International Journal of Computer Science & Applications
Vol. III, No. I, pp. 78 - 91

ENHANCING QoS IN WEB CACHING USING DIFFERENTIATED SERVICES

P.Venketesh[1], S.N. Sivanandam[2], S.Manigandan[3]

1. Research Scholar, 2. Professor and Head, 3. Research Scholar

Department of Computer Science and Engineering,
PSG College of Technology, Coimbatore, India

Email: venkateshp@cse.psgtech.ac.in, venkip_ms@yahoo.co.in

ABSTRACT

Web caching and Content Distribution Networks (CDNs) occupy strategic positions in the growth of Internet by effectively delivering data to the end users with reduced access latency. Due to diverse nature of applications, Quality of Service (QoS) is considered an important aspect in web caching. To achieve QoS in Caching scenario web requests are classified into different classes with each class holding different priority levels. In this paper, we show how differentiated strategies combined with dynamic memory allocation based on relative hit rate can provide improved performance for high priority classes with little or no performance degradation for low priority classes. We have developed and evaluated a model that classifies web requests into two classes (Premium, Best-Effort) based on object size that uses LRU and LFU-DA as replacement algorithms. The proposed model efficiently converge the hit rate of classes towards its specified desired hit rate with minimum overhead.

Key Words:

Web Cache, CDN, QoS, replacement algorithm, differentiated services

1. INTRODUCTION

Internet and World Wide Web have seen exponential growth over the years and continue to be the dominant source in providing information to the users. Proxies are deployed in between clients and server to reduce network traffic and improve user response time. They can be found in ISP network backbone nodes intercepting all web traffic, called as *forward proxies*; CDN (Content Distribution Networks) also deploys proxies, called as *Surrogate Servers*. Quality of Service (QoS) is considered an important

aspect in web caching. To achieve QoS better management of storage space is necessary and this can be achieved by considering differentiated strategy. Service differentiation improves hit rate of high priority classes providing both technical and economic gains.

In this paper, we have designed and implemented a differentiated service architecture that achieves proportional relative hit rate performance among content classes. Per class caching queue is maintained for better allocation/ deallocation of memory. We demonstrate how our model achieves service differentiation with improved hit rate considering separate replacement algorithm for each class based on its requirement specifications. Service differentiation can be applied effectively to both forward and reverse proxies.

The rest of the paper is organized as follows. Section 2 provides an overview of caching policies. Section 3 discusses the related work in differentiated services for web caching. Section 4 describes the architecture that implements differentiated service mechanism with dynamic memory allocation / deallocation for various classes. Section 5 evaluates the architecture and provides experimental results indicating performance achieved. Section 6 concludes the paper.

2. OVERVIEW OF CACHING POLICIES

The performance of caching policy is highly influenced by the access pattern of workload. Some of the workload properties observed by researchers [1-5] are:

- 60 -70 % of objects will not be accessed again.
- Most accessed objects are small; it is heavy tailed size distribution.
- Images are the most accessed file type followed by HTML.
- Multimedia objects account for only few accesses
- Number of access to dynamically generated objects is minimal (less than 20%)
- Web accesses exhibit properties of temporal and spatial locality

Cache replacement policy and the offered workload are responsible for determining efficient resource utilization in cache. Selection of object for removal from the cache is based on the metrics: recency, frequency and object size. The Least Recently Used (LRU) policy removes the object that has not been accessed for the longest time. The Least Frequently Used (LFU) policy maintains a reference counter for each object

and evicts the objects with the lowest reference count. The LFU-Aging policy considers both access frequency and age of object in cache. LFU-DA (LFU with Dynamic Aging) is based on LFU-Aging and its dynamic aging mechanism does not require parameterization. The Greedy Dual-Size [6] policy takes into account size and cost function for retrieving objects. It replaces the smallest key calculated by the function K (p) = C (p) / S (p) +L, where C (p) is retrieval cost, S (p) is object size, L is running age factor. GDS-F [7] is refinement of GDS policy and it includes frequency as parameter in key calculation, K (p) = C (p) * F (p) / S (p) +L.

