ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

DYNASEM - An Improvised Dynamic and Semantic based Web Cache Replacement Policy

Geetha K *
Department of Computer Science and Engineering,
National Institute of Technology
Tiruchirappalli-15, Tamilnadu, India
geethavalavan@yahoo.com

Ammasi Gounden N
Department of Electrical and Electronics Engineering
National Institute of Technology ,
Tiruchirappalli-15, Tamilnadu, India
ammas@nitt.edu

Abstract: This paper proposes a web cache replacement policy based on semantic content of the pages cached at the client side. Two models namely Clustered Model(CM) and Relational Model(RM) are proposed that focus on the Dynamicity which refers the dynamic nature of the content and the Semantic content which exhibits the relation of information available among cached web pages and hence the name DynaSem. The proposed policy marks the page for eviction prioritized by Eviction Index (EI) in CM and Relation Index (RI) in RM. CM uses an interface with a web browser incorporated into it. The Trie data structure that enables the searching process to be more efficient has been framed to store the well-known categories of cached content as clusters. Pages with highest EI are marked for eviction. RM employs a technique to reveal the relation among cached documents. It evicts documents that are less related(minimum RI) to an incoming document which needs to be stored in the cache to ensure that only related documents are cached; hence the contents of the cache represent the documents of interest to the user and those which are of more static in nature. The proposed policy has been developed to incorporate two algorithms- one to find the dynamic count of the given web page 'P' and the other to the find semantic relation between the pages cached. Both the models(CM and RM) are used to establish the semantic relation. The policy has been simulated under model driven simulation with the help of an input set consisting of a few web pages. The parameters pertinent to cache replacement algorithms are computed and the result shows there is a factual improvement compared to the original semantic based policies.

Keywords: Web caching, replacement policies, eviction, semantic relation, dynamism.

I. INTRODUCTION

Web caching is the process of storing the frequently accessed web pages or documents. The increasing demand for web services insisted the need for web caching that can indubitably reduce the Internet traffic, download time, network bandwidth usage, server load, and perceived lag. Cache being a limited resource in terms of size, becomes saturated quite frequently and hence eviction has to be made often. Especially in wireless network, size of the client cache at mobile terminal is very small that demands frequent replacement. The state of art dictates multitude policies based on recency, frequency, size, and combination of the above parameters as some function. A cache server stores web objects (e.g., HTML pages, images, and files) locally for the use of future requests to those objects. As cache size is finite, a cache replacement policy is needed to manage cache content. If a cache is full when an object needs to be stored, the policy will determine which object is to be evicted to make room for the new object. However, in practical implementation, a replacement policy usually takes place before the cache is really full.

The cache uses two water marks, high and low, to guide the replacement process. If the size of total cached objects exceeds the high watermark, the policy will evict objects until the low watermark is reached. The advantage of doing this is reducing the overhead of invoking the policy on demand. The goal of the replacement policy is to make the best use of available resources, including disk space, processing power, server load, and network bandwidth. The increasing use of the internet and its emerging applications tend to saturate the internet resources. Caching mechanism helps to reduce the network and server load by avoiding the transmission of web pages and documents that are already requested. There are few key performance metrics used to evaluate the performance of replacement policy. There is no policy studied so far that can be optimal for all the metrics. The metric to be considered depends on the environment in which the policy is applied. Following are the definitions of some of the key metrics normally considered and the targeted environments.

A. ***Hit Rate (HR)*** is defined as the % of requests that can be satisfied by the cache. It is of interests to system with small cache size. (HR are similar when cache is very large).

B. ***Byte Hit Rate (BHR)*** is defined as the number of bytes satisfied by the cache as a function of total bytes requested by client. It is of particular interest to systems with limited external network bandwidth. Larger popular object leads to higher BHR and smaller popular objects leads to higher HR.

C. ***CPU Utilization*** analyses the time complexity of the replacement policy. This parameter can be ignored if CPU is not going to be the performance bottleneck. It is of particular interest to busy cache servers with limited processing ability.

D. ***Latency Reduction (LR)*** is defined as the percentage of the object download latency that can be reduced. The object that consumes highest latency can be cached to increase the latency reduction. It is of particular interest to the users who want to minimize retrieval time.

II. RELATED WORK

It is well known that that there are already sufficient number of good replacement policies available for usage [1,2,3,4,5]. Hence research on new policies is less important. Indeed, most of the policies in various proposals provide evidence that they perform better than others for some specific applications. This leads to a state of confusion as to which policy should be used. Actually, there is no single policy that performs best in all environments. This is because different policies have different design rationale and are designed to optimize different resources.

