access pattern of the Internet will result in longer delays and inefficient usage of network resources in replicating data on remote servers.

7 Measured Characteristics

A simulation project of this nature is impossible to manipulate to reflect the actual browsing behavior on the Internet. Such an interpretation could be dangerously misleading and imprecise in its nature. The pull-push caching scheme that we have implemented resembles real-world implementation closely. The actual difference arises from the fact that we have a single level of cache mediation, whereas in the real-world we might have a hierarchy of proxy services. A lot of data have been collected at such proxy servers to observe how caching improves latency and help reduce the network utilization. A study that was carried out appeared in a technical report published by USENIX [18].

The data was collected from proxy caches located all across the world and for a twenty-one days of access traces from each proxy, totaling 23,700 clients and 47.4 million accesses. Some of the proxies that were used are listed:

1. The University of Hannover, Germany (HAN)
2. The Nation Wide Caching Project of Korea (KOR)
3. Digital Equipment Corporation (DEC)
4. Grant MacEwan Community College, Alberta, Canada (GMCC)
5. A major American University that has chosen to remain anonymous (AU)
6. Utrecht University, the Netherlands (UU)
7. The National Laboratory for Applied Network Research (NLANR)

The most relevant data collected is summarized here in the form of graphs:

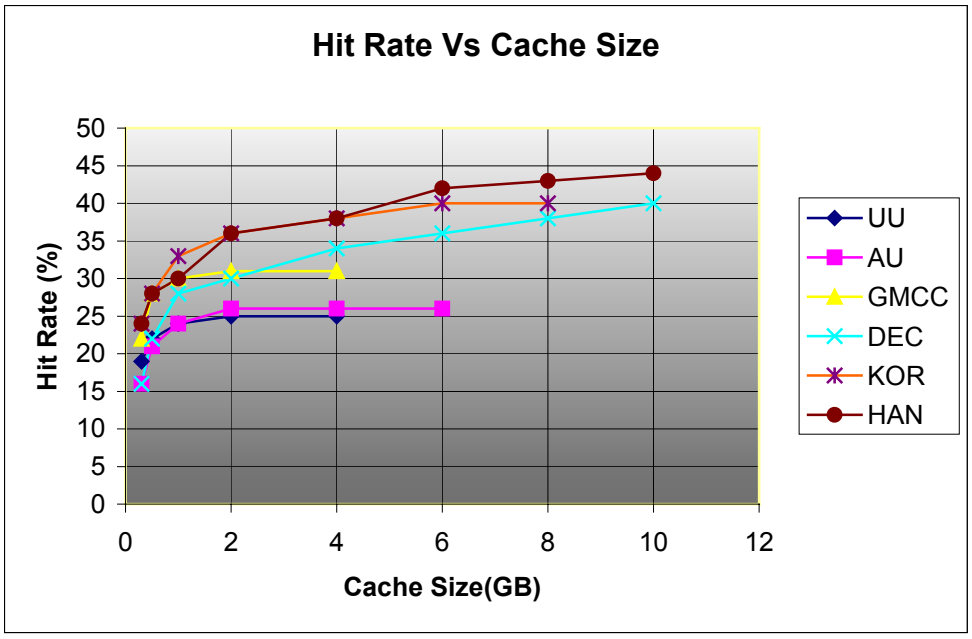

Figure 5: Cache hit rate for each trace as a function of cache size

Here was can observe that approximately 10 GB of disk space is usually enough to give as fill coverage of the cache hits that can be expected. Depending on the proxy service profiles and the number of clients that each of these services were able to serve it was observed that on average 30-40% of the requests result in a cache hit. This is a general performance efficiency that has been observed with caching proxies.

The other relevant study that was conducted was the efficiency of caching itself. A study was conducted to determine how many times the cache consistency protocols [19] were able to effectively deliver valid data. The following trend was observed:

