International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 6, Issue 11 (April 2013), PP. 15-22

A Novel Hybrid Policyfor Web Caching

SirshenduSekhar Ghosh[1], Vinay Kumar[2], Aruna Jain[3]
[1]Research Scholar, Dept. of InformationTechnology, Birla Institute of Technology, Mesra, Ranchi, Jharkhand,India
[2]M.Tech Student, Dept. of Information Technology, Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India
[3]Associate Professor, Dept. of Information Technology, Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India

Abstract:-Web caching is one of the most successful solutions for improving the performance of Web-based system by keeping Web objects that are likely to be used in the near future close to the clients. The core of the caching system is its page replacement policy, which needs to make efficient replacement decision when its cache is full and a new document needs to be stored. Several replacement policies based on recency, frequency, size, cost of fetching object and access latency have been proposed for improving the performance of Web caching by many researchers. However, it is difficult to have an omnipotent policy that performs best in all environments as each policy has different design rational to optimize different resources. In this paper, we have proposed a Novel Hybrid Replacement Policy for Web Caching which is an efficient combination of two traditional well-known policies, Least Recently Used (LRU) and Least Frequently Used (LFU). We have developed experimental codes along with trace-driven simulation and analyzed the results. Analysis of the results show that our proposed Novel Hybrid Policy gives better performance characteristics over other two policies used individually in terms of Hit ratio and Latency saving ratio.

Keywords: -Web Caching, Replacement policy, LRU, LFU, Hybrid replacement policy, Hit ratio, Latency saving ratio.

I. INTRODUCTION

Web caching is a technique which aims to reducing WWW network traffic and improving response time for the end users. It stores the popular Web objects already requested by users into a pool close to the client-side to avoid repetition of requests for such objects to the original Web Servers. Web caching mechanisms are implemented at three levels: client level, proxy level and original server level as shown in Fig 1.

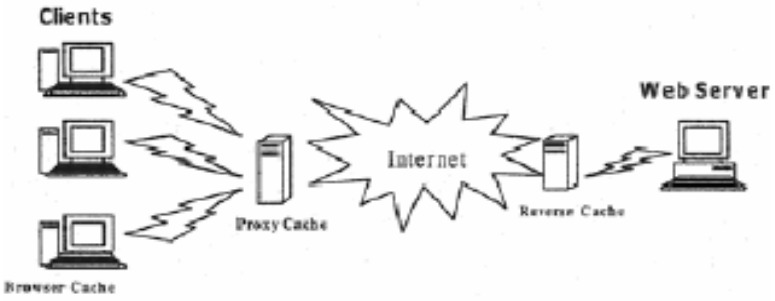

Fig.1: Possible Web cache locations

Amongst these, proxy servers play key role between users and Web servers in reducing the response time and thus saving network bandwidth. Users make their connection to proxy applications running on their hosts. The proxy connects the server and relays data between the user and the server. At each request, the proxy server is contacted first to find whether it has a valid copy of the requested object. If the proxy has the requested object this is considered as a cache hit, otherwise a cache miss occurs and the proxy must forward the request on behalf of the user to retrieve the document from another location such as a peer or parent proxy cache or the origin server.Once the copy of the document has been retrieved, the proxy cancomplete its response to the client. If the document is cacheable (based on information provided by the origin serveror determined from the URL) the proxy may decide to add a copy of the document to its cache. Proxy caches are often located near

network gateways to reduce the bandwidth required over expensive dedicated Internet connections. When shared with other users, the proxies serve many clients with cached objects from many servers.

Therefore, for achieving better response time, an efficient caching approach should be built in a proxy server as it can effectively server more user requests. It takes advantage of the Web object's temporal locality to reduce user-perceived latency, bandwidth consumption, network congestion and traffic.Furthermore, as it delivers cached objects form Proxy Servers; it reduces external latency and improves reliability as a user can obtain a cached copy even if the remote Server is unavailable. Due to the larger number of users connected to the same proxy server, object characteristics (popularity,spatial and temporal locality) can be better exploited increasing the cache hit ratio and improving Webperformance.

