Cache Optimization on Hot-Point Proxy Caching Using Weighted-Rank Cache Replacement Policy

S.P. Ponnusamy and E. Karthikeyan

The development of proxy caching is essential in the area of video-on-demand (VoD) to meet users' expectations. VoD requires high bandwidth and creates high traffic due to the nature of media. Many researchers have developed proxy caching models to reduce bandwidth consumption and traffic. Proxy caching keeps part of a media object to meet the viewing expectations of users without delay and provides interactive playback. If the caching is done continuously, the entire cache space will be exhausted at one stage. Hence, the proxy server must apply cache replacement policies to replace existing objects and allocate the cache space for the incoming objects. Researchers have developed many cache replacement policies by considering several parameters, such as recency, access frequency, cost of retrieval, and size of the object. In this paper, the Weighted-Rank Cache replacement Policy (WRCP) is proposed. This policy uses such parameters as access frequency, aging, and mean access gap ratio and such functions as size and cost of retrieval. The WRCP applies our previously developed proxy caching model, Hot-Point Proxy, at four levels of replacement, depending on the cache requirement. Simulation results show that the WRCP outperforms our earlier model, the Dual Cache Replacement Policy.

Keywords: Cache replacement, hot-point, multimedia streaming, proxy caching, weighted rank.

Manuscript received Sept. 8, 2012; revised Jan. 2, 2013; accepted Jan. 6, 2013.

S.P. Ponnusamy (phone: +91 9940802417, spponns2k1@rediffmail.com) is with the Department Commuter Science, Bharathiar University, Tamilnadu, India, and also with the Department of Computer Applications, Adhiparasakthi Engineering College, Anna University, Tamilnadu, India.

E. Karthikeyan (e_karthi@yahoo.com) is with the Department of Computer Science, Government Arts College, Bharathiar University, Tamilnadu, India.

http://dx.doi.org/10.4218/etrij.13.0112.0606

I. Introduction

The growth of the Internet has created an environment in which such attractions as online games, online tutorials, webinars, and live sports streams increasingly draw users. However, live video streaming and video-on-demand are challenging to network service providers due to the necessity to provide high bitrate data transfers and long playback with desirable quality of service (QoS). This QoS and the long playback media streaming have increased Internet traffic, consumption of bandwidth, and the maximum latency between client and server. Thus, it is necessary to improve proxy caching to offer multimedia streaming without break of play.

In the past, the content providers placed the replicated media using content delivery networks (CDNs). The implementation of CDN is very expensive and difficult for synchronization of replication in all places. As a result, many researchers developed several proxy caching schemes to provide better multimedia content delivery. These schemes store the initial part or interleaved parts of video objects into a proxy server's cache. The caching is done at first access by a client from a media server.

It is not possible to cache an entire object in a proxy. For that reason, the media objects need to be divided into smaller chunks of independent playable content, and caching is processed on the media chunks. The caching is done on media chunks in two ways. The first way is to cache the media chunks in a predefined manner when no data regarding the popularity of the media is available. The Hot-Point Proxy (HPProxy) scheme [1] and DISC proxy caching scheme [2] use the first method. The second method of caching is working based on the popularity of the video, such as seamless playback caching [3], fragmented proxy caching [4], prefix-suffix caching [5], cooperative proxy caching [6], and segment-based caching [7], [8].

Another challenging task for researchers is to meet the user access pattern. From [9], it is understood that user behavior is aimless, that the user is generally not interested in watching video content in a linear fashion and continuous manner as it plays. The user's interest can change during the video playback. Therefore, the user can move from one location to another location in the video playback using the player slide bar. The study done by Lee and others [10] claims that without initially knowing the content of the video, many impatient users may just stop viewing the video early, not knowing that the portion they want to watch is contained in the latter part of the video. To support this user behavior, the researchers have developed such proxy caching models as HPProxy [1], fragmental proxy caching [4], and distributed cooperative caching [10].

If the proxy server has available cache space, it is allocated whenever new objects come into the proxy. At one stage, the entire proxy cache space is occupied by the video objects. Thus, the cache must be optimized or freed from existing objects to accommodate new incoming objects. The optimization can be done with cache replacement polices [4], [5], [11]-[26]. Most cache replacement policies select an entire cached object or part of the cached object as a victim and apply the policy to it. The victim objects are identified based on the cache requirement and the popularity of the existing objects. There are two types of cache replacement policies: single-factor replacement policy and multifactor replacement policy. Single-factor replacement policies include LFU, LRU, FIFO, and RAND [16]-[21]. Multifactor replacement policies comprise many factors, such as access frequency, recency, latency, and size, which are called weighted functions [3]-[5], [11]-[15], [22]-[26].

Our scheme is a multifactor cache replacement policy. It uses access frequency percentage, the age of the object, mean access gap ratio, and other functions, which are mentioned in section IV. We refer to our scheme as the Weighted-Rank Cache replacement Policy (WRCP).

II. Related Works

All cache replacement techniques are developed as part of proxy caching methods to allocate the cache space for new incoming objects when there is no cache space in the proxy server. The selection of a victim object is carried out by using any one of the replacement policies, as discussed in section I. In our previous work, we used the Dual-Cache-Replacement Policy (DCRP) [11] on the HPProxy model. The DCRP model applies the replacement policy on some of the existing objects at three levels based on hot-point levels, such as major, sublevel, and non-hot-point. The DCRP uses the object access frequency percentage and other costs to calculate the weighted cost (*WC*). Based on the *WC*, the three-level cache replacement is performed, and even the parts of the objects that have not been accessed recently or are very old are kept. The DCRP does not consider the age or access gap of the objects currently available in the proxy server.

The study done by Lee and others [10] illustrated the three levels of cache replacement as the key-frame level, prefix level, and segment level. The study used the weighted LRU-*k* scheme, which considers the different priorities assigned to each level. In [10], key frames and prefixes were deemed to be of high priority, due to poor consumption of cache resources. The segments were considered to be of low priority since the segments consumed more resources. In the weighted LRU-*k* caching replacement algorithm, objects of low priority are more likely to be removed, and objects of high priority are less likely to be replaced.

