Evolving Systems (2018) 9:245–254
DOI 10.1007/s12530-017-9194-8

ORIGINAL PAPER

Simulation evaluation of a relative frequency metric for web cache replacement policies

Nadjette Benhamida[1,2] · Louiza Bouallouche-Medjkoune[2] · Djamil Aïssani[2]

Received: 24 February 2016 / Accepted: 16 June 2017 / Published online: 1 July 2017

Abstract In this paper, we interest in web cache replacement policies namely "frequency-aware" policies that provide, generally, the best results in term of data movement reduction in the network. For the simple reason that they take into account one of the most significant web traffic characteristic "the access frequency". However, the access frequency suffers from two main problems namely one-timer documents existence and cache pollution. Therefore, our aim is to replace the traditional frequency with a relative frequency; calculated using the access number and the document lifetime in the cache. Although the idea already exists in the literature, we strive to validate the relative frequency efficiency for the web proxy replacement policies. In this work, we implement three replacement policies namely least frequently used (LFU), least frequently used with dynamic aging (LFU-DA) and Greedy dual size frequency (GDSF). As well, their versions enhanced with relative frequency namely LFRU, LFRU-DA and GDSFR respectively are implemented and evaluated using synthetic and real workload. The simulation results show that the relative frequency is more effective, in terms of hit rate and byte hit rate, than the access number; i.e., the traditional frequency. Moreover, the simulation proves that the relative frequency solves the access frequency problems.

Keywords Caching · Performance evaluation · Proxy web cache · Replacement policies · WWW

✉ Nadjette Benhamida
benhamida_nadjette@yahoo.fr

[1] Doctoral School on Networking and Distributed Systems (ReSyD), Faculty of Exact Sciences, University of Bejaia, 06000 Bejaïa, Algeria

[2] Research Unit LaMOS, University of Bejaia, Bejaïa, Algeria

1 Introduction

World Wide Web (WWW) requests have continued to grow at an exponential rate for the reason that it, already, constitutes the dominant workload component for internet traffic (Wang 1999). The main implications related to this fact are the fastest and the easiest internet information access, the cheapest in use, the variety of information domains such as: scientific research, education, sport, teaching, business, etc. (Wang 1999). Also, the physical infrastructure internet network evolution represents another very important reason of this growth.

Since the majority of web objects are requested by different clients, caching them improves, significantly, the web performances by reducing (1) the server overload, (2) the data volume transferred between servers and clients, (3) the band-width consumption, (4) the network overload surcharge, (5) the clients' average latency. Also, the caching improves the information availability.

Nevertheless, the caching may increase, in some situations, (1) the data inconsistency rate, (2) the client latency in the case of a miss, and (3) the clients' average latency because of the limited number of clients per proxy. Also, the original server cannot have a clear idea about the clients' access model and the most popular files.

Therefore, caching has become the important variable to fully actualize the future internet dream by carefully solving the problems in frequency and recency in objects (Abdullahi et al. 2015). The main key of the proxy cache efficiency is the document replacement policy that can reach high performances. Determining which part of the content is to be cached? What is the most appropriate time for caching? How the object would be cached (placed and replaced) and also at what path would the object be cached? (Abdullahi et al. 2015). All this is the main role of

a replacement policy; i.e., it decides which documents to keep in the cache, which documents to replace and which future requests will have a cache hits. Many web caching policies have been proposed, in the literature, based on various web traffic characteristics such as: the document size, the access frequency, the access recency, the clients' average latency, etc.

Hence, frequency-aware replacement policies are the most effective, in the literature, for the reason that they take into account an important traffic characteristic that is the popularity of documents. However, the access frequency performances are reduced through two principals' disadvantages namely the cache pollution problem and the one-timer documents existing.

In this paper, we enhance the cache policies with a new principle called "the documents lifetime" in the cache. It provides an exact idea about the access frequency and recency. Hence, we replace the access frequency a relative frequency in the policies namely frequency-aware in order to improve the web caching performances.

The paper is made up of five sections. Section 2 presents the related works. Section 3 defines the details of our proposal. Then, Sect. 4 presents the evaluation of this work. Finally, Sect. 5 concludes the paper.

2 Related work

In a dynamically changing and evolving environment namely "the web", the unpredictable requests continue to evolve daily. For this reason, several web caching replacement policies have been proposed in the literature and they achieve an admissible work. Nonetheless, the question that arises now is "is there a better policy forever and in all domains?" which can provide the best performances. The answer is "there is not a policy that is the best forever (Wang 1999)". However, each policy can improve the web performances if it takes into a count the principal web documents characteristics such as the access frequency, the recency, the size, etc.

Moreover, various additional web traffic characterizations influence the web performances such as: the one-time referencing, the file types, the file popularity, the file size distribution, the correlation between the file size and the popularity, the temporal locality, the references concentration and the hot set drift analysis. Two types of replacement policies exist in the literature: the basic policies and their enhanced versions.

2.1 Basic policies

The basic policies are size (Abrams et al. 1995), lowest latency first (LLF) (Cao and Irani 1997), first in first out (FIFO) (Yanev 2005), random (Yanev 2005), LFU (Yanev 2005) and least recently used (LRU) (Yanev 2005; Arlitt et al. 1999). Generally, they perform well like LRU (Do et al. 2014). However, they have one main inconvenient that they rely, only, on one traffic characteristic, such as the access frequency, the recency, the latency or a randomized choice. Therefore, their performances are limited. For this reason, several enhanced versions of the basic policies are proposed and implemented.

2.2 Basic policies enhanced versions

Today, there exist various replacement policies based on several web document characteristics. For example, SzLFU(k) policy (Yanev 2005) is based on the access frequency and the document size. The main inconvenient of the SzLFU policy are: (1) the prior choice of the parameter k, which depends on the access model, and (2) its vulnerability to the cache pollution.