Table I. Summary of Different Replacement Policies

Policy	**Parameter**	**Criteria for retaining the pages**	**Context**
LRU (Recency Based)	Time	Last access time, average retrieval time, recently referenced page, expiry time, last modified time	When users are interested in same type of web pages at the same instance
LFU(Frequency based)	Number of previous accesses	Popularity in terms of frequency	For accessing stable and popular web pages
Size	Size	Small sized pages retained	Small sized pages retained
Function	Utility value	Time, frequency, size, etc.,	When swapping process leads to extra overhead
Random	Random seed	NIL	When the resources are limited and the process is less complex

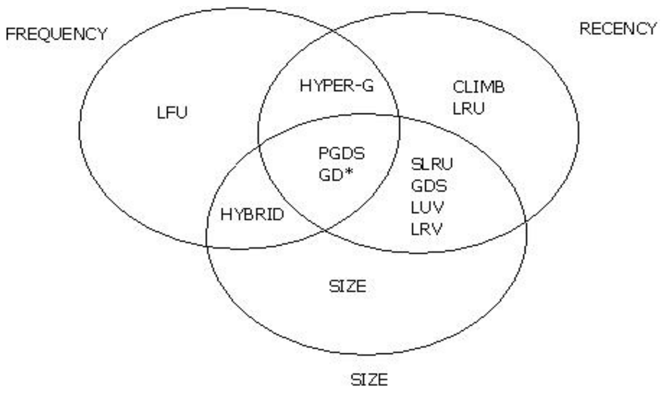

Figure.1. Classification of cache replacement policies.

Although Web cache replacement policies have been summarized in previous work, from small overviews [2,3] to more comprehensive surveys[4,5], many of them focus on discussing the operations of common policies and comparing their performance by running trace-driven simulations[6]. As a matter of fact, recently proposed solutions provide only slight improvement. Some policies[7] favor one resource at the expense of another. Among the proposed replacement policies the literature reports contradictory results. For example results in [7] reports LRU outperforms size interms of HR for few cache sizes, [8] show that the size based policy performs well and can achieve higher HR than LRU. Among LRU and LFU, [9] claims LRU is superior to LFU in terms of HR, whereas [10] shows LFU performs better than LRU in most cases. From the literature, it is understood that the performance of any policy depends highly on workload characteristics. One workload set may make a policy perform well and another may degrade the performance. Most of the existing replacement algorithms are listed in Table.1. It summarizes the kind of parameters that decide the replacement strategy, the criteria that decides the retention of cached pages, and the suitable application that can make use of the policy efficiently.

Apart from the above tabulated replacement policies, very few policies work on the semantic content of the web page[11,12,13,14]. All these policies assume that for a given time slot, the users' surfing pattern possesses a close semantic relation. Semantic relations between cached documents are considered for evicting the document for replacement [15]. Some attempts to create semantic caching mechanisms for database and query systems for well structured data are reported in [15]. They all claim that semantic based approach for replacement can perform better than traditional policies. The research work reported in [16] combines physical locality information with semantics which can produce good result in the field of wireless environment. A comprehensive list of the most relevant cache replacement policies based on the physical properties of objects is listed below:

A. *Recency Based Policies :*

- **a.** ***LRU (Least Recently Used*):** This policy removes the page that is least recently used by clients and is identified by the timestamps [17].
- ***b.*** ***RUMIN*** **:** This policy tends to keep in cache objects of smaller size to minimize the number of replacements. If a new object of size S has to be brought into the cache and if the cache is full then, among the objects with size at least S, the least recently used one is removed. Otherwise objects with size at least S/2 or S/4 and so on are removed until the required room is created [17]
- ***c.*** ***Pitkow/Recker*** *:* This policy removes the LRU, but if all the documents are accessed with same recency, then the largest one is removed.

B. *Frequency based Policies:*

This policy elects the page that is least frequently used by the clients and is implemented with many variations:

- **a.** ***In-Cache LFU:*** In this version, the access counter for an object is zeroed every time the object enters the cache.

b. ***Perfect LFU*:** In this version, the access counter for an object is zeroed only for the first time the object enters the cache. If the object that has been previously removed comes back to cache, it access the counter with the previous value used before removal.

c. ***Hyper-G:*** This is also a variation of LFU with last access time and size considerations.

