2009 15th International Conference on Parallel and Distributed Systems

Prediction-based Prefetching to Support VCR-like Operations in Gossip-based P2P VoD Systems

Tianyin Xu, Weiwei Wang, Baoliu Ye, Wenzhong Li, Sanglu Lu, Yang Gao
State Key Lab. for Novel Software and Technology, Nanjing University, Nanjing 210093, China

***Abstract*— Supporting free VCR-like operations in P2P VoD streaming systems is challenging. The uncertainty of frequent VCR operations makes it difficult to provide high quality realtime streaming services over distributed self-organized P2P overlay networks. Recently, prefetching has emerged as a promising approach to smooth the streaming quality. However, how to efficiently and effectively prefetch suitable segments is still an open issue. In this paper, we propose PREP, a PREdiction-based Prefetching scheme to support VCR-like operations over gossip-based P2P on-demand streaming systems. By employing the reinforcement learning technique, PREP transforms users' streaming service procedure into a set of abstract states and presents an online prediction model to predict a user's VCR behavior via analyzing the large volumes of user viewing logs collected on the tracker. We further present a distributed data scheduling algorithm to proactively fetch segments according to the predicted VCR behavior. Moreover, PREP takes advantage of the inherent peer collaboration of gossip protocol to optimize the response latency. Through comprehensive simulations, we demonstrate the efficiency of PREP by gaining the accumulated hit ratio close to 75% while reducing the response latency close to 70% with only less than 15% extra stress on the server side.**

I. Introduction

Video-on-Demand (VoD) streaming service over the Internet has become immensely popular in recent years, e.g., Internet TV, online video, distance education, etc. Peer-to-Peer (P2P) computing technology has been considered as an efficient approach for providing "play-as-download" VoD services over dynamic heterogeneous network environments [1][2][4][5]. In recent years, gossip-based P2P streaming has been widely investigated and proved to be scalable and reliable to provide large-scale VoD services. In a typical gossip-based P2P VoD system, each peer randomly connects to a set of other peers with near playback offsets, called partners. By periodically exchanging data availability information with its partners, a peer actively retrieves media segments from a subset of partners holding them. Compared to tree-based P2P approaches, gossip-based P2P VoD systems could achieve higher resilience to peer churn with lower protocol overhead, and thus have been widely deployed [1][11][12].

Supporting VCR-like operations is a challenging task in the design of true P2P VoD systems, which have attracted great research interests in the last decade [7][8][9][10]. On one hand, asynchronous user requests result in caching different portions of video file among the peers, which affects the efficiency of coordinating content delivery. On the other hand, the dynamic characteristics of user interactivity make it difficult for VoD systems to freely support VCR-like operations, such as jump, pause, fast forward and rewind. Most previous VoD schemes try to support VCR-like operations via optimizing the index overlay to realize fast segment relocation [7][8][9]. In these schemes, when a VCR-like operation occurs, a peer has to first relocate the newly requested segment, download it and then playback. Such locate-and-down process may lead to longer response latency, which severely deteriorates user's viewing quality, e.g., playback freezing or even blackout. Thus, fast relocation is necessary but far less sufficient.

To reduce the response latency of interactive requests, prefetching is employed in the resource abundant P2P streaming systems. In most existing prefetching schemes, media contents will be downloaded in advance with the sequential manner to overcome bursty package loss and smoothen the playback. Nevertheless, due to the dynamic characteristics of user interactivity, traditional prefetching schemes is not effective to support VCR-like operations. Prediction-based prefetching which proactively fetches segments that are likely to be requested by future VCR-like operations by thinking over user viewing behavior is surely desired. However, most gossip-based P2P VoD systems neither carefully consider user viewing behavior nor benefit from prediction-based prefetching. Consequently, they can hardly achieve free control for VCR-like operations.

In this paper, we propose *PREP* (PREdict and Prefetch), a prediction-based prefetching scheme to support VCR-like operations which can be used in any practical gossip-based P2P VoD systems. In the design of PREP, we model user interactive behavior based on the large volumes of user viewing logs collected on the tracker via state abstraction and *reinforcement learning* (RL). Each peer gets the model from the tracker at bootstrapping and then uses this model to do prediction-based prefetching in its viewing lifetime. Specifically we made the following contributions:

1) We introduce a practical reinforcement learning method to learn user interactive behavior and gain the prediction model from the large volumes of user logs collected on the tracker. For efficient learning, a novel state abstraction method is proposed as data preprocessing. The prediction model is incremental and efficient with a hit ratio about 38% in peer-level testing.
2) We design data scheduling based on the prediction model to support VCR-like operations. The data scheduling strategy takes advantage of the inherent peer collaboration of gossip protocols to optimize the prediction performance and thus efficiently reduce the response latency.
3) The efficiency of the proposed scheme is confirmed using extensive simulations. The results show that

1521-9097/09 $26.00 © 2009 IEEE
DOI 10.1109/ICPADS.2009.60

PREP obtains an accumulated hit ratio close to 75% while reduces the response latency about 70% with only less than 15% extra stress on the server side.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the system overview. Section 4 describes in detail the RL-based prediction model learning. The data scheduling and VCR-like interactivity support are introduced in Section 5 and 6. We evaluate the system performance by simulations in Section 7. Section 8 concludes the paper with the discussion of future work.

II. Related Work

Gossip-based P2P streaming approach has been widely investigated and proved to be an effective and resilient method to support VoD service with VCR-like interactivity. There is a lot of work on system deployment and overlay optimization in gossip-based P2P VoD systems [7][5][12].

