A Review of Web Caching Techniques and Caching Algorithms for Effective and Improved Caching

Pranay Nanda
(Post Graduate Student)
Amity School of Engineering and Technology
Amity University, Rajasthan

Shamsher Singh
(Astt Prof)
Department of Computer Science
AB College, Pathankot, Punjab, India

G.L. Saini
(Astt Prof)
Amity School of Engineering and Technology
Amity University, Rajasthan

ABSTRACT
Internet today has become a victim of its own success. As the internet is reaching a global community, the World Wide Web is becoming a global-scale data dissemination system. There has been an increase in user latency, bandwidth utilization and server loads because of the increased number of World Wide Web users. Web caching is a technology for overcoming such performance bottlenecks by storing copies of popular web objects closer to users instead of deliberately accessing them from origin servers. Our study aims to review few caching architectures. These architectures include proxy caching, cooperative caching, adaptive caching, push caching and active caching. Furthermore, as it has been repeatedly observed, same data is transmitted over same network links time and again to thousands of users. Such redundancies desire the need for caching algorithms that optimally utilize the finite cache space. Chapter 1 discusses the introduction to the study and requirement of such solutions as we further proceed to discuss those solutions in Chapter 2. Chapter 3 discusses about metrics and factors that influence caching performance and Chapter 4 discusses algorithms that are used for caching

General Terms
Web Caching Techniques

Keywords
Web Caching, Caching, Proxy Caching, Reverse Proxy Caching.

1. INTRODUCTION
A web cache (or HTTP cache) is an information technology for the transient storage space (caching) of web information, such as for instance HTML pages and graphics, to minimize bandwidth consumption, hosting server load, as well as identified lag. As website traffic on the internet increases, users are faced with ever-increasing hold-ups as well as downfalls in data delivery. Web caching is one of the key campaigns which has been investigated to further improve overall performance.A significant problem in several caching techniques is exactly how to determine what is cached where at any offered time period While Web usage has increased exponentially, the available network infrastructure has not. As a result, the network cannot keep us with user demands, and performance suffers. Factors leading to the increase in Web traffic and subsequent low performance include the increasingly multimedia nature of Web content;

- The widespread use of push technology;
- A mass migration of traditional services to Web-based applications (for example, online banking and investing);
- Millions of new users due to the high competition between Internet service providers and subsequent low-cost Internet access.

Researchers are exploring improvements to Internet performance from several angles:

- Infrastructure, Experimental gigabit networks are currently under construction.
- Cable modem and asymmetrical digital subscriber loop (ADSL) technologies will enable multimegabit access from homes.
- Protocols
- Compression
- Caching

Of the four approaches, caching promises the greatest performance gain [1] and can be implemented with current technologies. It is also the only approach that addresses the physical distances between users and Web objects. Most of the popular Web browsers implement client caches. Accessed objects are copied on the user's hard disk, circumventing the need for an Internet connection the next time the object is access. By the same token, a server can cache high-demand objects on local disks, thereby reducing the need to transfer the objects from their home locations, which may be on the other network drives

2. WEB CACHING TECHNIQUES
A cache is a storage area that is closer to the entity requiring it than the authentic source. Accessing this cache is customarily faster than being able to access the information from its original source. A cache is typically stored in memory or on disk. A memory cache is ordinarily more rapidly to read from than a disk cache, but a memory cache typically does not survive system restarts.

2.1 Proxy Caching
In proxy caching, the cache server receives the request for an object from a client [6]. If the object is present in its cache, it responds with the object. Else, it requests the source of object and ensures the client has the requested item. If required, the server may also deposit the object in its cache so as to reduce the network congestion next time the object is requested [2]. Caching server is placed close to client (at network gateway) to lower the latency and hops. The advantages of Proxy caching involve reduced latency and network traffic that makes experience of the web better and higher availability of the websites [5]. However, the disadvantages that lie with this approach are that cache is single point of failure, the browsers have to be configured and no such system exists that can dynamically add more caches when required [2] [3].

