iHPProxy: Improving the Performance of HPProxy by Adding Extra Hot-Points

Ponnusamy Pichamuthu[1] and Karthikeyan Eswaramurthy[2]
[1]Department of Computer Science, Bharathiar University, India
[2]Department of Computer Science, Government Arts College, Udumalpet, Bharathiar University, India

Abstract: *In recent years, the interest of Internet users turned into viewing videos such as Video-on-Demand (VoD), online movies, online sports, news, e-learning, etc., the researchers involved proxy caching with replacement to provide immediate content delivery to the client on request. One important aspect in the content delivery is continuous playability with random seek even the client wants to watch the video from a new location by jumping into that location. The continuous play is possible if the hit location is cached already, otherwise a delay will occur. The researchers allowed a small deviation from the desired location in the backward direction to reduce the delay. Our earlier model, Hot-Point Proxy caching (HPPrxoy), also supports the shift to the nearest cached Group Of Pictures (GOP) in the backward direction to play immediately, but, in some cases the deviation was large. Hence, we proposed a new model to provide a little deviation by adding extra hot-points between existing sub-level hot-points. However, this mechanism additionally consumes cache memory, it increases the byte-hit ratio, satisfies the user requirement in random seek and provide better cache replacement.*

Keywords: *Proxy caching, shift distance, HPProxy, cache replacement, multimedia streaming, VoD.*

Received May 9, 2013; accepted July 8, 2013; Published online December 23, 2015

1. Introduction

Providing multimedia content delivery like Video-on-Demand (VoD), online news, sports and online webinars are very challenging due to the high bit rate and longer playable. So, it is necessary to provide more attention in the field of proxy caching to offer multimedia streaming without break of play with good Quality-of-Service (QoS) [19]. To improve the performance on multimedia content delivery, the content providers placed the replicated video using Content Delivery Networks (CDN) which is placed closer to the client group. The implementation of CDN is very expensive and difficult to replicate in all the places. So, many researchers involved to provide better multimedia content delivery and developed proxy caching scheme. The proxy caching scheme stores the part of the video object into a proxy server.

The caching methodology offered for static objects, used mostly in HTML pages, could not be offered to multimedia streaming due to the need of lower latencies, the size of the object and continues playability. For that reason, the video objects need to be divided into smaller chunks of independent playable content. The researchers provide many caching schemes such as Hot-Point Proxy (HP Proxy) caching [15], fragmented proxy caching [21], cooperative proxy caching [5, 8], segment based caching [4], seamless playback caching [6], prefix-suffix caching [18] and so on. The cache replacement polices also involved to accommodate additional caching of popular objects or new objects. Even though the proxy caching improves the video streaming services by reducing network traffic and bandwidth conservation, the continuous playing of video is still lacking. The continuous playability was achieved by many prefetching schemes [1, 10, 11, 12] which prefetch the successive chunks while playing the earlier chunks. The prefetching could be made during the playing of a video or earlier.

The important aspect from the user point of view is supporting the random seek. Whenever the user hit placed on any chunks, immediately the content should be played to achieve the random seek support. The hit might be placed on the cached part or non cached part. If the hit is placed on cached part, the content might be delivered immediately and played. If it is not, the content must be fetched from video server and delivered to client by proxy. It leads a long delay to play the requested part of the video. So, the researchers were using the popularity [9, 19, 18] of the videos to cache the content in the proxy. This could be very easy when the video have sufficient access logs and popularity.

There are two kinds of popularity called object popularity and internal popularity used to determine which part of the chunk cached in the proxy. The object popularity was calculated on the overall popularity of object based on the user accessibility. It was calculated that how many times the object was accessed by the user over a bounded period of time. But, the internal popularity calculated as how many times the individual chunk has been accessed.

The popularity might not be available for newly loaded objects. HPProxy model [15], fragmental proxy caching [10] supports for this kind of video. In addition to, the continuous playability, the important characteristic of the VoD is supporting the random seek. Many researchers [3, 15, 18, 20, 21, 22], involved in random seek support like Video-Cassette-Recorder (VCR) functions.