Alternative functions-aware policies have been proposed such as lowest relative value first (LRV) (Wang 1999), hybrid (Wang 1999), Greedy dual size (GDS) (Arlitt et al. 1998), Greedy dual size frequency (GDSF) (Arlitt et al. 1999), LFU-dynamic aging (LFU-DA) (Arlitt et al. 1999), Least weighted usage (LWU) (cohen and dabran 2002), MIX (Ali et al. 2011a, b), least recently frequently used (LRFU) (Lee et al. 2001), Greedy dual* (GD*) (Vakali 2000) and history LRU (HLRU) (Vakali 2000). All these policies use functions calculated using several parameters and involve several additional treatments compared to the basic policies.

Moreover, several solutions, based on the access recency and the document size, exist in the literature such as LRU-threshold (Abrams et al. 1995), LRU-size (Abrams et al. 1995), LRU-min (Wang 1999; Cao and Irani 1997), Log(size) + LRU (Cao and Irani 1997) and Pitkow and Recker (1994). Their main inconvenient is that they remove from the cache the biggest documents. In this case, these policies risk of removing the most popular documents or documents having the great cost of downloading.

Others proposed policies are based on the access frequency and the recency such as segmented LRU (SLRU) (Arlitt et al. 1998; ElAarag 2013), LFU-aging (Vakali 2000), LFU*-aging (Arlitt and Williamson 1997), LRU-K (Arlitt et al. 1998) and frequency based replacement (FBR) (Arlitt and Williamson 1997). The main problem of the cited policies is the choice of the parameters values that can provide the best results.

Moreover, all the previous policies, cited in this section, increase the client latency and the space reserved to its execution compared to the basic policies.

2.3 Most recent works

In Ali et al. (2011a, b), several intelligent web caching algorithms have been presented and summarized. In addition, authors in Sajeev and Sebastian (2011) have proposed a novel classification scheme for web cache objects that uses a multinomial logistic regression (MLR) technique. In Ali et al. (2014), LRU policy have been enhanced using popular supervised machine learning techniques such as a support vector machine, a naïve Bayes classifier and a decision tree. Furthermore, authors in (Kumar and Norris 2008) have proposed a new proxy-level web caching mechanism that takes into account aggregate patterns observed in user object requests. Also, authors in (Kaya et al. 2009) have evaluated an admission-control policy for proxy server caching that augments the LRU algorithm. In (Ali et al. 2011), support vector machine (SVM) is used to enhance the performance of conventional web proxy caching such as LRU and GDSF. Moreover, a replacement policy has been proposed based on mobile insertion position "MIP" (Abad et al. 2015).

2.4 Frequency-aware policies performances discussion

LFU versions, which exist in the literature, use several specific parameters or complex function compared to the basic policies, which decreases their performances. For example, the LFU-aging and LFU*-aging policies use two additional parameters the references maximum value ($M_{ref)}$ and the documents average age ($A_{max)}$. LRU-K policy uses the parameter K that requires the storage of K last access dates of each document in the cache, LFU-DA policy is based on the parameter K and its choice is very difficult. SLFU policy uses an additional parameter to define the percentage of each partition space that augments the policy complexity and execution time. LRFU is based principally on a function which determines the influence of the access frequency and recency on the probability that a document will be requested in the future. HLRU uses the parameter h which requires the storage of the h last access dates for each document. FBR is based, principally, on the two parameters F_{new} and F_{old} used in partitioning the cache space.

The main common inconvenient of all these policies is the choice of the parameter values that can provide the best performances. Moreover, they increase the volume of data transferred in the network (as SLFU policy). However, policies based on a performance index such as LWU and LRV increase the clients' average latency.

3 Proposal

3.1 Problematic formulation

LFU policy and their enhanced versions are among the most-used replacement policies and the most effective efficient in the literature. This policy provides better performances than other basic policies such as Random and FIFO. The reason is that it takes into account an important traffic characteristic which is the documents popularity. However, LFU has two principal deficiencies that affect its performances: the cache pollution problem and the existence of one-timer documents. The first point means that LFU prevents unpopular documents having large frequency from being removed. The second point means that the one-timer documents remain in the cache for a long period. Several LFU versions have been proposed in order to improve the performances by taking into account LFU benefits and other traffic characteristics such as the access recency, the document size, etc. Unfortunately, that increases the execution time of these policies and consequently the client latency.

Also, another interesting feature is that the document having the greatest frequency is not always the most popular one, because its frequency can be obtained during a caching time (its lifetime) larger than the lifetime of other documents having lower frequencies.

In order to solve the problems of frequency-aware policies previously explained, we propose a new mechanism namely "the relative frequency". This mechanism will replace the access frequency, which means the access number, in the frequency-aware policies.

3.2 Solution based relative frequency

In this paper, we propose as a novelty a new mechanism to replace the traditional access frequency with a relative frequency calculated using the documents access number and its lifetime in the cache. Consequently, our aim is to ensure an improvement of the frequency-aware policies using a relative frequency (called F^r_i in the Formula 1). The relative frequency depends on the number of access for each document i and calculated as follows:

$$F_i^r = \frac{F_i}{now - D_i}, \tag{1}$$

where D_i is the date of entrance of the document i to the cache, F_i is the document frequency and now is the current system date.

As a result, we obtain the access frequency to a document for each unit of time. In this way, we avoid many

overheads by using a very simple function which does not require many additional treatments and which improves the caches replacement policies performances. Moreover, by using the relative frequency, we will be able to solve the problem related to the one-timer documents. In fact, by using the relative frequency function; the frequencies of one-timer documents remain constant (equal to one) but the documents lifetimes in the cache continue to increase. Consequently, their relative frequencies decrease. Also, the cache pollution problem should disappear for the reason that the relative frequency of the documents having the great values of the access frequency will increase over the time.

