SNACS: Social Network-Aware Cloud Assistance for Online Propagated Video Sharing

Haitao Li†, Yanfang Le†, Feng Wang‡, Jiangchuan Liu†, Ke Xu*
† Simon Fraser University, Email: {haitaol, yanfangl, jcliu}@sfu.ca
‡ The University of Mississippi, Email: fwang@cs.olemiss.edu
* Tsinghua University, Email: xuke@mail.tsinghua.edu.cn

***Abstract*—The deep penetration of Online Social Networks (OSNs) has made them as major portals for video information sharing. Propagated through chains of friends, the coverage of OSN-shared videos can be much broader with stronger micro- and macro-dynamics. Given that the contents are still hosted by external Video Sharing Sites (VSSes), such distinct access patterns from OSN users have created significant new challenges to VSSes. In this paper, we present SNACS, a cost-effective social network-aware cloud assistance for video sharing. The SNACS module sits between VSSes and an OSN, and is managed by the OSN to improve its users' video access experience using both centralized cloud resources and edge servers. Given the strong dynamics of the access patterns, we are particularly interested in the content management and update strategies in the SNACS' implementation. Motivated by real world data traces, we show that conventional cache replacement can be quite inefficient in this context. We then develop optimal offline algorithms with minimized cache misses and replacements, which also motivate an online solution that makes effective use of the video sharing patterns in the OSN. Our design has been extensively evaluated and its superiority has been validated under diverse network and user configurations.**

I. Introduction

The deep penetration of Online Social Networks (OSNs), e.g., Facebook and Twitter, has made them major portals for information sharing. It is known that a significant portion of the accesses to such conventional video sharing sites (VSSes) as YouTube are now coming from OSN users. Earlier measurement based on YouTube data has found that between April 2009 and March 2010, 25% of views on YouTube come from social sharing [1]. The statistics published by YouTube showed that, as of January 2011, more than 500 tweets per minute containing a YouTube link, and over 150 years worth of YouTube video is watched by Facebook users every day; till June 2012, the numbers have increased to 700 tweets and 500 years [2]. According to comScore's latest statistics in September 2013 [3], Facebook ranked No.2 in terms of the number of viewers, and ranked No.3 in terms of the number of video views. Besides Facebook or Twitter, we have seen similar trends in other OSNs; for example, in RenRen [4], the biggest Facebook-like OSN in China and Tencent Weibo [5], a large Twitter-like OSN in China. It has been shown that more than 54 million unique RenRen users have participated in video viewing and 20 million participated in sharing, generating 12.4 million views, and 1.64 million shares every day [4].

Traditionally, VSS videos are mainly discovered through search engines, front pages, and related videos. The OSNs however offer quite different sharing mechanisms, where the video links are propagated through chains of friends. The coverage of such OSN-shared videos can be much broader with much faster propagation speeds. It also leads to more micro- and macro-dynamics in the access pattern, as a super user with a great number of friends can easily trigger a surge of accesses [5], and in the long run, a video often has a series of peaks in term of user access. Given that the video contents are still hosted by VSSes, such distinct access patterns from OSNs have created significant challenges to VSS service providers, particularly for resource provisioning.

There have been pioneering works on joint design and optimization for both VSSes and OSNs with shared information [5][6][7]. In the real market, however, VSS and OSN operators are not necessarily close collaborators, nor the VSSes are to be urgently and completely re-engineered for OSN shared videos given that the demands from traditional users remain strong. On the other hand, for OSN operators, building their own video storage and distribution services is not necessarily the best business model, either, not to mention the complexity and cost involved in joint design. Instead, we believe that, as an OSN knows best about the video sharing patterns from its users, it should provide necessary assistance for its users to access the external VSSes, which in turn will also mitigate the impact to the VSSes.

To this end, we develop SNACS (social network-aware cloud assistance for video sharing), which provides a cost-effective enhancement for video accesses from an OSN. The SNACS module sits between VSSes and an OSN, and is managed by the OSN to improve its users' experience in retrieving videos from the VSSes. It utilizes both centralized cloud resources (e.g., Amazon S3) and edge servers (e.g., Amazon CloudFront) to collectively serve video accesses from the OSN, which otherwise cannot be well served by the external VSSes. Given the strong dynamics of the user access patterns, we are particularly interested in the content management and update strategies in the SNACS' implementation. Motivated by the data traces from our real world measurement, we show that the conventional cache replacement for video objects can be quite inefficient in SNACS. We then develop an optimal offline replacement algorithm that generates minimum misses in this new context.

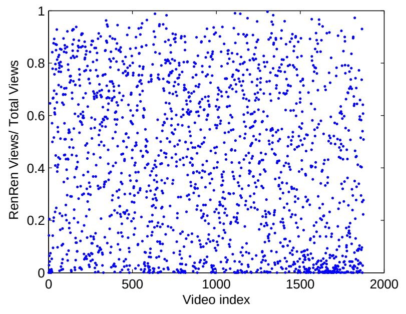


Fig. 1. Distribution of fraction of RenRen views over the total views.