The Lowest Relative Value [8] policy replaces the object with the lowest relative value that is calculated using access time, frequency and size information. LRU-K [9] is self-reliant page replacement algorithm and it considers recency and frequency information when making replacement decisions. Service differentiation is not provided by any of the replacement policies mentioned above, which is considered important for content publishers and service providers.

3. RELATED WORK

Service Differentiation can be applied to both Content Distribution (CDN) Servers and Proxy Caches. It achieves both technical and economic gains to content publishers and service providers. Initially it has been used extensively for networking domain [10, 11] and more recently, it is also applied efficiently to web server domain [12, 13]. To support differentiated services in web caching, requests are classified into different classes with each class holding their own queue. The classification is based on the mechanisms employed by web server QoS Schemes [14]. The objectives of resource allocation and different service differentiation can be achieved by employing different classification policies. Classification policies used to implement service differentiation can be broadly categorized as [15]:

- *Source - based*: Source hostname / IP Address
- *Content - based:* File type (HTML, Images, Video etc)
- *Economics –based*: Payment by object owner
- *Popularity –based*: Object Popularity
- *Size- based*: Object size

Various factors favor usage of service (or) performance differentiation to increase the efficiency of caching. From End-Users perspective, preference can be given to regular web content over wireless content, where users consider access speed as prominent factor. From Content Centric perspective, download latency time is considered important, so HTML requests will be classified as premium class and embedded objects will be treated as best effort.

Differential QoS in proxy cache space allocation is considered an important aspect and many researchers have previously focused in achieving the desired QoS. *T.P.Kelly et al* [16] proposed a simple weighted replacement policy that provides differential QoS in caching. The proposed method does not guarantee differentiated services due to lack of adaptive mechanism. *Lu et al* [17, 18] proposed an adaptive control scheme based on approximate linear difference equation models, which achieves QoS with a self-tuning performance regulator. The scheme suffers from drawbacks, such as complexity in adaptive control, maintenance of single queue for all class objects, impractical design for more than two classes. *Dong Zheng* [19] implemented an adaptive model to provide differentiated services among classes using webLRU-2 algorithm as replacement policy. The model uses one queue per class for better allocation / deallocation decisions. The adaptor of the model periodically monitors the output performance of each class, updates the class weights accordingly, and then decides upon the space allocation needed to reach target relative hit ratio.

4. DIFFERENTIATED MODEL

4.1 Architecture

To provide service differentiation among multiple classes of content cached in proxy cache, we proposed architecture as shown in Fig.1 with the following components: *Classifier, Cache Server, Output Monitor, Adaptor* and *Space allocation/ deallocation* module. In our model, we classify web requests into different classes based on object size. Separate queue is maintained for each class to achieve efficient space allocation/ deallocation. Cache Replacement policy is used for object removal whenever cache space is full or object is not used for long duration. To enhance the performance, our architecture employs separate replacement policy for each class. The main goal of differentiated services is to ensure that relative hit rate of each class approaches towards

its desired relative hit rate in an efficient manner, thus providing improved performance for high priority classes.

