Special Issue of International Journal of Computer Applications (0975 – 8887)
on Electronics, Information and Communication Engineering - ICEICE No.6, Dec 2011

Optimize Prefetching in P2P VoD Systems

Parag Bafna   Annappa

National Institute of Technology Karnataka, Surathkal,
Mangalore-575025 India

ABSTRACT

In Peer-to-Peer Video on Demand system free VCR-like operations are very frequent. The uncertainty of frequent VCR operations makes it difficult to provide play as download services. To address this problem, we propose Optimize prefetching for Peer-to-Peer Video-on-Demand systems. Through simulation results, we demonstrate that the proposed prefetching scheme significantly reduces the seeking delay compared to the Distributed prefetching scheme.

1. INTRODUCTION

MULTIMEDIA communications has been in continuous state of evolution over the past few years. After the application level multicast, video streaming is boosted by P2P networks.
Multimedia streaming can be categorized into live streaming and on-demand streaming. In live streaming systems, the source servers broadcast videos, and all the clients are synchronous. Successful examples include CoolStreaming [1], and PPLive [2]. On the other hand, On-demand streaming or Video on Demand is an interactive multimedia service, which delivers video content to the users. Differing live streaming, for on-demand streaming, mostly clients demand different video or different parts of the same video.
Video on Demand are systems which allow users to select and watch content on demand. This results in increase seeking delay and stress on streaming server. The uncertainty of frequent VCR operations makes it difficult to provide high quality real-time streaming services. To address this problem, prefetching is proposed. Different strategies are adopted. In this paper we proposed Optimized Prefetching strategy and compared our strategy with Distributed Prefetching strategy.
The rest of this paper is organized in different sections as followed. In section II, we discussed the related work. In section III, we discussed proposed Optimize Prefetching strategy. Section IV illustrates the performance evaluation and comparison of our strategy with Distributed Prefetching strategy. Section V presents a brief conclusion.

2. RELATED wORK

Recently, on-demand streaming through peer-to-peer overlays has attracted significant attentions. A series of Prefetching strategy have been proposed to improve performance of VoD systems, including distributed prefetching[5], random prefetching[6] and etc.
The random prefetching blindly fetch segments of video from other peers. Unpredictable user viewing behaviors have not been addressed in random prefetching. As a result high seek latency.
The distributed prefetching[5], prefetch the segment according to user viewing behavior. In this technique user viewing behavior logs are maintained by a tracker server. The distributed prefetching techniques improves hit ratio by considering user's access patterns, however extracting a user viewing pattern from user viewing behavior logs require large computation to be performed by tracker server.

3. OPTIMIZE PREFETCHING

To overcome drawbacks of different prefetching techniques discussed earlier in section II, we proposed a new prefetching technique called *"**Optimize prefetching strategy**"*. In this technique, each peer maintains the record of playback segments (we divided video into segments) by other peers in the same session. This information is obtained through gossiping. Once the state information are collected from all peers (in same session), each peer creates a table of available segments in that particular session. Figure 1 shows the state information table received by a particular peer I.

Peer ID	Records
I	1,2,5,8,11,20
J	5,6,7,8,9,11
K	1,2,8,15,16,17
L	6,8,9,11,12,17
M	9,10,11,12,13
N	4,5,6,7,8,9,20
O	2,3,4,6,7,8
P	3,4,5,6,7,8,9

Fig 1: Information Received by peer I

Each peer performs the necessary computation to create a list shown below (Fig. 2). First row of the list contain segment number and second row contain number of occurrence of that segment in record. The peer then requests for a segment near to its play head position, which didn't exist in that session (count=0). In case of Fig. 2 missing segments like 10, 14, 18, and 19 would be requested from peers in other session depending on current play head position. If there is no such segment exists than peer request for a segment which has highest count in list (count is present in second row of list). If there is no response from shortcut neighbors, the desired segment is requested from server as a last resort. It is important to note that each peer also prefetch the segments near to its play head position as an urgent downloading target. If bandwidth allows, the peer also tries to fetch *anchors*. Anchors are segments each consisting of 10 continuous seconds, and are distributed throughout the media file with fixed interval (e.g. 300 seconds). The algorithm for optimize prefetching strategy is presented below.

