LP-based Optimization of Storage and Retrieval for Distributed Video-on-Demand

Zhuolin Xu S.-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{fanniexu,gchan}@cse.ust.hk

***Abstract*—In a distributed large-scale video-on-demand (VoD), a content provider often deploys local servers close to their users. A movie is partitioned into k segments which the servers collaboratively store and retrieve ($k \geq 1$). A critical but challenging problem is how to minimize overall system deployment cost due to server bandwidth, server storage, and network traffic among servers. In this paper, we address this problem through jointly optimizing movie storage and retrieval in the server network.**

We first formulate the optimization problem to an integer program. To address its tractability, we propose a novel, effective and implementable heuristic. The heuristic, termed LP-SR, decomposes the problem into two computationally efficient linear programs (LPs) for segment storage and retrieval, respectively. The strength of LP-SR is that it is *asymptotically optimal* in terms of k, and k does not need to be high to achieve near optimality (around 5 to 10 in our study). Through extensive simulation study, LP-SR is shown to perform significantly the best as compared with other state-of-the-art and traditional schemes, reducing the deployment cost by a wide margin (by multiple times in many cases). It attains performance very close to the global minimum cost.

***Index Terms*—Distributed video-on-demand; optimization; segment storage and retrieval; linear programming**

I. Introduction

In order to provide cost-effective video-on-demand (VoD) service scalable to large number of users, a content provider often deploys distributed servers placed close to user pools. These servers cooperatively replicate and retrieve movies given movie popularity. Such architecture is able to greatly reduce network load and scale up the streaming and storage capacity of the network. In this paper, we consider the critical and challenging problem of minimizing the system deployment cost through optimizing movie storage and retrieval in the servers. The cost model we use is general and comprehensive, capturing server storage, server bandwidth utilization and network traffic among the servers.

A typical distributed and cooperative VoD network consists of a central server (or repository) storing all the movies and proxy servers placed close to user pools.[1] While the central server stores all the movies, the proxy servers are of possibly heterogeneous storage which may be able to replicate only a fraction of the movies. Each user has a home (or local) server to serve his request. If the request is a hit, the home server directly streams to the users. Otherwise (a miss), the home server pulls the content from a remote server (either a proxy server or the central server) to serve the request. In other words, the bandwidth of the servers[2] are used to stream not only its own home users (if any), but also remote servers requesting their contents.

The deployment cost of such a VoD network mainly consists of two major components, *server cost* due to the storage and bandwidth usage of the servers, and *network cost* due to streaming among servers to serve the misses [1], [2]. A challenging problem is hence which movies to store/replicate and where to access them in order to minimize the deployment cost.

For efficient server storage and retrieval, each movie is considered to be partitioned into k segments ($k \geq 1$). We formulate the cost-minimization problem of optimizing movie storage and retrieval. To make it tractable, we propose a novel and efficient heuristic termed LP-SR which decomposes the problem into two linear programs (LPs) for segment storage and retrieval, respectively. The salient feature of LP-SR is that it is *asymptotically optimal* in k, i.e., as k increases, its performance approaches global optimum. Furthermore, our results show that k does not need to be large (say $5 - -10$) for the system to be closely optimal (within 6.5% deviation).

Our contributions are three-folds:

- *Comprehensive consideration of system deployment cost:* We consider a realistic, general and comprehensive model on the deployment cost of VoD, which includes server bandwidth utilization, storage and network transmission cost. (Previous work in VoD seldom considers all these factors together.) We formulate the joint optimization problem in movie storage and retrieval.
- *LP-SR: Achieving asymptotic optimality for video-on-demand:* We propose LP-SR which decomposes the original problem into two linear programming (LP) problems for segment storage and retrieval, respectively. These

This work was supported, in part, by an HKUST grant (FSGRF12EG05), and the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209).

[1] In this paper, we use "client" and "user" interchangeably. We also use "movie," "video" and "content" interchangeably.

[2] In this paper, we use the term "servers" to collectively refer to the central and proxy servers.

978-1-4673-0921-9/12/$31.00 ©2012 Crown

LPs can be efficiently solved in polynomial time. LP-SR is asymptotically optimal in k, i.e., the system cost approaches the exact minimum as k increases. With LP-SR, the network is able to make the best use of limited server storage, efficiently utilize server bandwidth, and substantially save network traffic cost due to server access.

