ISSN:0975-9646

Deepak Sachan et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3112 - 3115

Performance Improvement of Web Caching Page Replacement Algorithms

Deepak Sachan, Dhawaleshwar Rao ch

School of Computer Science,
Lovely Professional University, India.

***Abstract*- As the number of World-Wide Web users grows, this increases both network load and server load. Caching can reduce both loads by migrating copies of server files closer to the clients that use those files. Several types of caching are used over the Internet, including client caching, server caching, and more recently, proxy caching. In this paper, we present the overview of some popular web caching replacement algorithms, Least recently used (LRU), Least frequently used (LFU), SIZE and implementation of these algorithms to improve the performance of page replacement.**

Keywords- World Wide Web, Proxy caching, Web Caching Replacement Algorithm.

INTRODUCTION

Web caching is a technology used to reduce the transmission of network traffic. Main focus of web caching is to enhance accessibility to the Web. It can be beneficial to a wide spectrum of users including those dependent on slow network connection as well as those relying on faster internet connections. The word, caching refers to the process of saving documents for future use. Web caching, to reduce traffic load, saves copies of content, obtained from the Web, closer to the end user, in order to improve accessibility to the content. Some of the main reasons for which a user would opt for Web caching include the following:

To increase the bandwidth availability by curbing the transmission of redundant data,

For reducing network congestion,

For improving response times and

For achieving savings in terms of cost.

The fundamental question in Web caching is how we know if something can be cached or perhaps cannot be cached. It is not hard to guess that certain types of content may not be worth caching. For example, cookies , personalized content and dynamically generated content. However, it would be desirable to cache some of these (if it was possible to do so) for the sake of efficiency, by adopting some means or the other. It may of course be worthwhile to cache content that is requested often. On the other hand, it may not be beneficial to cache content that is not likely to be requested frequently. Hence, we see that there are many factors that determine the cache ability of content. [1]

In proxy caching, a proxy server gets HTTP requests from clients and holds it, and on tracing the object asked for in its cache, gives to the client. If the object requested not found in the cache, then the request is send on behalf of the user to the origin server.

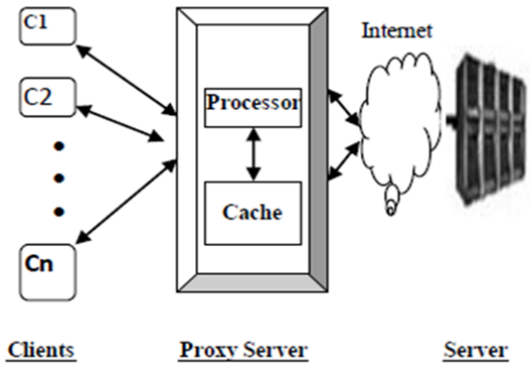

Fig.1 Proxy server Caching.

CACHE DEPLOYMENT OPTIONS [7]

WWW caching provides an efficient remedy to the latency problem by bringing documents closer to clients. Caching can be deployed at various points in the Internet: within the client browser, at or near the server to reduce the server load, or at a proxy server.

- Forward Proxy Caching
- Reverse Proxy Caching
- Transparent Caching.

Forward Proxy Caching: In forward proxy caching, caches are generally positioned at the edge of a network. This is done so that a large number of internal consumers may be serviced. Placement of proxy caches can lead to bandwidth savings, quicker response times, and enhanced accessibility to static Web objects. However, proxy cache placement could become a single point of failure in the network which can cause problem.

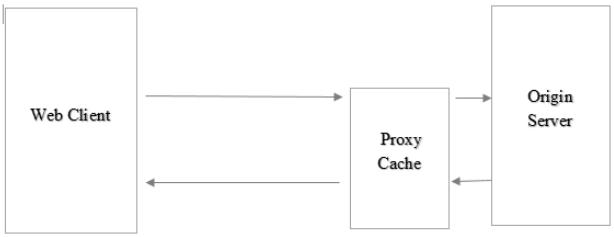

Fig.2 Forward Proxy Caching

Reverse proxy caching: In reverse proxy caching, caches are placed near the source of the content rather than the destination of the content. This placement of caches is helpful for servers that counters a large number of requests and desire to maintain a superior quality of service.

