International Journal of Power Control Signal and Computation(IJPCSC)
Vol 8. No.2 (2016) Pp.166-174
©gopalax Journals, Singapore
available at : www.ijcns.com
ISSN: 0976-268X

Improving the Performance of HPProxy Caching using Clustered Prefetching

Dr. S. P. Ponnusamy[1] and B. Devanathan[2]
[1]Assistant Professor, Department of Computer Science,
[1]Thiruvalluvar University Model Constituent College of Arts and Science, Tittagudi, Tamil Nadu, India.
[2]Assistant Professor/Programmer, Computer Science Wing, DDE,
[2]Annamalai University, Chidambaram, Tamil Nadu, India.
[1]spponns2k1@rediffmail.com, [2]sakthidevanathan@gmail.com

Abstract ***–In recent years, the interest of Internet users turned into watching videos such as video-on-demand, online movies, news, online learning, etc. To improve the performance of media playability, researchers developed prefetching scheme which fetch the media well ahead before the request of the media and stores in the proxy server. We proposed adaptive linear predictor and clustered prefetching model on a Hot-Point Proxy (HPProxy) using k-level prefetching. Our model uses Group-of-Pictures (GOPS) as smallest playable unit and prefetches the group of non cached GOPs in linear manner based on the output of adoptive linear predictor from the current viewing position. The prediction and prefetching work is determined by fixing three windows; Seek-Boundary-Window (SBW), playBack-Boundary-Window (BBW) and Prefetch-Boundary-Window (PBW). The PBW and SBW will be moved in forward direction when the BBW is moved due to the play of the media. The PBW and SBW will be located into new location immediately when user jumps into new location which always fall any one of the hot-point either major or inner sub levels. Our study is performed on different level of prefetching based on k-value and show that the storing cache space is increased to and Byte-Hit-Ratio (BHR) factor is increased.***

Keywords **– Proxy Caching, Prefetching, Hot-Point Level, Multimedia streaming, Video-on-Demand.**

1. INTRODUCTION

It is necessary to provide more attention on this filed to offer multimedia streaming to the user without break with good Quality-of-Service (QoS). This QoS media streaming increased the Internet traffic, huge consumption of bandwidth, and maximum latency between client and server due to the above mentioned parameters. The implementation of CDN is very expensive and difficult to replicate in all the places which are closer to the clients. e factors while transferring media to clients from media servers. The transferring of media content is a big challenge since the Internet provides the best-effort service which does not provide the guaranteed best quality of service. In addition to continuous playability, the streaming must support forward and backward playback from current location, and random seek i.e., jumping from current location to any location in forward and backward manner. To improve the performance on multimedia content delivery, the content providers place the replication of media using Content Delivery Network (CDN) which is closer to the client group. The implementation of CDN is very expensive and difficult to replicate in all the places. The replication of CDN servers need to be coordinated to provide same services and QoS to any client compare with all clients. So, many researchers involved on this area and developed proxy caching scheme which stores the part of the object into a proxy server. Even though the proxy caching improves the media streaming services by reducing network traffic and bandwidth conservation, the continuous playing of media is still lacking.

The proxy caching for web content such as HTML files contain text and images has been provided the better performance and the same technique cannot be offered for the multimedia content delivery due to the large size. For that reason, the media objects need to be divided into smaller chunks of independent playable content. The researchers studied various proxy caching schemes [22] and provide many caching schemes such as hot-point proxy caching [1], seamless playback caching [4], fragmented proxy caching [15], prefix-suffix caching [16], cooperative proxy caching [17], segment based caching [18] and so on. The cache replacement polices also involved in the above

factor to provide frequent requested fragments. Caching schemes don't always guarantee the continuous delivery because the portions to be viewed may not be cached in the proxies on time.

The layered encoding techniques could be followed to reduce the delay in play back by varying QoS of the content based on the availability of bandwidth, serving capacity of media server and receiving capacity of the client. To achieve these, researchers [19], [20] used quality based approaches by keeping multiple qualities of video streams. These video streams are provided to the clients depending on the above said three parameters. The lower bandwidth connection clients receive low QoS videos and high bandwidth connection clients receive high QoS videos which have enriched layers of videos.

To further improve the performance of continuous delivery, prefetching schemes are used i.e., fetch-ahead the media before the request of the media by the client. Many prefetching schemes haven been proposed such as seamless playback prefetching [4], cooperative prefetching [5], adaptive and partial aggressive prefetching [6], web prefetching [7], adaptive prefetching [8], aggressive prefetching [11], distributed prefetching [12], optimal prefetching [13] and so on. The prefetching schemes overcome the limitation of cache mechanisms through preprocessing contents before a user request comes. Prefetch schemes expect future requests through log file analysis and prepare the expected requests before receiving it. Compared with web cache schemes, prefetch schemes focus on the spatial locality of objects when current requests are related with previous requests.

