EVALUATION OF RANDOMIZED REPLACEMENT POLICIES FOR WEB CACHES

F.J. González-Cañete, J. Sanz-Bustamante*, E. Casilari, A. Triviño-Cabrera
Department of Tecnología Electrónica, University of Málaga
Universidad de Málaga, E.T.S.I. Telecomunicación, Campus de Teatinos, 29071, Málaga, Spain
{fgc,ecasilari,atc}@uma.es
*jsb_ultrasonica@hotmail.com**

ABSTRACT

This paper presents a comparison of the performance of seven randomized replacement policies proposed for Web caching (RAND, HARMONIC, LRU-C, LRU-S, CLIMB-C, CLIMB-S and RRGVF) and the classical LRU scheme widely used in Web proxy caches. This comparison is performed using a Web cache simulator that implements the aforementioned replacement policies and uses a workload of real proxy traces to simulate the behaviour of a real system. Because of the randomized nature of those algorithms, the simulations have been executed sufficient times to obtain good estimators of the performance of each scheme. Finally, although there is not a replacement policy that outperforms the others for the metrics used, RRGVF can be considered the best choice for all cache sizes.

KEYWORDS

Randomized Replacement Policies, Web Caching.

1. INTRODUCTION

Due to the exponential growth of the World Wide Web traffic the proxy caching technique was proposed in (Luotonen & Altis, 1994). This technique consists in storing the documents requested by the Internet users in a middleware system located between the users and the Internet Web servers called proxy cache. In that way, when a user requests a document and this document is stored in the cache, it is served directly from the proxy instead of the original Web server. As the proxy cache is usually placed near the users, the retrieval latency perceived by the users is decreased and the Internet traffic and Web servers load is also reduced because the requests and responses do not reach them. Although these consequences could be considered clear advantages, the utilization of Web caching techniques may be associated to some drawbacks for the Webmasters because they lose the control over the access count of their servers and documents. Another problem that is present with this kind of caches is that the documents they store may be copyrighted and hence they could violate the copyright laws if they store those documents without permission or paying taxes (Wessels, 2001). In spite of those problems, Web caching is widely used over the Internet.

The operation of a Web cache is quite simple. When a request reaches the cache, it looks for the requested document in the storage space. If the document is found in the cache it is served to the user, in another case, it is requested to the original Web server, stored in the cache and served to the user. In the case that the document is going to be stored in the cache, there are two possible situations: there is enough space for the new document and hence it is stored or there is not sufficient space and hence it is necessary to evict some documents from the cache to make room for the new one. The decision about which are the documents to be evicted from the cache is performed by the replacement policy algorithm.

The task of a replacement policy is to select those documents that have the lowest probability of being referenced again in the future for eviction, but this is not an easy issue because many parameters could be taken into account such as the recency, that is, how recently the document was referenced, the frequency or how many times the document was referenced, the size of the documents, the latency, and so on. In that way, many replacement policies have been proposed and each of them takes into account one or more of the

aforementioned document characteristics. Therefore the replacement policies can be classified depending on those characteristics that they consider.

Several replacement policies have been proposed. In the classification proposed in (Jin & Bestabros, 2001) three groups are considered:

- Recency based: Algorithms that take into account the time (and size or cost) of the last request of the documents (LRU - Least Recently Used).
- Frequency based: Replacement policies that consider the frequency of access to the documents (LFU – Least Frequently Used, Hybrid…).
- Recency/frequency based: They use the recency as well as the frequency to make the decisions (LRU-K, LFU-DA…).

This classification has the drawback that it does not consider those replacement policies that do not take into account the recency or frequency such as GDS (Greedy-Dual Size), or those that only consider the size of the documents such as LFF (Largest File First).

In (Balamash & Krunz, 2004) two classifications are proposed. Firstly the replacement policies are classified in Deterministic and Randomized, and the second classification is recency, frequency and size based. This second categorization was also proposed in (Khayari, 2003) and (Khayari, Best, & Lehmann, 2005) and it is not an excluding classification because a replacement policy can belong to more than one category.