The performance of a Web proxy caching scheme, mainly dependent on the cache replacement policy (identify the objects to be replaced in a cacheupon a request arrival) which has been enhanced by the underlying proxy server. If a cache is full and an object needs to be stored, the policy will determine which object to be evicted to make room for the new object. Replacement policies play a key role in picking out which documents to sacrifice. In practical implementation a replacement policy usually takes place before the cache is really full. The goal of the replacement policy is to make the best use of available resources including disk space, processing speed, server load, and network bandwidth. Therefore, we must resort to an approach, which will predict the future users' requests and retain in cache the most valuable objects thus improving the Web latency.

Cache size is also an important factor influencing the cache performance. The larger the cache size is the more documents it can maintain and the higher the cache hit ratio is. But, cache space is expensive and in today's computing world where cache size is not a big issue but some significant problems such as updating the large cache which involves complexity and thus results in an increase in response time. Therefore, an optimal cache size involves a tradeoff between cache cost and cache performance. Document size is also associated with cache performance. Given a certain cache size, the cache can store more small sized documents or fewer large sized documents. Maximum cacheable document size is a user-defined factor that places a ceiling on the size of documents that are allowed to be stored in the cache. Furthermore, there are two others factors which are cooperation and consistency. Cooperation refers to the coordination of users requests among many proxy caches in a hierarchical proxy cache environment. Cache consistency refers to maintaining copies of documents in cache that are not outdated. There are also factors which indirectly affect proxy cache performance such as protection copyright, which increases the complexity of proxy cache design.

Motivated by the wealth of research in replacement policies of Web caching [1,2,3,4], in this paper we have proposed a Novel HybridCache Replacement Policy combining two traditional policies LRU and LFU.These policies are particularly popular because of their simplicity and fairly good performance in many situations. LRU policies evict the least recently referenced object first. It is designed on the assumption that a recently referenced document will be referenced again in the near future.It has the advantage that it is a very simple policy which can be implemented to work very efficiently. But it has the handicap that it does not take into account the frequency and size of documents.On the other hand, LFU is a simple policy that evicts the least frequently referenced object first. But it has the disadvantage that documents that have been referenced very often in the past but they are not popular any more are maintained in cache and not evicted. This effect is called "cache pollution". It does not take into account the document size either.

In this paper we have proposed a novel hybrid combination of the above mentioned two traditional policies which effectively integrate them and as well as give better response time in terms of Hit ratio and Latency saving ratio. We have also proposed a system architecture scheme deploying the Hybrid policy which can be easily incorporated in present Web infrastructure.

The rest of the paper is organized as follows.Section 2 describes the background and related work in this area. Section 3 describes the proposed Novel HybridCache Replacement Policy. Section 4 describes the proposed System Model and its components deploying the HybridReplacement Policy. Section 5 shows the results and discussions. Section 6 concludes the paper showingfuture scope towards this research direction.

II. RELATED WORK

The size and complexity of Web pages is increasing drastically day by day due to increasing dynamic and flashy contents. Hence it is imperative to have a combined and intelligent kind of policy for identifying the documents for replacement to improve cache performance.Web caching has been studied from many different anglesby the research community over the years.All these replacement policies are mainly based on the following attributes: recency, frequency, size, cost of fetching object, access latency of object, modification time, expiration time etc.

Balamash et al. [3] give an overview of various replacement algorithms. They classify these strategies into different categories, namely recency, frequency, recency/frequency and function based and also conclude

that GDSF outperform when cache size is small. However their survey did not provide insight on which of these strategies perform best, or how do these strategies compare to each other.