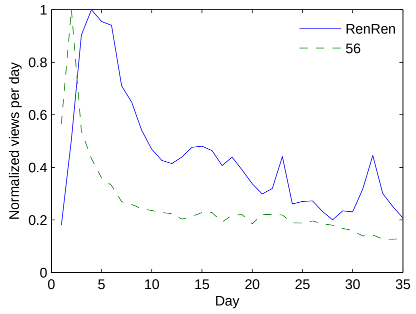


Fig. 2. Overall video popularity evolution (normalized by maximum values of daily views).

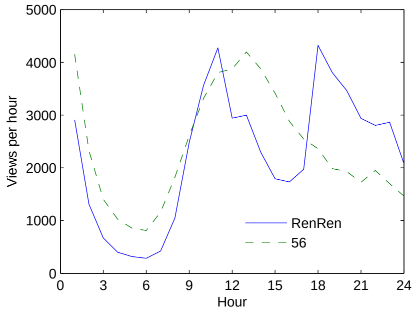


Fig. 3. Video popularity evolution of a single video in one-day period.

We further offer guidelines and extend the algorithm to minimize replacements among the solutions of the minimum misses. The optimal offline solutions not only provide a benchmark for comparison but also motivate the design of an online replacement algorithm, which makes effective use of the video sharing patterns in the OSN. Our design has been extensively evaluated and its superiority has been validated under diverse network and user configurations.

The rest of the paper is organized as follows. We present background and motivation in Section II. Section III proposes our framework and discusses major design issues. We develop optimal offline algorithms in Section IV, and an online algorithm in Section V, respectively. The results of performance evaluations are presented in Section VI. Finally, we conclude the paper in Section VII.

II. Background and Motivation

There have been significant data-driven measurement and modelling studies on the videos shared within OSNs, e.g., tracking social cascades of YouTube links over Twitter [8], video clip sharing on Twitter's Vine [9], and video popularity distribution and propagation pattern in Tencent Weibo, a Twitter-like OSN in China [5]. Our earlier works have examined video propagation patterns over RenRen, a Facebook-like OSN in China [4], which leads to the design of a synthetic traffic generator for video requests from OSNs [10].

Our work is motivated by these studies. To further understand the distinct characteristics of video request patterns from OSNs as compared with traditional video accesses and their impact to resource provisioning, we closely collaborate with 56*.com*, one of the most popular VSSes in China to analyze its server access logs. The logs record video requests within the 56.com website as well as requests from external OSNs. Our analysis of server access logs from 56.com website shows that among all the video requests, over 36% of them are from OSNs, most notably from RenRen. For individual videos, however, the ratio of requests from RenRen to the total requests varies significantly, as shown Fig. 1. The Pearson correlation coefficient[1] between video views in RenRen and the total views is 0.59, which is statistically insignificant. In other words, while statistical histories have often been used to predict the video popularity, it can hardly predict the percentage of the requests from OSNs for a newly uploaded video.

We also examine how the popularity evolves for videos shared by both 56.com and RenRen. Fig. 2 compares the overall video popularity evolutions over 5 weeks. We can see that the views from RenRen users exhibit much stronger dynamics, with more peaks when compared to the overall views in 56.com. Fig. 3 shows how the popularity evolves for a single video in a smaller time scale. We find that, in OSNs, there are super users with a large number of friends or followers, and such users, once propagating a video, can trigger a significant number of follow-up accesses. This can lead to a peak of accesses even long after the release of the videos, in which stage the accesses from traditional VSSes users have long decayed. As such, today's VSSes, even equipped with state-of-the-art prediction and resource provisioning modules, can still experience frequent under-provisioning [7].

A series of pioneering works have offered service enhancements of social video sharing by joint design and optimization for both VSSes and OSNs [7][5]. As video services are critical to social network users, OSN operators do have strong motivation to offer better service quality to their users. Yet whether they fully disclose the social information to external VSS providers remains questionable in the current market, and building their own video content services is not necessarily the best practice, either. On the other hand, the accesses from traditional VSS users remain strong (over 50%), and there is no immediate need for a VSS to re-engineer its services. Therefore, we need a new framework which works effectively without integrating OSN and VSS into a unified system. Considering that OSN has the best knowledge of its video requests and the propagation patterns, we suggest that

[1]It has been widely used for measuring the strength of linear dependence between two variables. The range is from -1 to 1, where a value greater than 0 indicates positive correlation, and less than 0 indicates negative correlation.

the OSN should take the initiative to offer assistance to its users accessing videos. In turn, it will also benefit external VSSes given a large portion of accesses from the OSN are absorbed by OSN servers.

