BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **11**, No 4

Sofia • 2011

SAT – A Split-Up Cache Model to Boost the Performance of Web Cache Replacement Policies

Geetha Krishnan[1], *Ammasai Gounden*[2], *Nanjappa Gounder*[2]

[1] *Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli-15, Tamilnadu, India 620015*
[2] *Department of Electrical and Electronics Engineering, National Institute of Technology, Tiruchirappalli-15, Tamilnadu, India 620015*
Emails: geethavalavan@yahoo.com ammas@nitt.edu

Abstract: *This work intends to design a new model for web cache by fragmenting the cache into three slices namely Sleep Slice (SS), Active Slice (AS) and Trash Slice (TS) and hence the name SAT for this cache model. This model is explored only for client cache. Slicing is done to group cached pages based on the hit count so that latency of retrieving them can be reduced. By discriminating one time hit pages from the remaining pages, hot pages are usually made available. Simulations have been carried out to investigate the desirable percentage of each slice with respect to the total cache capacity for different cache replacement policies. The outcome of the proposed SAT model ranked by the performance metrics such as File Hit Ratio, Speedup, Delay Saving Ratio and Number of Evictions reports thriving progress to the basic cache model. Three types of replacement policies based on the key parameters recency, frequency, size and semantic relation are considered for the model proposed. These replacement policies are implemented only in AS. SS replacement is done with respect to frequency of cached pages whereas TS replacement is done only on demand. The proposed SAT model has exhibited about 15-20% improvement on an average for the above mentioned replacement policies and performance metrics.*

Keywords: *Web caching, replacement policy, victim cache, sliced cache.*

1. Introduction

A major concern for Internet surfers is to avail an effective retrieval of stored information from the web servers. As there is a drastic increase in Internet usage, popular web sites are overloaded and simultaneous access to these web sites cannot deliver the requested information with minimum latency. Web caching and replication are the two important approaches for enhancing the efficient delivery of web contents, thereby reducing latencies experienced by users. Improving the response time and access latency for clients has become quite an important and challenging issue. Reference [1] reports a large number of one time hit pages in the web request streams. SLRU [2] deals explicitly with this factor and proposes the use of a small auxiliary cache to maintain metadata of evicted objects. Of course, this method has to decide the size of the auxiliary cache. The advantages of modeling the client side cache are presented in [3]. Further the impact of cache pollution is highlighted in [4]. Cache partitioning has been adopted by some algorithms, but not for the purpose of the isolation of one-time hit pages [5]. As mentioned in [6], inactive pages must be identified and marked for eviction from cache to prevent performance degradation due to pollution. The literature has analyzed numerous web cache replacement policies [7-10]. Most of these replacement policies consider a single parameter for replacement which is not appropriate for a wide range of applications, since each policy tries to optimize only a few of the performance metrics. Recently Kin has made the following recommendations regarding the performance metrics and the corresponding replacement policy [10]: i) To achieve higher hit rate with small cache, recency based policies; ii) For commercial based proxy to achieve higher BHR with a limited bandwidth and complex traffic characteristics, GD-size based policy. For systems in which the popularity distribution of objects is highly skewed, frequency based policies can be adopted [11-12]. In addition to frequency, recency and size based policies, semantic relatios among cached pages are also considered for replacement as discussed in [13-15]. To attain the combined effect of the above merits, many hybrid policies can be adopted, but at the cost of more complexity. Instead of having a separate auxiliary cache, an attempt has been made to slice the cache itself so that each slice can make out its own explicit purpose and can be customized to improve the overall performance. Each slice capacity can also be allowed to grow or shrink depending on the characteristics of the request streams.

2. Motivation

Ample research has been ongoing in the area of web caching but the state of art dictates not to devise new replacement policies, rather to provide incremental improvements in the existing policies focusing on the target environments. In this context, instead of introducing changes to the existing policies, the primary structure of the cache can be remodeled so that it can reveal the essential factors exhibited by the policies as such in cache itself [16]. The following factors motivated to design and propose a new cache model for client side cache:

i) protection of the cache from one-time hit pages, but without preventing the cache from adapting to a changing access pattern; ii) customizing the cache model as per the client access pattern that reflects qualitative characteristics of the request streams; iii) slicing the cache based on the vital parameters that decide the replacement policies such as recency, frequency and size allows the proposed cache structure to inherit the characteristics of various replacement policies. The main objective is to achieve upgraded values for the performance metrics and at the same time stay away from cache pollution. Both the objectives can be attained by selecting a proper blend of replacement policies and designing the cache with flexible slicing options. The advantages of memory fragmentation have been discussed in [17]. All the benefits that are attained by fragmenting the memory into segments can also be achieved with respect to cache memory slicing notion. By fragmenting the cache into slices, pages of a specific feature can be assigned in the respective slice, so that the search time can be minimized. The cache capacity at client side needs not be considered as a limiting factor, because the hard disk space can be augmented for extension of the cache on demand for TS.