RINDY [7] designs a gossip-based ring-assisted P2P overlay to implement fast relocation of random seeks. In RINDY, each node maintains some near and remote neighbors in a set of concentric rings according to their relative distance. PPLive [5] is a real deployed large-scale P2P VoD system. It uses gossip protocol to do content discovery and peer overlay management, which cuts down on the reliance on the tracker, and makes system more robust. Kangaroo [12] resembles a gossip-based overlay in which a smart tracker is used to implement peer coordination. Kangaroo aims at providing low buffering times and high swarming throughput under user VCR-like operations.

However, the above schemes do not consider user viewing behavior nor benefit from prediction-based prefetching. Thus, VCR-like operations may cause long response time. Some previous work has already focused on modeling and utilizing the user behavior in VoD service [13][25][10].

Huang and Hsu [13] propose a user-aware prefetching mechanism which adopts simple Apriori algorithm to mine association rules from user viewing logs. Prefetching is directed by the association rules. This mechanism is designed for centralized web proxies and the simple Apriori algorithm is rather inefficient. Zheng et al. [25] propose a hierarchical prefetching scheme based on the optimal quantization theory. The scheme minimizes seeking distortion while gets high utilization ratio. VOVO [10] serves asynchronous peers with the combination of batching and patching. It leverage an on-line association rule mining to find the frequent patterns of user VCR-like interactivity, based on which a hybrid prefetching scheme is performed.

In this paper, we design a prediction-based prefetching scheme for practical gossip-based P2P VoD systems, which supports user interactivity in a proactive manner.

III. System Overview

The peer using PREP in gossip-based P2P VoD systems mainly consists of three components: 1) P2P overlay management, 2) Data scheduling algorithm, and 3) RL-based prediction model. Fig. 1 shows the system architecture of a PREP peer.

A typical P2P VoD system usually contains two overlay networks, i.e., *index overlay* for locating peers with expected

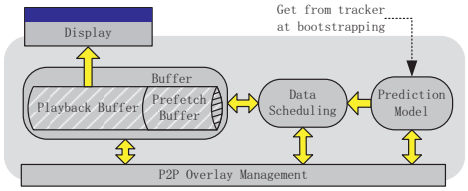

Fig. 1. The generic system architecture of a PREP peer.

segments and *data overlay* for coordinating content delivery. Since PREP is designed on the purpose of competent for any gossip-based system, we do not specify the index overlay assuming that a user's interactive requests can be responded within a locate-and-download period. For data overlay, we adopt the *general gossip model* in which each peer maintains a *partner list* containing M partner peers with close playback positions and overlapped playback buffers. These M partners are connected with the local peer by TCP connections and the periodical data exchange is performed, which forms an unstructured gossip overlay. If a partner is found to be failed or supplied little data, it will be replaced by a more healthy peer through overhearing the routing messages passing by.

The data scheduling algorithm works on the unstructured gossip overlay. It schedules the retrieval of data segments from connected partners according to the periodical data availability exchanging. PREP scheduling algorithm retrieves two kinds of data segments: the urgent segments into *playback buffer* for continuous normal playback and the predicted segments into *prefetch buffer* for supporting upcoming user VCR-like interactivity.

When joining the P2P VoD system, a PREP peer gets the prediction model from the tracker at bootstrapping. Then the peer utilizes this model to predict segments that are likely to be requested by future VCR-like operations, and triggers prefetching. To obtain the prediction model, we employ RL in PREP. RL is widely used in decision making problems in which an agent learns behavior through trail-and-error interactions with the environment [17][18][19][20]. Moreover, RL works incrementally. In this scenario, we use RL to learn user interactive behavior and build the prediction model that decides which segments should be prefetched.

In most existing gossip-based P2P VoD systems, trackers are used to keep track of user viewing records and help peers to initialize partner lists at bootstrapping [5][11][12]. Thus, a tracker maintains large volumes of user viewing logs based on which prediction model can be built off-line. With the rise of user population, the prediction model would incrementally become accurate to match real user viewing habit.

In media streaming systems, a video stream is divided into segments of uniform length and each segment is equivalent to one unit of playback time [9][14]. Both the playback buffer and the prefetch buffer are maintained as a size-limited sliding window to cache the most recently received segments, which could be supplied to other subsequent peers.

IV. Predict User Interactive Behavior

In this section, we introduce the design of our RL-based prediction model. We first present some useful characteristics

TABLE I
THE USER VIEWING RECORD (UVR) FORMAT

User ID	Video ID	Start pos.	Duration	Jump Pos.

and analysis of user interactive behavior in VoD systems which inspire our design. Then we describe the format of user logs collected by the tracker. Finally, data preprocessing as well as the RL-based learning method is elaborated.

A. Observation of User Behavior

The design of the prediction model in PREP is based on the following observations/conclusions investigated in many measurement studies [4][11][21-25].

1) VoD users often view only a portion of the full video stream, in a departure from the classic start-to-finish playback style. This phenomenon is more prominent for long videos (e.g., longer than 30 minutes [4]) such as movies and sports videos. Thus, peers in P2P VoD systems are highly dynamic and much difficult to collaborate with unstable connections.
2) There exist user access patterns to different period of a video stream. Some popular periods (e.g., hot scenes in movies, goals or penalties in sports videos) always attract far more user accesses than other non-popular periods. Such popularity is often described by skewed distributions such as Zipf-like distribution [24], and log-normal distribution [22].
3) A user's seeking position is not always accurate as expectation. Although new VCR-like features like *bookmark* [22] are proposed to help reduce browsing-like behavior, it is still hard for users to jump accurately. As a result, some existing VoD systems provide near segments (e.g., *anchors* [11], beginning segments [21]) instead of the accurate jump segment to satisfy user interactive requests.