C. Size Based Policies :

a. ***Size Adjustment LRU (SLRU):*** This is like an Knapsack problem. It sorts the objects based on cost and size and the largest index is evicted from the cache when a replacement is required[17].

b. ***Segmented Cache****:* This policy splits the cache into partitions based on the size of the objects and chooses the one for replacement that has high variability noticed in WWW [18].

c. ***Pyramidal Selection Scheme with Award:*** In this, the objects are classified based on the size using a log function combined with frequency access rate [19].

D. Function Based Policies :

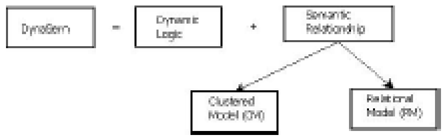

Figure.2. DynaSem Policy

These policies employ general functions for different factors such as last access, entry time of an object into the cache, transfer cost and time to live of an object [20].

a. ***Lowest Relative Value LRV :*** This policy assigns a relative cost for each object in the cache based on their utility value. The object with the Lowest Utility Value(LUV) is marked for removal.

b. ***Lowest Latency First:*** This policy removes the object that has lowest download latency first in order to minimize the object latency [20].

$$f = \frac{\left(C_s + \frac{W_b}{W_s}\right) \times (n_p)^{W_n}}{Z_p}$$

c. ***Hybrid* :** This policy aims at reducing the total latency. It replaces the objects which has the lowest value calculated using the function: where s is a server, p is an object located in s, Cs is the time to connect to s, bs is the bandwidth of the server s, n_p is the number of times p has been requested since it was brought into the cache, Zp is the size (in bytes) p, and W_b and Wn are constants.

Web pages have increased drastically in the recent past due to increased dynamic and flashy content. Hence it is imperative to have a cache which improves performance. Several researchers have done web analysis with respect to typical behavior using web histogram distribution. However, the results of these web analyses have not been incorporated by the existing replacement policies. As web access phenomenon is dynamic, dynamic access modeling approach can only provide a deep insight. But no attempt has been made so far in all the existing semantic based replacement policies to rate the dynamic level. In this paper an attempt has been made to devise an improved policy that integrates dynamism and semantic relation of the web pages.

III. CONTRIBUTION OF THIS PAPER

The primitive idea of the proposed approach is motivated by the observation that the web access patterns of the user are not random. They exhibit some minute relation from the user behavior and application programs. The present work focuses on designing a new semantic based replacement policy that also considers the influences of the dynamicity of the web pages, the rapid dynamic pages not being cached. In the proposed work, the semantic analysis and ranking of cached pages based on dynamic count have been implemented in Java platform. This puts this work in a unique position to exploit the nature and the updating pattern of web pages that helps in taking effective decision for web cache management. This combination of dynamism and semantic relations of cached pages has resulted in much improved performance. The simulation of the proposed policy has been carried out employing a model-driven approach. The synthetic workload generated in this work reflects the actual access log trace of the Bharathidasan University, referred from the records generated by the squid proxy server covering a week period.

A. Proposed Replacement Policy:

The proposed policy consists of two logical models. The first one called dynamic logic determines the dynamic count of the cached page and the second module called semantic relation further consists of CM and RM to establish the semantic relation among the cached pages. Fig.2 shows the arrangement of the modules in the proposed policy

B. Dynamic Logic:

This logic aims at rating the dynamic count based on the nature of the document. It is a very important heuristic parameter that can be used to prioritize the pages enjoying the same semantic relation marked for purging. Dynamic Count (DC) ranks the page into one of the following four categories:

a. DC = 4 (More dynamic): This type of document is more susceptible for eviction. It holds the content that keeps changing in small time intervals. A typical example would be the web site hoisting the scores of a cricket match.

b. DC = 3 (Dynamic): Content of such page changes with periodic frequency. A typical example is the site hoisting the newspaper. Contents of such pages need to be refreshed daily or every 24 hours.

c. DC = 2 (Partially dynamic): Portion or subset of the web pages alone changes. For e.g. few pages of railway web site will contain static information and remaining pages like reservation chart will be kept updated frequently.

d. DC = 1 (Static): These pages contain static information that will be updated at rare occasions. For e.g. in websites of educational institutions, syllabus will remain static at least for one academic year.

The dynamicity of the requested web page is assigned by using the following algorithm.