Another challenging area in the video streaming is preserving cache space on the proxy server to accommodate new objects. Researchers developed many cache replacement policies [7, 13, 14, 17], to replace the old or non-popular objects from the server. The researchers suggested that multi-factor replacement policies produce better replacement of objects than the single factor replacement policies. multi-factor replacement policies comprise many factors such as access frequency, recency, latency, size, etc., whereas single factor replacement policies comprise any one of the factors. From the factors discussed above, our model supports: Proxy caching of video objects mainly for non-popular objects, random seek with forward and backward jump, and weighted-cost cache replacement policy.

2. Related Works

Ponnusamy and Kathikeyan [15] developed a HP Proxy caching model to support playback with random seek. The authors primarily considered a caching mechanism for non popularity objects and for preserving more cache space to keep more objects. The HPProxy model divides the object into smaller size independently playable objects called as Group of Pictures (GOPs). The model caches the GOPs based on the number of hot-points selected. The major hot-point sector cached more GOPs and the sub-level hot-points cached less number of GOPs. The number of GOPs cached was reduced when the level of hot-points moved into inner level. Wang and Yu [21] proposed the fragmental proxy caching scheme divide the object into two types of interleaved fragments; three consecutive streamed fragments (Sfrag) which are served from video server through proxy and two consecutive cached fragments (Cfrag) which are served directly from proxy. This model caches 40% of object in the proxy server.

Yu *et al*. [18] proposed variable size segment caching (dynamic segment caching) scheme, in which the prefix of segment structure is determined based on the popularity of the segment. The high popular segment gets a high amount of prefix whereas low popular segment gets the less amount of prefix to be cached. Ip *et al*. [8] developed cooperative proxy caching. In this model, the prefix of the video is cached in the proxy server and prefix-of-suffix of the video is cached in the client's side. Most studies of prefix methodologies provide the concept that the prefix caching reduces the initial delay of video playback.

The cache replacement policy developed by Ponnusamy and Kathikeyan [13, 14] works based on the weighted-cost replacement policy on various levels of objects. It dealt with major hot-point, sub-level hot-point and non-hot-point sectors as different level and replaced the cached GOPs. A Rank Value (RV) replacement policy was developed by Gopalakrishnan and Jayarekha [7] for multimedia objects. The video objects were ranked based on the access trend by considering the factors such as size, frequency and cost. Anusuya *et al.* [2] mainly considered the play back time and play out buffer size of the video player used at the receiver to provide better QoS.

The challenging task for the researchers is meeting the user requirement during the video watching. Since, the user may move the current play location to anywhere in the object and expect immediate play of the video. As per the study done by Brampton *et al.* [3] the user always not view the object as-in-as play and jump into the new location where they need the appropriate content. The study shows that small forward seeks were used a combined 24.9% of the time, whereas backward seeking was only used 7.67%. These actions only accounted for the relatively small seeks (10, 30 and 60s), whereas potentially large seeks (seek-bar and following bookmarks) made up 34.5% of all operations. Other studies related to random seek to support VCR like functions done by Shen *et al*. [16, 18, 21].

3. The System Model

Usually, the proxy server cache the object which has already been streamed to one of its clients and served later if another client is requesting the same object. The proxy server preserves memory for the caching. The object was divided into many numbers of GOPs which were individually playable and 6 numbers of GOPs form a sector. The entire object was divided by '*n*' number of hot-points which produce equal size region consisting of '*m*' sectors. These hot-points are called as major hot-points. The equal size regions were called as hot-point regions or Group Of Sectors (GOSs) and denoted as α. Each hot-point region was further divided using binary division on both sides of the middle part of the region until the division value reaches 1. For each divided place, a new sub-level hot-point was placed. All major and sub-level hot-points were fixed at the start of the sector. The number of GOPs to be cached in a sector was determined based on the level of hot-points. More GOPs cached at major hot-points and less GOPs cached at inner levels.

The process of placing various levels of hot-points within a hot-point region is shown in Figure 1 and the number indicates the order of the division. The number of hot-points placed in a region is called as *k*-factor means there are *k* number of sub-hot-points placed on each side of the middle hot point. Hence, there are 2*k*-1 hot-points in a region. The major hot points used to cache more GOPs and the number of GOPs cached in sub-level hot-points is reduced by one number when moving into inner sub-level hot-points. For example, consider *α*=16 and its *k* is 4. Figure 2 shows the number of GOPs could be cached based on HPProxy model. All non-hot-point sectors caches one GOP each.