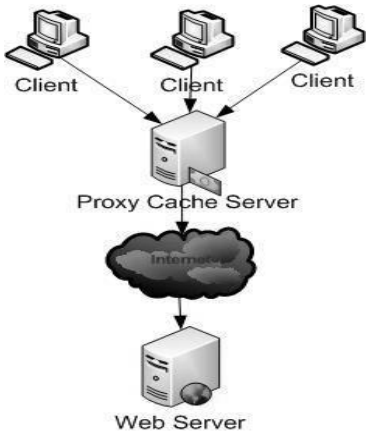

Figure 1. Proxy Caching

2.1.1 Reverse Proxy Caching

Instead of placing cache server close to client, the cache server is placed close to the servers in such a version of Proxy Caching [7]. When a server is expected to receive sumptuous requests simultaneously, in such a scenario, reverse proxy caching is an effective solution as it pretends as the origin server for the requests being generated. This is advantageous as it keeps and maintains uptime of the server substantially high and assures high quality of service (QoS) [2] [3]. It is a useful solution in scenarios where virtual domains have been mapped to a single physical site [12]. Alongside forward proxy caching, Traffic Server deals with web data requests to beginning servers on account of the visitors asking for the information. Reverse proxy caching (also known as server acceleration) is distinctive simply because Traffic Server acts as a proxy memory cache on the part of the fundamental cause servers that preserve the information. Traffic Server is designed to behave outwardly as source server which the client is attempting for connecting to. In a typical scenario the promoted hostname of the origin server eliminates to Traffic Server, which serves client requests immediately, taking information from the true origin server when necessary.

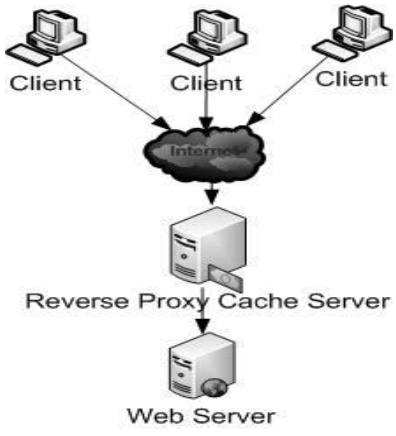

Figure 2. Reverse Proxy Caching

2.1.2 Transparent Caching

The problem with proxy server approach is that it requires configuration of the web browser [2] [3]. Transparent web caching on the other hand intercepts HTTP requests at the gateway without being visible and redirects them to web cache or clusters [8]. There are two ways to deploy transparent caching [5]: switch level and router level. Router level transparent caching uses policy based routing to direct requests to appropriate cache or server. Switch level transparent caching offers switch to act as a dedicated load balancer. It is more preferable approach because switches generally cost cheaper than routers [12]. As video streaming and rich media downloads keep up to flood operator networks, with no end in perceive, network operators are examining as well as deploying transparent Internet caching inside of their networks to address a much wider selection of Internet content. The intent is two-fold. The first is to alleviate the network infrastructure and data transfer usage costs associated with over the top (OTT) content and the second is to distinguish their consumer broadband servicing as well as deliver better user performance. By getting rid of any potential slows down associated with the Internet or even the content origin, caching enables the operator to emphasize their investment being made in the access network and deliver more content material at top speeds.

2.2 Adaptive Web Caching

Adaptive Web caching involves replacement algorithms that analyze requests generated by the client and deposits the most accessed objects in the cache deposit [13]. Adaptive caching consists of multiple, distributed caches which dynamically join and leave cache groups based on content demand [11]. The general idea of Adaptive Caching imagines a tight mesh of self-organizing, overlapping multicast groups of servers that adapt when necessary to changing conditions [2]. This mesh forms an implicit and scalable hierarchy that is used to efficiently diffuse popular web content towards the demand [14]. The two main components are underlying communication paths between the neighboring caches and flow of requests for data along paths [12].