3. Proposed model

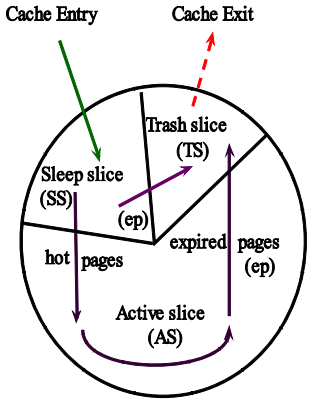

Fig. 1. SAT cache model

In the model proposed the cache is sliced into three fragments, namely SS, AS and TS, as shown in Fig. 1. SS stores the one time hit pages. The requested page from the server enters the cache only into the SS. If the same page has a larger number of hits with recent timestamp, it is then shifted from SS to AS. Appropriate replacement policy is employed only at AS which tries to retain all sorts of hot pages. If the pages have reached expiry time, then such pages even if they are hot, will be removed from AS. Instead, if that page is not requested again (one time hit) for a specific time span, it gets shifted to TS. If they become cold or outdated, then meta data of those pages alone are cached into TS and removed from AS to make a room for further entry. If the pages are highly dynamic, they will be marked for purging and will be moved into TS. All meta data about the deleted pages from their headers will be restored in the TS to ease and speed up the future retrieval of the same pages. The size of each slice in terms of the total cache capacity can be

varied and can be fixed, based on the user's request pattern and constraints. It can be made flexible by keeping a track of percentage of one time hit pages. For each user it can be customized and relevant size is evaluated and the respective performance measures like File Hit Ratio (FHR), Delay Saving Ratio (DSR), Speedup (S) and Number of Evictions (NE) are studied. The speedup factor is evaluated using byte hit ratio.

4. Implementation details

All the requests received from the user are first processed in order to filter out the meaningless and unavailable requests. The requests will become meaningless if a typographical error was made in the URL name. A request will be unavailable if the server is down or not found. Then the requested web site is searched in the Cache Index Table (CIT). Each entry in the CIT stores Unique ID, Cache_in_time, Freqeuncy Count (FC), Size, Server location. If the entry exists, it is a hit in the cache, otherwise it is a miss. In case of a hit, the respective slice is identified based on the FC from CIT. If FC is one, it is one time hit page and hence it can be fetched from SS. If FC is more than one, then the page can be fetched from AS. If both slices do not posses the requested page, then it is a miss. In case of a miss, the page has to be fetched from the respective server. For cacheable pages, the size of the fetched page is compared with a threshold (normally it is initialized to the size of SS). The entire cache operation of the proposed SAT model is portrayed in the flow diagrams of Figs 2-4.