However, despite these benefits, three difficulties prevent prefetch schemes from being exploited in web cluster systems [8]. First, it is difficult to find which objects are related with the incoming requests. At the server side, access patterns are dynamic because of the web cache mechanism. Second, it is difficult to find an optimal prefetch rate. Too aggressive prefetch schemes may hurt overall performance due to the shortage of memory. Finally, a prefetch scheme in a system should be considered along with an efficient resource management. The inappropriate resource management drains the resource of one backend server, while other backend servers have the available resources. To overcome these difficulties, we propose an adaptive and partially aggressive web prefetch scheme over proxy cache. The adaptive prefetching scheme defines which media chunks to be prefetched and adopts the dynamic change in media access pattern in server. The partial aggressive prefetching technique used to prefetch the media chunks in a interleaved manner.

2. RELATED WORK

The prefetching mechanisms are used to try to avoid the discontinuity of streaming media delivery by prefetching the portions to be viewed while clients are viewing current positions. Ponnusamy S P et al [1] developed a Hot-Point proxy caching model to support playback with random seek. The authors primarily considered for preserving more cache space to keep more objects. The object was divided into many numbers of Group-of-Pictures (GOPs) which were individually playable and 6 numbers of GOPs form a sector. Sectors were grouped as hot-point region based on the number of hot-points. Each hot-point region wsa further divided by sub level hot-points and fixed in the start of sector. The caching of a sector is determined based on the level of hot-points. More GOPs cached at major hot-points and less GOPs cached at inner levels. K.-Y. Wong et al.[3] proposed web caching using site-based approach decides the caching of objects based on the website instead caching the objects alone. The approach is applied on all types of networks like Local Area Network (LAN), Wide Area Network (WAN), etc. The model achieved 21-50% of reduction in disk access time compared to conventional URL-based caching.

Recent studies on prefetching schemes [2, 5, 6, 7, 8, 10, 11, 12, 13, 14, 18] used to improve the performance of continuous delivery by fetch-ahead the media even before the request of the media by the client is made. The proxy caching scheme combined with prefetching schemes improves the latency considerably, while proxy caching alone offers less latency improvement. This combination makes the seamless playback, i.e., users allowed to jump from one point to another point, of media content without interruption or delay. It also improves the response time and user experience.

Chao Liang et. al [2] presented a Adaptive Taxation scheme to normalize the prefetching on heterogeneous systems. The bandwidth of a system was treated as its wealth and the video prefetching of the systems were treated as income. Resource-rich system contributes more bandwidth to the network and finance for the resource-poor systems. The tax-regulated redistribution of system wealth helps improve the social welfare, and then, reduce server cost. A distribution protocol with the adaptive taxation is to be presented to balance the systems due to the dynamic nature of systems.

Ubaid abbasi et. Al [5] proposed a cooperative prefetching algorithm to support the prefetching between the proxies which are cooperated themselves. In this technique, each peer maintained a record of playback segments and details of segments cached by other peers in the same session. The record of segments maintained without redundant segments. If any one of segments was missed in the session, the missed segment was fetched from shortcut neighbor list peers. This missed segment was obtained by broadcasting rare segment request from current peer to the shortcut neighbor list peers based on the current playhead location. In the same manner all contents were played by cooperating themselves.

Ponnusamy et. al [6] presented a partial aggressive prefetching algorithm in which two engines namely adaptive predictor engine and prefetcher used for prefetching prediction and fetching predicted chunks. The adaptive predictor frequently gets updates from media server and updates its prediction for

future. The prefetcher got the predicted streaming media from the predictor and verifies it in the cache to look the chunks which were already cached. After verification the prefetcher prefetch the media chunks partially in to the cache from the media server. The prefetcher did not fetch the entire media into cache, instead it prefetch 10% of media in continuous manner from first, middle and last part of media and fetch the media chunks alternatively in the remaining part used for latter request.

Heung Ki Lee et al. [8] proposed the adaptive prefetching model with three components; Double Prediction-by-Partial-Match Scheme (DPS), Adaptive Rate Controller (ARC) and Memory Aware Request Distribution (MARD). The results produced 10% of performance improvement on average in various workloads. Domenech et al. [9] studied the impact of the web architecture on the limits of latency reduction. They concluded that latency reduction depends on the predictor location: it can be reduced by 36%, 54%, and 67% when the predictor is located at the server side, client, or proxy, respectively. Latency reductions higher than 90% could be obtained if the predictor works collaboratively at different elements of the architecture.