Segment number	1	2	3	4	5	6	7	8	9	10
Count	2	3	2	4	4	5	4	7	5	0
Segment number	11	12	13	14	15	16	17	18	19	20
Count	4	2	1	0	1	1	2	0	0	2

Fig 2: List Created by Peer I.

.

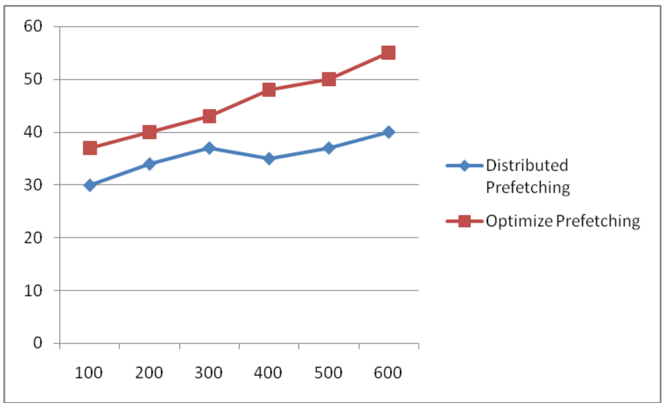
A. Algorithm

```
Segment **Select ()** //Find the next segment to prefetch
{
 Find the segments Si that didn't exist in buffermap//count =0
 Return S; //S is the desired segment;
}

Segment **Select1()**
{
 Find the segment Si that did exist in buffermap with highest count
 Return S;
}

Void **Prefetch ()** // The function to do prefetching
{
 While (prefetching set is not empty)
 {
  segment S = select ();
  else if(S ==null)
  segment S= select1();
  else if(S==null && bandwidth_is_available)
  Download anchor;
  Broadcast(S); //Broadcast the prefetching of segment
  If (segment S is cached by a peer in same session)
  //situation where same segment is also requested by some other peer
  {
   Download segment S;
   Remove the segment S from prefetching list;
  }
  else if (segment S is located on a remote peer P)
  {
    Connect with the peer P;
    Download Segment S;
    Remove segment S from the prefetching set;
  }
  else // when timeout expires
  {
    Send the segment request to server;
    Connect with server;
    Download segment S;
    Remove segment S from the prefetching set;
  }
 }
}
```

4. PERIONFORMANCE EVALUATION

We implemented an event-driven simulator coded in C++ to conduct a series of simulations in this section. For the end-to-end latency setup, we employ real-world node-to-node latency matrix (2500*2500) measured on Internet [8].We do not consider the queuing management in the routers. The default streaming rate is set to 500kbps.The upload and Download bandwidth of peers is set to 500kbps.

Fig. 3 shows the comparison of percentage of VCR request satisfied by segments prefetched for Optimize and Distributed prefetching [5]. Optimize prefetching focus on downloading segment which is near to its play head position and rarest in the session or have highest popularity (highest count). On the other hand, Distributed prefetching, prefetch segment according to user behavior, while ignoring segments rarity. In Fig. 3 it is observed that our scheme has higher percentage of VCR request satisfied by segments prefetched than Distributed prefetching.

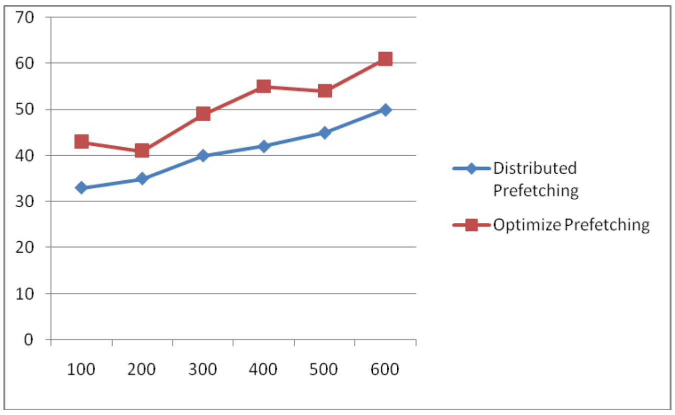


X-axis=Number of peers,
Y-axis= VCR request satisfied by segments prefetched(%)