III. SNACS Approach

Motivated by the above ideas, we propose a new framework called SNACS, shortened from social network-aware cloud assistance for video sharing. The SNACS module sits between VSSes and an OSN, and is managed by the OSN to improve its users' experience in retrieving videos from the VSSes. It utilizes cloud resource to serve video accesses within the OSNs that otherwise cannot be well served by external VSSes. Fig. 4 offers a detailed view of the work flow in SNACS. An OSN user will initially request video data from the VSS servers. According to the feedback of the downloading speed, the video request may redirect to the OSN-operated content cloud if it cannot be well served by VSSes. As illustrated in Fig. 5, the cloud service for content (e.g., videos) delivery usually consists of an origin server and a distributed delivery network which includes multiple edge servers distributed in different geographical locations. Initially, a cloud customer should apply an origin server to store its video files, and choose several edge locations to serve its user requests. When the videos are ready to be delivered, they will be first uploaded to the origin server, and then copied to the edge locations. Take Amazon's Cloud as an example, to delivery video content globally, it suggests Amazon S3 as the origin server, and Amazon CloudFront as the distributed delivery network.

To implement SNACS, we need to consider three critical issues, including selecting the right edge locations for each request, reducing the number of misses (which corresponds to good user experience on watching videos), and running at a low cost. The cost to use content cloud service consists of three parts (1) charges for storing objects with original servers (e.g., Amazon S3), (2) charges for data size that transfers between original servers and edge locations (e.g., CloudFront), and (3) charges for serving data from edge locations. The storage is charged by usage time with a per unit time rate; and let P_s denote the unit storage price of storing objects in cloud origin servers. The last two are by traffic volume on a per byte rate; and let P_e be the unit data size price of serving objects from edge locations, and P_c be the unit data size price of copying objects to edge locations. Given a sequence that has a length of time T, the cost during time T can be formulated as

$$Cost = S_z \cdot P_s \cdot T + B_c \cdot P_c + B_e \cdot P_e \tag{1}$$

where S_z is the storage size of the original server; B_c is the size of video objects copied from original servers to edge locations; B_e is the size of videos served from edge locations. Given the time T and a fixed storage size, $S_z \cdot P_s \cdot T$ is fixed. As Amazon's CloudFront has already offered edge locations selection algorithms that are known to be effective [11], we thus need to minimize the number of misses and replacements, where the former is closely related to the user experience as well as improving the cost-efficiency of the cloud assistance, and the latter can further reduce the overall cost, particularly the second part of the cost in Eq. 1.

IV. Optimal Off-line Algorithm

In this section, we propose off-line solutions that assume the user requests are known *a priori* and show that they can yield optimal results, which then motivates our design of an online algorithm in the next section.

A. Scheduling with Minimum Miss Rate

We start from proposing a scheduling algorithm that minimizes the miss rate and proving its optimality. We then extend the algorithm to further minimize the replacement rate in the next subsection. It is worth noting that our problem is different from the classic miss and replacement problem [12], since in our scenario, even a miss happen, we may not always do the replacement as in the classic problem. For example, we assume the request sequence is "$A\ B\ C\ B\ A$". The storage is of size 2 and initially empty. When a miss happens, the optimal solution (denoted as S_{FF} for Farthest-in-Further scheduling [12]) for the classic problem will always take in the missed video to replace the video in the storage whose next request occurs furthest in the future request sequence, which leads to 4 misses and 2 replacements for the aforementioned example. Yet we can easily find the optimal solution for our problem is 3 misses and 0 replacement, if we do not update C into the storage when its request misses. This shows that the solution for the classic problem does not work well in our scenario. To this end, we propose a new algorithm S_{OPT_M} as shown in Algorithm 1 to address this new problem.

Algorithm 1 S_{OPT_M}

```
1: if current request to video $d$ misses then
2:     search the rest request sequence until each video $v_i$
       currently in the storage and $d$ occur at least once;
3:     if $\exists\ v_i$ such that dist[$d$]$<$ dist[$v_i$] then
4:         find the video $v$ in the storage that maximizes dist[$v$];
5:         replace $v$ for video $d$;
6:     else
7:         do no replacement;
8:     end if
9: else
10:    do no replacement;
11: end if
```

Let dist[x] denote the distance from the current position to the position where the first request to video x occurs after the current position. We also define a *reduced scheduling algorithm* if in the algorithm, a replacement can only happen when a request misses (although when a request misses, a replacement may not always happen.). We thus have the following two lemmas:

Lemma 1. *For any giving scheduling algorithm S, there exists a reduced version $\bar{S}$ of S, where $\bar{S}$ is a reduced scheduling algorithm that replaces at most as many videos as the scheduling algorithm S does.*

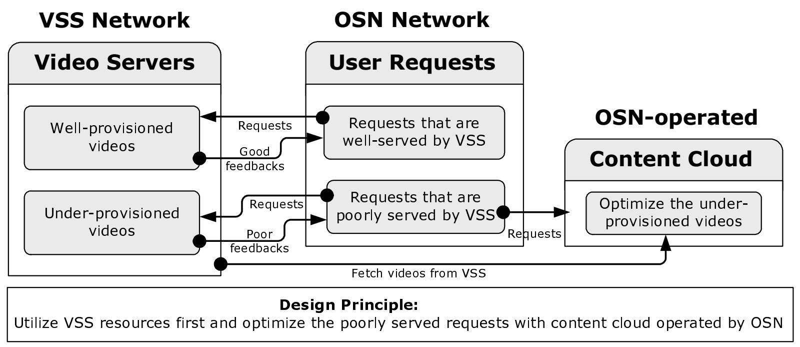

VSS Network
Video Servers
Well-provisioned videos
Under-provisioned videos
OSN Network
User Requests
Requests that are well-served by VSS
Requests that are poorly served by VSS
Requests
Good feedbacks
Requests
Poor feedbacks
OSN-operated
Content Cloud
Optimize the under-provisioned videos
Requests
Fetch videos from VSS
Design Principle:
Utilize VSS resources first and optimize the poorly served requests with content cloud operated by OSN