Balamash et al. [10] proposed a mathematical model for a web prefetching architecture. The model dynamically computes the "optimal" number of documents to prefetch in the subsequent client's idle (think) period. Their results showed that prefetching were profitable even with the presence of a good caching system. Junli Yuan et al [11] projected the aggressive prefetching scheme aims to reduce the startup delay for first-time accessed objects by aggressively prefetching them in advance. To ensure the prefetching accuracy, they introduced the assistance of the media servers by having the servers to locate the most popular media objects and provide such information to proxies as hint for prefetching. To handle the large size problem, they adopted the segment prefetching mechanism. Trace-driven simulation results show that this scheme can effectively reduce the ratio of delayed requests by up to 38% while introduce very marginal increase in traffic.

Yifeng He et. al [13] proposed an optimal prefetching which used 2-D segment access probability, considering both starting position and destination of the seeking behavior. The system collects the seeking information in a distributed manner and maintained the updated cache set. Yoshihisa T et. al [14] proposed a division based prefetching model to support video broadcasting systems. In this paper they presented two kinds of prefetching model to support Division-based Hybrid Streaming Delivery (DHSD) with Full (DHSD-F) and Modular (DHSD-M). In DHSD-F, the full part the segment was prefetched while fetching a segment from server whereas in DHSD-M, the ¼ part of segment was prefetched while one segment was fetched.

All the prefetching models prefetch the look-ahead media part based on the popularity of the media segment and support dynamic seek. This kind of media popularity will be obtained after thousands of user logs collected and kept in the media server. There are two drawbacks in prefetching model which used the popularity.

i). Not possible to obtaining popularity of segments of a media from initial uploading time to certain period of time (the time needed to collect access logs) and we considered this kind of media as zero-popular (PM-0) media.

ii). Some of the media segments were denoted as poor popular segments and we denoted as zero popular (ps-0) segment. But, these segments may be needed by the user when the user seeking a location in the play dynamically if there is no idea about the popularity of the segment.

Our paper depicts a prefetching model to support PM-0 Medias and ps-0 segments with dynamic seeking position in both forwarding and backwarding directions from major and inner level hot-points using the Hot-Point Proxy (HPProxy) caching model [1] where the entire media was equally divided using the predefined number of hot-points. Our system does not prefetch aggressively and does not increase the traffic in a great deal.

3. SYSTEM MODEL

Usually, the proxy server cache the media which is already streamed to one of its clients and served latter if another client is requested the same media. The proxy server preserves memory for the caching. In our architecture, the proxy also uses the cache memory for storing the prefetched media GOPs. The prefetching process is divided in two major mechanisms such as a predictor engine, and a prefetching engine. The predictor engine predicts the next user accesses pattern by using previous logs information. Generally, the log information is kept in the media server which is severed for proxy server to prefetch prediction. Since our model considering the non popular media and segments, the predictor engine predicts the user's next access using the user's current access on the same playback. The proxy server takes care of predicting and prefetching the media GOPs. In our architecture, the Adaptive Linear Predictor (ALP) and Clustered GOPs Prefetcher (CGP) are situated in the proxy server and shown in fig.1. The ALP-CGP model is applied on the hot-point sectoring model [1]. In this paper, we have considered 8 GOPs forms a sector to easily represent the cached status of GOPs in 8 bit binary form.

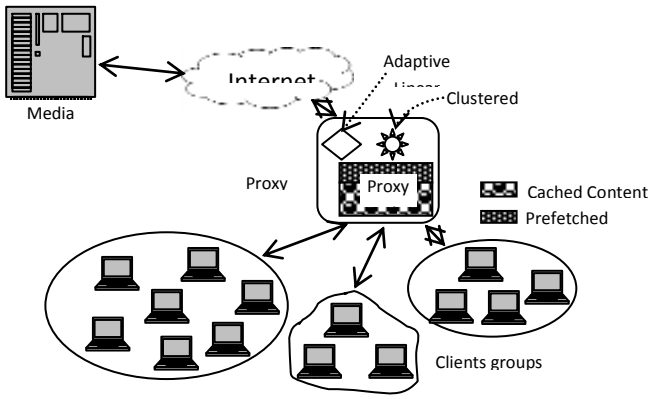


Fig. 1. Prefetching Architecture

A. Boundary Window System

In practice, most of the user hits follows the first case of linear fashion and the second case of seeking pattern happened on a small number of occasions. To satisfy both kind of seeking fashion, we introduce four kinds of windows; playBack-Boundary-Window (BBW), Prefetch-Boundary-Window (PBW), Seek-Boundary-Window (SBW) and Prefetch Sliding Window (PSW). Since the predictor does not know any idea about the access pattern on PM-0 Medias and ps-0 segments, it uses the above windows are putting in place. The clustered prefetcher works based on the predictor definition on the windows boundary and prefetch the media GOPs in clustering fashion. The clustering of media GOPs is a group of GOPs determined from the successive sectors of the PSW. The time frame of each window is illustrated in the figure 2.