B. User Viewing Log

Since the most basic user activity is the continuous viewing of a stretch of a video, a PREP peer maintains such basic activity in a *user viewing record* (UVR) in playback time. The important parts of an UVR format is shown in Table I. In most cases, as soon as user finishes recording one UVR, a new UVR is initialized to record the next viewing duration. A sequence of UVRs forms a *user viewing log* which represents a complete user viewing history. For example, $\{(U1, V1, 1, 6, 73), (U1, V1, 73, 3, 4), (U1, V1, 4, 3, \text{End})\}$ depicts a viewing history as follows: a user $U1$ first views the video $V1$ from the 1st second for 6 seconds, then seeks forward to the 73-th second and views until the 75-th second, and finally seeks backward to the 4-th second, re-views for 3 seconds and finishes viewing. User viewing logs can also be represented in *sequence format* as $\{s_0, s_1, \cdots, s_i, \cdots, s_{n-1}\}$, where s_i denotes that the user has viewed the s_i-th second of the video. In this example, the corresponding sequence format is $\{1, 2, 3, 4, 5, 6, 73, 74, 75, 4, 5, 6\}$. Sequence format is used for mining patterns in the preprocessing stage.

C. Data Preprocessing

Typical video stream on the Internet such as movies and sports videos takes more than 1.5 hours (5400 seconds/segments) long. If simply use "segment" as the unit to do learning, it would generate too many fine grained intermediates that brings difficulties for learning the prediction model. As a result, data preprocessing extracts the strongly associated segments into *abstract states* by distilling large volumes of user logs on the tracker and maps the raw user viewing logs into *state transitions*.

1) States Abstraction: A simple method to get abstract state is directly dividing the entire video stream into periods with equally length. However, such method would not work well as it does not take user access patterns into consideration. Take a sports video as an example, if most users view segments from s_1 to s_n continuously as a goal happened in that period, we should try to aggregate the continuous segments $\langle s_1, s_2 \cdots s_n \rangle$ as a state or some continuous states.

From this point of view, the tracker in PREP employs the frequent sequential pattern mining algorithm *PrefixSpan* [6] to mine frequent patterns from the large volumes of user viewing logs in sequence format. We modify the PrefixSpan implementation [3] by adding the pruning sub-routine, as we aim at finding the continuous sequential patterns. For example, original PrefixSpan algorithm will find patterns like $\langle 1, 2, 4, 5, 6, 7, 10 \rangle$ which are not allowed in our result. We need patterns like $\langle 4, 5, 6, 7 \rangle$ which are not only sequential but also continuous. Moreover, we do such mining process using different *support values* (support value indicates the occurring frequency of a pattern) to let the result set covers the entire video file.

As the patterns found are largely overlapped, we need to split the patterns into intervals of different length which are not overlapped. The splitting algorithm scans over the sequential patterns and divides them into intervals without overlapping. We then divide the contiguous intervals into states with fine granularity in order to fit the prefetch buffer. A too large interval makes no sense for prefetching because the prefetch buffer is size-limited. Here, we set two tunable parameters *min-state-len* and *max-state-len* which are the minimum and maximum state length allowed considering the peer buffer size. The *min-state-len* avoids splitting an isolated segment as a state while the *max-state-len* is set as the prefetch buffer size in our experiment. Thus, the whole video file can be represented by the combination of states.

2) Map Raw Logs into State Transitions: With the abstract states generated from the above step, we can easily map raw user viewing logs into state transitions. The mapped state transition is in the form of $< s, s' >$, where s is the state the current playback position is in, while s' means the next state the viewer will jump into after the current state due to VCR-like operations. For a single inter-seek duration, several state transitions could be generated as the duration may be larger than the value of *max-state-len*.

D. RL-Based Prediction Model

With the abstract states and user logs in state transition style, we employ reinforcement learning to build a prediction model to predict user behavior, i.e., the seek action.

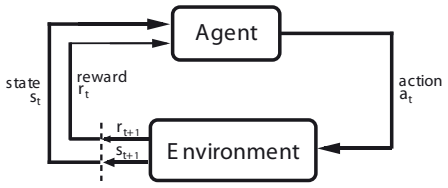

Fig. 2. The agent-environment interaction in reinforcement learning.

1) Reinforcement Learning: Reinforcement learning (RL) is learning what to do – how to map situations to actions, so as to maximize a numerical reward signal [18]. A RL *agent* learns knowledge from interaction with the *environment*, as shown in Fig. 2. To put it simply, the agent jumps from a state to another state and receives feedbacks (*rewards*) from the environment which indicates whether the jump (*action*) is good or not. From the rewards, the agent tries to learn a policy which guides the agent to gain as more rewards as possible. We normally describe an RL problem using *Markov decision processes* (MDPs). A MDP is a four tuple $< S, A, T, R >$ [18] consisting of a finite state space S, a finite action space A, a transition function $T : S \times A \times S \to \mathcal{R}$, and a reward function $R : S \times A \to \mathcal{R}$. From a given state $s \in S$, a given action $a \in A(s)$ produces an expected reward of $R(s, a)$ and transitions to another state $s' \in S$ with probability $T(s, a, s')$. To earn as more rewards as possible, we need to learn an optimal policy $\pi(s)$ which maps a state s to an action a. Sate-action value function $Q(s, a)$ or state value function $V(s)$ are often used to get such a policy. $Q(s, a)$ indicates how good the action a is in state s while $V(s)$ indicates how good the state s is among all the states. Once we get $Q(s, a)$, it is very easy to map it to a policy, $\pi(s) = \arg\max_a Q(s, a)$. With $V(s)$, we just choose the action a which brings the agent to a next state $s' \in S(s)$ with a maximum state value $V(s')$, where $S(s)$ represents the state set the agent can reach from state s.