Podlipnig et al.[4] conducted an extensive survey on Web cache replacement strategies, and classified them as (a) recency-based (b) frequency-based (c) recency/frequency-based (d) function-based and (e) randomized. Recency-based strategies were based on "least recently used" (LRU), which uses the concept of temporal locality. Frequency-based strategies were modeled after "least frequently used" (LFU) and were best suited for static environments. LFU-based strategies tend to suffer from cache pollution if an appropriate aging factor is not used. Recency/frequency-based strategies consider both recency and frequency in making their decisions. Function-based strategies apply a function to candidate items and select the item for replacement based on its value. Randomized strategies are those that are nondeterministic, and these can be fully random.

Ghosh et al. [5] presented a very detailed survey on different Web caching and prefetching policies towardsWeb latency reduction. They also studied various Web cache replacement policies influenced by the factors recency, frequency, size,cost of fetching object and access latency of object. They also concluded that Hit Ratio (HR), Byte Hit Ratio (BHR) and Latency Saving Ratio (LSR) are the most widely used metrics in evaluating the performance of Web Caching.

Martin et al. [6,7] used trace-driven simulations to assess the performance of different cache replacement policies for proxy caches and utilized a trace of client requests to Web proxy in an ISP environment to assess the performance of several existing replacement policies. The results in this paper are based on the most extensive Web proxy workload characterization yet reported in the literature.

Williams et al. [8] discussed that SIZE outperforms than LFU, LRU and several LRU variations in terms of different performance measures; cache hit ratio and byte hit ratio. In their experiments, they fail to consider object frequency in decision making process.

Rachid et al. [9] proposed a strategy called class-based LRU. C-LRU works as recency-based as well as size-based, aiming to obtain a well-balanced mixture between large and small documents in the cache, and hence, good performance for both small and large objects requests. The caching strategy class-based LRU is a modification of standard LRU.

Triantafillou et al. [10] employ CSP (Cost Size Popularity) cache replacement algorithm which utilizes the communication cost to fetch Web objects, object's sizes, their popularities, an auxiliary cache and a cache admission control algorithm. They conclude that LRU is preferable to CSP for important parameter values, accounting for the objects' sizes does not improve latency and/or bandwidth requirements, and the collaboration of nearby proxies is not very beneficial.

Kumar et al. [11] proposed a new proxy-level Web caching mechanism that takes into account aggregate patterns observed in user object requests. The proposed integrated caching mechanism consists of a quasi-static portion that exploits historical request patterns, as well as a dynamic portion that handles deviations from normal usage patterns.

Rassul et al. [12] present two modified LRU algorithms and compare their performance with the LRU. Their results indicate that the performance of the LRU algorithm can be improved substantially with very simple modifications.

Kaya et al. [13] evaluated an admission-control (screening) policy for proxy server caching which uses the LRU (Least Recently Used) algorithm. The results obtained are useful for operating a proxy server deployed by an Internet Service Provider or for an enterprise (forward) proxy server through which employees browse the Internet. The admission-control policy classifies documents as cacheable and non-cacheable based on loading times and then uses LRU to operate the cache. The mathematical analysis of the admission control approach is particularly challenging because it considers the dynamics of the caching policy (LRU) operating at the proxy.

Luo et al. [14] focused on making proxy caching work for database-backed Web sites. There are several decisions that must be made when designing a Web proxy cache, such as cache placement and replacement. Many proposals can be found in the literature regarding these Web cache decision processes. Houtzager et al. [15] proposed an evolutionary approach to find an optimal solution to the Web proxy cache placement problem, while Aguilar and Leis [16] addressed the replacement problemwith cache coherency of proxy caching.

Radhika et al. [17] proposed an efficient Webcache replacement policy which uses frequency and recency of references to an object to calculate and associate a benefit value (bValue) with the object; an object with the lowest bValue is chosen for eviction. The policy is simple to implement than other policies and does not require any parameter tuning. But their result does not show much improved than Least Recently Used (LRU) and Least Unified Value (LUV) algorithms in terms of HR, BHR and DSR. Also their proposed replacement policy does not consider frequency of access of Web objects as well.