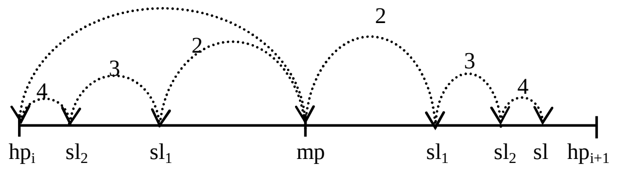


Figure 1. Placement of various hot-point levels in a hot-point region.

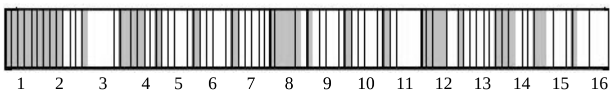


Figure 2. Number of GOPs cached at various hot-point levels.

The HPProxy caches only 34% of GOPs in the proxy cache for good *α* and *k*-factor. HPProxy cached lesser than the above percentage when the *k*-factor is less for *α* closer to 2^k. The cache space requirement for an object is required at a maximum of 34% and a minimum of 30%. The fragmental caching cached a constant cache space 40% in the proxy server for an object. The dynamic segment caching model cached at variable size chunks depends on the popularity of chunks. For comparison, 50 % as the proxy caching has been averaged by the dynamic segment model.

4. Reducing Shift Distance

As earlier said, the main challenging task of the researcher is to provide continuous play without any delay whenever a user jumps into a new location. This jumping into a new location is called random seek which falls in forward or backward direction from current location and the distance is called as seek distance. The seeking pattern may be as follows:

1. Seeking in Linear Fashion: Moving in forwarding direction only and play for a while then continue the same seeking and play till reaches end of the object (seek distance may be very short).
2. Seeking in Non-Linear Fashion: Moving in forward and backward directions in a random manner to find the desired location and play (seek distance may be longer for the first seek and short for successive seeks).

The play can start from the new location when the proxy has the cached part of the video; otherwise it has to be fetched from the video server. It makes a long delay and user not accepting this delay. Hence, the researchers [15, 16, 18] shift the hit location in backward direction little bit to start the play from the cached part from the proxy and this shifting is not sensed by the user. If the shift distance is high, it might be sensed by the user. It is understood the importance of the shift distance and applied the reduction of shift distance in the earlier HPProxy model. The following section explains the placing extra hot-points between existing sub-level hot-points within every hot-point region.

4.1. Shift Distance

As discussed in Ponnusamy and Kathikeyan [15] the major and sub-level hot-points are used as hit-points. In some cases, the user hit may fall on non hot-point sector which cached the minimal number of GOP. In this situation, user viewpoint is shifted to the nearest hot-point sector GOP in backward direction and starts play instead the actual hit sector's GOP. This shifting of actual hit location to shifted hot-point location is called as shift distance and denoted as *d*. It is calculated as:

$$d= \varepsilon_{al}- \varepsilon_{hp} \tag{1}$$

Where ε_{al}: Is actual hit GOP's location value, and ε_{hp}: Is the shifted hot-point starting GOP's location value.

This *d* value might be high for large file size when the minimal number of hot-point value is used to divide the object. The larger *d* value creates dissatisfaction for the user. To rectify this, we have added an extra hot-point between two sub level hot-points with the same number of cached GOPs as in sublevel hot-points. This creates the shorter shift and the shift is not realized by the user. Table 1 show the notations used in this paper.

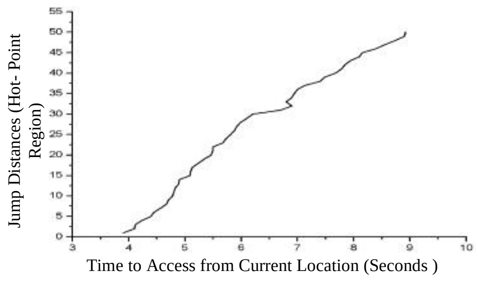
Table 1. Notations.