Algorithm: Dynamic (Requested URL:(P)) [Dynamic Count(DC): 4-Highly dynamic,

```
3-Dynamic,2-Partially dynamic,1-Static]
Check `P' with the set of URLs identified as dynamic links stored with their respective DC.
if any match is found
set DC(P) = DC(i)   where i is the matching URL.
return(DC(P))
check the extensions of the files accessed  from P to decide the  dynamicity
switch (file extension)
case(pdf or mhtml or doc or txt)
    DC = 1
case(html or htm)
    DC = 2
case(xml or php or asp or aspx or wma or wav)
   DC = 3
end case.
return(DC)
check the important keywords of P with the built-in list.
if any match is found
   DC = 2.
return(DC)
if (time of day = expiry time(P))
  DC = 4.
return(DC)
count the number of web links available in P
if (count > threshold)
DC=3// threshold is a dynamic parameter  initialized in the text file
if (response = ``yes") //get user preference to cache the page or not
   DC = 1
else if(response = ``no")
   DC = 4
return(DC)
if(clearbrowsercache = 1) then
 empty the cache.
```

C. Semantic Relation:

Semantic relation among cached documents and the incoming document is established through the following two different models of CM and RM with the same objective of calculating the semantic distance/offset.

a. Clustered Model:

The simplest definition of clustering is grouping together of similar data items into clusters. The mathematical definition of clustering as stated in [21 is: let $X € R^{m*n}$ a set of data items representing a set of m points x_i in R^n. The goal is to partition X into K groups Ck such that each data belonging to the same group is more "alike" than the data in different groups. Each of the k groups is called a cluster. The result of the algorithm is an injective mapping $X \rightarrow C$ of data items Xi to clusters Ck.

The following definitions are assumed in the clustering algorithm:

$X € R^{m*n}$ denotes a set of data items representing a set of m points x_i in R^n., Ck denotes the k -th cluster , K represents the number of clusters, $C^j{}_k$ denotes the k-th cluster center at the iteration j.

Non- parametric approach is adopted for this clustered model. Two good representative examples of this approach are the agglomerative and divisive algorithms, also called hierarchical algorithms that produce dendograms [22]. A dendogram is simply a tree that shows the group of clusters that were agglomerated in each step. It can be easily broken at selected links to obtain clusters or groups of desired cardinality or radius. Of course the major limitation of this dendogram is, its sensitivity to merging of clusters that occurred in previous steps i.e, data are not permitted to change cluster membership once assignment has taken place. The key advantage of this method is that they no assumptions are made about the underlying data distribution. For implementing the clustering algorithm a matrix with pairwise similarities is required and it demands storage space of the order of m^2 for m number of data points to be clustered.

b. Hierarchical Clustering Algorithm:

The algorithm adopted in CM model is the variation of Hierarchical Clustering Algorithm namely complete linkage clustering (Maximum or Furthest - Neighbor Method). The dissimilarity between two clusters i and j is the minimum dissimilarity between member of the cluster i and member of the cluster j. This method tends to produce a very tight clusters of similar cases. This method uses centroid as initial set of clusters that will select the starting cluster close to the mass centroid of data set. Each cluster center is calculated adding a small random perturbation to the centroid of the dataset.

```
Algorithm: Let D(i,j) be the distance between clusters i, J and N(i) the nearest neighbor of  cluster i.
Initialize 'N' clusters // N = No. of pages stored in cache.
For each pair of clusters (i,j) computer D(i,j)
For each cluster i compute N(i)
Repeat until the desired number of clusters is obtained
Determine i,j such that D(i,j) is maximized
Agglomerate cluster i and j
Update each D(i,j) and N(i) as necessary
End of repeat
```

Important keywords of the web page contents are identified and clusters are obtained in the form of trie like data structure [23] and then the semantic offset is calculated. All the common categories are first listed out. Then word count is calculated for these keywords and based on the word count, the keywords are added to the corresponding category list for that page. Number of similar and dissimilar categories between the pages are counted. The page which has more number of dissimilar categories accumulated with respect to the other remaining pages is assigned a high eviction index EI.

A web server is simulated and the document manager acts like a web server that returns the document when queried, if it is found in the cache. If the requested document is not in the cache, it redirects the request to the appropriate server and fetches the requested page. The newly fetched page is then cached if it can be accommodated, else replacement has to be made for want of space. The page with the highest EI can be evicted for replacement. If more than one pages hold the same EI then the dynamic logic will prioritize eviction process based on DC. The newly cached page has to be inserted into appropriate cluster and the keywords have to be arranged properly into the already established trie data structure that can be further processed to assign new EI.

c. Relational Model:

This idea is proposed based on the underlying assumption that the user access pattern is not random and is greatly influenced by the behavior of the users and their invocation pattern of the application programs. If the user access pattern exhibits semantic relation among the cached pages, then the cache replacement policy could be enforced based on the semantic relation. Those pages that possess more dynamism by nature can be exempted from caching. The semantic relation in terms of semantic distance is evaluated between the incoming page and the cached pages. The page which has minimum Relation Index RI (less related) can be marked for eviction. If more than one pages hold the same minimum semantic distance then the dynamic logic will prioritize the pages based on the dynamicity for eviction. The relation between the files can be expressed in two categories namely: (i) Inter file relations and (ii) Intra file relations.