- *Extensive performance study:* We conduct extensive simulation and comparison study of LP-SR with both state-of-the-art and traditional schemes. Our results show that LP-SR achieves substantially the lowest system cost, outperforming them by a wide margin (by multiple times in many cases). The results show that many existing heuristics are still far from the optimum, and LP-SR can achieve performance very close to such optimum.

We briefly review previous work as follows. Many heuristics have been proposed to address movie storage and retrieval problem (e.g., [3]–[7]). It is often not clear how well they perform as compared with the optimum, while the proposed LP-SR is asymptotically optimal in k. In contrast with some previous algorithms based on iterations [5], [8], LP-SR is based on LP formulations and hence guarantees to converge to a solution even for a large network. The work in [9], [10] considers how to support user interactivity through efficiently searching for movie segments. While the heuristics are strong and impressive, they have not considered cost optimization issue. For the works studying the cost issue for VoD [3], [11], [12], many of them have not sufficiently considered the general case with network access cost, storage constraint *and* bandwidth utilization of the servers. Our model captures all these elements, leading to a more complete, realistic and practical formulation.

This work is organized as follows. In Section II, we formulate the joint optimization problem for VoD. We present LP-SR in Section III. In Section IV, we present illustrative simulation results on the performance of LP-SR. We conclude in Section V.

II. Problem Formulation

In this section, we present the joint cost-optimization problem of movie storage and retrieval to minimize deployment cost.

We show the important symbols used in Table I.

The overlay network is modeled as an undirected graph $G = (V, E)$, where V is the set of servers and repository and $E = V \times V$ is the set of overlay edges connecting nodes in V (the extension to directed graph is straightforward given our current formulation). Let M be the set of movies and $L^{(m)}$ be the movie length (in seconds). Let $p^{(m)}$ be the popularity of movie m, which is the probability that a user requests movie m, where $0 \leq p^{(m)} \leq 1$ and $\sum_{m \in M} p^{(m)} = 1$.

A server v has a certain storage space B_v (in seconds). Let

$$I_v^{(m)} \in \{0, 1\}, \forall v \in V, m \in M, \tag{1}$$

indicating whether server v stores movie m. Note that for the repository, $I_v^{(m)} = 1, \forall m \in M$. We obviously must have

$$\sum_{m \in M} I_v^{(m)} L^{(m)} \leq B_v, \ \forall v \in V. \tag{2}$$

Let

$$r_{uv}^{(m)} \in \{0, 1\}, \ \forall u, v \in V, m \in M, \tag{3}$$

indicating whether the requests for movie m at server v are "pulled" from server u. As the server cannot supply more than that it stores, we must have

$$r_{uv}^{(m)} \leq I_u^{(m)}, \ \forall u, v \in V, m \in M, \tag{4}$$

and, by definition, $r_{vv}^{(m)} = I_v^{(m)}$.

Each user retrieves data from the servers (including his home server), we hence must have

$$\sum_{u \in V} r_{uv}^{(m)} = 1, \forall v \in V, m \in M. \tag{5}$$

Let λ_v be the total movie request rate at server v (requests per second); the request rate for movie m at server v is hence $p^{(m)} \lambda_v$. Further let $\alpha^{(m)} L^{(m)}$ be the average holding (or viewing) time for movie m, where $\alpha^{(m)} \geq 0$. Then the average data streamed is $\alpha^{(m)} r_{uv}^{(m)} L^{(m)}$, as the actual amount of streamed data is assumed to be directly proportional to the viewing time.

Let b be the movie streaming bitrate (bits/s). Hence, the

TABLE I
Major symbols used in this paper.

Notation	Definition
V	The set of servers (repository and distributed proxy servers)
M	The set of movies
$L^{(m)}$	The movie length (in seconds)
$p^{(m)}$	Access probability of movie m
$I_v^{(m)}$	0 or 1 variable indicating whether server v stores movie m
B_v	Storage space of server v (in seconds)
$r_{uv}^{(m)}$	0 or 1 variable indicating whether whether the requests for movie m at server v are streamed from server u.
λ_v	Request arrival rate at server v (requests per second)
$\alpha_m L^{(m)}$	Average holding (viewing) time of movie m $\alpha^{(m)} \geq 0$
Γ_{uv}	Network transmission bandwidth from server u to v (bits/s)
b	Movie streaming rate (bits/s)
R_v	Uploading bandwidth of server v for streaming to remote servers (bits/s)
C_v^S	Cost for server v (per second)
C_{uv}^N	Network cost due to traffic from server u to v (per second)
C	Total deployment cost (per second)

data rate the server v "pulls" from server u for movie m is $p^{(m)}\lambda_v\alpha^{(m)}r_{uv}^{(m)}L^{(m)}b$. Therefore, the total network transmission bandwidth (bits/s) from server u to v is

$$\Gamma_{uv} = \sum_{m\in M} p^{(m)}\lambda_v\alpha^{(m)}r_{uv}^{(m)}L^{(m)}b, \forall u, v \in V, \quad (6)$$

for $u \neq v$, and, by definition, $\Gamma_{uu} = 0$.