In (Podlipnig & Böszörmenyi, 2003) an extension of the previous categorization is proposed adding two new groups:

- Function based: A function is used to assign a value to each document. This function depends on some parameters such as size, latency or frequency. The document to be evicted is the one with the slowest value.
- Randomized: The selection of the documents to be ejected is randomly based.

Finally, (Nagaraj, 2004) proposed to classify the replacement policies in three groups:

- Traditional replacement policies: Such as LRU and LFU.
- Key based: It includes the algorithms that consider a primary key to evict the documents. This primary key is one of the characteristics of the document such as the size or latency.
- Cost based: It includes the replacement policies that value the documents using a cost function based on some characteristics of the documents.

In this work we will evaluate and compare the performance of seven randomized category replacement policies. Although these replacement policies have been evaluated separately they have never been compared among them to determine which the best random replacement policy is.

The rest of this paper is organized as follows. Section 2 describes the randomized replacement policies considered in this study as well as the LRU replacement policy. Section 3 illustrates the processing performed over the traffic trace used in the simulations and some statistics. Section 4 explains the simulation considerations as well as the metrics utilized and the performance comparison between the replacement policies. Finally, section 5 shows the conclusions of this work.

2. REPLACEMENT POLICIES

In this section, a detailed explanation of the randomized replacement policies considered in this work is presented. First of all, the LRU replacement is considered because it is the algorithm usually implemented in Web proxy caches and consequently it will be compared to the randomized replacement schemes.

Some of the replacement policies use one or multiple characteristics associated to the documents. Specifically, for a document i, s_i denotes the size of i, c_i represents the cost of retrieving the document i from the original Web server and t_i symbolizes the last request time to document i.

The analyzed replacement policies are:

- LRU: This replacement policy evicts the document that was least recently referenced, that is, the document i with the lowest t_i value. This algorithm was originally developed for memory caching in CPUs and it is based on the supposition that a document that is requested will most probably be requested again in the near future. It has the advantage that it is a very simple algorithm which can

be implemented to work very efficiently. On the other hand it has the handicap that it does not take into account the frequency or size of documents.

- RANDOM: The documents to be evicted are randomly selected using a uniform distribution, that is, all documents have the same probability of being evicted. It is a very simple replacement policy but it shares some of the disadvantages of LRU.
- HARMONIC (Hosseini-Khayat, 1997): The probability of a document *i* to be evicted is shown in Eq. 1. Where *N* is the number of documents stored in the cache. Therefore, the probability of a document to be evicted is higher as the document size increases and it is inversely proportional to the cost of retrieving the document.

$$P(i) \propto \frac{s_i}{c_i} \qquad \forall i \in [1..N] \qquad \text{Eq. 1}$$

- LRU-C (Starobinski & Tse, 2001): It is a randomized version of LRU. In this algorithm, when a document that is in the cache is requested again, the probability of being moved to the head of the LRU queue, that is, to the most recently used position is presented in Eq. 2, where the denominator represents the maximum cost of the *N* documents that the cache contains. This equation assigns a normalized probability to be evicted to each document depending on the cost of retrieving it. Those documents with the higher cost are more probably moved to the head of the LRU queue.

$$P(i) \propto \frac{c_i}{max[c_1, c_2, \ldots c_N]} \qquad 1 \leq i \leq N \qquad \text{Eq. 2}$$

- CLIMB-C (Starobinski & Tse, 2001): It works in a similar way as LRU-C, but when there is a cache hit (i.e. the document is already present in the cache when the request is performed), the document climbs a position in the cache queue, instead of moving to the head of the queue as occurs in LRU, with the probability shown in Eq. 2.
- LRU-S (Starobinski & Tse, 2001): This replacement policy uses the size instead of the cost of the documents to perform the same algorithm as LRU-C. Thus, the probability of a document to be moved to the head of the LRU queue when there is a cache hit is presented in Eq. 3. This equation assigns a normalized probability to be evicted to each document depending on the size. Those documents with higher size are less probably moved to the head of the LRU queue.