An interfile relation exists between two files A and B, if B is the next file opened following A being closed. A is called B's precursor. A heuristic policy INTER is defined as quoted in [24] to show the relation of the file with its precursor. A heuristic parameter, INTER(i), is defined to represent the importance of a file i with respect to the interfile relations with its precursors. The stronger the relation, the less likely it will be replaced. Thus it is used to calculate RI. The following parameters are logged for each cached page p to calculate RI:

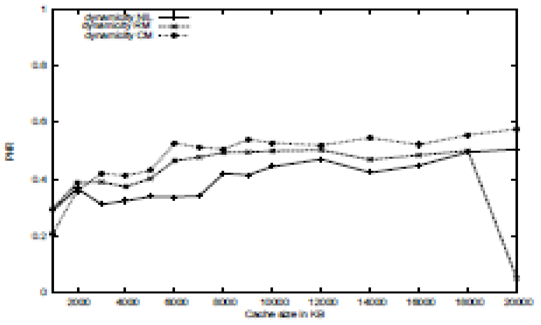

Figure 3. Page hit ratio for varying cache size with and without dynamic logic.

N_p : number of time the page p is accessed

T_p : time since the last access is made to p

T_j : represents the time since the last access to file j where j is a precursor of p .

$$INTER(p) = \frac{N_p}{T_p + \sum_{i=1}^{n}\left((T_j - T_p) \times \frac{N_j}{N_p}\right)} \quad (1)$$

N_j : represents the number of times file j precedes file p .

INTER (p) calculates the semantic relation of the file with other files based on the recency, popularity and based on precursor relations. Another heuristic parameter, INTRA(p), is defined as quoted in [24] to represent the unimportance of a file p based on its intra file relations. This is opposite to the definition of the importance based on interfile relations explained above, in the sense that the bigger the value the less important is the file. Here rather than depending on precursors, INTRA(p) is defined based on the files that shared time.

T_p: represents the time since the last access to file p

T_j: represents the time since the last access to file j where j is open before p is closed

$S_{p,j}$: represents the shared time of file p with respect to file j where p is closed before j

S_t: represents the total shared time with all files that are open before p is closed

$$INTRA(p) = T_p + \sum_{i=1}^{n}\left((T_j - T_p) \times \frac{S_{p,j}}{S_t}\right) \quad (2)$$

where,

$S_{p,j} = C(p) - MAX(O(p), O(j))$

$O(p)$: Open time for file p

$O(j)$: Open time for file j

$C(p)$: Close time for file p

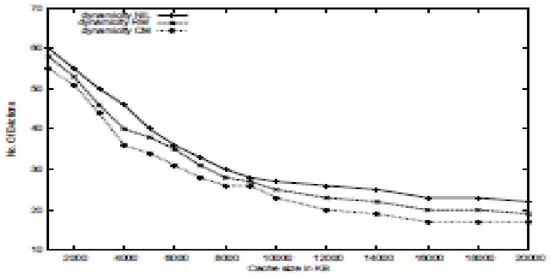

Figure 5. Number of evictions for varying cache size with and without dynamic logic.

The INTRA (p) gives the unimportance of the file based on the shared time of the files.INTER(p) and INTRA(p) are combined to be reflected as a single value as RI(p) which is given as

where $A = \sum_{i=1}^{n}\left((T_j - T_p) \times \frac{N_j}{N_p}\right)$

and $B = \sum_{i=1}^{n}\left((T_j - T_p) \times \frac{S_{p,j}}{S_t}\right)$

$$RI = \frac{N_p}{T_p + A + B} \quad (3)$$

$$V = MAX(DC(i)) \quad (4)$$

The larger the value of RI(p), the less likely it is to be replaced. When the file p is to be replaced, calculate RI(p) with respect to the remaining file in the cache and select the victim file V which has a the lowest RI. If more than one file possess same RI then the dynamic counts of those pages are calculated and the one with maximum count is considered to be more dynamic and hence that file V is selected for replacement.

where DC(i) is the Dynamic Count of page i then V = Max(DC(i)) for all i that has same RI.

