International Journal of
Science and Engineering Investigations vol. 1, issue 2, March 2012

Survey of Effective Web Cache Algorithm

Vinit A. Kakde[1], Prof. Sanjay K.Mishra [2], Prof. Amit Sinhal[3] ,Mahendra S. Mahalle[4]
[1]M.Tech(IT),TIT, Bhopal
[2]Assistant Lecturer, Department of IT, TIT ,Bhopal
[3] Professor, Dept. of CSE, TIT, Bhopal
[4]M.E.(CSE), SIPNA COE, AMRAVATI
([1]vinit.kakde@gmail.com, [2]sanjaymishra2006@gmail.com, [3]amit_sinhal@rediffmail.com, [4]mahalle.mahendra85@gmail.com)

***Abstract*-** The increasing demand for World Wide Web (WWW) services has made document caching a necessity to decrease download times and reduce Internet traffic. To make effective use of caching, an informative decision has to be made as to which documents are to be evicted from the cache in case of cache saturation. This is particularly important in a wireless network, where the size of the client cache at the mobile terminal (MT) is small. Several types of caching are used over the Internet, including client caching, server caching, and more recently, proxy caching. In this article we review some of the well known proxy-caching policies for the Web.

We describe these policies, show how they operate, and discuss the main traffic properties they incorporate in their design. We argue that a good caching policy adapts itself to changes in Web workload characteristics. We make a qualitative comparison between these policies after classifying them according to the traffic properties they consider in their designs.

***Keywords*-** *Web Cached;Hit Ratio;Replacement policies.*

I. Introduction

The World Wide Web has become one of the predominant services to distribute various kinds of information. With the explosive growth of the World Wide Web, web caching has become an important technique to improve network performance. Due to the limited cache space, it is impossible to store all the web objects in the cache. As a result, cache replacement algorithms are used to determine a suitable subset of web objects to be removed from the cache to make room for a new web object. An overview of web caching replacement algorithms can be found in. However, the improvement of network performance, such as access latency reduction achieved by caching web objects, does not come completely for free. In particular, maintaining the content consistency with the primary servers generates extra requests. Many web cache implementations depend on some consistency algorithms to ensure a suitable form of consistency for the cached objects.

The Web documents are cached either directly by the browser or by a proxy server which is located close to the clients. Cache replacement algorithms, which dynamically select a suitable subset of documents for eviction from the cache, play a central role in the design of any caching component; these algorithms have been extensively studied in the context of operating system virtual memory management and database buffer management. Cache replacement algorithms usually maximize the cache hit ratio by attempting to cache the data items which are most likely to be referenced in the future. Since the future data reference pattern is typically difficult to predict, a common approach is to extrapolate from the past by caching the data items which were referenced most frequently. This approach is exemplified by, e.g., the LRU and LFU cache replacement algorithms. In addition to maximizing the cache hit ratio, a cache replacement algorithm for Web documents should also minimize the cost of cache misses, i.e., the delays caused by fetching documents not found in the cache. Clearly, the documents which took a long time to fetch should be preferentially retained in the cache. For example, consider a proxy cache at Nagpur University. The cache replacement algorithm at the proxy found two possible candidates for replacement. Both documents have the same size and are referenced with the same rate, but one document originates from the University of RGPV while the other is from SGBAU University. The cache replacement algorithm should select for replacement the document from the University of RGPV and retain the document from SGBAU University because upon a cache miss the former can be fetched much faster than the latter.

II. RELATED WORK

Figure 2.1 shows generalized web cache architecture in web caching if the client is requesting a page from server it will fetch from the server and will give response to the server. A web cache sits between web server and a client and watches request for web pages. It caches web documents for serving previously retrieved pages when it receives a request for them.