Let C_{uv}^N be a monotonically non-decreasing piece-wise linear function for network cost due to the traffic from server u to v, i.e.,

$$C_{uv}^N = \mathbb{C}_{uv}^{\mathbb{N}}(\Gamma_{uv}), \forall u, v \in V, \quad (7)$$

with $C_{uu}^N = 0$. The total network cost C^N is hence

$$C^N = \sum_{u,v\in V} C_{uv}^N. \quad (8)$$

The bandwidth used in a server to serve the other remote servers depends on where to store and how to retrieve a movie. For any server $v \in V$, the total rate (bits/s) that it serves other servers is given by

$$R_v = \sum_{u\in V, u\neq v} \Gamma_{vu}, \forall v \in V. \quad (9)$$

The servers help each other using "cache and stream" model, i.e., a remote server streams to a user *through* his home server. Therefore, the total bandwidth of server v to serve its local users is given by $\sum_{m\in M} p^{(m)}\lambda_v\alpha^{(m)}L^{(m)}b$. This is a fixed quantity given local traffic, and hence will not be considered in our cost optimization.

Let C_v^S be the cost of operating server v, which is a monotonically non-decreasing piece-wise linear function in B_v and R_v, i.e.,

$$C_v^S = \mathbb{C}_v^{\mathbb{S}}(B_v, R_v), \forall v \in V. \quad (10)$$

In another words, the server cost is a function of its storage and streaming bandwidth. The aggregated server cost C^S is hence

$$C^S = \sum_{v\in V} C_v^S. \quad (11)$$

Therefore, the total system deployment cost C is

$$C = C^N + C^S. \quad (12)$$

We state our joint cost-optimization problem as follows:

JOSR: Joint Optimization on Movie Storage and Retrieval Problem to Minimize Deployment Cost: Given topology G, user demand $\{\lambda_v\}$, storage capacity $\{B_v\}$, movie popularity $\{p^{(m)}\}$, and cost functions $\{\mathbb{C}_{uv}^{\mathbb{N}}\}$ and $\{\mathbb{C}_v^{\mathbb{S}}\}$, we seek to minimize the total cost given by Equation (12), subject to Equations (1) to (5). The output is movie storage in each server (i.e., $\{I_v^{(m)}\}$) and movie retrieval between servers (i.e., $\{r_{uv}^{(m)}\}$).

III. LP-based Segment Storage and Retrieval

Note that JOSR is an integer program (IP), which is intractable to solve. In this section, we present our novel and efficient heuristic called LP-SR, which decomposes the optimization problem into two LPs for segment storage (LP-S) and retrieval (LP-R). LP-SR works as follows: we first relax the problem stated above to an LP which yields optimal segment storage (Section III-A). Our discretization process is asymptotically optimal (Section III-B). Given the storage, we solve the optimal segment retrieval problem by another LP (Section III-C).

A. LP-S: Relaxation to a Linear Program for Segment Storage

In order to address the tractability, we relax the constraint in Equation (1) as

$$0 \leq I_v^{(m)} \leq 1, \forall v \in V, m \in M, \quad (13)$$

and Equation (3) as

$$0 \leq r_{uv}^{(m)} \leq 1, \forall u, v \in V, m \in M. \quad (14)$$

After such relaxations, our problem becomes an LP, where $I_v^{(m)}$ refers to the fraction of movie m that server v stores, and $r_{uv}^{(m)}$ refers to the proportion of requests for movie m streamed from server u to server v for a homed user.