Fig. 4. Workflow of SNACS

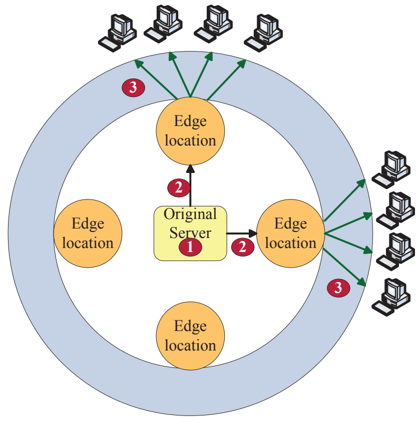


Fig. 5. System model of content cloud

Proof: We prove the lemma by constructing $\bar{S}$ as follows: each time when S replaces a video d that has not been requested into the storage, we can defer the replacement of video d until d is actually requested. Hence, the number of replacements by $\bar{S}$ is at most as many as S. ■

Lemma 2. *Let S be a reduced scheduling algorithm that makes same decisions as S_{OPT_M} through the first j requests in the request sequence, for a number of j. Then there is a reduced scheduling algorithm S' that makes the same decisions as S_{OPT_M} through the first $j+1$ requests, and incurs no more misses than S does.*

To prove that there must exist such a reduced scheduling algorithm S', we should construct S' by trying to get the storage content back to the same state as S as quickly as possible, while not incurring more misses. If the storages of S and S' are the same, we can finish the construction of S' by just making it behave exactly same as S afterwards. Due to the space limitation, we omit the detailed proof and it can be found in our technical report [13]. The optimality of our S_{OPT_M} algorithm can then be shown by the following theorem:

Theorem 1. *S_{OPT_M} incurs no more misses than any other schedule S and hence is optimal in term of achieving the minimum miss rate in our problem.*

Proof: To prove that the scheduling algorithm S_{OPT_M} is optimal, we begin with an optimal schedule S^*, and use Lemma 2 to construct a schedule S_1 that agrees with S_{OPT_M} through the first step. We then continue applying Lemma 2 inductively for $j = 1, 2, 3, \cdots, m$, producing schedules S_j that agree with S_{OPT_M} through the first j requests. Each scheduling algorithm incurs no more misses than the previous one. We then have $S_m = S_{OPT\_M}$, since it agrees with it through the whole sequence. ■

B. Minimize Miss Rate and Replacement Rate

Although S_{OPT_M} minimizes the miss rate, we find it may not always minimize the replacement rate. For example, assume the video request sequence is "*A B C B C A*", and the storage size is two and initially empty. S_{OPT_M} will put video C into the storage at the third request by replacing A, and it will lead to 4 misses and 1 replacement. While a better way could be 4 misses and 0 replacement if there is no replacement when the third request to C misses in the storage. To further reduce the cost for cloud assistance (according to Eq. 1), it is necessary to find a solution which minimizes the replacement rate while guarantees a minimum miss rate.

To this end, a naïve approach is to use the exhaustive search on the scheduling decision tree, where for each request in the request sequence, we need to make decisions such as whether to do replacement and if so, which video in the storage should be replaced out and which video should be taken in. However, even some of the exhaustive search branches can be filtered out by using our S_{OPT_M} as a bound on the miss rate, the solution space can still be very large. To this end, we first introduce two rules that can help improve the optimality of S_{OPT_M} for the replacement rate, which, together with S_{OPT_M}, will also lead to the design of a much more efficient guided search algorithm to be discussed later. The two rules are as follows:

Rule 1 If a miss for requesting video d happens, and for each video v currently cached in the storage, we have $dist[d] \geq dist[v]$, then there is no replacement for d.

Rule 2 If there is a miss and a replacement is required, then only replace the video v in current storage such that $dist[v] \geq dist[v']$ for any video v' in current storage.

We then have the following two lemmas:

Lemma 3. *Given a scheduling algorithm S, if Rule 1 is broken at least once, then there always exists a scheduling algorithm S' that never breaks Rule 1 and incurs no more misses and replaces than S does.*

To prove Lemma 3, we can assume that when a miss for requesting video d happens, S breaks Rule 1 and replaces video f for d. To construct S', we still try to have S' agrees with S in the storage content as quickly as possible. We can then finish the construction of S' by setting $S' = S$ thereafter. Note that, after requesting d misses, S and S' are slightly different in that S has video d and S' has video f. We can then use an approach similar to the proof for Lemma 2 to prove this lemma. Due to the space limitation, we omit this proof as well as the proofs for Lemma 4 and Theorem 2. The detailed proofs for Lemma 3, Lemma 4 and Theorem 2 can be found in [13].