ElAarag and others [14] compared function-based proxy cache replacement schemes based on hit rate, byte hit rate, and removal rate. They suggested that the LRU and GDSF methods were not suitable for video streaming, whereas function-based methods such as M-Metric, MIX, and Greedy-Dual worked better on byte hit rate and hit rate metrics. Swain and others [22] used the adaptive weighted ranking algorithm (AWRA) for page replacement. In AWRA, the frequency index and recency index were used as parameters to determine the weight index. The page with the lowest weight index was selected as the victim and replaced.

A rank value (RV)-based replacement policy was developed by Gopalakrishnan Nair and others [23] for a multimedia server cache. In rank-based policy, the video objects are ranked based on the access trend by considering such factors as size, frequency, and cost. In [23], the video with the higher ranking was named "hot," while the video with the lower ranking was named "cold." The cold objects were replaced after determining the RV value of all objects. Podlipnig and Böszörmenyi [25] analyzed various web cache replacement strategies. They classified cache replacement into four major categories of strategy: recency-based, frequency-based, function-based, and randomized. The adaptive replacement approach combines the strategies based on recency, frequency, and function, thereby yielding superior multimedia caching.

Wang and others [26] proposed the Criteria Weighted (CW) algorithm for a peer-to-peer cooperative proxy caching system. The criteria for choosing the replacement victim includes access frequency of the document, size of the document, number of replicas of the document known to the proxy, and time lapsed since the last access to the document. In the CW algorithm, the victim objects are not evicted from the cache;

instead, the objects are pushed into a neighboring proxy, which does not have the object content.

Analysis shows that of the cache replacement algorithms studied in previous works, the multifactor cache replacement policies produce better results on byte hit ratio and hit rate on multimedia streaming. Using a multifactor cache replacement policy, we apply four parameters to find the weighted rank (*WR*) of the available object, and we evict either all cached parts of the object or some cached parts of the object, based on the cache space requirement. Most previous studies used access frequency, age, and other cost functions for the existing objects in the proxy cache. In addition to using these cost functions, we use a new factor called "mean access gap ratio." These four parameters and the calculation of *WR* are defined and explained in section IV.

III. System Model

The WRCP method is applied on our earlier developed proxy caching model HPProxy [1], which uses the DCRP [11] cache replacement policy for cache optimization. In HPProxy, the video files are divided into fixed length individual playable units, each referred to as a group of pictures (GOP). These GOPs consist of 16 frames and they consist of 15 I or P frames and 1 B frame inserted between the I or P frames. The frame rate is 32 fps, which is the equivalent of two GOPs per second. Six GOPs are grouped into a sector (same as segment). To meet the random user hit, the video files are divided into a fixed number of equal length regions. The regions are referred to as hot-point regions. On the first GOP of each region, a point is placed. The positioned points are called major hot points because all user hits are positioned on these points. Each hot-point region consists of a group of sectors (GOS). To provide a degree of additional fairness on random seek of the selected object, the hot-point region or GOS is divided into a number of sublevel hot points, depending on the size of the GOS. The sublevel hot points are also placed on the first GOP of the sector. The number of sublevel hot points and number of GOPs cached on each hot-point level in a GOS are determined based on the *k*-factor. It is calculated based on the condition $\alpha/2 < 2k \le \alpha$ (where α is the size of the hot-point region or GOS), as illustrated clearly in [1]. Figure 1 shows the position of hot points of a hot-point region with the size of 16 sectors and a calculated *k*-factor of 4.

Our system model consists of a proxy server simulator, a user request generator [4], and a cache replacement analyzer (CRA), as shown in Fig. 2. The proxy server simulator performs the streaming and determines the GOPs to be cached based on the HPProxy model. The user request generator generates the user request within the mean arrival time, *t*. The

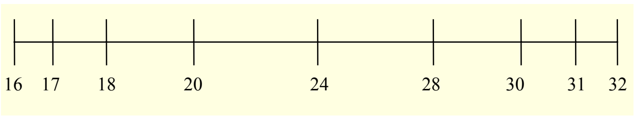

Fig. 1. Hot-point sectors selected based on *k* in GOS for caching.

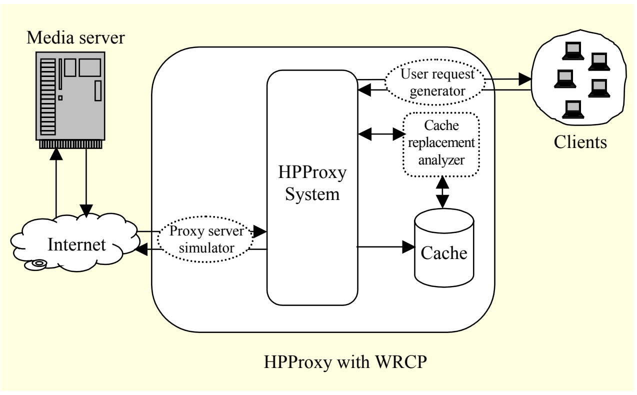

Fig. 2. System model of HPProxy with WRCP.

heart of the simulation model is the CRA. The CRA calculates the available free cache space in the proxy server and requires cache space to be available for the new arrival objects. The CRA also performs the decision-making system to determine the level of cache replacement needed to satisfy the cache space needed.

Consider that *q* number of objects ($O_1, O_2, \ldots, O_q$) enter the proxy in the *t*. The available proxy cache space is C_a, and the required cache space for *q* objects is C_r. Now, the CRA must allocate the space C_r by calculating the available cache space and needed cache space C_n. Hence,

$$C_n = C_r - C_a. \tag{1}$$

If the C_n is greater than 0, then CRA applies the WRCP cache replacement algorithm for the absolute value of cache space C_n; otherwise, the CRA allocates the C_r from the C_a.

IV. Weighted Rank Cache Replacement Policy

All cache replacement methods select a victim object and evict either the entire object or part of the object from the cache, depending on the need of the object for future use. The need of the object is determined based on the discussed parameters in section I. In this approach, an entire object includes all cached parts of an object and a partial object includes some cached parts of an object on the sector level. The selection of victim objects and victim sectors are chosen using the proposed WRCP. The WRCP method assigns the *WR* for objects available in the proxy cache, based on the calculated *WC*. In this paper, four parameters are considered to calculate the *WC* of the *j*-th object O_j, given as follows.