Symbol	Notations
O	Video Object
$\mathbf{W_i}$	Size of Object i
$\mathbf{s_i}$	ith Sector
h	Hot-point
α	Size of Group-of-Sectors
$\mathbf{O_n}$	Number of video objects
$\mathbf{R_i}$	ith hot-point region
k	Number of hot point level in a GOS
$\mathbf{h_l}$	Hot point level within a region, where $1 \le l \le k$
d	Shift distance
$\boldsymbol{\varepsilon_i}$	ith GOP in a R
$\mathbf{t_m}$	Movement time from one location to another location
$\mathbf{t_s}$	Shift time

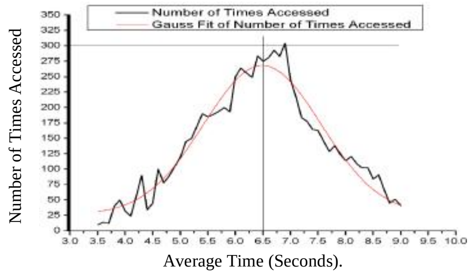
4.2. Placing Extra Hot-Point

Placing an extra hot-point is determined by the average amount of time taken to move the viewpoint from current location to a new location and denoted as t_m. To determine t_m, we have collected real test values from 100 users with more than 2000 hits into different locations.

Figure 3-a shows the movement time required for various seek distances by means of number of sectors in a hot-point region and Figure 3-b shows the average amount of time taken by the user to jump into a new location. From the Figure 3-b, we understand that more users taken 3.5 seconds to move into new location by dragging and 1.5 seconds to direct hit into a new location. So, the average time requirement for a movement is 4 seconds. To match the t_m with the shift time t_s, the following conditions are set.



a) Movement time for different jump distances.



b) Average time for most accessed time movement.

Figure 3. Time required moving from one location to another location and average movement time.

$$\begin{cases} t_m \approx t_s \\ t_s = d * 0.5 \leq 4 \ \ \textit{seconds} \end{cases} \tag{2}$$

The extra hot-point is placed when the worst case condition $d>20$ is met, that is, there are at least 3 sectors existing between two sublevel hot-points. If an extra hot-point is placed in the middle between existing two sublevel hot-points, the time difference between sublevel hot-point and extra hot-point is reduced to 4 seconds from 8 seconds. So, the user movement time is matched with our extra hot-point placement. The algorithm for caching GOPs based on HPProxy and placing extra hot-point is shown in Algorithm 1.

Algorithm 1: Caching and placing extra hot-point (h_e).

```
foreach($O_i$ is loaded in server)
  {
    Divide the $O_i$ by n hot-points region
    Find sublevel hot-points in $R_i$
    Find the k-factor for $R_i$ {$\alpha/2 < 2^k < \alpha$}
    Find d between $h_l$ and $h_{l+1}$
    do
      {
        if($d$>20)
          {
            place $h_e$ in middle
          }
        Find d between $h_l$ and $h_{l+1}$
      } while ($d$>20)
    Cache GOPs in all $h_l$ based on l
  }
```

4.3. Cache Replacement Policy

In this paper, the earlier cache replacement model WRCP [14] is used to replace the cache space from the proxy sector. The selection of victim objects and victim sectors are chosen based on Weighted-Rank (WR) of objects. It assigns WR value for objects available in proxy cache based on the calculated Weighted Cost (WC). The four parameters, access-Frequency (aF^j), mean-Aging (mA^j), mean-access-gap-Ratio (mR^j) and other cost functions (ϕ_j) of an object O_j are considered to calculate *WC*. The *WC* was calculated at the current mean arrival time, t_c, as:

$$WC^j_{tc} = aF^j_{tc} + mA^j_{tc} + mR^j_{tc} + ¤_j \tag{3}$$

After *WC* is calculated, the objects are arranged in an ascending order using *WC* and grouped into five categories named as WR_0, WR_1, WR_2, WR_3 and WR_4 objects. The entire object replacement is done on WR_0 objects. The partial object replacement is done on WR_1, WR_2 and WR_3 objects with various threshold ranges. High number of GOPs is replaced in WR_1 and lower number of GOPs is replaced in WR_3. No cache replacement is carried out with WR_4 objects. At each level, equal numbers of cached GOPs from trailing places of a sector are selected as victim GOPs and evicted. The cache replacement was started from WR_0 and moved towards WR_3 when more and more cache space is needed.