$$P(i) \propto \frac{min[s_1, s_2, \ldots s_N]}{s_i} \qquad 1 \leq i \leq N \qquad \text{Eq. 3}$$

- CLIMB-S (Starobinski & Tse, 2001): This replacement policy works like CLIMB-C but using the Eq. 3 as the probability to climb a position in the queue when there is a cache hit.
- RRGVF (Randomized Replacement with General Value Functions) (Psounis & Prabhakar, 2001): This replacement policy selects randomly *N* documents and it evicts those with the lowest useful value. The selection of the usefulness function is not part of the algorithm and it could be selected as desired. The rest *M* (with $M < N$) of documents not evicted are maintained in memory and $N - M$ documents are selected randomly from the cache. In the next eviction, the lowest useful documents are chosen taking into account those *M* and $N - M$ documents. The relationship between the value of *N* and *M* is presented using Eq. 4, where the parameter *n* is a factor that represents the error. An error is presented when the evicted document is not part of the *n%* of the least useful documents in the cache.

$$M = N - \sqrt{\frac{100 \cdot (N+1)}{n}} \qquad \text{Eq. 4}$$

3. WORKLOAD CHARACTERIZATION

To evaluate the performance of the replacement policies mentioned in the previous section, a workload trace that contains HTTP requests from a proxy of the IRCache project has been utilised (IRCache Home,

2004).This proxy is located in the Research Triangle Park (North Carolina, USA). The traces include requests from the 7th to the 11th of June 2004 generated by the Squid Web proxy cache software (Squid Proxy Cache, 2004) The traces include information for each HTTP request processed by the proxy, such as the time the request was initiated, the document size, the URL, the document type (content-type), the request method (GET, POST, …) and the response code from the server.

This trace has been preprocessed to purge those requests that have been generated dynamically by CGI (Common Gateway Interface) because the documents returned by these kind of requests are unique for each request and therefore they should not be cached (Zhang, 2000). Because of this fact, the requests that contain the strings 'cgi', 'cgi-bin' or '?' have been discarded. Those requests that contain the string ':3128' have been filtered as this is the port that IRCache utilises to interchange information between collaborating caches and hence they must not be cached. As cacheable response codes, 200 (OK), 203 (Partial), 206 (Partial Content), 300 (Multiple Choices), 301 (Moved) and 302 (Redirects) have been taken into account. For the 304 (Not Modified) response code, the size shown in the traces corresponds to the size of the response and not to the real document size (it informs that the document version in the client cache has not been modified since it was requested last time), consequently these documents have been requested again to the original server to obtain the real size. The data of the original trace has been filtered and only the time of the request, document identification, size and content-type have been considered. A unique numeric identification has been assigned to each document in order to simplify and speed the simulation process.

Table 1 summarises the basic characteristics of the workload after its process.

Table 1. Main workload characteristics

Number of Requests	4,040,036
Size (GB)	40.4
Distinct documents (%)	42.4
One-timers (%)	32.1
Mean (bytes)	10,006
Median (bytes)	1,720

The distinct documents parameter shows the number of documents that are requested in the workload as a function of the total number of requests. Therefore, as the distinct documents percentage increases the number of requests to the same documents decreases and hence the performance of the cache also decreases. The one-timers are those documents that are requested only once in the workload and consequently they are not useful to be cached because they are not going to be requested again when they are stored in the cache. As this parameter increases, the performance and usefulness of the cache decreases.

Figure 1 represents the histogram of the number of requests to documents as a function of the document sizes. As it can be observed, when the document size increases the number of references to the documents decreases and the probability of being requested again decreases. That is the reason why some replacement policies such as LRU-S evict those documents with the highest size.