2) Model Building: We apply Q-learning which is a classic method in RL to learn the prediction model from state transitions generated in preprocessing. Q-learning is an off-policy temporal difference control algorithm [18] to learn *state-action* values. The simplest form, i.e., one-step Q-learning, is defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \times [r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where s_t is the current state of the learning agent, a_t is the action selected by the agent. $Q(s_t, a_t)$ represents how good the action a_t is in state s_t. α is learning rate. r_{t+1} is the reward feedback from the environment. γ is a discount parameter. As previously stated, an optimal policy can be generated from the learnt optimal $Q(s, a)$ function.

Having the state transitions ready, we propose our prediction model learning algorithm (See Algorithm 1) by modifying the classic Q-learning algorithm.

The action in our algorithm is just the next state s' for each state s in a state transition and the reward is always 1.0. Each state transition in training dataset is used to update the state-action value $Q(s, a)$ repeatedly. Multiple runs of the above process is need to be done on the same training data to build a prediction model and cross-validation is required to avoid over-fitting. In certain sense, $Q(s, a)$ actually represents the preferences of users' next seeking destinations when in current state s.

Algorithm 1 Prediction Model Learning Algorithm

1: Initialize $Q(s, a)$ to zero;
2: **repeat** (for each transition)
3: $s = current$, $s' = next$;
4: Set action $a = s'$;
5: Get reward r;
6: $Q(s, a) = Q(s, a) + \alpha \times [r + \gamma \times \max_{a'} Q(s', a') - Q(s, a)]$;
7: **until** Q is stable or loop limit reached.

After the training process, a model is ready and can be used to do prediction or select action. An action here means the jump position the model predicts, that is the action a the model selects in state s according to $Q(s, a)$. We choose *roulette wheel selection* [18] in our application after comparing it to greedy action [18] and Boltzmann selection [18], another two action selection methods popularly used in RL. Roulette wheel selection does selection according to the selection probability generated from state-action values. For example, if there are four actions $A = \{a1, a2, a3, a4\}$ in state s, the selection probability of each action is calculated in this way $p(a|s) = Q(s, a)/\sum_{a \in A} Q(s, a)$.

Notice that our learning method is incremental. When user behavior changes and new user viewing logs are collected, the tracker does not need to abandon the old model but uses Algorithm 1 with the old model on the newly gathered data. The learnt prediction model will adapt to the change of user preferences. In this way, our model is scalable and keeps pace with times easily.

V. Data Scheduling

PREP peers use *buffer map* (BM) to represent the availability of segments in both playback buffer and prefetch buffer. A peer periodically exchanges BM with its partners, and then schedules which segment to be fetched/prefetched from which partner. An exchange period is called a *scheduling period*.

PREP adopts a two-stage scheduling strategy combining *fetching* and *prefetching* as follows. For every scheduling period, a peer first utilizes the download bandwidth to fetch *urgent segments* into playback buffer for guaranteeing the continuity of normal playback. If there still residual download bandwidth left, the peer predicts segments that are likely to be requested in upcoming VCR-like operations and tries to prefetch them into prefetch buffer, so as to reduce response latency. The urgent segments refers to the segments whose playback deadline is to expire immediately. In PREP, we use the urgent line mechanism [16] in playback buffer and set the urgent ratio α as 0.1, i.e., we have

$$id_{urgent} = \{\ id_x \mid id_{play} < id_x \leq id_{play} + 10\ \}$$

where id_{urgent} is the ID of urgent segments while id_{play} denotes the ID of the current playing segment. Notice that

the aggregate bandwidth usually exceeds the current user demand bandwidth in all within a significant period [4]. Thus, there is typically surplus upload capacity beyond what is needed to satisfy the normal playback demand. In this case, a PREP peer uses its RL-based prediction model to predict the user's next seeking state, denoted as $\mathcal{S}$ and tries to prefetch segments in state $\mathcal{S}$ that are not contained in the local buffer from its partner according to their BMs. Here, we use a greedy rarest-first strategy to get the rarest segments as early as possible[1]. Segment i's rarity is considered as the maximum of the probability that it will remain in each partner's buffer, which we think is more reasonable than the traditional computation $rarity_i = 1/n'$, where n' is the number of peers that can supply segment i. The rarity of a segment in PREP is calculated through the following equation.

$$rarity_i = max\{\mathcal{R}_{i_1}, \mathcal{R}_{i_2}, \cdots, \mathcal{R}_{i_n}\}$$

$$\mathcal{R}_{i_x} = \begin{cases} 0 & i \text{ is not in peer } x\text{'s buffer} \\ p_l^{(i,x)}/B_l^{(x)} & i \text{ is in peer } x\text{'s playback buffer} \\ p_r^{(i,x)}/B_r^{(x)} & i \text{ is in peer } x\text{'s prefetch buffer} \end{cases}$$

where n is the number of partners of a peer, $p_l^{(i,x)}$ and $p_r^{(i,x)}$ denotes the segment i's position in the x-th partner's playback buffer and prefetch buffer respectively, and $B_l^{(x)}$ and $B_r^{(x)}$ denotes the size of the x-th partner's playback buffer and prefetch buffer. Both the playback buffer and the prefetch buffer use the LRU policy as the buffer replacement strategy. If no segment in state $\mathcal{S}$ is found in all partners, the peer uses its residual bandwidth to pull the predicted segments from the media server.