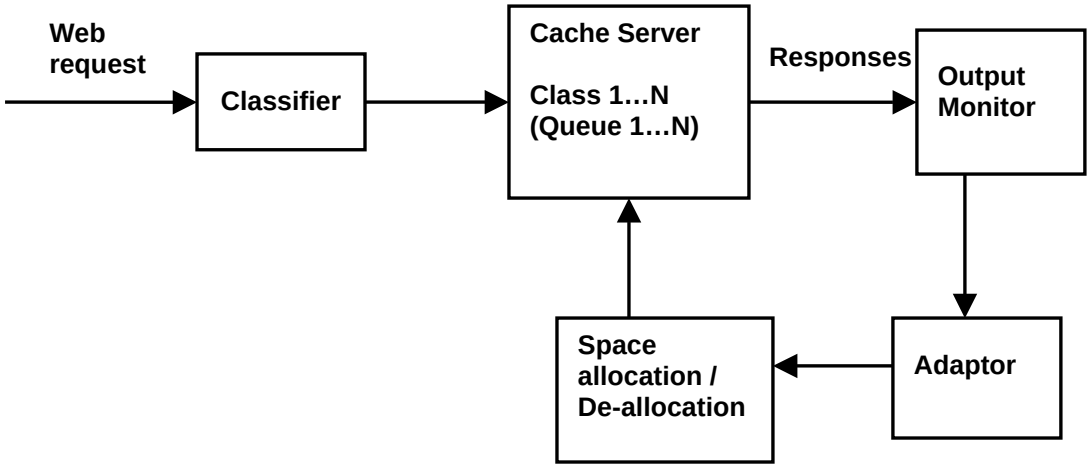


Fig 1: Differentiated Service Model

Classifier

It is responsible for classifying requests (based on Classification policies) into various classes and assigning priority levels to them based on its QoS requirements. Different Priority levels are: Premium, Best-Effort, Assured, others. Depending on class priority, the classifier will select appropriate replacement algorithm that improves hit rate or byte hit rate towards their desired value.

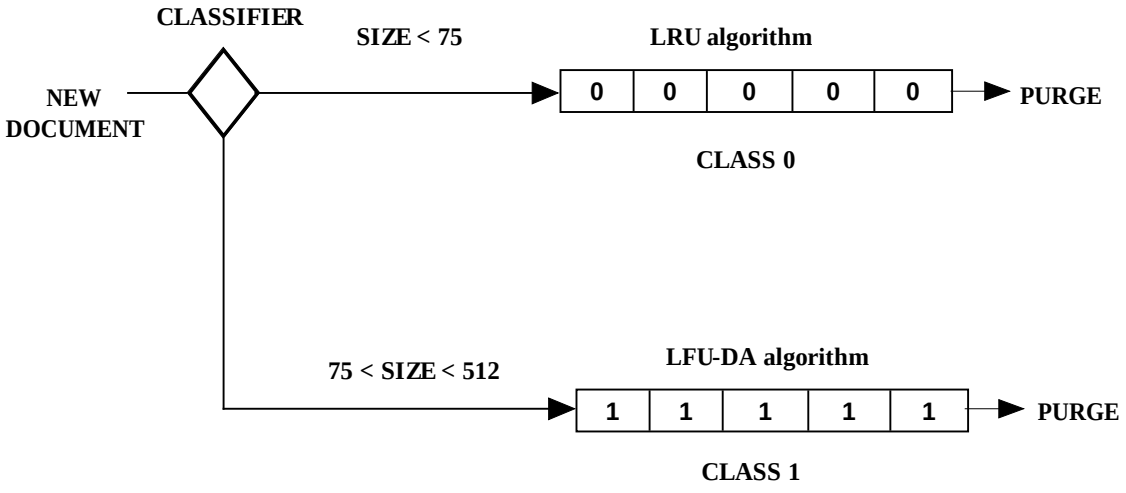


Fig. 2: Object Classification Model

In our model, requests having size less than 75 KB are classified as class 0 (Premium) and those requests having size between 75 & 512KB are classified as class 1 (Best-Effort). The implementation uses LRU for class 0 and LFU-DA for class 1. Since most objects accessed in web are of small size, class 0 is assigned the Premium class.

Output Monitor

It is responsible for measuring the relative hit rate achieved for each class at specified intervals. This value is used to determine the deviation from desired relative hit rate. Adjustment to cache space is performed only when the deviation is equal to or above some desired deviation threshold in order to avoid unnecessary fluctuation in space allocation.

Adaptor

The module is responsible for performing both allocation and deallocation of cache space based on the measured relative hit rate. Cache space deallocation is possible if a particular class relative hit rate is greater than desired relative hit rate or under utilization of cache space. Space adjustment is performed in such a way that each class achieves a smooth convergence towards its desired relative hit rate.

