A Comparison And Review Of Web Caching Techniques And Caching Algorithms For Effective And Improved Caching

Jyothi Kadapala

ABSTRACT

The internet's popularity has led to delays, increased bandwidth use, and server strain. Web caching improves performance by storing frequently accessed content closer to users. This paper explores caching methods like proxy, cooperative, adaptive, push, and active caching. Redundant data transmission emphasizes the need for efficient caching algorithms optimizing limited storage space. Let us discuss the introduction to the techniques and requirement of such solutions. And we'll delve into specific caching solutions. Also explore the metrics and factors that affect caching performance, and finally focus on the algorithms used in caching.

1. INTRODUCTION:

Web caching, or HTTP caching, acts as a temporary storage for web data like HTML pages and images, aiming to reduce internet bandwidth usage, ease server loads, and minimize data delivery delays. The rise in internet users has led to increased slowdowns, prompting the exploration of web caching as a solution. The challenge lies in managing stored information across various caching methods. Despite the internet's exponential growth, the network infrastructure hasn't kept pace, causing performance issues exacerbated by multimedia content, push technology, and the shift to web-based services.

Researchers address internet performance by enhancing infrastructure, deploying technologies like cable modems, and utilizing compression. Among these approaches, caching is highlighted for its practicality and effectiveness, particularly in dealing with the physical distance between users and web content. In simple terms, caching stores frequently accessed web content closer to users, reducing retrieval delays and improving overall internet performance.

2. PROBLEM DEFINATION

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 a significant technique that has been explored to enhance overall performance. Caching promises the greatest performance gain and can be implemented with current technologies. It is also the only approach that addresses the physical distances between users and Web objects. In this Term Paper a review on caching techniques will be made and discuss about metrics and factors that influence caching performance and discuss

algorithms that are used for caching. This analysis can aid and assist us the importance of caching when designing and maintaining a website.

3. WEB CACHING TECHNIQUES

A cache is a storage space closer to where it's needed than the original source, making information retrieval faster. Caches are typically in memory or on a disk. Memory caches are quicker but don't persist after system restarts.

3.1 Proxy Caching

In proxy caching, the server checks if it has what the client wants. If yes, it provides it; if not, it fetches it from the source. Cached items speed up future requests. The server, close to the client, reduces delays and improves the web experience by lowering network traffic. Downsides include a single point of failure in the cache, requiring specific browser settings, and a lack of a system to add more caches as needed.

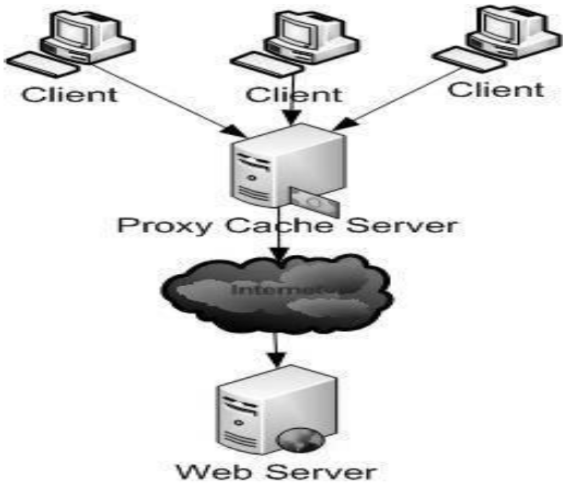

Figure 1. Proxy Caching

3.1.1 Reverse Proxy Caching

In reverse proxy caching, the cache server is placed near the servers instead of the client. It's effective when servers anticipate numerous requests simultaneously, acting as the main server to ensure uptime and high-quality service. This is beneficial for multiple virtual domains linked to a single physical site. Unlike forward proxy caching, Traffic Server manages requests for web data from origin servers. Acting as a memory cache, it accelerates server performance, quickly serving client requests by obtaining information from the actual origin server only when necessary.