3.3 Implementation

The relative frequency is proportional to one of the two following parameters the time or the access number to the cache. In the first case, the relative frequency represents a time period obtained from a physical timer. In the second case, it represents the access occurrences number to the cache after the last access to this document i.e., concept of the logical timer (Formula 2). Therefore, for each period of time t_c we calculate:

$$D(min) = min(Di) \ \forall i \ in \ the \ cache, \tag{2}$$

where D_i represents the access occurrence number to i.

Afterwards, we perform for each document j in the cache:

$$Dj = Dj - D(min). \tag{3}$$

In fact, Formula (3) provides the relative frequency of the document j.

3.4 Illustrative example

Let's assume two documents A and B in the cache, where $frequency\ (A)=3$ and $frequency\ (B)=2$. Whereas, the lifetime of A in the cache is 10 time units and the lifetime of B is 4 time units. For the reason that $frequency(A) > frequency(B)$, the frequency-aware policies will remove B before A. However, actually B is more popular than A because $F_A^r = 3/10 = 0.3$ and $F_b^r = 2/4 = 0.5$. Consequently, B is the most probably re-requested in the future.

3.5 Relative frequency advantages

We summarize the advantages of the relative frequency solution as follows:

- It solves the cache pollution problem, because it depends on the documents lifetime. Therefore, it will be impossible to find a document with a great relative frequency during all its lifetime in the cache.
- It solves the problem of the existence of one-timer documents for a long period of time in the cache for the reason that the lifetime increases and the frequency still equal to one. Consequently, the one-timer documents will disappear.
- It provides a very clear idea on the documents popularity in the cache proxy. Indeed, it is possible to compare documents according to the relative frequencies because the access frequency of every document is given by one unit of time.
- It reduces the client latency because it increases the number of requested served directly from the cache. Consequently, the information availability will increase.
- It reduces the bandwidth consumption for the reason that it decreases the network traffic.
- It reduces the servers overload because it decreases the number of request sent towards the web servers.

4 Evaluation

In order to evaluate the efficiency of our solution, we implement our simulator in C++ language. Three frequency-aware policies are implemented LFU, LFU-DA and GDSF in addition to their versions enhanced with relative frequency namely LFRU, LFRU-DA and GDSFR respectively. In order to validate our simulator, we use short log files that can be checked by hand. Afterwards, we compare the results obtained in the both cases. This procedure is repeated several times. Moreover, we compare our original policies results with the results of some existing studies.

Furthermore, we use both simulations; execution-driven and trace-driven. Execution-driven simulation is used in order to demonstrate the efficiency of our solution in the proxy caches. In fact, the web traffic used, in this case, is generated using ProWGen (Busari and Williamson 2002). Knowing that, authors in (Williams et al. 2005) confirm that there are no dramatic changes in web server workload characteristics. The obtained results are presented and discussed in the Sect. 4.3.1.

Trace-driven simulation is used in order to reveal the efficiency of the relative frequency for mobile clients. Consequently, a real log are recorded and used by our simulator. The results obtained are presented and discussed in the Sect. 4.3.2.

Moreover, a real experimental test is used in order to prove the efficiency of our solution and the exactitude of the simulation results mentioned in this paper.

4.1 Performances metrics

In order to evaluate our simulator, we use two metrics called the hit rate and the byte hit rate because they are the most used in this domain (ElAarag 2013).

The hit rate (*HR*): provides the percentage of the requests served directly from the cache among the total number of requests (Formula 4) (Cao and Irani 1997; Arlitt et al. 1999):

$$HR = \frac{\sum_{i=1}^{m} \theta_i}{m}, \tag{4}$$

where m is the total number of requests, θ is a parameter that determines if the requested document exists in the cache (=1) or does not exist (=0).

The byte hit rate (*BHR*): provides the percentage of bytes transferred directly to clients from the cache among the total size of requests (Formula 5) (Cao and Irani 1997; Arlitt et al. 1998, 1999):

$$BHR = \frac{\sum_{i=1}^{m} size_i\, \theta_i}{\sum_{i=1}^{m} size_i}, \tag{5}$$

where $size_i$ is the size of the documents i.

4.2 Model and factors

In this study, we use a model with one level of web proxy (Fig. 1). This model operates as follows, when the proxy receives a client request, it must search the requested file in its cache. In the case of a hit, the answer is turned to the applicant else the required document must be requested from the original server. While receiving the server response, the proxy must send it to the applicant and store a copy in its cache. If the free space is not sufficient for this new document, a replacement policy must be executed in order to release space. Therefore, in our model, the replacement policy is always executed in demand.

Moreover, various factors influence the replacement policies performances such as:

4.2.1 Document size

When the documents that visit the cache have small sizes, the cache capacity (in term of the number of files that it can contain) is increased. This increases the requests hit rate and the opposite is true. For this reason, it is very important to respect the real distribution of the web document size. In fact, this distribution is of type "heavy-tailed" (Busari and Williamson 2002). The Fig. 2 shows the document size distribution in our simulation.

4.2.2 Requests number/time

The number of used requests generated in each interval of time (100 s) is shown in Fig. 3.