5. Analysis

In this section, the performance of HPProxy Caching is improved based on the evaluation metrics show in Table 2. The improved HPProxy (iHPProxy) mainly focus on meeting the user random seek with minimal shift distance, better byte-hit ration and good cache replacement policy. The average cache space requirement comparison done based on the optimum hot point group and various object sizes. The synthetic data used in the HPProxy caching [15], fragmental [21] proxy caching and dynamic caching [18] studies are used in iHPProxy to reproduce its simulation results in order to have fair comparisons.

Table 2. Default simulation parameters.

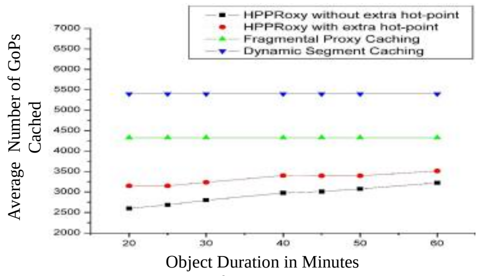
Description	Value
Minimum Object Size	30 minutes
Maximum Object Size	150 minutes
Gop Size	16 frames
Prefix Size	30 GOPs
Sector Size	6 GOPs
Frame Rate	32 frames/sec
Optimum Hot-Point Group	{20,25,30,40,45,50,60}
Default Optimum Hot Point	40
Mean Object Size	7 200 GOPs
Layering	Single
Mean Arrival Time	100s

Number of Threshold Points	10
Default Access Percentage	
Hot Point Sectors	0.05
Sub-Level Hot-Point Sectors	0.03
Non Hot Point Sectors	0.01

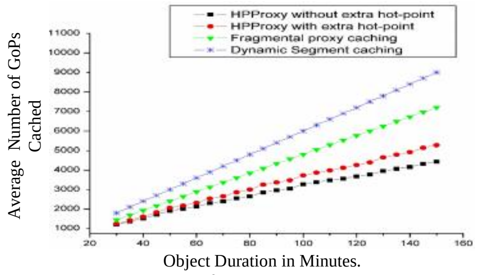
In iHPPrxoy simulation as in HPProxy, the size of the prefix is fixed as 30 GOPs which is playable for 15 seconds. The prefix is sufficient to fetch the remaining non cached GOPs of successive sectors to provide continues content delivery constant bit rate and single layering video objects are used in the simulation study.

5.1. Cache Space Requirement Analysis

The average cache space requirement of iHPProxy is compared with existing three modes for the above mentioned optimum group hot-points and various object sizes in the unit of minutes are shown in the Figure 4. Figure 4-a shows the average cache space requirement for each object duration varying by 5 minutes and starting from 30 minutes to 150 minutes. Each object duration work under all the hot-points from the optimum hot-point group. The cache space requirement of fragmental proxy caching and dynamic segment caching is still same as discussed in HPProxy. Hence, we need to compare how the cache space utilized by HPProxy and iHPProxy? As from the Figure 4-a, maximum 33.4% cached by HPProxy and 34.4% cached by iHPProxy for 30 minute object. The percentage of cache requirement is decreased for the object size increases, because the number of hot-points still from the same group. The minimum 24.6% cached by HPProxy and 29.3% cached by iHPProxy for maxim object size 150 minute. The average cache requirement is 27.97% for HPProxy and 31.37% for iHPProxy. Hence, the iHPProxy consumes 3.2% more cache space than HPProxy.



a) Cache space requirement for each object duration under all optimum hot points.



b) Cache space requirement for each optimum hot-point under all object duration.

Figure 4. Cache space requirement comparison for all four models.

In the same manner, Figure 4-b shows the average cache space requirement for each hot-point of optimum hot-point group {20, 25, 30, 40, 45, 50, 60} work on object duration from 30 minutes to 150 minutes. Our simulation results show that, the HPProxy model consumes cache space 24.05% as a minimum, 29.83% as maximum and 26.93% as average. The iHPProxy model consumes cache space 29.15% as a minimum when objects are divided by less number of hot-points, 32.5% as a maximum when objects are divided by the maximum number of hot-points and 30.73% as average. This result also shows that our improved model iHPProxy consumes only 3.79% more cache space than HP Proxy. From the cache space requirement, we understood that the placement of extra hot-point works well for shorter and longer object size and only consume 3.5% more cache space as an average.