Lemma 4. *Let S be a minimum-miss scheduling algorithm and agrees with Rule 2 through the first j requests. There exists a scheduling algorithm S', which agrees with Rule 2 through the first $j+1$ requests and incurs the same number of misses and no more replacements than S.*

Algorithm 2 S_{OPT_MR}

```
1:  if reach the end of the request sequence then
2:    compare the current schedule with the best schedule
      found till now and keep the better one;
3:  else
4:    if current request to video $d$ misses then
5:      search the rest request sequence until each video $v_i$
        currently in the storage and $d$ occur at least once;
6:      if $\exists\ v_i$ such that dist[$d$]< dist[$v_i$] then
7:        recursively handle next request with $S_{OPT\_MR}$;
8:        // enforce Rule 2
9:        find the video $v$ in storage that maximizes dist[$v$];
10:       replace $v$ for video $d$;
11:       recursively handle next request with $S_{OPT\_MR}$;
12:     else
13:       // enforce Rule 1
14:       do no replacement;
15:       recursively handle next request with $S_{OPT\_MR}$;
16:     end if
17:   else
18:     do no replacement;
19:     recursively handle next request with $S_{OPT\_MR}$;
20:   end if
21: end if
```

Lemma 3 tells that by enforcing Rule 1, in Algorithm S_{OPT_M}, we can not only achieve minimum miss rate, but also incur no unnecessary replacements. Lemma 4 tells that when Rule 2 is enforced in Algorithm S_{OPT_M}, if a replacement is necessary, replacing by Rule 2 can still keep the solution optimal in terms of replacement rate while ensuring the minimum miss rate. Therefore, by incorporating S_{OPT_M} and the two rules, we can design an efficient guided search (denoted by S_{OPT_MR} as outlined in Algorithm 2) that only explores the branches potentially leading to the optimal solution on both miss rate and replacement rate, while intelligently cutting off all the others. We then have the following theorem:

Theorem 2. *S_{OPT_MR} incurs no more replacements than any other schedule S that has minimum miss rate and hence is optimal in term of achieving the minimum replacement rate over all the minimum miss rate solutions.*

V. Online Scheduling

Different from the offline scheduling algorithm for which the optimality on the miss rate and replacement rate is of the first importance, the online scheduling implementation requires that the solution is simple and highly efficient yet achieving reasonably good performance and only based on the information that the system currently has, i.e., not relying on the future information as the offline algorithm does.

For the classic miss and replacement problem, LRU (Least Recently Used) is a well accepted implementation that approximates the optimal offline scheduling algorithm S_{FF}. Thus one straightforward solution is to directly apply LRU to our problem. However, like S_{FF}, LRU also has the same limitation for solving our problem, i.e., it always does replacement when a miss happens. In addition, LRU simply replaces the least recently used item when a miss occurs, and hence fails to consider the specific features of online social video sharing. Yet one major principle that we can still learn from LRU is that it actually uses historical statistics to approximate the future request sequence used in S_{FF}. Therefore, in this section, we will first discuss how we can approximate future user requests in our problem. We will then propose our online scheduling algorithm that can successfully incorporate what we have learned from the optimal offline scheduling algorithm discussed in section IV.

A. OSN-based Future Request Approximation

To incorporate the lessons learned from the offline optimal algorithm, for each video v currently cached in the storage and the missed video d, we need to know $dist[v]$ and $dist[d]$. In other words, we need to know how soon each of these videos will be requested in the future. Thus, we need to predict the popularity of these videos based on the information in the OSN. There are a number of studies [14][7] to address this problem. Yet most of them are based on a relatively large time scale, say, one hour or even one day. To afford a finer time granularity which is essential to our online solution, we develop an efficient approximation solution based on the approach proposed in [14].[2]

The approximation solution works as follows: we first search backwards within previous K video requests and identify those users who recently issue the requests for the videos currently cached in the storage as well as for the missed video. If a user decides to share the video after watching the requested video, we then look at its neighbors in the OSN and count those who have not requested this video. We also maintain the popularity of this user. In particular, for each video previously shared by this user, we count the number of neighbors who actually viewed the video and divide it by the total number of neighbors. The popularity of this user is thus the average value on all the videos previously shared by

[2]Note that our focus here is on how to use a highly efficient prediction solution to effectively approximate the future user requests for our online implementation. While proposing an algorithm with better predication results for OSNs is very important and can be useful to our solution, it is generally orthogonal to the problem studied here.

this user. For each video, we calculate the sum for each user who requests this video by adding the number of the potential viewers of this user weighted by the popularity of this user. We then use the reciprocal value of this sum to approximate how soon this video will be requested in the future. We call this reciprocal value as the approximation value. Note that the rationale of this approximation is that, if a video is very popular in the OSN, i.e., there are a large number of OSN users who tend to request it (which is different from a large number of users who have viewed and shared it), then it is likely that this video will be requested soon in the near future.