2.3 Push Caching

Servers decide when and where the objects are cached, this idea was inducted by push caching [12]. Servers realize which clients require that data often and place it close to them. Data is dynamically mirrored [15]. An assumption about push caching is that the ability to launch caches traverses administrative boundaries [2]. Push caching is an effective solution for content providers [4]. Push-caching has subsequently diminished the actual quantity of network traffic without considerably impacting almost every primary server's load. If only primary servers could store duplicated documents then push-caching would be of questionable value. The virtue of push-caching, however, is that it is very easy to add additional servers. Proxy-servers, for example, are ideal candidates for accepting duplicated things simply because they're definitely running web caching software, continuously are affixed to large disks, and are usually not hidden behind firewalls. Push-caching can circulate the stress from overloaded primary servers onto proxy-servers and other servers without distinguished an unacceptible load because all servers caching replicated objects may refuse alternative objects at any time.

2.4 Active Caching

The scheme allows servers to supply cache applets to be attached with documents, and requires proxies to invoke cache applets upon cache hits to furnish the necessary processing without contacting the server [10]. Cache applets allow servers to obtain the benefit of proxy caching without losing the capability to track user accesses and tailor the content presentation dynamically [2] [12].

2.5 Cooperative Caching-Swalla Architecture

Most traditional **cooperative caching** schemes were developed for network file systems but were not designed **for cluster-based web servers** with content-based requirements/ An approach to improve web performance is in recognizing that the bottleneck access of website is process utilize.tion rather than network bandwidth. Swalla is a distributed and multi-threaded web server which runs on a cluster of workstations and shares cache information and cache data between nodes [4].

Swalla is solution to web sites that make extensive requests for dynamic content such as CGI requests. It cooperatively caches the results of CGI requests [14]. The server stores the meta-data for cached content in the cache directory. Each note communicates with each other to exchange cached data and meta-data [5]. Each Swalla node consists of a HTTP module and a cache module. Other caching architectures are: Internet Cache Protocol (ICP), Summary Cache and Cache Digest. ICP works similar to proxy caching, is an application layer protocol running on top of UDP and us used to coordinate proxy web caches. Summary Cache and Cache Digest use Bloom Filter to represent the directory of cache content [5]. Bloom Filter is a memory efficient data structure that is used to test whether an element is member of set or not [9]. The difference between summary cache and cache digest is that summary cache extends ICP to update the directory whereas Cache Digest uses HTTP to transfer directory information [2].

2.6 Performance evaluation of Caching Architectures

Although there are many web caching techniques, no technique is such omnipotent that performs well in all scenarios [14]. Every technique has a different architecturethat justifies optimal use ofdifferentavailableresources based on its design. An effective caching mechanism is the groundwork of any distributed-computing architecture. The focus of this article is to understand the importance of caching in designing effective and efficient distributed architecture.

Table 1:Web Caching Techniques:

Web Caching Technique	Methodology	Pros	Cons	Suitable Environment
Proxy Caching	Place proxy servers close to clients	Reduced latency and network traffic, bandwidth savings, increased availability	Single Point of failure, has to be explicitly configured, no dynamic method to add more caches	Clients generating high amount of requests
Reverse proxy caching	Place proxy server close to servers	Ensures high quality of service(QoS)	Single point of failure and filtering against malicious attacks	Content Providers
Transparent Proxy Caching	Intercept HTTP requests at gateway and redirect them to cache clusters	Does not have to be explicitly configured	Violates end-to-end statement by not maintaining constant connection to end point of connection	Places where administrative controls over caching is possible or required
Adaptive Web Caching	Learning by example to adapt to requests for objects based on their demand	Adapts to each client individually, self-organized	Complex implementation, initially requires training	Sites that generate dynamic content
Push Caching	Cached data is placed close to clients that request them frequently	Servers curate caches for clients	Ability to launch caches may cross administrative boundaries	Content providers
Active Caching	Cache applets are used to customize objects that otherwise would not be cached	Use of cache applets to perform personalized caching locally instead at originating servers	Need of coding cache applets for objects	Sites that serve dynamic and personalized content
Swalla Architecture	Distributed and multi-threaded architecture of server nodes that share cache and cache information among each other	Effective for dynamic content requests such as CGI	Complex architecture, increased administrative architecture at distributed caches	Sites that generate great dynamic content

3. PERFORMANCE METRICS AND FACTORS

Various metrics and factors affect the decision to select apt caching policy for an environment. To deliver maximum efficiency, it is helpful to assess performance of various algorithms based on the factors and metrics relevant to the environment.