Note that for any arbitrary linear functions of C_{uv}^N (Equation (10)) and C_v^S (Equation (7)) the relaxed problem becomes a linear programming (LP) problem which can be solved efficiently. Due to variable relaxations ($I_v^{(m)}$ and $r_{uv}^{(m)}$ above), our LP solution is expected to obtain lower cost solution than its original IP formulation, i.e., $C^{LP^*} \leq C^{IP^*}$. We call C^{LP^*} the *super-optimum* solution which is no worse than the IP solution. We will see later in Section IV that LP-SR asymptotically approaches the super-optimum, meaning that both the IP solution and LP-SR are very close.

B. Asymptotically Optimal Segment Storage

The LP solution above is used for segment storage (and hence LP-S). We propose an asymptotically optimal segment storage algorithm here. Each movie is partitioned into k segments ($k \geq 1$). In order to sufficiently utilize the fractional solution derived from LP-S, we place some of the k segments to match as closely as possible the optimal movie storage $I_v^{(m)}$. The major issues are how many segments of a movie should be locally stored (i.e. segment space allocation) and which segments should be stored (i.e. segment placement).

1) *Segment space allocation*: The number of segments of movie m that server v stores is

$$n_v^{(m)} = I_v^{(m)}k, \forall v \in V, m \in M. \quad (15)$$

For finite k, $\{n_v^{(m)}\}$ needs to be discretized to integral values. We present below a simple discretization approach where each server tries to match the optimal LP solution as much as possible through integer rounding.

We first round *down* the result $\{n_v^{(m)}\}$ as obtained in Equation (15) to its closest integers. For each server v, it first allocate segment space according to these integers for each movie. This clearly does not violate its storage constraint (given in Equation (2)). For the residual storage

the server then allocates space in decreasing order of the unmatched portion, i.e., according to $n_v^{(m)} - \lfloor n_v^{(m)} \rfloor$, until its total storage is exhausted. It is clear from above that the new $n_v^{(m)}$ are of integral values.

Note that our segment storage *asymptotically* approaches the optimal solution as k increases. It is because the rounding effect decreases with increasing k.

2) *Segment placement*: With the knowledge of integral $n_v^{(m)}$, each server then selects its $n_v^{(m)}$ out of k segments to store.

The guiding principle of our placement algorithm is that all the segments of a movie should has similar number of replicates in the whole network. Accordingly, we use *rarest first* in segment placement. Specifically, when a server makes a segment placement, it selects the segment which is the least stored until the $n_v^{(m)}$ segment budget is fully consumed.

C. LP-R: Optimal Segment Retrieval as a Linear Program

The optimal solution of $\{r_{uv}^{(m)}\}$ given by LP-S is no longer appropriate due to our segment storage. We hence need to formulate another LP (called LP-R) to derive optimal segment retrieval given segment storage above.

Let $S = \{1, 2, ..., k\}$ be the set of segment indices of any movie. Let $I_v^{(ms)} \in \{0, 1\}$ indicates whether server v stores segment s of movie m, which has been derived the solution given in Section III-A. We further let $r_{uv}^{(ms)}$ be the probability of requesting server u from server v segment s of movie m. The segment retrieval problem can then be stated as follows:

- *Arrival rate*: A request for a movie leads to streaming of all its k segments. Therefore, the request rate for *segments* at server v, given movie request rate λ_v, is

$$\lambda'_v \quad = k\lambda_v, \quad \forall v \in V. \tag{16}$$

- *Length*: A movie is equally divided into k segments; hence we have

$$L^{(ms)} \quad = \tfrac{L^{(m)}}{k}, \quad \forall m \in M, s \in S. \tag{17}$$

- *Popularity*: The popularity of the segments of the same movie is given by

$$p^{(ms)} \quad = \tfrac{p^{(m)}}{k}, \quad \forall m \in M, s \in S. \tag{18}$$

Using the above, we can formulate optimal segment retrieval problem as an LP (LP-R), i.e.

$$\min \left(\sum_{u,v \in V} \mathbb{C}_{uv}^{\mathbb{N}}(\Gamma_{uv}) + \sum_{v \in V} \mathbb{C}_v^{\mathbb{S}}(B_v, R_v) \right)$$

subject to

$$0 \le r_{uv}^{(ms)} \le I_v^{(ms)}, \forall u, v \in V, m \in M, s \in S,$$
$$\sum_{u \in V} r_{uv}^{(ms)} = 1, \forall v \in V, m \in M, s \in S.$$

IV. Illustrative Numerical Results

In this section, we first present our simulation environment and performance metrics to study the performance of LP-SR, followed by illustrative results at steady state.