VI. Supporting VCR-like Interactivity

Typical VCR-like operations include seek, pause, fast forward and rewind. As reported in [21][23], fast forward and rewind occupy little (about 1%) in the workload while the majority of user interactions are seeks (48%) and pauses (51%). Most VCR-like operations can be implemented by the jump process: the combination of leaving the current playback position and then requesting a new position. A seek or pause jumps only once, and a fast forward or rewind generally consists of a series of jump process [8]. Thus, we focus on the jump process in this section.

Without prediction-based prefetching, when a VCR-like operation emerges on a peer, the peer should first locate the requested segment and then download it for playback. This *locate-and-download* process leads to user playback freezing with long response latency (response latency refers to the period starting from the instant that a peer request a jump command to the instant that the peer resumes playback). In this case,

$$Lat = T_{loc} + T_{conn} + T_{down}$$

where T_{loc} is the locating time, T_{conn} is the time period for building streaming connection and T_{down} denotes to the time period for downloading the requested segments.

[1]The problem that how to choose a proper supplier for every segment so that the number of missing segments can be the minimal is NP-hard (known as the parallel machine scheduling problem [15]).

PREP can effectively reduce the response latency through predicting and prefetching. A PREP peer handles a jump request according to the following cases:

1) If the requested segment is already prefetched on the current peer, the peer directly plays out the segment without any sojourn time. In this case,
$$Lat_1 = 0$$
2) If the requested segment is not cached on the current peer but cached on its partners' buffer, the peer downloads the segment from the partner and resumes playback. In this case,
$$Lat_2 = T_{down}$$
3) If the requested segment is neither cached on the local buffer nor cached by the partners, the peer has to launch the locate-and-download process. In this cases,
$$Lat_3 = T_{loc} + T_{conn} + T_{down}$$

Suppose the occurring possibility of the three cases above is p_1, p_2 and p_3 respectively ($p_1 + p_2 + p_3 = 1$). Then the expectation of response latency is,

$$E[Lat] = p_1 \times E[Lat_1] + p_2 \times E[Lat_2] + p_3 \times E[Lat_3]$$

It is clear that $E[Lat_1] < E[Lat_2] < E[Lat_3]$. Thus, higher p_1 and p_2 leads to lower $E[Lat]$. Notice that p_1, p_2 and p_3 are decided by the prefetching scheme. Generally speaking, the accuracy the prediction model can achieve decides p_1, while the peer collaboration optimized by the data scheduling decides p_2. We show how PREP reduces response latency in the next section.

Based on the observation of the inaccuracy property of user seeking behavior, if a requested segment cannot be satisfied in time, PREP provides the requester with the segments in the same state. As the strong association in one abstract state (states are frequent sequential patterns), we think the strongly associated segments can also be satisfied by the requester. Specially, when a PREP peer uses segments in state $\mathcal{S}$ to handle VCR-like interactivity, the peer computes the priority of available segments in $\mathcal{S}$. A segment i's priority is defined as the *distance* between i's ID and the ID of the requested segment, i.e.,

$$priority_i = dist_i = |id_i - id_{req}|$$

Sorting available segments according to their priority, the peer use a greedy strategy trying to pull high-priority segments close to the requested position, which seems more likely to be satisfied by the requester.

VII. Performance Evaluation

In this section, we evaluate the performance of PREP via simulations, and compare it with other solutions.

A. User Log Generation

In the experiment, we generated a synthetic dataset of user viewing logs according to the measurements and statistics in [22]. The chosen video is a 8200-second sports video which is the Argentina vs. Serbia and Montenegro football match in World Cup 2006. This match has about 10 manually marked hot scenes, either of which is a goals or a kick-off. In

[22], segment popularity, session length as well as inter-seek duration follows some probability distributions (see Table II).

TABLE II
METRICS WITH THEIR CORRESPONDING DISTRIBUTION

Metric	Distribution	R-Square
Segment popularity	Log-normal, μ=0.016, σ=1.35	0.0941
Session length	Log-normal, μ=4.835, σ=1.704	0.127
Inter-seek duration	Log-normal, μ=1.4796, σ=2.2893	0.0358

For the generation, we modified the GISMO streaming generator [27] to be able to produce 4000 user viewing logs in UVR format. We set most parameters of GISMO generator to the values in Table II. Moreover, we modified the jump subroutine in GISMO using a log-normal distribution to let users trend to jump around hot scenes. Fig. 3 shows the normalized accumulated playback frequency of segments in the video. We believe our synthetic dataset can reflect the user viewing behavior in reality.