Transparent Caching: In transparent caching, the caches intercept HTTP requests and anticipate them to Web cache servers. Transparent proxy caching overcomes the problem of configuration of Web browsers encountered in proxy caching. Transparent caching avoids the need for configuring browsers of users.

PROS AND CONS OF WEB CACHING

Web caching system when properly deployed can provide substantial of bandwidth wastage. This leads to cost savings, network traffic reduction, improved access and better content availability. Content may be fetched from nearby caches instead of faraway origin servers. This also helps when connections to the origin servers are not readily available. If updation of web content is not done regularly then they may be returning stale content to the users. If content is not found in a cache then a cache miss occurs and this results in an increase in the response time.

There are many other factors which causes in bottleneck of web caching like poor configuration, cost consideration, Quality of service (QoS), effort involved in setup of cache and security and privacy issues. [7]

PERFORMANCE METRICS

Replacement policies rely on key metrics to achieve their goals. Such policies attempt to optimize various performance metrics, including the file hit ratio, the byte hit ratio, the average download time, and the delay saving ratio. [2]

FILE HIT RATIO: The hit ratio is the percentage of all requests that can be satisfied by searching the cache for a copy of the requested object.

BYTE HIT RATIO: The byte hit ratio represents the percentage of all data that is transferred directly from the cache rather than from the origin server.

Metric	Definition
File hit ratio	$\frac{\sum_{i \epsilon R} \boldsymbol{h_i}}{\sum_{i \epsilon R} \boldsymbol{f_i}}$
Byte hit ratio	$\frac{\sum_{i \epsilon R} \boldsymbol{S_i}.\boldsymbol{h_i}}{\sum_{i \epsilon R} \boldsymbol{S_i}.\boldsymbol{f_i}}$
Saved bandwidth	Directly related to byte hit ratio
Delay saving ratio	$\frac{\sum_{i \epsilon R} \boldsymbol{d_i}.\boldsymbol{h_i}}{\sum_{i \epsilon R} \boldsymbol{d_i}.\boldsymbol{f_i}}$
Average download time	$\frac{\sum_{i \epsilon R} \boldsymbol{d_i}.(\mathbf{1}-\boldsymbol{h_i}\|\boldsymbol{f_i})}{\|\boldsymbol{R}\|}$
Notation: S_i =size of document i f_i =total no of requests for document i h_i =total number of hits for documents i d_i =mean fetch delay from server for document i R= set of all accessed documents ||R||=size of R	

Figure 3.various performance metrics and definition.

COMMON WEB CACHING REPLACEMENT ALGORITHMS

There are several ways to categorize cache replacement algorithms for an overview of Web cache replacement strategies: [1]

- Traditional algorithms
- Key based algorithms
- Cost based algorithms

Traditional algorithms

Least Recently Used (LRU) evicts the object from the cache that was requested for the least number of times.

Least Frequently Used (LFU) evicts the object that was retrieved least frequently from the cache. [3]

Key based algorithms

Key based replacement algorithms evict data from caches on the basis of a primary key. Ties may be broken by the use of secondary or tertiary keys or in other ways. [4]

SIZE, this algorithm removes large size documents first. As it removes big objects first, therefore keeps small objects in the cache, resulting high request hit rate.

LRU-Min, LRU- Threshold, HYPER-G are the main key-based page replacement algorithms.

Cost based algorithms

The algorithms in this class evicts object from cache on the basis of a cost function based on parameters such as the last access time, the time at which an object was put into the cache, the expiration time and so on.[5]

Greedy-Dual-Size (GDS) tags a cost with every object and expels the object that has the lowest cost or size. It is also possible to tag a utility function with every object and expel the object that is least useful for diminishing the total latency.