A. Setup and Performance Metrics

Movie popularity follows the Zipf distribution with skewness parameter s, i.e., the request probability of the ith movie is proportional to $1/i^s$. For our baseline parameters of $s = 0.6$ and $m = 100$ movies, the top 30% of the movies account for close to 60% (56.72%) of the traffic.

Requests arrive at each proxy server according to a Poisson process with total rate λ (req./second). The central server has no home users. The proxy servers have heterogeneous storage space and bandwidth following a Zipf distribution (independent of each other). The repository stores all the movies with a streaming capacity twice of the average streaming capacity of the proxy servers.

Unless otherwise stated, we use the default values as follows for our system parameters (the baseline case): $k = 5$; 10 proxy servers; 100 movies; average server storage is 10 movies; skewness of server storage is 0.4; average proxy server bandwidth capacity is 160 Mbits/s; skewness of server bandwidth is 0 (i.e., same bandwidth); skewness of movie popularity is 0.6; movie length is 90 minutes; average movie holding time is Movie length (i.e., $\alpha^{(m)} = 1$); movie streaming rate is 1 Mbits/s; total request rate in the network is 0.3 req./s (equally distributed to the proxies); c_{uv} between central and proxy server is 0.01 unit/s; c_{uv} between proxies is Zipf with skewness 0.6 and mean 0.005 unit/s.

In the simulation, we consider the network cost function from server u to server v to be proportional to the bandwidth between them, i.e.,

$$C_{uv}^N(\Gamma_{uv}) = c_{uv}\Gamma_{uv}, \qquad \forall u, v \in V. \tag{19}$$

where c_{uv} is some constant (by definition, $c_{vv} = 0$). The server cost is a function of its storage and its total bandwidth used to serve the remote servers, modelled as

$$C_v^S = \sigma_B B_v + C_v(R_v), \qquad \forall v \in V, \tag{20}$$

where σ_B is a constant ($\sigma_B = 0.02$ in our simulation), and $C_v(R_v)$ is a piece-wise linear function monotonically increasing in R_v. We show in Figure 1 streaming cost $C_v(R_v)$ versus R_v/U_v in our simulation, where U_v is the streaming capacity of the server and hence R_v/U_v is the bandwidth utilization of the server. The cost increases with the bandwidth utilization at the server. There are three linear segments formed by points $(0, 0)$, $(0.8, 0.125)$, $(0.93, 0.4375)$ and $(0.99, 1.925)$ (these coordinates are obtained from the queuing model $\sigma_S/(U_v - R_v)$, where σ_S is some constant). As the consumed bandwidth R_v approaches the bandwidth capacity U_v, the server cost increases sharply.

The performance metrics we are interested in are:

- *Total cost* (unit/s), which is the sum of server cost and network cost according to Equation (12). This is the total deployment cost of the network.

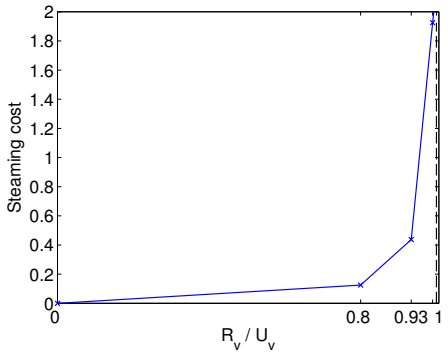

Fig. 1. Streaming cost model at a proxy server.

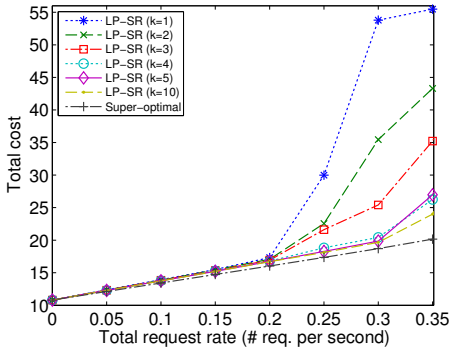

Fig. 2. Total cost versus request rate given k.

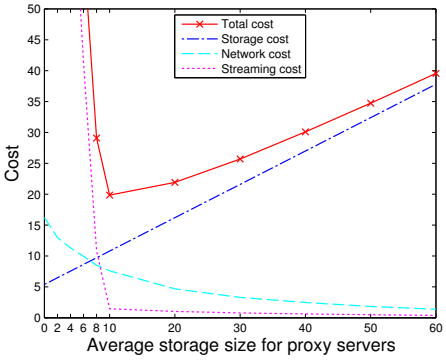

Fig. 3. Cost versus average proxy storage.