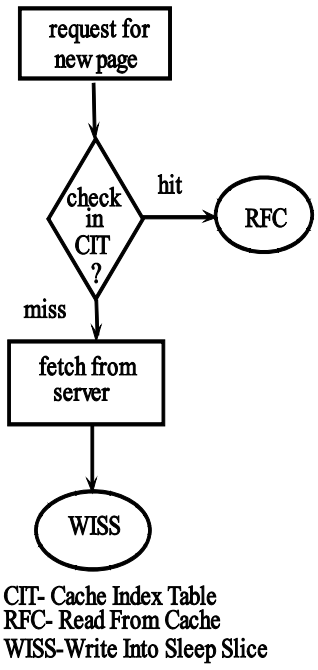

Fig. 2. Cache invocation flow diagram

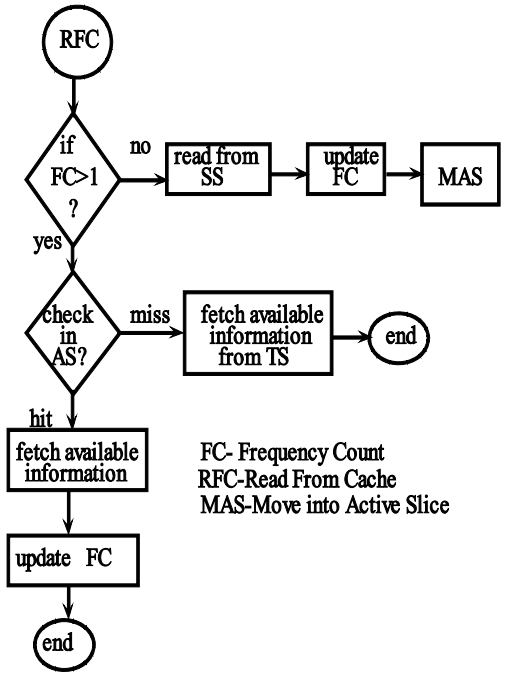

Fig. 3. Read from a cache flow diagram

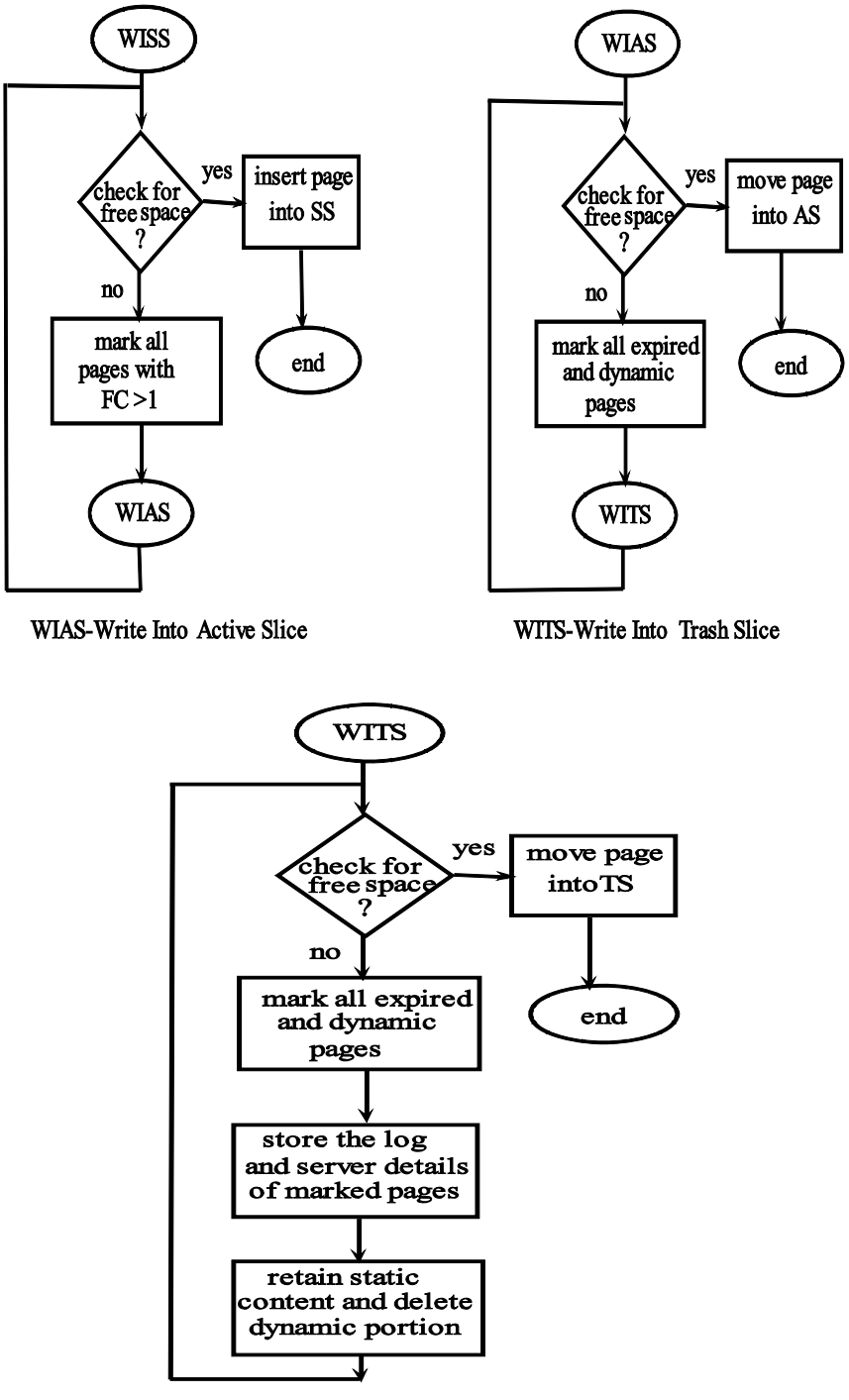

Fig. 4. Write into a cache flow diagram

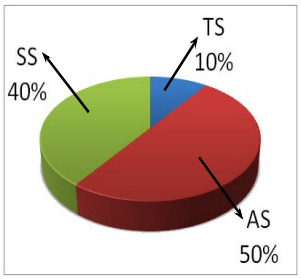

Fig. 5. SAT model-1

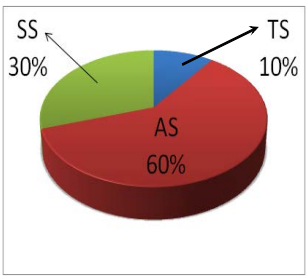

Fig. 6. SAT model-2

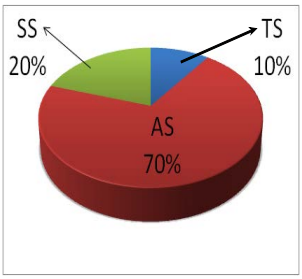

Fig. 7. SAT model-3

Simulation has been carried out for the basic unsliced cache and three different sliced cache models namely SAT model-1, SAT model-2. and SAT model-3, as shown in Figs 5-7. These three models are designed by changing the size of each slice. SS:AS:TS proportion for SAT model-1 is 40%:50%:10%, for SAT model-2 is 30%:60%:10% and for SAT model-3 is 20%:70%:10%. Since AS contains hot pages, the size of AS is always chosen to be the largest among the three slices in all the three models. TS is always kept at minimum. Access logs were collected for a period of 2 months. Each entry in the access log contains information on a single request that includes an IP address, time of request, requested URL, size of the response in bytes and time required to complete the response. Due to the availability of massive information in the access log, it is further processed by filtering certain information due to time and space constraints. The collected and useful log information is stored in CIT by assigning an unique integer identifier for distinct URL. Object modification and user aborted connections are speculated by monitoring the change in the size of the object. It has been observed that approximately 92% of the requested objects were cacheable. Among the user initiated requests it also has been observed that approximately 60% of the requested objects are one time requests. Table 1 summarizes the access log collected for observing the performance of the basic cache model in comparison with SAT model-1, SAT model-2 and SAT model-3 for the three replacement policies, namely DSLV, SEMALRU and SIZE where DSLV is a combination of three logics Dynamic, Semantic, and LFU with Victim tracer, SEMALRU considers the semantic relation of the cached pages and recency and hence the name and SIZE based replacement policy is considered as such without any variation.