B. Incorporate Offline Optimal Solution

Algorithm 3 S_{OSN_MR}

```
1:  if current request to video $d$ is not in the storage then
2:      search the request sequence backwards for $K$ requests;
3:      calculate the approximation value for each video $v_i$
        currently in the storage and for the missed video $d$;
4:      if $\exists\ v_i$ such that approx[$d$]< approx[$v_i$] then
5:          find video $v$ in the storage that maximizes approx[$v$];
6:          replace $v$ for video $d$;
7:      else
8:          do no replacement;
9:      end if
10: else
11:     do no replacement;
12: end if
```

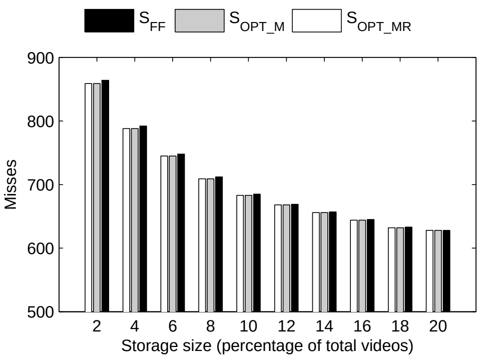
In this subsection, we will combine what we have learned from the optimal offline solution and this approximation solution into our online scheduling algorithm. Recall that, from the design of optimal offline scheduling algorithm, we have learned the following lessons: (a)When a video request misses, a replacement may not always happen. Especially when the situation in Rule 1 happens, we should never do the replacement; (b)When a replacement must be done, we should always do the replacement according to Rule 2. With the approximation solution, we can now interpret Rule 1 as follows: if the approximation value of the missed video is larger than that of any video currently in the storage, there is no replacement for the missed video. Similarly, Rule 2 can be interpreted as follows: if there is a miss and a replacement is required, we should replace the video with the largest approximation value in current storage.

Based on these interpretations, we propose our online scheduling algorithm for OSN video sharing, called S_{OSN_MR}, as shown in Algorithm 3, where approx[x] denotes the approximation value of the video x. Compared to the aforementioned straightforward solution using LRU, our solution still achieves similar simplicity and efficiency. More importantly, it fully exploits the specific features of online social video sharing and successfully incorporates what we have learned from the optimal offline solution, improving both user experiences and cost-efficiency through the assistance of cloud.

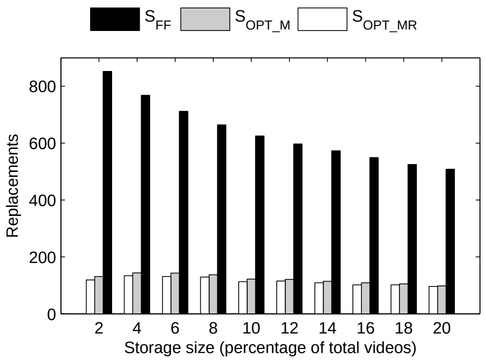
VI. Performance Evaluation

We have conducted extensive trace-based simulations to evaluate our solutions for SNACS. To this end, we collected the traces of video requests in the RenRen OSN over three months. We find the video requests show a strong daily [10], [15] pattern and thus choose a trace in a typical one-day period (April 19th, 2011) for our simulations, which contains $19,473,512$ requests and involves $278,922$ unique videos.

A. Comparison of Offline Algorithms



(a) misses

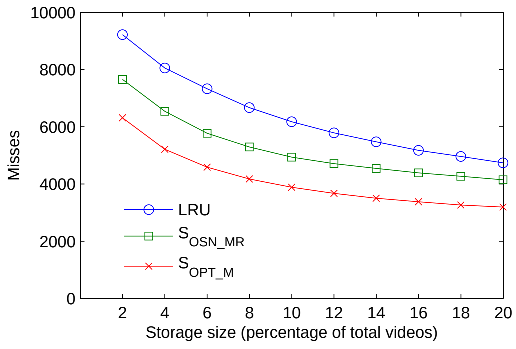


(b) replacements

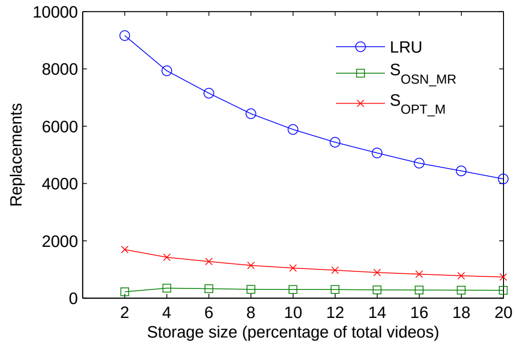
Fig. 6. Comparison between offline algorithms

To evaluate our offline algorithms S_{OPT_M} and S_{OPT_MR}, we also implement the optimal offline solution S_{FF} for the classic miss and replacement problem. Fig. 6 shows the results on the miss number and the replacement number[3]. It is clear that both S_{OPT_M} and S_{OPT_MR} outperform S_{FF} for solving our problem. In particular, although S_{FF} is the optimal solution for the classic miss problem, by allowing no replacement, our S_{OPT_M} and S_{OPT_MR} can achieve less number of misses (Fig. 6(a)), which becomes even observable as the storage size decreases. This result further confirms with our theoretical analysis on optimality.