1. Access frequency (aF^j) of object O_j — number of times accessed compare with all the available objects.
2. Mean age (mA^j) of an object O_j — mean age of object O_j compared with all the available objects.
3. Mean access gap ratio (mR^j) of object O_j — it is calculated at every mean arrival time.
4. Other cost function (ϕ_j) such as fetching latency and size of object O_j.

To clearly determine the victim objects and apply the above-mentioned parameters, the objects are categorized into three kinds, based on the entry time in the proxy.

i) Elderly loaded object (O_E) — object that enters very early and has sufficient amount of access patterns and access frequencies.

ii) Recently loaded object (O_R) — object that was loaded recently, that is, few hours or few days earlier, and does not have sufficient amount of access patterns and access frequencies.

iii) Newly loaded object (O_N) — object that was just loaded, that is, few seconds or few minutes earlier, and does not have an access frequency.

The combination of parameters 1 and 2 produces a high value for the objects accessed most and the oldest objects (O_E). However, the same combination produces less value for recently loaded objects (O_R and O_N). Hence, if these two parameters are used, the cache replacement policy always removes the objects from O_R and O_N. To save these objects, a third parameter, mR^j, is introduced, apart from initial default aF^j percentage value assigned. The mR^j produces desirable results on frequently accessed objects at the current mean arrival time, t_c. Hence, the combination of all the parameters keeps objects from O_R and O_N while the cache replacement policy is applied. To ensure fairness in object replacement, a fourth parameter is introduced.

1. Access Frequency Percentage

The aF^j is used to determine the objects that are accessed most by users. Object O_E might have been accessed many times earlier but might not have been accessed recently. Hence, the aF^j value of object O_E ($[aF^j]O_E$) can be greater than or equal to the aF^j value of object O_R ($[aF^j]O_R$). Object O_E has a desirable value on parameters 1 and 2. The objects O_R and O_N each have a poor value on the first and second parameters. The aF^j percentage of object O_N, ($[aF^j]O_N$), is also unavailable in the initial stage. Hence, a fair mechanism is used to save the objects O_R and O_N from selection of victim objects using default aF^j percentage value, aF_0^j, for all new entry objects. The aF_0^j value is determined based on the default aF^j value assigned to all sectors based on the positions of the hot points of the concerned sector. For example, the major-level hot-point sectors are initialized with high value, sublevel hot-point sectors are initialized with less value, and non-hot-point sectors are initialized with zero. The aF_0^j value is set such that aF_0^j is greater than the mean access frequency, mF, of all objects.

Consider that $a_{t_c}^j$ is the total number of times object O_j is accessed by users at t_c. The mF value at t_c is calculated as

$$mF_{t_c} = \frac{\sum_{j=1}^{m} a_{t_c}^j}{m}, \tag{2}$$

where m is the number of objects available in the proxy at t_c. From (2), we can calculate the aF^j of object O_j at t_c as follows:

$$aF_{t_c}^j = \left(\frac{aF_{t_{c-1}}^j + a_{t_c}^j}{mF_{t_c}} \right). \tag{3}$$

The other cost function value of object O_j, ϕ_j, is assigned based on the fetching latency of a packet and the size of O_j. This parameter is considered a secondary parameter and assigned randomly to the objects using the Poisson distribution with a probability of 0.6. This ϕ_j is added with $aF_{t_c}^j$ while calculating the WC of O_j.

2. Mean Age and Mean Access Gap Ratio Estimation

It is easy to determine the age of object O_j by keeping the date and time it entered the proxy for the first time. Computers store the date and time in a combined value. Hence, aging, A^j, at t_c of O_j is calculated as

$$A_{t_c}^j = T_c - T_e^j, \tag{4}$$

where T_c is current time and T_e^j is the time object O_j entered the proxy. The mA^j of O_j is calculated from the elapsed time at t_c. The elapsed time E_{t_c} is determined by the time difference between T_c and the entry time of the oldest object available in the proxy, that is, O_1.

$$mA_{t_c}^j = \left(\frac{E_{t_c}}{A_{t_c}^j} \right). \tag{5}$$

The mean access gap (mG^j) of O_j is calculated by measuring the time gap between the current time of access and the last time of access. The existing mG^j at time t_{c-1} is $mG_{t_{c-1}}^j$, and the current access gap at time t_c is $g_{t_c}^j$. Consider that i is the last access number of O_j and T_i is the time of the most recent access. The $g_{t_c}^j$ is calculated as follows:

$$g_{t_c}^j = \begin{cases} T_i - T_{i-1} & \text{if } i\text{-th access exist in } t_c \\ T_d - T_i & \text{if no access exist in } t_c \end{cases} \tag{6}$$

where T_d is the time of proxy at the end of t_c. Hence, the current mean access gap value is

$$mG_{t_c}^{j} = \left(\frac{mG_{t_{c-1}}^{j} + g_{t_c}^{j}}{2} \right). \quad (7)$$

The mG^j is high if the object has rarely been accessed, and the mG^j is low if the object has been accessed frequently and recently. This determines how infrequently an object is accessed, even if the aF^j value of the object is desirable enough. Hence, the high mG^j value object must be replaced before the low mG^j value object. Therefore, the value of mG^j must be reversed to adopt in the calculation of the *WR* and defined as mR^j of O_j at t_c, $mR_{t_c}^{j}$, calculated as follows:

$$mR_{t_c}^{j} = \left(\frac{t_c}{mG_{t_c}^{j}} \right). \quad (8)$$

The combination of aF^j, mA^j, and mR^j determines the necessity of the object during the selection of victim objects.