Table 1. Summary of the access log collected for a period of two months (filtered data set)

Total Requests	144000
Total Contents	1.66 GB
Unique cacheable requests	129600
Total uncacheable requests	6269
Unique cacheable bytes	0.49 GB
Total uncacheable bytes	0.14 GB

The first policy DSLV performs replacement in AS, based on dynamicity, semantic relation, LFU of the cached pages and also makes use of the victim tracer. This policy which is a refinement of LFU has conferred a promising result in our previous work that has made use of DYNASEM [18], which is an augmented policy

with LFU in addition to referring to the trace of victimized pages maintained in the victim slice. The second policy is based on the semantic relation between the cached pages and the new incoming page, combined with LRU and hence called as SEMALRU [19]. The third policy is based on SIZE where the replacement of a page is done by marking the page with the largest size to get enough room for the incoming page. If the size of the marked page for purging is not enough to accommodate the new incoming page, then the same process is repeated until enough space for the incoming page is acquired in AS. These replacement policies are implemented only in AS. SS replacement is done with respect to the frequency of cached pages whereas TS replacement is done only on demand.

5. Performance analysis

For FHR as given in Fig. 8, approximately 23% improvement is attained by DSLV, 18% by SEMALRU and 24% by size based replacement policy. It can also be noted that SAT model-3 provides a good result for DSLV and SIZE, whereas SEMALRU performs well in SAT model-1. Since SEMALRU tries to retain semantically related pages in AS, it demands a smaller size for AS compared to the other two policies. DSR value as displayed in Fig. 9, is evaluated using BHR and nearly 33% increase is there for DSLV and 31% increase in SEMALRU in SAT model-1, whereas 32% hike is achieved in SAT model-3 by SIZE based policy. Fig. 10 illustrates the speed up and for all the three policies SAT model-1 outperforms the other models, of which DSLV achieves the best improvement compared with the other two.