To characterize the popularity of the documents Zipf law has been utilized (Breslau, Cao, Fan, & Phillips, 1998) (Breslau, Cao, Phillips, & Shenker, 1999). This law asserts that the probability *P(i)* for the *i-th* most popular document to be requested is inversely proportional to its popularity ranking as shown in Eq. 5.

$$P(i) \propto \frac{\beta}{i^{\alpha}} \quad with\ \alpha\ close\ to\ 1 \qquad \text{Eq. 5}$$

The parameter α is the slope of the log/log representation of the number of references to the documents as a function of its popularity rank and the β parameter is the displacement of the function. Figure 2 illustrates this representation where the slope of the function has been calculated using a minimum square regression obtaining a value of α of 0.64. As this parameter is close to one, the number of repeated references increases. The popularity is a good estimator of the cachability of documents because a document that is very popular is more probable to be referenced again in the near future, so the probability of cache hit increases.

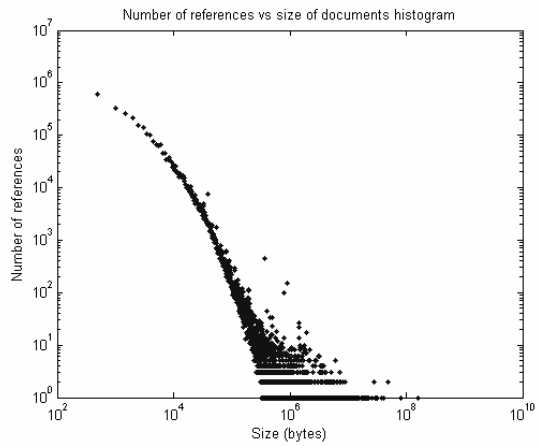

Figure 1. Histogram of the number of references as a function of document sizes

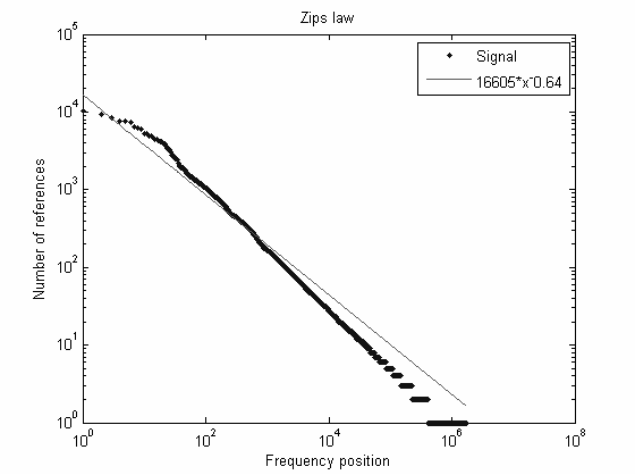

Figure 2. Calculation of Zipfs law coefficients

4. SIMULATIONS

The metrics used to evaluate and compare the performance of the replacement policies considered in this work are the classical HR (Hit Ratio) and BHR (Byte Hit Ratio) defined as follows:

- HR: It is the total number of requests that cause a hit in the cache divided by the total number of requests.
- BHR: It is defined as the summation of the document sizes that cause a hit in the cache divided by the size of the processed documents.

The HR gives us an idea of the reduction of the latency perceived by the users because the requests will be served sooner from the cache than the original server since the cache is located near the user. On the other hand, the BHR indicates the saved bandwidth between the proxy cache and the original Web servers.

In order to perform high-quality simulations some considerations have to be taken into account. The first consideration is the 'warm up' parameter that is the time the cache is going to be working before the performance metric starts. If we start measuring the performance when the cache is empty the number of cache misses (i.e. the document is not in the cache) will give us a distorted value of the performance thus the measuring should start at least when the cache is full and a few replacements have been done.

Another subject to be taken into account is how to distinguish the modification of the documents. According to (Arlitt, Friedrich, & Jin, 1999) if the difference between sizes of successive requests to the

same document is less than 5% of the document size, a modification of the document can be considered and it has to be treated as a new document; otherwise a cancel of the transfer is considered.

In order to perform the simulation of the replacement policies using the traces analyzed in the previous section, a simulator has been designed. This simulator has been developed using the C++ language and the Borland C++ Builder IDE and it takes the filtered traces as input and executes the simulation according to some parameters such as the replacement policy, the storage space assigned to the cache and the 'warm-up' time. As well as the replacement policies mentioned in section 2, this simulator also implements some more such as LFU (Least Frequently Used), LFU-DA (LFU Dynamic-Aging) and GDSF (Greedy-Dual Size with Frequency).