Fig 3 : Comparison of percentage of VCR request satisfied by segments prefetched

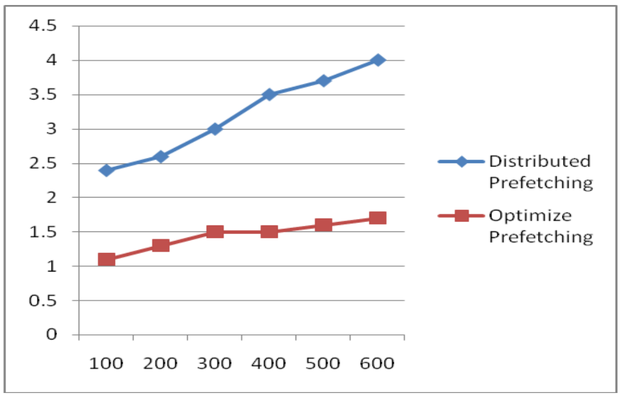
Fig. 4 shows comparison of the ratio of the amount of played data to the amount of download data for complete session, of Distributed prefetching and Optimize prefetching. In Fig. 4 it is observed that Optimize prefetching have higher ratio of the amount of played data to the amount of download data than Distributed prefetching.

Fig. 5 shows comparison of network traffic due to Distributed prefetching and Optimize prefetching. We considered gossip message as network traffic. In Fig. 5 it is observed that distributed prefetching had high network traffic than Optimize prefetching.



X-axis=Number of peers,
Y-axis= ratio of the amount of played data to the amount of download data(%)

Fig 4: Comparison of ratio of the amount of played data to the amount of download data



X-axis=Number of peers,
Y-axis= Network traffic

Fig 5: Comparison of network traffic

5. CONCLUSION

In order to provide a VCR-oriented VoD service for P2P networks, we proposed an Optimize prefetching strategy. Our strategy focuses on improving the availability of rarest content in a session. Extensive simulation experiments were conducted. Experimental results demonstrate that the proposed prefetching scheduling algorithm scheme can significantly improve the seeking performance

6. ACKNOWLEDGEMENT

We thank Douglas Schmidt for his helpful advice on peer to peer video streaming.

7. REFERENCES

[1] Cool Streaming www.coolstreaming.us/ [20-Nov-2010]

[2] PPTV www.pptv.com/ [20-Nov-2010]

[3] Y. He, Y. Liu, “VOVO: VCR-Oriented Video-on-Demand in Large-Scale P2P Networks”. In Proc of IEEE Trans. Parallel and Distributed Systems vol 20, april.2008

[4] H. Yu, D. Zheng, B. Zhao, W. Zheng, “Understanding User Behavior in Large-Scale Videoon-Demand Systems”. In Proc of EuroSys 2006.

[5] C. Zheng, G. Shen, S. Li, “Distributed Prefetching Scheme for Random Seek Support in P2P Streaming Applications”. In Proc of ACM workshop on Advances in P2P multimedia streaming 2005.

[6] B. Cheng, H. Jin, X. Liao, “Supporting VCR functions in P2P VoD services using ring assisted overlays”. In Proc of ICC 2007.

[7] C. Huang, J. Li, K.W. Ross, “Can Internet Video-on-Demand be Profitable”. In Proc of ACM SigComm 2007.

[8] “Meridian node to node latency matrix (2500*2500),” 2005, meridian project. http://www.cs.cornell.edu/People/egs/meridian/data.php

[9] B. Cheng, X. Liu, Z. Zhang, and H. Jin, “A Measurement Study of Peer-to-Peer Video-on-Demand System,” In *Proc. of IPTPS’07*, Belleue, USA, Apr. 2007.