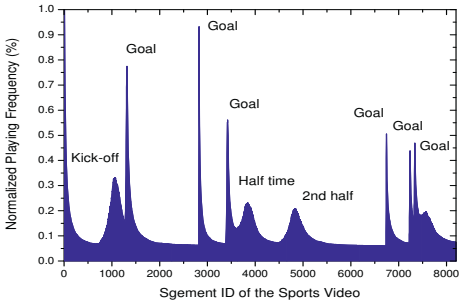

Fig. 3. Accumulated playback frequency.

B. Simulation Methodology

The simulation is built on top of a topology of 4000 peer nodes based on the transit-stub model generated by GT-ITM [28]. The streaming rate is $S = 256$ Kbps (most video stream over the Internet today is encoded in the 200-400 Kbps range [4]). The length of the video stream, denoted as L, is 8200 seconds. The default size of the playback buffer and prefetch buffer, denoted as B_l and B_r, is 25Mbytes and 5Mbytes respectively, i.e., each peer can cache 100 recent segments and 20 predicted segments in total. The arrival of peers follows the Poisson Process with $\lambda = 5.4$ according to the arrival rate in [24]. After joining the system, a peer is allocated a viewing history in the UVR format. When finishing playback, the user immediately disconnects and leaves the system. We set the download bandwidth of each peer be randomly distributed in $[1.5S, 5S]$ while the upload bandwidth be randomly distributed in $[5S, 10S]$. For comparison, we implement random prefetching scheme using the same settings as PREP for fair comparison.

C. Simulation Results

1) **Performance of the Prediction Model:** We evaluate the performance of the RL-based prediction model in terms of hit ratio. We first apply our learnt model to do prediction on the state transitions in the form of $< s, s' >$ generated by raw 4000 user viewing logs. We split the whole user viewing logs to training set and test set with a 1:9 split. The total state transitions include 125215 entries. Fig. 4 is the prediction error curve on the test set using cross validation. The x-coordinate is the transition pair in the form of $< s, s' >$, and the y-coordinate is the prediction error between the real jumping state and the predicted one. From Fig. 4, we can find that most points are concentrated near the line $y = 0$, which indicates that the predicted state is equal or very close to the real jumping state in most cases.

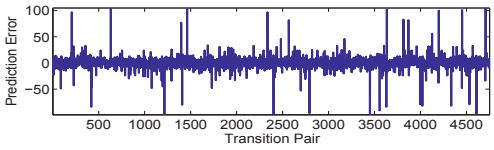

Fig. 4. Prediction error between predicted state and real seeking state.

Then we evaluate the hit ratio on peer level without considering the collaboration among partners. We assume all the segments are available for all the peers. Each PREP peer just predicts the next seeking state and sequentially prefetches segments in the predicted state into its prefetch buffer according to its download bandwidth. This scenario can be considered as the client-server model in a static environment where each peer pulls segments from the media server. The result in Fig. 5 shows that the peer-level hit ratio trends to be stable when the user population exceeds 1000. And the stable hit ratio is about 38%.

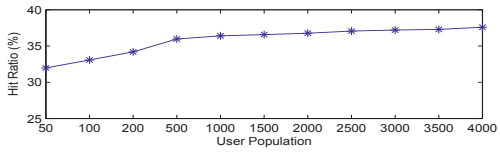

Fig. 5. Hit ratio in peer-level testing.

2) **Performance in Gossip-based P2P VoD Systems:** Due to limit of the prefetch buffer size, a peer can not prefetch all the predicted segments into its buffer. This could be compensated by peer collaboration through gossip protocol. Moreover, the diversity of predicted state through RL-based prediction strengthens the effect of peer collaboration. We evaluate the performance of PREP in terms of accumulated hit ratio in gossip-based P2P VoD systems. We let 4000 peers join the system with the arrival rate $\lambda = 5.4$ and record each peer's hit ratio. We differentiate the hit in the local buffer, denoted as LHR and the hit in the partners' buffer, denoted as PHR. Fig. 6 shows the accumulated hit ratio (AHR) of each peer, where AHR = LHR + PHR. We can see that the accumulated hit ratio of most peers are about 75%. Among them, the LHR of most peer is about 35%, which is close to the peer-level prediction performance in static environment.

We compare PREP with the system using random prefetching scheme. In PREP, we evaluate two kind of action selection methods, the roulette wheel selection and the Boltzmann selection with $tau = 5$. The result is shown in Fig. 7.

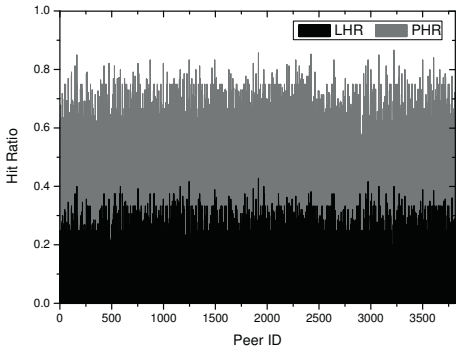

Fig. 6. Accumulated hit ratio through peer collaboration.

We can see that PREP significantly outperforms the random prefetching scheme in terms of AHR. According to Fig. 7, when user population exceeds 4000, the AHR is about 75% for roulette wheel selection and 55% for Boltzmann selection, while the random prefetching only obtains about 6% ARH. Within PREP, the roulette wheel selection seems more effective than the Boltzmann in this scenario. The Boltzmann selection would diverse the density of predicted states, which may cause bad impact on prediction performance. As a result, we use the roulette wheel selection as the default action selection method in this scenario.