Once the simulations are finished the simulator provides a file with statistics such as the HR, BHR, number of documents modified, total number of documents in the cache and final size occupied in the cache.

To perform the evaluation we first simulated a cache with infinite size to determine the total size filled in the cache. The next simulations were performed using the 40%, 30%, 20%, 10% and 5% of this maximum size. Due to the randomized nature of the replacement policies studied, the simulations have been executed five times per replacement policy in order to obtain an average performance. Finally, a 50% of the workload was used to 'warm-up' the cache.

Some of the replacement policies include parameters or cost functions that have to be assigned. In that way for the HARMONIC replacement policy we have considered a constant cost function, for the LRU-C the cost function is the number of packets needed to transfer the document and finally, the RRGVF algorithm will consider the number of references to the document as the utility function and values of N=30 and M=12 for the parameters of the replacement policy (Psounis & Prabhakar, 2001).

Figure 3 and figure 4 shows the HR and BHR respectively of the eight replacement policies evaluated as a function of the cache size. In these figures only the average performance of the randomized replacement policies has been depicted to clarify the figures, but the variation over the five simulations were in the range of only 1%.

For the HR metric HARMONIC clearly outperforms the other schemes. The difference with the others increases as the cache size decreases. The second best policy is RRGVF with a performance slightly better than LRU. Finally RANDOM, LRU-C, LRU-S, CLIMB-C and CLIMB-S obtain a similar performance but lower than HARMONIC, LRU and RRGVF

For the BHR metric RRGVF outperforms the other replacement policies although LRU obtains lower but similar results. The rest of the schemes achieve analogous behaviour but worse that RRGVF and LRU. Even HARMONIC obtains the worst results for small caches.

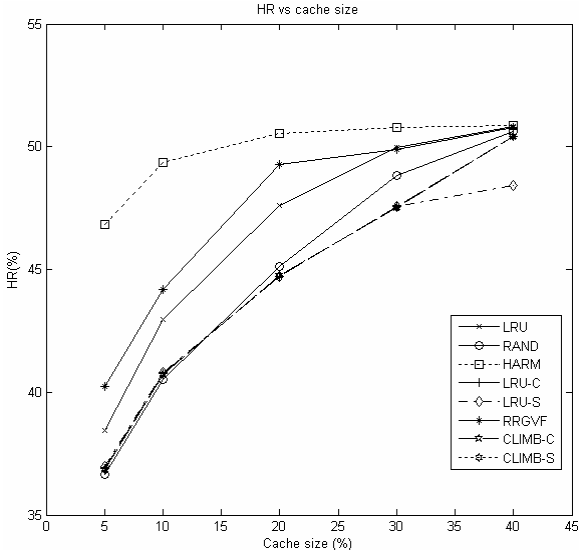

Figure 3. HR versus cache size

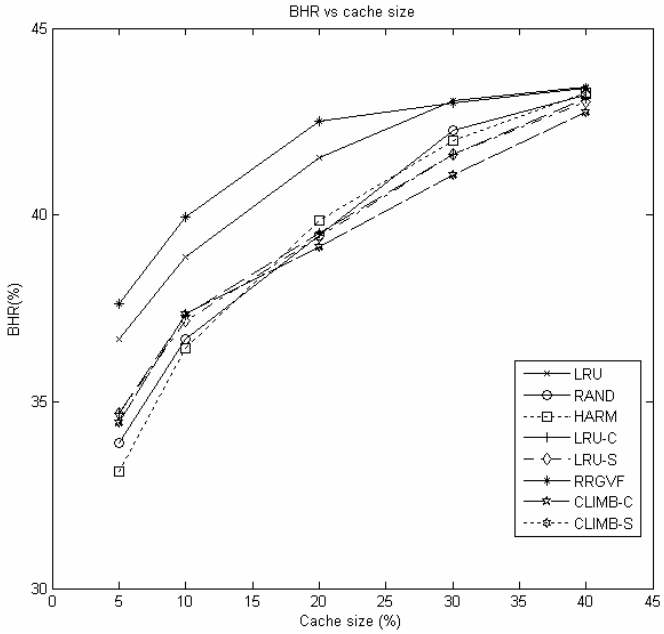

Figure 4. BHR versus cache size

As can be observed there is not a replacement policy that outperforms the others for both metrics. If we want to maximize the HR, the HARMONIC replacement policy should be chosen but we will also obtain poor BHR. If we want to maximize the BHR the RRGVF algorithm should be chosen and we will obtain a good HR.