4.2.3 Cache size

Obviously, the cache performances depend on its size; when the cache size increases the replacement policies performances increase (Figs. 4, 5). However, when the cache size increases, extremely, the entire replacement policies have the same performances and the documents time access

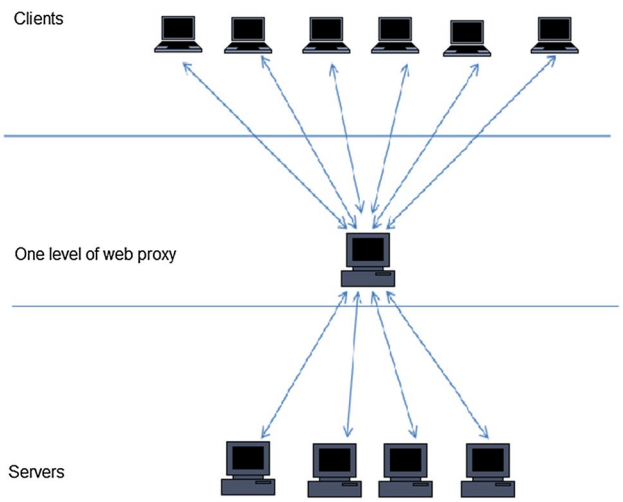

Fig. 1 The web proxy model

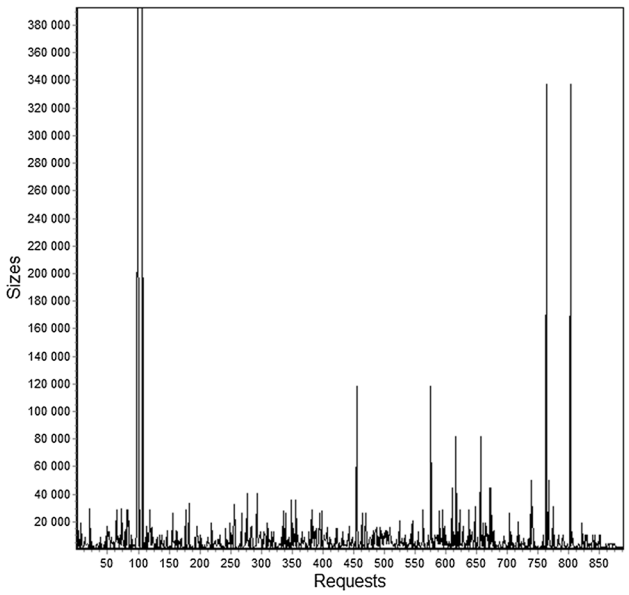

Fig. 2 The document sizes distribution

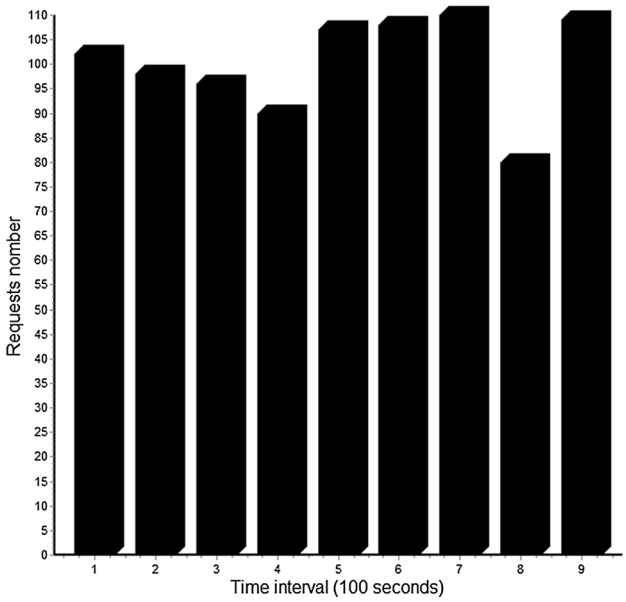

Fig. 3 The requests number distribution / 100 s

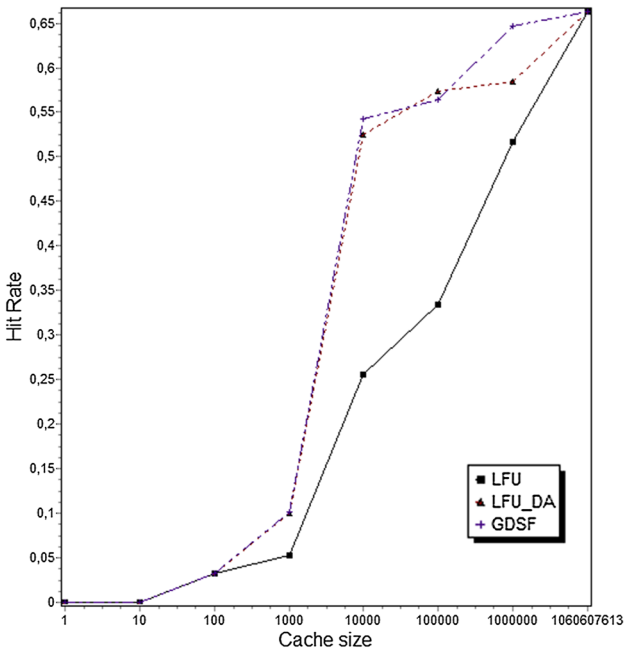

Fig. 4 The relation between the cache size and the hit rate

increases (search time, processing time, etc.). In this case, the advantages of using caches memories may decrease.

4.2.4 Warm-up period

In this period, the cache remains not completely full and the majority of misses appeared during this period are of type