Jiang et al. [18]proposed a novel replacement algorithm Low Inter-reference Recency Set (LIRS) which effectively addresses the limitations of LRU by using recency to evaluate Inter-Reference Recency (IRR) of accessed blocks for making a replacement decision.

Later Zhanshenget al. [19]proposed a novel replacement policy that switches between LRU and LFU on runtime. The implementation of this scheme is complex which negatively affectsthe performance.

Dongheelee et al.[20]proposed a replacement policy called Least Recently/Frequently Used(LRFU) which subsumes both the LRU and LFU, and provides a spectrum of block replacement policies between them according to how much more they weigh the recent history than the older history.

Adwan et al.[21] proposed combined approach of LRU+5LFU in which weight of object play an important role but calculating weight of an object is a complex task.

As it can be observed, many Web cache replacement policies have been proposed for improving performance of Web caching. However, it is difficult to have an omnipotent policy that performs well in all environments or for all time because each policy has different design rational to optimize different resources. Moreover, combination of the factors that can influence the replacement process to get wise replacement decision is not an easy task because one factor in a particular situation or environment is more important than other environments. Hence, there is a need for a combined approach to efficiently manage the Web cache which satisfies the objectives of Web caching requirement. This is the motivation in adopting our Novel HybridCache Replacement Policyin solving Web caching problem.

III. PROPOSEDNOVELHYBRIDCACHEREPLACEMENTPOLICY

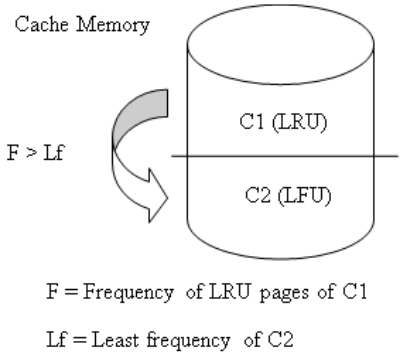

Fig.2: Cache division of the Proposed HybridReplacement Policy

C1=first logical section of particular size of the cache
C2=second logical section of particular size of the cache

```
1. Begin
2.      If (PageRequest is private)
3.              Send PageRequest to original server
4.      Else
5.      While (hash_value of cache! = NULL)
6.      Do
7.              If (hash_value == URL_hash)
8.                      Send page to client browser and
                        place it at the top of the cache
 9.             If (no match found)
10.                     Send request to original server
11.     If (c1! =FULL)
12.             Shift every item and place therecent page on top of the cache
13.     Else
14.             If (frequency of lower page in c1 > smallest frequency of c2)
15.                     Free the page from c2 having lowest frequency and store the page c1 to that space
16.             Else
17.                     Free the page coming from c1 to c2
18. End
```

We divide the cache memory as shown in Fig.2 into two logical halves, the first half (C1)is using LRU and the second half (C2) is using LFU as replacement policybecause we always access recently used pages and then after we calculate their times of occurrences. So, LRU policy will be used in first half (C1) and second

half(C2) will be using LFU. The criteria for Web pages to migrate from first half (C1) to second (C2) is the frequency of the last element in first half is more than the page having least frequency in second half otherwise the page from the first half will be removed from the cache.