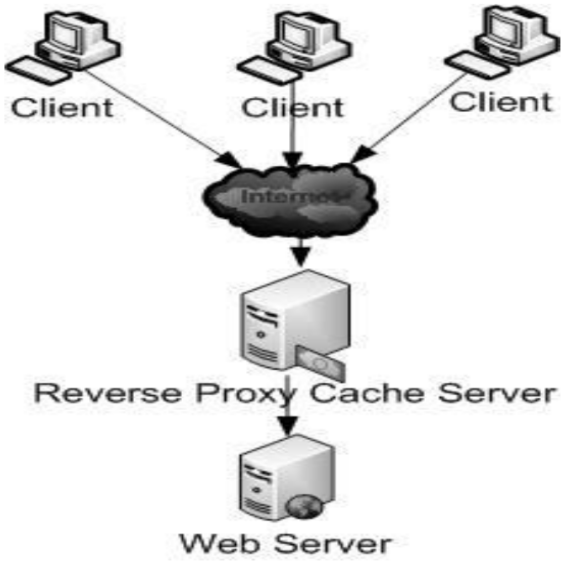

Figure 2. Reverse Proxy Caching

3.1.2 Transparent Caching

The proxy server approach requires browser setup, while transparent web caching intercepts HTTP requests at the gateway, redirecting them to caches without user awareness. It can deploy at the switch or router level, with router-level using policy-based routing and switch-level acting as a cost-effective load balancer. To address the strain from video streaming, operators implement transparent internet caching to handle diverse internet content. This reduces infrastructure strain, lowers data transfer costs for over-the-top (OTT) content, and enhances consumer broadband services, improving user performance by eliminating potential internet and content origin slowdowns. Caching allows operators to prioritize investments in the access network and deliver content at optimal speeds.

3.2 Adaptive Web Caching

Adaptive web caching uses replacement algorithms to analyze client requests and stores frequently accessed items in the cache. It involves multiple distributed caches that dynamically join and leave cache groups based on content demand. The concept envisions a flexible network of self-organizing server groups adapting to changing conditions. This network forms a scalable hierarchy, efficiently distributing popular web content to meet demand. The key components include communication paths between neighboring caches and the flow of data requests along these paths.

3.3 Push Caching

Push caching allows servers to decide when and where to cache objects, placing frequently accessed data closer to clients. This reduces network traffic without significantly affecting primary server loads. Push caching is beneficial for content providers, especially when additional servers are easily added. Proxy servers, with caching software and ample storage, are ideal for handling duplicated items. This approach effectively shifts the burden from overloaded primary servers to proxy servers and others, distributing the load without causing unacceptable strain.

3.4 Active Caching

The scheme lets servers provide cache applets with documents and needs proxies to use these applets when there's a cache hit. This way, proxies can process the necessary tasks without reaching out to the server. Cache applets enable servers to enjoy the advantages of proxy caching while keeping the ability to monitor user accesses and customize content presentation as needed.

3.5 Cooperative Caching-Swalla Architecture

Traditional cooperative caching schemes were created for network file systems, not for cluster-based web servers with specific content needs. To enhance web performance, the bottleneck is recognized as process utilization rather than network bandwidth. Swalla, a distributed and multi-threaded web server running on a workstation cluster, shares cache data and information between nodes. Swalla is beneficial for websites with extensive dynamic content requests like CGI requests, cohesively caching CGI results. Nodes communicate to exchange cached data and metadata. Other caching architectures include Internet Cache Protocol (ICP), Summary Cache, and Cache Digest. ICP coordinates proxy web caches, while Summary Cache and Cache Digest use Bloom Filters for memory-efficient representation of cache content directories. The difference lies in how they update directories, with Summary Cache extending ICP and Cache Digest using HTTP.

3.6 Performance evaluation of Caching Architectures

While various web caching techniques exist, none is universally effective in all scenarios. Each technique has a distinct architecture that optimally uses available resources based on its design. A reliable caching mechanism forms the foundation of any distributed computing architecture. This term paper aims to emphasize the significance of caching in creating an effective and efficient distributed system.

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

Table 1: Web Caching Techniques

4. PERFORMANCE METRICS AND FACTORS

Various measures and factors influence the choice of a suitable caching policy for a given environment. To achieve optimal efficiency, evaluating the performance of various algorithms based on the factors and metrics specific to the environment is beneficial.

4.1 Performance metrics

4.1.1 Hit ratio

The hit ratio is usually the proportion of objects retrieved through a caching policy compared to the total number of requests. A higher hit ratio suggests a more effective caching policy. However, this measure may only be meaningful when the objects are of uniform size.