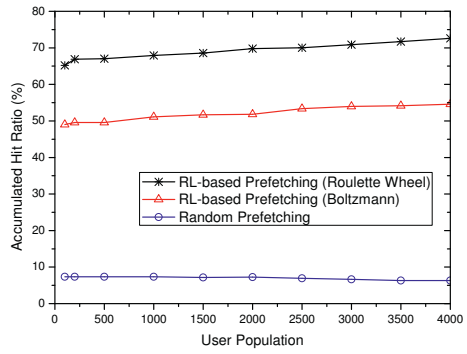

Fig. 7. Accumulated hit ratio comparison.

In addition, we discuss the accumulated hit ratio for different prefetch buffer size B_r. Intuitively, larger B_r implies more predicted segments can be held concurrently, which can potentially increase the possibility of hit events. On the other hand, large prefetch buffer imposes high requirements for client-side peers. Fig. 8 shows the accumulated hit ratio for varying user population with different B_r = 1.25Mbytes, 2.5Mbytes, 5Mbytes, 7.5Mbytes and 12.5Mbytes respectively. When B_r is small, increasing B_r effectively improves hit ratio. For example, when B_r increases from 1.25Mbytes to 5Mbytes, the ARH increases 10.61% for 4000 user scale with LHR 13.52%. However, while B_r exceeds 20Mbytes, the improvement of hit ratio is not obvious. The reason is explained as follows. A smaller B_r leads to low coverage of predicted segments in peers. At this time, increasing B_r effectively increases the prediction accuracy. When the coverage of the predicted segments is dense enough, increasing B_r is less meaningful.

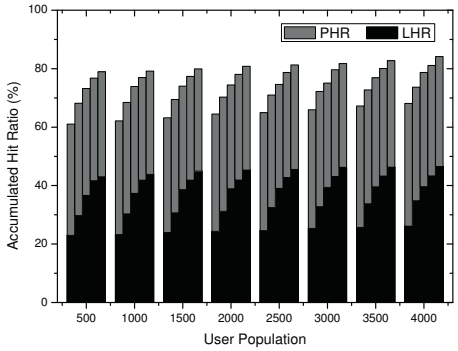

Fig. 8. Accumulated hit ratio for different prefetch buffer size.

3) **Response Latency:** Low response latency means that the VoD system responds quickly to user VCR-like commands, which leads to good user experience. We set the latency of one overlay hop from $50ms$ to $500ms$ according to the physical distance in the topology and we omit the time period for streaming connection. We trace the response latency of PREP compared with the random prefetching scheme and without prefetching. Fig. 10 shows average response latency of user VCR-like request. We can see that the average response latency of PREP decreases about 70%, as in most cases the requested segments by VCR-like operations can be found in the local prefetch buffer or partners' buffers, which eliminates the locate-and-download process. However, the response speed of random scheme improves little ascribing the poor prediction performance.

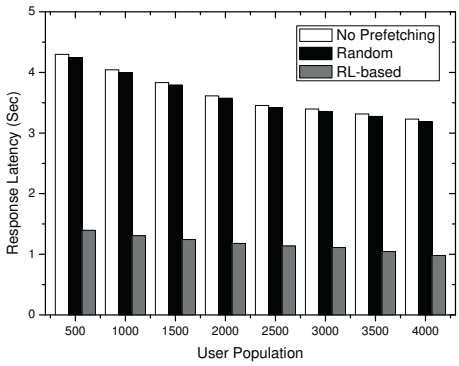

Fig. 9. Response latency comparison.

4) **Server Stress:** After predicting a next jumping state, a PREP peer first tries to prefetch the segments in that state from its partners. If no partner contains segments in the state, the peer pulls these segments from the media server, which causes extra server stress. Fig. 10 shows the server stress measured by the number of concurrent streams with different prefetching schemes. We can see that PREP increases 15% server stress compared with no prefetching when the user population exceeds 4000. This server stress increment is

much less than the server stress of random scheme, which imposes about 50% extra server stress. According to the RL-based prediction model, peers viewing similar video contents would get similar prediction results. So it is very likely for a PREP peer to get predicted segments from its partners. However, the system using random prefetching scheme is not so fortunate. The randomness of the prefetched result affects the efficiency of peer collaboration and thus leads to much higher server stress than PREP.

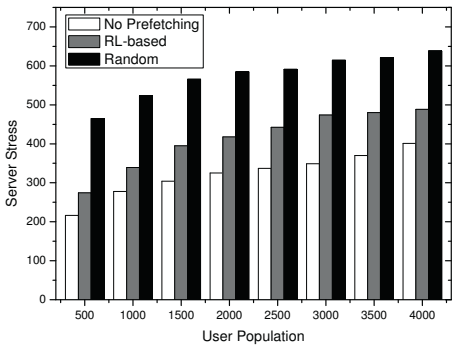

Fig. 10. Server stress comparison.

VIII. Conclusions and Future Work

In this paper, we propose PREP, a prediction-based prefetching scheme to support VCR-like operations designed for large-scale gossip-based P2P VoD systems. We use reinforcement learning technique to learn prediction model from the large volumes of user viewing logs collected on the tracker. The prediction model predicts the segments that are likely to be requested by future VCR-like operations with high hit ratio while can adapt to newly collected user logs incrementally. Through data scheduling, PREP takes advantage of the inherent peer collaboration of gossip protocols to optimize prediction performance and thus reduce response latency with only low extra sever stress. The efficiency of the proposed scheme is confirmed by extensive simulations.