[3]In worst case, each offline algorithm may traverse the whole trace to find if a video is requested in the future and thus causes enormous time to finish. To this end, we shrink the trace to a subset by randomly sampling 10% of the requests in the April 19th trace.



(a) misses



(b) replacements

Fig. 7. Online algorithms vs. The optimal algorithm

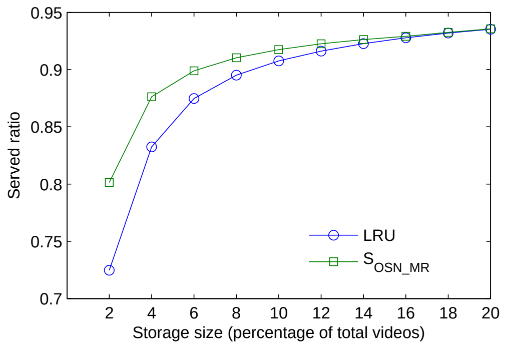
In term of replacement number (Fig. 6(b)), our S_{OPT_M} and S_{OPT_MR} perform even much better than S_{FF}, by successfully reducing the replacement number for 75% to 85%. One interesting observation is that for the replacement number, S_{OPT_M} performs very close to S_{OPT_MR}. This further demonstrates the effectiveness of our Rule 1 and Rule 2 derived in Section IV-B, since these two rules are also reflected in S_{OPT_M} implicitly.

B. *Online vs. Offline*

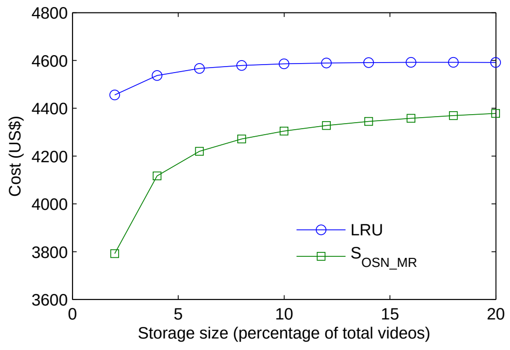
We next compare the performance of our online implementation S_{OSN_MR} with the offline solution. As S_{OPT_M} performs close to the optimal offline algorithm S_{OPT_MR} and is more efficient in running time, we thus use S_{OPT_M} for this comparison. In addition, we implement LRU to represent the classic online replacement algorithms. The results on miss number and replacement number are shown in Fig. 7. It is clear that our S_{OSN_MR} outperforms LRU and stays close to S_{OPT_M}, especially in term of the replacement number. This is because our solution can well approximate the video popularity and the user requests in the future with the information from OSNs, and LRU always replaces the least recently requested video when a miss occurs.

Another observation comes from comparing the miss number and replacement number together. Unlike LRU, which performs bad on both the miss number and replacement number, S_{OSN_MR} incurs fewer replacements than S_{OPT_M} at a cost of slight increase in the miss number. Therefore, it is interesting to further investigate such tradeoff between miss rate and replacement rate as a future work.

C. *Impacts on Served Ratio and Cost of OSN*



(a) performance



(b) cost

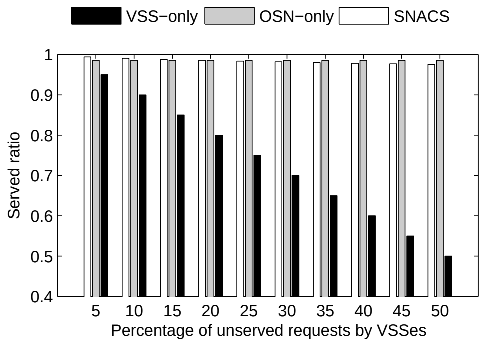
Fig. 8. Comparison between online algorithms

Besides the miss number and replacement number, we also investigate the impacts on the cost by using the cloud assistance. To this end, we adopt a typical setting as used in [7]. We assume each video has same file size denoted as F_z. Then the storage size can be represented as the number (denoted as N_s) of stored videos in the cloud. The size of the object that is copied to edge locations can be represented as the product of the number of edge locations (N_e) and the number (N_r) of video replacements in the cloud. The size of videos served from edge locations can be represented as the number (N_h) of hit requests by the cloud. The cost in Eq. 1 can be rewritten as Eq. 2.

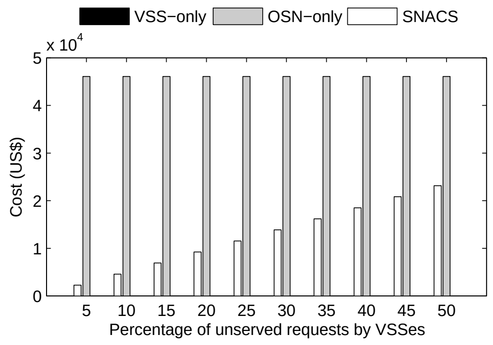
$$Cost = N_s \cdot F_z \cdot P_s \cdot T + N_r \cdot F_z \cdot N_e \cdot P_c + N_h \cdot F_z \cdot P_e \quad (2)$$

According to Amazon pricing model, we set P_e=\$0.12 per GB, P_s=\$0.08 per GB per month, P_c=\$0.02 per GB and we set K=5, F_z=20MB. We also conduct simulations under other parameter settings and the results generally follow a similar trend.