QoS Cache Server

It is responsible for satisfying user requests and improving the hit rate based on its service agreements. Cache server maintains separate queue for each class, with each queue holding their own replacement policy. Cache Server is analyzed by splitting requests into two classes and assigning replacement policies LRU and LFU-DA.

4.2 Working Principle

Model variables have the following Meaning:

$\mathbf{R_i}$ = Measured relative hit rate for each $class_i$ taken at regular intervals

$\mathbf{D_i}$ = Desired relative hit rate (fixed) for each $class_i$

$\mathbf{D\_Space_i}$ = Desired Cache Space assigned for each $class_i$ based on its hit rate.

$\mathbf{C\_Space_i}$ = Counter indicating actual amount of cache space used by $class_i$

(i= 1 to N where N denotes number of classes)

Desired relative hit rate for each class is assigned based on service differentiation policy. Our architecture assigns value based on previous measurements taken without differentiation criteria. The desired relative hit rate of all classes is normalized such that it equals 1.

Relative hit rate is measured as:

$$R_i = \frac{H_i}{H_1 + H_2 + \text{-----} + H_n} \quad (1)$$

$$D_i = \frac{C_i}{C_1 + C_2 + \text{-----} + C_n} \quad (2)$$

Where H_i denotes hit rate of each $class_i$ and C_i denotes constant factor representing QoS for $class_i$.

Difference between measured relative hit rate and desired relative hit rate is used to adjust the space allocated to particular class. Let the difference value be DV_i $(D_i - R_i)$. Cache space allocation will be performed if $R_i < D_i$ and $C\_Space_i >= D\_Space_i$. Allocation of storage space will not be performed if $C\_Space_i < D\_Space_i$, because that particular class has not received enough requests to fill the allocated cache space, so further allocation will not improve its hit rate.

Cache space deallocation is possible if a particular class relative hit rate is greater than desired relative hit rate or cache space is under utilized compared to its assigned space. When a requeste d web page is not available in proxy it will be downloaded from origin server. In such case if C_Space_i is greater than D_Space_i before space adjustment, then appropriate replacement algorithm will be used to purge an object of particular class to store the newly requested page on the proxy server.

Let S be the total cache space available. Based on number of classes the space will be equally assigned to them. In our model, we classified requests into two classes (Class 0 and Class 1), where Class 0 cache space is denoted as D_Space_0, Class 1 cache space is denoted as D_Space_1. The value of D_Space_0 and D_Space_1 will be assigned in such a way that it does not exceed the total space available.

$$D\_Space_0 = (S * D_0) /100 \quad (3)$$

$$D\_Space_1 = (S - D\_Space_0) \quad (4)$$

In Equation (3), S is total cache space (size of 50MB Considered); D_0 is desired relative hit rate for class 0. The equation takes value 100 since cache space considered is in megabytes and relative hit rate is represented in percentage.

At periodic intervals, compute the value of R_i and compare it with D_i. Based on the difference value DV_i ($D_i - R_i$) obtained, adjust the cache space of corresponding class dynamically so that hit rate will improve in the next time period; i.e. modify D_Space_i of each class. Space adjustment will be carried out only when the difference DV_i is equal or above some deviation threshold d in order to avoid unnecessary fluctuation in cache space. The proposed model takes the value of d as five percent. If $DV_i >= d$ then space adjustment is carried out.

At time interval t, modification done to cache space is

$$D\_Space_i(t) = D\_Space_i(t-1) + DV_i \quad (5)$$

when $C\_Space_i \sim D\_Space_i$

Modification to cache space of particular class will be carried out only when counter value (C_Space_i) is nearly equal to the desired cache space (D_Space_i). Else, space allocation to that class will lead to wastage and does not improve the hit rate. The value of DV_i can be either positive or negative, depending on which desire d cache space is adjusted.