IV. PROPOSEDSYSTEMMODEL IMPLEMENTINGTHENOVELHYBRID REPLACEMENT POLICY

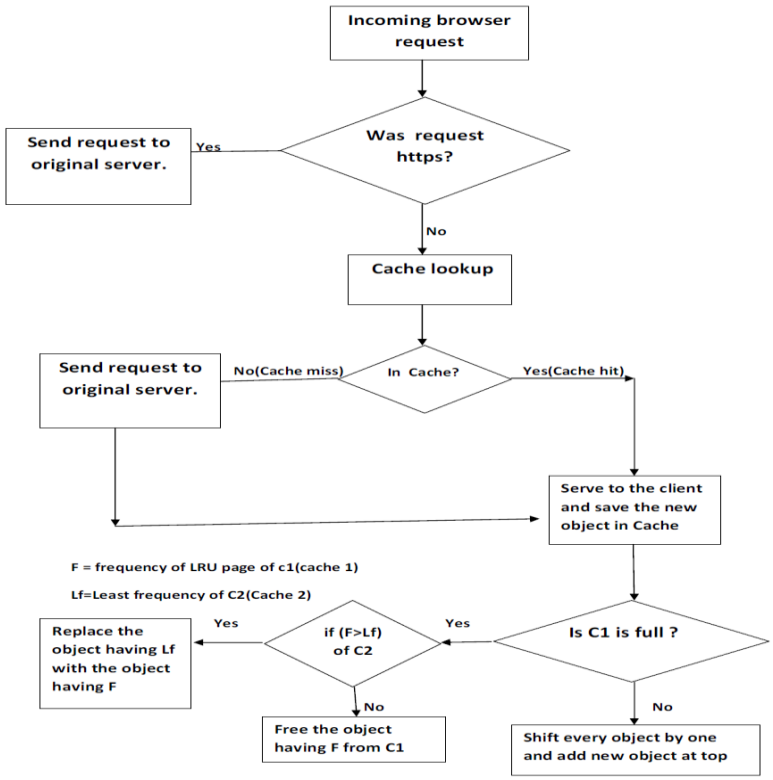

Fig.3: Proposed Hybrid System Model

The proposed Caching System Modelimplementing the Novel Hybrid Replacement Policy is shown in Fig.3where we are tracking both recently accessed and frequently accessedWeb pages. We have used MD-5 Hashing algorithm for securing URLs in the cache so that any user could not know the URLs stored in the cache. Our first work is of securing URLs in the cache from other users so that what pages users are requesting no one knows. MD-5 is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. An MD-5 hash is typically expressed as a hexadecimal number, 32 digits long. As the user makes the request, URL will pass through MD-5 hashing algorithm. After getting 32 digits long hash value, Cache lookup will perform and if a match occur it will be calculated as a cache hit and the page will be send to the requested client. If no match found miss occur and the request will be sent to the original server. After getting response from original server, cache need to store the page, so URL hash value and the content of requested page will be stored in the cache.We have also removed redundancy from cache so as to protect it from flooding attack.

V. EXPERIMENTALRESULTSANDDISCUSSIONS

This section presents the experimental results on the performance of the proposed Novel Hybrid Cache replacement policy. The dataset for testing our proposed policy is obtained from an Internet café containing 10 computers within BIT campus which is extremely popular among students, faculty members and staffs. We have executed our program code written in C programming language with the obtained dataset. We have constructed a trace-driven simulation to study our proposed Hybrid policy using a set of Internet users' traces. In our experiment the traces refer to the period from 01/March/2013:09:20:04 to 01/March/2013:21:05:02, Monday of one busy working day. The trace is composed of 1,165,845 Web requests with total of 72 users on that day. The simulations were performed at different network loads.

In order to evaluate our proposed Hybrid policy, we use two performance metrics Hit Ratio (HR) and Latency Saving Ratio (LSR) which are the most widely used metrics in evaluating the performance of

Webcaching. HR is defined as the percentage of requests that can be satisfied by the cache. LSR is defined as the ratio of the sum of download time of objects satisfied by the cache over the sum of all downloading time.
Let N be the total number of requests (objects).
δ_i= 1, if the request i is in the cache, while
δ_i= 0, otherwise.
Mathematically, this can be expressed as follows:

$HR = \frac{\sum_{i=1}^{N} \delta i}{N}$ (i)

$LSR = \frac{\sum_{i=1}^{N} ti\delta i}{\sum_{i=1}^{N} ti}$ (ii), where $t_i$is the time to download the i^{th}referenced object from server to the cache.