3. Weighted Rank Calculation

The *WR* is used to determine the necessity of the object to the users. The objects with a high ranking are kept in the cache without replacement. Each object with a medium or low ranking is selected as a victim object and replaced with either an entire object or a partial object from the cache. As stated already, the *WR* of an object is assigned based on the calculated *WC* value of the object. The *WC* is calculated using (3), (5), and (8).

$$WC_{t_c}^{j} = aF_{t_c}^{j} + mA_{t_c}^{j} + mR_{t_c}^{j} + \phi_j. \quad (9)$$

After the *WC* is calculated, the objects are arranged in ascending order according to their *WC* values. Now, these sorted objects are categorized into five groups of objects using the *WC* values and these *WC* ranges are set as threshold ranges (*tr*). The first threshold range starts from 0 and ends with the first set point (th_0) and the final threshold range ends with the highest *WC*.

Figure 3 illustrates the five threshold ranges and their lengths. The values of all five threshold ranges are given in section V and explained. Each threshold ranged object is assigned a unique *WR* as follows:

$0 \le tr_0 < th_0$	$\rightarrow WR_0$,
$th_0 \le tr_1 < th_1$	$\rightarrow WR_1$,
$th_1 \le tr_2 < th_2$	$\rightarrow WR_2$,
$th_2 \le tr_3 < th_3$	$\rightarrow WR_3$,
$th_3 \le tr_4 <$ high *WC*	$\rightarrow WR_4$.

4. Victimization and Cache Replacement Policy

Cache replacement is done to optimize cache space when there is not enough memory available to store new cached GOPs. In this approach, a weighted rank algorithm is used to find an object or part of an object that should be replaced. The algorithm for WRCP is shown in Algorithm 1. The victimization of objects is applied at the following two levels.

i) Evicting entire object.

ii) Evicting partial object.

At the first level, the lowest *WC* object is selected as a victim object from the WR_0 group of objects and evicted. The number of objects chosen for eviction from the WR_0 group of objects depends on the cache requirement for new entry objects. The second level is applied when even more cache space is required after the first level eviction is applied. At the second level, objects are chosen from WR_1. When more cache space is needed for new objects, objects are chosen from WR_2 and then WR_3. However, no objects are chosen from WR_4 as victim objects.

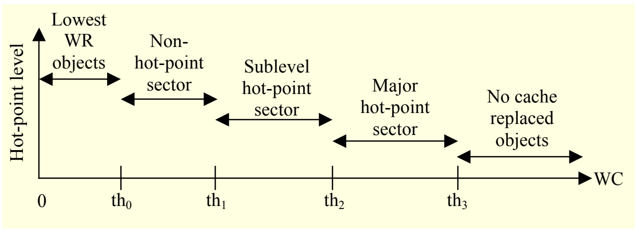

Fig. 3. Threshold levels for cache replacement.

Algoritm 1. Weighted-rank cache replacement policy algorithm.

```
1:  Assign $WR_i$ for all $O_j (0 \le i \le 4$ and $1 \le j \le n)$
2:  while new object group ($O_1, O_2, O_3, \ldots O_q$) comes in $t$
3:      calculate the required cache space ($C_r$)
4:      check the available cache ($C_a$)
5:      calculate $C_n = C_r - C_a$
6:        if $C_n \le 0$ then
7:            allocate the cache for $q$ new objects
8:               { no cache replacement required }
9:        else
10:           allocate $C_a$ cache space from proxy
11:            { cache replacement required }
12:           apply the WRCP for $C_n$ cache space
13:       $l$=0
14:     while ($l$<4)
15:           apply level l cache replacement from $WR_0$
                  objects
16:           calculate $C_r = C_r - C_a$
17:           if $C_r > C_a$ then
18:                $l$=$l$+ 1
19:             else
20:                exit
21:           end if
22:           end while
23:     end if
24:  end while
25:  update $WR_i$
```

To keep a fair amount of cache space in the WR_1, WR_2, and WR_3 groups for future use, partial replacement is applied using three more levels of cache replacement at the sector level after identifying the victim objects. The number of GOPs kept in a sector is greater in WR_3 than in WR_2 and WR_1. Three more sector level cache replacement procedures are as follows.

i) Evicting part of cached GOPs of non-hot-point sectors from WR_1 objects.

ii) Evicting part of cached GOPs of sublevel hot-point sectors from WR_2 objects along with level (i).

iii) Evicting part of cached GOPs of major level hot-point sectors from WR_3 objects, along with level (i) and (ii).

At each level, an equal number of cached GOPs from trailing places of a sector are selected as victim GOPs and evicted. In WRCP, the cache replacement starts from the full object eviction and moves to partial object eviction at three levels when more and more cache space is needed for new objects.

V. Simulation Setup

In our experiment, event-driven simulations are used to enhance the earlier developed HPProxy with the DCRP cache replacement model. In the DCRP model, the cache replacement is done only at three levels with partial replacement of objects. In the WRCP approach, four levels of victimization are used, as discussed in section IV. The ranges of threshold value for cache replacement at four levels are shown in Table 1. In our simulation model, the threshold ranges are set between two values from a low range group and a high range group for a level. The low range value is set as a lower bound while the high range value is set as an upper bound. This boundary setup is used to find the number of object available for selection of victim objects.

For example, at the first step of Level 1, the starting lower bound is set as 2.6 from the threshold low range, and the starting upper bound is set as 2.9 from the threshold high range, and this is referred to as Threshold Point 1. Now, all the objects that have a *WC* value range from 2.6 to 2.9 are set as WR_1 and chosen as victim objects for Level 1 cache replacement. In the same manner, all ranges are set and applied to the simulation model to find the required cache space. At each level, ten threshold points are used. Depending on the cache space requirements for new incoming objects, the threshold points can be set from 1 to 10 at Level 0. If the threshold point crosses 10 at Level 0, the cache replacement algorithm is applied at Level 1 when more cache space is required. This procedure continues until the last level.

Table 2 shows the default parameters of our simulation model. Figure 4 shows the number of objects selected as victim objects for various threshold points in 100 cycles sampled.

The default parameters are used throughout the performance study unless otherwise specified in any place. The prefix is fixed as 30 GOPs, which is playable for 15 seconds. This prefix is not used for cache replacement in any level except Level 0. The basic caching model suggested in [2] keeps only 291 objects at the maximum in the proxy server. But the existing HPProxy model can keep 1,029 objects at the maximum at a time under the default values. The object arrival rate and user request rate are generated using Poisson distribution and the mean arrival time is fixed as 100 seconds. For every mean arrival time, the *WC* of all objects is calculated, and objects are assigned with a *WR* as discussed in section IV. The objects are numbered in the order of entry into the proxy server. The object numbers are reassigned in the same order of entry after every cache replacement is done. Hence, the first object is always the

Table 1. Threshold ranges for four-level cache replacement.