- *Server cost* (unit/s), which is the sum of its storage and streaming defined in Equations (11) and (20). We further examine the following cost components:
  - *Storage cost*, which is the total cost due to server storage; and
  - *Streaming cost*, which is the server bandwidth cost to support other servers.
- *Network cost* (unit/s), which is network transmission cost defined by Equations (8) and (19).
- *Cost of each movie* (unit/s), which is the average cost to access movie m by any user.

We compare LP-SR with the following traditional and state-of-the-art movie replication schemes:

- *Random*, where each server randomly stores movies without considering their popularity. This is a simple storage strategy.
- *MPF (Most Popular First)*, where each server stores the most popular movies. This is a greedy strategy, but does not take advantage of cooperative replication.
- *Local Greedy* [3], which divides the movies into three categories, those popular ones which all servers store (full replication), those medium popular ones which only one proxy server store (single copy), and those unpopular ones which only the repository stores (no copy). By formulating a LP problem, it seeks to minimize network cost. As Local Greedy assumes homogenous access cost, we set its access cost to be equal to the average access cost between servers in our network.

In all the comparison schemes, upon a miss request, the home server v chooses an available server u which has the requested content with a probability proportional to $1/c_{uv}$. It is a reasonable, simple and effective strategy because the server with lower access cost has higher chance to be chosen. With this probabilistic approach, a server with low access cost is not always selected so as to avoid congestion, and hence high network cost, at the server.

B. Illustrative Results

We plot in Figure 2 the total cost versus request rate given k. The total cost increases with the request rate mainly because of the increase in network load. As k increases, the network approaches the super-optimal case (given by relaxing the integer constraints on movie storage $\{I_v^m\}$ and retrieval $\{r_{uv}^m\}$). However, for humble value of k (say 5), the performance is already very close to the optimum (less than 6.5% deviation in our default setting). This shows that our network is highly efficient, with closely optimal performance even for the practical finite value of k.

We show in Figure 3 the cost components and total cost versus the average storage space for servers. The total cost falls off initially but rises up again, showing a minimum. At the beginning when the proxy servers have little storage, all the traffic concentrates on the repository, leading to high overall streaming cost. As proxy storage increases, the repository load is reduced and hence the streaming and network transmission cost. As storage further increases, storage cost becomes a major component. It is clear that LP-SR can balance the cost between storage and bandwidth and achieve its optimality by provisioning optimal network resources.

We plot in Figure 4 the total cost versus the skewness of movie popularity given different schemes. The total cost in general decreases with the popularity skewness. This is because skewed popularity means that more requests are concentrated on fewer popular movies. Consequently, there is lower miss rate, leading to lower streaming and network cost. LP-SR achieves substantially the lowest cost, even for low skewness (i.e. when the popularity is quite uniform). This shows that LP-SR makes good movie placement and retrieval decisions. Local Greedy performs better than MPF because it takes network cost into consideration. The cost of Random increases with skewness because it is popularity-blind. As a result, the popular movies, because of their copies not increasing with their popularity, suffer from high streaming and network cost.

We compare in Figure 5 the server cost for different schemes. We sort the proxy servers according to their storage in ascending order (as their streaming capacity is the same in our baseline), and the last server is the repository. It is clear that LP-SR utilizes very well the storage and bandwidth resources of proxy servers, leading to low repository streaming cost. All the other schemes suffer from high repository cost

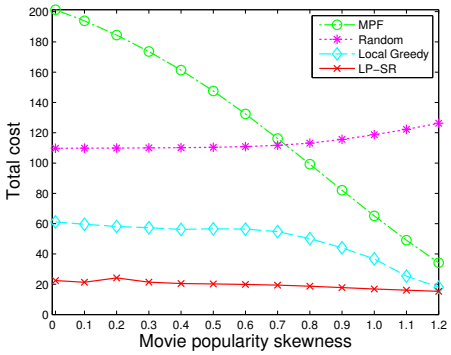

Fig. 4. Total cost versus the skewness of movie popularity given different schemes.