Lowest Relative Value, Size-adjusted LRU, Least Normalized Cost Replacement, Server-assisted scheme, Hierarchical Greedy Dual, Bolot/Hoschka are some other useful page replacement algorithms which are generally used now a days.

PROPOSED WORK

According to definition of SIZE algorithm, it evicts objects from the cache having largest size. Sometimes, it may be possible that there is a case of tie between different objects in a cache. For e.g. we open different sites which having same size, cached sequentially in cache .so in this case which site have to be evicted, it's very difficult to decide. So to solve this problem, we have to use any secondary keys as a parameter. In our proposed work, we will use the Least recently used (LRU) algorithm to solve this type of problem. That means our secondary parameter will be time since last access. Using this technique will definitely improve the performance of page replacement. Hit ratio will be used as performance metrics.

PRESENT WORK

To implement our proposed work, we prepare an interface developed in C#. The tool used for development is visual studio 2010. In this interface we will calculate the hit ratio separately for LRU, SIZE and our proposed algorithm. We use dataset of 100 different websites to check our performance in the basis of hit ratio.

In our interface, we calculate the time of access and size of webpage simultaneously. Using these values, we able to evict the object from cache and calculate the hit ratio and generate a graph of hit ratio values.

	LRU HIT	Size HIT
	6	7
Page Faults	31	30	0

LRU	Time	Size	bytes
https://www.onlinesbi.com	4/21/2014 9:17:57 PM	https://www.irctc.co.in	52736
https://www.sbi.co.in	4/21/2014 9:18:40 PM	https://www.baidu.com	0
https://www.netpnb.com	4/21/2014 9:16:34 PM	https://www.einfoedu.com	0
https://www.irctc.co.in	4/21/2014 9:19:15 PM	https://www.yepme.com	0

Lru Hit Ratio	0.193548387096774
Size Hit Ratio	0.233333333333333

Fig.4 Interface of Proposed work.

CONCLUSION AND FUTURE WORK

After checking performance on the basis of hit ratio, using dataset of 100 different websites, we concluded that performance of size algorithm is better than least recently used algorithm. As our proposed algorithm, using time since last access as a secondary key to overcome the drawback of size algorithm, it performs better than previous algorithms.

In future, we can also work on performance improvement of size algorithm using tertiary key. As we seen in our implementation result, size algorithm is not able to maintain the frequency of objects in cache. So to maintain the frequency, we would use frequency as tertiary key.

REFERENCES

[1]. Abdullah balamash and Marwan krunz, "An Overview of Web Caching Replacement Algorithms", IEEE Communications Surveys & Tutorials, 2004, Vol.6 (2).

[2]. Martin Arlitt, Rich Friedrich, and Tai Jin," Performance Evaluation of Web Proxy Cache Replacement Policies", Internet Systems and Applications Laboratory HP Laboratories, HPL-98-97(R.1) October, 1999.

[3]. Yogesh Niranjan, Shailendra Tiwari," Design and Implementation of Page Replacement Algorithm for Web Proxy Caching", International journal of Computer Technology & Applications, Vol.4 (2),2013.

[4]. Harshal N. Datir, Yogesh H. Gulhane, P.R. Deshmukh," Analysis and Performance Evaluation of Web Caching Algorithms", International Journal of Engineering Science and Technology, feb.2011.

[5]. Vinit A. Kakde, Prof. Sanjay K.Mishra, Prof. Amit Sinhal, Mahendra S. Mahalle," Survey of Effective Web Cache Algorithm", International Journal of Science and Engineering Investigations, vol. 1, issue 2, March 2012.

[6]. S.V.Nagaraj, "Web Caching and Its Applications", Kluwer Academic Publishers, 2004.

[7]. Jia Wang," A Survey of Web Caching Schemes for the Internet", Cornell Network Research Group (C/NRG), Department of Computer Science, Cornell University.