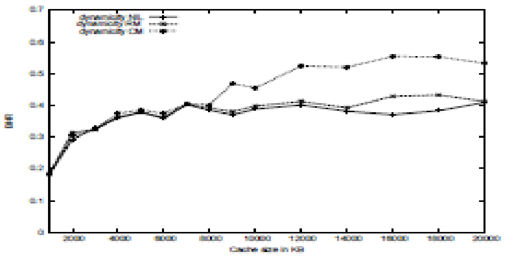

Figure 4. Byte hit ratio for varying cache size with and without dynamic logic.

Figs.3, 4 and Fig 5 show the graphs plotted for the key cache performance metrics like PHR,BHR and number of evictions respectively for varying cache size from 1000 to 20,000 KB. Simulation has been carried out without dynamic logic(dynamicity_NIL) and with dynamic logic. Inclusion of dynamic logic is done by using both the models CM (dynamicity_CM) and RM (dynamicity_RM). EI obtained from CM marks the pages for purging. For the same test cases RI obtained from RM marks the pages for purging.

From the results observed and depicted, it is clearly understood that dynamicity_CM outperforms dynamicity_RM and dynamicity_NIL for all the above mentioned parameters. Though both the models (CM and RM) enjoy same time complexity, CM performs well. The only limiting factor of CM is the need for some extra space. It is consumed while framing the matrix for grouping the clusters. As far as PHR is concerned there is a marginal improvement for dynamic logic as shown in Fig.3. The inclusion of dynamic logic gives better performance especially in terms of BHR from 8000KB onwards as shown in Fig. 4. As the cache size is increased keeping the requests as the same, the hit ratio grows steadily and finally attains a constant saturation value [Figs. 3 and 4] which turns out to be approximately 55 to 60%. Also after attaining a saturation value, there is not much variation in the hit ratio even if the cache size is increased further. It is imperative to choose a proper cache size. Too less is going to be less efficient and too high can be very expensive. Hence the size needs to be optimal. This implies that from all the above metrics, the optimal cache value can be freezed to 20,000 KB beyond which there is no reasonable improvement for the above requests made. Though it is obvious that number of evictions can be minimized if the cache size is more, from Fig. 5 it can be perceived that it follows a straight line in the range 15,000 KB to 20, 000 KB. Theoretically if the cache is large enough, communication between the client and server can be greatly reduced; however, practically large caches aren't feasible yet on resource constrained devices.

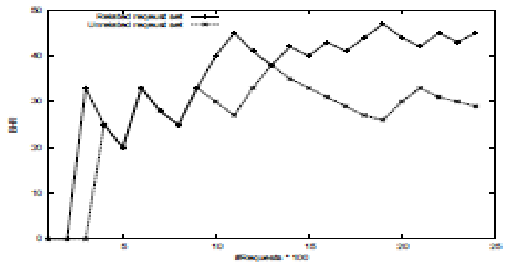

Figure 6. Page hit ratio for varying request sets.

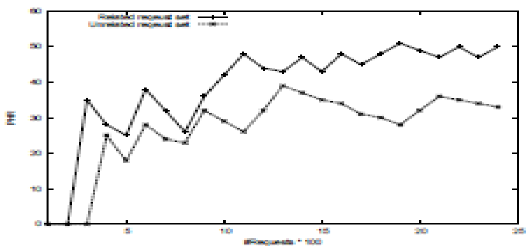

Figure 7. Byte hit ratio for varying request sets.

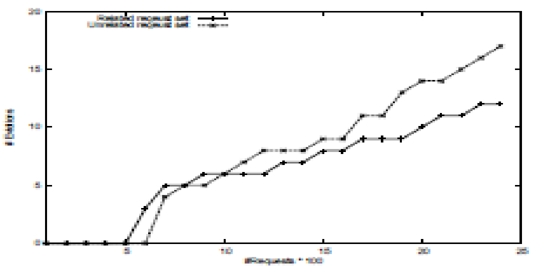

Figure 8. Number of evictions for varying request sets.

Since dynamicity_ CM gives better performance as observed in Fig. 3,4 and Fig. 5, other two types namely dynamicity_RM and dynamicity_NIL are ignored for further experiments. The next set of simulations are carried out for related request set and unrelated request set with repspect to an individual user. In has been proved that the efficiency of dynamicity_CM has not been degraded even if the user triggers unrelated request set, as shown in Figs.6,7 and Fig. 8. In both the cases for related request set, it is apparent that both PHR and BHR increase sharply in the beginning and tend to settle around 50% which is supposed to be a good hit rate i.e. one in two pages is found in the cache. For unrelated request set, on the other hand, the hit ratio tends to settle around 30%. Hence there is a remarkable improvement in using this policy for related request set. If the user requests for pages that are semantically related and within quick succession, then the hit ratio are increased by almost 20% from unrelated request set which contains random pages that are less related. Fig. 8 illustrates the improvement in the number of evictions made for related request set compared to unrelated for a large set of requests submitted.