The sliced models realize a minimum number of evictions in comparison with the basic model as shown in Fig. 11 and nearly 24% reduction is obtained by all the three policies, of which SAT model-1 gives a good result for DSLV and SEMALRU, whereas SAT model-3 provides a better result for SIZE based policy. It can be inferred that the size of each slice, especially the ratio between AS and SS can be presumed based on the user's interest, and request- set that tries to target a particular parameter. It can be deduced that in all the above metrics, either SAT model-1 or SAT model-3 performs well and hence the slice capacity can be dynamically fixed and customized as per user's option.

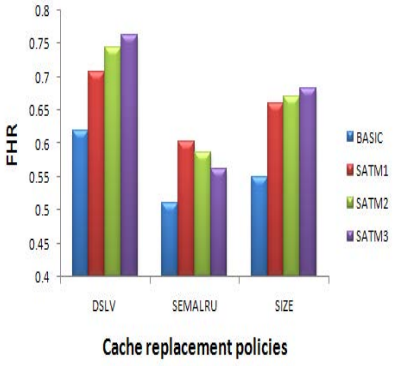

Fig. 8. FHR versus specific replacement policies for different cache models

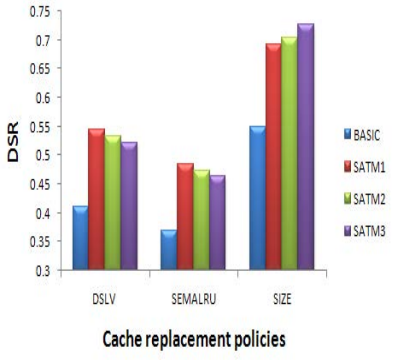

Fig. 9. DSR versus specific replacement policies for different cache models

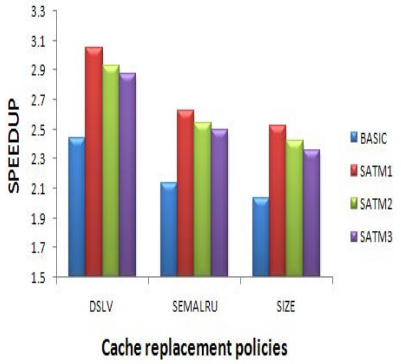

Fig. 10. Speedup versus specific replacement policies for different cache models

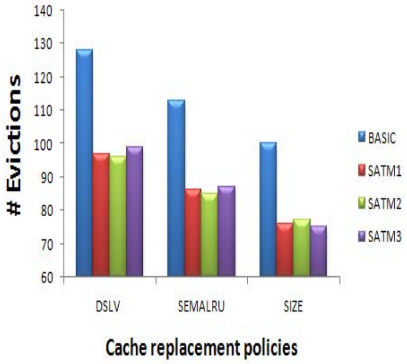

Fig. 11. Number of evictions versus specific replacement policies for different cache models

6. Conclusion

Sliced cache always minimizes the search space and gives appropriate weightage to user access pattern by identifying one time hit pages with the remaining accessed pages. It gives a kind of protection to the hot pages that reside in the AS. The entry and exit of the web pages to cache always materialize through SS. Since the sliced cache model inherently inherits both the temporal and spatial locality of reference at the same instance in different slices, the missing parameters of the adopted replacement policies are incorporated in the proposed cache structure itself. This factor improves the quality of response for user's query. TS is efficiently utilized by making use of the meta data that has been logged before removal of the pages from either SS or AS. TS size is always freezed to 10% of the total size of the cache and can be even minimized if the hard disk can be extended to meet this requirement. TS replacement seldom occurs but needs to be initialized and reset for every login session. Though the time complexity of a sliced model is relatively more compared to the basic model, it can be nullified in a multithreaded environment as the read, write and move operations into each slice can be realized by appropriate threads. The size of each slice can be dynamically varied and can be customized as per the users request style to guarantee improved performance. The size of the slices can be chosen based on the replacement policy implemented and the target parameter to be tuned. The proposed SAT model has exhibited about 15% to 20% improvement on average for the above mentioned replacement policies and performance metrics. The fact that among the user initiated requests about 60% are one-time references, forms the basis for overall improvement of this proposed SAT model. It can be ascertained from the result that all the above mentioned replacement policies can achieve enhanced outcome in terms of most of the crucial parameters used to evaluate the performance of cache replacement policy when tested with sliced model. It is evident that the same model will exhibit better performance and will also substantiate a prudent effect for other web cache replacement policies also. This sliced model is investigated only with isolated client side cache and a possible extension to this work could be to experiment the same with co-operative caches

and for caches deployed at intermediate levels and sever side. In such scenario, speculating the multiple users' interest and working with multiple co-operative caches will be a real challenge.