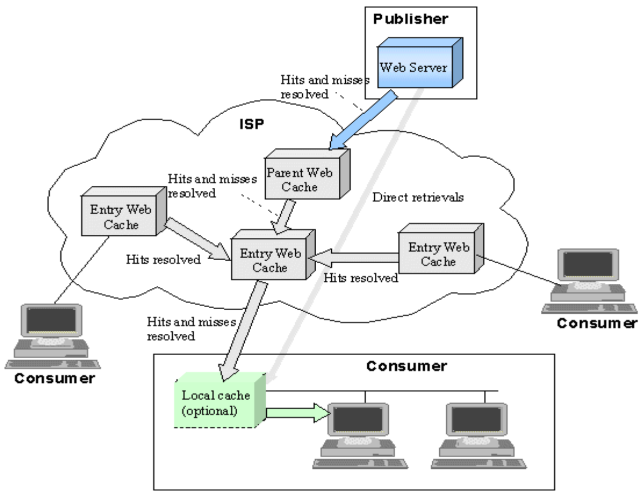

Figure 1: Generalized Web Cache Architecture

A number of caching policies have been proposed in an attempt to minimize several cost metrics such as hit ratio, byte hit ratio, average latency and total. A good survey that describes several cache replacement polices can be found in .This survey classifies cache replacement policies into categories. The focus of our work is on evaluating cache replacement policies that aims at improving cache hit ratio, which is the most general metric to evaluate the performance of a caching system.

III. EVALUATION METHODOLOGY

This section discusses our methodology. The performance metrics used in our evaluation are discussed in Section A Section Section B presents the caching system simulator used in our analysis.

A. *Performance Matrics*

Since we are interested in evaluating periods of poor and good performance we measure several metrics over time. This analysis allows us to detect periods of variations of performance and to investigate the factors that cause these variations. Each metric was measured in different time scales. The metrics used in our performance evaluation are:

Hit Ratio - The effectiveness of a caching policy can be measured by its hit ratio (the percentage of requests satisfied by the cache). The hit ratio of each time interval considers only the requests of the analyzed interval; however the workload imposed to the cache contains re-quests of all the past intervals.

Maximal Improvement -We evaluate the maximum hit ratio a new caching replacement policy can improve over the simple LRU policy. In order to define the maximal improvement, we first define as the percentage of *first timers* on a given interval. Moreover, let be the hit ratio on when the LRU is the current cache replacement policy used by the caching system. We define the Maximal Improvement as follows: As expected, the maximal improvement is measured for each time interval, considering only the requests of current interval. This metric can be understood as: the maximum hit ratio (100%) minus the hit ratio of a given replacement policy (we choose LRU for simplicity) minus the percentage of objects that appeared for the first time on trace. The result is the potential hit ratio that another replacement cache policy could obtain over LRU.

B. *Caching System Simulator*

Since the goal of the experiments is not to analyze cache replacement policies, but to evaluate the maximal improvement that can be applied to caching system and understand the main reasons for variations on cache performance over time, we developed a simulator that assumes the simple LRU policy. We use a cache size of 10% of the number of distinct objects in each trace. The results for the other values of cache sizes are equivalent, considering that for large cache sizes, almost all objects are kept in cache which increases hit ratio.

IV. CACHING REPLACEMENT POLICIES

Cache replacement policy plays an extremely important role in web caching. Hence, the design of efficient cache replacement algorithms is required to achieve highly sophisticated caching mechanism. In general, cache replacement algorithms are also called web caching algorithms. As cache size is limited, a cache replacement policy is needed to handle the cache content. If the cache is full when an object needs to be stored, the replacement policy will determine which object is to be evicted to allow space for the new object. The optimal replacement policy aims to make the best use of available cache space, to improve cache hit rates, and to reduce loads on the origin server. The different page replacement policies are as follows:

A. *Least Recently Used(LRU) Page Replacement Policy*

The simplest and most common cache management approach is Least-Recently Used (LRU) algorithm, which removes the least recently accessed objects until there is sufficient space for the new objects.