The behavior of this DynaSem policy (dynamicity_CM) is dependent on the nature of the requests being made and hence different sets of requests are generated and its performance is evaluated and compared against the standard LRU policy, SEMANTIC, SEMALRU in terms of PHR, BHR and number of evictions. SEMANTIC policy evicts pages only based on semantic relation, whereas SEMALRU combines semantic policy with LRU. The results are shown in Figs. 9, 10 and Fig. 11.

Both the hit rates follow similar pattern for SEMALRU and DYNASEM and beyond 50000 KB, DYNASEM performs better than the remaining policies. The values initially rise to a certain limit and later attain saturation. In Certain systems the size of the cache might be a critical factor or a bottleneck. For such systems the optimal value of the size of the cache could be easily fixed from the above observation. The above results have been noted only for related request set and DynaSem policy supersedes the other policies in all the above metrics.

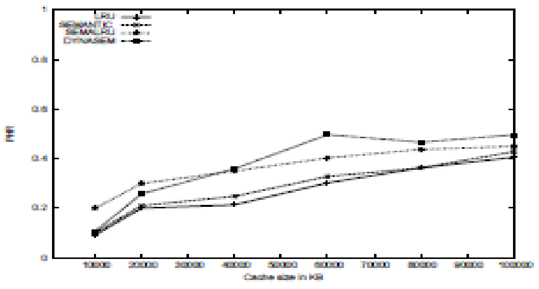

Figure 9. PHR VS varying cache size for different policies.

IV. CONCLUSION AND FUTURE WORK

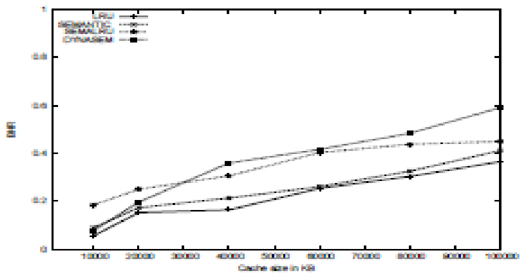

Figure 10. BHR VS varying cache size for different policies.

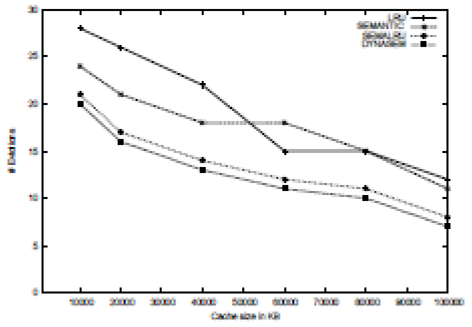

Figure 11. Number of replacements Vs varying cache size for different policies.

This work aims at improvising the semantic based web cache replacement policy by considering the level of dynamism among the pages possessing the same relation. A modified policy termed `DynaSem' has been developed. A formal framework for the DynaSem policy has been designed that incorporates two logics - dynamic and semantic relations. Rating the document based on the dynamic count leads to conservative improvement in most of the performance metrics used to evaluate the cache replacement policy. The semantic relation has been established using clustered model and relational model. Both models aim at identifying the dependencies among the cached document and the new incoming document. The request set is generated by a simulation process and the clustered model can be chosen, if the user access pattern is going to explore all the possible sub links provided in a web site. Relational model can be preferred if the user switches between various web sites for related information. Though both models experience same time complexity, better results can be obtained in CM in spite of its increased space complexity. Using model driven simulation, the performance of DynaSem policy has been analyzed for related and unrelated request sets. Even if the user access pattern is not related, this policy has not deteriorated much from other policies namely LRU, SEMANTIC and SEMALRU. Ranking the documents based on their dynamism shows commendable results and hence it can be used as a vital parameter for tuning the performance of all prevalent replacement strategies that ignore file relations and communication overhead. An increasingly important technique to enhance the web caching performance is to prefetch web pages. Prefetching can happen in a predictive manner or in an interactive manner. For predictive prefetching, the proposed policy can be used to predict the reference probability of new requests after analyzing the user access pattern The experiments conducted in the proposed work are restricted to only isolated cache. A possible extension to this work could be to experiment with a grid of cooperating caches. This policy tackles single user interest and can be extended to satisfy group of users. Instead of trie structure used in CM, standard vector model that is widely employed in search engines for information retrieval can also be adopted.