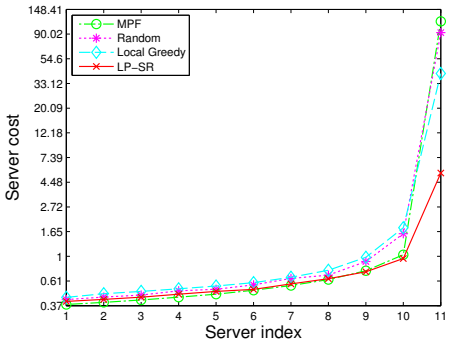

Fig. 5. Server cost distribution given different schemes.

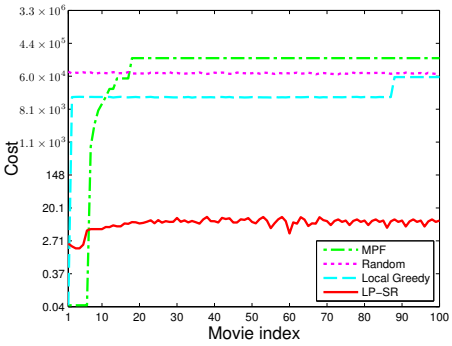

Fig. 6. Movie cost for different schemes.

(note that log scale) due to misses in the proxies. The figure shows that LP-SR has strong server cooperation to achieve near-optimal performance. As MPF only stores the most popular movies at the proxy servers, it has lower proxy server cost but much higher repository cost (due to miss traffic). In MPF, the proxies barely contribute their bandwidth and storage to help each others. Local Greedy, with network cost optimization, outperforms Random in both proxy server cost and repository cost.

We compare in Figure 6 the cost to access a movie for different schemes. The movies are sorted according to their descending popularity. The popularity-based schemes (i.e., LP-SR, Local Greedy and MPF) tend to locally store the popular movies, and hence those movies enjoy lower cost. LP-SR makes much better decision by cooperatively storing the movies. LP-SR accomplishes much better optimality of a rather uniform movie cost, with the cost of unpopular movies strikingly much lower by orders of magnitude than the other schemes. For MPF, the cost of popular movies are negligible at much sacrifice of less popular ones. Random treats each movie equally and thus has the most uniform cost distribution. The figure shows that LP-SR makes intelligent decisions on movie segment and retrieval to achieve low deployment cost.

V. Conclusion

In this work, we have studied optimal segment storage and retrieval to minimize VoD deployment cost with distributed proxy servers. The deployment cost captures the costs of server streaming, server storage and network transmission cost.

For efficient server storage and retrieval, each movie is partitioned into k segments ($k \geq 1$). We first formulate the joint problem to an integer program. To address its tractability, we propose LP-SR, a novel and efficient heuristic which decomposes the problem into two linear programs (LPs) for segment storage (LP-S) and retrieval (LP-R), respectively. In stark contrast with much of the previous work where heuristics are often proposed without knowing how they perform with respect to the optimum, our solution is *asymptotically optimal* in k, and k does not need to be large to achieve near optimality (k is around 5 achieving less than 6.5% deviation in our study).

We have conducted extensive simulation to compare its performance with other traditional and state-of-the-art schemes. The results show that our scheme substantially outperforms the other schemes by a wide margin (multiple times in many cases). LP-SR achieves very close to optimality with much lower deployment cost.

References

[1] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?" in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 133–144.

[2] S.-H. G. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 125–136, Apr. 2001.

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proceedings of IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[4] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.

[5] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimizing file availability in peer-to-peer content distribution," in *26th IEEE International Conference on Computer Communications (INFOCOM)*, May 2007, pp. 1973–1981.

[6] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. R. Rodriguez, "Is high-quality VoD feasible using P2P swarming?" in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007, pp. 903–912.

[7] P. R. R. S. Annapureddy, C. Gkantsidis and L. Massoulie, "Providing video-on-demand using peer-to-peer networks," in *Microsoft Research Technical Report, MSR-TR-2005-147*, Oct. 2005.

[8] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Proceedings of IEEE INFOCOM 2012*, Mar. 2012, pp. 2444 –2452.

[9] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications Special Issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1717–31, Dec. 2007.

[10] Y. He, G. Shen, Y. Xiong, and L. Guan, "Optimal prefetching scheme in P2P VoD applications with guided seeks," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 138–151, Jan. 2009.

[11] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving VoD server efficiency with bittorrent," in *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*. New York, NY, USA: ACM, 2007, pp. 117–126.

[12] C. Dana, D. Li, D. Harrison, and C. N. Chuah, "BASS: Bittorrent assisted streaming system for video-on-demand," in *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, Nov. 2006, pp. 1–4.