5.2. Shift Distance Analysis

Before going to analyze the average shift distance carried out on both models, we discuss the number of extra hot-points placed based on our algorithm explained in Algorithm 1. The extra-hot points placed are calculated for the object sizes varying from 30 minutes to 150 minutes and shown in Figure 5. Each object is divided by all hot-points from optimum hot-point group.

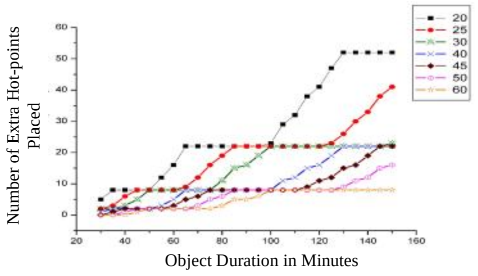
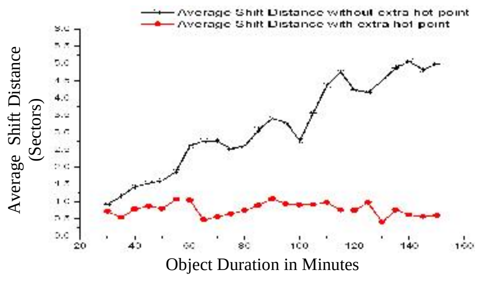


Figure 5. Number of extra hot-points placed for various object durations.

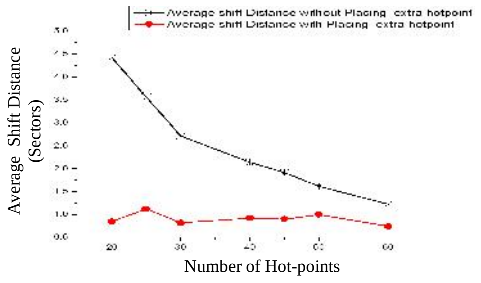
Minimum number of extra hot-points placed for the above object durations are 5, 2, 2, 1, 0, 0, and 0 for hot-points 20, 25, 30, 40, 45, 50, 60 respectively and maximum number of extra hot-points placed are 52, 41, 23, 22, 22, 16 and 8. The average number of extra hot-points placed is 28, 19, 15, 11, 9, 6, and 5. From the simulation results, we understood that maximum number of extra hot-points is placed for selecting a low number of hot-points and good enough for high number of hot-points.

Figure 6 shows that the average shift distance is much deviated when the object size is increased and a number of hot-point values is decreased by HPProxy model, whereas the iHPProxy model always maintain a constant deviation for any size object and for any value of hot-points. The average shift distance for each object size under all hot-points from the optimum hot-point group is 3.1 for HPProxy model and nearly 0.8

for iHPProxy model. In the same manner, the average shift distance for each hot-point from the optimum hot-point group under various object sizes is 2.5 for HPProxy model and nearly 1 for iHPProxy model. From the simulation results, our model produced better user satisfaction on the random seeks without much deviation from the hit location.



a) Average shift distance for each object duration under all optimum hot-points.



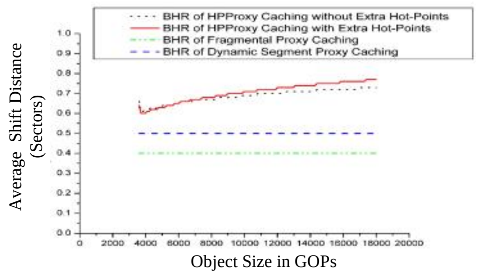
b) Average shift distance for each optimum hot-point under all object duration.

Figure 6. Average shift distance comparison.

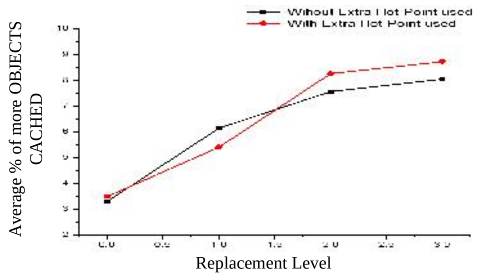
5.3. Byte Hit Ratio and Cache Replacement Analysis

In proxy caching another important aspect of video streaming is Byte Hit Ratio (BHR) of the cached part of the video. It determines the percentage of video served from the proxy to the user. As we did in HP Proxy model, here also used the same measurement to determine the percentage of video served. The BHR obtained for the object sizes varies from 3,600 GOPs to 18,000 GOPs and each object is running with all hot-points from the optimum group of hot point. Figure 7-a shows the BHR percentage is calculated based on each level of hot point caching using different percentage of user hits. The HP Proxy model produced 66% to 74% BHR whereas iHPProxy produced 64% to 77% BHR. HP Proxy and iHPProxy models produced 68.7% and 70% of the average BHR percentage respectively. Our iHPProxy model also improves BHR compare to other three models studied.