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

References

1. Williamson, C. On Filter Effects in Web Caching Hierarchies. – ACM Transactions on Internet Technology, Vol. **2**, 2002, No 1, 47-77.
2. Aggrawal, C., J. Wolf, P. S. Yu. Caching on the World Wide Web. – IEEE Transactions on Knowledge and Data Engineering, Vol. **11**, 1999, No 1, 94-107.
3. Teng, W. G., C. Y. Chang, M. S. Chen. Integrating Web Caching and Web Prefetching in Client-Side Proxies. **–** IEEE Transactions on Parallel and Distributed Systems, Vol. **16**, 2005, No 5, 444-455.
4. Rizzo, L., L. Vicisano. Replacement Policies for a Proxy Cache. – IEEE/ACM Transaction Networking, Vol. **8**, April 2000, No 2, 158-170.
5. Arlitt, M., R. Friedrich, T. Jin. Performance Evaluation of Web Proxy Cache Replacement Policies. – Elsevier Performance Evaluation, Vol. **39**, February 2000, No 1&4, 149-164.
6. Wanf, J. A Survey of Web Caching Schemes for the Internet. – ACM SIGCOMM Computer Communication review, Vol. **29**, 2006, No 5, 36-46.
7. Balamash, A., M. Krunz. An Overview of Web Caching Replacement Algorithms. – IEEE Communications Surveys, Vol. **6**, Second Quarter 2004, No 2, 44-56.
8. Podlipnig, S., L. Boszormenyi. Web Cache Replacement Strategies. – ACM Computing Surveys, Vol. **35**, 2003, No 4, 374-398.
9. Davison, B. A Web Caching Primer. – IEEE Internet Computings, Vol. **5**, July/August 2001, No 4, 38-45.
10. Wong, Kin-Yeung. Web Cache Replacement Policies: A Pragmatic Approach. – IEEE Network, Vol. **20**, January/February 2006, No 3, 342-351.
11. Robinson, T., M. V. Devarakonda. Data Cache Management Using Frequency-Based Replacement. – In: Proceedings of the ACM SIGMETRICS, 1990, 134-142.
12. Prischepa, V. V. An Efficient Web Caching Algorithm Based on LFU-k Replacement Policy. – In: Proceedings of the Spring Young Researchers, Colloquium on Database and Information Systems, 2004.
13. Calsavara, A., R. G. dos Santos, E. Jamhour. The Least Semantically Related Cache Replacement Algorithm. – In: ACM Latin America Conference on Towards a Latin American Agenda for Network Research Proceedings of the 2003 IFIP, October 2003, 21-34.
14. Ren, Q., M. Dunham, H. Kumar. Semantic Caching and Query Processing. – IEEE Transactions on Knowledge and Data Engineering, Vol. **15**, 2003, 192-210.
15. Stollberg, M., M. Hepp, J. Hoffmann. A Caching Mechanism for Semantic Web Service Discovery. – In: LNCS 4825. Vol. 15. Berlin, Springer-Verlag, 2007, 480-493.
16. Arlitt, F., C. Wzpnson. Internet Web Servers Workload Characterization and Performance Implications. – IEEE/ACM Transaction on Networks, Vol. **5**, 1997, No 5, 631-645.
17. Denning, P. J. Virtual memory. – ACM Computing Surveys, Vol. **28**, 1996, No 1, 153-189.
18. Geetha, K., N. A. Gounden. Dynasem an Improvised Dynamic and Semantic Based Web Cache Replacement Policy. – International Journal of Advanced Research in Computer Science, Vol. **2**, September-October 2011, No 5, 637-644.
19. Geetha, K., N. A. Gounden, S. Monikandan. SEMALRU: An Implementation of Modified Web Cache Replacement Algorithm. – INC-09 International Symposium on Innovations in Natural Computing IEEE Computer Society, December 2009, 1406-1410.