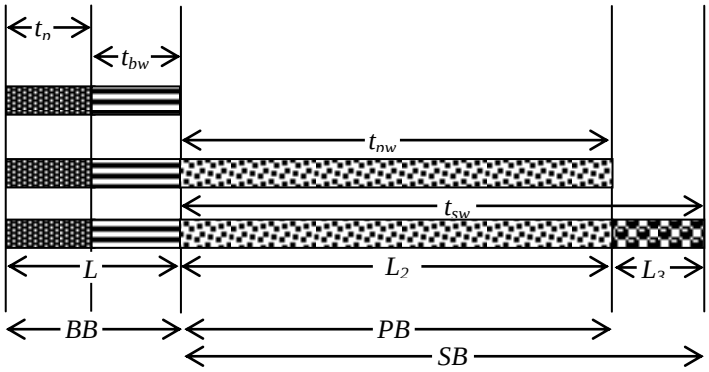
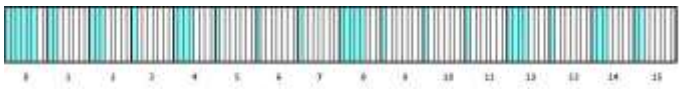


Fig 2. Time frame for all three windows

The L1, L2 and L3 illustrate the priority level (pli) of prefetching applied on the three windows in such a way that pl1>pl2>pl3. Since the tp is current playing window, all the GOPs must be prefetched or cached earlier than others. If part of GOPs is not cached in tp, top priority pl1 is assigned by predictor to prefetch the GOPs from server. The next level prefetching priority pl2 is assigned to the tbw, if there is any part of GOPs is not cached or prefetched. The prefetch boundary window's prefetch priority is assigned in third level because the PBW level GOPs are played in feature. Least priority is assigned to the SBW-PBW part due to the later played area. Always the tpw is assigned much greater time than other three time frames for the reason that the SBW is set for long duration than paying window.

B. GOPs availability and Request to media representation

As we enlightened in section 3, each sector consists of 8 GOPs instead of 6 GOPs as stated in [1]. The availability status in proxy of GOPs of a sector is stored into an external file in the proxy server. One bit is enough to know the status that 1 is to represent availability and 0 is to represent non availability. Since the unsigned character uses 8 bits to store its value, we can store availability 1 sector's GOPs availability status. So, one sector GOPs takes only one byte in memory for 4 seconds video. For example, 1½ hour movie requires 1350 bytes additionally (nearly 1 kb) to store the GOPs availability. The figure 3 gives you an idea about all representation. The figure 3(a) shows the cached GOPs in a hot-point region R with 16 numbers of sectors, figure 3(b) shows its corresponding binary representation and figure 3(c) shows corresponding character representation.



(a). Cached GOPs in a hot-point region *R*

11111100 11000000 11100000 10000000 11110000 10000000
10000000 10000000 11111000 10000000 10000000 10000000
11110000 10000000 1110000 11000000

(b). binary representation of Cached GOPs

üÀà€ð€€€ø€€€ð€àÀ

(c). Character representation of Cached GOPs in a text file

Fig 3. Representation of GOPs availability status

The same character group pattern is stored in the entire text file for all hot-point regions when there is no prefetching done for all processed sectors [1] and varying character pattern for unprocessed sectors. It is enough to store two set of character group to store with number of repetition in the file. But the bit pattern is varying while performing prefetching on various levels in different boundary system as discussed in section 4C. So, it is necessary to store the representation for all sectors in the file. The character is read from the file for every reference of the sector to know the GOPs cached availability. The corresponding bit in the binary representation is altered every time a GOP of a

sector is cached (1) or replaced (0) and finally the corresponding character is stored in the same location of the file. In our simulation, a group of sector GOPs can be altered from the current location and the corresponding group of altered characters stored from the current location.

Fig 4. Sliding of PSW on every sector is prefetched

The list of GOPs needed from server should be decided by the predictor and submit it to the prefetcher. This list is prepared based on the current playing location's GOPs available status in the proxy cache. Initially, the PSW is set to BBW+PBW+SBW, because there is chance of some GOPs will not be available in BBW when the user hit is placed on out of SBW. So, the predictor decides more GOPs prefetched in the BBW than other twos. The PSW is adjusted as PBW+SBW when all the GOPs are prefetched on BBW. The PSW is automatically slide in to the next sector in forward direction whenever the entire sector GOPs are prefetched before playing. This automatic slide of illustrated in the figure 4. The request set is prepared in pair within in the PSW for all three windows with the starting sector number and bit pattern for number of GOPs to be prefetched. This is represented as follows for two cases;

{(snBBW, gop_bit_pattern), (snPBW, gop_bit_pattern), (snSBW, gop_bit_pattern)} where pl1>pl2>pl3

if GOP needed from BBW

{ (snPBW, gop_bit_pattern), (snSBW, gop_bit_pattern)} where pl1>pl2>pl3

if no GOP needed from PBW

where sn represents the starting sector number in the corresponding boundary window. If no GOP needed to prefetch in BBW window, only two pairs are represented in the prefetch request to the server. It is happened when the BBW is fully prefetched within the PSW. The bit pattern represented with 0s and 1s for no need to prefetch and need to prefetch respectively.