If $C\_Space_i < D\_Space_i$ of particular class (i.e. difference is more than 5 %), then it can be considered for deallocation and that space can be given to other high priority class. The system ensures that total cache space will not be exceeded even though each class adjusts its space independently based on its difference value DV_i.

The selection of cache replacement policy for each class is based on specifications like improving hit rate or byte hit rate, minimizing amount of data to be transferred between Origin Server and Proxy Server etc. Since class 0 (Premium) focuses on small sized objects (HTML documents), it will have high degree of temporal locality due to frequent requests for same object by the user. So, we considered LRU algorithm for class 0. In case of class 1 (Best effort), size of objects is large and it requires high byte hit rate.

Since LFU-DA will maximize byte hit rate it is assigned to class 1. Other replacement policies can be considered depending on the QoS specification expected by each class.

4.3 Implementation

To evaluate our differentiated model that provides QoS, the architecture has been implemented in widely used Squid Proxy Server. Squid is an open-source, high performance, Internet Proxy Cache [20] that services HTTP requests on behalf of clients. Squid maintains cache of documents that are requested to avoid refetching from the web server if another client makes the same request. Hit rate (H) is considered an important parameter in measuring the Cache efficiency.

Using original Squid configuration, Hit Rate is measured for different classes without varying the space allocation. Relative hit rate for different classes is assigned based on the initial measurement taken. The goal of proposed model is to achieve differential hit rate that is converging close to the desired relative value. The cache space is initially partitioned and assigned to different classes based on the desired relative hit rate. Depending on the measured relative hit rate the partition size assigned to each class may change. The model uses different cache replacement algorithm for each class based on its QoS requirements to improve its relative hit rate. Desired Relative Hit rate for class 0 is fixed as 0.65 and for class 1 it is fixed as 0.35. Premium class should have high hit rate compared to that of other classes. So its desired relative hit rate is taken such that it improves performance by slightly degrading the performance of other classes.

5. EVALUATION

The performance of differentiated caching architecture is tested using synthetic traces. To generate requests that reflect the characteristics of actual web proxy traces, we used a tool called SURGE (Scalable URL Reference Generator) [21]. The tool provides great flexibility in testing sensitivity of results by tuning the values of system parameters. The main advantages of Surge is that it generates web references matching empirical measurements of 1) server file size distribution 2) request size distribution 3) relative file popularity 4) embedded file references 5) temporal locality of reference 6) idle periods of individual users.

In our experiment, web requests are classified into two classes and parameters are set such that workload generated contains balanced number of requests for different classes in order to get accurate measurement (Default value =1000 requests). Timing

interval for measuring hit rate of each class is set to be 50 seconds. Cache space used for evaluation is 50MB; it is partitioned and assigned to different classes. In order to generate large number of requests to be sent to proxy server, SURGE is made to run on two client machines. Apache web server [22] is used to simulate the working of origin server. It satisfies web requests sent by proxy on behalf of client machines.

Initially the experiment is carried out without modifying the original cache server; i.e. cache space is not partitioned for different classes. Evaluation is carried out with different replacement policies (LRU, LFU-DA and GDSF). The average hit rate achieved without service differentiation is appreciable. Since premium class is not given priority, they do not converge well towards the desired value. Fig. 3a, 3b and 3c shows relative hit rate measured in original squid without service differentiation.