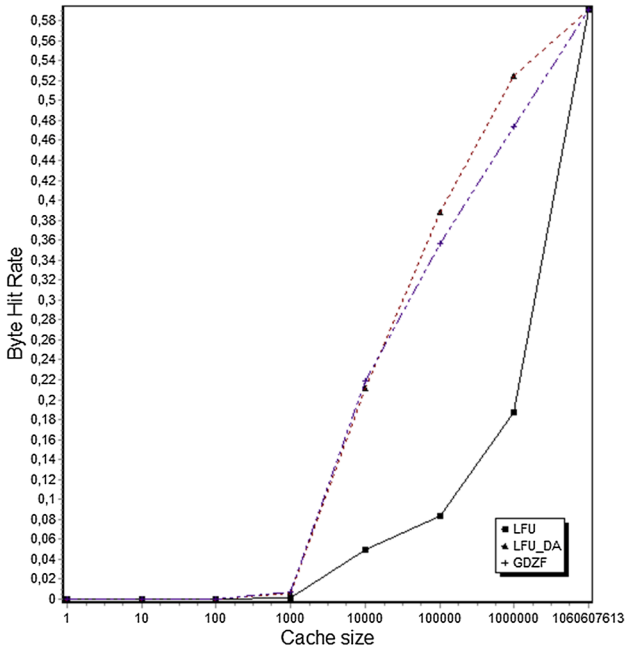

Fig. 5 The relation between the cache size and the byte hit rate

"cold miss". Moreover, it is possible to find a few misses of type "consistency miss" but misses of type "capacity miss" do not appear. Warm-up period depends basically on the sizes of documents referenced in this period and the cache size. The Figs. 4 and 5 show that the replacement policies performances have two phases: during warm-up period [0, 100] and after this period [100, 10,000,000]. During the first phase, all the replacement policies have the same performances and have weakest values, because the cache was nearly empty, in comparison with the second phase.

4.2.5 Replacement policy

After having fixed the cache size and when the simulation exceeds the warm-up period, the replacement policy used in the cache has a very significant role in the web performances. This policy chooses which document(s) must quit the cache in order to release space for the new document recently required. The cache performances change from one policy to another. Therefore, it is necessary to well choose the policy that will provide the best achievable performances.

The original policies implemented in our simulator are LFU, LFU-DA and GDSF (Figs. 4, 5). It is clear that the GDSF policy provides the highest hit rate (Fig. 4) because it takes into account the documents size, the access frequency and recency. This policy replaces the documents with the biggest sizes. Consequently, it increases the

number of small documents which remain in the cache. As a result, the number of requests served directly from the cache is increased. LFU-DA policy provides lower hit rate than GDSF because it does not take into account the documents sizes.

At the same time, LFU-DA hit rate is very high compared to LFU policy because it considers the access recency by using the aging factor. However, LFU-DA provides better results than GDSF and LFU respectively (Fig. 5) in term of byte hit rate. For the reason that LFU-DA keeps in the cache the documents of big sizes, it increases the number of bytes served directly from the cache. Whereas, GDSF policy removes the largest files, reduces the number of bytes served from the cache and increases the volume of data transferred towards proxy and the server.

4.3 Performances analysis

In order to evaluate the relative frequency performances, we present in the next section the original policies performances in terms of hit rate and byte hit rate. Afterwards, we introduce the performances of the relative frequency aware policies namely LFRU, LFRU-DA and GDSFR. Finally, we compare the performances of the whole policies implemented in our simulator. It is logical that all the replacement policies have the same performances during the warm-up period for the reason that the cache is not yet full. However, their efficiency appears after this period.

4.3.1 Comparison according to the requests number

In this section, we exhibit the replacement policies performances in terms of hit rate and byte hit rate by varying the requests number.

4.3.1.1 LFU, LFU-DA and GDSF results Figure 6 shows that GDSF policy provides the best performances in term of hit rate followed by LFU-DA and LFU respectively. Whereas, LFU-DA provides the best byte hit rate (Fig. 7) followed by GDSF and LFU respectively. In fact, GDSF and LFU-DA provide very close performances unlike LFU performances. Consequently, we conclude that the performances of a replacement policy based on several web traffic characteristics are better than those of another policy based on less number of characteristics.

4.3.1.2 LFRU, LFRU-DA and GDSFR results Figures 8 and 9 show the hit rate and the byte hit rate of LFRU, LFRU-DA and GDSFR policies. It is clear that LFRU, LFRU-DA and GDSFR provide very close performances. However, GDSFR is better than LFRU-DA and LFRU respectively in terms of hit rate and byte hit rate. Consequently, we conclude that the use of different keys with this relative frequency provides best performances.