4. Adaptive Linear Predictor and Clustered GOP Prefetcher Model

The adaptive linear predictor predicts the next access pattern from recent seeking positions on the current media playback and adopts current location for prediction. The prediction from current location is predicted in linear manner which falls from previous hit location to new location in any one side. If there are no previous hit locations found on current playback media, the default boundary will be set. The seeking pattern may be as follows;

i). 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 media (seek distance may be very short),

ii). seeking in non-linear fashion; moving in forward and backward directions in random manner to find the desired location and play (seek distance may be long for first movement and short for successive movements).

The prediction and prefetching processes are very simple for the first case and difficult for the second case.

A. Shift distance

As discussed in *ponnusamy et al* [1], the video files are divided by predefined number of hot-points and these hot-points are placed at equal distances in the video called as major hot-points. The region between two hot-points is called as hot-point region. In turn, each region is divided into sublevel hot-points. These major and sub-level hot-points are used as hit-points. The major hot-point is placed on a start of a sector and sub-level hot points are placed inside the sectors. In some cases, the user hit may fall on non hot-point sector which cached the minimal number of GOPs. In this situation user view-point is shifted to nearest hot-point sector GOP in backward direction and starts play instead the actual hit location's GOP. This shifting from actual hit location to shifted hot-point location is called as shift distance and denoted by *d*. It is calculated as

$$d = \varepsilon_{al} - \varepsilon_{hp} \quad \ldots\ldots\ldots (1)$$

where ε_{al} is actual hit GOP's location value and ε_{hp} is the shifted hot-point's starting GOP's location value.

TABLE I
NOTATIONS

Symbol	*Notations*
O	*Video object*
W_i	*Size of Object i*
s_i	*i^{th} Sector*
h	*Hot point*
O_n	*Number of video objects*
R_i	*i^{th} hot-point region*
k	*number of hot point level in a GOS*
h_l	*hot point level within a region , where $1 \leq l \leq k$*
t_p	*time frame of current playing window*
t_{bw}	*time frame of immediate playback boundary window*
t_{pw}	*time frame of prefetch boundary window*
t_{sw}	*time frame of seek boundary window*
pl_i	*i^{th} level priority applied on various windows*
d	*shift distance*
ε_i	*i^{th} GOP in a R*
t_m	*movement time from one location to another location*
t_s	*shift time*
σ_i	*cumulative access pattern value of an object O_i*

B. Access pattern percentage and hit ratio of a hot-point

The access pattern of the media what we have considered is unavailable at the starting stage. So the predictor cannot predict any user's access pattern at the earliest and it is available in later stages after accessing the videos several times. In the initial, the access pattern value is considered as the level value of all hot-point sectors i.e., the major level hot-points are having maximum value and sublevel hot-points are having lesser value depends on the inner level as depicted in the figure 2. The access pattern of every hot-point sector is evaluated for every access or hit by the user for both forward and backward direction. The access pattern percentage on each hot-point is calculated based on the number of times user's view starts from that hot-point for both direct hit and shifting. The forward access pattern percentage of an i^{th} hot-point (Δ_i) is calculated as

$$\Delta_i = \frac{\omega}{fd(hn_i)} \qquad \text{..... (2)}$$

where ω is the total number of hits on the media and $fd(hn_i)$ is the number of hits placed on i^{th} hot-point hp_i in forward direction. The backward access pattern percentage of an i^{th} hot-point (∇_i) is calculated as

$$\nabla_i = \frac{\omega}{bd(hn_i)} \qquad \text{..... (3)}$$

where $bd(hn_i)$ is the number of hits placed on i^{th} hot-point hp_i in backward direction.

The hit ratio (r_i) of an i^{th} hot-point hp_i is calculated based on the number of hits (dh_i) placed directly on the hp_i in both direction versus the number of hits (sh_i) placed on s_i and shifting into the hp_i indirectly in both direction. The r_i is evaluated as

$$r_i = \left(\frac{dh_i}{dh_i + sh_i} \right) \qquad \text{..... (4)}$$

The hot-point hp_i is adjusted to the s_i if the r_i is less than 0.5. Since the hp_i is hit lesser time than the sector s_i by the user. Later stages, the prefixed hot-points automatically placed in more accessed locations and meets the users' random seek directly.