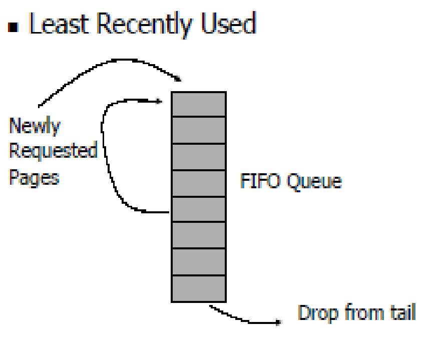

Figure 2: Example of LRU page replacement policy

LRU is easy to implement and proficient for uniform size objects, like in the memory cache. However, it does not perform well in web caching since it does not consider the size or the download latency of objects. This algorithm exploits the temporal locality of the user's accesses, and it is very simple to implement because the eviction mechanism requires only the access time-stamp.

B. *Least Frequently Used (LFU) Page Replacement Policy*

It is another common web caching that replaces the object with the least number of accesses.

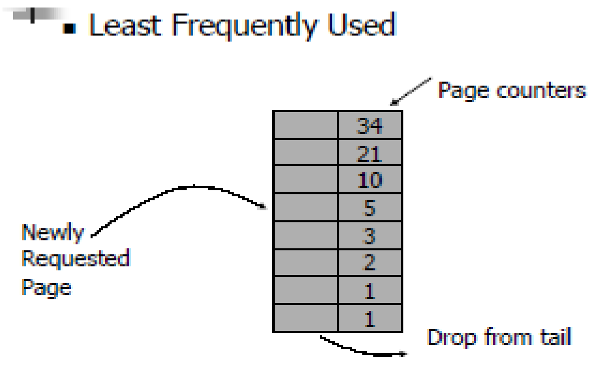

Figure 3: Example of LFU page replacement policy

LFU keeps more popular web objects and evicts rarely used ones. However, One potential drawback of LFU is that some objects may accumulate large reference counts and never become candidates for replacement, even if these objects are no longer in the active set (i.e., the cache could become polluted with inactive objects).

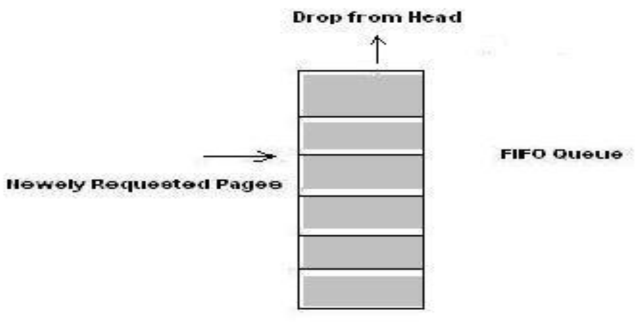

Figure 4: Example of MRU page replacement policy

C. *Most RecentlyUsed (MRU) Page Replacement Policy*

The MRU is also called as fetched and discard policy. MRU algorithm removes the most recently used resource first. As shown in figure 4.3 the pages are drop from head in the queue which maintains the list of older web pages. MRU works contrast to LRU policy. The algorithm is the best choice when access of resources is highly unpredictable. MRU policy is most often used where historical information is to be accessed.

D. *Segmented LRU(SLRU)Page Replacement Policy*

To alleviate the web object pollution problem The Segmented LRU (SLRU) policy designed and it is use in a disk cache we include it in this study because it considers both frequency and regency of reference when making a replacement decision. The SLRU policy partitions the cache into two segments: an unprotected segment and a protected segment (reserved for popular objects). On the initial request for an object, the object is added to the unprotected segment. When a cache hit occurs, the object is moved to the protected segment. Both segments are managed with the LRU policy. However, only objects in the unprotected segment are eligible for replacement. This allows the once popular objects to remain in the cache for a longer period of time in case they regain their popularity. If space is needed to add these objects, the least recently used objects in the unprotected segment are removed. This policy requires one parameter, which determines what percentage of the cache space to allocate to the protected segment. The SLRU policy performs best when a balance is found that allows for popular objects to be retained for long periods of time without becoming susceptible to pollution.