A higher HR indicates the user's satisfaction and defines an increased user servicing. On the other hand, a higher LSR improve the network performance and reduce the user-perceived latency (i.e. bandwidth savings, low congestion etc.).

We keep track of the HTTP requests made by each computer and we find total number of hits in all three policies (LRU, LFU, combined LRU and LFU) with 512 MB of proxy cache. To evaluate the effectiveness of theproposed policy, we have executedour program code in a system equipped with 32-bit Intel Core 2 Duo 3 GHz processor, 6M cache, 4 cores with 4 GB of RAM with theproxy server dataset obtained from the cyber café. The results we have got after implementation are better than LRU and LFU used individually in terms of Hit ratio and Latency saving ratio. It was the closest to optimum theoretical.

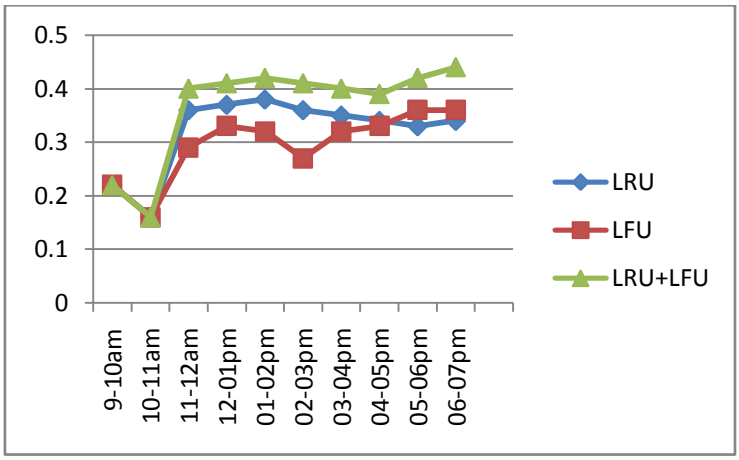

Fig.4: Hit ratio (HR) calculated with all three policies

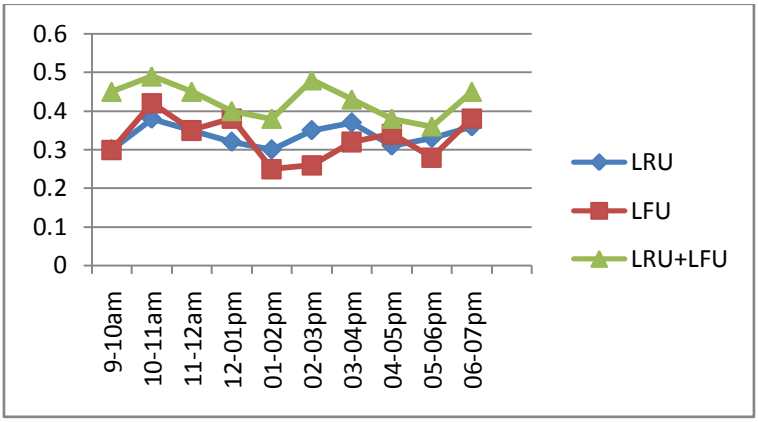

Fig.5: Latency saving ratio (LSR) calculated with all three policies

We have shown in Fig. 4 and 5 the results of our Hybrid policy compared to two other policies in terms of Hit ratio and Latency saving ratioas performance evaluation parameters.As time goes by in a busy working day, our proposed Hybrid policy outperforms in comparisonwith other two policies individually in case of both same and large size caches.

Further to elaborate with respect to HR, we observed that as the time is increasing in the busy day, the average Hit ratio of LRU is 0.32, LFU is 0.29 and that of our combined approach is 0.37. Concerning the LSR,

the average Latencysaving ratio for LRU is 0.34, LFU is having 0.32 and our combined approach is having 0.42. Now if we look into the data we find that average increase in Hit ratio of our policy to other two policies is increased by 19% and latency saving ratio is increased by 27%.