We plan to improve our work in the following aspects. First, much improvements could be done to further increase the prediction accuracy. Second, we consider using model transfer or transfer learning [26] techniques to address the multiple trackers with different user logs in gossip-based P2P VoD systems, which can make PREP more practical. Finally, the QoS differentiation should also be considered due to the extra server stress caused by prefetching, especially in more heterogeneous pervasive environments.

Acknowledgement

We thank Yuan He of Hong Kong University of Science and Technology for his helpful advice on synthetic data generation.

This work is partially supported by the National High-Tech Research and Development Program of China (863) under Grant No. 2006AA01Z199; the National Natural Science Foundation of China under Grant No. 90718031, 60721002 and 60903025; the Natural Science Foundation of Jiangsu Province under Grant No. SBK200921645; the National Basic Research Program of China (973) under Grant No. 2009CB320705.

References

[1] "PPLive", http://www.pplive.com/.
[2] "Joost", http://www.joost.com/.
[3] "PrefixSpan: An Implementation of PrefixSpan," http://www.cb.k.u-tokyo.ac.jp/asailab/tabei/prefixspan/prefixspan.html.
[4] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-Demand be Profitable?" In *Proc. of ACM SIGCOMM'07*, Kyoto, Japan, Aug. 2007.
[5] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Liu, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," In *Proc. of ACM SIGCOMM'08*, Boston, USA, Aug. 2008.
[6] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "Mining Sequential Patterns by Pattern Growth: The PrefixSpan Approach," In *IEEE Transaction s on Knowledge and Data Engineering (TKDE)*, 16(10):1424-1440, Oct. 2004.
[7] B. Cheng, H. Jin, and X. Liao, "Supporting VCR Functions in P2P VoD Services Using Ring-Assisted Overlays," In *Proc. of IEEE ICC'07*, Glasgow, Scotland, Jun. 2007.
[8] D. Wang and J. Liu, "A Dynamic Skip List-based Overlay for On-Demand Media Streaming with VCR Interactions," In *IEEE Transaction on Parallel and Distributed Systems (TPDS)*, 19(4):503-514, Apr. 2007.
[9] T. Xu, J. Chen, W. Li, S. Lu, Y. Guo, and M. Hamdi, "Supporting VCR-like Operations in Derivative Tree-Based P2P Streaming Systems," In *Proc. of IEEE ICC'09*, Dresden, Germany, Jun. 2009.
[10] Y. He and Y. Liu, "VOVO: VCR-Oriented Video-On-Demand in Large-Scale Peer-to-Peer Networks," In *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 20(4):528-539, 2009.
[11] B. Cheng, X. Liu, Z. Zhang, and H. Jin, "A Measurement Study of a Peer-to-Peer Video-on-Demand System," In *Proc. of IPTPS'07*, Belleue, USA, Apr. 2007.
[12] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Rodriguez, "Kangaroo: Video Seeking in P2P Systems," In *Proc. of IPTPS'09*, Seattle, USA, Apr. 2009.
[13] C. M. Huang and T. H. Hsu, "A User-Aware Prefetching Mechanism for Video Streaming," *World Wide Web: Internet and Web Information Systems*, Kluwer Academic Publishers, 6(4):353-374, 2003.
[14] X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," In *Proc. of IEEE INFOCOM'05*, Miami, USA, Mar. 2005.
[15] T. Cormen, C. Leiserson, and R. Rivest, "Introduction to Algorithms," *MIT Press*, Cambridge, MA, USA, 1990.
[16] Z. Li, J. Cao, and G. Chen, "ContinuStreaming: Achieving High Playback Continuity of Gossip-based Peer-to-Peer Streaming," In *Proc. of IPDPS'08*, Miami, USA, Apr. 2008.
[17] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Morgan Kaufmann Publisher, 4:237-285, 1996.
[18] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," *MIT Press*, Cambridge, MA, USA, 1998.
[19] W. D. Smart and L. P. Kaelbling, " Effective Reinforcement Learning for Mobile Robots," In *Proc. of ICRA'02*, Washington, USA, May. 2002.
[20] D. Silver, R. Sutton, and M. Müller, "Reinforcement Learning of Local Shape in the Game of Go," In *Proc. of IJCAI'07*, India, Jan. 2007.
[21] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "DISC: Dynamic Interleaved Segment Caching for Interactive Streaming," In *Proc. of IEEE ICDCS'05*, Columbus, USA, Jun. 2005.
[22] A. Brampton, A. MacQuire, I. A. Rai, N. J. P. Race, L. Mathy, and M. Fry, "Characterising User Interactivity for Sports Video-on-Demand," In *Proc. of ACM NOSSDAV'07*, Urbana-Champaign, USA, Apr. 2007.
[23] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto, "Analyzing Client Interactivity in Streaming Media," In *Proc. of ACM WWW'04*, New York, USA, May. 2004.
[24] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-Scale Video-on-Demand Systems," In *Proc. of ACM EuroSys'06*, Leuven, Belgium, Apr. 2006.
[25] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," In *Proc. of ACM P2PMMS'05*, Singapore, Nov. 2005.
[26] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," Technical Report HKUST-CS08-08, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China, Nov. 2008.
[27] S. Jin and A. Bestavros, "GISMO: A Generator of Internet Streaming Media Objects and Workloads," In *Proc. of ACM SIGMETRICS'01*, Cambridge, MA, USA, Jun. 2001.
[28] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," In *Proc. of IEEE INFOCOM'96*, San Francisco, USA, Mar. 1996.