V. CONCLUSION

This paper provides a study about the viability of deployment of new caching replacement policies. We analyze whether new caching replacement policies can significantly improve a cache system performance and the main characteristics that cause poor and good performance in caching systems. The enormous raise of the traffic in the internet due to the exponential augmentation of user’s interactions with web servers is generating a lot of blockages. As a corollary, users frequently experience high delay when access these web pages and even more aborting or resetting connection with real-time web applications, such as online flight booking, online banking amongst others. This topic finds the elucidation to this predicament with the prologue of and web cache. The study is made on different types of page replacement policies in cache system and the comparative study is made between them. After performing the experimentation on real-time application experimental results have revealed that the proposed approach can improve the performance of hit ratio (HR). The cache replacement algorithms are implemented to reduce the response time of the user. The existing algorithms are being studied and the improvements in the algorithms are being proposed.

REFERENCES

[1] NLANR. National Laboratory for Applied Network Research (NLANR) http://www.nlanr.net.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira. Characterizing Reference Locality in the WWW. In *Proc.of PDIS*, December 1996.

[3] A. Bestavros, R. Carter,M. Crovella, C. Cunha, A. Heddaya, and S. Mirdad. Application-Level Document Caching in the Internet. In *Proc. IEEE SDNE*, 1995.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications.In *Proc. of IEEE Infocom*, April 1999.

[5] S. Jin, A. Bestavros. GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams, Technical Report of Boston University, 1999- 009, August 21, April 4, 2000.

[6] P.Lorensetti, L.Rizzo, L.Vicisano. Replacement Policies for Proxy Cache.Manuscript, 1997.

[7] S.Williams, M.Abrams, C.Stanbridge, G.Abdulla, E.Fox: Removal Policies in Network Caches for World-Wide Web Documents. In Proceedings of the ACM Sigcomm96,August,1996, Stanford University.

[8] E. O'Neil, P. O'Neil, and G. Weikum, "The lruk Page Replacement Algorithm for Database Disk Buffering," *Proc. ACM SIGMOD Int'l.Conf.Management of Data*, Washington, D.C., USA, May 1993, pp. 297–306.

[9] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-Based Traces," *IEEE/ACM Trans. Net*., vol. 1, no.3, Jan 1999, pp. 134–233.

[10] N. Niclausse, Z. Liu, and P. Nain, "A New Efficient Caching Policy for the World Wide Web," *Proc. Internet Server Perf.Wksp. (WISP '98)*, Madison, WI, USA, June 1998, pp. 119–28.

[11] A. Foong, Y.-H. Hu, and D. Heisey, "Adaptive Web Caching Using Logistic Regression," *Proc. 1999 IEEE Signal Processing Society Wksp.*, Madison, WI, Aug. 1999, pp. 515–24.

[12] A. Tanenbaum, *Modern Operating Systems*, Prentice Hall, Inc, 1992.

[13] Measurement and Analysis Web Page Response Time Understanding and measuring performance test results by Alberto Savoia.

[14] Caching Behaviors of web browser by Dawn Pazych AccelerationSystem Architecture (ACA) Nov-07.

[15] Web Caching: Optimizing for internet and Web Traffic (White paper).

Vinit A. Kakde received the B.E.degree in Information Technology from SGBAU University, Amravati, Maharashtra, in 2007. From He is currently working toward his M.Tech degree at the University of RGPV, Bhopal. His research interests are in theory of computation, traffic analysis, network systems modeling, and performance evaluation.

Prof. Amit Sinhal complete his B.E. in Computer Engineering from NIT Surat in 1996, M.Tech in Computer Science & Engineering from SATI Vidisha in 2005 and pursuing Ph.D. from Rajiv Gandhi Technical University, Bhopal. He worked in various reputed software development companies as Project Lead and University Institute of Technology, Barkatullah University Bhopal as Assistance professor. Currently he is working in Technocrats Institute of Technology, Bhopal as professor in Computer Science Engineering department.

Prof. Sanjay K. Mishra is-currently working in TIT, Bhopal as assistant lecturer in Information Technology Department. He is currently working toward his PhD degree. His research interests are in network systems modeling and performance evaluation.

Mahendra S. Mahalle received the B.E.degree in Computer Science & Engineering, from SGBAU University, Amravati, Maharashtra, in 2008. From He is currently working toward his M.E degree at the University of SGBAU, Amravti. His research interests are in Data communication, database and network analyzer.