Anyway, using both metrics the LRU never obtains the best performance. That will be a good reason to use a replacement policy such as RRGVF in the proxy caches instead of the LRU. Furthermore the complexity of those algorithms is in the same order as LRU meanwhile they obtain better performance.

5. CONCLUSIONS

In this work we have evaluated the performance of seven randomized replacement policies and we have compared them to the LRU scheme widely used in the Web proxy caches. To perform this evaluation a cache simulator that implements the replacement policies mentioned has been developed. The performance has been measured using the classical HR and BHR metrics. The workload driven simulations have shown that to maximize the HR the HARMONIC replacement policy is the best choice although it obtains poor BHR. Considering the BHR, RRGVF obtains the best performance for all cache sizes. Because this scheme also obtains a good HR it will be adequate to maximize both metrics. On the other hand RRGVF outperforms LRU for both metrics. Therefore, it will be a good choice to use the RRGVF replacement policy instead of LRU in the proxy caches because they have a comparable computational complexity.

ACKNOWLEDGEMENT

We would like to thank Duane Wessels for the access to the workload traces. This work has been partially supported by the public project TEC2006-12211-C02-0.

REFERENCES

Arlitt, M. et al, 1999, *Workload Characterization of a Web Proxy in a Cable Modem Environment,* Hewlet-Packard Laboratories. Technical Report HPL-1999-48.

Balamash, A. and Krunz, M., 2004, An Overview of Web Caching Replacement Algorithms, *IEEE Communications Surveys and Tutorials* , Vol. 6, No. 2, pp. 44-56.

Breslau, L. et al, 1998, On the Implications of Zipf's Law for Web Caching, *3rd International WWW Caching Workshop,* Manchester, England.

Breslau, L. et al, 1999, Web Caching and Zipf-like Distributions: Evidence and Implications*, IEEE Infocom, XX.*

Hosseini-Khayat, S., 1997, *Investigation of generalized caching.* Washington: Ph.D. dissertation.

IRCache Home Page. http://www.ircache.net

Jin, S. and Bestabros, A., 2001, GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams. *Journal of Computer Communications ,* Vol. 24, No. 2, pp. 174-183.

Khayari, R., 2003, *Workload-Driven Design and Evaluation of Web-Based Systems.* Osnabrueck, Germany: Der Andere Verlag.

Khayari, R. et al, 2005, Impact of Document Types on the Performance of Caching Algorithms in WWW Proxies: A Trace Driven Simulation Study. *IEEE 19th International Conference on Advanced Information Networking and Applications (AINA 2005).* Taiwan.

Luotonen, A. and Altis, K., 1994, World-Wide Web Proxies. *Computer Networks and ISDN Systems ,* Vol. 27, No. 4, pp. 147-154.

Nagaraj, S., 2004, *Web Caching and its Applications.* Netherland: Kluwer Academic Publishers.

Podlipnig, S. and Böszörmenyi, L., 2003, A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys,* Vol. 35, No 4, pp374-398.

Psounis, K. and Prabhakar, B., 2001, A randomized Web-cache replacement scheme, *In Proceedings of the IEEE INFOCOM*, pp. 1407–1415

Squid Proxy Cache Home Page. http://www.squid-cache.org

Starobinski, D. and Tse, D., 2001, Probabilistic methods forWeb caching. *Performance Evaluation ,* Vol. 46, pp. 125–137.

Wessels, D., 2001, *Web Caching,* O'Reilly.

Zhang, X., 2000, *Cachability of Web Objects.* Technical Report 2000-19.