C. Access Ratio of prefetched GOPs

Our prefetching scheme prefetches the non cached GOPs in linear manner within in the *SBW* until the user jump into another location. Due to this arbitrary jump, some prefetched GOPs or Sectors might not be used. It is very important to produce the good access ratio while prefetching without wasting the bandwidth. More part of the prefetched GOPs is used when the access ratio is high or low otherwise. The access ratio (a_r) of the prefetched content is defined as

$$a_r = \frac{g_p}{g_u} \qquad \text{..... (5)}$$

where g_p is the total number GOPs prefetched in SBW and g_u is the number of GOPs used to play by user.

D. *Cache Replacement Policy*

The cache replacement is done for optimizing cache space when there is not enough memory available for storing new cached or prefetched GOPs in the proxy cache. Here we are applying the earlier developed model WRCP [21] on perfected GOPS. There are four possible level of victimization can be used depends on the need of memory and depends on the availability of the GOPs in the proxy cache. The four possible victim levels are;

i). prefetched and cached GOPs of non hot-point sectors
ii). prefetched and cached GOPs of sublevel hot-point sectors
iii). prefetched and cached GOPs of major level hot-point sectors
iv). One trailing available GOP of sectors for every need of cache replacement

The algorithm for prefetching and replacing of GOPs based on hot-points division is shown in the table 2.

TABLE II
PREFETCHING AND REPLACING

1. **while** user request hits on a place
2. set the BBW, PBW, SWB and PSW attributes from current hit location
3. Load the availability GOP status in PSW from the file
4. **repeat**
5. Prefetch GOPs based on $pl_1 > pl_2 > pl_3$ using request pattern RP_t
6. RP_t is designed such that $(sn_{XBW}, gop\_bit\_pattern)$ for all *BBW*, *PBW* and *SBW*
7. Apply hybrid cache replacement when cache space is needed
8. Move *PSW* by 1 sector when s_i is fully fetched
9. Update the GOPs availability status in character form into a file
10. Update Δ_i and ∇_i
11. Find r_i using $\left(\frac{dh_i}{dh_i + sh_i} \right)$
12. Adjust *h*i into s_i when $r_i(h_i) < 0.5$
13. **until** new jump into new location or end of play
14. **end while**

5. ANALYSIS

In this section, the performance of Hot-Point proxy caching is evaluated based on the simulation metrics. The HPProxy is primarily focused on the proxy cache consumption for perfecting GOPs in addition to the optimum hot-point groups. The improvement of byte hit ratio of video objects is also compared with initial cached objects. The experiment results are presented on three parameters such that cache consumption, byte-hit-ratio and cache replacements with clustered prefetching and without .

A. Test Setup and Metrics

The simulation parameters are set as per the specification of the system model which is described in section 3. The The ALP-CGP model takes full responsibility of caching and prefetching actions as in a real video server. The default parameters are shown in table 3.

In HPPrxoy simulation size of prefix is fixed as 30 GOPs which is playable for 15 seconds. This playing time of prefix is sufficient to fetch the remaining non cached GOPs of successive sectors to provide continues media delivery to the client. Constant bit rate and single layering video objects are used in the simulation study.

TABLE III
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	10800 GOPs
Layering	Single
Zipf Factor	0.8

B. Prefetch Determination Analysis

The additional cache space requirements for a 90 minutes object duration under the optimum group of hot-points in a proxy server using the clustered prefetching is measured and shown in figure 5.

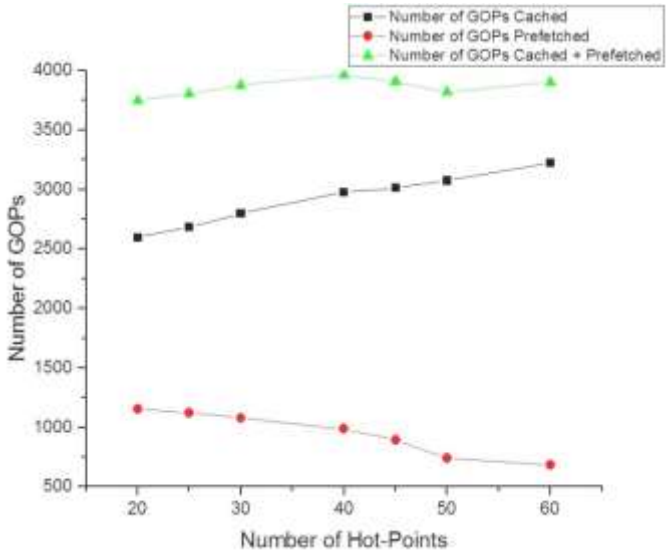


Fig 5. GOPs requirement of Clustered Prefetching

The Clustered Prefetching model requires 8.8% of additional cache space into the existing HPProxy model for an average of 10 minutes user watch.