Level	Number of threshold points	Threshold range					
		Threshold low			Threshold high		
		Start	Incr	End	Start	Incr	End
0	10	0	--	--	2.5	0.1	3.4
1	10	2.6	0.1	3.5	2.9	0.3	5.6
2	10	2.9	0.3	5.6	3.3	0.4	6.9
3	10	3.3	0.4	6.9	3.8	0.5	8.3

Table 2. Simulation parameters.

Description	Value
Cache size	2,100,000 GOPs
Mean object size	7,200 GOPs
GOP size	16 frames
Prefix size	30 GOPs
Sector size	6 GOPs
Frame rate	32 fps
Layering	Single
Bitrate	Constant
Hot point size	40
Mean arrival time	100 s
Number of threshold points	10
Default access percentage	
Hot-point sectors	0.05
Sublevel hot-point sectors	0.03
Non-hot-point sectors	0.01

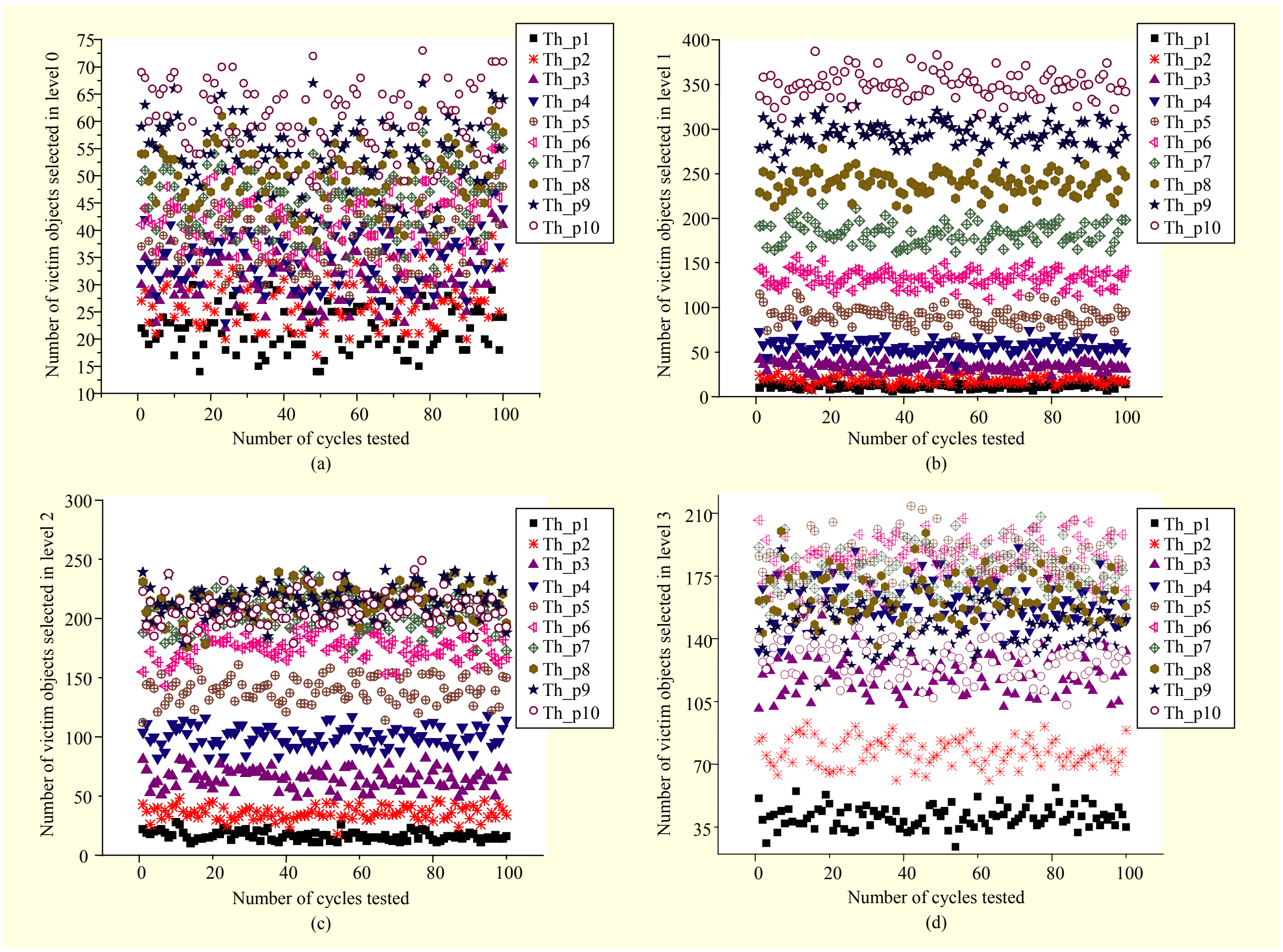

Fig. 4. Number of victim objects selected on different threshold points in Levels (a) 0, (b) 1, (c) 2, and (d) 3.

oldest object of the available objects.

VI. Result Analysis

In this section, the result of the simulation model is presented and analyzed with the earlier DCRP model on the HPProxy system. To understand the fairness of the threshold point selection, Fig. 5 shows the mean, median, mean deviation, and standard deviation of all levels of selection of victim objects for the different threshold points that are closer at all levels of cache replacement. The number of objects selected as victims for every threshold point does not deviate much, and the WRCP model selects the constant number of objects as victim objects at all levels of cache replacement under all threshold points. Hence, the number of objects selected as victims at all levels is almost always a constant value. The mean and median values decrease at higher threshold points in Level 2 and Level 3 objects due to a smaller number of new objects entering the proxy. The *WC* of these newer objects is relatively high. The *WC* of many objects available for selection as victim objects is essentially equal to the mean *WC* value. So, the peaks exist at Threshold Point 8 in Level 2 and at Threshold Point 6 in Level 3. However, there is an increasing number of objects available for selection as victims at all threshold points in Level 0 and Level 1. The reason is that Threshold Point 8 and Threshold Point 6 come under the same threshold range in Levels 2 and 3, respectively.