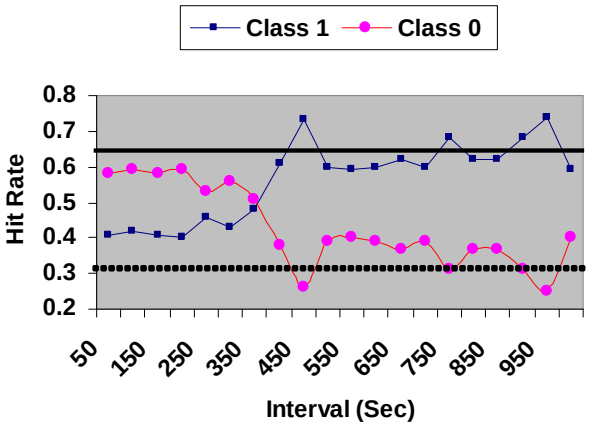


Fig 3-a: GDSF algorithm without QoS

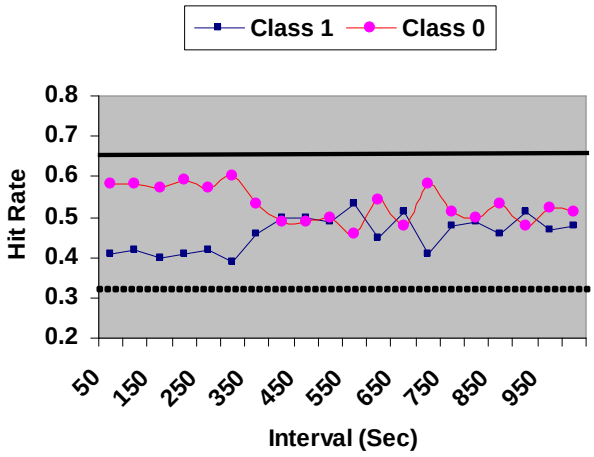
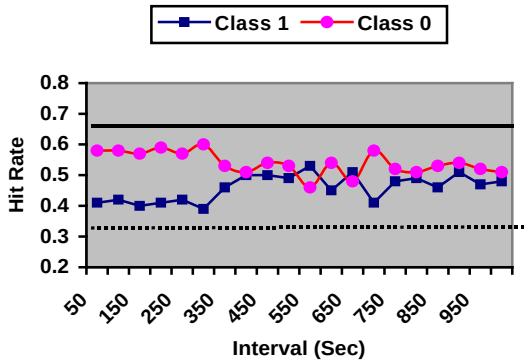


Fig 3-b: LFUDA algorithm without QoS

In all the three cases, average hit rate is reasonable. Since there is no differentiation, none has converged well towards its desired relative hit rate. The graphs indicate hit rate achieved by using single replacement policy for the entire cache space.



Desired Relative Hit Rate (Class 1) .....

Desired Relative Hit Rate (Class 0) _____

Fig 3-c: LRU algorithm without QoS

Usage of multiple replacement policy enhances the performance of all classes including that of premium class. Fig. 4 shows relative hit rate achieved by using separate replacement policy for each class without space adjustment. It achieves good average hit rate but still do not converge well towards desired relative hit rate because of static storage allocation.