VI. CONCLUSIONAND FUTURESCOPE

In recent years, Web caching has been an active area of research among the research community due to its several advantages. As the number of internet users increased exponentially, the problems of Web traffic, increasing bandwidth demand and server overloading became more and more serious. Many efforts have been made in this direction to improve the Web performance, such as reducing Web latency and alleviating server bottleneck. But the existing solutions are not adequate to tackle this problem. Hence motivated by this, we have proposeda Novel Hybrid policy for Web proxy caching by combining LRU and LFU in an efficient manner to reduce the network latency in accessing particular Web services. The performance of the cache is improved, hence the overall network traffic, user-perceived latency, network congestion and traffic will minimize.Increasing the size of the cache could improve the hit ratio, but after certain value it is not improved much. To accommodate more Web documents in cache, increasing the cache size beyond the optimal value may cause increased latency of searching a Web documents in the cache. Hence average cache size is more desirable. Also highly dynamic document need not be kept in the cache, because those documents are modified frequently. This work may be extended in future by considering dynamism of the Web content.

Today the available caching techniques are usually designed keeping in mind specific types of Web site, thus the need of some standard benchmark cannot be overruled. In all caching schemes content replication, update propagation, and consistency management are always a concern to the management. Further there is need to develop a feedback mechanism that should be used to tune the page pre-generation process to match the current system load. Designing a set of benchmark on the set of these metrics will help the designers to decide which caching technique can be most suitable for them. Most often a combination of one or more caching solutions is used for a site. Dividing the Website in components and using suitable caching technique for each component should be an ideal solution. The optimal replacement policy aims to make the best use of available Cache space, to improve Cache hit ratios, and to reduce loads on the origin Server. Further, we must resort to an approach, which can predict the future users' requests and retain in Cache the most valuable objects. The performance of the Web caching will be significantly improved by integrating Web Prefetching technique which refers to the process of deducing clients' future requests for Web objects and getting those objects into the cache, in the background, before an explicit request is made for them. Our future research work is in progress towards this direction. The research frontier in Web performance improvement lies in developing efficient, scalable, robust, adaptive, stable Web caching scheme that can be easily deployed in current and future Web infrastructure.

ACKNOWLEDGMENTS

Our thanks to Biresh and Robin of the cyber café named "Technosoft" of BIT who helps towards providing proxy server logs of a busy working day.

REFERENCES

[1]. SarinaSulaiman, S.M.Shamsuddin, A. Abraham, S. Sulaiman, *"Web Caching and Prefetching: What, Why, and How?",* IEEE, 2008.

[2]. A.K.Y. Wong, *"Web Cache Replacement Policies: A Pragmatic Approach"*, IEEE Network magazine, 20(1), (2006), pp.28–34.

[3]. A. Balamash, M.Krunz, *"An Overview of Web Caching Replacement Algorithms"*, IEEE Communications Surveys & Tutorials, Second Quarter, 2004.

[4]. Podlipnig, S. and Böszörmenyi, L.,*"A Survey of Web Cache Replacement Strategies"*, ACM Computing Surveys, Vol. 35, Ner 4, pp. 374-398, 2003.

[5]. SirshenduSekharGhosh and Dr. Aruna Jain, *"Web Latency Reduction Techniques: A Review Paper"*, IJITNA vol.1 No.2 pp 20 – 26, September, 2011.

[6]. Martin F.Arlitt and Carey L. Williamson,*"Trace-Driven Simulation of Document Caching Strategies for Internet Web Servers Simulation"*, vol.68, Jan. 1997, pp.23-33.

[7]. Martin F. Arlitt, LudmilaCherkasova, John Dilley, Rich Friedrich, Tai Jin, *"Evaluating content management techniques for Web proxy caches"*, SIGMETRICS Performance Evaluation Review 27(4): 3-11 (2000).