Figures 6 and 7 illustrate the additional number of GOPs and additional number of objects accommodated, respectively, after applying the WRCP model. As already stated, the number of objects available with WR_3 is low compared with other levels, and the proxy server can accommodate few extra objects at Level 3. The remaining level cache replacements produce extensive space to accommodate more objects.

In the earlier DCRP model, removing entire objects was not done. In the WRCP model, the technique of replacing the entire object is incorporated in Level 0 cache replacement. Level 0 improves the cache space optimization from 3.01% to 6.41% for low and high threshold values, respectively. This percentage of improvement allows for 31 more objects at minimum and

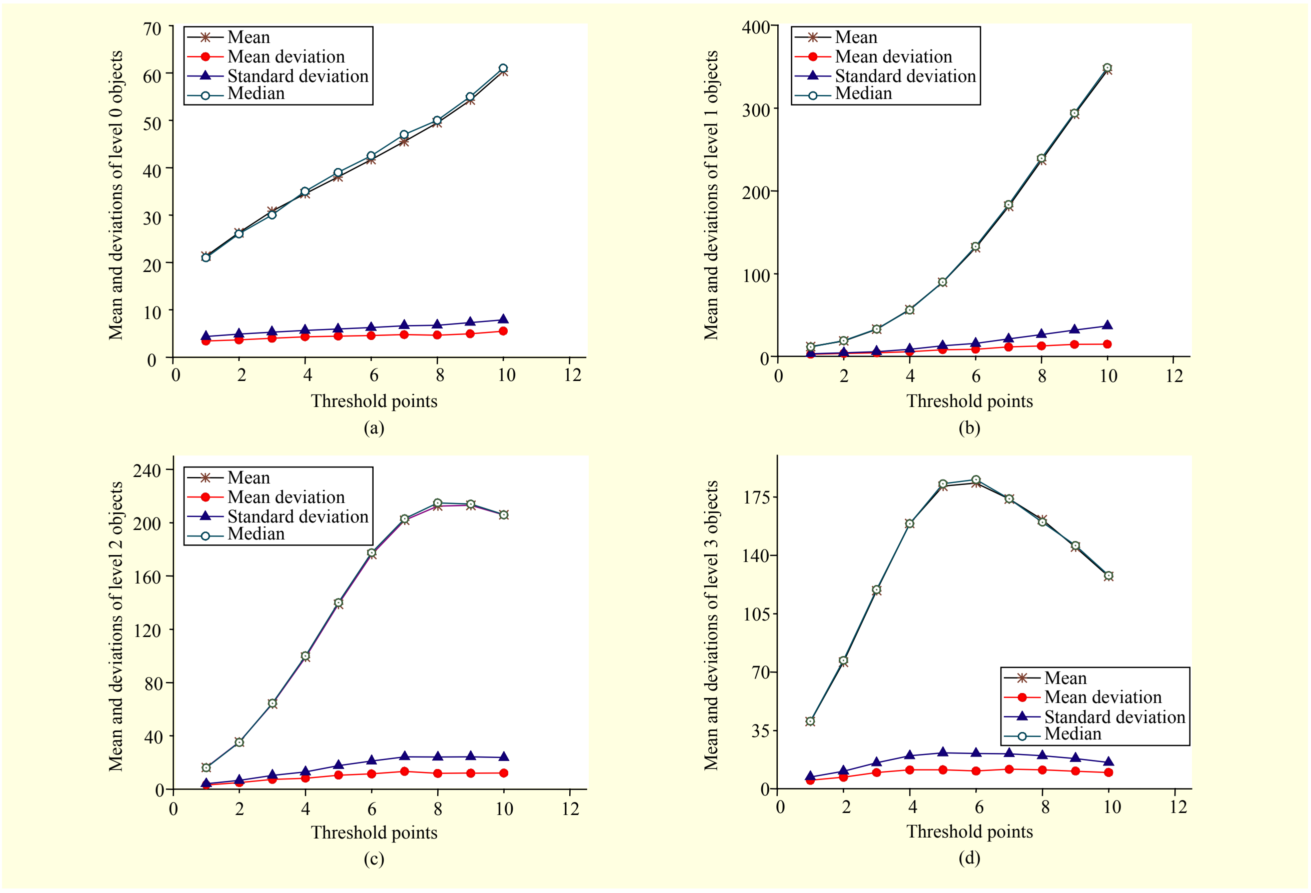

Fig. 5. Mean, median, mean deviation, and standard deviation of (a) Level 0, (b) Level 1, (c) Level 2, and (d) Level 3 objects as victims for various threshold points.

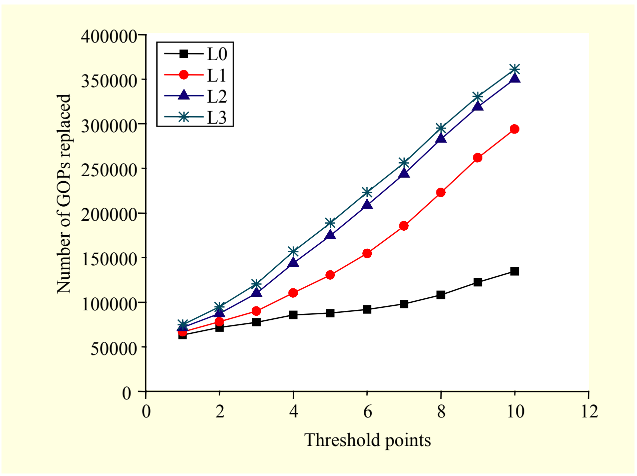

Fig. 6. Number of GOPs freed from proxy server using WRCP.