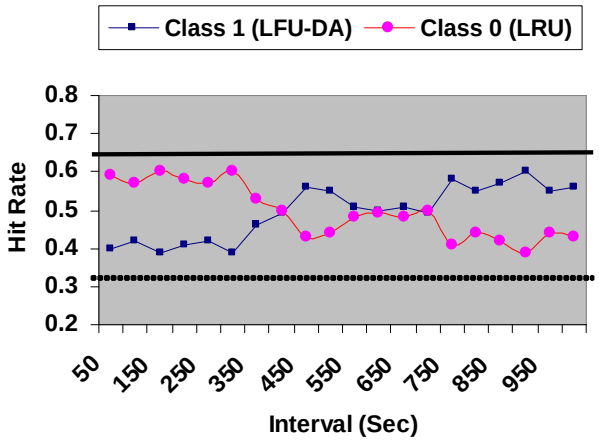


Fig 4: LRU and LFUDA without QoS

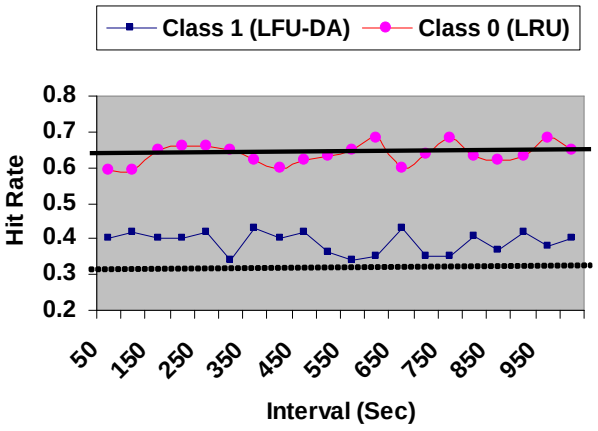


Fig 5: LRU and LFUDA with QoS

In Fig 5, we perform dynamic storage allocation / deallocation based on the hit rate measured during periodic intervals. The graph shows smooth convergence of hit rate towards its desired relative hit rate. It achieves efficient service differentiation with little overhead. The performance of best-effort class is least affected. The model shows good average hit rate that is closely equivalent to hit rate achieved without service differentiation.

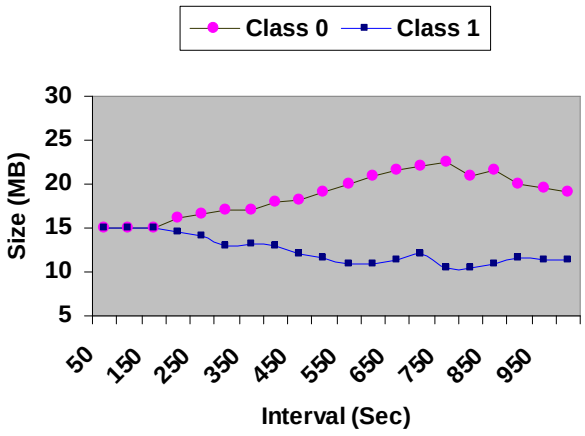


Fig 6: Space allocation for LRU and LFUDA with QoS

Dynamic adjustment of cache storage space to improve the hit rate of classes is shown in Fig.6. The graph shows smooth variation in space allocation, thus avoiding unnecessary fluctuation both in storage space usage and hit rate. Hit rate will increase logarithmically when cache size is increased [2]. Based on this notion, we have used only small cache size in testing our approach (Size = 50MB).

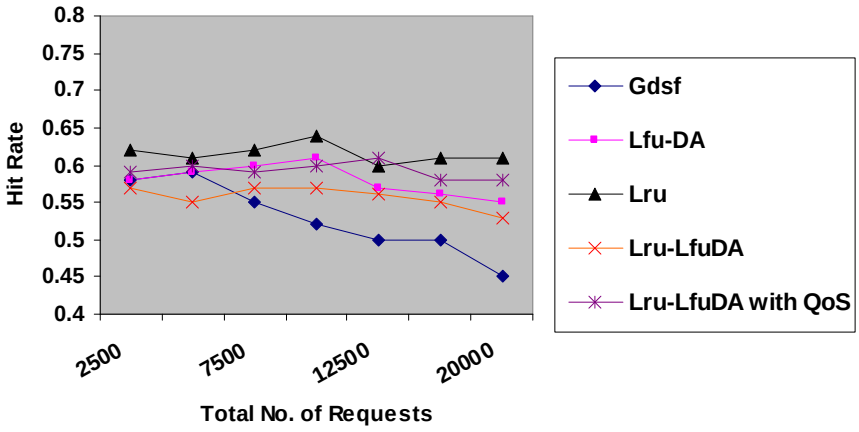


Fig. 7: Comparison of Replacement Policies

In fig. 7, we compare performance of QoS mechanism with that of other policies that does not provide differentiated services. The QoS Model using LRU-LFUDA shows improved performance that is equivalent /better than other policies.