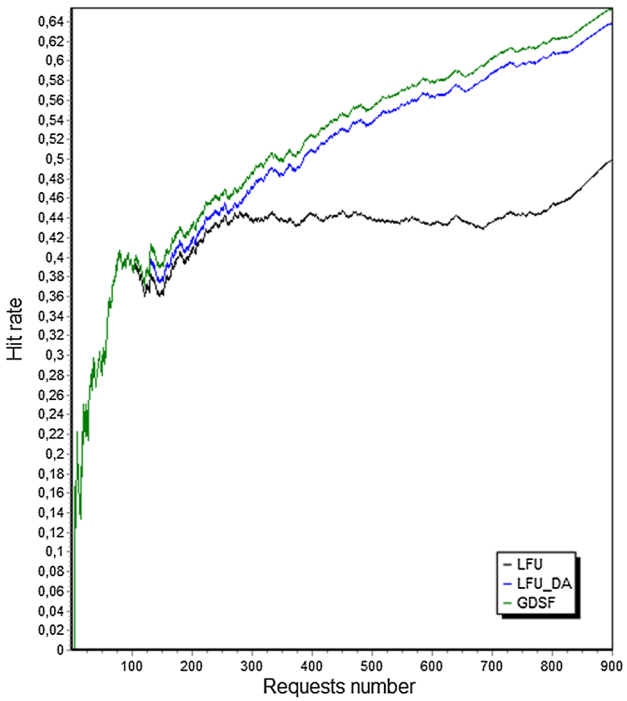

Fig. 6 LFU, LFU-DA and GDSF hit rate

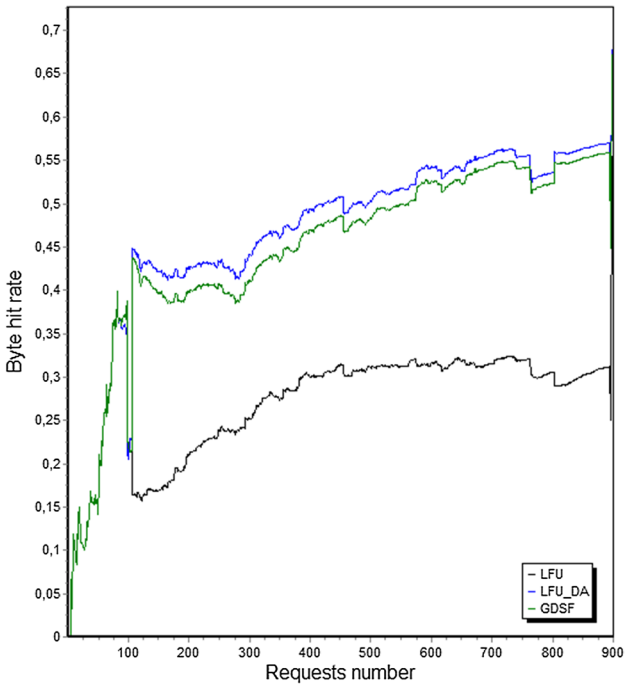

Fig. 7 LFU, LFU-DA and GDSF byte hit rate

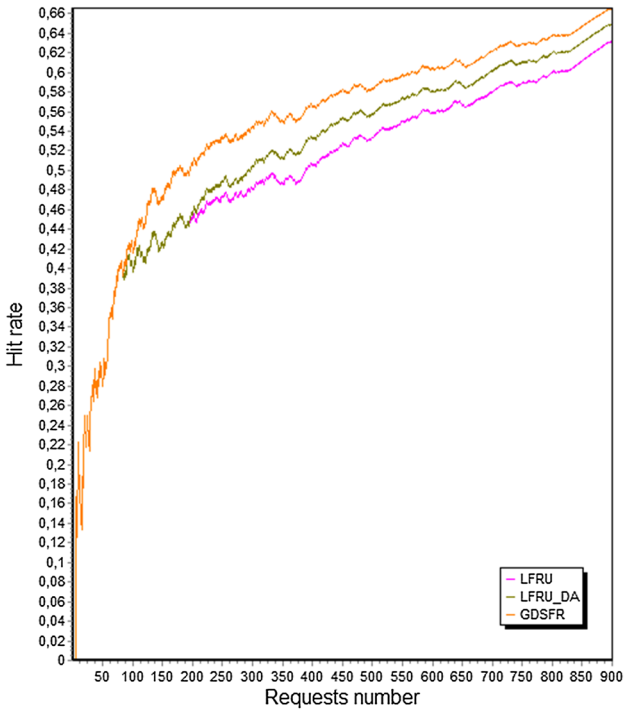

Fig. 8 LFRU, LFRU-DA and GDSFR hit rate

Unlike the Fig. 7, the Fig. 9 shows that GDSFR provides better byte hit rate than LFRU-DA. In fact, the relative frequency corrects the GDSFR weaknesses for the reason that the small documents do not remain in the cache for a long time.

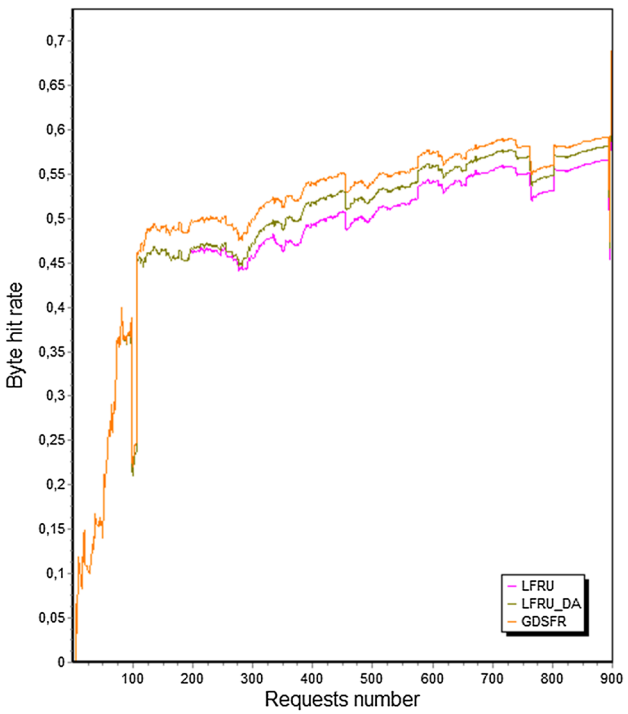

Fig. 9 LFRU, LFRU-DA and GDSFR byte hit rate

4.3.1.3 LFU, LFU-DA, GDSF, LFRU, LFRU-DA and GDSFR results Figure 10 confirms that the relative frequency improves the original policies performances. Hence, GDSFR provides better hit rate compared to LFRU-DA and GDSF respectively. Moreover, the simulation results confirm that the relative frequency can replace several parameters in the function-aware policies. For the reason that LFRU and LFU-DA hit rates are very close. As well, LFRU-DA hit rate is very close to GDSF hit rate.

However, policies based on the relative frequency provide the better byte hit rates than the original policies byte hit rates (Fig. 11). It is also noticed that, LFU policy, which is the basic policy of all these policies, provides the lowest performances. Therefore, the use of a relative frequency instead of the access number improves the performances of the frequency-aware policies. This relative frequency solves the access frequency problems; cache pollution and existing of one-timer documents because of the lifetime documents. Moreover, the Figs. 10 and 11 prove that the replacement policies based on different traffic characteristics provide higher performances such as GDSFR policy.