C. Byte-Hit-Ratio Analysis

The Byte-Hit-Ratio (BHR) is an important parameter to determine the efficiency of the proxy caching. The BHR is determined as in HPPRoxy with prefetching and without prefetching. The BHR is obtained for the object sizes varies from 3,600 GOPs to 18,000 GOPs and each object is running with all optimum hot-points and shown in figure 6.

The HPPRoxy model with Clustered Prefteching model produced 85% to 95% BHR whereas HPProxy model producded 61% to 74% BHR. The low BHR value found with short

duration objects and high BHR value is obtained for long duration objects.

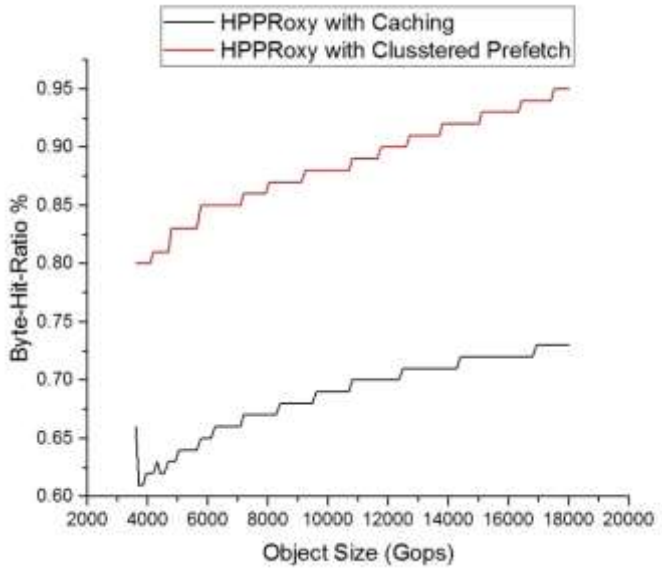


Fig 6. BHR of Clustered Prefetching

D. Cache Replacement Analysis

The cache replacement is applied for cache space requirement of new object when the cache space in the proxy server is exhaust. Initially, the proxy server occupied lesser number objects with clustered prefetching compare with objects without prefeteched.

The WRCP model applied for cache replacement on four levels with the proxy size of 21,00,000 GOPs. The HPPRoxy stores 924 objects of size 2280 GOPs cached. The HPProxy with Clustered Prefetching stores 692 objects of 3032 GOPS cached and prefetched. The comparison cache replacement is shown in figure 7.

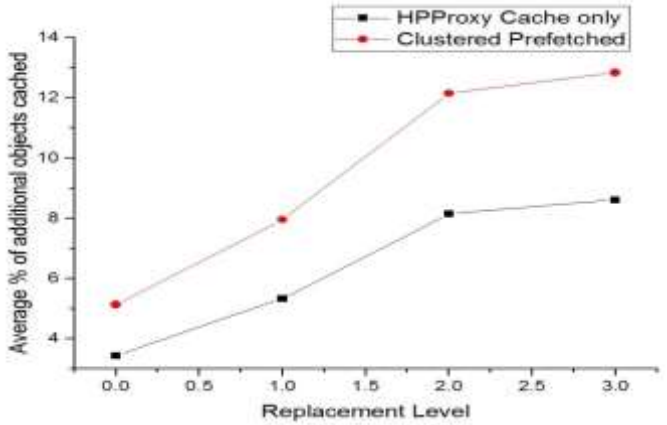


Fig 7. Cache Replacement with Clustered Prefetching

The WRCP model produced better results on Clustered Prefeteching with more GOPs eviction. The model produced minimum 5% cache space creation and maximum of 12.8% .

6. CONCLUSION

In this paper, a Clustered Prefeteching model is proposed to improve the BHR and to avoid the shifting the hit location to nearest cached GOPs. This model also reduced the load of the proxy server while playing videos with unexpected user behavior of VCR like operations. The Clustered Prefetching with HPProxy provided the better results on BHR and cache replacement with little scarification of additional cache space for slding PSW way prefetching. This cache space could be replaced by the WRCP cache replacement model.