In our iHPProxy study, our earlier developed WRCP cache replacement model used to find out the performance of cache replacement and compared with HPPorxy. The cache replacement comparison is shown in Figure 7-b. The WRCP model did not evict entire object instead replaces part of the cached GOPs from the proxy server in case of better weighted ranked objects. The very worst weighted objects evicted entirely from proxy by the WRCP mode. Figure 7-b shows that the iHPProxy model produced better cache replacement over HP Proxy. HP Proxy model cached 2040 GOPs for the mean object size 7200 GOPs whereas iHPProxy cached 2280 GOPs for the same mean object size. The HP Proxy model could accommodate 6.25 more objects whereas the iHPProxy model accommodates 6.38 more objects in the proxy server. Hence, BHR and cache replacement are producing better performance.



a) BHR percentage for each object size under all optimum hot-points.



b) Average percentage of more objects cached for all four replacement level.

Figure 7. BHR and cache replacement comparison.

6. Conclusions

In this paper, we proposed a new model to improve the performance of our earlier developed HPProxy model and named as iHPProxy. The iHPProxy model supports random seek with minimum deviation from the user hit location to start the play immediately from shifted location. In the HPProxy model, the distance between two consecutive hot-point sectors was very high for longer objects which are divided with minimum number of hot-points. Hence, the user hit might be shifted long distance to place on the nearest hot-point location to provide immediate play. This long distance shift might dissatisfy the user to watch. To remove this barrier and provide shorter deviation, we placed additional sub level hot-points between two existing sub-level hot-points when there is a long distance.

The placement of extra hot-points occupies 3.2% more cache space on the proxy server than HPProxy model. But, the iHPProxy model increases BHR percentage and always maintains the same shift deviation while user performing the random seeks. The average shift distance for various object sizes and various optimum hot-points are nearly 1 sector which yields only 2 seconds shift deviation. The iHPProxy model increased the 2% of BHR than HPProxy model. iHPProxy model with WRCP cache replacement also provides additional cache space in the proxy than HPProxy. Hence, iHPProxy model improves the performance of HPProxy model in the area of user satisfaction of random seek with very smaller deviation, increased BHR percentage, better cache replacement even it consumes little more cache space.

References

[1] Abbasi U. and Ahmed T., "Architecture for Cooperative Prefetching in P2P Video-on-Demand System," *International Journal of Computer Network and Communication*s, vol. 2, no. 3, pp. 126-138, 2010.

[2] Anusuya K., Thirunavukkarasu, K., and Sundaresan S., "Implementation of Adaptive Buffer in Video Receivers Using Network Processor IXP 2400," *The International Arab Journal of Information Technology*, vol. 6, no. 3, pp. 289-295, 2009.

[3] Brampton A., MacQuire A., Fry M., Rai I., Race N., and Mathy L., "Characterising and Exploiting Workloads of Highly Interactive Video-on-demand," *Multimedia Systems*, vol. 15, no. 1, pp. 3-17, 2009.

[4] Chen S., Shen B., Wee S., and Zhang X., "Segment-based Streaming Media Proxy: Modeling and Optimization," *in IEEE Transactions of Multimedia*, vol. 8, no. 2, pp. 243-256, 2006.

[5] Du F., Jiao Y., Jean E., and Hurson A., "Cooperative Caching on Location Aware Query Proxy," *in Proceedings of International Conference on Computer Science and Software Engineering*, Wuhan, pp. 576-581, 2008.

[6] Gao B., Jansen J., Cesar P., and Bulterman D., "Beyond the Playlist: Seamless Playback of Structured Video Clips," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1495-1501, 2010.

[7] Gopalakrishnan N. and Jayarekha P., "A Rank Based Replacement Policy for Multimedia Server Cache Using Zipf-Like Law," *Journal of Computing*, vol. 2, no. 3, pp. 14-22, 2010.