V. ACKNOWLEDGMENT

The authors would like to thank Mr. N. Ramasubramanian, Associate Professor, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli for permiting to conduct the simulation work in his lab and for his valuable suggestions in making this work possible

VI. REFERENCES

[1] Kin Yeung Wong, “Web Cache Replacement Policies: A Pragmatic Approach”, IEEE Network , 20(3):342-351, 2006.

[2] J Wanf, “A Survey of Web Caching Schemes for the Internet”,ACM SIGCOMM Computer. Communication review, 29(5):36-46, 2006.

[3] Abdullah Balamash And Marwan Krunz, “An Overview of Web Caching Replacement Algorithms”, IEEE Communications Surveys, 6(2):44-56, Second Quarter 2004.

[4] S Podlipnig and L Boszormenyi, “Web cache Replacement strategies”, ACM Computing Surveys, 35(4):374-398, 2003.

[5] Brian D Davison, “A Web Caching Primer”, IEEE Internet Computings, 5(4):38-45, July/August 2001.

[6] K Psounis and Balaji Prabhakar, “A Randomized Web-Cache Replacement Scheme”, INFOCOM - Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, 4, 1407-15, April 2001.

[7] L Rizzo and L Vicisano, “Replacement policies for a proxy cache”, IEEE/ACM Trans. Networking , 8(2):158-70, April 2000.

[8] S Williams et al., “Removal Policies in Network Caches for World-Wide Web documents”, Proc. ACM SIGCOMM, 293-305, Aug. 1996.

[9] P Cao and S Irani, “Cost-Aware WWW Proxy Caching Algorithms”, Proc. USENIX Symp. Internet Tech. and Sys.., 193-206, Dec 1997.

[10] L Breslau et al., “Web caching and Zipf-like Distributions: Evidence and Implications”, Proc. INFOCOM, 126-34, Aug. 1999.

[11] Alcides Calsavara, Rogerio Guaraci dos Santos, Edgard Jamhour, “The Least Semantically Related Cache Replacement Algorithm”, ACM Latin America conference on Towards a Latin American agenda for network research Proceedings of the 2003 IFIP, 21-34, October 2003.

[12] Ren Q, Dunham M H, and Kumar, “Semantic caching and query processing”, IEEE transactions on Knowledge and Data Engineering, 15, 192-210, 2003.

[13] Michael Stollberg, Martin Hepp, and Jorg Hoffmann, “A Caching Mechanism for Semantic Web Service Discovery”, LNCS 4825, Springer-Verlag Berlin, 15, 480-493,2007.2

[14] K Geetha, and N Ammasai Gounden, and S Monikandan, “SEMALRU: An implementation of modified web cache replacement algorithm”, INC -09 International Symposium On Innovations In Natural Computing IEEE Computer Society, 1406-1410, Dec. 2009.

[15] Chidlovskii, B Roncanico C, “Semantic cache mechanism for heterogeneous web querying”, www8/computer networks, 31(11-16):1347-1360, 1999.

[16] Zheng B, et al., “Cache invalidation and replacement strategies for location-dependent data in mobile environments”, IEEE Transactions on computers, 10(51):1141-1153, 2002.

[17] C Aggarwal, J Wolf, and P Fellow, “Caching on the World Wide Web”, IEEE Trans. Knowledge and Data Eng., 11(1):94-107, Jan./Feb. 1999.

[18] Murta et al., “Analyzing performance of partitioned caches for the WWW”, Proceedings of 3rd International WWW caching workshop, 1998.

[19] Cheng K, Kambayashi, “Advanced replacement policies for www caching.,” In web-Age Information Management, 239-244, 2000.

[20] R Wooster and M Abrams, “Proxy Caching that EstimatesPage Load Delays”, Proc. 6th Intl. World Wide Web Conf., Santa Clara, 32534, Apr.1997.

[21] D Fasulo, “An analysis of recent work on clustering algorithms. of the Technical report”, 1999.

[22] Glenn Fung, “A Comprehensive overview of Basic Clustering Algorithms” of the technical report, June 2001.

[23] Seung-Hyun Oh, and Jong-Suk Ahn, “Bit-map trie: a data structure for fast forwarding lookups”, INC -09, Global Telecommunications Conference, GLOBECOM '01. IEEE, 3, 1872-1876, Dec.2001.

[24] Sharun Santhosh and Weisong Shi, “A Semantic-based Cache Replacement Algorithm for Mobile File Access”, WWW2005 conference, 2005.