[8]. S.Williams, M.Abrams, C.R. Standbridge, G.Abdulla and E.A.Fox, *"Removal Policies in Network Caches for World-Wide Web Documents"*, Proceedings of the ACM Sigcomm96, August, 1996, Stanford University.

[9]. Boudewijn R. Haverkort, Rachid El AbdouniKhayari, RaminSadre, *"A Class-Based Least Recently Used Caching Algorithm for World-Wide Web Proxies"*, Computer Performance Evaluation / TOOLS 2003: 273- 290.

[10]. P. Triantafillou and I. Aekaterinidis, *"Web Proxy Cache Replacement: Do's, Don'ts, and Expectations"*, Proc. of the second IEEE Int. Symposium on Network Computing and Applications (NCA'03), 2003.

[11]. Chetan Kumar, John B. Norris, *"A new approach for proxy level Web caching, Decision support Systems"*, 46:1, 52-60, 2008.

[12]. Rassul A, Y.M. Teo and Y.S. Ng, *"Cache Pollution in Web Proxy Servers"*, IEEE, 2003.

[13]. Cuneyd C. Kaya, Guoying Zhang, Yong Tan, Vijay S. Mookerjee, *"An admission-control technique for delay reduction in proxy caching"*, Decision Control Systems, 46: 594-603, 2009.

[14]. Luo et al., *"Form-based proxy caching for databasebackedWeb sites: keywords and functions"*, The VLDB Journal, the International Journal on Very Large Data Bases, Volume 17, number 3, pp. 489-513, 2008.

[15]. G Houtzager, C Jacob, C Williamson, *"An Evolutionary Approach to Optimal Web Proxy Cache Placement"*, IEEE Congress on Evolutionary Computation, 2006.

[16]. Aguilar J. and Leis E.L., *"A coherence-replacement protocol for Web proxy cache systems"*, International Journal of Computers and Applications, Volume 28, Issue 1, pp.12-18 (January 2006).

[17]. A. RadhikaSarma and R. Govindarajan,*" An Efficient Web Cache Replacement Policy"* In the Proc. of the 9th Intl. Symp. on High Performance Computing (HiPC-03), Hyderabad, India, Dec. 2003.

[18]. S. Jiang and X. Zhang, *"Making LRU Friendly to Weak Locality Workloads: A Novel Replacement Algorithm to Improve Buffer Cache Performance"*, IEEE Trans. Comput., vol. 54, 2005.

[19]. Zhansheng L., Dawei L., and Huijuan B., *"CRFP: A Novel Adaptive Replacement Policy Combined the LRU and LFU Policies"*, in Proceedings of IEEE 8th International Conference on Computer and Information Technology Workshops, Sydney, pp. 72-79, 2008.

[20]. Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, Chong Sang Kim, *"LRFU (Least Recently/Frequently Used) Replacement Policy: A Spectrum of Block Replacement Policies"*, March 1996.

[21]. Adwan Abdel Fattah, Aiman Abu Samra, *"Least Recently Plus Five Least Frequently Replacement Policy (LR+5LF)"*, The International Arab Journal of Information Technology, Vol. 9, No. 1, January 2012.

[22]. K.Geetha, N.AmmasaiGounden, Monikandan S, *"SEMALRU: An Implementation of modified web cache replacement algorithm"*, World Congress on Nature & Biologically Inspired Computing, (NaBIC 2009).

[23]. DeeptiMehrotra, RenukaNagpal, Pradeep Bhatia, *"Designing Metrics for Caching Techniques for Dynamic Web Site"*, Int'l Conf. on Computer & Communication Technology (ICCCT'10).

[24]. Nagaraj, S.V., Web Caching and its Implications, Kluwer Academic Publishers, 2004.

[25]. Brian D. Davison, AWeb Caching Primer, IEEE. Reprinted from IEEE Internet Computing, Volume 5, Number 4, July/August 2001.

[26]. L. D. Wessels. "Web Caching". USA: O'Reilly. 2001.