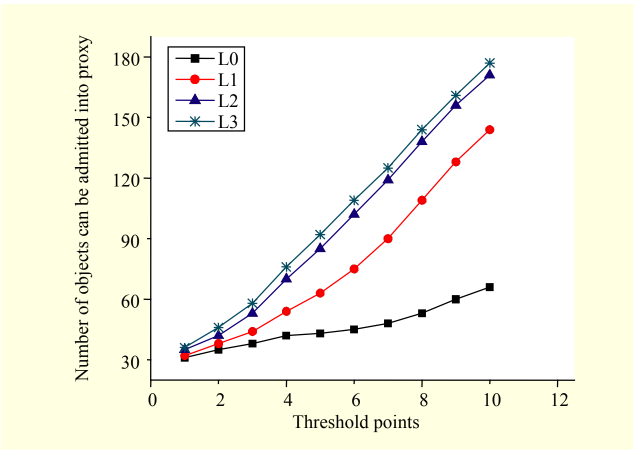

Fig. 7. Number of objects admitted into proxy server using WRCP.

66 more objects at maximum in the proxy, in addition to the existing 1,029 objects with the cache size of the proxy and the mean object size as shown in Table 2. In the same way, Level 1 can accommodate 3.11% to 13.99% more objects in the proxy.

The simulation result at Level 2 cache replacement produces3.40% at minimum and 16.62% at maximum cache space for new objects, which is equivalent to 35 and 171 more objects, respectively. In the same manner, Level 3 cache replacement accomodates a minimum of 36 and a maximum of 177 more objects in the same proxy server with the same cache size.

Figures 8(a) and 8(b) show the minimum and maximum

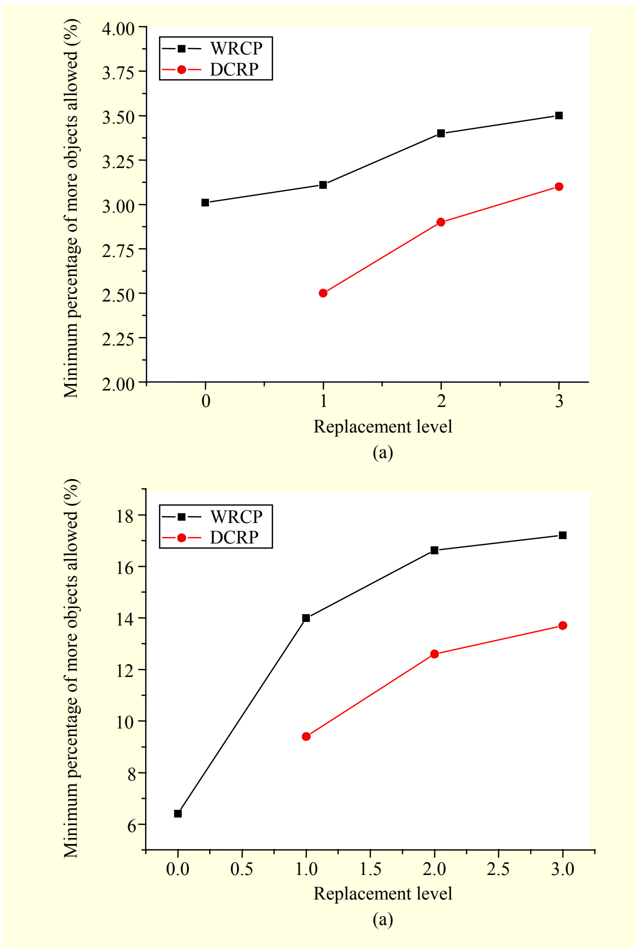

Fig. 8. Comparison of WRCP and DCRP models (a) minimum percentage of additional objects allowed and (b) maximum percentage of additional objects allowed in proxy server.

percentage of additional objects that can be allowed in the proxy server by means of accommodating more objects compared with the earlier DCRP and WRCP models. The number of objects that can be stored in the proxy server by the WRCP model increases by 2% on average, which allows 20 more objects, and by 3.5% at maximum, which allows 30 more objects, compared with DCRP. In the WRCP model, WR_4 objects are not used for victimization because these objects have been frequently accessed and recently added. The level of cache replacement and setting threshold point on a level can be selected depending on additional cache space required in the proxy server. As a result, the new WRCP model outperforms the existing DCRP model on the HPProxy caching system.

VII. Conclusion

In this paper, a new WRCP cache replacement policy was proposed to improve the performance of the existing HPProxy model that uses DCRP. In DCRP, the least weighted object is selected as the victim, and certain parameters are used to calculate the *WC*. However, in WRCP, a *WR* is assigned to each available object based on the *WC*, primarily using the following parameters: how frequently an object is accessed, its age, and its access gap ratio. The cache replacement is done at four levels, 0 to 3. At Level 0, the entire object is replaced since these objects rank very poorly. Cache replacement is moved from one level to another, depending on the cache space required for new objects. From Level 1 to Level 3, the minimum GOPs are kept in a cached sector to fulfill future requests. From the analysis, the cache replacement methodology optimizes the cache space by 3.01% in the worst case and 17.2% in the best case, which means that 31 more objects can be stored in the cache at minimum and 177 more objects can be kept at maximum.

Compared with the earlier cache replacement model, DCRP, the new WRCP can accommodate 3.5% more objects at best and 2% more objects on average. This WRCP is well suited for popularity-based and non-popularity-based cache replacement. The WRCP performs well on the HPProxy and provides cache space for new objects without losing the byte hit ratio on frequently accessed objects and new objects. The WRCP also outperforms the earlier proposed cache replacement model, DCRP.

References

[1] S.P. Ponnusamy and E. Kathikeyan, "HPProxy: Hot-Point Proxy Caching with Multivariate Sectoring for Multimedia Streaming," *European J. Sci. Research*, vol. 68, no. 1, Jan. 2012, pp. 21-35.

[2] L. Guo et al., "DISC: Dynamic Interleaved Segment Caching for Interactive Streaming," *Proc. ICDCS*, June 2005, pp. 763-772.

[3] B. Gao et al., "Beyond the Playlist: Seamless Playback of Structured Video Clips," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, Aug. 2010, pp. 1495-1501.

[4] J.Z. Wang and P.S. Yu, "Fragmental Proxy Caching for Streaming Multimedia Objects" *IEEE Trans. Multimedia*, vol. 9, no. 1, Jan. 2007, pp. 147-156.