4.1.2 Byte hit ratio

The byte hit ratio is the proportion of bytes retrieved from the cache compared to the total bytes accessed. When dealing with objects of varying sizes, the byte hit ratio serves as a more effective metric for measurement.

4.1.3 Bandwidth utilization

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

4.1.4 User response time

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

4.1.5 Cache server CPU and I/O utilization

This refers to the portion of total CPU cycles or the time it takes to retrieve objects from disk. Latency is linked inversely to the object hit ratio, as a cache hit is quicker than fetching from the origin server. However, improving one metric doesn't necessarily improve the other. For instance, a higher hit rate doesn't automatically reduce latency.

4.2 Performance factors

4.2.1 User Access Patterns

If a user frequently accesses small-sized objects, these smaller objects become obvious candidates for caching. User access patterns change over time, so an effective caching algorithm should be dynamic as well. Cache replacement algorithms determine which objects to discard when the cache is full.

4.2.2 Cache Removal Period

The cache removal period dictates when documents will be removed, typically when the cache is full. Continuous cache removal means the cache lacks space for the currently accessed object. Fixed cache removal occurs only at the beginning of the removal period.

4.2.3 Cache Size

A larger cache size allows storing more objects, increasing the hit ratio. However, a larger cache is more costly in terms of processing and complexity [3]. Cache size becomes a trade-off between cost and performance. In a small cache, the mechanism may store many small objects or a few large ones. The maximum cacheable object size is a user-defined limit on the object size.

4.2.4 Cooperation

Coordination between user requests and multiple proxy caches in a hierarchical proxy cache environment.

4.2.5 Consistency

Consistency involves maintaining updated replicas of objects in the cache. Factors like copyright protection add complexity. Non-cacheable objects are also a concern.

5. WEB CACHING ALGORITHMS

Three important components significantly impact caching management: cache algorithm, cache replacement, and cache consistency. Cache replacement is the core of web caching. Efficient cache replacement algorithms are essential for a sophisticated caching mechanism. Given the limited cache size, the replacement policy dictates which object gets removed to make room for new ones, ensuring optimal use of cache space. Effective cache replacement algorithms prevent "cache pollution," where the cache holds infrequently used objects, leading to inefficient use of space.

Several caching algorithms include:

- **LRU** (Least Recently Used): Eliminates the least recently used objects first. While simple and efficient for uniform objects, LRU overlooks document size and download latency, suffering from cold cache pollution.

- **LFU** (Least Frequently Used): Removes the least frequently used objects first. It's simple but may keep obsolete objects indefinitely, leading to hot cache pollution.

- **SIZE**: Removes large objects first, keeping a number of small ones for a high hit rate. However, it may retain small documents indefinitely, even if not accessed recently, resulting in a low byte hit rate.

- **GD-SIZE** (Greedy-Dual-Size): An extension of SIZE that assigns a key value to each object, removing the one with the lowest key value when the cache is full. It overcomes SIZE's drawbacks but doesn't consider the previous frequency of access.

- **GDSF** (Greedy-Dual-Size-Frequency): Extends GDS by including access frequency in assigning key value. However, GDSF doesn't predict future accesses.

6. CONCLUSION

Web caching proves to be the most suitable and sustainable solution for reducing internet traffic, minimizing bandwidth consumption, and offering a cost-effective means to decrease web latency. Proxy caches, commonly employed globally, play a significant role in bandwidth reduction. Diverse caching techniques address various global needs, enhancing the web experience by reducing latency. Different algorithms accompany these techniques, caching objects and implementing replacement policies for optimal use of cached space. This paper reviews various caching techniques and algorithms, discussing their advantages and drawbacks in alleviating data transmission bottlenecks on the web.

Given the web's diverse nature, algorithms designed for CPU caches (like LRU, LFU) may not be well-suited for web objects. CPU cache algorithms are more accustomed to static and uniform objects, whereas the web's dynamic and complex structure requires more adaptive algorithms. Future work could involve building adaptive web caching systems that utilize machine learning to adjust to significant changes in usage patterns, effectively storing appropriate objects in the cache.

7. 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] 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

[4] 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

[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