Fig. 8 shows the results on the served ratio (the fraction of the video requests from the OSN that are well served by



(a) performance



(b) cost

Fig. 9. Comparison between three architectures

VSS or OSN servers over the total requests from the OSN) and the money spent by the OSN. Again our S_{OSN_MR} still outperforms LRU especially when the storage size is small. Moreover, compared to LRU, our S_{OSN_MR} can also greatly reduce the total costs by 5% to 15%.

Besides comparing under the SNACS framework, we also conduct simulations to compare our general SNACS solution with the *VSS-only* and *OSN-only* solutions. The *VSS-only* solution, which is the current development architecture in real life, assumes that all the video requests from OSNs are served by VSSes. The *OSN-only* solution, following the idea in [7], assumes all the video requests from OSNs are served by the video servers operated by the OSN. We vary the percentage (5% to 50%) of daily requests that are unserved by VSSes due to under-provisioning, and examine how the served ratio and cost change. Fig. 9 shows the results. It shows that while both *OSN-only* and SNACS can significantly improve the user experience, SNACS achieves the similar performance with much less cost, thus providing a more viable option for OSNs to improve the user experience more cost-efficiently in practice.

VII. Conclusion

In this paper, we proposed the SNACS framework for the OSN to cost-effectively enhance its video viewing experience by leveraging content cloud service. Given the strong dynamics of the video access patterns in the OSN, we were particularly interested in the content management and update strategies in the SNACS' implementation. We showed that conventional cache replacement strategies can be quite inefficient in SNACS. We then developed an optimal offline replacement algorithm that generates minimum misses in this new context. We further offered guidelines and extended the algorithm to minimize replacements among the solutions with minimum misses. The optimal offline solutions not only provide a benchmark for comparison but also motivate the design of an online replacement algorithm, which makes effective use of the video sharing patterns in the OSN. The superiority of our design was confirmed by the trace-driven simulations.

VIII. Acknowledgement

Jiangchuan Liu's research is supported by an NSERC Discovery Grant and an NSERC Strategic Project Grant. Feng Wang's research is supported by a Start-up Grant from the University of Mississippi, MS. Ke Xu's research is supported in part by NSFC Project (61170292, 61472212).

Reference

[1] T. Broxton, Y. Interian, J. Vaver, and M. Wattenhofer, "Catching a viral video," in *Processings of ICDM*, Sydney, Australia, 2010.
[2] "http://www.youtube.com/t/press_statistics," [Online; accessed on January 10, 2011 and June 10, 2012].
[3] comScore, "http://ir.comscore.com/releasedetail.cfm?releaseid=860971," [Online; accessed on 28-July-2014].
[4] H. Li, J. Liu, K. Xu, and S. Wen, "Understanding Video Propagation in Online Social Networks," in *Processings of IWQoS*, Coimbra, Portugal, 2012.
[5] Z. Wang, L. Sun, X. Chen, W. Zhu, J. Liu, M. Chen, and S. Yang, "Propagation-Based Social-Aware Replication for Social Video Contents," in *Processings of ACM Multimedia*, Nara, Japan, 2012.
[6] Z. Wang, L. Sun, C. Wu, and S. Yang, "Enhancing internet-scale video service deployment using microblog-based prediction," *IEEE Transactions on Parallel and Distributed Systems*, 2014.
[7] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. Lau, "Scaling social media applications into geo-distributed clouds," in *Processings of IEEE INFOCOM*, Orlando, FL, USA, 2012.
[8] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft, "Track Globally, Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades," in *Processings of WWW*, Hyderabad, India, 2011.
[9] L. Zhang, F. Wang, and J. Liu, "Understand instant video clip sharing on mobile platforms: Twitter's vine as a case study," in *Processings of NOSSDAV*, Singapore, 2014.
[10] H. Li, H. Wang, J. Liu, and K. Xu, "Video requests from online social networks: Characterization, analysis and generation," in *Processings of INFOCOM mini-conference*, Turin, Italy, 2013.
[11] D. Rayburn, "Comparing CDN Performance: Amazon CloudFront's Last Mile Testing Results," Frost & Sullivan, Tech. Rep., 2012.
[12] J. Kleinberg and Éva Tardos, *Algorithm Design*. Addison-Wesley, 2005.
[13] "SNACS: Social Network-Aware Cloud Assistance for Online Propagated Video Sharing," Tech. Rep., 2014, [Online; https://s3-us-west-2.amazonaws.com/technical-report/SNACS.pdf].
[14] H. Li, X. Ma, F. Wang, J. Liu, and K. Xu, "On Popularity Prediction of Videos Shared in Online Social Netowrks," in *Processings of ACM CIKM*, San Francisco, CA, USA, 2013.
[15] H. Li, H. Wang, J. Liu, and K. Xu, "Video Sharing in Online Social Network: Measurement and Analysis," in *Processings of NOSSDAV*, Toronto, ON, Canada, 2012.