REFERENCES

1. Ponnusamy S P, Kathikeyan E, "HPProxy:Hot-Point Proxy Caching with Multivariate Sectoring for Multimedia Streaming", *in European Journal of Scientific Research*, Vol. 68, No. 1, pp. 21-35, January 2012.
2. Chao Liang, Zhenghua Fu, Yong Liu, and Chai Wah Wu, "*Incentivized Peer-Assisted Streaming for On-Demand Services", in IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 9, pp. 1354-1366, September 2010
3. K.-Y. Wong, K.-H. Yeung, "Alternative web caching design: a site-based approach", in *The Institution of Engineering and Technology communications*, Vol. 4, Iss. 12, pp. 1504–1515, March 2010.
4. Bo Gao, Jack Jansen, Pablo Cesar, and Dick C. A. Bulterman, "Beyond the Playlist: Seamless Playback of Structured Video Clips", in *IEEE Transactions on Consumer Electronics*, Vol. 56, No. 3, pp. 1495-1501, August 2010.
5. Ubaid Abbasi and Toufik Ahmed, "Architecture for Cooperative Prefetching in P2P Video-on- Demand System", in *International Journal of Computer Network and Communications*, Vol. 2, No. 3, pp. 126-138, May 2010.
6. Ponnusamy S. P., Karthikeyan E, "An Observed Study on Improved Caching by Adaptive and Partial Aggressive Prefetching" in International Journal Computer Science and Application, pp. 190-194, vol 1, 2010.
7. B.de la Ossa, J. Sahuquillo, A. Pont and J.A.Gil, "An Empirical Study on Maximum Latency Saving in Web Prefetching", in *IEEE/WIC/ACM proceedings of International Conference on on Web Intelligence and Intelligent Agent Technology",* Vol.1., pp.556-559, September 2009.
8. Lee, H. K., An, B. S., and Kim, E. J. "Adaptive Prefetching Scheme Using Web Log Mining in Cluster-Based Web Systems", in *Proceedings of the 2009 IEEE international*

Conference on Web Services - Volume 00, pp: 902-910, July 2009.

9. J. Dom`enech, J. Sahuquillo, J. A. Gil, and A. Pont, "The impact of the web prefetching architecture on the limits of reducing user's perceived latency," in *Proc. of the International Conference on Web Intelligence*. IEEE, 2006.
10. A. Balamash, M. Krunz, and P. Nain, "Performance analysis of a client side caching/prefetching system for web traffic," *Computer Network.*, vol. 51, no. 13, pp. 3673–3692, 2007.
11. Junli Yuan, Qibin Sun, Susanto Rahardja, "A More Aggressive Prefetching Scheme for Streaming Media Delivery over the Internet," in *Proc. Visual Communications and Image Processing*, Feb. 2007, vol. 6508, pp. 650818-1–8.
12. A. Sharma, A. Bestavros, and I. Matta, "dPAM: A distributed prefetching protocol for scalable asynchronous multicast in P2P systems," in *Proc. IEEE INFOCOM*, Mar. 2005, vol. 2, pp. 1139–1150.
13. Yifeng He, Guobin Shen, Yongqiang Xiong and Ling Guan, "Optimal prefetching scheme in P2P VoD applications with guided seek", in *IEE Transactions of Multimedia*, Vol.11, No. 1, pp:138-151, January 2009.
14. Tomoki Yoshihisa, Shojiro Nishio, "An Interruption Time Reduction scheme with Prefetch for Hybrid Video Broadcasting Environments", *in Proceedings of IEEE : Wireless Communications and Networking Conference 2011 - Service and Applications,*, pp. 2071-2076, May 2011.
15. James Z. Wang and Philip S. Yu, " Fragmental Proxy Caching for Streaming Multimedia Objects" *in IEEE Trans. Multimedia.*, vol. 9, no. 1, pp. 147–156, January 2007.
16. Wei Tu, Eckehard Steinbach, Muhammad Muhammad, and Xiaoling Li, "Proxy Caching for Video-on-Demand Using Flexible Starting Point Selection", in *IEEE Transactions on multimedia*, Vol. 11, No. 4, pp. 716-729,June 2009.
17. Alan T.S. Ip, Jiangchuan Liu, and John Chi-Shing Lui, "COPACC: An Architecture of Cooperative Proxy-Client Caching System for On-Demand Media Streaming" in *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 1, pp. 70–83, January 2007.
18. Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang, "Segment-based streaming media proxy: modeling and optimization" in *IEEE Trans. Multimedia*., vol. 8, no. 2, pp. 243-256, April 2006.
19. Yongfeng Li and Kenneth Ong, "Optimized scalable cache management for video streaming system" in *Springer Multimedia Tools and Applications*, vol 44, number 1, pp. 65-86, April 2009.
20. Fang Du, Yu Jiao, Evens Jean, Ali R. Hurson, "Cooperative Caching on Location Aware Query Proxy" in *International Conference on Computer Science and Software Engineering*, vol. 4, pp.576-581 December 2008.
21. Ponnusamy S.P. and Kathikeyan E., "Cache Optimization on Hot-Point Proxy (HPProxy) using Weighted-Rank Cache replacement Policy," in *ETRI journal, Korea*, Vol.25, No. 4, pp. 687-696, August 2013.
22. Ponnusamy S. P, " A Study on Proxy Caching Methods for Multimedia Streaming", in *International Journal of Applied Engineering Research*, Volume 10, Number 2 (2015) pp. 4973-4990.