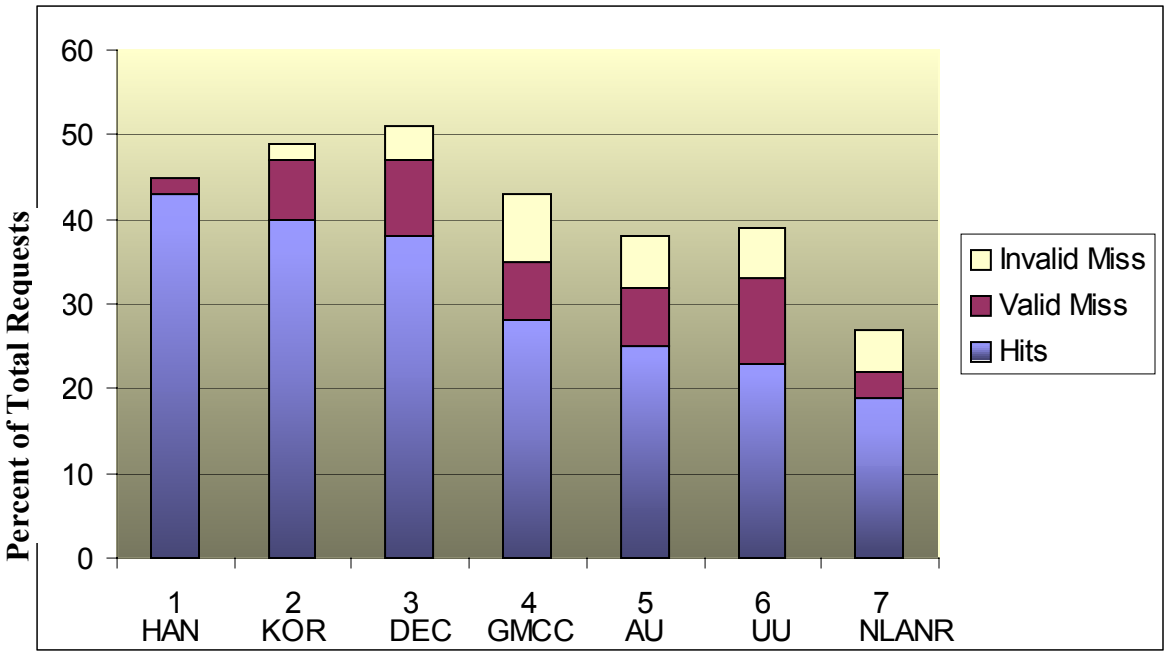

Figure 6: Portion of requests resulting in consistency misses

The height of the 'valid miss' bar is an indication of the efficiency of cache coherence protocols. The invalid miss bar indicates that the document had to be retrieved again due to TTL expiry or due to a *if-modified-since* reference made by the coherence protocol. In conclusion we can see that valid misses account for only 2% and 7% of all references.

8 Conclusion

Over the past decade the number of static web pages has grown exponentially and is estimated to be more than 7 billion static pages [16]. Having fully exploited the hardware resources that are available to us, it has become crucial to organize the World Wide Web to achieve the best performance and minimize resource utilization. One effective strategy that we can employ as is the provisioning of web resources through caching. By organizing and efficiently managing these resources, we can minimize document retrieval latencies and maximize network resources. In this paper we gave an overview of such an implementation, which shows that web caching is indeed an effective technique to improve the overall situation. However, there are improvements that can be made such as group caching [15], transparent proxies, improved data consistency, security etc. The future of the Internet will be shaped by the development and deployment of stable caching systems that can effectively co-ordinate and serve users so as to make browsing an informative and enjoyable experience for the future generation.

References

1. Towards an Accessible Web by Applying PUSH Technology - Tuula Käpylä, Isto Niemi, and Aarno Lehtola

2. Adaptive Push-Pull of Dynamic Web Data - *Pavan Deolasee, Amol Katkar, Ankur Panchbudhe, Krithi Ramamritham and Prashant Shenoy*
3. Distributed Commit and Failure Recovery - *Prashant Shenoy,* University of Massachusetts
4. Data Dissemination by Broadcast – *Evaggelia Pitoura* Computer Science Department, University of Ioannina, Ioannina, Greece
5. www.cse.iitb.ernet.in:8000/proxy/chandra/~gracias/webcaching/html/node4.html
6. Balancing Push and PULL for Data Broadcast – *Swarup Acharya, Michael Franklin, Stanley Zdonik.*
7. The Case for Geographical Push-Caching- James S. Gwertzman , Margo Selzer.
8. Reference site for push-pull caching- www.eecs.harvard.edu/~vino/web/push.cache/node1.html
9. Silicon Press http://www.silicon-press.com/briefs/brief.webcaching/brief.pdf
10. A Workgroup Model for Smart Pushing and Pulling – Gail Kaiser, Christopher Vaill and Stephen Dossick
11. Hinted Caching in the web – Jeffrey C. Mogul
12. Using Predictive Prefetching to Improve World Wide Web Latency – Venkata N. Padmanabhan, Jeffrey C. Mogul
13. Enhancing the web's infrastructure : From Caching to Replication – IEEE Internet Computing.
14. Web Caching Tutorial at http://www.web-caching.com/mnot_tutorial/how.html
15. Jia Wang, "A Survey of Web Caching Schemes for The Internet," ACM Computer Communication Review, 36-46.
16. An article in www.cnn.com, http://www.cnn.com/2000/TECH/computing/07/26/deepweb.ap
17. *Venkata N. Padmanabhan and Jeffrey C. Mogul*. "Using predictive prefetching to improve world wide web latency." ACM SIGCOMM Computer Communication Review, July 1996
18. *Bradley M. Duska, David Marwood, and Michael J. Feeley,* "The Measured Access Characteristics of World-Wide-Web Client Proxy Caches," Department of Computer Science University of British Columbia Proceedings of the USENIX Symposium on Internet Technologies and Systems", December 1997. Technical Report TR-97-16.