6. CONCLUSION

Web caching and Content Distribution requires efficient method to provide QoS. The issue is addressed by employing an efficient mechanism called differentiated services. In this paper, we have explored the advantages of providing QoS using differentiated services and proposed a method that uses different replacement policy for web request classes. The proposed model proves to be efficient in achieving the desired relative hit rate with little implementation overhead. The model is tested by using two replacement algorithms (LRU, LFUDA). Based on QoS specification other algorithms can be selected for evaluation. The adaptive model can be extended for testing more classes with little overhead towards implementation and analysis. To assign desired relative hit rate an efficient mechanism needs to be explored for better flexibility when number of class increases. In case of more classes, the overhead in fixing the relative hit rate and managing separate queue for each class will increase thus degrading overall performance to a certain extent.

REFERENCES

[1] G. Abdulla, E. A. Fox, M. Abrams and S. Williams, "WWW proxy traffic Characterization with application to caching", *Technical Report*, Computer Science Department Virginia Technology, (CS-97-03):1--20, Mar 1998.

[2] V. Almeida, A. Bestavros, M. Crovella and A. Oliveira, "Characterizing reference locality in the WWW", In *IEEE International Conference in Parallel and Distributed Information Systems*, Miami Beach, FL, USA, Dec 1996.

[3] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications", World *Wide Web,* 2(1): 15-28, Jan 1999.

[4] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker, "Web caching and Zipf-like Distributions: evidence and implications", In *Proceedings of Infocom*, 1999.

[5] M. Rabinovich, O. Spatscheck, "Web Caching and Replication", *Addison Wesley*, Dec 2001

[6] P. Cao, J. Zhang and K. Beach, "Cost-aware WWW proxy caching algorithms", In *Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, 193-206, Dec 1997.

[7] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", *Technical Report* HPL-98-69R1, Hewlett-Packard Laboratories, Nov 1998

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

[9] E. J. O'Neil, P. E. O'Neil and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering", In *Proc. of ACM SIGMOD Int. Conference on Management of Data*, 1993

[10] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, 1994.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, 1998.

[12] T. F. Abdelzaher and N. Bhatti, "Web server QoS management by adaptive content delivery", *In International Workshop on Quality of Service*, London, UK, 1999.

[13] N. Vasiliou and H. L. Lutfiyya, "Providing a differentiated quality of service in a World Wide Web server", *Performance and Architecture of Web Servers Workshop*, Santa Clara, CA, 2000

[14] N. Bhatti and R. Friedrich, “Web server support for tiered services”, *IEEE Network*, 13(5): 64-71, Sep. 1999.

[15] M. Feldman and J. Chuang, “Service Differentiation in Web Caching and Content Distribution”, http://citeseer.nj.nec.com/554650.html.

[16] T. P. Kelly, Y. M. Chan, S. Jamin and J. K. MacKie-Mason, “Biased Replacement Policies for Web Caches: Differential Quality-of-Service and Aggregate User Value”, In *Proc. of the 4th International Web Caching Workshop,* San Diego, CA, Mar 1999

[17] Y. Lu, A. Saxena, and T. F. Abdelzaher, “Differentiated Caching Services: A Control-Theoretical Approach”, *in International Conference on Distributed Computing Systems*, Phoenix, Arizona, 2001

[18] Y. Lu, T. Abdelzaher, C. Lu, and G. Tao, “An Adaptive Control Framework for QoS Guarantees and its Application to Differentiated Caching Services”, In *10th IEEE International Workshop on Quality of Service*, 23-32. May 2002

[19] Dong Zheng, “Differentiated Web Caching – A Differentiated Memory Allocation Model on Proxies”, Thesis submitted to School of Computing, Queen’s University, Kingston, Ontario, Canada, 2004

[20] J. Dilley, M. Arlitt, and S. Perret, “Enhancement and validation of the squid cache replacement policy”, In *4th International Web Caching Workshop,* San Diego, CA, March 1999

[21] P. Barford and M. E. Crovella, “Generating representative web workloads for network and server performance evaluation”, In *Proceedings of Performance ’98/ACM SIGMETRICS ’98*, pages 151–160, Madison, WI, 1998.

[22] R. T. Fielding and G. Kaiser, “The apache http server project”, *IEEE-Internet-Computing*, 1(4): 88–90, July 1997.