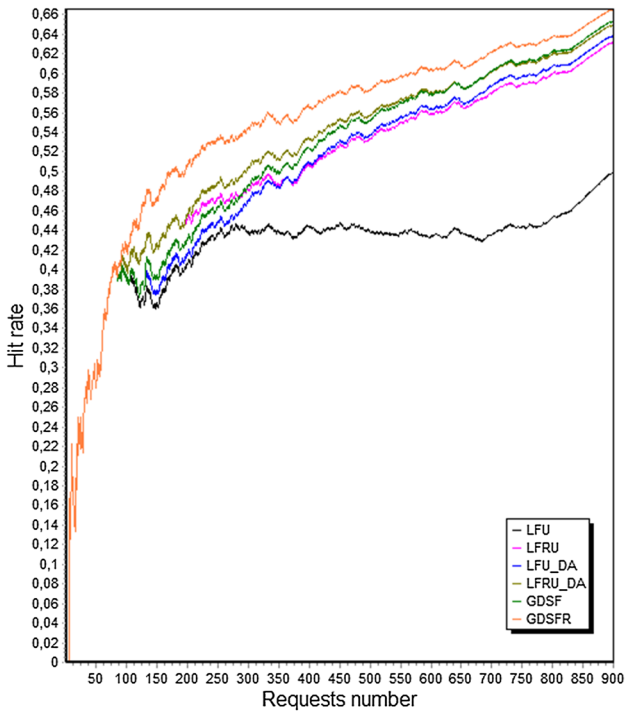

Fig. 10 LFU, LFU-DA, GDSF, LFRU, LFRU-DA and GDSFR hit rate

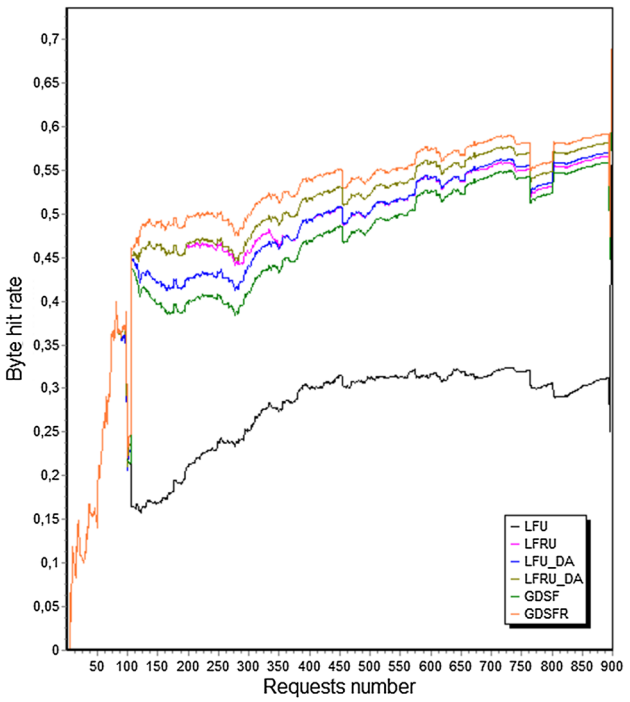

Fig. 11 LFU, LFU-DA, GDSF, LFRU, LFRU-DA and GDSFR byte hit rate

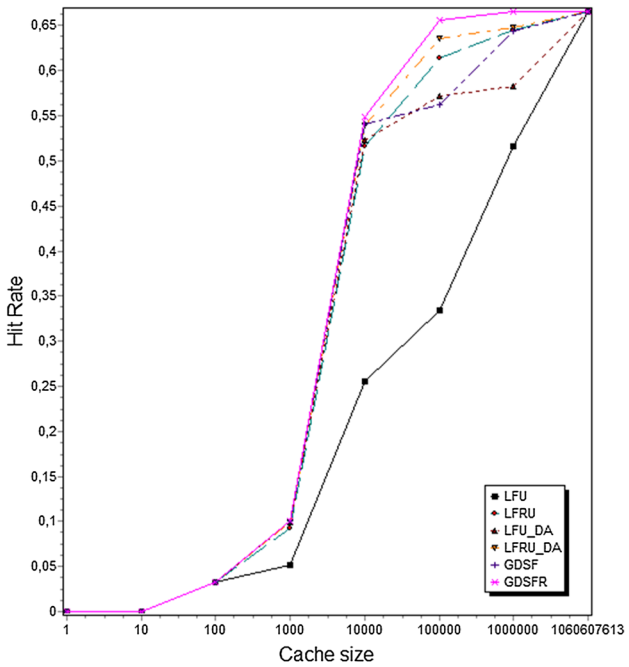

Fig. 12 LFU, LFU-DA, GDSF, LFRU, LFRU-DA and GDSFR hit rate

4.3.2 Comparison according to the cache size

In what follows, we present the results of the replacement policies using a real trace obtained and evaluated according to different cache sizes.

From Figs. 12 and 13, it is clearly visible that GDSFR, LFRU-DA and LFRU, respectively, provide better performances than their original versions in terms of hit rate and byte hit rate. Thus, it is clear that our solution improves the performances of the frequency-aware policies whatever the size of the used cache.

Moreover, policies based on the relative frequency performances always overtake their original policies performances (Figs. 12, 13). In addition, the performances of each relative frequency based policy exceed the performances of all original policies.

Finally, the previous figures prove that the size of caches is limited and cannot exceed a certain value (10,000,000 in Figs. 12, 13). After this value, all policies have the same performances i.e., the max values of hit rate and byte hit rate. However, the client latency will continue to increase.