3.1 Performance metrics

3.1.1 Hit ratio

Hit ratio is generally the ratio of objects obtained through caching policy versus the number of requests made [2]. Higher hit ratio indicates better caching policy. However, it may only be relevant if the objects are homogenous in size.

3.1.2 Byte hit ratio

Byte hit ratio is the ratio of bytes accessed from the cache to the total bytes accessed [2]. In case of objects being of heterogeneous sizes, Byte hit ratio is better metric for measurement.

3.1.3 Bandwidth utilization

It is an important count where an algorithm that reduces consumption of bandwidth is better [2].

3.1.4 User response time

It is the amount of time a user waits for the system to retrieve the requested object [2]. It is also known as latency.

3.1.5 Cache server CPU and I/O utilization

The fraction of total available CPU cycles or disk and object retrieval latency [2].Latency is inversely proportional to object hit ratio because a cache hit can be catered more swiftly than a request to origin server. However, optimizing one metric does not imply that it will optimize other too. For example, increasing hit rate does not essentially minimize latency [2].

3.2 Performance factors

3.2.1 User access patterns

If a user accesses objects small in size more frequently, then the objects that are small in size stand obvious candidates for caching. User access patterns are not static and therefore a good caching algorithm should not be static either [2] [11]. Cache replacement algorithms decide what objects to be discarded when the cache is full.

3.2.2 Cache removal period

Cache removal period dictates that documents will be removed when there is no space in the cache [14]. Continuous cache removal period implies that cache has no space to hold the object currently being accessed. Fixed cache removal period means that objects will be removed only at the beginning of removal period [2].

3.2.3 Cache size

Larger cache size implies that the cache can store more objects which means increased the hit ratio. Cache size is however expensive in terms of processing cost and complexity [3]. Therefore cache size is therefore a trade-off between cache cost and cache performance. In a small cache, a caching mechanism may either store many small sized objects or few large size objects. Maximum cacheable object size is a user defined factor that puts a limit on the size of objects that can be stored in the cache [2].

3.2.4 Cooperation

Coordination between user requests and many proxy caches in a hierarchal proxy cache environment.

3.2.5 Consistency

Consistency indicates maintaining updated replicas of objects in the cache.Other factors like protection copyright increase complexity. Also, non-cacheable objects are a subject of concern [2].

4. WEB CACHING ALGORITHMS

Three main important components have profound impact on caching management: cache algorithm, cache replacement and cache consistency. Cache replacement is however the heart of web caching. The design of efficient cache replacement algorithms is required to achieve highly sophisticated caching mechanism [11]. As cache size is limited, a cache replacement policy will determine which object is to be evicted to allow new objects and maximum utilization of the cache space efficiently. An efficient cache replacement algorithm is used to eliminate 'cache pollution'. Cache pollution means that the cache contains objects that are not frequently used in the near future which eventually lead to inefficient use of cache space. There are two kinds of cache pollutions. The term "cold cache pollution" refers to unpopular objects that remain in the cache for a long time whereas "hot cache pollution" refers to objects that stay in cache for long those were once but are no longer popular. Some of the caching algorithms are:

- **LRU:** Least recently used objects are eliminated first from the cache. LRU is simple to implement and is efficient in case of CPU memory where objects are uniform. However, LRU does not consider size of download latency of documents [14]. LRU suffers from cold cache pollution.
- **LFU:** Least frequently used objects are removed first from the cache. LFU is advantageous as it is simple to apply [3]. LFU does not however consider the size or download latency of objects and may keep obsolete objects indefinitely in the cache. LFU suffers from hot cache pollution.
- **SIZE:** Large objects are removed first from the cache. SIZE removes large objects from the cache and keeps a number of small ones and therefore has high hit rate. SIZE has a disadvantage as it may keep small documents indefinitely in the cache even if they have been not accessed by user in recent past [3]. It also has low byte hit rate.
- **GD-SIZE:** Greedy-Dual-Size (GDS) is an extension of SIZE policy. The algorithm combines several factors and assigns a key value for each object stored in cache. When cache space becomes saturated and new object is required to be stored in cache, the object with lowest key value is removed [14]. GDS removes objects which are no longer requested by users and therefore overcomes the drawbacks of SIZE policy. GDS does not take previous frequency of access for web objects in account [4].
- **GDSF:** Greedy-Dual-Size-Frequency (GDSF) extends GDS by containing the access frequency aspect in assigning key value. However, GDSF does not predict future accesses [4].

5. CONCLUSION

Web caching is the most suitable and most sustainable solution to reduce internet traffic and bandwidth consumption. Also, it is a low cost technique for minimizing web latency. Proxy caches are regularly used to reduce bandwidth globally. Various caching techniques cater various needs around the globe and improve experience of the web by minimizing latency. Various algorithms are employed with these techniques that cache objects and implement a replacement policy for optimum utilization of cached space. In this paper, we survey various caching techniques and algorithms while discussing their pros and cons that may help to reduce the bottleneck in data transmission through web.Due to heterogeneous nature of the web, the algorithms suited for CPU cache (Such as LRU, LFU) are not conditioned to favour web objects. Algorithms for CPU caches are more accustomed to static and uniform objects. However, the web has an ever-changing dynamic and complex structure and more dynamic algorithms are required. Future work may comprise of adaptive web caching systems need to be built that use machine learning to adapt to the important changes in usage patterns and store appropriate objects in cache while utilizing the cache space effectively.

6. REFERENCES

[1] Achuthsankar S. Nair, J.S. Jayasudha, "Improving Performance by World Wide Web by Adaptive Web Traffic Reduction", Proceedings of World Academy of Science, Engineering and Technology, Volume 17, December 2006

[2] Athena Vakali, George Pallis, "A study on web caching architectures and performance"

[3] Dhawaleswar Rao. CH, "Study of the web caching Algorithms for Performance Improvement of the response speed", Indian Journal of Computer Science and Engineering", Volume 3 – No. 2, April-March, 2012

[4] Farhan Mohamed, Abdul Samad Ismail, Siti Mariyam Shamsuddin, "Web caching and prefetching: Techniques and analysis in World Wide Web", Proceedings of the Postgraduate Annual Research Seminar, 2005

[5] Hossam Hassanein, Zhengang Liang and Patrick Martin, "Performance comparison of Alternative Web Caching Techniques", Proceedings of the Seventh International Symposium on Computers and Communications, 2002

[6] https://docs.trafficserver.apache.org/en/5.3.x/admin/http-proxy-caching.en.html

[7] https://docs.trafficserver.apache.org/en/5.3.x/admin/reverse-proxy-http-redirects.en.html

[8] https://docs.trafficserver.apache.org/en/5.3.x/admin/transparent-proxy.en.html

[9] https://en.wikipedia.org/wiki/Bloom_filter

[10] http://pages.cs.wisc.edu/~zj/active-cache/

[11] Lixia Zhang, Sally Floyd and Van Jacobson, "Adaptive Web Caching", April 25, 1997

[12] Mukesh Dawar, Charanjit Singh, "A review of web caching techniques", International Journal of Advanced Research in Computer Science and Engineering, Volume 4, Issue 3, March 2014

[13] Scott Michel, Lixia Zhang, Sally Floyd, "Adaptive web caching: Towards a new global caching architecture", Computer Networks and ISDN Networks, November 1998

[14] Waleed Ali, Siti Mariyam, Shamsuddin, Abdul Samad Ismail, "A Survey of Web caching and Prefetching", International Journal of Advanced Soft Computing Applications, Volume 3- No. 1, March 2011

[15] http://www.eecs.harvard.edu/vino/web/push.cache/node2.html