[5] W. Tu et al., "Proxy Caching for Video-on-Demand Using Flexible Starting Point Selection," *IEEE Trans. Multimedia*, vol. 11, no. 4, June 2009, pp. 716-729.

[6] A.T.S. Ip, J. Liu, and J.C.-S. Lui, "COPACC: An Architecture of Cooperative Proxy-Client Caching System for On-Demand Media Streaming," *IEEE Trans. Parallel Distr. Syst.*, vol. 18, no. 1, Jan. 2007, pp. 70-83.

[7] S. Chen et al., "Segment-based Streaming Media Proxy: Modeling and Optimization," *IEEE Trans. Multimedia*, vol. 8, no. 2, Apr. 2006, pp. 243-256.

[8] Y. Li and K. Ong, "Optimized Scalable Cache Management for Video Streaming System," *Multimedia Tools Appl.*, vol. 44, no. 1, Apr. 2009, pp. 65-86.

[9] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," *Proc. P2PMMS*, 2005, pp. 29-38.

[10] S.-J. Lee, W.-Y. Ma, and B. Shen, "An Interactive Video Delivery and Caching System Using Video Summarization," *Comput. Commun.*, vol. 25, issue 4, Mar. 2002, pp. 424-435.

[11] S.P. Ponnusamy and E. Kathikeyan, "Cache Optimization on Hot-Point Proxy (HPProxy) Using Dual Cache Replacement Policy," *Proc. ICCSP*, Apr. 2012, pp. 108-113.

[12] C.-F. Kao and C.-N. Lee, "Aggregate Profit-Based Caching Replacement Algorithms for Streaming Media Transcoding Proxy Systems," *IEEE Trans. Multimedia*, vol. 9, no. 2, Feb. 2007, pp. 221-230.

[13] F.J. Gonzalez-Canete, E. Casilari, and A. Trivino-Cabrera, "Characterizing Document Types to Evaluate Web Cache Replacement Policies," *Proc. 4th European Conf. Universal Multiservice Netw.*, Feb. 2007, pp. 3-11.

[14] H. ElAarag and S. Romano, "Comparison of Function Based Web Proxy Cache Replacement Strategies," *Proc. Int. Symp. Performance Evaluation Comput. Telecommun. Syst.*, vol. 41, Aug. 2009, pp. 252-259.

[15] H.-P. Hung and M.-S. Chen, "On Designing a Shortest-Path-Based Cache Replacement in a Transcoding Proxy," *Multimedia Syst.*, vol. 15, no. 2, Apr. 2009, pp. 46-62.

[16] A. Satsiou and M. Paterakis, "Frequency-Based Cache Management Policies for Collaborative and Non-collaborative Topologies of Segment Based Video Caching Proxies," *Multimedia Syst.*, vol. 12, no. 2, Sept. 2006, pp. 117-133.

[17] K. Li, K. Tajima, and H. Shen, "Cache Replacement for Transcoding Proxy Caching," *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Sept. 2005, pp. 500-507.

[18] K. Kim and D. Park, "Least Popularity-Per-Byte Replacement Algorithm for a Proxy Cache," *Proc. 8th Int. Conf. Parallel Distrib. Syst.*, June 2001, pp. 780-787.

[19] J. Dilley and M. Arlitt, "Improving Proxy Cache Performance: Analysis of Three Replacement Policies," *IEEE Trans. Internet Comput.*, vol. 3, issue 6, Dec. 1999, pp. 44-50.

[20] X. Li, W. Tu, and E. Steinbach, "Dynamic Segment Based Proxy Caching for Video on Demand," *Proc. IEEE Int. Conf. Multimedia Expo*, Apr. 2008, pp. 1181-1184.

[21] A. Wierzbicki et al., "Cache Replacement Policies Revisited: The Case of P2P Traffic," *Proc. IEEE Int. Symp. Cluster Comput. Grid*, Apr. 2004, pp. 182-189.

[22] D. Swain, B. Paikaray, and D. Swain, "AWRP: Adaptive Weight Ranking Policy for Improving Cache Performance," *J. Comput.*, vol. 3, issue 2, Feb. 2011, pp. 209-214.

[23] T.R. Gopalakrishnan Nair and P. Jayarekha, "A Rank Based Replacement Policy for Multimedia Server Cache Using Zipf-Like Law," *J. Comput.*, vol. 2, issue 3, Mar. 2010, pp. 14-22.

[24] K. Samiee, "A Replacement Algorithm Based on Weighting and Ranking Cache Objects," *Int. J. Hybrid Inf. Technol.*, vol. 2, no. 2, Apr. 2009, pp. 93-104.

[25] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *J. ACM Comput. Surveys*, vol. 35, issue 4, Dec. 2003, pp. 374-398.

[26] J.Z. Wang, A. Pal, and P.K. Srimani, "New Efficient Replacement Strategies for P2P Cooperative Proxy Cache Systems," *Proc. DASD*, Apr. 2004, pp. 97-106.

S.P. Ponnusamy received his MCA in computer applications from Bharathidhasan University, Trichy, India, in 1996 and his MPhil from Alagappa University, Karaikudi, India, in 2007. He is currently earning his PhD in the Department Commuter Science, Bharathiar University, Tamilnadu, India. He is currently working as an associate professor in the Department of Computer Applications at Adhiparasakthi Engineering College, Anna University, Chennai, India. His research interests include video streaming, network security, mobile computing, and peer-to-peer networks. He is also a Cisco Curriculum lead and instructor of CCNA Academy. He is a lifetime member of ISTE and the Computer Society of India.

E. Karthikeyan received his PG in 1996 and his PhD from Gandhigram University, Dindigul, India, in 2008. His area of research is network security and cryptography and advanced networking. He has authored 17 papers published in international journals and provided material for more than 15 conferences at the national level and international level. He has also authored a book entitled "Text Book on C: Fundamentals, Datastructures and Programming," published by PHI. He has delivered lectures at various conferences and conducted workshops at various colleges. He is a lifetime member of CSI, CRSI, and IASCT. He is an editor-in-chief for the *International Journal of Advanced Networking and Applications*.