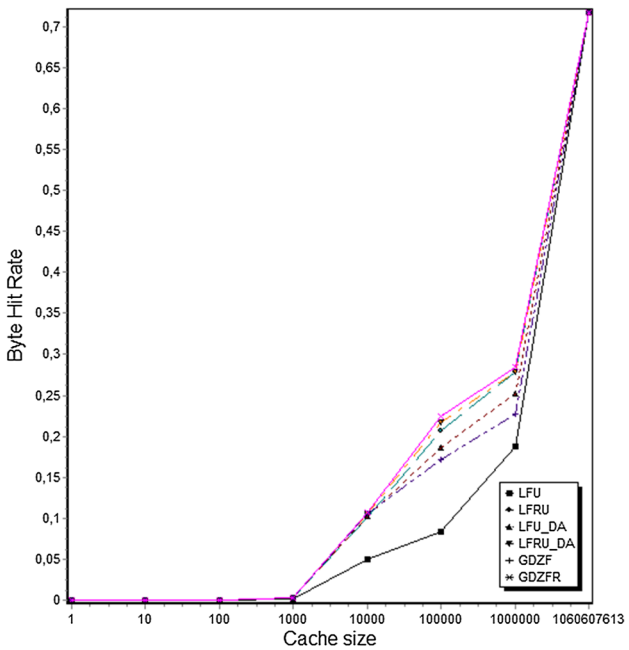

Fig. 13 LFU, LFU-DA, GDSF, LFRU, LFRU-DA and GDSFR byte hit rate

5 Conclusions and future work

The results obtained in our paper are summarized in the following points: firstly, the use of the relative frequency, instead of the access number in the frequency-aware replacement policies, improves the performances of these policies. This is due to the fact that the relative frequency solves the access frequency problems: the existence of the

useless documents namely "one-timer" in the cache and the cache pollution. However, the use of different keys with this relative frequency provides the best performances. Secondly, we demonstrated by simulation that the relative frequency corrects the frequency-aware policies weaknesses because the performances of the relative frequency-aware policies are very close. Furthermore, the replacement policies based on different traffic characteristics provide higher performances such as GDSFR policy. Moreover, the real experimental test proves the efficiency of our solution and the exactitude of the simulation performance.

It will be interesting to study the efficiency of our proposition in the mobile environment such as Ad hoc and sensors networks. Let us notice that the main problems in this type of networks are the availability of information and the minimization of the consumption of energy. We look for designing a very simple algorithm and very efficient replacement which does not require several additional treatments in order to reduce the energy consumption.

References

Abad P, Prieto P, Puente V, Gregorio JA (2015) Improving last level shared cache performance through mobile insertion policies (MIP). Parallel Comput J 49(C):13–27

Abdullahi I, Arif S, Hassan S (2015) Survey on caching approaches in information centric networking. J Netw Comput Appl 56(C):48–59

Abrams M, Stanbridge C, Abdulla G, Williams S, Fox E (1995) Caching proxies: limitation and potentials. In: Proceedings of the 4th international conference on World Wide Web, pp 119–133

Ali W, Shamsuddin SM, Ismail AS. 2011a. A survey of web caching and prefetching. Int J Adv Soft Comput Appl 3(1)

Ali W, Shamsuddin S, Ismail A. 2011b. Intelligent web proxy caching approaches based on support vector machine, informatics engineering and information science book. Springer, Berlin, pp 559–572

Ali W, Sulaiman S, Ahmad N (2014) Performance improvement of least-recently-used policy in web proxy cache replacement using supervised machine learning. Int J Adv Soft Comput Appl 6(1):1–38

Arlitt M, Williamson C (1997) Trace-driven simulation of document caching strategies for internet web servers. Simul J 68(1):23–33

Arlitt M, Friedrich R, Jin T (1998) Performance evaluation of web proxy cache replacement policies, Hewlett-Packard Technical Report "HPL-98-97(R.1)", Internet Systems and Applications Laboratory

Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T (1999) Evaluating content management techniques for web proxy caches. In: Proceeding of the 2nd workshop on internet server performance WISP'99

Busari M, Williamson C (2002) ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches. Comput Networks 38 No(6):779–794

Cao P, Irani S (1997) Cost-aware www proxy caching algorithms. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97), 193–206

Cohen R, Dabran I (2002) The "last copy" approach for distributed cache pruning in a cluster of HTTP proxies*, Book: protocols for high speed networks. In: Proceedings of the 7th IFIP/IEEE international workshop, pp 84–99

Do CT, Kim J, Hwang I, Kim S, Ch Hong Kim (2014) A novel last-level cache replacement policy to improve the performance of mobile systems. Adv Sci Technol Lett 46(Mobile and Wireless 2014):24–28

ElAarag H (2013) A quantitative study of web cache replacement strategies using simulation. In: Web proxy cache replacement strategies. Springer, London, pp 17–60

Kaya CC, Zhang G, Tan Y, Mookerjee VS (2009) An admission-control technique for delay reduction in proxy caching. Decis Support Syst 46(2):594–603

Kumar C, Norris JB (2008) A new approach for a proxy-level web caching mechanism. Decis Support Syst 46(1):52–60

Lee D, Choi J, Kim J, Noh SH, Min SL, Cho Y, Kim CS (2001) LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies. J IEEE Trans Comput 50(12):1352–1361

Pitkow J, Recker M (1994) A simple yet robust caching algorithm based on dynamic access patterns. In: Proceedings of the 2nd international conference on WWW, 1039–1046

Sajeev G, Sebastian M (2011) A novel content classification scheme for web caches. Evol Syst 2(2):101–118

Vakali A (2000) Lru-based algorithms for web cache replacement. In: Proceedings of the first international conference on electronic commerce and web technologies EC-WEB '00, 409–418

Wang J (1999) A survey of web caching schemes for the internet. ACM SIGCOMM Comput Commun Rev 29(5):36–46

Williams A, Arlitt M, Williamson C, Barker K (2005) Web workload characterization: ten years later. Web Inf Syst Eng Internet Technol Book Ser 2:3–21

Yanev K (2005) Cache replacement algorithms in web caches: caching of various media types. Master's Thesis, HELSINKI University, Department of Computer Science