[8] Ip A., Liu J., and Liu J., "COPACC: An Architecture of Cooperative Proxy-Client Caching System for On-Demand Media Streaming," *in Proceedings of the 4th International IFIP-TC6 Networking Conference*, Waterloo, pp. 70-83, 2007.

[9] Kang B., Lee E., and Park S., "Popularity-based Partial Caching Management Scheme for Streaming Multimedia on Proxy Servers over IP Networks," *in Proceedings of International Conference on Ultra Modern Telecommunications and Workshops*, St. Petersburg, pp. 1-7, 2009.

[10] Lee H., An B., and Kim E., "Adaptive Prefetching Scheme Using Web Log Mining in Cluster-Based Web Systems," *in Proceedings of IEEE international Conference on Web Services*, Los Angeles, pp. 902-910, 2009.

[11] Ossa B., Sahuquillo J., Pont A., and Gil J., "An Empirical Study on Maximum Latency Saving in Web Prefetching," *in Proceedings of International Conference on Web Intelligence and Intelligent Agent* Technology Milan, pp. 556-559, 2009.

[12] Ponnusamy P. and Karthikeyan E., "An Observed Study on Improved Caching by Adaptive and Partial Aggressive Prefetching," in *International Journal Computer Science and Application*, vol. 1, pp. 190-194, 2010.

[13] Ponnusamy P. and Kathikeyan E., "Cache Optimization on Hot-Point Proxy (HPProxy) using Dual Cache Replacement Policy," in *Proceedings International Conference on Communications and Signal Processing*, Chennai, pp. 108-113, 2012.

[14] Ponnusamy P. and Kathikeyan E., "Cache Optimization on Hot-Point Proxy (HPProxy) using Weighted-Rank Cache Replacement Policy," *ETRI Journal*, Korea, vol. 25, no. 4, pp. 687-696, 2013.

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

[16] Shen L., Tu W., Steinbach E., "A Flexible Starting Point based Partial Caching Algorithm for Video on Demand," *in Proceedings of IEEE Conference on Multimedia and Expo*, Beijing, pp. 76-79, 2007.

[17] Swain D., Paikaray B., and Swain D., "AWRP: Adaptive Weight Ranking Policy for Improving Cache Performance," *Journal of Computing*, vol. 3, no. 2, pp. 209-214, 2011.

[18] Tu W., Steinbach E., Muhammad M., Li X., "Proxy Caching for Video-on-Demand Using Flexible Starting Point Selection," *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 716-729, 2009.

[19] Yu J., Chun C., Xu D., and Wang T., "Internal Popularity of Streaming Video and its Implication on Caching," *in Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, Vienna, 2006.

[20] Wang D. and Liu J., "A Dynamic Skip List-Based Overlay for On-Demand Media Streaming with VCR Interactions," *IEEE Transactions on*

Parallel and Distributed Systems, vol. 19, no. 5, pp. 1-12, 2008.

[21] Wang J. and Yu P., "Fragmental Proxy Caching for Streaming Multimedia Objects," *IEEE Transactions of Multimedia*., vol. 9, no. 1, pp. 147-156, 2007.

[22] Wang W., Xu T., Gao Y., and Sanglu Lu S., "Probabilistic Seeking Prediction in P2P VoD Systems," available at: http://cseweb.ucsd.edu/~tixu/papers/ai09.pdf, last visited 2009.

Ponnusamy Pichamuthu received his Ms degree in Computer Applications from Bharathidhasan University, India and the MPhil degree from the Alagappa University, India. Currently, he is working as an Associate Professor in Department of Computer Applications at Adhiparasakthi Engineering College, Anna University, Chennai, India. He has published 5 papers in International Journals and 10 papers in Conferences. His research interests are video streaming, network security, mobile computing and peer-to-peer networks.

Karthikeyan Eswaramurthy received his PG degree from Bharathidhasan University, India and PhD from Gandhigram University, Dindigul, India. His area of research is network security and cryptography and advanced networking. He has published 17 papers in International Journals and more than 15 conferences National and International level. He has also published a book entitled "Text Book on C: Fundamentals, Data structures and Programming" by PHI. He is an Editor-in-Chief for an International Journal of Advanced Networking and Applications.